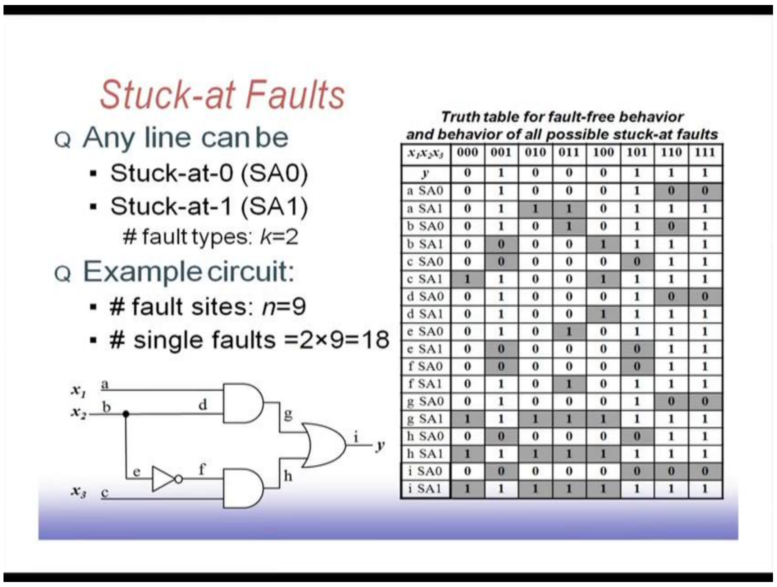
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 03
Introduction (Contd.)

Next we will discuss on stuck at faults. So, when a line is stuck at, but a particular value when we are talking about digital circuits.

(Refer Slide Time: 00:24)



Stuck-at Faults

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)
  - # fault types: *k*=2
- Example circuit:
  - # fault sites: *n*=9
  - # single faults =2×9=18

Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	1
i SA0	0	0	0	0	0	0	0	0
i SA1	1	1	1	1	1	1	1	1

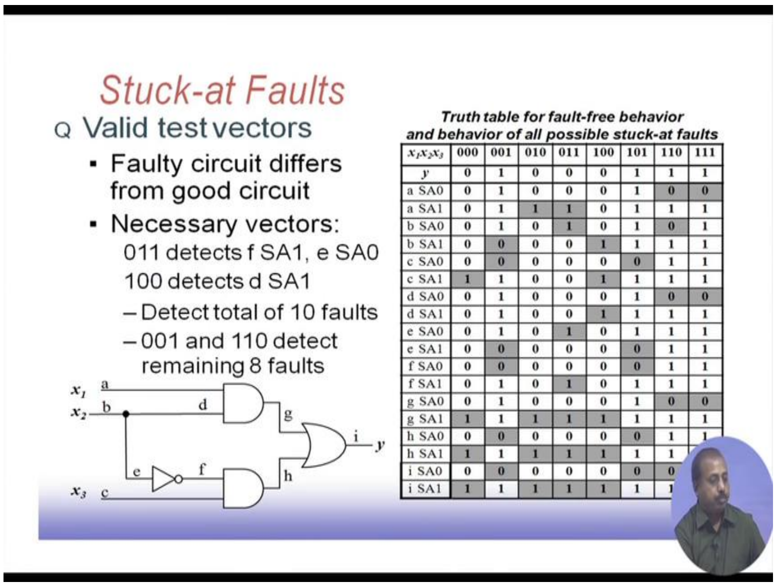
So, naturally the stuck at value can be stuck at 0 or stuck at 1. So, it is taken as the line is permanently 0 or permanently 1. So, this very manifestation of some defect, like a line getting shorted with the supply line or supply line. So, that we it is a stuck at 1 similarly a line getting shorted with the ground line. So, that may it be deflected as a stuck at 0. So, the physical defects are modeled as these faults stuck at 0 faults and stuck at 1 fault. So, in under the stuck at fault model number of fault types that is k equal to 2; so this is an example circuit where you can see that we have got if a 2 and gates 1 or gate and 1 inverter.

So it is realizing the function a b or b bar c. Now in this case so they are all together 9 lines. So, the fault sites equal to n equal to 9, so 1 2 3 4 5 6 7 8 and 9. So, there are 9 points at which their fault can occur. So, this all this a b c d e f g h i. So, they are identifying them now. So, each of these lines their it can be permanently stuck at 1 or permanently stuck at 0, as a result there can be 2 faults associated with every line. So, all together if I consider only single faults. So, there are 2 into 9 that is total 18 faults are possible. Now if you just consider the truth table for this for occurrence of these faults. So, this x 1 x 2 x 3, so this site we have listed all the inputs all possible input patterns that can come.

So, this next line y next row y, so it identifies the fault free responsive, so if there is no fault in the circuit. So, this is the type of response that we give. Now if a is stuck at 0, if a is stuck at 0 then this particular pattern 0 0 0, the response is same as the originally a good circuit; because a is stuck at 0 and in this case I am applying 0 to x 1 that is 0 to a. So, naturally there cannot be any distinction. So, all these cases were 0 0 0 to 1 0 1 the response remains same as the good circuit; however, for these 2 patterns 110 and 1 1 1 the pattern the result is different. So, value of y is different in this case good circuit is 1 and bad circuit is 0 similarly here.

So we see that this truth table; from these truth table we can identify all these boxes where it has been color the gray. So, though who are they are the response is differing from the good circuit response. So, that way that fault can be detected by the corresponding pattern.

