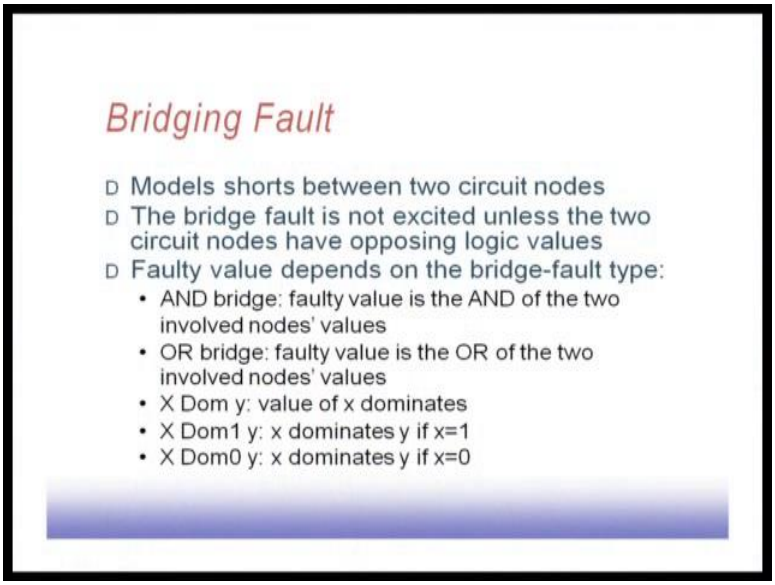
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 23
Test Generation (Contd.)

Bridging fault: so this models shorts between two circuit nodes.

(Refer Slide Time: 00:22)

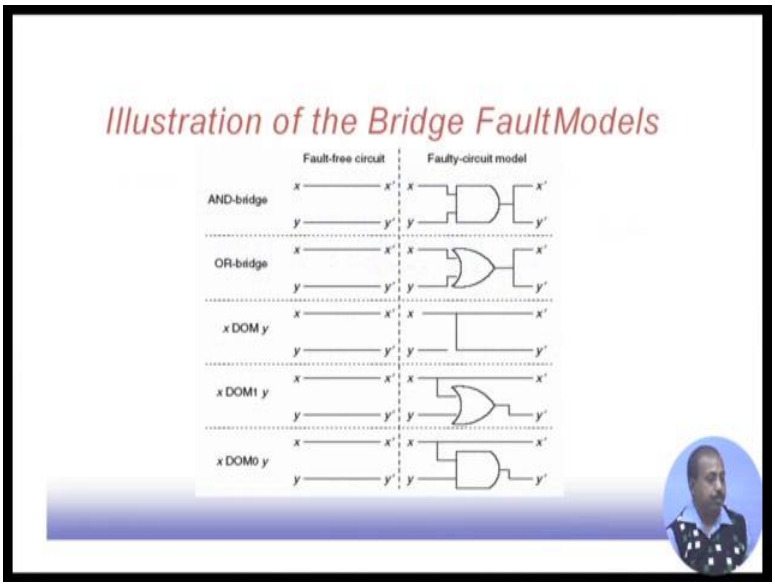


So, this is another type of physical defect where two lines have been shorted. So, the effect of this shorting depends on the technology that we have, and also some other factors that is directed by the technology. So, this is bridge fault it is not excited unless two circuit nodes have opposing logic values. If both the lines are having same logic value then naturally that is not counted as a fault; both of them being 0 both of them being 1, so it is not a not an issue.

However, if the signal values of two lines are different then which one will persist; so that becomes the question. And there can be several cases of this bridging. First one is the AND bridging. So, AND bridging is the faulty values is the AND of two involved nodes values. So, if it is and of AND values then that is that is one type of bridging. Similarly OR bridging means that it will be having two values all of them. So, that will come as the new value down the line. So, these are the two cases.

