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

Lecture No #31
Testing (Part – II)

So in our last class we were talking about this single stuck at faults model. So today we first talk little bit about the multiple stuck at fault model before going on to other fault models.

(Refer Slide Time: 01:16)

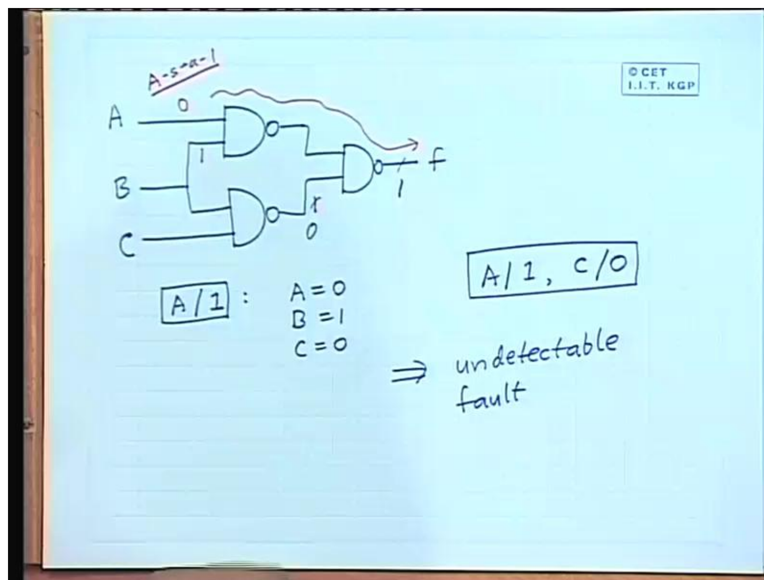
Multiple Stuck-at Faults

- Any set of lines is permanently stuck-at some combination of (0,1) values.
- The total number of single and multiple stuck-at faults in a circuit with k single fault sites is $3^k – 1$.
- A single fault test can fail to detect the target fault if another fault is also present.
  – called *fault masking*, which is rare.
- Statistically, single fault tests cover a very large number of multiple faults.

Now the multiple stuck at faults model as I mentioned in the last class that here we assume that more than one line in the circuit can have a stuck at. So any number of lines can be stuck at 0 or stuck at one. Now the point to note here is that the number of possible multiple stuck at faults are much higher as compare to single stuck at faults which is if you just compare it is only 2k. The number of single stuck at fault is only 2k. In contrast multiple stuck at fault is 3 to the power k minus 1. How this number is coming? You see in a circuit there are a k number of lines okay. Now each of these lines can be in one of three states good stuck at 0, stuck at 1. So there are three combinations in each of the line, so three into three into three it becomes three to the power k.

And there is one combination where all these lines are good. So you subtract that one from this, so three to the power k minus 1. This is the total number of multiple stuck at faults which are possible. Now as I would mentioned that since are test set which can be detect single stuck at faults only can also detect a very large percentage of multiple stuck at faults. So people usually do not consider multiple stuck at faults. More over multiple stuck at faults have some problems which also need to be addressed like one such problem is called fault masking. Now this fault masking says that if more than one fault occurs at a time, then a test set to detect one fault maybe invalidated if another fault is also present. Well I will just illustrate this with an example.

(Refer Slide Time: 03:35)



Let us take a simple gate level circuit two level NAND circuit. So the inputs are A, B and C and the output is f okay. First we consider a fault say A stuck at one. In order to detect A stuck at one, what is the test vector you need to apply? See for A stuck at one, you have to apply a reverse logic value to the line A. So A has to be zero, this A has to be zero in the first place and in order that this I am trying to detect A stuck at one. So in order that this possible change in the input propagates here this has to be put to one. Because if this is zero the output will always be one, this means B has to be 1. Now again the effect of the fault width trying to propagate through this path so the output of this NAND gate must be at 1. Because if it has 0, f will again always be at 1, there will be no change.

So in order to make it 1, C must be 0. So to detect A, stuck at one I have to apply this this input vector. Now suppose in addition to A stuck at one I have another fault which is also occurring C stuck at zero. So what I am saying is that if you want to also detect C stuck at 0, then the line C has to be put at 1. This one will make this line at 0 which will intend make the line of, f at 1. So the fault A stuck at one will not be detected. So this example shows that if two faults occur together then we can have some faults which become undetectable faults. So a fault f1 may become undetectable if a fault f2 also occurs in the circuit okay fine. So these are the most popular gate level fault models which people used.

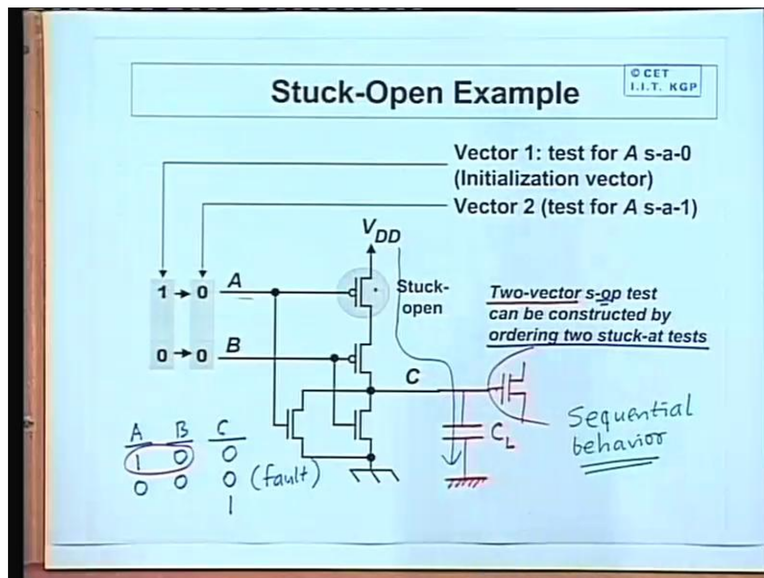
(Refer Slide Time: 06:45)

Now we also sometimes consider transistor level fault models because of two things. Because number one most of today's VLS circuits are designed using MOS technologies the CMOS. Number two the MOS faults they executes some peculiar behavior there are certain kinds of faults which cannot be detected by the conventional single stuck at fault test set. So they have to be considered separately. So let us see what these are. See the transistor is sometimes also called as switch an ideal switch. That is why a transistor faults or sometimes also called the switch level fault models. Most transistors is concerned as an ideal switch ideal switch in the sense that if you have a most transistors like this, control by the gate if it is n mask if the gate is exact logic 0 I say

that the switch is off if the gate is at logic one I say the gate is on it is conducting. It is just like an ideal switch. Now the transistor level faults are two types stuck open and stuck short.