(Refer Slide Time: 03:16)



Stuck-at Faults

- Valid test vectors
  - Faulty circuit differs from good circuit
  - Necessary vectors:
    - 011 detects f SA1, e SA0
    - 100 detects d SA1
    - – Detect total of 10 faults
    - – 001 and 110 detect remaining 8 faults

Truth table for fault-free behavior and behavior of all possible stuck-at faults

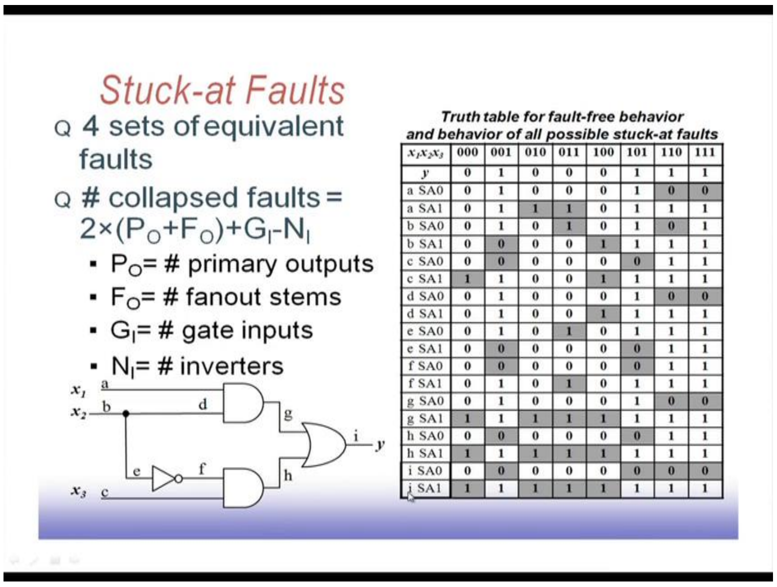
$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	[illegible]
i SA0	0	0	0	0	0	0	0	[illegible]
i SA1	1	1	1	1	1	1	1	[illegible]

Now, faulty circuit differs from good circuit. So, if it is the test vectors are valid test vectors like this 0 1 1 is a valid test vector, when we are talking about this a stuck at 1 fault. So, fault free response is 0, fault free response is 1. Now some vectors they are termed as necessary vectors. So, necessary vectors are those vectors that are masked for detecting some fault.

For example this f stuck at 1. So, this f stuck at 1 fault. So, this is detected by this particular pattern 0 1 1, and if you look into these row carefully we will see that none of the other test patterns can detect this particular fault f stuck at 1. So, that way f stuck at 1 becomes the necessary vectors. So, any test vectors set that does not include this particular pattern 0 1 1. So, for that test for the test vectors it will not be able to detect this particular fault f stuck at 1.

So, that way it is a necessary vectors; similarly e stuck at 0. So, that is also detected by this particular vector. So, e stuck at 0. So, this one, here also it is detected by 0 1 and 1. So, that way 0 1 1 becomes the necessary vector. Similarly 1 0 0 detects d stuck at 1. So, all together, so can detect total of 10 faults, and this 0 0 1 and 1 1 0 detect the remaining 8 faults. So, this is the in this case we are lucky because applying this test patterns say all 0 to all 1 we could detect all possible faults in the circuit, this has happened because a circuit does not have any redundancy. If the circuit have redundancy then of course, this will not be the case, for there will be certain faults which will not be detected whatever test pattern we apply to the circuit.

(Refer Slide Time: 05:17)



Stuck-at Faults

- 4 sets of equivalent faults
- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= # primary outputs
  - F_O= # fanout stems
  - G_I= # gate inputs
  - N_I= # inverters

Truth table for fault-free behavior and behavior of all possible stuck-at faults

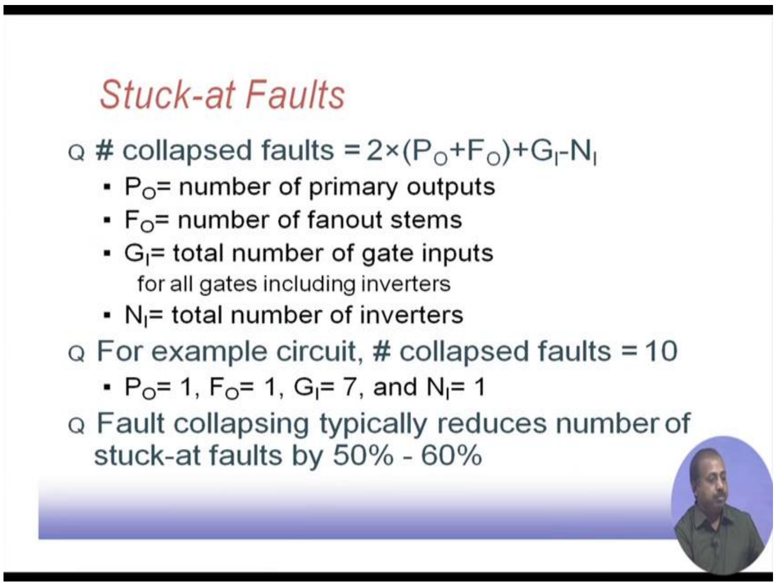
$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	**0**	**0**
a SA1	0	1	**1**	**1**	0	1	1	1
b SA0	0	1	0	**1**	0	1	**0**	1
b SA1	0	**0**	0	0	**1**	1	1	1
c SA0	0	**0**	0	0	0	**0**	1	1
c SA1	**1**	1	0	0	**1**	1	1	1
d SA0	0	1	0	0	0	1	**0**	**0**
d SA1	0	1	0	0	**1**	1	1	1
e SA0	0	1	0	**1**	0	1	1	1
e SA1	0	**0**	0	0	0	**0**	1	1
f SA0	0	**0**	0	0	0	**0**	1	1
f SA1	0	1	0	**1**	0	1	1	1
g SA0	0	1	0	0	0	1	**0**	**0**
g SA1	**1**	1	**1**	**1**	**1**	1	1	1
h SA0	0	**0**	0	0	0	**0**	1	1
h SA1	**1**	1	**1**	**1**	**1**	1	1	1
i SA0	0	**0**	0	0	0	**0**	**0**	**0**
i SA1	**1**	1	**1**	**1**	**1**	1	1	1

