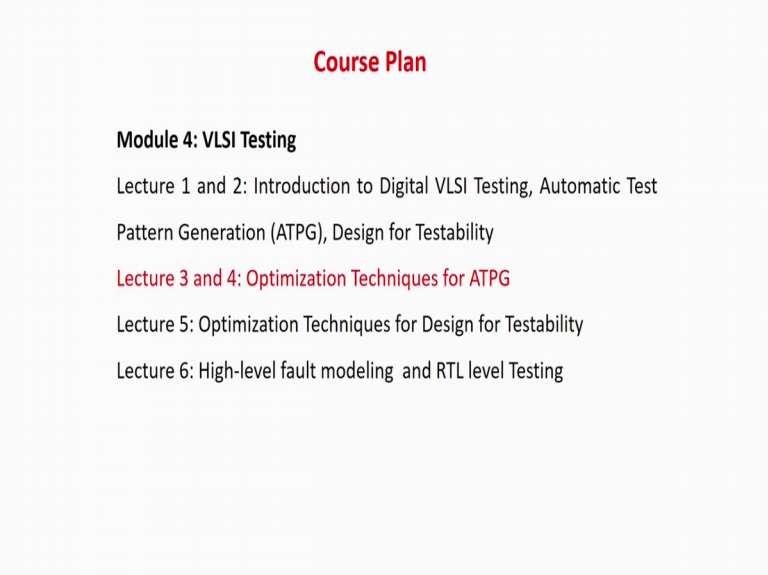
Optimization Techniques for Digital VLSI Design

Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 15
Optimization Techniques for ATPG

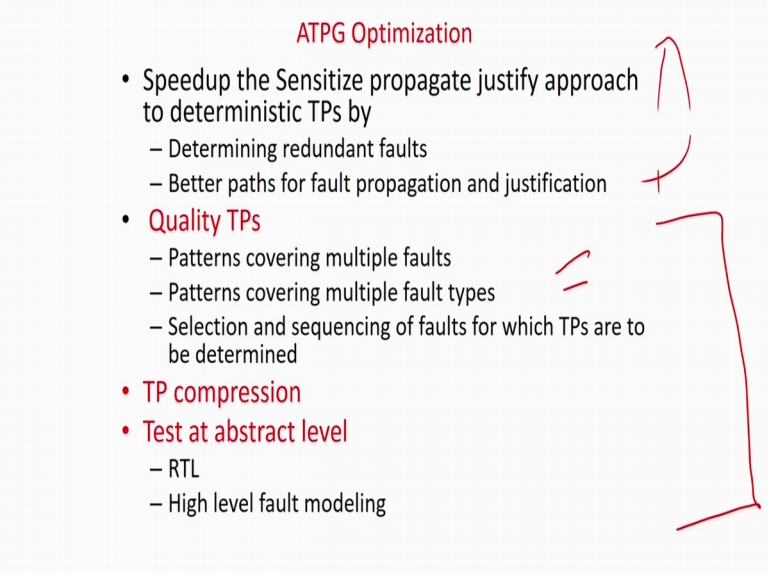
Welcome to the second lecture on the module on VLSI testing. So, in the last two lectures basically we tried to give you an overview of digital testing, automatic test pattern generation, scan chains etcetera; which is are actually prerequisite to understand the optimization techniques for testing which is applicable for circuits of higher complexity like NOC and SOCs.

(Refer Slide Time: 00:36)



So, basically as we are throughout discussing in the course that basically we have first covering the basic principles of digital design, which are applicable for slightly mediocre level of a VLSI circuits and then we are moving into some of the techniques which are required for modern day complex circuits. So, in that series on testing now we will be covered starting with optimization techniques for automatic test pattern generation.

(Refer Slide Time: 01:11)



So, as already discussed optimization in the in the context of this course is look at the aspects which are required for modern day complex circuit which cover systems of systems like system on chip NOCs multiple course on a single die something like that.

So, basically if you see, what are the steps for automatic test pattern generation optimization. So, one of the basic default idea will be the speed up the ATPG approach. So, as I have already discuss in the last two preliminary lectures that test patterns can be generated by two methods, one is the test for simulation method in which case we give a pattern a random pattern and try to see how many faults get covered by the method.

Basically, we cover up all the easy faults and then we go for basically testing of so called difficult to test faults by sensitize propagate and justify approach which is a deterministic approach. So, basically that is where optimization is required, but that is not exactly a part which is to be covered in advanced test cases for advanced circuits, basically for even for the smaller or medium scale circuits we have to go for a optimization at that level.

So, we will be just a covering this part in a very brief manner and then we will try to cover the other aspects of automatic test pattern generation or test optimization like generation of quality test patterns, which can covered different type of advanced faults which are more relevant towards the technology, which is relevant for modern day VLSI fabrication. Then we look at techniques to compress this test pattern so that, we can have a smaller tests test set or compress test set which can lead to better optimal results.

Then will also look at some kind of the some concepts of like test of the abstract level so; that means, as we have already discussed that if you are going at the gate level, the things will be too complex and too elaborate or basically in other words test will be too complex or we have to go to very geniality level. So, you have to do at higher levels of abstraction like a register transfer level even the fault models has to be a very at a very abstract level and so forth.

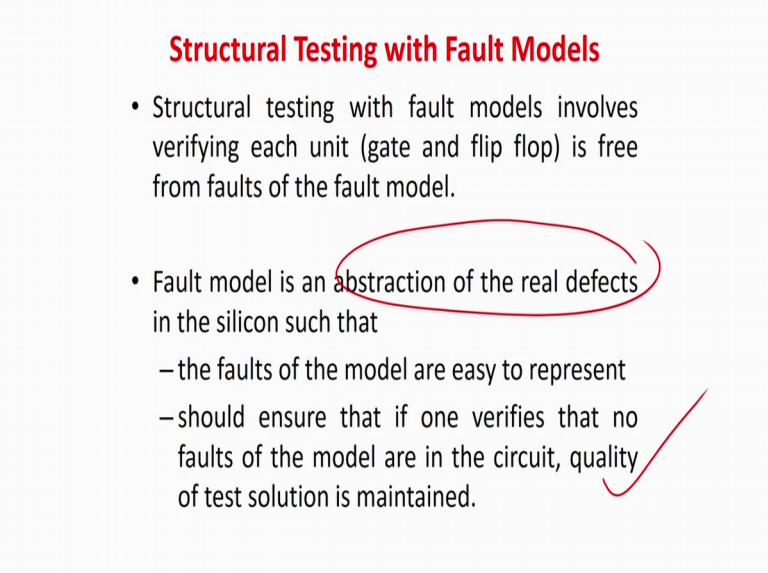
So, basically our main coverage in this not only today's lectures, I mean including today's and the next few lectures on this module, we will be mainly covering trying to look at this aspects that is good quality test patterns. So, that it becomes relevant to modern fault models, how to compress test and go for an abstraction level. But does before we touch we will just see what people also have tried of different kind of optimization, which is again relevant towards the traditional circuits that is how we can go for a better sensitized of test pattern generation by sensitized propagation justify approach.

Just before I start this, I will just I mean just on try to highlight one important point there are two parts in testing. One is test pattern generation which is actually one time activity you take a circuit and then try to find out what are the patterns and then if the circuit is very large, you may take days and months to the find out test patterns.

But still that is because you are going to do it just for a one time and second is basically, the test patterns which is to be applied to the circuits which is be done for all the samples because we know that yield is a yield is not too high. So, the is not as high as 99 percent plus you have to test each and every sample. So, at that point a number of test pattern should be applied should also be less, but if you have a large number of test patterns and all the test patterns has to be practically applied in all the circuits. So, the number of test patterns required to do the testing should also be brought down. So, that is a more important aspect there is using the time to generate the test patterns.