Stuck open says that the transistor is never turning on. Whatever you are applying on the gate it is permanently stuck in the open, opens. That means the source and the drain they are never conducting okay. Stuck short is the reverse there is a short circuit between the source and the drain. So single transistor is permanently shorted irrespective of its gate voltage. Now these two kinds of faults are peculiar. Peculiar in the sense that the stuck open faults some of the stuck open faults cannot be detected by a single test vector. They will require a pair of test vectors for detection, this we will see why. And similarly stuck short fault model sometimes may require the measurement of current from VDD to ground. Just by using any logic measurement we cannot detects such faults okay. Now let us see why these kinds of peculiar behavior exhibit in case of a transistor fault okay.

(Refer Slide Time: 09:25)



First we look at the transistors stuck open example. Now stuck open example before I explain this. Let me, tell you one thing that stuck open is interesting because this is an example of a two input NOR gate. Now stuck open fault can make combination circuit execute sequential behavior. Well what I mean by saying sequential behavior is that the combinations circuit now behaves somewhat like sequential circuit. This means that the output which you get in the

present instance depends on the output which was there in the previous instance of time. This is not in the split of combination circuit because in a combination circuit the output depends only on the applied inputs at the present point and time okay.

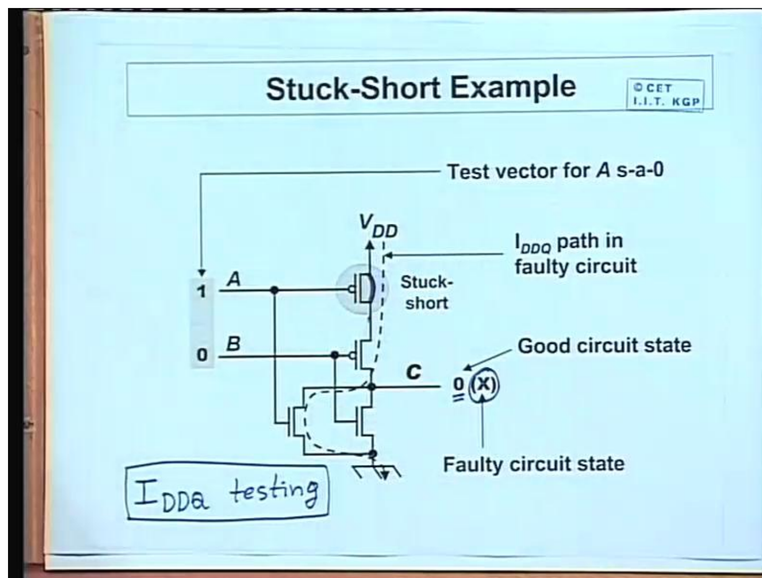
So let us see why this happens okay. This is the circuit diagram of our two input NOR gate. We assume that this pull up transistor is stuck open this is always off. Well here we observed one; thing this is an isolated gate. Now this gate maybe driving some other gate in the next stage. Some other gates mean what it will driving the gates of some other transistor. It will be driving the some other gate like this. So equivalently we can say that the output is driving a capacity load CL, this is a load capacitance. So you ignore this transistor equivalently. You can say that the output has a load capacitance connected to this load capacitance is actually the gate to ground capacitance gate to. Now here suppose I want to test this transistor stuck at the stuck open.

Now let us see what happens here. Stuck open means this line is this line is always open circuit right. Now you just imagine a situation suppose this A B at the inputs. Suppose I apply an input pattern which is say 1 0, as this source A is 1, B is 0. If I apply 1 0 these are P most transistor. So this will be off, this will be on and one of the bottom transistors should be on. So the output will be connected to ground and this capacitance CL will get discharged so the output C will be at 0. Now suppose I apply an input say 0 0, then what will happen 0 0; 0 0 means both the pulled down transistors will be off none of them are conducting both the pull ups where supposed to conduct. But since this transistor has a stock open fault this is open.

So now this output lines C is like floating it does not have a connection to ground NOR it has a connection to V_{DD} so current cannot flow. So what will happen to CL is that CL will be holding the previous logic value whatever was there. So in presence of the fault C will be zero because capacitor was already discharged this is in presence of the fault. Capacitor was already discharged it remain discharged because that is no path for charging. But if this fault was not there, then for anyway apply 0 0 both these two transistor should be on and the capacitor could get charged. So this path and this C value could be one okay. So here we can distinguish whether this fault is present or absent. But you try to think that if I do not have if I had not applied the previous text vector could I detect this fault? Because the previous test vector was applied with the sole intention that I wanted to discharge the capacitor to known state.

But if I simply apply 0 0 0 0 means the output is open. But what will be the output value that is do not know that depends on whatever was the previously. So in order to detect the fault I first set the output to a known state by applying one zero for I could apply 1 1 also. Then by applying 0 0, I am exiting this fault okay. So this is one example using which I can detect a stuck open fault. But I have to apply two vectors in sequence. This is called two vectors stuck open test. Now as I told you a normal stuck at fault test set may not have these two vectors coming one of two other sequence. But you can sometimes a reorder the stuck at test by moving them up and down. So that this sequence appears one of to the other and using that you can detect many stack open faults also okay. So this is one peculiarity of CMOS circuits.