But in this case this is not redundant as a result we could detect all the faults. Now another thing is the set of equivalent faults; now there are certain faults like say this a is stuck at 0 and d is stuck at 0. So, if you look into these 2 faults to see that their responses are exactly same for all the test vectors like it is 0 1 0 0 0 1 0 0, here also it is 0 1 0 0 0 1 0 0. So, that response are same. So, with in none of these test pattern can distinguish whether a fault that has occurred whether it is a stuck at 0 or d stuck at 0. So, these 2 faults cannot be distinguished. So, that makes an equivalent class of faults.

So, if we look still there more such faults like say a stuck at 0 then these d stuck at 0, and g stuck at 0 they form 1 equivalence class. Similarly we have. So, these for g stuck at 1, h stuck at 1, and this one this i stuck at 1. So, they form another equivalency class because for all the applied test patterns the responses are same. So, we cannot still distinctively which fault has exactly occurred. So, we cannot distinguish between all those faults. So, these faults are equivalent faults. So, there is no point trying to differentiate between equivalent faults. So, we from this equivalent state we can consider only 1 fault its generation and knowing fully well that the remaining faults are also detected by this pattern, though we cannot say very distinctively which one of them have occurred.

Now you can come up with a formula that says a number of such collapsed faults. So, collapsed faults means when you have removed all these equivalent from all equivalence classes I have taken only one fault, so in that case the number of collapsed faults is given by 2 into primary input plus fan out stems, plus number of gate inputs minus number of inverters. So, this formula it can give us the number of collapsed faults in a circuit. So, in the previous example that we had; so number of collapsed faults equal to 10, primary input is 1. So, if there is only 1 primary output is 1 the y only.

(Refer Slide Time: 07:26)



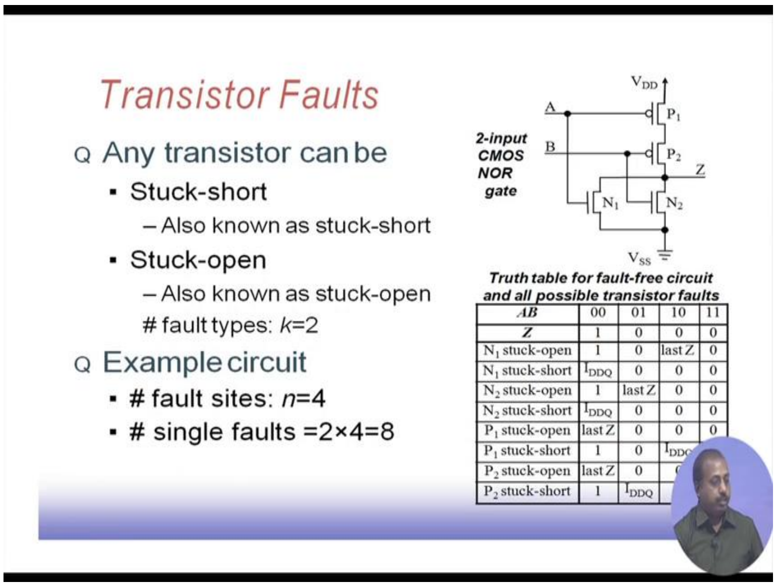
Stuck-at Faults

- # collapsed faults = $2\times(P_O+F_O)+G_I-N_I$
  - P_O= number of primary outputs
  - F_O= number of fanout stems
  - G_I= total number of gate inputs for all gates including inverters
  - N_I= total number of inverters
- For example circuit, # collapsed faults = 10
  - P_O= 1, F_O= 1, G_I= 7, and N_I= 1
- Fault collapsing typically reduces number of stuck-at faults by 50% - 60%

Similarly for fanout stem is one, because there is only 1 point where there is a fanout. So, this point there is a fanout; then we have G i equal to 7, G i is the number of gate inputs. So, 1 2 3 4 5 6 and this inverter 1 7; so there are 7 input that are be there. So, and N i number of inverters is equal to 1. So, if you put into this formula the turns out to be 10. So, that way this formula can give us number of collapsed faults for any circuit under stuck at fault model and faults collapsing is very important because it can be seen that it can reduce the number of faults by 50 to 60 percent. So, that is a huge savings.

So, if we have a big circuit and it has got a large number of lines and gates in it, large number of fanout in it, so as a result the number of fault will be very high. So, these if we can collapse if we consider the collapsed fault in the faults come down to 50 percent of the original one. So as a result it can reduce this fault significantly number of false significantly.

(Refer Slide Time: 08:46)



V_DD
A
B
P₁
P₂
Z
N₁
N₂
V_SS

Transistor Faults

- Any transistor can be
  - Stuck-short
    - Also known as stuck-short
  - Stuck-open
    - Also known as stuck-open
    - # fault types: *k*=2
- Example circuit
  - # fault sites: *n*=4
  - # single faults =2×4=8

2-input CMOS NOR gate

Truth table for fault-free circuit and all possible transistor faults

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0
N_1 stuck-short	I_{DDQ}	0	0	0
N_2 stuck-open	1	last Z	0	0
N_2 stuck-short	I_{DDQ}	0	0	0
P_1 stuck-open	last Z	0	0	0
P_1 stuck-short	1	0	I_{DD}	[illegible]
P_2 stuck-open	last Z	0	[illegible]	[illegible]
P_2 stuck-short	1	I_{DDQ}	[illegible]	[illegible]

Next we will look into the transistor fault. So, at the transistor levels so if you consider. So, a transistor can be stuck short or a transistor can be stuck open. So, stuck short means there is a short between this drain and supplier, drain and source lines; and stuck open means there is a transistor is open. So, this connection is not there to the gates, so as a result whatever input we apply here that does not reach this gate and until and unless this gates enough for our potential no channel is established between the drain and source as a result the transistor is taken as open. So, there are 2 types of faults that we will consider for transistor; one is called stuck short for another is called stuck open faults.