So, that is why we actually call quality of test patterns, that is we should have very good quality test patterns so that, you can do testing more effectively with the less number of test patterns, this more important than corresponding to speed of the algorithm to demining the test patterns. Never the less, that is also an important aspects because if you have a very large circuit, you may land up in years to generate the test patterns ok.

(Refer Slide Time: 04:59)



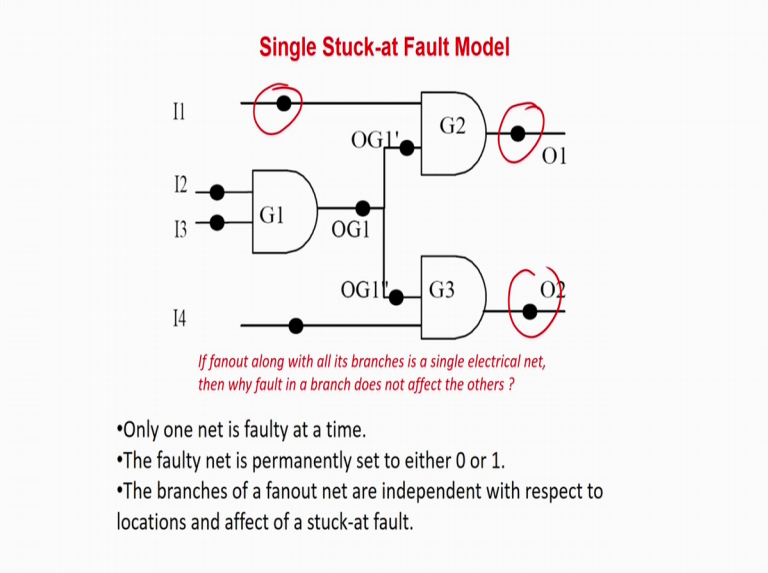
So, as already discuss nobody can go for functional testing, because that will actually making things very very complex and you may not get a solution in years. So, you always go for fault models. So, what is the fault model? Fault model basically you assume that there is certain kind of faults that may typically appear in the circuit and you have to validate that those faults are not there.

So, fault model is basically an abstraction of the real defects. That is if go for transistors level testing if you are go for physical means layout or the such kind of testing of this testing of the devices at the fabrication level like testing of the VRS, testing of the interconnects that may blow up the thing blow up all the algorithms. And the time required to do the testing, and it is also not required because it has been found out that if you have really good quality test fault models then basically the second point should hold that if one verifies that there are no faults of the model in the circuit, then the quality of test solutions is maintained. That is the correlation with real functional test or the real so called the most accurate way of testing like you test of the most rigorous level.

So, even if you can have such good kind of fault models, which can tell that if no such fault model is there then you can be 99 point plus sure are the correlation level is so high, that if you are assured that no fault on the fault list is in the circuit, then more you can be very very assured that the circuit does not have any kind of real defects. So, we require

both kind of fault model. So, that is are the de facto standard which you are always applying.

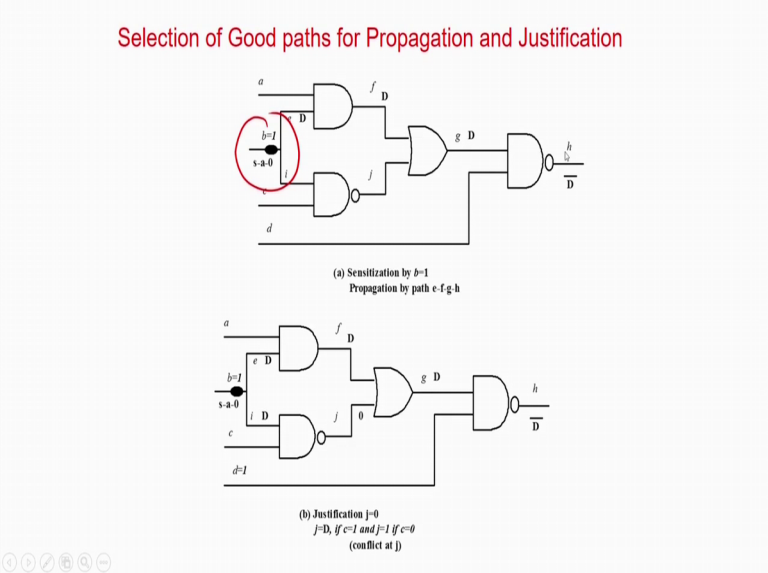
(Refer Slide Time: 06:27)



Again you discussed that is single stuck at fault model is one of the most widely accepted fault model, because till a last 10 years back, if you could verify that there is no single stuck at fault the circuit is act can be consider to be having no faults, and the correlation was also very very high.

So, what was the stuck at fault model? You assume that each of the lines of circuit can be as a stuck at 0 or stuck at 1, and you have to verify that by test pattern application that none of them occurs. And generally you called the single stuck at fault model because you considered only one line at a time and that was actually the one of the most de facto app and we apply test patterns. So, this is the first fault model on which every application or test for almost every circuit was based on.

(Refer Slide Time: 07:11)



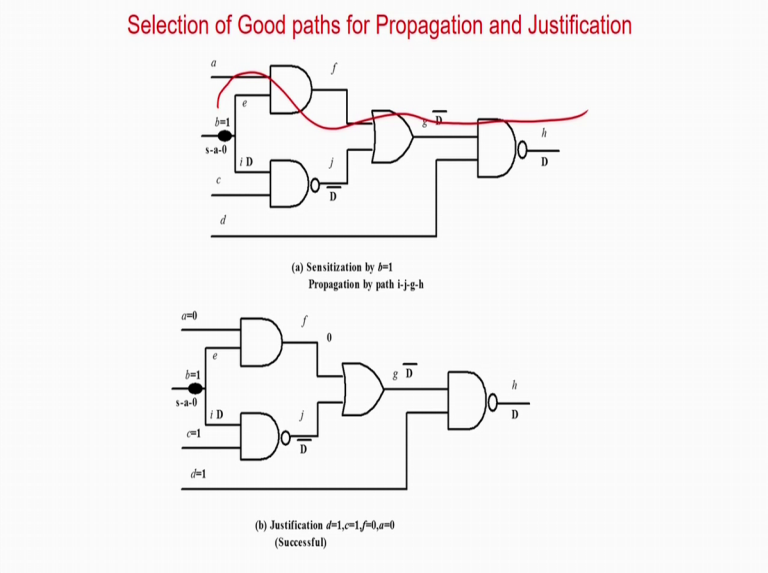
So, now, we will just give a very brief idea that how people thought of optimization at that level. So, given this fault model how people tried to optimize. The first part is basically in sanitized propagate and justify approach is that, you have to sensitize like this example already we have discuss we have taken a stuck at fault over here let me zoom this, just I am not going to elaborate because already we had discuss this example then what we have do we have to sanitize the fault stuck at 0 fault.

So, you have to apply a 1, then you have to find out one path where you propagate the values of the output. Like in this case we have selected d sorry e f g and h, and then actually D will propagate and then you have justify. So, then if you just look at this part we try it try to justify this approach then actually it is d prime.

So, you have to apply a one over here we require a d the output of an or gate. So, you require g D or 0 over here, but if you look at it if it is a stuck at 0 you apply one D and then basically the output j cannot be a 0 because to get a 0 we require a one and one. So, there is a conflict.

So, what actually it says that one of the lots of words have been done you can find out in the references which will be uploaded in the course side, that lot of first level optimization we people thought on ATPG if on the classical level of circuits is that how can you find out good paths of propagation and justification. Like for example, we have found a we have taken this path which actually failed and there was a conflict.

