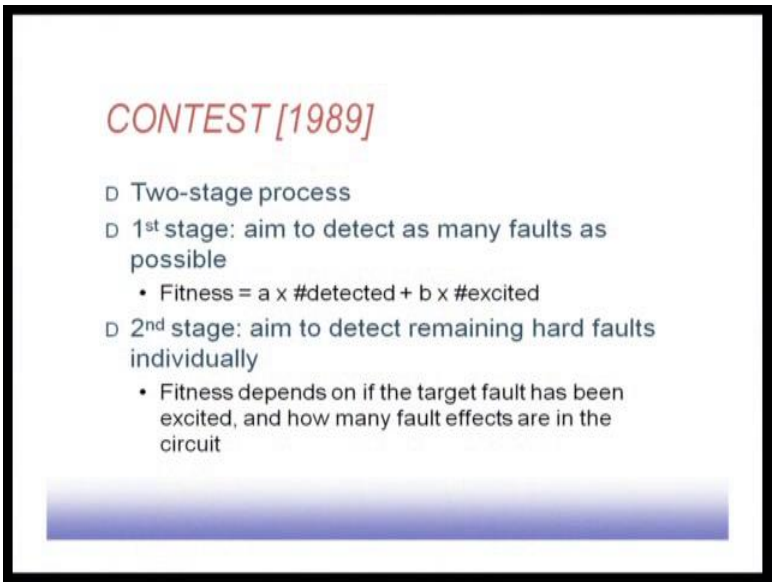
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 22
Test Generation (Contd.)

Next we will look into some of the tools that have been developed using this simulation based ATPGs approach.

(Refer Slide Time: 00:21)



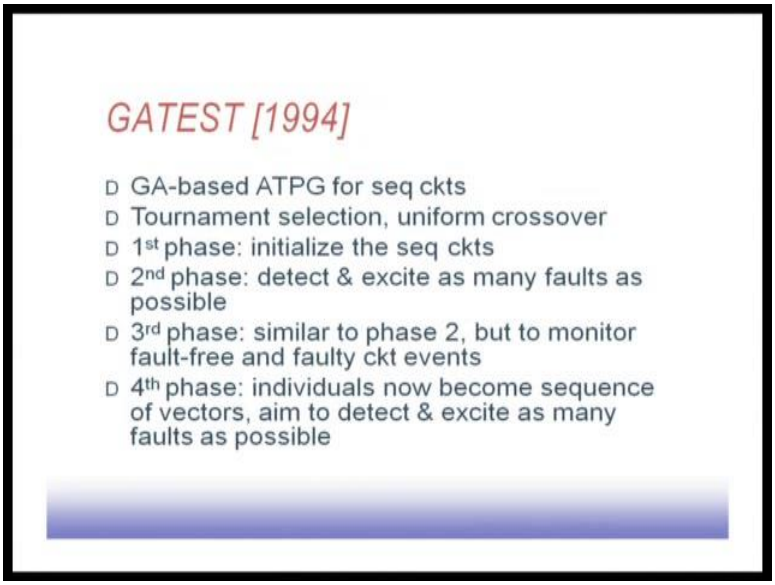
The first one is contest it was way back in 1989. So, it uses a two stage process: the 1st stage it aims to detect as many faults as possible. So, naturally fitness function is given by a into number of faults detected plus b into number of faults excited. So, this is the fitness function for the individuals that we have. So, that was used. Initially since we are trying to detect as many a faults as possible so that was the fitness function. So, it does not talk about the faults which are detected multiply by different test vectors; any individual maybe detecting large number of faults, but they might have already been detected by some other patterns as well. So, these are the problem.

The 2nd stage; so at the first other end of first stage we will achieve some amount of some data fault coverage, but there will be a number of faults which are not detected because they are hard to detect faults. So, the second stage it will aim to detect the remaining hard faults individually. Now the fitness at this stage you will depend on if the

target fault has been excited and how many faults effects are in the circuit. So, this is the fitness function for the second stage.

So, it is targeting only the hard faults; targets one fault and try to generate individual in that direction. Whether the target fault has been excited and how many fault effects have been excited in by the pattern. So, that way we get the fitness of these individuals.

(Refer Slide Time: 02:03)



Another one GA test which was reported in 1994; so this GA base ATPG first sequential circuits. So, it uses tournaments selection, uniform cross over that we have discussed previously.

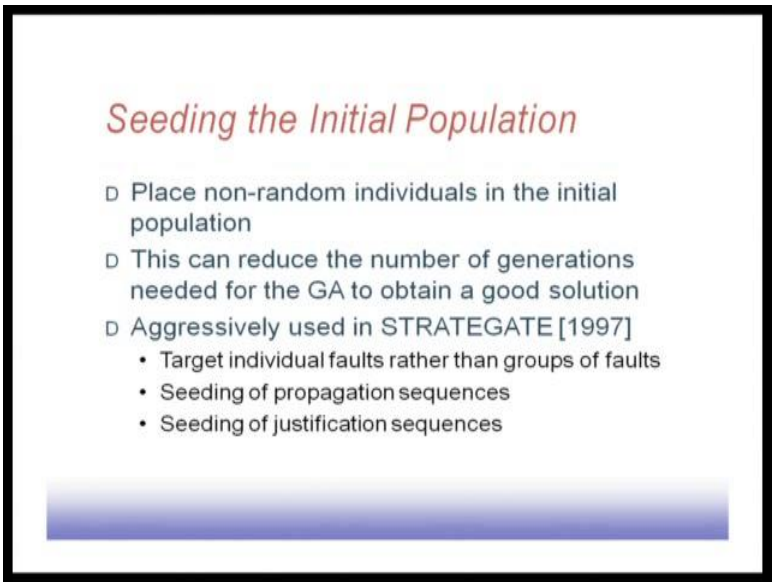
The first phase it talks about initializing the sequential circuits, second phase it detect and excite as many faults as possible, third phase similar to phase two but to monitor fault free and faulty circuit events, and the four phase the individual now becomes consequence of vectors and now aim to detect and excite as many faults as possible. You see that the sequential circuits. So, first sequential circuit initialization that is a big challenge; so the set of patterns that can initialize this sequential elements to some values, so they are important. So, what are the patterns that can initialize these sequential elements to some distinct values? So that is the fitness function of the first stage.