Another possibility is the dominance. So, x dominates y, so value of x will dominate then x dominates 1 y. So, x dominates y if x is equal to 1. If x equal to 0 it will not dominate, but for x equal to 1 it will dominate. Similarly x dominance 0 y; so x will dominate y if x value is equal to 0. So, these are the different types of cases and actually they have been derived from the physical defects that occur, so they based on that. So, these are the different cases that can happen.

(Refer Slide Time: 02:13)

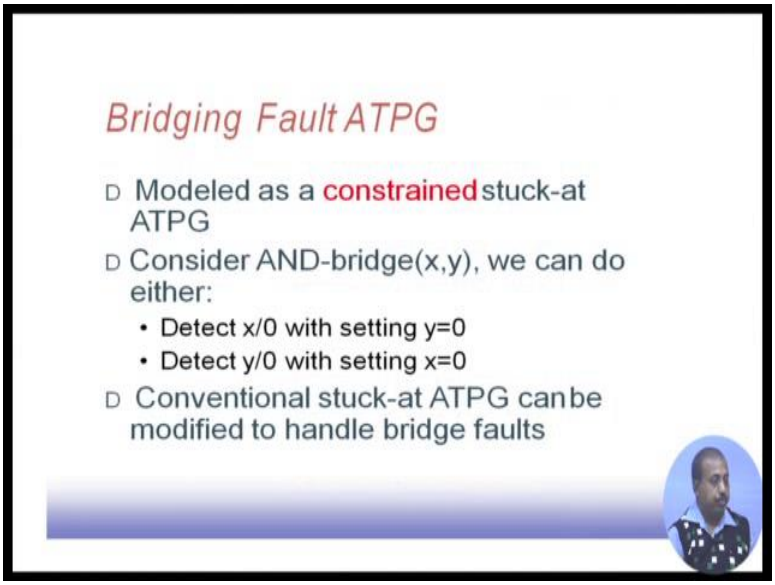


So, these are the symbolic representation like in the fault free circuits this x line and y lines they are running. In the faulty circuit if there is a AND bridging then what will happen, these two values as if there is an AND gate and the AND gate output derives the lines x dash and y dash. OR bridging: these values are odd and OR gate output is deriving the two lines x dash and y dash. Then x dominant y; as if the line y is broken; so in the fault it can moderate like this as if the line y is broken. And there is a fan out coming up line x and feeding the line y dash.

Then x dominant 1 y: if x will dominant over y if x equal to 1. So, this is modeled like into introduction of this OR gate. So, if x equal to 1 then x and y dash they both of them are 1, but if x equal to 0 then y comes to y dash. So, that dominance is not there. Similarly, x dominant 0 y: so that is y AND gate. So, x comes here and this y comes here x, so x value is 0 then irrespective of the value of y y dash value will be 0.

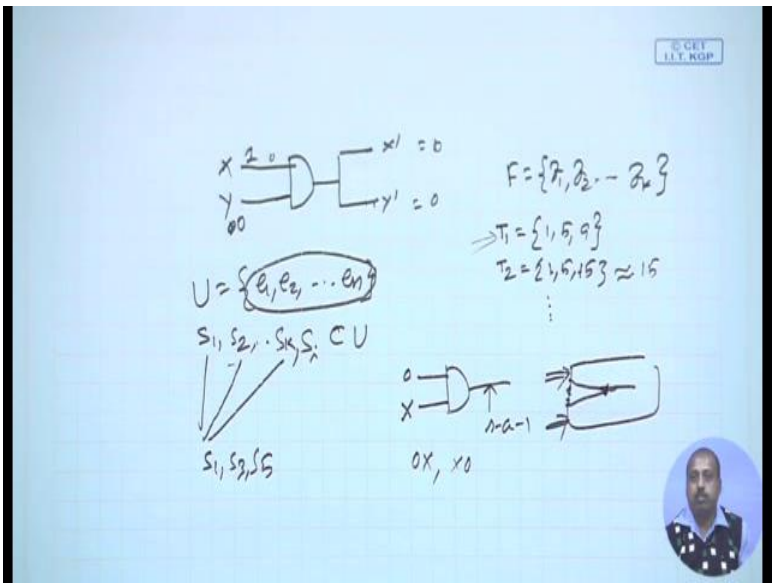
As a result this is another type of faulty circuit model. So, we can modify the circuit to this faulty circuit model and run some ATPG tool to generate the bridging fault patterns.