(Refer Slide Time: 08:37)

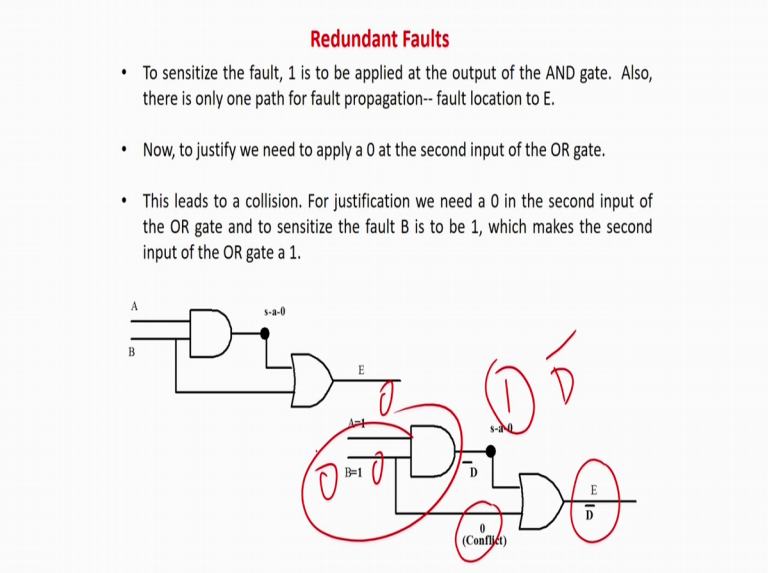


But now we have already seen in the last class that if you take the other path like same thing as sensitized by applying a one path. Now I am using the path i j g and h. So, in this path basically you can see D inversion D prime D means normal one fault 0. So, you will become out through this path and then if you have to this a nand gate. So, justification DS should be equal to 1 as your or gate. So, it should be 0 and you can easily make f equal to 0 by applying a equal to 0. So, basically what right one to say is that the second path basically this is a good path for propagation and then you can get the solution.

So, first level optimization that people try to think is to find out good quality test paths for propagation. Because in this case there are only two path, but they are real circuits you will have hundreds of options. So, which option to try? So, lot of work has been develop to find this such kind of good paths and lot of work had been done. So, that you can speed up the test pattern generation algorithm in this course as will be mainly looking at the circuits which are beyond the simple typical VLSI circuits.

So, we will not go into depth of such algorithms you can read lot of such papers because they are more than around 10 to 15 years or even 20 years out dated, but still whatever the VLSI test algorithms are still running in picture, always trying to find out which are the typical good candidate path because if you are taking a wrong path basically then you may have to keep on integrate.

(Refer Slide Time: 10:01)



Second very very interesting part is something called redundant faults. Redundant faults are such faults which cannot be tested because they are does not exist any patterns which can test it, but that how do you know? Because you will have a fault like in this case you have fault on you are multiple paths for propagation and justification.

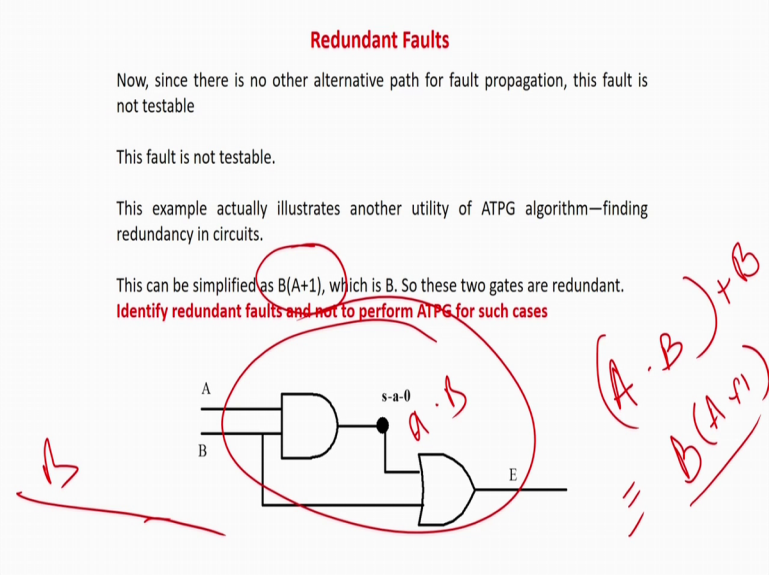
So, if what is the redundant fault? Redundant fault is fault which cannot be tested, there does not exist any patterns which can test it, but how do you know that? You will be able to find out only by applying the exhaustive enumeration of all the paths through which is can be propagated and then you find out the they cannot be justified. So, if there are like for example, in this case may be there are 100 fan outs from this all the paths you have to try then only you will able to find out the that they are does not exist any patterns test it and then you know that this fault can be test it.

But is a huge weights waste of test computation resource time. Because you will be trying so many paths and the end you will find out the there is no pattern to test it. So, another important optimization principle, which people try to find out if how basically you can find out redundant faults that without trying for an ATPG approach. So, just let me give you an example. So, you can easily appreciate the fact. So, this is a circuits with the single stuck at 0. So, it is a stuck at 0 of course, you have to apply 1. So, you have to you have to have a sensitization one over here. So, normal 0 fault 1 D prime and of course, there is a only one path to the output. So, your output is a D prime at E.

Now, of course, is a or gate. So, you required a d over here to get a d over here of course, you require a 0 over here, but in fact, you see, but to get a one of this output of this and gate you require a 0 and 0. So, there is a basic is a conflict. So, simple conflict in this case and you know that this fault cannot be escape as simple as that. Because there is only one path through which can be outputted. So, there is no other option to try. So, just the one iteration you can find out there does not exist any pattern to test it, because they does not exist any other path for propagation because the path we have taken for propagation there was a conflict.

But if I told you; So, this a example of redundant fault the fault which cannot be tested. But if there would have been many many fan outs than actually would have been a real suit because you to try so many paths, maybe I am assuming that there was no other paths for which default could be successfully propagated and justified and having the assumption, but then only are there so many huge number of iteration. So, you would be able to conclude that not a good way of handling. So, basically now people try to find out why such things appear.

(Refer Slide Time: 12:15)



So, people have found out that such things appear, because the circuit is actually redundant the circuit is not optimal. If you look at this the binary expression for the circuit is A plus B, A dot B and basically you have plus sorry it is A dot B plus B correct.

So, this is this is a dot b and this a dot b plus b. So, you can easily reduce it to the form like A B plus B. So, this is the what is the expression that is equivalent to B into A plus 1.

So, A plus 1 is nothing, but 1 and then basically is nothing, but 1. So, the whole expression is nothing, but the simple straight line so; that means, what is the redundant circuit; that means, the circuit has not been properly sorry the A plus 1 is 1 then the circuit actually B. So, this is basic nothing with the gate B. So, this is simple line the output e is equal to B just a straight line straight wipe, but b is connected to be output so; that means, what? The circuit is not an optimize circuit you have put lot of redundant case.