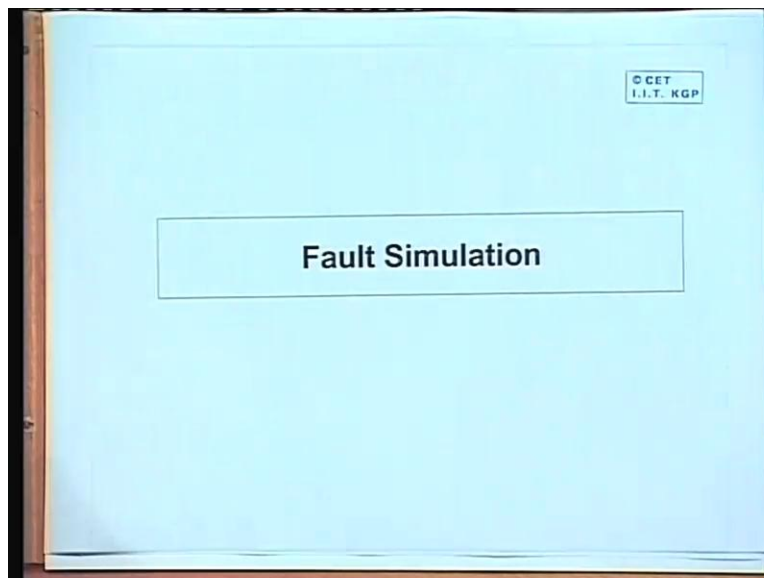
(Refer Slide Time: 16:29)



The other one is this stuck short fault. Stuck short fault is different. See this pull up transistor has a stuck short fault which means there is a permanent connection like this. Now if I apply 1 0, input vector here, then well under normal circumstances the pull up network could be off pull down could be on and we were expecting a logic value 0 in the output. Because the output capacitor could discharge to ground. But since this transistor short now from V DD to ground there will be a current flowing path. So this circuit will act as a voltage divider there will be a high current flowing and the output will neither be at logic zero level nor be at logic quantum somewhere in between it will be in determinate. This kind of a think cannot be simply detected

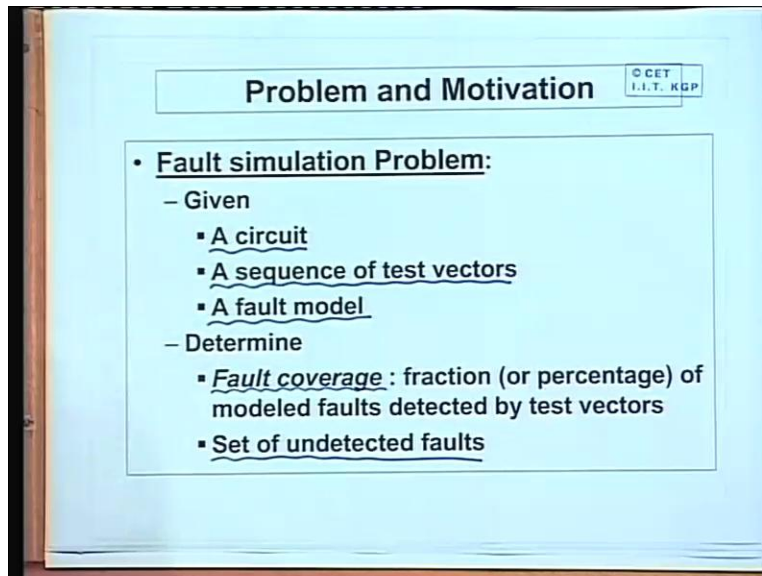
by observing the logic value of the output rather here you will have to measure the current that flows from V DD to ground. This is sometimes called I_{DDQ} testing. You are measuring the quiescent current from V DD to ground this is I_{DDQ}. So I_{DDQ} testing or measuring this high current sensing this high current flow is required for detecting such stuck short faults in most transistor okay. So we have looked at some of the fault models that people normally who is, we are constructed only on the stuck at and the transistor models. Because there are more important from circuit design point of view. Of course people also talk about functional and high level fault models as I mentioned. But we will not be discussing them as part of this discussion. We now move on to the problem of fault simulation.

(Refer Slide Time: 18:46)



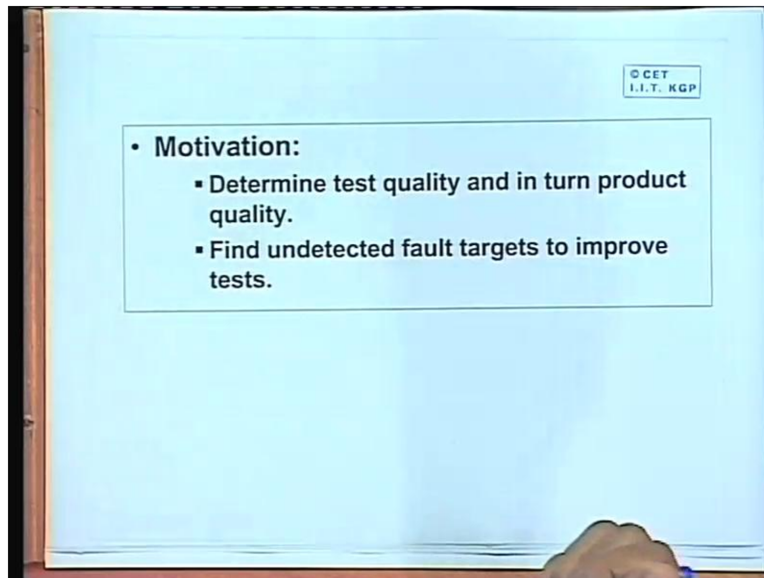
Well now we have talked about the fault models we know that which faults are to be considered. Now fault simulation what is this. This is a process whereby we create model of a circuit on the computer. We also have the set of faults that need to be considered, that also is represented in the computer in some way. Then what we do using the program the simulated program, we find out that given a test vector. What are the different faults that are getting detected okay?