(Refer Slide Time: 03:42)



So, this is modeled as a constrained stuck at ATPG. It is like this, suppose I am trying to generate a test pattern for AND bridge x y. So, we can do it like this that detect x equal to y, x stuck at 0 with setting y equal to 0. So, AND bridge y. So, AND bridge of x and y situation was like this, this x and y they were coming and they were going to the line x dash and y dash.

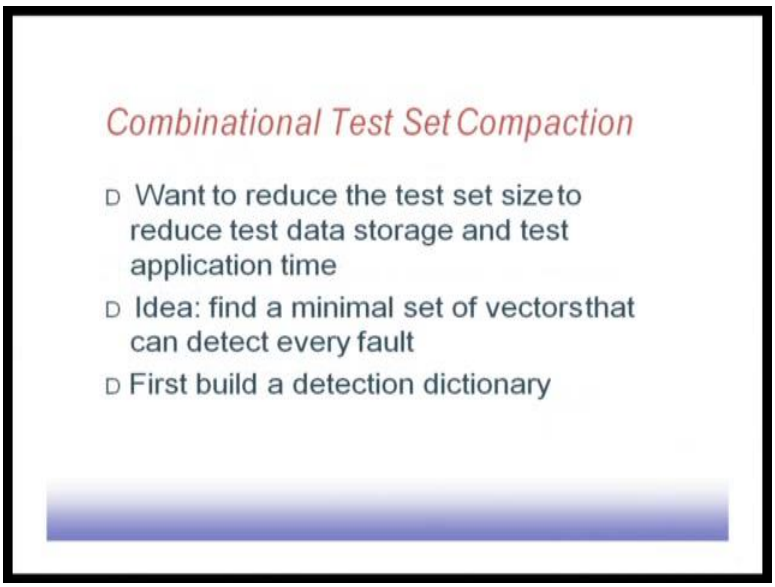
(Refer Slide Time: 04:06)



So, this was the situation. Now, it detect: if we detect x stuck at 0 faults with the setting y equal to 0. So, y is made equal to 0 and under that condition we are trying to detect x stuck at 0. So, x stuck at 0 if you want to detect then we will put a 1 at this point and we will try to see the output here x dash and y dash. So, these two outputs will live to see. So, if this bridging is not there then I will get x dash is equal to 1 and y dash equal to 0, but if this bridging is there then this line is also 0; as a result due to this bridging this gate will become equal to 0. So, that way it can detect this fault.

So, under the constraint that y equal to 0, if we try to detect x stuck at 0 fault then we can find a bridging between x and y. Similarly we can also do this by setting x to 0 if we try to detect y stuck at 0 faults; that also will find out the bridging between x and y. So, you can do either of them, or by doing that you can generate the test pattern for this AND bridging of x and y. So, this conventional stuck at ATPG is they can be utilized for modified for these bridging faults.

(Refer Slide Time: 05:42)



So, like this AND bridging for others also I think you can understand what to do. Like for this OR bridging also you have to do this stuck at 1 stuck at type of thing. Similarly this x dominance y, so we can put this line to some value x to some value and y to some value and we say that we modify the circuit so that this line is coming. We just see that under the constraint that this y value is 0. So, are we value is getting the value of x at the y dash. And similarly at value of y being equal to 1, are we getting the value of the x at

the y dash. So, ATPG tool can be modified can be modified throughput and this type of constants and we can see this pattern generated by them for this purposes.