Second phase the objective is to detect and excite as many faults as possible. This is similar to the previous one of this previous test pattern generation we have talked about.

The third phase it will monitor fault free and faulty circuit events. So, how many faults are new faults that I am getting detected, and how many faults, what are the fault free operation, and what are the fault free circuit operation, so how many of them a different, though that comes as the cost function.

And a fourth phase now for sequential circuits you need to apply a sequence of vectors. So, we try to frame the sequence now individuals they are sequence of vectors. For a particular sequence what are the patterns, what are the faults that it can detect and excite. So, this is the fitness function. So, actually in the fourth phase is does the sequential ATPG, in the previous stages it is similar to the combinational ATPG.

(Refer Slide Time: 03:57)



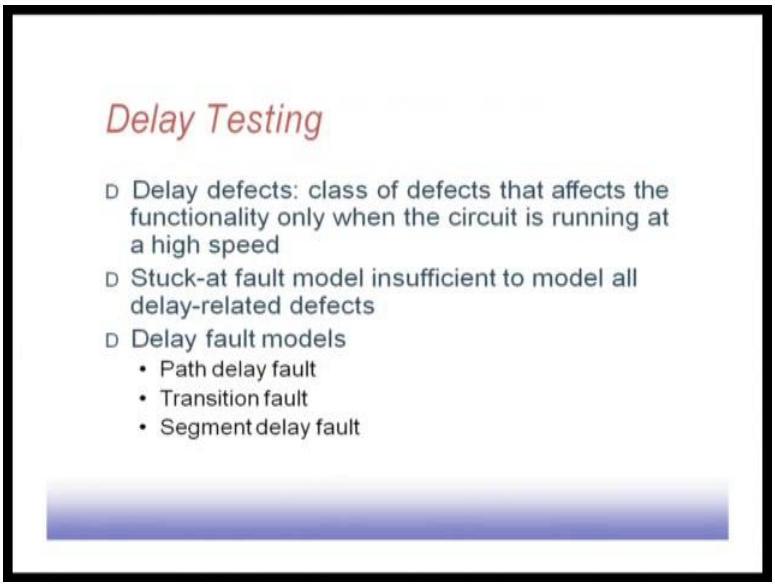
There are some approaches to improve the quality of this solution; this GA solution produced. One possibility is to seed the initial population. So, what we have said previously is that the initial population is generated randomly, but random initial population generation so the problem is that it may not have many good solutions in it. As a result the evolution becomes a question. So, that way under evolution it may not generate a good number of individual, good types of individuals of different varieties. So, what is required is that apart from these random individuals we also need to put some non random individuals in the initial population. So, what can we do? We can use some other algorithm to generate some non random patterns, and those non random patterns they also become part of the initial population.

So, remaining patterns are generated randomly, but these non random patterns are generated by some other say deterministic algorithm. So, this can reduce the number of generations needed for the GA to obtain a good solution. So, convergency rate will improve definitely. And this has been used by this particular tool called Strategate, so it targets individual faults rather than group of faults and seeding of propagation sequences and seeding of justification sequences. So, what it does is that for a particular line to be justified to be put say particular line you want to get the stuck at 0 so that line has to be set to 1. For that purpose what is the input sequence.

So, that input sequence is made part of some individual for solution. Similarly for propagating one fault with the primary output some primary inputs are required to some required to be said to some value. So, that particular setting becomes part of some individual so in the individual that the chance of detecting that fault become more. So, that way those type of verities are introduced into the population so that this generation and propagation. So, these are facilitated within the individuals in the population.

And then when it evolves then possibly this will remember that this will be able to capture this intelligence; that this way I can propagate the fault on this way I can excite the fault. So, that is actually taken care of by this process.

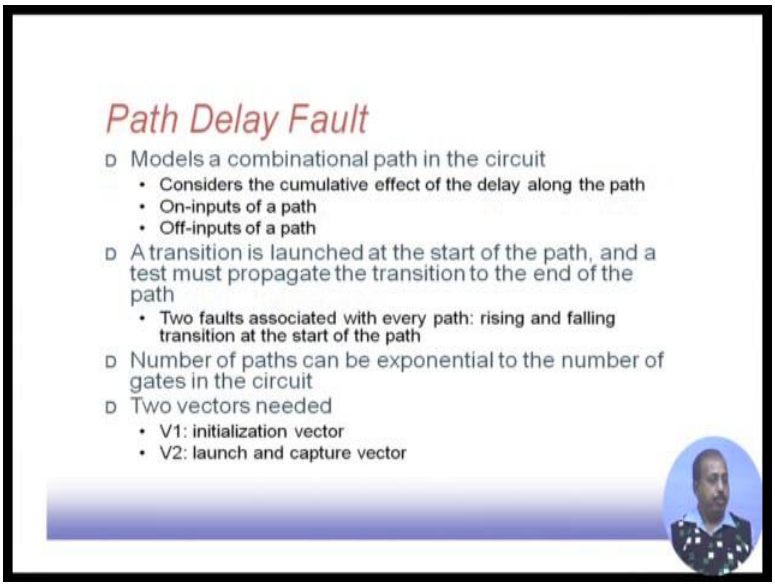
(Refer Slide Time: 06:38)



So, apart from this random testing this feeding with these non random patterns this helps a lot in the evolution process. Then for delay testing; so we can have this delay defects,

but this stuck at delay defects are faults that are occurring that effect the functionality when the circuit is running at a high speed. So, this is there because this is that delay of various parts. So, stuck- at fault model is in sufficient to model all delay fault models and faults delay fault model we know that there are several model that a path delay fault, transition fault, segment delay fault etcetera. But this type of simulation based ATPG, so they are not very good for doing this delay testing.

(Refer Slide Time: 07:25)



So, actually for delay fault testing we need to have a different type of test pattern generation. First will look into the path delay faults; so path fault it models a combinational path in the circuit and it considered, so any you find out a combinational path in the circuit and then considered the cumulative effect of delay along the path. So, what is the total delay? If there is a transition at the input of the path then how much time that transition takes to propagate to the output. So, if it is within a reasonable delay then it is reasonable time then we say that the path does not have any delay fault. But it is not within that if the transition reaches after that then we say that there is a delay fault and we can find a particular input pattern that can excite that delay fault and we can see the delay fault at the output.