(Refer Slide Time: 19:33)



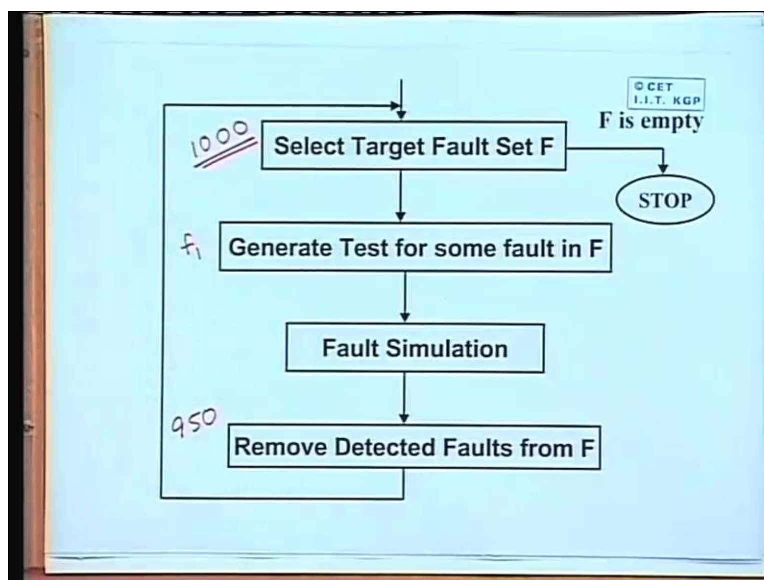
So the faults simulation problem and motivation are as follows. In the problem we are given a circuit typically or net list of gates were also given sequence of test vectors which you want to evaluate and of course a fault model. These are the three things which are given to us. So what do you want to determine? We want to determine the quality of the test vectors. We want to determine a parameter called fault coverage. Fault coverage is the fraction or the percentage of the modeled faults which are actually detected by this given test vectors. Moreover the fault simulation can also give you a set of undetected faults for which you may have to generate some additional test vectors okay. So faults simulation is useful in evaluating the quality of a test vector which you have with you okay. So the motivation mentions that very clearly you determine the quality of the test vectors.

(Refer Slide Time: 21:00)



Now since your testing your product using this test vector, see if you can improve your test quality your product quality will also get improved. And undetected faults if you also find that you can generate some additional test vectors to detect them okay.

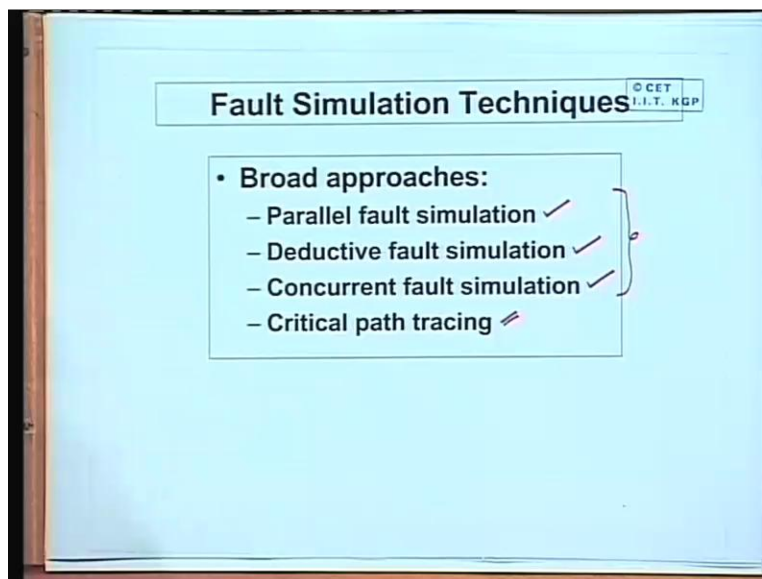
(Refer Slide Time: 21:25)



So the typical flow is like this. You start with a fault set F. You generate the test for some fault in F. Well you need not generate fault for all the faults in F. For example these faults set F may contain thousand faults for example there are 1000 faults here. You take any one of those faults and try to find out R test vector for detecting that fault. Why? Because if you carry out faults simulation after that say this test vector was generated for a fault F1. Now faults simulation will tell you that instead of F 1, another 50 faults are getting detected by that same test vector. So what happens is that although you have generated a test for a single fault, but the corresponding test vector is able to detect 50 faults.

So after this you can remove detected faults from F. So now this set of 100 faults becomes 950. This process is iterative. So the idea is that instead of iterating 1000 times at each step you reducing the number significantly. So that the number of times you will have to generate the test becomes less. So this faults simulation is typically used in conjunction with test generation in order to improve the quality of test. So as the fault test fault set it becomes empty or becomes very small. You can stop this is stopping criteria. Now in practice there are number of different techniques which are used for fault simulation.

(Refer Slide Time: 23:36)



Broadly the approaches which are followed can be categorized as parallel fault simulation deductive concurrent and critical path tracing is of course a specific technique which can be used only for combination circuits. But the first three can be used also for sequential circuits its some modification. So let us try to see briefly for these methods really are okay.