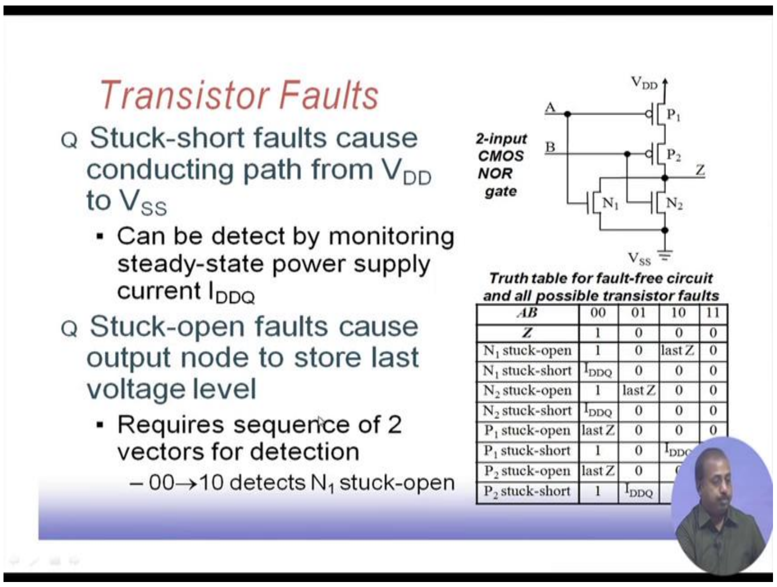
So here also k is equal to 2, and for this example circuit's number of there are 4 transistors. So, 4 transistors the so there are 4 faulty sites n equal to 4, and number of single faults. So, each of them can be stuck open and stuck for short. So, total number of faults is 2 into 4 that is 8. Now if you apply some test pattern like say A B are the 2 inputs to this nor gate 2 input nor gate. So, this 0 0 0 1 1 0 1 1 also these are the combinations. So, in under fault free condition so this is the output 1 0 0 0; now if n 1 is stuck open this transistor is open. So, whenever A is equal to 1. So, whenever A is equal to 1 so this case. So, this transistor will not be able to conduct. So, if this transistor is unable to conduct; and N 2 if N 2 is 0. So, if N 2 is 0; that means, there is no path through this N 2 as a result this point Z.

So, it will continue to remember its previous value. So, if there is no discharging path for this Z, so as a result Z will continue to remember its previous value that is why N stuck at open. So, this is this last Z value will be remembered. Similarly if N 1 is short then whenever we apply say 0 0. So, this point will get some this point Z will get charged to one; however, since this transistor is short. So, there will be a current flow through this.

So, if we detect this current flow through this I DDQ test. So, we will come to that later. So, we can measure the current flowing to the device and ideally if the transistors are all then putting A and B both equal to 0 these transistors will be off. So, only a very small amount of leakage current should flow from through these transistors. So, that if this transistor is stuck short then naturally the amount of current flowing will be larger.

So this is this I DDQ test can measure the current that is flowing through the device and accordingly it can detect if there is a fault in the transistors. So, this way all these transistor faults can be characterized and we can in for like which of them have occurred by looking into the corresponding patterns and the responses.

(Refer Slide Time: 11:53)



Transistor Faults

- Stuck-short faults cause conducting path from V_{DD} to V_{SS}
  - Can be detect by monitoring steady-state power supply current I_{DDQ}
- Stuck-open faults cause output node to store last voltage level
  - Requires sequence of 2 vectors for detection
    - 00→10 detects N_1 stuck-open

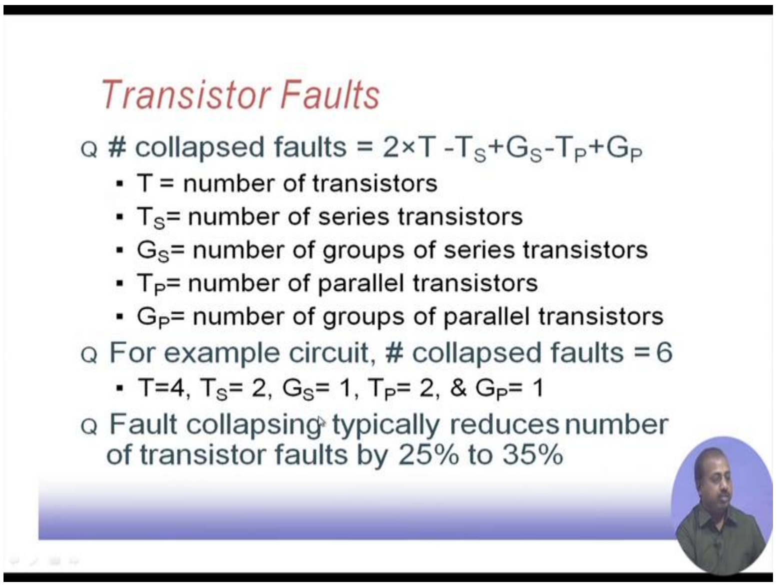
Truth table for fault-free circuit and all possible transistor faults

AB	00	01	10	11
Z	1	0	0	0
N_1 stuck-open	1	0	last Z	0
N_1 stuck-short	I_{DDQ}	0	0	0
N_2 stuck-open	1	last Z	0	0
N_2 stuck-short	I_{DDQ}	0	0	0
P_1 stuck-open	last Z	0	0	0
P_1 stuck-short	1	0	I_{DDQ}	[illegible]
P_2 stuck-open	last Z	0	[illegible]	[illegible]
P_2 stuck-short	1	I_{DDQ}	[illegible]	[illegible]

So, this stuck short fault cause conducting path from V DD to V SS as we have already said. So, can be detected by monitoring steady state power supply current I DDQ. So, this is the I DDQ is the current flowing to the device. So, if there is a stuck short then we can do this thing stuck open faults it will cause output node to store last voltage level, that we have seen this transistors is open and this is off then this will remember the last value.