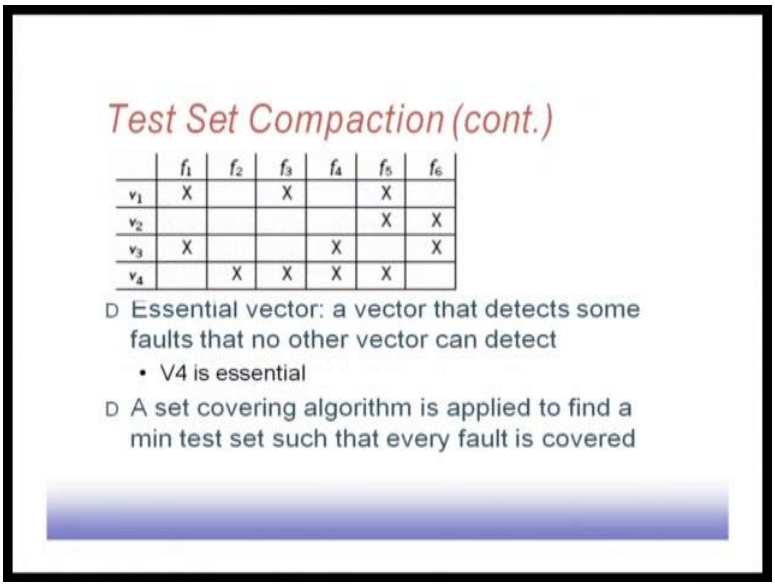
Next, we can look to another very important issue in this pattern generation if which is known as test set compaction. So, what happens is that in the testing process that ATPG algorithm; so it is initially gives the set of false F. So, F is the set of also f 1, f 2 up to f k the k number of faults. Now the test pattern generically ATPG generates the first pattern t 1. Now when it is generates t 1 it finds out the faults that it can cover, may be it covers the false 1, 5 and 9. Then it generates fault pattern t 2 and it again detects some faults.

Now, it may so happen that as we are being the generating more and more patterns that mean faults that are getting covered this by patterns start decreasing. Maybe t 2 detects some faults 1, 5, 15. So with the knowledge that we have as already detected 1 and 5, we can say that t 2 as detected only the fault 15. So, this way the amount of new false that are detected start decreasing as we are generating more and more patterns.

Now, if we want to minimize this number of the patterns in the test set then what is required to be done. We need to select the set of vectors that can be detect also the faults, but the number of patterns is very small. So, what is done? So, what is required is that we can reduce that test set size to reduce the test data storage and test application time, because this cost of this automated testing equipments or ATE they depend on the amount of memory it needs and the amount of channel number of channels that are there. And naturally the test time; that how much if the test set is b then they need to be stored on bigger storage. And also the transfer time from 80 to chip, so that is also going to be high. So, this test application time increases and the test data storage increases as a result the test cost increases. So, if we can reduce the test set size then these factors will come down.

So, what is need is to find a minimal set of vectors that can detect every fault; every detectable fault definitely. Now for this purposes how do we proceed first we build the detection dictionary. So, in a detection dictionary for every pattern we list down the faults that it can detect.

(Refer Slide Time: 09:07)



So, say we have got a situation like this that is ATPG tool it as given you these four patterns v 1, v 2, v 3, and v 4 and there are faults f 1, f 2, f 3, f 4, f 5 and f 6. So, these are the six faults that are in the circuit.

Now, by doing fault simulation we find that v 1 can detect the faults f 1, f 3 and f 5; if v 2 can be detect f 5 and f 6, so v 3 can detect f 1, f 4 and f 6 and v 4 can detects f 2, f 3, f 4 and f 5. Now out of these vectors if you look carefully you will see that there is a the vector v 4 that detects the fault f 2, and this fault f 2 is not detected by any other pattern. So, if you are looking for good coverage then this vector v 4 must be included in the final set of the test vectors vector v 4 must be included. That way v 4 becomes an essential vector. A vector that detects some faults that are no other vector can detect. So, that is the essential vector.