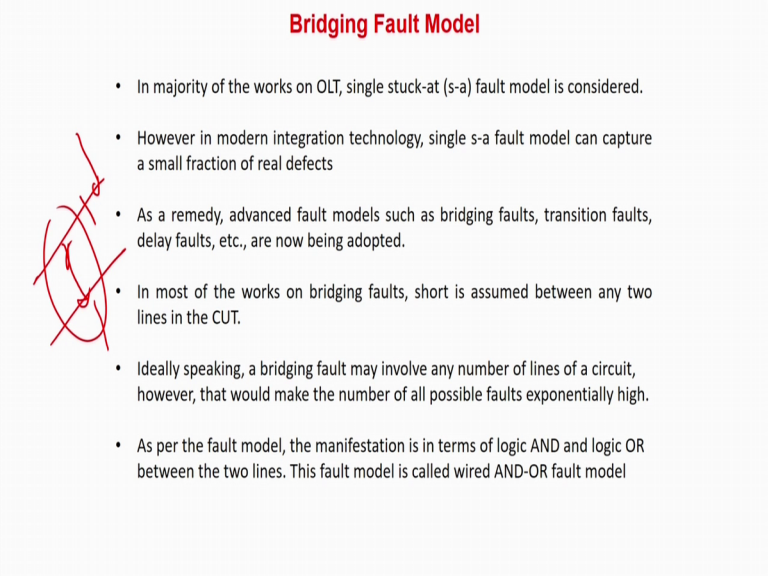
So, in VLSI design sometime lot of redundants are put to avoid hazards and put buffering arrangements etcetera generally to avoid hazards people lot sometimes put redundant dates, but then somebody should identify the redundant circuits beforehand because if you go for a real optimal VLSI design which has already been covered by Chandan Sir in the case of design, the generally we do not try to give any kind of redundant circuits.

Because there will unnecessary not do any job, but it appear area over here. But so, such cases generally we keep it for a very explicit purpose because no automation tools or design tool we give you a redundant circuit, just in this case because you should optimize this by any case standard technique like espresso, kern of map (Refer Time: 13:54) you will going to get the solution e equal to B.

But then what is this explicit circuit has been kept? Sometimes people keep to avoid hazards you can look at any standard digital design or VLSI design text book. So, to avoid hazards people try to maintain such kind of randencies. So, in such case redundancy should be explicitly told to the test engineer so that, you should avoid such kind of faults in those parts of the circuit because they are generally untestable.

So, if there is redundant fault that will actually take lot of your test computation time because you will be unnecessarily keep on up to finding of the loops but these two aspects mainly that is how to find out good test paths for propagation and how to find out redundant faults an eliminating them are basically the first ideas of optimization which people tried on classical VLSI circuits. Now we will coming to the real part of our course on optimization of test. So, the first part which leads to be applicable to modern SOCs NOCs to advance the false model fault model.

(Refer Slide Time: 14:56)



Because as of you might be thinking that stuck at fault model is a very simple fault model, you assumed that of the lines have of stuck 0 or stuck 1 and you have to and you to find out by test patterns that none of them exist and then the correlation is 99.9 percent.

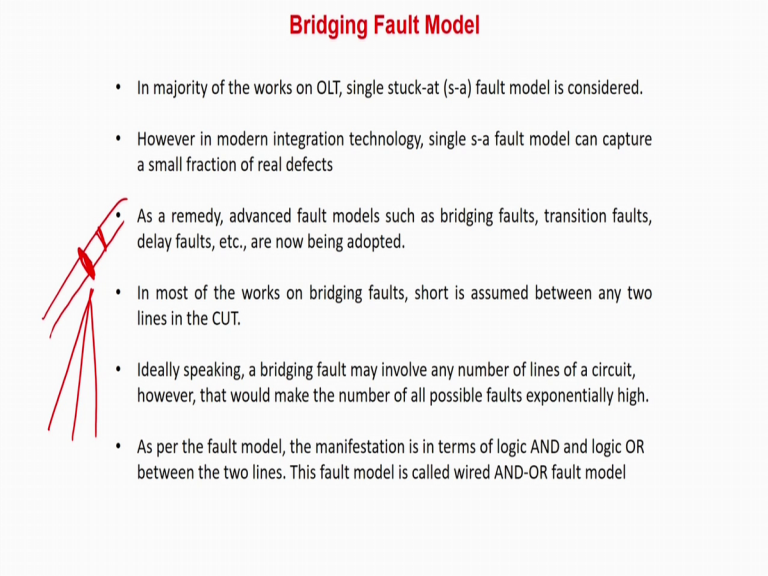
So, one question all of you might be having is that, is really such a simple fault model going to accurately match my functional results in a very very dip sub micron technology the answer is no. When stuck at fault model was a very nice model and very highly accurate model actual the. So, when we are actually the talk talking about that phase of time, when stuck at fault model was a de facto standard and you could get very good results, the micron was around 5 micron or more or it was not very dip sub micron at that level, it was some 5 microns or higher than that. So, in that case what happens to the higher the micron technology, the more is the gap between the two lines which will be laid out in the VLSI circuit.

So, in such case as the gap in between the two lines are high. So, that probability of faults occurring there will be lower or in fact with the even the faults where there, they generally laid to something lines the lines being cut that is the stuck at 0 or sometimes something is to happen into interference etcetera these line is or else be stuck at 1. So, generally other than that more complex there will of a very very rare. So, stuck at fault model somehow intuitively or statistically they gave very good results, but now in dip

sub micron technology and also at a time the frequency was also less, not in the level of gigahertz.

So, interference etcetera were white less compare to the modern day dip sub micron technology in dip sub micron what happens in 45 nano meters etcetera the line being laid are so, near to each other that sometimes there is a short in between in these two lines.

(Refer Slide Time: 16:29)



There is a line started get gap gets couple or there is lot of interference, giving to something called more advance type of fault which you occurred there is something called a bridging fault model. Impact in is a fault again bridging is the fault model; basically these two typically line bridging what we assume that two lines have got stuck and then we have try to find out that that such fault does not occurring the circuit or is not present in the circuit for that you have to find out test pattern.

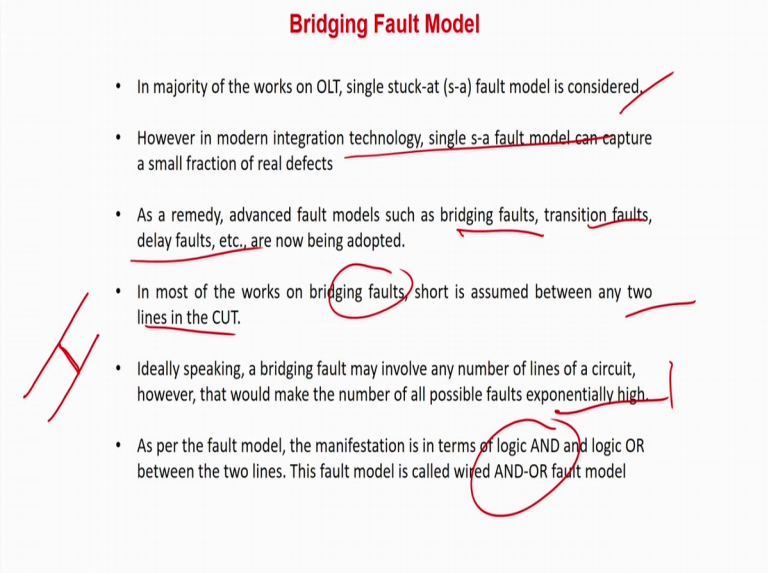
But again as it is a fault model always you cannot assume that is 92 2 lines will be stuck and you are going to get a short and that is the real fault which have to validity. Real defects can be extremely different, it may be happening something like that that n number of lines might have course started a point or it may be like that lines are not exactly start, but they are lot of electromagnetic inference in which case the signals are getting coupled and value of one signal is dominating over the other size vice versa.

So, impact, but the modulation is that in case of bridging faults, you take two or more lines at a time. In stuck at you take only one line at a time and in case of bridging fault you take multiple lines to be coupled in a fault, and that is more relevant because we are because when dip sub microns technology started coming into picture people who were finding out the cold nature of stuck at fault is going down to even 70 percent or less.