Now, for a path so there are set of on-inputs that is the inputs which are affecting the gate switch are on the path and there are some off-inputs of the path so they are not affecting the part they are feeding the other inputs. A transition is launched at the start of the path

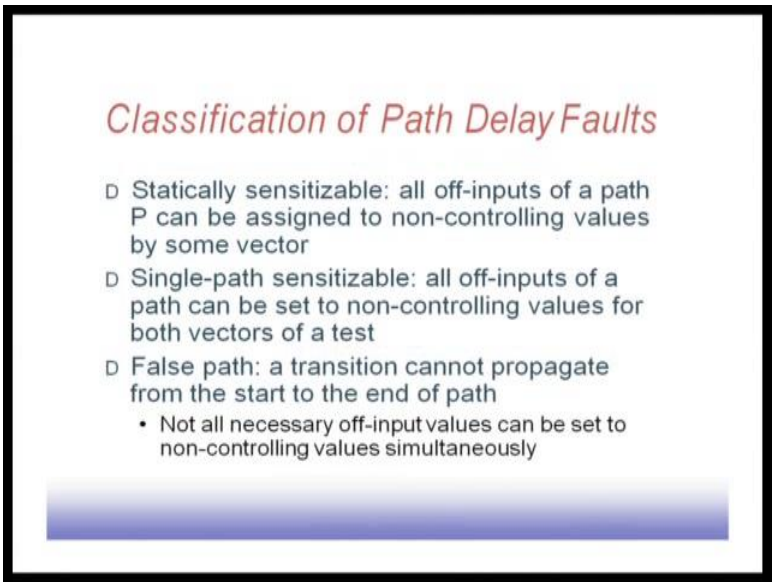
and the test must propagate the transition to the end of the path. So, that is what I was talking about. So, transition will be launched at the start of the path and it must come to the end of the path and two other inputs are to be said to some proper values to facilitate this process. So, they must be put to some value so that this transition propagates through the path to the output.

So, there can be two types of faults associated with every path: one is called rising transition, another is falling transition. These are the two types of delays that can occur; rising delay and falling delay they are normally not same for any logic gate so they are to be taken separately. Now the problem is that the number of paths can be exponential to the number of gates in the circuits. From the primary output if you back track and try to look to the primary input so they are innumerable number of paths. And as the size of the circuit grows number of paths will also be very high.

So, even if particularly if there is the fan out in the circuit somewhere in between then the number of paths will increase even further. So, that way there is normally this number of paths is going to be exponential to the number of gates in the circuit. So, we need two types of two vectors for testing this individual fault. One is called initialization vector and the other is called launch and capture vector. So, initialization vector you will set the, so if you have trying to for example trying to check a rising fault at some point then first of all the point has to be set to a low value or logic 0 value.

So, that requires the initialization. And after that we have to give input so that this particular point sees a transition from low to high, so that is the launch and capture vectors. So, we need two vectors for testing any path delay fault: one is the initialization vector another is the launch and capture vector.

(Refer Slide Time: 10:38)

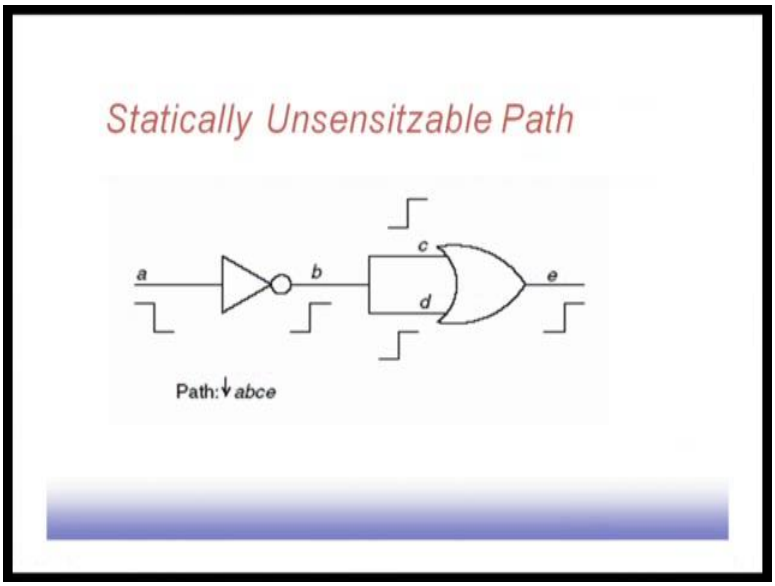


Now this path delay faults they can be classified into three categories: one is called statically sensitizable path, another is called the single path sensitizable faults, and another is a faults path. So, in sense is statically sensitizable case all off-inputs of a path e can be assigned to non-controlling value by some vector. So, what it says is that the apart from the paths of all the inputs which are not affecting this path they are off-inputs of the path so they can be said to some non-controlling value so that they do not affect the propagation of this transition from transition point to the output.

So, they can be said to some non-controlling values. If you can do this thing then such a path will be called statically sensitizable. Single path sensitizable: so it says that all off-inputs some a path can be said to non-controlling values for both vectors of a test. So, they are for both the vectors we can say the values, so that this off-inputs can be said to non-controlling value. And there is a false path if a transition cannot propagate from the start to the end of the path. Naturally, if it requires conflicting setting for the primary inputs for the transition to propagate from start to end so that is a false path.

So, not all necessary off-input values can be set to non-controlling values simultaneously. In this case the path is a false path, so this path cannot be tested for the delay fault.