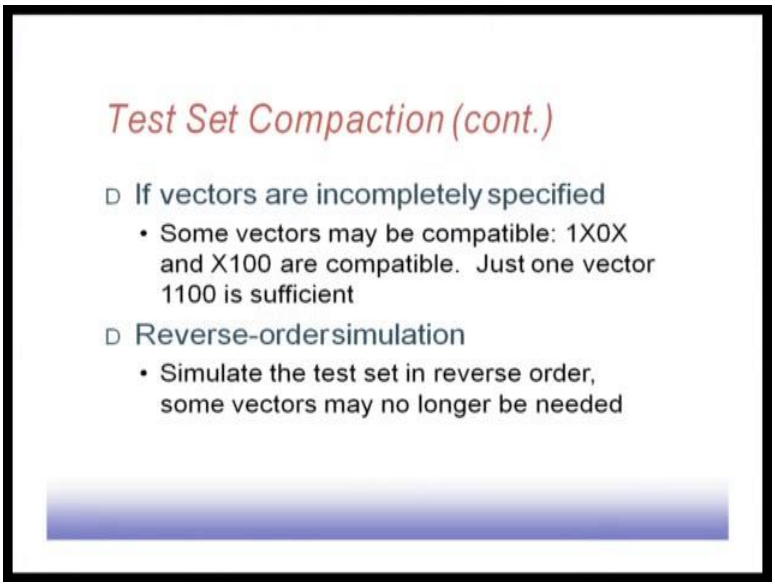
So, we include that essential vector into to our test set. So, v 4 is included in the test set. After that we can have some sort of test set covering algorithm to a find the minimum test set that every fault is covered. So, set covering algorithm is that we have got a universe U of elements; so e 1, e 2, up to e n. And we have got this individual sets s 1, s 2, s 3 etcetera which are actually subsets of this U.

Now, if there may be a say (Refer Time: 10:51) sets each s i is the subset of U. Now the question is that from this s 1 s 2 etcetera I need to select a minimum set of subsets, minimal set of subsets. Maybe I select s 1 s 3 and s 5 such that all this elements will be

covered by that. So, that is the basically the set covering problem. But the point is the set covering problem is an NP hard problem, so you cannot have the exact solution that will be give the optimum solution and the algorithm will run polynomial time; that is not going to happen is the hard problem.

But what we have is basically is the variant of the set covering problem, what the universe is the set of all faults then the subsets are this the faults covered by this individual vectors v 1, v 2, v 3, v 4. And the problem is to select the subsets, like select the subsets v 1, v 2 here from the set of v 1, v 2, v 3, v 4 so that all faults are covered, so universe has to be covered. So, if we can do this thing we can do a test set compaction.

(Refer Slide Time: 12:02)