Now if you want to detect this stuck open faults, then naturally we will require 2 vectors; because first of all these Z it should attain some voltage level. So, that has to be done. So, therefore, that pattern for that purpose we have to apply 0 0here, so that these 2 transistors are on at this Z the gets charged to some value, and after that we apply 1 0, so that we expect that the Z will gets discharged by a N 1. So, if N 1 is stuck open naturally the discharging will not happen. So, it will remember the previous value. So, the stuck short faults can be detected by this I DDQ test, and the stuck open faults for that you need to apply 2 successive test patterns for one for exciting the fault, and another for detecting the fault.

(Refer Slide Time: 13:10)



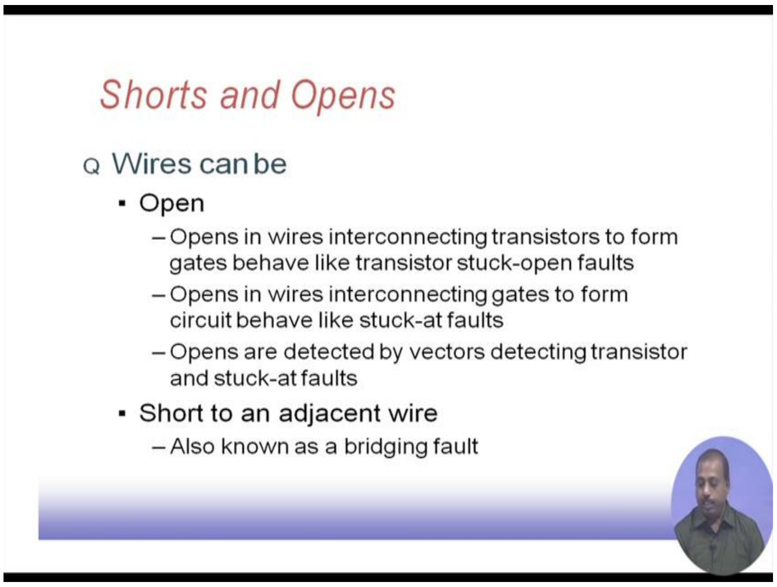
Transistor Faults

- # collapsed faults = $2\times T - T_S + G_S - T_P + G_P$
  - T = number of transistors
  - T_S= number of series transistors
  - G_S= number of groups of series transistors
  - T_P= number of parallel transistors
  - G_P= number of groups of parallel transistors
- For example circuit, # collapsed faults = 6
  - T=4, T_S= 2, G_S= 1, T_P= 2, & G_P= 1
- Fault collapsing typically reduces number of transistor faults by 25% to 35%

So in case of transistors, if the you can come up with a formula; number of collapsed faults is twice number of transistors, minus number of trans series transistors class, number of groups of series transistor, number of parallel transistors and number of groups parallel transistor. So, we can put into this formula and see what is the number of collapsed faults. So for this particular circuit that we are considering; so we have got 4 transistors, number of series transistors is 2, G s is the number of groups of series transistors one group of series transistors.

So, that is 1, T p is the number of parallel transistors so that is again 2, and G p is the number of groups of parallel transistors that is equal to 1. So, we put into this formula it turns out to be 6. So, there are 6 equivalence for non equivalence faults, that we need to consider again. Fault collapsing can reduce number of transistor faults by 25 to 35 percent that is a huge reduction in the number of faults, considering the fact that a circuit considering a transistor level. So, transistor level number of transistor is much much higher compared to say the corresponding gate count when you are considering the stuck at fault model.

(Refer Slide Time: 14:35)



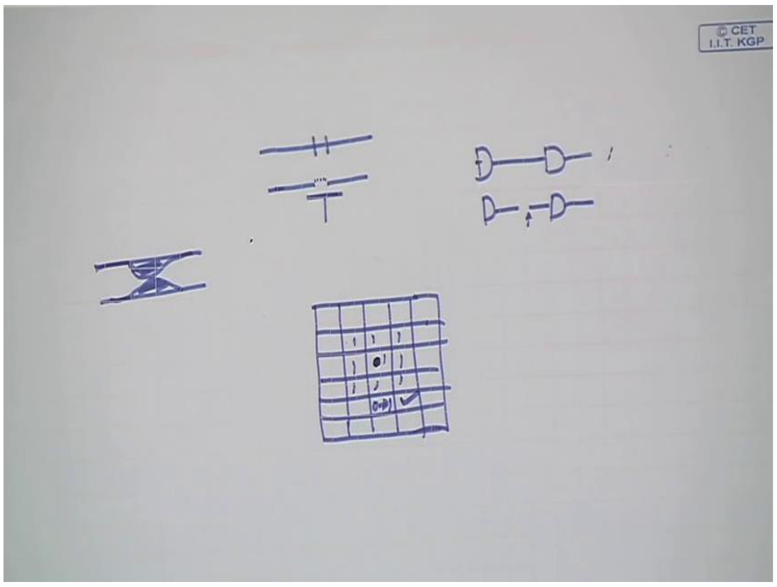
Shorts and Opens

- Wires can be
  - Open
    - Opens in wires interconnecting transistors to form gates behave like transistor stuck-open faults
    - Opens in wires interconnecting gates to form circuit behave like stuck-at faults
    - Opens are detected by vectors detecting transistor and stuck-at faults
  - Short to an adjacent wire
    - Also known as a bridging fault

So, naturally 25 35 percent saving is saving large number of transistor chips. For the wires; so wires may be opened or wires may be shorted. So, open wire means it interconnecting opens in wires interconnecting transistors, to form gates behave likes transistors stuck open fault.