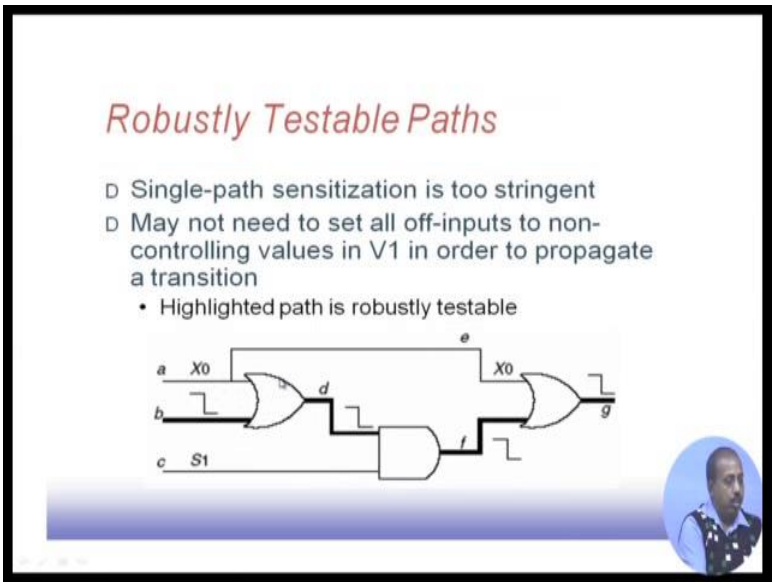
(Refer Slide Time: 12:11)



So, this is statically unsensitizable path, because this path a, b, c, e, we want to said the value so we are we are trying to for check for this transition that is the high to low transition for a, b, c. So, this is making this transition high to low, but for making this high to low these are the respective transition. So, this is unsensitizable because for whenever this transition occurs, so all the inputs all the inputs are also making transition.

We cannot make other inputs to some proper value, so that these transitions can proceed.

(Refer Slide Time: 13:02)

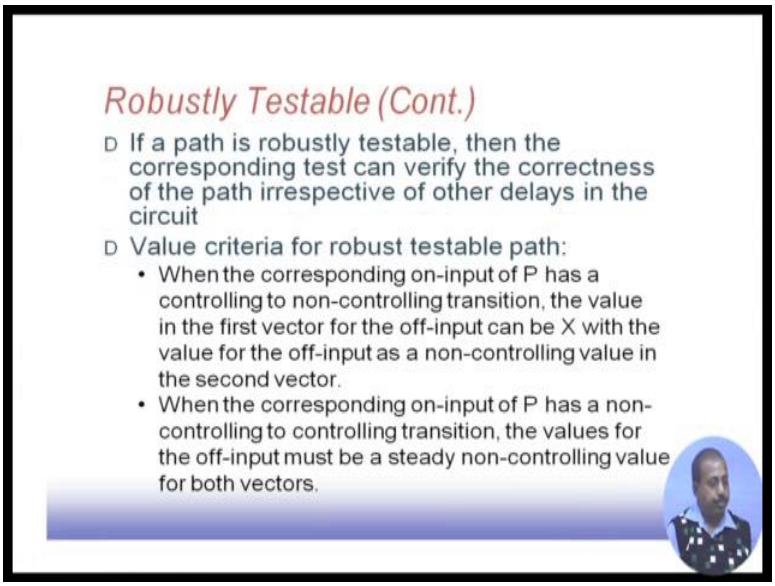


Then there is a robustly testable path. So, single path sensitization is two stringent because single path sensitization tells that all off-inputs are to be set to some proper values, but robustly testable path says that we may not need to said all off-inputs to non-controlling values in V 1 in order to propagate a transition. So, you see that here I am trying to test this particular fault also this is line b for this path, so it is going for a high to low transition.

But when it is going from high to low transition initially I have to set the line to high. When I set the line to high, I can with this line a and this line a need not be said to any values these can be x. Whereas, this line c has to be said to 1 always, so this is a solid one or fixed one that will be required. But this a line did not for V 1 the first vector this b line is say 1 c line is 1, but this a line need not be set to any value. So, this does not require setting any value. So, that gives us flexibility.

So, second vector for the transition vector of course, we have to have this a value also said to 0 so that will be required, but for V 1 it is not required. So, this type of situation this is known as robustly testable path. So, we do not need to set all the off-inputs in the V 1 vector as well.

(Refer Slide Time: 14:36)



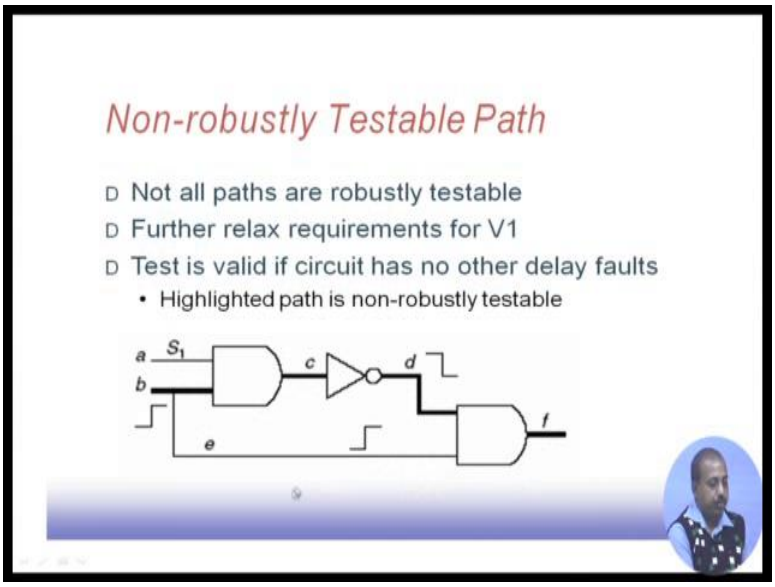
So, if a path is robustly testable than the corresponding test can verify the correctness of the path irrespective of other delays in the circuit. This is a result that can be proved that it is a irrespective of other delays in the circuit, so we can test this path. Now, what are

the value criteria? When the corresponding on-input of P has a controlling to non-controlling transition like say this is an OR gate, so when the OR gate is 1 input is 1 so output is controlled by that input. So, it is having a controlling to non-controlling transition you see.

So, what it says- if it is have a controlling to non-controlling transition the value in the fast vector for the off-input can be x; with the value for the off-input as a non-controlling value in the second vector. So, what it says? The value of the off-input can be x for the first vector. So, you see for the first vector; for the first vector I have got the other value set to x. And it says that in the second vector the off-input should get a non-controlling value. So, you see for the off-input for the second case that is for V 2 b is equal to 0 and this should also assume a non-controlling value, so this has got a 0. So, this is one rule, value criteria for robustly testable fault.