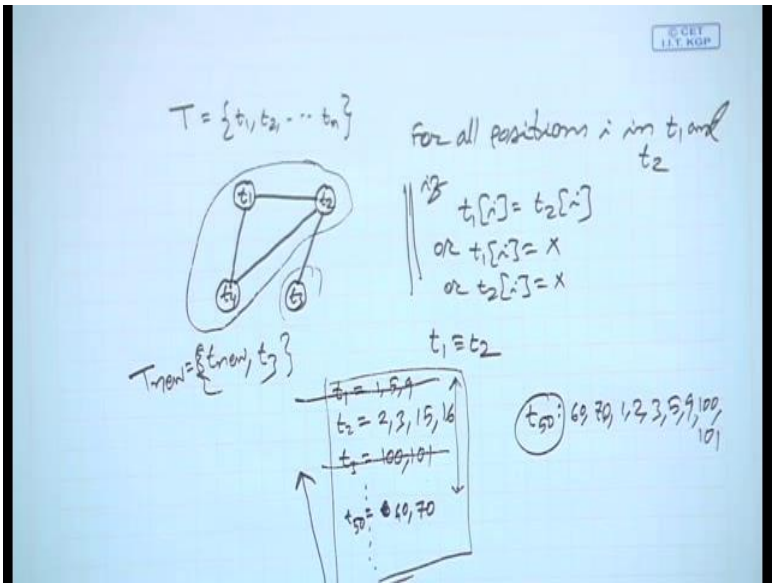


Now, sometimes what happens is that the vectors are the incompletely specified. So, some vectors may be that as we have seen that in many cases for generating say some test set pattern. Say if we want to test this line start at 1, then it is sufficient to put a 0 here. So, these lines do not care. So, the test pattern for this stuck at one is 0 x or x 0. So, whatever we do that they are basically there may be some do not cares.

So, if I have a large circuit then what will happen is for testing for the particular fault it may be required to set on the subset inputs to proper value. And the remaining inputs here and here they may be left unpassed by the test pattern. So, test pattern will keep those bits as do not cares. This do not cares can be used effectively default this compaction purposes.

For example it may so happen that some of these vectors we are compatible vectors. Means, in one vectors you suppose a 1 x 0 x and the other vector is x 1 0 0. Then what happens is that, instead of taking two vectors if I just take a single vector which is 1 1 0 0. So, what is happening here is that 1 1 0 0; so this satisfies the first vector as well as first satisfies the second vector. So, instead of taking two vectors, we can select a single vector and that is sufficient for covering both the vectors. So, faults that are detected by either of these two vectors will be detected by this new vector as well. So, that way we can reduce the size of this the test set.

(Refer Slide Time: 13:57)



So, what we can do? It is after the ATPGs given you the set of test pattern t consisting of test set t 1, t 2 up to say t n; n test patterns are there. And they have they are having do not cares. So, this t 1, t 2 t 3 and they have got do not cares. So, we can view this test pattern set as a graph consisting of this vectors t 1 t 2 say- t 1 t 2 t 3 t 4 etcetera. Then we put on edge between t 1 and t 2 if they are compatible.

So, compatible means for all positions i in t 1 and t 2; if t 1 i equal to t 2 i or t 1 i equal to do not care or t 2 i equal to do not care. If any of these three cases occurred then we say that if all positions if this thing happens then you say that t 1 and t 2 they are compactable. So, in this case we can replace t 1 and t 2 by some vector. So, what we do? We will we construct this graph or by put the edge between the vectors which are

comfortable. It may so happen that this t 2 is compactable with t 3 t 4 and t 1. Similarly t 1 is compactable with say t 2 and t 4.

Now, you see that if you do click partitioning of this graph then this becomes a click. So, these three vectors can be replaced by the single vector and this is another vector. So, ultimately I can if that new vector is say t new, so that is one vector which replaces this t 1 t 2 t 4 by properly some of the excess in them and they will gets specified. So, t new and t 3 that is the new test vector say that has got only two vectors in it. Instead of four vectors in the original set now I have a got only two vectors in.

So, this way we can have these compatibilities and accordingly we can reduce the size of this set. Another way of reducing that is test set size is by reverse order simulation. So, what happens is that while we are doing that test pattern generation; the first is the pattern that is generated it is detecting some faults as I was telling you may be 1 5 and 9; t 2 is detecting again some faults. In the ATPG process what we are saying as soon as one fault is detected by the test set by a test pattern if those faults are dropped from the candidate fault list so that successively that ATPG tool is not bothered about generating test patterns for the those faults again.