(Refer Slide Time: 24:11)

Parallel Fault Simulation

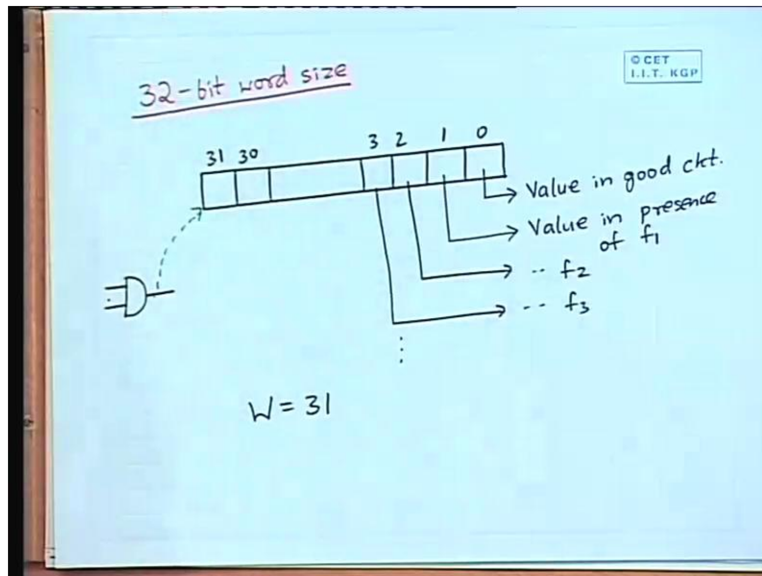
- The good circuit and a fixed number (W) of faulty circuits are simultaneously simulated.
  - For F faults, we require ⌈F/W⌉ passes of simulation.
- How simulation is carried out?
  - Using bitwise logical operations.
  - Fault insertion is carried out using *mask words*.

Before going into the parallel faults simulation, let us first try to understand what simulation really means. See we have a gate level net list with us. Now gate level net list it can be represented on the computer as a graph. The gates or the vertices of the graph the edges are the interconnections. So representation of a circuit in a computer is not difficult. And also the note types have to be stored which is AND which is OR which is NAND and so on. Now if we have some input which is applied we have an input vertex vector, suppose you have four input circuit I have 1 0 1 0 applied. So on that graph if we systematically trace from the primary input side. Towards the primary output side we can evaluate the nodes one after the other depending on the type and finally we can find out the logic value of the primary output okay. Now this is called true value simulation. True value simulation means in the actions of faults if I apply this test vectors this will be the output.

Now I can modified this simulation process a little bit I can possibly inject some fault in some in some circuit. For example I have an AND gate somewhere I say that the output of this AND gate is stuck at one. So if I keep this information then while I am evaluating that AND gate during the process of simulation, whatever maybe the input I will always evaluate the output as one because it is a stuck at one. So I can inject faults like this and I can find out the output value. For a given input test vector if I see that my good or fault free output and the faulty outputs are different then I can announce that this particular fault is getting detected. This I can repeat for all the faults okay. So if there are 1000 faults I have to repeat 1000 times okay. Now let us see for this parallel fault simulation means this is a method to speed up the process of faults simulation. Here it says the good circuit and a fixed number of faulty circuits are simultaneously simulated we will see how.

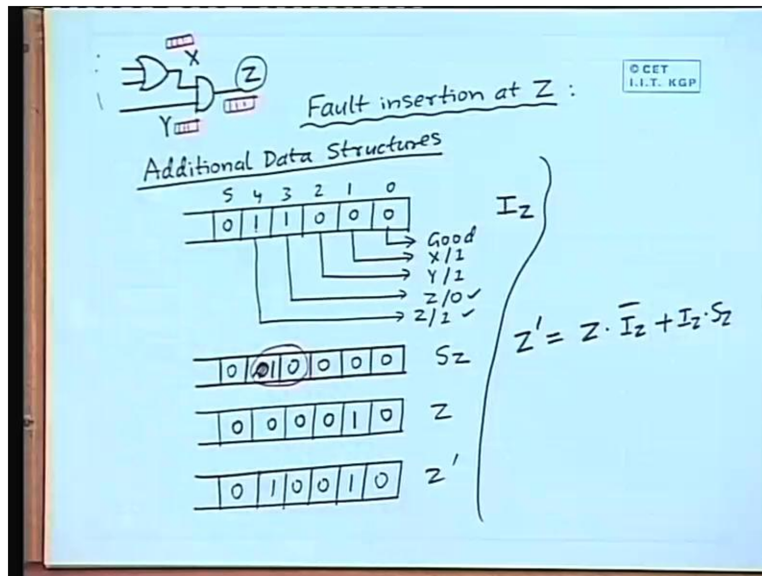
See in the normal scenario you can simulate only one circuit at a time. So either you are simulating the good circuit or you are injecting a fault you are simulating the faulty circuit. But here what I am saying that we are parallel simulating the good circuit and W number of faulty circuit all together okay. So if there are total of F faults, since we can simulate W number of faulty circuits at a time, so we will be needing F divide by W sealing of that so many passes of simulation. Now how we are doing this means how we are simulating this in parallel. See for this we are utilizing the inherent parallelism that is there in the arithmetic logic unit of the computer. See ALU or the instruction set of computer has instructions like OR, AND, XOR and so on. They do a bit by bit OR, bit by bit AND of the corresponding bits in a word. So if it is a 33 bit word I can have I can have a 32 bit OR with another 32 but number bit by bit OR. So I can utilize this bitwise logical operator to do this kind of parallel activity. So we will show you how.

(Refer Slide Time: 28:51)



First thing is that okay say we take typical example we have a computer with 32 bit word size. We take a 32 bit word. These are the bit numbers 0 1 2 3 30 30 1okay. Now what I do is that these 32 bits of a word I assign some meaning to each bit positions. For each gate in the circuit suppose there is an AND gate I assign a word like this to the output of this AND gate. For each and every gate of the circuit I have this kind of a word as so in fact this word will be assign to each line in the circuit not necessarily the gate output there can be fan outs also. So the significance is that well as a matter of convention bit number 0 we will be using to indicate the logic value in good circuit. Bit number 1 we will indicate the value in presence of fault f1. Bit number 2 value in presence of fault f2 value in presence of f3 and so on. So what I say is that I am storing 33 bits or 32 values for the gate output at the same time one of them will represent the good value and the other 31 will represent values in presents of 31 different faults. Which means, I am simulating 31 different faults at the same time okay? Now let us see how we can do this.