Similarly, when the transition is non-controlling to controlling transition; so it is for OR gate it is going from say 0 to 1 so that type of transition. In the values off-input must be study non-controlling value for both the vectors. So, this case is that this is one AND gate. So, what is happening is that it is going for a transition 1 to 0. So, non-controlling to controlling transition is there for these get. So, the seconds rule; the input as a non-controlling to controlling transition in it says the values off-input must be steady for both the vectors.

(Refer Slide Time: 16:43)

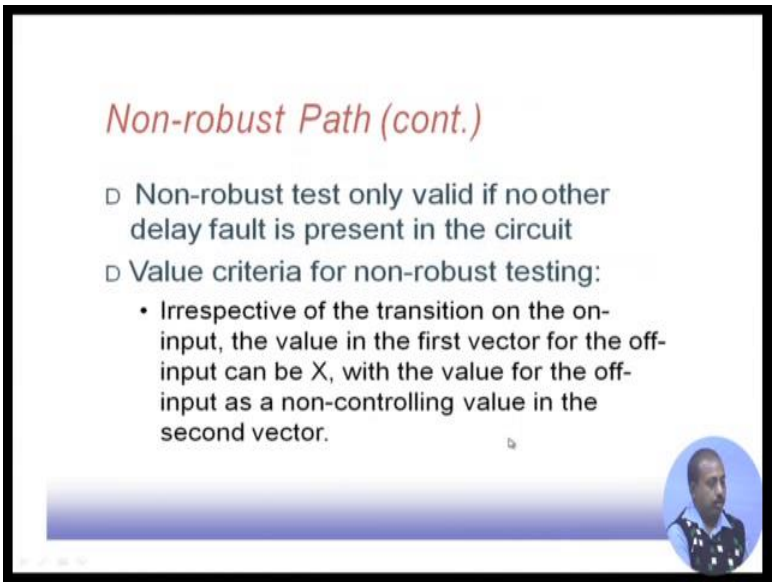


So, steady non-controlling value, sorry. So, this c input for both the cases it is held at 1. So, for and get 1 is a non-controlling value, since it is making a non-controlling to controlling transition I should maintain the other input a steady one for both the vectors. This gives us the rules by which we can assign values to the input lines or input lines of various gates so that this the fault can propagate.

There is another class of path which is known as non-robustly testable path. So, not all paths are robustly testable; so it is not that we can do the robust path for all the cases. So, we can relax the requirement for V 1. We say that the test is valid if circuit has no other delay faults. If there is no other delay faults in the circuit then even for non-robust path testable path also we can generate this is pattern. So, this is a path which has got for which we are interested. And this circuit does not have any other delay faults. So, you are interested only in testing this particular delay fault.

In that particular case we do not need to consider other cases, we can we can apply this transition from low to high here. And this a line remains as study one. So, this is controlling to non-controlling transition, so this line remains at study one and accordingly it propagates to the output.

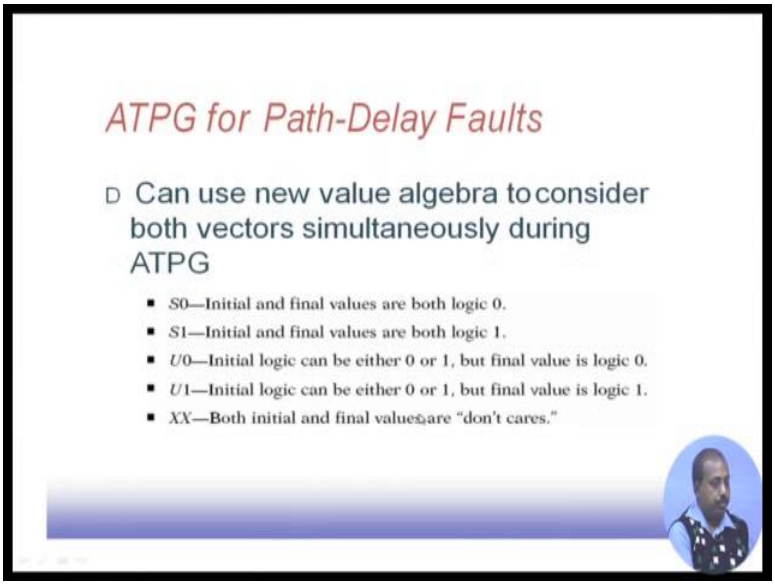
(Refer Slide Time: 18:22)



So, non-robust test only valid if no other delay fault is present in the circuit. And value criteria: irrespective of transition on the on-input the value in the vector for the off-input

can be x with the value of the off-input as a non-controlling value in the second vector. So, first value can be x and second value has to be non-controlling value.

(Refer Slide Time: 18:49)

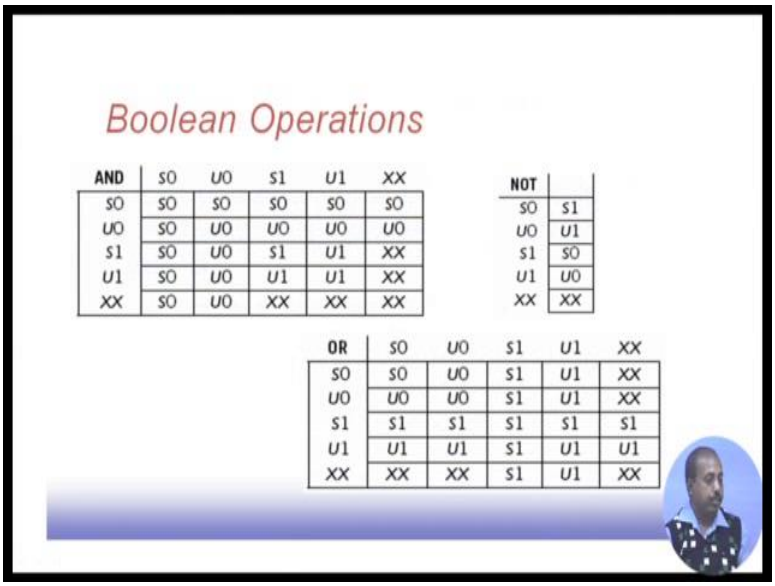


Now, for the having this ATPG for this path delay faults: how to generate this path delay faults? We can use some algebra. So, this algebra it consists of several new symbols. Like S 0. So, S 0 we say that the initial and final values are both logic 0. Basically, what we are trying to do is we are talking about two vectors: the first vector, the exciting vector and the second vector is the launch and capture vector.