That is your having test patterns and you are trying to find out faults , but the problem what was happening is that, the correlation is 70 percent or even less because as I told you in dip sub micron the lines are coming nearer to each other. So, we have to consider such kind of fault model, in which multiple lines are considered into the modeling factor.

So, the first modeling technique which is applicable to advance circuits is a bridging fault model.

(Refer Slide Time: 18:19)



So, as I have told majority of stuck at fault majority of stuck at fault are consider, but in modern integration technology single stuck at fault can capture only a small fraction of real defects that is why the correlation is coming down. So, advance fault model is bridging fault, transition fault, delay fault are actually now being adapted. So, in today's lecture basically will try to see how this modern fault modern what are these modern fault models, and how basically you can go for ATPG of this faults and then very interestingly we will find out that if you are covering the advance fault models

automatically stuck at fault models are getting covered that is stuck at fault model such as smaller set this advance fault model are the bigger set.

So, if you are starting for stuck at faults, you are actually leaving some out of space. But for the advance faults if you cover, all other things are get basically getting covered. So, those philosophies will try to develop. So, one of the first such type of advance fault model is basically the bridging fault model. So, what they assume? They assume that any two lines in the lines in the circuit under test that is the cut may get stuck or may get couple, and then you have to find out whether they exist or not , but again this is again an optimization.

So, once I just to forgot to tell you optimization basically I have told you that two type of optimization one is the deducting the number of test patterns, and one is the reduction in the test pattern generation type. But there is another factor actually whether you are going to do the optimize at the cost of some quality of service or your optimization does not involve any quality of cost. Like if I reduce the test pattern generation time by finding out a good alternative paths of propagation etcetera, it does not hamper your quality of circuits. But if I somehow reduce the number of test patterns there can be multiple ways of doing it one can be by slightly reducing the fault coverage.

You can tell that this part of the circuits are not prone to failure I do not require to test them so that is one very simple way of reducing number of test patterns to be applied, which will reduce your test application time as well as test pattern generation time, but again to be remember that that will slightly bring down the quality of service. And again such type of information are not any algorithmic or any kind of a algorithmic approach to be that it is statistics and based on the designer feedback, which will tell you that this region of the circuit has to be tested well this region you may drop it etcetera etcetera. So, there is no such algorithm to do that.

So, in such case we for optimization, but at the compromise of some kind of a quality of service. Like in case of bridging fault, we should assume like is stuck at fault model, we assume that only one fault stuck at fault can happen at a time there is again a comp fight compromise with the number of quality of service to a very little level, because any two stuck at fault can happen together, but NC 2 NC 3 two can happen together, three stuck

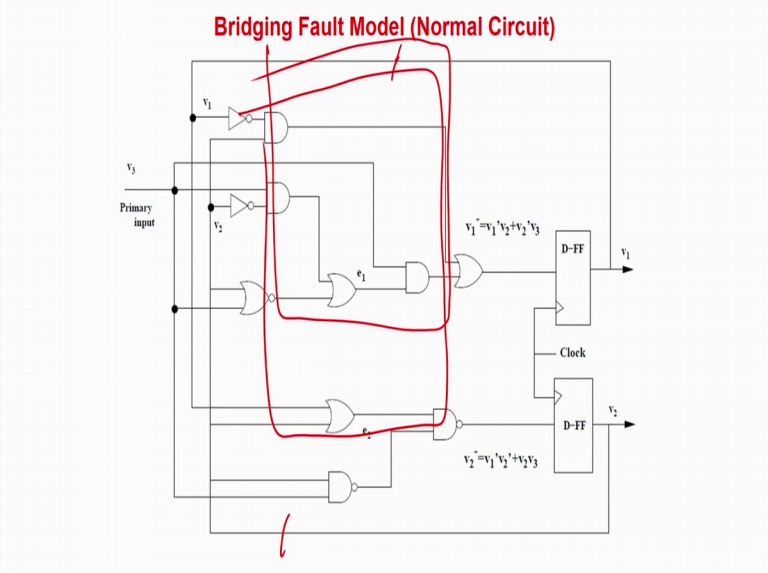
at fault can happen together, but if you take it the number of all fault combination will be huge it will be three to the power n kind of a stuck normal stuck at one stuck at 0.

So, each line there will be 3 to the power n and it will be exponential blow of complexity. So, here we are assuming a single stuck at fault model basically, which is nothing, but slightly compromising on your test quality solution like in bridging fault, basically it can happen that in a region this is a region multiple lines are passing together. It can happen that all of the lines may get stuck or any two lines may stuck, any three line may get. So, all such combinations we have to study, that will again that will again blow up the fault (Refer Time: 21:25) model complex like anything.

So, people assume that any two fault any two lines can be stuck at together and that further we will be considering as the bridging fault model. So, any two lines means again we are compromising on the quality to speed up or optimize the test development and time as well as the test pattern application time because the test pattern will be reduced. So, ideally that (Refer Time: 21:47) I am saying that huge all lines may be being a problem, but that will leads to an exponentially high.

So, as per the fault model what they take is something called a and or logic fault model and or bridging fault model. So, what they assume that the two lines are there, they get coupled by coupling the affect is a and gate or a or gate I will give a example.

(Refer Slide Time: 22:07)



So, let us take a very simple circuit example which will be using 4. So, this a 5 now will be mainly talking about sequential circuit as example because we already know this (Refer Time: 22:15) So, (Refer Time: 22:16) means you can directly set or reset your flops.

So, these are simple circuit the two lines. So, any two lines may have a stuck at fault thus any two lines may have a bridging fault. So, now, one thing is there. So, generally again one optimization people will do that is actually (Refer Time: 22:30) of bridging fault. Like for example, this line and this line may not get coupled, because they are quite far in the lay out. So, generally people will try to take a window that may be only these two lines these lines may have a stuck in this we have done a bridging fault because of the coupling because of the close proximity, then they will move the moving window.

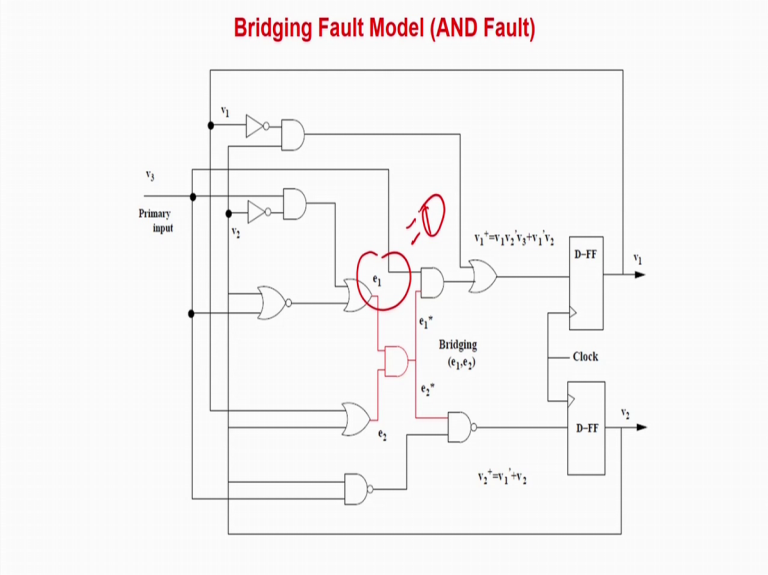
So, this way we will have the moving window in the 2 D space. So, that you can actually find out which are the line which are in close proximity, only that can have a fault. So, n c even if two lines taken together two lines are far apart in the lay out, need not may not have a coupling fault or impact they will not have a coupling failure or a bridging fault.