(Refer Slide Time: 31:29)



Let us take a very simple example and illustrate. Suppose I have a circuit like this I have an OR gate. This is driving and NAND gate the output value is Z, this is X, this is Y. Now this 32 bit word data is set. There will be a word assign to each of these X, Y as well as Z. Right. Now we are concern with the line Z. We will show you how we carry out fault insertion at till line Z. See we will be systematically proceeding or carrying out the simulation from the primary input side towards the primary output. We assume that we have just reached this AND gate and we are trying to evaluate this taking into account the additional faults in the output set.

Let see how we can do this. Well you maintain some data structure at Z not only the word which stores the values, we have some additional data structures. Let us see what these are first thing that we stored. This is also a list these are the bit positions okay. This is first the word which I am this list that I am I am just talking about. Say just I am taking an example. Suppose the first bit indicates the value for the good circuit. The second bit is the value for the fault X stuck at once. Sometimes we also write like this X slash one means stuck at one. This is for say Y is stuck at one. This is for Z stuck at 0. This is for Z stuck at 1 and so on.

So X and Y stuck at 0 have been collapsed. So they do not appear. So there will be other forces circuit. Also there can be some faults W V lines like here though they will also be there. So I am showing some of the faults which you can see from here. But why you reach here you are more

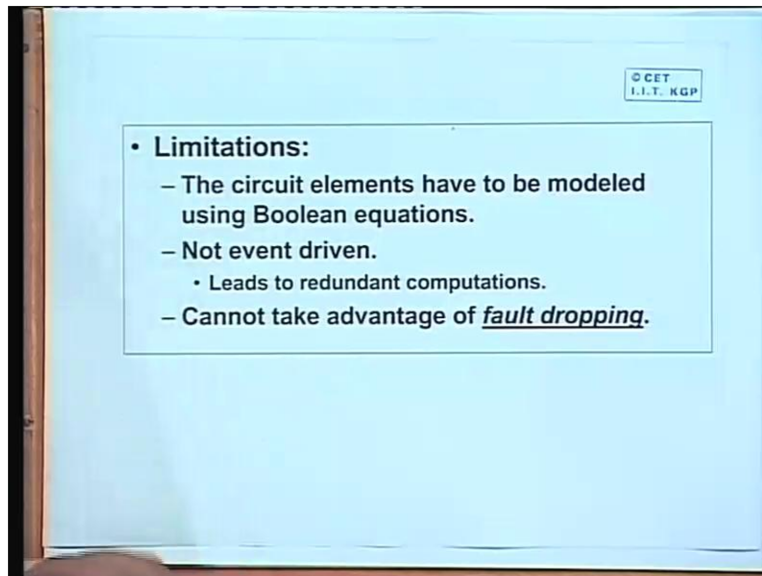
interested in the false on the line Z the output of the list. So this list we are calling I_z there is some other data structures also I am showing you. There is another data structure called S_z switch will tell you that well. Now this I_z is what okay I am telling you what this means. This I_z is not Z of the list rather, it tells you that here right now here we are trying to inject a fault on Z.

So these are the two faults of our interest. So the faults which are of interest now I am setting those bits to one rest all are zeros okay. These are I_z and S_z says that these are the two faults in the consideration. Fine. But first of them first one is stuck at 0. The stuck at sorry this is stuck at 1 and the second one is stuck at 0. The rest all bits will be 0. Only these two bits are relevant this 1 and 0. They will indicate the polarity of the stuck at faults on this line. Bit number 4 will indicate stuck at one, bit number 3 will indicates stuck at zero okay. Now the value of Z you can calculate by taking simple AND of the list you have X and Y.

Suppose I have Z value calculated as follows 0 1 0 0 0 0. And the formula that we used is that you can calculate the new value of Z dashed taking in the account the faults at this particular side using a formula like this Z and I_z bar or I_z and S_z. The first one says I_z bar means what other than these faults all the other faults. So ignoring fault on Z, you take out only those two bit positions from Z and return all other bit position as it is for those two bit positions u and I_Z with S_z which is stuck at 0, stuck at 0 insert it there. So if you do this Z dash will become 0 1 0 0 1 0. So in this way, see these additional data structures are used only for the purpose of inserting in the faults.

Because as you go on simulating faults on this particular gate we will have some effect in determining that which are the positions that will be to modified. So this formula is used to make this modification. And in essence this is how the method of parallel for simulation works. That you start from the primary input side you go on systematically stressing the circuit from input to the output. At each line the circuit you maintain this data structure this list you manipulate the list you will try to evaluate that what is the value get for um for that particular output of the gate before you proceed to the next level and you go on repeating this. Now this method is alright it works but it has some limitations.

(Refer Slide Time: 38:56)



Limitation in the sense first thing is that this method works only if the circuit elements can be modeled using Boolean equations. Because you are trying to utilize the inherent parallelism in the ALU's in the instruction set so unless you can map the basic operation into operations like AND, OR, NOT, you cannot. So you cannot apply if you cannot apply a scheme to slightly higher level modules like a multiplexer or half adder for example. It has to be only the primitive gates AND, OR, NOT, XOR, NOT okay. And 17 is that it is not event driven. Event driven means sometimes in order to speed up this simulation process. What do we do? Suppose we apply one input to the circuit we simulate to evaluate the output. Now you apply the next input. After plus the next input we find that only in small portion of the circuit there some changes in the logic values. So you only re-evaluate that part; the other part you do not reevaluate.

But in parallel faults simulation you will have to evaluate all the gates every time in, means each of the parts. Because in the same word you are encapsulating so many faults together. So may be first some of the faults it is not relevant. But for some other part it may relevant. So you will have to evaluate all of them okay. So this may leads to redundant computation and more over you cannot take advantage of this so called fault dropping. Well we did not mention this term. We just explain this that as the faults get deducted do you remove them from the fault list. But in parallel fault simulation because since we have already pack them in the machine words, we

cannot very easily take them out from there okay. So fault dropping is not possible it is not event driven and it cannot be used for higher level function blocks. These are the drawbacks.