So if we open if a line is broken. So, it will behave like transistor as if that it has got one side of it. So, this is a wire and there is a break at this point.

(Refer Slide Time: 15:13)



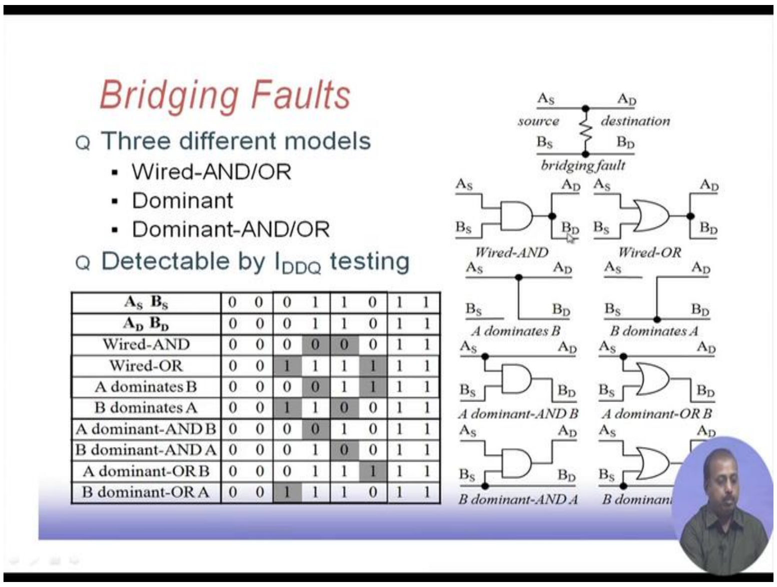
So, this side I have got one part, this side I have got one part and then this is an as if this it can be modeled as a transistor for because if there is a gate here, if I apply some gate voltage then this channel we will get established. So, as a result this will behave as a transistor in that case. So, we can say that this opens in wires interconnecting transistors to form gates behave like transistors stuck open fault.

Similarly opens in wires interconnecting gates to form circuit, behave like stuck at fault. So, if I have got 1 line say this is a gate output connecting another gate input. So, if this line is open then it is like this. So, this is depending upon the logic depending upon the technology family that we are considering. So, this line is open. So, this line is open so this can be treated as logic albro or logic high depending upon the technology we have. So, this will be a as if the line is permanently stuck at some value for the second gate.

So when we did interconnecting gates it is behaving like a stuck at fault; when it is interconnecting transistors. So, it behaves likes stuck open transistors. So, naturally we can apply, we can have test vectors for detecting transistors and stuck at faults for checking this wire open faults. Similarly short to an adjacent wire, so 2 wires may be shorted; so I have got one wire running like this, another wire line running like this while physically for manufacturing this thing maybe the thickness of these wires are not maintained could not be maintained very a accurately. So, as a result may be this part becomes something like this and this is thick and similarly this part becomes thick like this.

So, as a result there is a shorting between these 2 lines. So, that way it gives raise to something called a Bridging faults.

(Refer Slide Time: 17:16)



Bridging Faults

- Three different models
  - Wired-AND/OR
  - Dominant
  - Dominant-AND/OR
- Detectable by I_{DDQ} testing

A_S B_S	0	0	0	1	1	0	1	1
A_D B_D	0	0	0	1	1	0	1	1
Wired-AND	0	0	0	0	0	0	1	1
Wired-OR	0	0	1	1	1	1	1	1
A dominates B	0	0	0	0	1	1	1	1
B dominates A	0	0	1	1	0	0	1	1
A dominant-AND B	0	0	0	0	1	0	1	1
B dominant-AND A	0	0	0	1	0	0	1	1
A dominant-OR B	0	0	0	1	1	1	1	1
B dominant-OR A	0	0	1	1	1	0	1	1

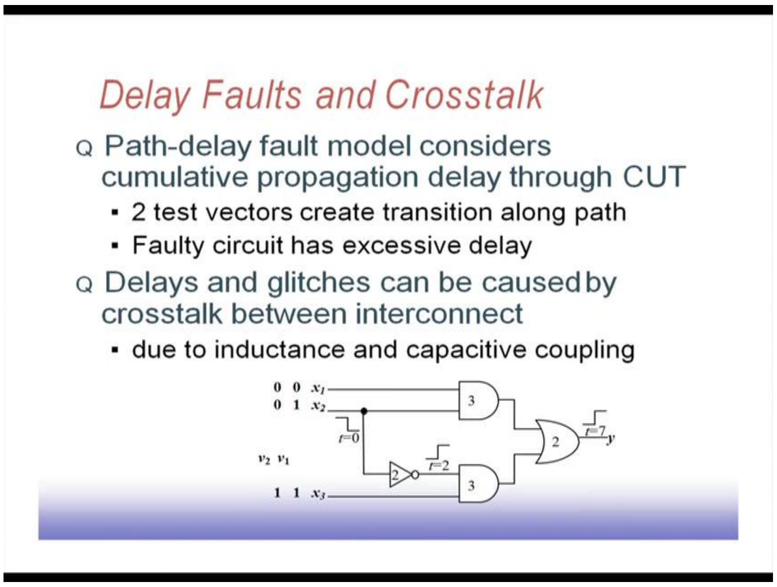
Now, what happens under these bridging faults? So, like here I have got these A and B as 2 lines. So, there is a connection established between them through the creation of the bridge. Now depending upon the technology that we are using, so there can this is the manifestation may be of different types like we can have this weird and or type of connection, then we dominant connection and dominant and or connection like say this one this is wired and connection. So, A s and B s as if these 2 values happen ended and the result is available at A D and B D. So, if A s and B s both are 1 then A D and B D are both are 1, if 1 one of them is 0 then they both the destination values they become 0 this is the wired and connection.