So, this we can talk about that initial, so the S 0 both in both the vectors the value is at logic 0. Then S 1 both initial and final values are at logical 1. U 0, initial logic can be unknown it can be either 0 or 1, so it does not matter. But the final value must be 0. Then U 1 initial logic can be either 0 or 1, but the final logic must be 1. And xx both final and initial and final values are do not cares.

So, we can use this algebra to consider both these vectors V 1 and V 2 simultaneously for generating test patterns through the ATPG technique.

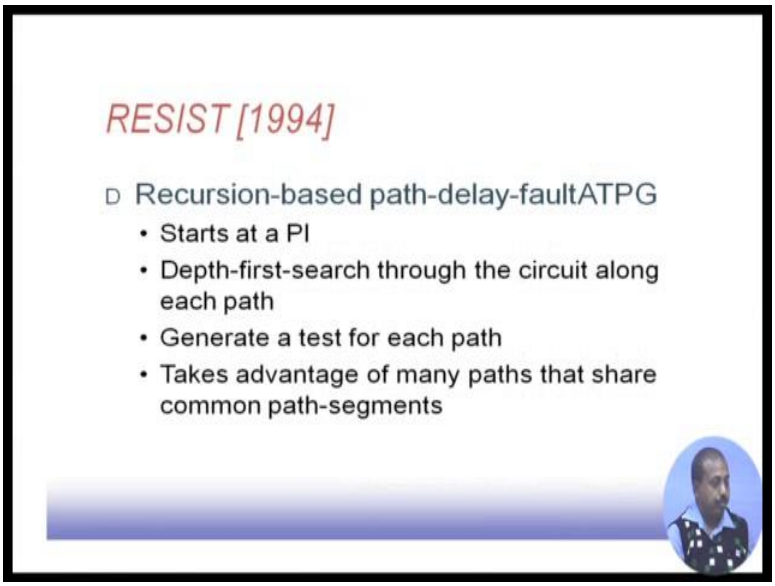
(Refer Slide Time: 20:02)



Now, we can define the Boolean operations over this algebra. So this AND: so S 0, S 0 whenever it is static it is fixed at 0. So, both S 0 means both are fixed at logic 0. So, you see that whatever be the other input this will remain as S 0. So, this always remains 0. Similarly U 0: so it this is first point it is unknown and next it is 0, here first it is always 0 so this will be S 0. Similarly unknown 0 this will be unknown otherwise this is these are all unknown 0.

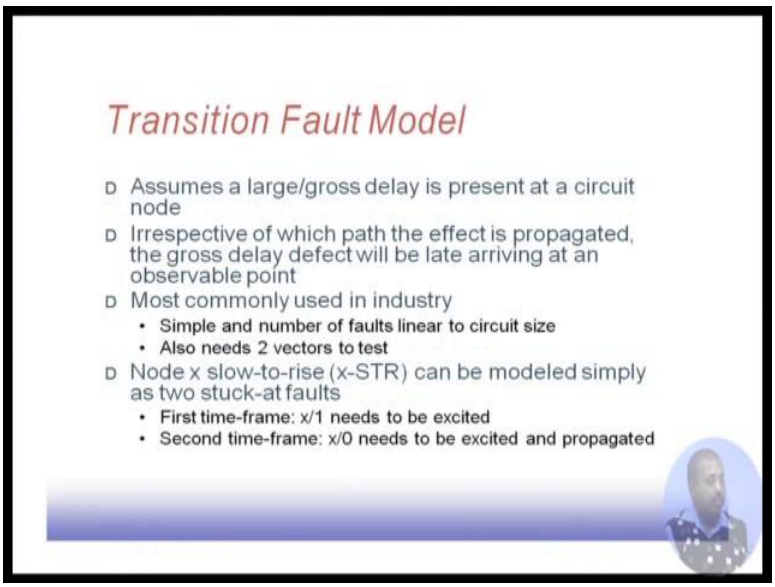
Similarly, S 1 and S 0: this will be S 0. So, you can find out this particular Boolean operations, AND operation, OR operation by taking this individual vectors separately. So, you can define what will be output of this and or not operations.

(Refer Slide Time: 21:04)



Now, there are several algorithms for these it delay fault ATPG. There is a recursion based path delay fault ATPG resist. So, starts at a primary input and then it performs a depth first search to the circuit along each path, and for each path it generates a test. So since it is doing a depth first search so many path segments may get marks after sometime because of this fan out and all that. Naturally it saves in that path. So, up to that point it has got a test pattern and that is that can be extended for them. So, this take over recursion based path delay fault ATPGs have been reported.

(Refer Slide Time: 21:43)



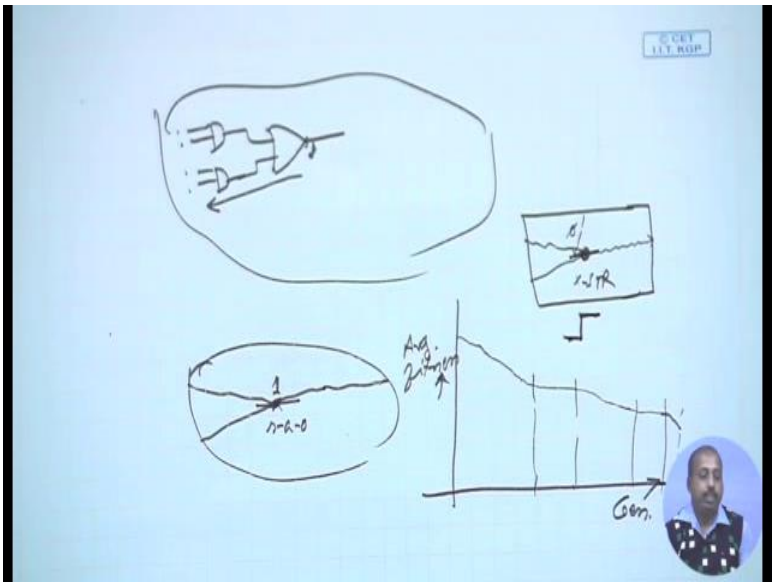
Another type of delay fault which is quite common and it is mostly used in industry also is the transition fault model. So, what it assumes is that there is a large or gross delay present at a circuit node. One particular circuit node has got very large delay. So, what happens is that if one particular node has got a very large delay then the other points, other delays in the circuit they become insignificant.