So, people optimize on that level and to my understanding that does not actually need too much lowering of quality of service or not even a single percentage of fault will be lost or defect coverage will be lost, if you are taking a like a assumption like only lines in the close proximity will have such kind of I mean coupling. But it can happen that this three lines may get coupled because they are in close proximity, but to avoid complexity people generally try to take only two at a time. But now slight advances are also happening in which they will try to restrict the window size, but in that window they will take 3 or 4 or 5 lines in coupling.

So, there are lot of optimizing techniques in this course basically as a 20 hours course. So, we try to give the basic philosophy to you. So, one way optimization is that you just take two at a time and then have a slightly bigger window size that is the proximity in which the lines can have a short of coupling or in other way what people do much which is not the very well popular approach, but still people some people have found out that there is good solution, they try the keep the proximity window small and in that they assume that multiple lines more than two lines may get coupled.

But anyway let us come back to our old problem which we are dealing with it. So, let us assume that we are going to take e 1 and e 2 two line which are in close proximity have a bridging fault that is they may have a coupling.

(Refer Slide Time: 24:23)

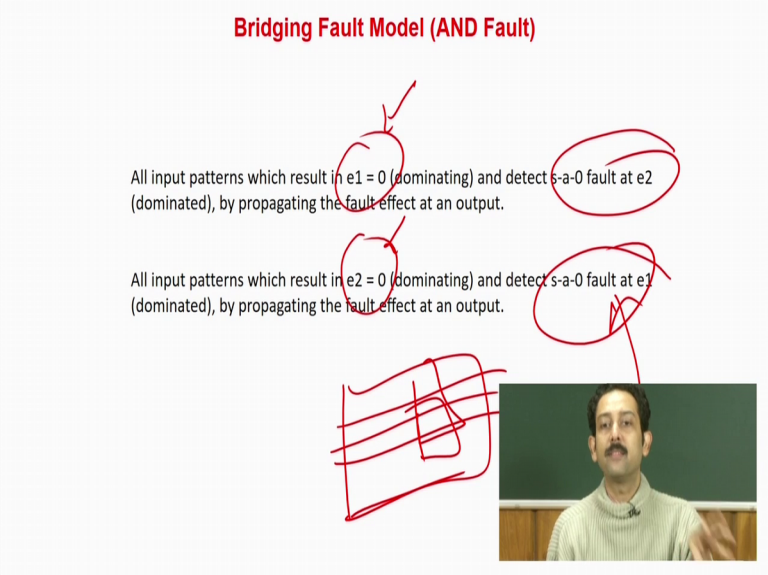


So, what is that coup coupling about? So, the assume that these two lines e and e 2 which are involved in coupling they have some logical interference like is a and interference. So, e 1 will be fed to this line as e 1 star let me zoom this part. So, if you look at it e 1 should be initially was feeding to e 1 was feeding to e 1 star this line and e 2 was feeding to e 2 star directly, but now due to coupling there will be and logic. So, e 1 and e 2 are not independent, but generally what will happen is that, the value of e 1 and e 2 will be ended and they will be feeding it to the output.

So, of course, if e 1 is equal to one and e 2 will be equal to 0 or vice versa, they are basically a failure in the output of the circuit basically, but 0 0 1 1 there will be no affect 0 0 means both the output of and gate is 0 and vice versa and for one it will be; obviously, 1. But it is not only about the and gate there should be another dual of it first you should replace by and gate and then you should replace by or gate. Anyway the or gate is it will be just same up just in this case we have to replace the or gate by a and gate by a or gate an apply the same principle.

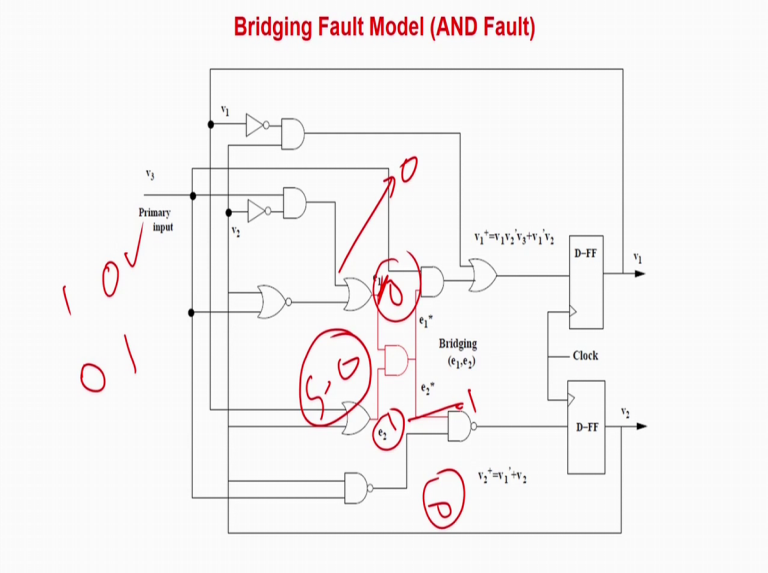
So, I am not going to discuss it in detail, but that is just the dual of it that is same redundant to avoid time wastage of time I am not going to cover the same example is the or gate you can try that at home.

(Refer Slide Time: 25:37)



So, there are two parts basically one is called dominating and one is called being getting dominant. All input patterns which result equal to 1 that is e 1 equal to 1 sorry e 1 equal to 0 sorry e 1 equal to 0 and gates. So, e 1 equal to 0 and detect stuck at fault e 2 by propagating to an affect to an output now I will give you the philosophy, then we will the slide. So, basically what happens? So, when there can be a problem?

(Refer Slide Time: 26:05)



So, there can be a problem if this is a 0 and this is a 1, that is e 1 equal to 0 and e 1 equal to 1. So, what should happen ideally speaking here you should be able to propagate 0 and here should be able propagate a 1, but now this is not happen because this and gate. So, there will be a coupling and e 2 will be converted to 0 because it will be pulling it down.

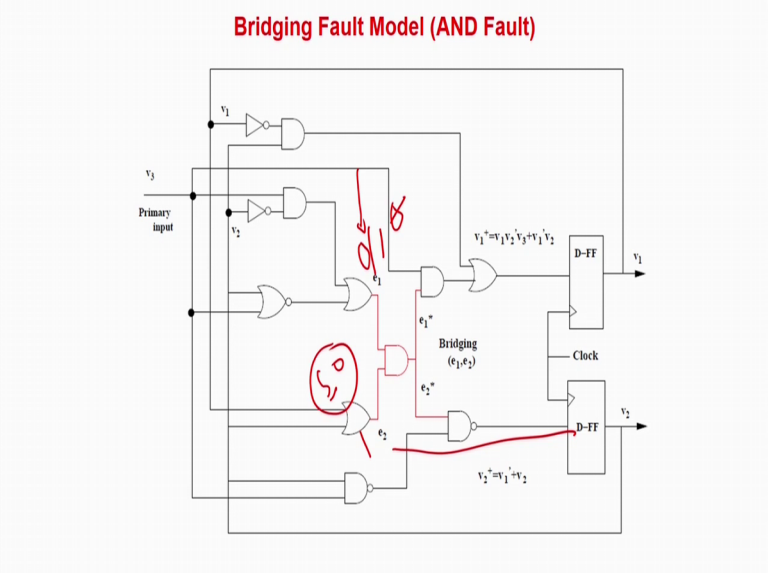
So, in this case what is happening? We call that e 1 is dominating e 2 and in the other case when e 2 is equal to 0 and e 1 equal to 1 we will call e 2 is dominating e 1; that means,; obviously, whose value gets propagated is dominating line and whose value gets changed is the line being dominated.