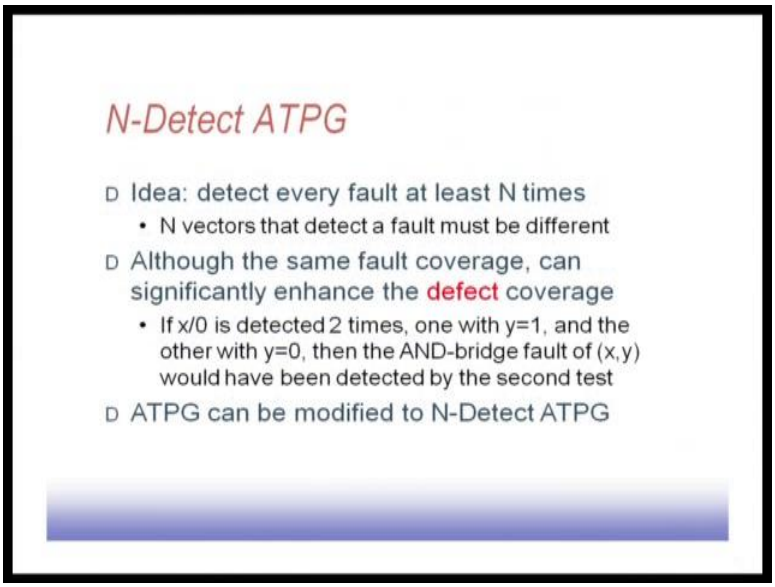
So, if you drop these faults 1 5 and 9, then this t 2 will detect some other faults. So, it may be detecting 2, 3 and 15; 2, 3, 15, 16. Then may be t 3; it will detect faults which are not presenting this set may be detects of false 100 and 101. So, that way this ATPG has been made to cover the faults. So, it has been given me this big set. Naturally, as you go down and down in this lay list last two patterns they will detect very few faults may be one fault each or something like that.

But, you see that any patterns say t 50 it has detected some false, so it as detected the faults 60 and 70. So, these are the two faults detected by this way. But the point is that at this point of time, so all these faults that have been detected by the previous patterns they have been dropped from the set of faults. Apart from this 60 and 70 this t 50 may have the potential to detect some of these previously detected false as well. So, if you do a false simulation of t 50 you may find that t 50 detects not only 60 and 70, but it also detects the faults like 1, 2, 3, 5, 9, 100, and 101; so it may have the potential to detect all this faults.

So, what is required is that if we do a fault simulation in the reverse order then if we include t 50 into the set then this pattern t 1 is not required because if 1, 5, 9, it is already covered by t 50, so this will go. And similarly t 3 having 100 and 101 this will also go. So, if you do a reverse order fault simulation, then it may be possible that we can drop some of the previously generated patterns and we can drop them from our consideration.

So, this is what is the reverse order simulation. So, simulate the test set in reverse order and some vectors may no longer be needed, so that way also we can reduce the test set. So, this is the test set compaction.

(Refer Slide Time: 19:13)



Another important issue the ATPG is the N-detect. So, N-detect tells that sometimes it is necessary that you detect every fault N times. So by default N is equal to 1. So, once a fault is detected by a test vectors it is no more considered for a test pattern generation by just test pattern generation tool. But suppose you make N equal to 2; that means, the first time is fault is detected it is not dropped from the set of considered candidate faults, but if it is detected by one more pattern it will get dropped from the set.

So, this idea is that detect every fault at least N times. So, N vectors that detect a fault must be different. So for every fault now I have got N different vectors. So, the fault coverage remains same, but it can enhance the defect coverage. So defects coverage why, because see suppose x stuck 0 is detected two types: one with y equal to 1 and the other with y equal to 0. Then this n bridge fault of x y would have been detected by the