So, as a result with the only if whatever effect comes. This is basically due to the large delay that is present with that particular node. So, irrespective of which path the effect is propagated, the gross delayed effect will be late arriving at an observable point. So, basically that is going to happen. Most commonly used in industry; simple and number of faults linear to the circuit size, because if they if there are n nodes in the circuit then all these n nodes may have some large delay. As a result there are n faults. And this also needs to vectors to test: one is called node X slow to raise X-STR slow to raise another is naturally slow-to-fall.

So, this can be modeled using two stuck-at faults. In the first time frame you want that x should be at 0 k, so we want to put the line x 2 0. If we have a stuck-at fault generator ATPG for stuck-at faults then we can ask that ATPG to generate test pattern for x stuck at 1. So, it will put the line x to 0 my first part is solved. Now in the second part: now this x value should become 1 and that can be done by applying a test pattern with this which can detect the fault x stuck at 0, because it will try to put a one at that point so this x stuck at 0.

So, first time frame we apply the pattern corresponding to x stuck at 1 and the necessity is that it must excite the fault. So, we do not need to propagate the fault to the output at this point of time. However, in the second time frame I want that x stuck at 0 so that fault has to be excited and propagated.

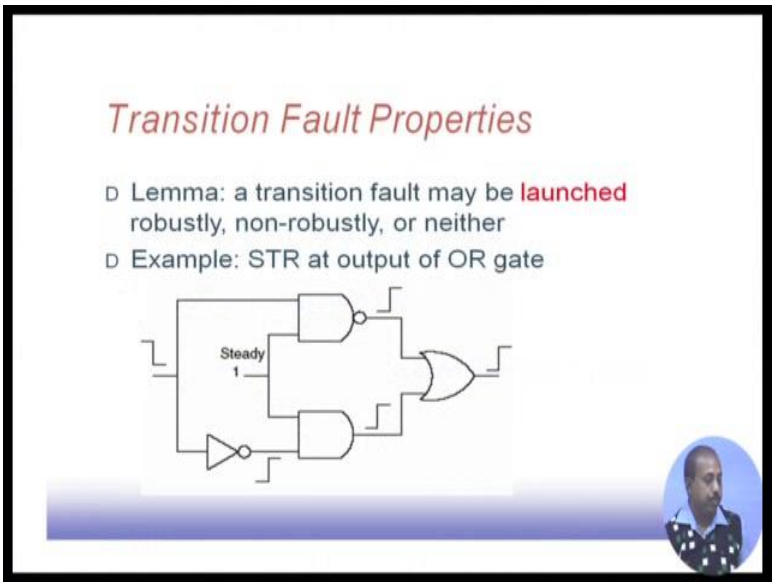
(Refer Slide Time: 24:20)



Because, if this is the circuit and at this point I want to do that X-STR slow-to-raise; so initially from the primary input I have to do something so that this point becomes equal to 0. This effect need not be propagated to the output, because I do not need to see the output now, but after that I want to put this line high. I want to low to high transition, so initially I have made the line low and now I need a transition to go to high. So, I need to apply a pattern so that this line is forced to 1; this line is forced to 1.

And in that case I would like to propagate this effect to some primary output so that I can see that; what is the delay in the process? So, by using this ATPG for stuck-at faults we can generate this transition fault patterns as well. So, we transition fault maybe launch robustly non-robustly or neither. So, this is typical result.

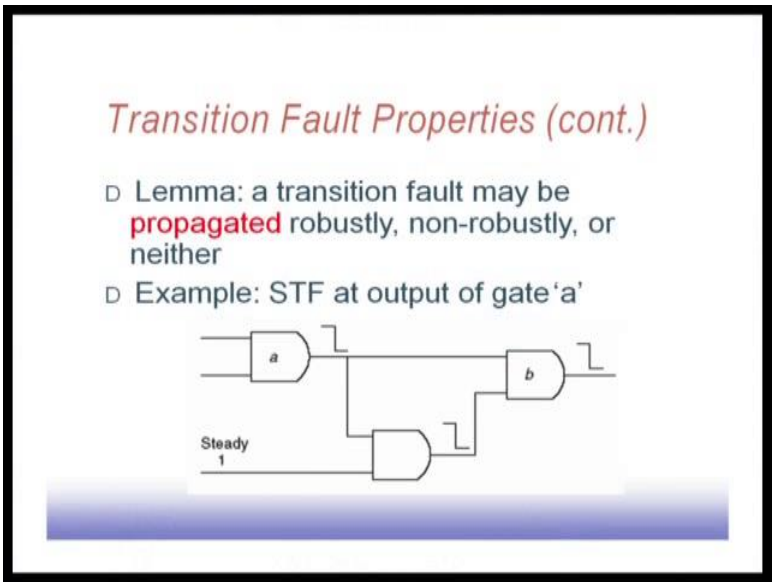
(Refer Slide Time: 25:16)



So, here there is an example of this slow to rise at the output of OR gate. So, this OR gate output, so this is slow to raise fault by one to be capture. So, for that pattern you see that we can we apply some ATPG so that first it makes this line stuck at 0, accordingly the test pattern is generated which will make this line high and this line will be maintained at 1; these the two primary inputs.