So, therefore, therefore, say that bridging fault testing is very simple similar to testing of a stuck at 0 or stuck at fault, but with a slightly added condition like in this case you see for example, I want to test this bridging fault. So, what I have to do? I have to apply any one of these pattern either 0 1 or 1 0 because both of them 0 0 1 1 will have not have any fault sensitization affect. So, fault sensitization in this case is what? You have to apply a 0 or 1or vice versa let me just take this pattern. So, apply a 0 over here and I apply a 1 over here this will sensitize the fault because in that case what should happen ideally one should go over here now 0 should go over here.

So, this is will be a stuck at 0 fault. So, just look at the philosophy I will actually require to test a stuck at 0 fault at e 2 that is what I have to do. And at the same time I just require another simple condition that e 1 should be equal to 0. So, basically bridging

fault testing is nothing, but a simple stuck at fault test only, but with a slightly different problem that some other lines in the context should be having a different value which will be set and reset. For example, let me just assume that we are not going for any kind of a bridging fault test, I am just assuming that this line I am going to test for a stuck at 0 this is what I am going to do just assume in this case what is going to happen?

(Refer Slide Time: 28:04)



Some I have to justify this line. So, sensitize this line by 1 and then may be propagate the value of D over here and that I have to go for the back tracking which is actually going to justify all the values. But in that case I will not bother that whether e 1 is a 0 or e 1 is a 1 that I do not have to bother because in ATPG it will just give me any value. But in that case you have to now think about the optimality; if somehow I could have put a 0 over here I could have tested the stuck at 0 fault as well as I would have tested the bridging fault over here, but if I would have put a one over here this is just going to test the stuck at 0 fault.

So, basically testing of a bridging fault is similar to testing of a stuck at fault model only, but we some added constraint like this, I will make the things more (Refer Time: 28:51) so, will understand this. So, just develop the basic philosophy that testing a bridging fault is very similar to testing stuck at fault, but with some added constraint in some other lines. So, that the bridging this coupling IRS activated. So, in the new method of

optimization what people will do is that, people try to first find out the test patterns for a bridging fault.

So, when all bridging faults will be covered you will find out that lot of stuck at fault that have been already been covered because bridging fault testing is nothing, but stuck at fault test, which form additional constraints in the line whichever fault stuck at faults will remain that you can go for a transitional approach when it will be done.

So, here actually you are within a very nice I mean solution, which people have found out that optimization is at this level do not stuck start at stuck at fault you start at the advance fault model of course, you will find out by testing the stuck at form in the advance fault model lot of stuck faults will be by defect covered, then just start test for the remaining and you are going to get the solution. Because as I told you are not going to touch upon every combination of means patterns every combination of lines in a circuit. Typically as it is very large we always restrict the bridging fault to certain bridgings which may have very compact layout or which you know may be prone to failures.

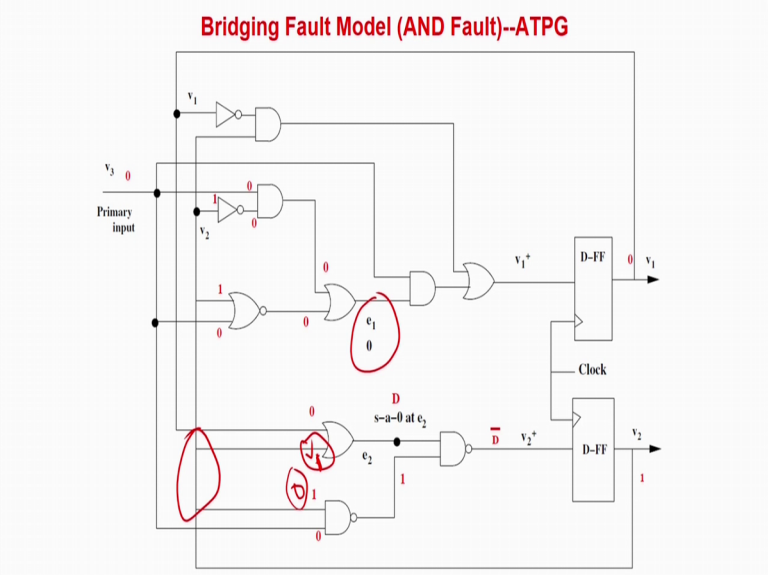
So, for certain reasons of the circuits we go for such sophisticated bridging fault test; because test generation in the last in stuck at fault is just to sensitize propagate and justify, but in case of bridging fault also some additional constrain has to be met. So, test generation time is slightly higher. So, restrict the test part to certain parts of the circuits and then we find out by testing the bridging faults what addition stuck at fault already covered and for remaining (Refer Time: 30:25) So, this is another optimization, people try to apply this is one of the most important optimization rather which people try to do when they are covering the fault models in the model context.

So, this what I am say I just can brief to the slide. So, all the input patterns with result e 1 equal to 0. So, is the and gate. So, which line is 0 will try to try the other gate. So, it is basically a dominating line and detect a stuck at fault in e 2 by propagating the effect to the output and this is the other one, in which case e should e 2 should be dominating and you should propagate the value of e 1 to the output. So, these stuck at 0 fault will be de facto tested at both the lines, but only with an additional constraint that in one case e 1 should be made 1 in the other case e 2 has to be made 1.

So, this is anyways similar to stuck at fault, but this an additional constraint. So, it will circuit is very very big you can understand that even adding one more complexity will lead to more amount of time, more amount of iteration because it is not about just setting another constraint. Setting another constraint mean you may have to reiterate because you may always have start having conflicts. More constraint you put more conflicts you should have and more iteration only test pattern generation rounds has to be there.

So, people try to restrict the bridging fault models to a small part of the circuit may be assume there is lot of congestion in the layout over here and that also they will start making a moving window concept. So, this advance fault model generally a restricted to some specialize zones ok.

(Refer Slide Time: 31:45)



So, now let us look at the example we were discussing in a more colour animated in a better manner.

So, as I told you we are just taking the example that when e 1 is having the value of 0 and let me zoom this part. So, we are going to try to take this e 1 is equal to 0 and D have to generate the find out that whether e 1 is dominating e 2 that is it is 1, but actually it is making a 0. So, you have to generate a design a stuck at 0 fault to 1, but this; that means, by testing the bridge and bridging fault because and bridging fault you have to test visa versa not only about this one dominating this also you have to see whether this one is dominating this both have to tested.

So, both will be testing the stuck at 0; one case you will testing the stuck at 0 fault and this case and the other way you the take testing the stuck at fault at e 1. And then again I am not showing that again you have to also consider the or bridging fault. So, in that case stuck at one value will be tested because it just a dual you can cal do that hand calculation at your home and you able to find out, but again as I am telling you.

So, what is the beauty or what is the optimization over here that is first you go for bridging faults bridging fault test and even find out the lot of stuck faults already have been covered like in these case you will be finding out that stuck at 0 fault at the both lines and stuck at one fault both the lines will be covered when your considering the and or bridging fault respectively. But again problem here is that you have to even additional constraint like when you are testing a stuck at 0 fault at e 2 you have to make e 1 equal to 0, and similarly when your bit going for stuck at 0 fault over here you have to make e 2 equal to 0.

So, this additional constraint means some more justification, you have to do in the circuits and more you try to justify lot of conflicts may happen and you may have to reiterate to design the circuits. So, in this example I have not shown the iterations, but if you just take a wrong path or you just take wrong values, but I try I will try to show you that different variations of solutions are possible by chance if you take a wrong value, you may have to again refer a iteration that is very important which we have to understand.

That means what? That when you are justifying many times you get an option either to give a 0 or 1 at the some line, but if you take the wrong choice by chance then you may have conflicts. So, large circuit more options will be there and larger conflicts may happen. So, more types of constraint like e 1 equal to 0 along with the stuck at 0 fault, we need to more type of constraints conflicts and more iterations you may have to do. So, that is why bridging fault base by generation is the more time complex solution that is why people optimize by trying to restrict the a window size and also the regions ok.