Similarly there maybe situation where you get and wired or connection; so wired or connection will be as if the source and 2 source values will be ored and the result will be available at the both the destinations. So, both the destination gets the same value. So, that is the wired and or type of fault, then you can have a dominant fault; where is say that one line dominates over the other over. So, this a line dominates over b line. So, that is as if it can be as if there is a connection established with this A line and this B line is broken at this point. So, A dominants B similarly we have B dominates A. So, like this. So, then we have got a dominant and or faults. So, A dominant and B, so A is there are and B is there and this ANDing of them if they are it is becoming the B is destination, A continuous with the previous value.

So unlike this wired and connection where it was affecting both the lines, where it the shorting effect only the B line not the A line; similarly this A dominant or B, so this connection. So, this will B line will get dominated by A and this A continues to its previous value, but this is dominated by this or function and the other way B dominant A and B dominant and A and B dominant or A. So, these are possible. So, accordingly we can get this truth table and in this case also if we want to detect this type of faults then we have to take help of this I DDQ testing, which measures the current flowing through the (Refer Time: 20:02).

So, these are the different type of fault that we can have, and you see that this wired and connection. So, in this case this both of them becomes 0, where as this destinations are supposed to be 1 that way it can be detected.

(Refer Slide Time: 20:21)



Delay Faults and Crosstalk

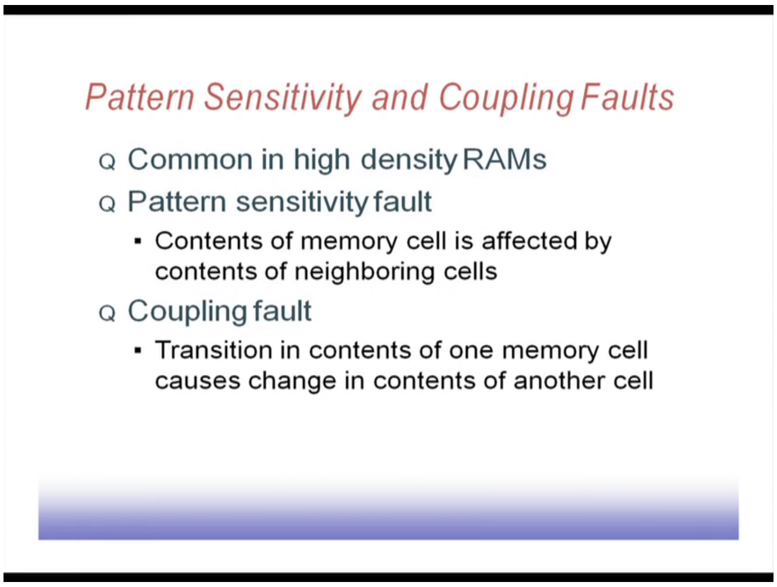
- Path-delay fault model considers cumulative propagation delay through CUT
  - 2 test vectors create transition along path
  - Faulty circuit has excessive delay
- Delays and glitches can be caused by crosstalk between interconnect
  - due to inductance and capacitive coupling

Another class of faults which are known as delay defaults, and also there is something called a crosstalk. So, delay faults means a circuit and as per as its function is concerned it functions properly, but the timing is a problem; like it does not produce the output within the desired time. So, it has got a problem because if the output of a certain stage of logic does not come up with in the desired type, so when that output is fed as input to the next level, so it may be that that next level will be considering the value before it has settled. So, as a result we have got a problem in the setting of values. So, that type of faults so they are delay fault that a occurring. So, delay fault can be modeled using cumulative propagation delay, that is one type of fault and there may be delays and glitches so that can be caused by crosstalk. So, crosstalk means when there are 2 lines running parallely, then these lines 1 line value may be affecting the other line.

So, we have got path delay fault. So, path delay fault is the cumulative propagation delay through the circuit. So, there are for detecting such faults. So, we have to have 2 test vectors; the first test factor will set the value of output value at some point like say in this particular case if we apply the first the pattern 0 1 1. So, this value becomes 0, this value becomes 1. So, sorry if this value is also 0 1 1 this is inverter. So, this also becomes 0. So, as a result over all I get a value of 0 at the output. And now if I apply the pattern 0 01 and this inverter has got some amount of delay so 2 unit, where are these end gets they have got delay of 3 units, and this has got a delay of 2 units, then for some amount of time so it will be able.

So, that though the result is expected within this if there delay was not there the result is expected immediately, but now the result will come after time 7; at t equal to 7 this transition occurs, whereas at t equal to 0 the output has the input x 2 has changed from 1 to 0, but in at t equal to 7 only we get this. So, this is the delay that we have so that delay may or may not be acceptable depending upon the circuit specification. Why this thing happened? This thing happened because due to the delays of individual where gets, plus if I have got a number of lines running parallely signal lines running parallely, then there may be the effect of 1 line coming on to other line. So, 1 line before other line transition either positively or negatively.