(Refer Slide Time: 41:22)

Deductive Fault Simulation

- Simulates the good circuit and *deduces* the behavior of *all* faulty circuits.
- Data structure used:
  - A fault list L_i associated with every line *i*.
  - During simulation L_i is the set of all faults *f* that cause the value of *i* in the faulty circuit to differ from its fault-free value.
  - If *i* is the primary output, L_i denotes the set of faults detected by the test vector.

Let us look at some other fault simulation techniques like detective faults simulation. Now deductive faults simulation is an interesting method. This is an event driven faults simulation in the sense that you only simulate the gate. If some event occurs in one of the input if there is no event occurring then there is no need to simulate that gate. And unlike parallel for simulation you are not maintaining any words at the lines which store information about many faults. Rather you maintain as set on each line a set of faults which are getting detected just something called a fault list. So you assign a fault list with every line of the circuit and using some simple rules if propagate this fault list from the input towards the output side. So let us see the basic idea is you simulate the good circuit in the absence of faults. So when you simulate the good circuit you know the fault free logic value in all the lines of the circuit from that you try to deduce the behavior of all faulty circuits and in the process you also try to deduce the difference faults that may get deductive.

Now for this purpose we use some data structure, the data structure is like this. You have a fault list Li associated with every line Li i. Now Li is what is that during simulation the interpretational Li is that, this Li will contain the set of all faults that will cause the value of the

line i in the faulty circuit to differ from the fault free value. Which means if you have an AND gate for instance and the good value simulation says that the output value is 0. So the fault list for this output will contain all those faults which will result in this gate output value to be one the other value okay. So you continue this process finally you arrive at the primary output. So in the primary output the corresponding set will indicate those faults which will actually get detected because those faults which will change the output value from the fault free value. Means those are the faults which are getting detected. So we will explain how but there is another important step in this method this is called fault list propagation.

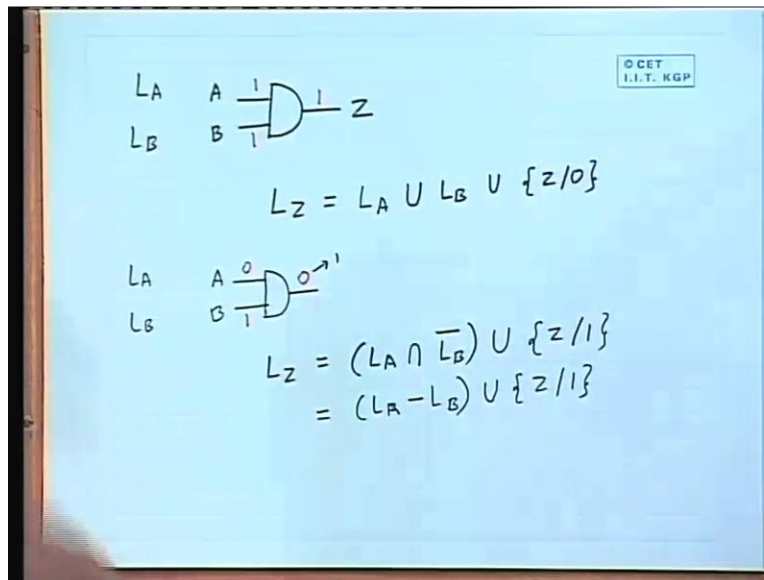
(Refer Slide Time: 44:36)

- Fault list propagation is carried out under two conditions:
  - Logic event
    - Change in signal values
  - List event
    - Change in fault list

Fault list propagation means you can have the output of the gate feeding some other gate okay. So you have a fault list out here. So this fault list may be propagated to the output of the next gate with some modification. There are some rules. Now a fault list will get propagated provided one of these two conditions occur. First is logic event. Logic event means suppose you have to applied say one test vector 1011 for that you are doing this. After you have done this you have applied another input vector. Now in response to this input vector some of these signal values of this gate achieve. For example this was zero now this has become one. So if there is a change in signal value we call it a logic event. Then you need to have a relook at this set because some of the entries in the set may have to changed. Similarly there is another kind of event called list

event. This says if the fault list of the preceding it here there is some change out here. Then possibly this fault list will also get changed. So if either there is a logic event or this is a list event you need to re-evaluate the fault list okay. Now let us try to explain how this is done.

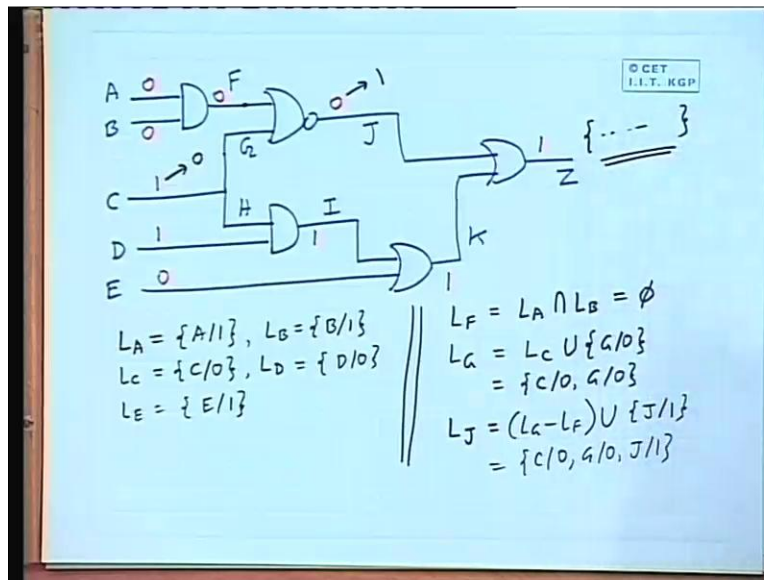
(Refer Slide Time: 46:16)



Suppose I have n lets take a simple example. First I have a NAND gate A B Z. Suppose will this is an isolated and the inside the circuit I am assuming that the fault list L_A and L_B are already available to me. These are two sets I am trying to find out the value of L_z. Now under the present scenario say the logic values are 1, 1 and 1, this is the true value logic. Now you try to see that what will be there in L_Z in L_Z we will have all the faults which will change the output logic value to zero. What are the things that may change. First is L_A. Because L_A the correct value was one if any of these occur it will change this value. So L_A means those set of faults which will change the line A to 0. So this is one condition when Z can change to 0 or union $L_{B;}$ if B becomes 0 then also union and new faults comes in Z stuck at 0. So if you have a AND gate with these applied inputs then the fault list of the output will be or it can be calculated like this. But let us take the same AND gate the same input faulty say L_A and L_B with set the applied input values 0, 1 and 0 okay.