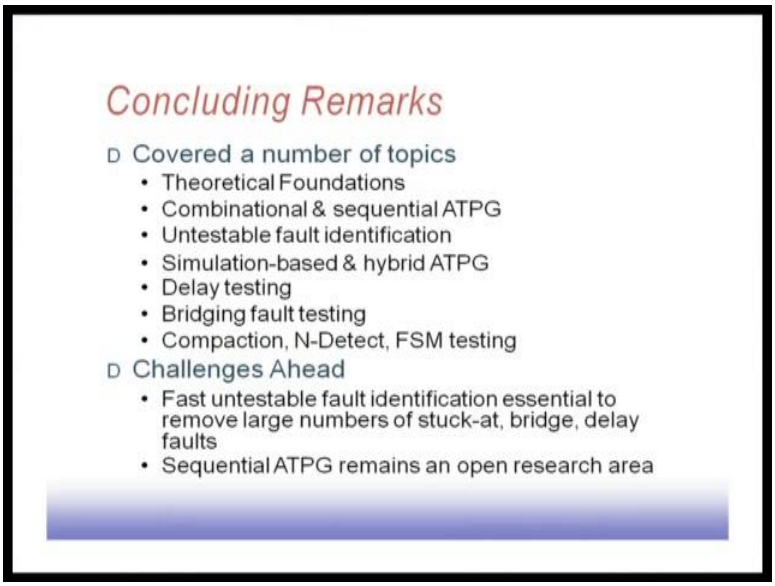
second test. So, this is if detect this is detected two times: one with y equal to 1 and then y equal to 0; that means we can be sure that there is the bridge fault between x and y. So, for bridge fault detection this may be one issue.

If we are trying to get some confidence in the diagnosis process ok: if we are instead to get a diagnosis process then you get some surety that this particular fault has occurred, this particular defect has occurred. Then if a fault is detected many times by the applied test patterns so you get confidence is the number of times a fault is detected gives as the a surety that this particular faults has occurred in the circuit; so that may be an issue. So, that way this N-detect is used.

Also there may be some other points like; suppose we want to generate a test pattern set it consumes a less power. In that case also what we can do possibly, we can generated redundance set of test patterns where every fault is detected by many patterns and then try to do a pick and choose. So, while doing this pick and choosing it pick up those vectors were that coverage is high at the same time the power consumption will be low. So, taking these two into consideration we can have this pick and choose type of approach.

And in that case also these N-detect is going to be help us, because it is going to be give us an ATPG for N patterns for every fault. Now an ATPG can be very easily modified to work as an N-detected ATPG. So, what is required is that you just for change the fault dropping criteria.

(Refer Slide Time: 22:10)



So, to conclude: we have seen in this test pattern generation as a some theoretical foundation in terms of this Boolean difference and all that. We have seen combinational ATPG, sequential ATPG we said that after the scan convection their process sequential ATPG it is rarely use accepting some partial scan based circuits, but mostly it is combinational it that is coming into picture.

Then we have seen how to identify the untestable faults. So, there are the untestable fault identification can help this ATPG process, because it will be having less number of faults to up to. Then we have seen some simulation based techniques. And there are some hybrid ATPG as well which actually talks about this analog and digital combined ATPG. And this simulation and also we have seen that simulation based ATPGs, so they have been the clubbed with this deterministic initial pattern generation; the initial population generation so that is modified so that, that is also hybrid ATPG.

Then for delay testing we have seen something: we have seen some techniques for delay fault generation. Then bridging fault testing also we have seen, then compaction N-detect how to compact it is set then N-detect how to detect the fault N times, if a some testing is a part of sequential testing. So, we did not discussed in detail, but essentially one possibility is that first we apply a sequence of patterns so that this FSM goes to particular state and in that particular state we want to test a fault, so then we apply some long chain captured pattern for testing that particular fault.

So, the challenges that are we remaining is the fast untestable fault identification essential to remove large number of stuck at bridge and delay faults. So, this is one of the very important issues, because with the increasing the complexities of the circuits under number of faults in the circuit are increasing tremendously. So, if you can identify untestable faults early then we do not put effort on doing them so that way it can reduce the number of faults to be detected.

So, this is becoming even more challenging; so identify the untestable faults and doing it fast. Sequential ATPG is still an open research area because of the fact that this initialization itself a problem. But anyway with the introduction of this scan circuits and all that this sequential ATPG problem is addressed as a modified combinational ATPG problem, but still in some cases sequential ATPG will also be required. And other issues that are there is how to generate test patterns say which can be reduced power consumption, which can be reduce, that temperature of the circuit. So, those issues will see in the later part of the course.