(Refer Slide Time: 23:28)

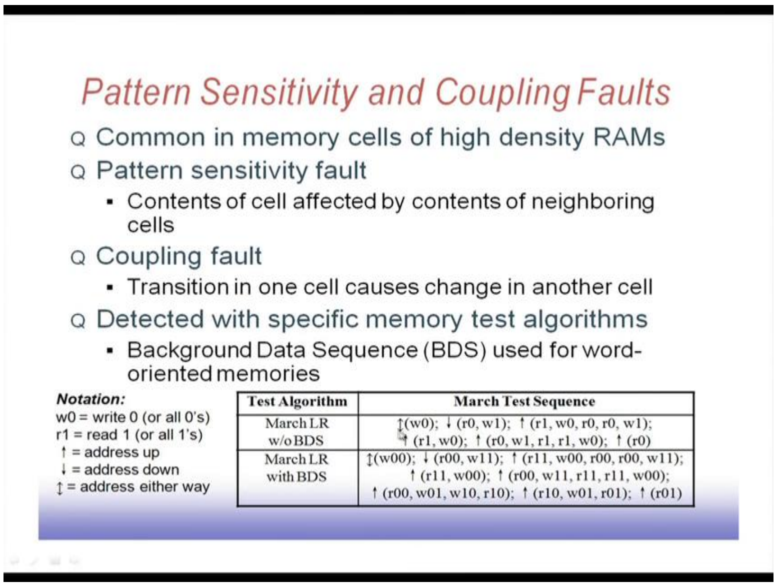


Pattern Sensitivity and Coupling Faults

- Common in high density RAMs
- Pattern sensitivity fault
  - Contents of memory cell is affected by contents of neighboring cells
- Coupling fault
  - Transition in contents of one memory cell causes change in contents of another cell

Another type of faults which are common in memories, so they are called pattern sensitivity and coupling faults, they are common in high density memories; so pattern sensitivity fault. So, this actually what it says is the contents of a memory cell is affected by contents of neighboring cell and there is a coupling fault which says a transitioning contents of 1 memory cell causes change in the contents of another memory cell.

(Refer Slide Time: 24:58)



Pattern Sensitivity and Coupling Faults

- Common in memory cells of high density RAMs
- Pattern sensitivity fault
  - Contents of cell affected by contents of neighboring cells
- Coupling fault
  - Transition in one cell causes change in another cell
- Detected with specific memory test algorithms
  - Background Data Sequence (BDS) used for word-oriented memories

Notation:
w0 = write 0 (or all 0's)
r1 = read 1 (or all 1's)
↑ = address up
↓ = address down
↕ = address either way

Test Algorithm	March Test Sequence
March LR w/o BDS	↕(w0); ↓ (r0, w1); ↑ (r1, w0, r0, r0, w1); ↑ (r1, w0); ↑ (r0, w1, r1, r1, w0); ↑ (r0)
March LR with BDS	↕(w00); ↓ (r00, w11); ↑ (r11, w00, r00, r00, w11); ↑ (r11, w00); ↑ (r00, w11, r11, r11, w00); ↑ (r00, w01, w10, r10); ↑ (r10, w01, r01); ↑ (r01)

So, what we actually mean is suppose this is a memory, so memory is bit organized. So, these are the rows these are the columns. Now if we are considering say value at this point value at this cell. So, it may be that due to some manufacturing defect, I cannot change this independently. So, if I want to put if these value all these values are say 1 and say this value is 0 then this value automatically becomes 1. So, that way there is a pattern sensitivity faults. So, certain pattern in it is neighborhood may force a point to go to some other value and another difficulty is the transition fault, where the coupling fault it means if I want to change in locations value against the same thing; suppose I want to change this location value from 0 to 1, from 0 to 1, it may so happen that when I am doing this thing. So, this location value also changes.

This is known as the coupling fault. So, this it is common in memory cells of high density rams, and this pattern sensitivity fault so it contents of cell affected by contents of neighboring cells, and this coupling fault we means a transition in 1 cell causes change in another cell. Now how to detect this type of faults? For detecting this type of faults, so we have to we have got a special type of testing technique which has specific for memories which a so one such testing is known as March test. So, in March test what we do, we write and some pattern to the memory cells and try to read the content and see whether this has been check properly this has been done properly or not. So, this is one possible march test sequence; what it says is w 0 means I will write 0 and then this upper row and down. So, I also is there is an upper row means I will be writing in increasing order address; similarly this down arrow means I will write in the decreasing order address.

So it says that it can since it is a both directional bidirectional arrow, so I can just do it in I any sequence that I want see. If I decide that I will do it in the upward address direction. So, first where for the lowest say I will write is 0 there then the next cell then the next cell this way I will write go on writing all the cells the 0 value; then once it is done I will come to the next element, the next element cell tells that the from the highest address to the lowest address we first we read operation, and in the read operation we are expecting is 0 to be 0 to be rate because previously we have written a 0, and after reading this 0 that cells content is changed to 1.

So it is basically checking whether these zeroes have been written properly and then it is trying to change this 0 to 1; then once I have written the entire memory checked and written the entire memory, now in the increasing address direction I try to read the value and the expecting that at to be 1, try to write then I write is 0 there, then the read operation on that cell expecting it to 0 and other read operation. So, basically if 1 read operation distracts the content of the cell, so that will be detected by this particular test because I am reading 0 then again reading 0. So, 2 reading read operations they should give me the correct values then this write one will be done.

So, this way I can design test sequence which can detect many of the faults in the circuit, in the in the memory cells and that is without any background data for word oriented memory. So, there is another issue which where you have to put some background data sequence. So, here are to give this background data that was previously it was 0 similarly, previously it was 1. So, like that are to put some background data also.

So, we will come to that in detail, when we go to the memory testing yes stop here continue in the next lecture.