Now let us try to see under this condition how we will calculate L_Z. See this AND gate output is 0. So L_Z will contain all those faults which will make this 1. Now let us see that how we can have this at 1. There are two conditions one condition is that B will not change B will remain one. But A will change okay. So A will change B will not change how do you write that we write it like this L_A. L_A means all faults here it will changing intersection L_B not of that. L_B not of that means none of these faults which change B intersection of this if there is anything common there that can have this 1 1. This union this additional fault Z stuck at 1. Now L_A intersection L_B bar in security form we can also express it us L_A minus L_B union Z stuck at 1. So I have illustrated this process for AND gate you can similarly think and extended to OR, NAND and NOR gates. The process is similar. We will have to see the type of the gate and the applied inputs and you will have to systematically trace it out.

(Refer Slide Time: 50:05)



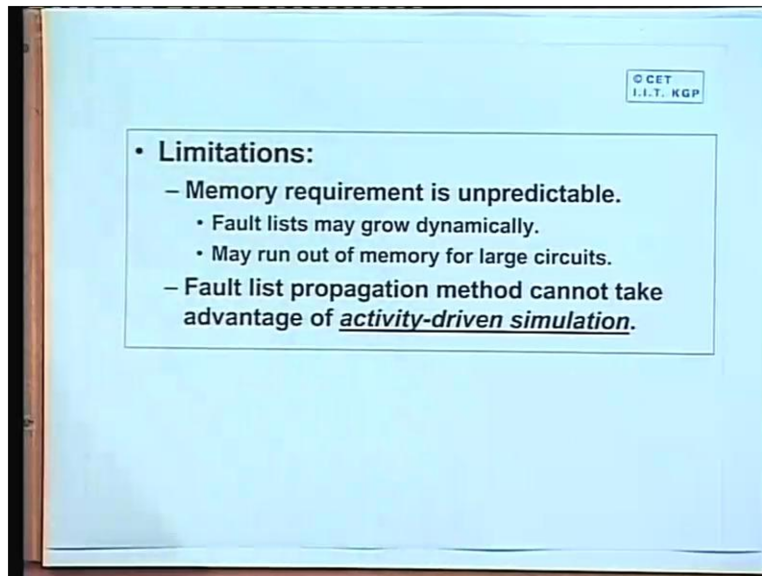
Now let us take biggest circuit and try to find out the fault list. Let us take a circuit like this. A, B this is D this is E and gives a name to the circuit called this F, G, H, I, J, K and output as Z. And let us say that we have applied the input vectors 0, 0, 1, 1 and 0. So the true values simulation says AND gate output is 0, OR gate one of the input is 1. So this will also be 0 1 1. This will be one OR gate will be 1, this will be 1 okay. So now if we try to systematically compute the fault list starting from the primary input L_A. A is that 0. So L_A will contain only A stuck at 1.

Similarly L_B will contain only B stuck at one. L_C will contain C stuck at 0. L_D okay L_D will contain D stuck at 0 and L_E will contain E stuck at 1. Now from this list you can systematically go ahead with calculating the fault list.

For example F, suppose are you want to know calculate LF, so this is an AND gate output is zero. So output can be one if both of the inputs change state. So here the conditional will be L_A intersection L_B okay. Now L_A intersection L_B there is nothing. So this set is pie empty. Similarly you can find out L_G. For example well L_G will be the same L_G will be now L_G will be this is a fan out. So it will be L_C. L_C will be same union and additional fault G stuck at 0. So now it will be C stuck at zero and G stuck at zero. Similarly if you calculate L_J for example L_J what will be LJ. This, a NOR gate with one input at 0, other input at 1 0, if you can one to make it one you will have to make this into 0. So this you can make how this you can make it 0 by having a logic event here and not having sorry G not C it will G logic event here and not having a logic event here.

So it will be L_G minus L_F union to fault J stuck at 1. So L_G minus L_F will be C stuck at 0, G stuck at 0, J stuck at 1. So in this way we can systematically proceed till you will get some fault list in the output Z. So whatever you get in the output Z, those will be the faults which are actually getting detected. Now after this pass when you apply the second set of input vectors you do not through away with the list which you have already calculated. You return them and wherever in the circuit there a change in the changes in the logic value you recomputed the fault list. And if there is a list event you propagate that list event to the output of the next gate in sequence and you redo it. So we will find that in the next step again you will get some other fault is this Z. So there will be some faults will be detected. So this is how this method works.

(Refer Slide Time: 55:21)



The method works faster than fault simulation that the problem is that the memory requirements in is this unpredictable. Because the fault this can grow dynamically you cannot estimate the total memory size for very large circuits. So in general it may run out of memory for large circuits. And the method of fault list propagation which is used in this method it cannot take advantage of some technique called activity driven simulation. Because this activity driving simulation whenever the activity you are simulating others not simulating that you really cannot take advantage of in this scheme. But as such these schemes are very interesting method just by maintaining the fault list and doing a true value simulation. You can find out all the faults which are getting detected in the output circuit. So in this lecture we looked at some of the faults simulation algorithms. So the next class will be continuing with our discussion in this regarding. Thank you.