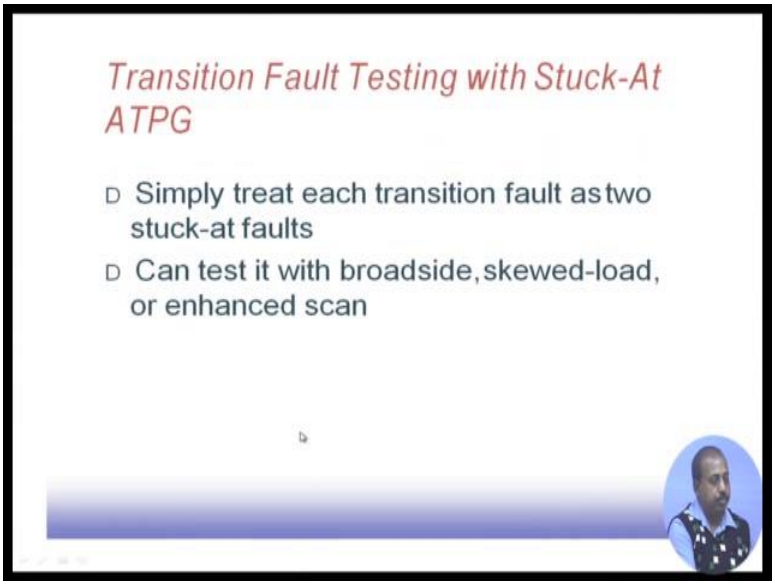
And after that I will run this ATPG for generating a stuck at 0 fault at the output of the OR gate. And the stock at 0 fault it will try to put a 1 at this point accordingly it will make this line low and this line will continue to be 1; so this way using this stuck-at fault ATPG we can generate the test pattern for transition faults.

(Refer Slide Time: 26:16)



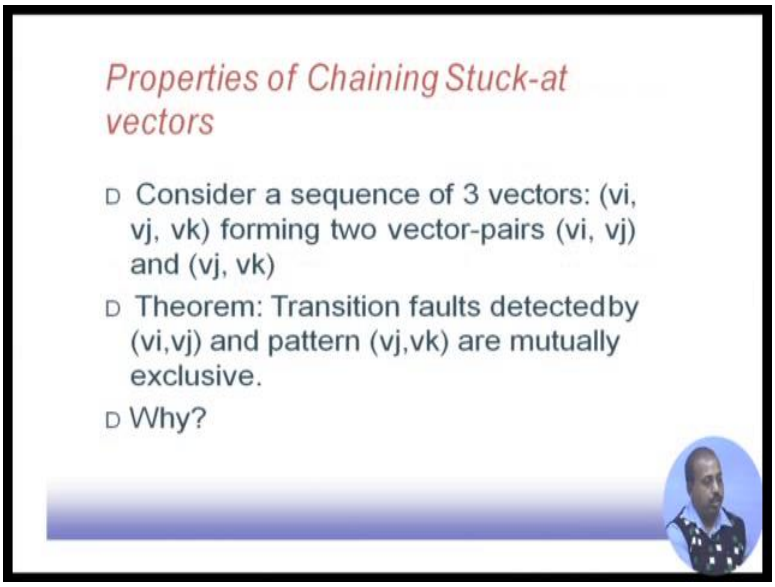
A transition fault may be propagated robustly non-robustly or neither so that is also another property. So, slow to fall at output gate a, so this is basically say this output, so this slow to fall we want to get first we make it 1 and then we try to get a 0 there. And then by using the similar method that we have discussed with that can. So, it is propagated basically in a robust fashion where this path has been put to some proper value.

(Refer Slide Time: 26:52)



So, transition fault testing with stuck at ATPG. So, simply treat each transition fault as two stuck-at faults or it with in a broad sides, skewed load or enhanced scan. So, there are various types of this launch capture methods by which we can do this scan testing. So, all of them can be used for the transition faults as well.

(Refer Slide Time: 27:17)



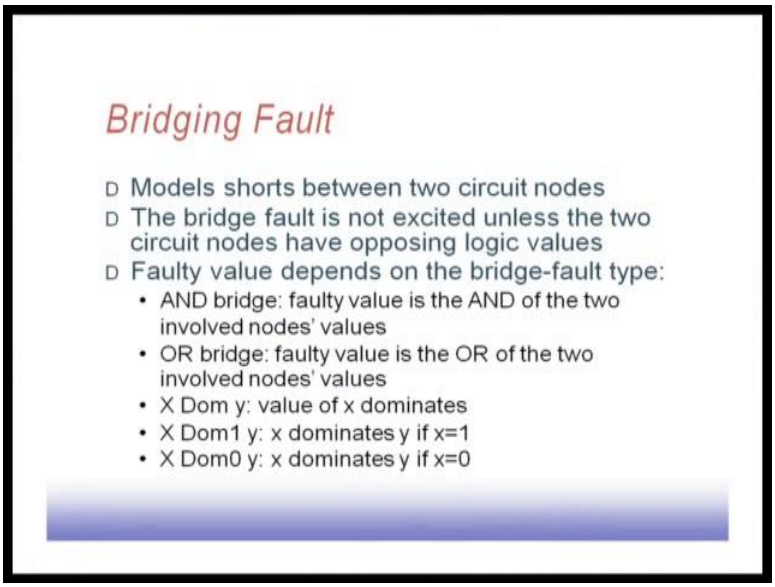
There are certain properties of this stuck-at fault chaining, so there is chaining of this stuck at vectors. So, suppose I have got three vectors vi, vj and vk. And out of them we form two vectors: vi vj and vj vk. Then we can come up with a result that says at the transition faults detected by vi vj and vj vk are mutually exclusive. Why, because vi vj any transition fault any says slow to raise or slow to fall transition fault this will put vj the vj will take it to some value in the first pattern.

In the second case if it also did the second person also detect this fault then the requirement of setting of that particular line by vj in this and this, so they are contradictory. Because if it is a say a slow to rise type of faults that is detected then vi has put it to 0 and vj has put it to 1. Now if this pair also detects that particular fault then vj will put put the line to 0 and vk will put the line to 1. So that is the contradiction like here vj is making it 1 here vj is making it 0. So, that is a contradiction, so it is not possible.

So, any pair of vectors that will always detect some mutually exclusive set of transition faults. If we have got a set of if you have got a sequence of these vectors for this stuck at

vectors then you just make two-two pairs; two consecutive once makes a pair also. That way if you do it then you can detect a large number of transition faults as well because each of each pair will detect some mutually exclusive set of faults.

(Refer Slide Time: 29:08)



Next that is bridging fault, we will do it in the next class.