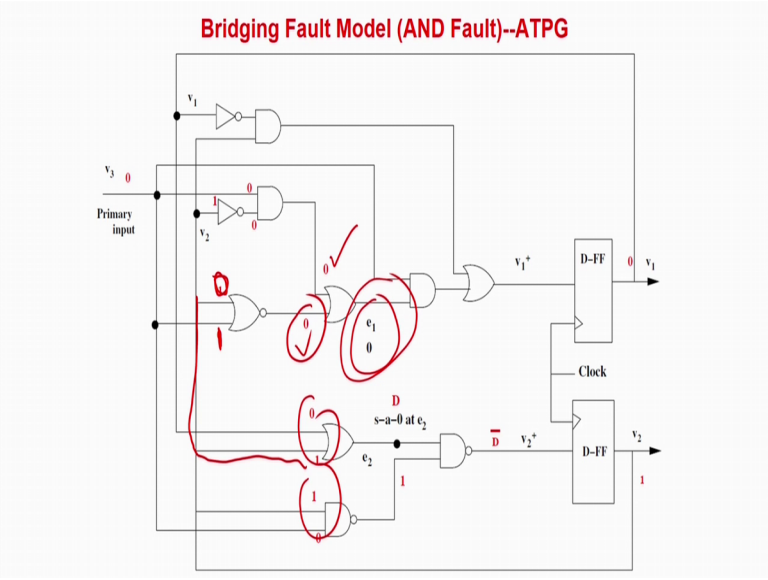
So, these are stuck at 0 fault. So, I have to apply a 0 over here which is the constraint additional constraint and then I go for a propagation. Again as I told you as we are having a scan circuit so in fact, I give the flops are not there and I just go for something called a normal ATPG. So, normal ATPG means stuck at 0 I apply a 1 D inverted D

prime that is what the I am propagating the value to this and here I require an a additional 0 over here. Now basically I have to try to do a back track. So, if I try to do a back track then we just look at this picture. So, an and gate I have to put a 1 I can again this is a 0. So, I require 1 over here. So, again a here I am trying to take the options like is an or gate how can you get an 1? You can get a one of 0 1 1 0 and 11.

So, three options are there. So, here I have try taken explicitly I have done some calculation and I have taken the option 0 1, but if you take the one as an option you are going to have conflicts. Like for example, here the requirement is a 1 in this case the requirement is a 1. So, it is an and gate you can get a 1 by 0 0 1 0 and 0 1. So, any of the three options would do, but by chance if you take a wrong option then you are gone basically you see I can have a let me just say just give you. So, you can also have a 0 over here it is possible.

Because 0 0 also give you a one and as I told you can also have a one as I told you it is possible because 1 0 1 1 0 1 both can be give a one at the output of the or gate that is at e 2, but now if you take as 1 over here and a 0 over here as the same line there will be a conflict. So, there will be a issue. So, you can see there is a lot of a conflict over here and in fact. So, then you have to again reiterative. So, these can be reiteration. So, for example, say I am intelligent and I am putting one and one. So, no conflicts is there, but you can just see that I have to again go for another kind of a justification over here.

(Refer Slide Time: 36:05)



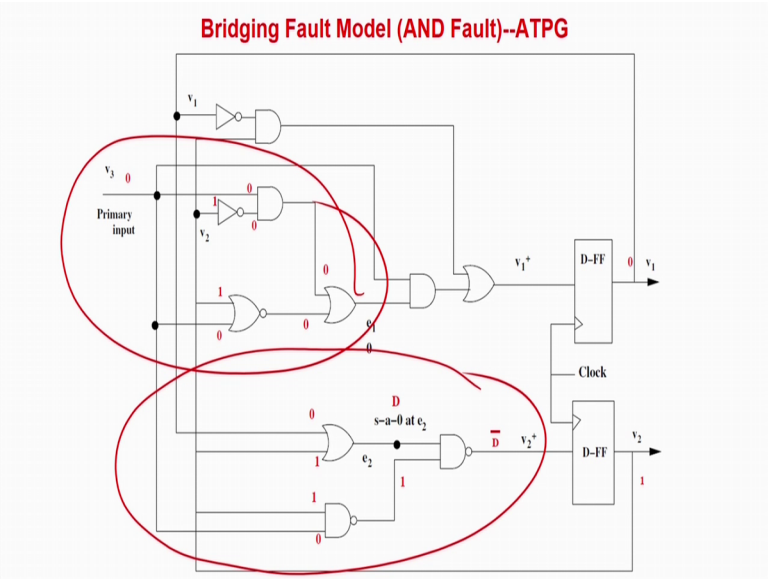
So, this is a new justification required over here sorry another constraint like e 1 equal to 0 you have to put explicitly, because is not a stuck at fault test it is basically a bridging fault test. Say for example, I am putting a 0 over here, but as I know that to put a 0 as the or gate output you require a 0 over here and 0 over here.

Now, as you can see that this is a nor gate. So, nor gate output 0 means basically you can have the solution 0 0 1 0 or 11. So, what may be the case. So, for example, basically if I have the answer here as a excepting 0 0 at this position it should not be 0 0 this is not allowed un unless all other cases like 1 0 0 1 1 you can also have the value of 0 at the output. So, if I for example, this is also going to hold and this is also going to be a valid solution to get a 0 over here, but if you see if I make a solution like this.

So, this line is basically connect to this line and there can be a conflict. So, therefore, what I am saying trying to emphasis on these point is that. So, more number of constants we have and more bigger the circuit is there are a lot of options like here I am getting three options, here I am getting three options, here also I am getting three options. So, if you are not taking a good option you can always have more number of conflicts and you should have you must will be having a reiteration.

So, that what it tells that. So, even just testing a stuck at fault means just you have to have a constraints on this part of the circuit.

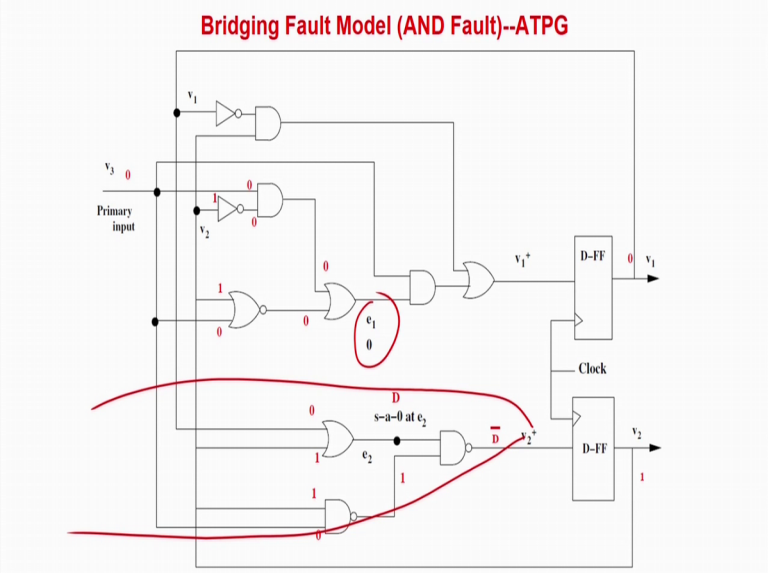
(Refer Slide Time: 37:33)



But as you saw bridging fault also you have to consider some additional part of the circuit based on some consistence has to be applied. So, more number of conflicts can happen and test generation time will be higher. So, that is why by op for optimization reason again I am repeating they try to restricted, but the advantage part of this advance fault model is that, if you test the bridging fault automatically some stuck at faults will be tested anyway let us come back.

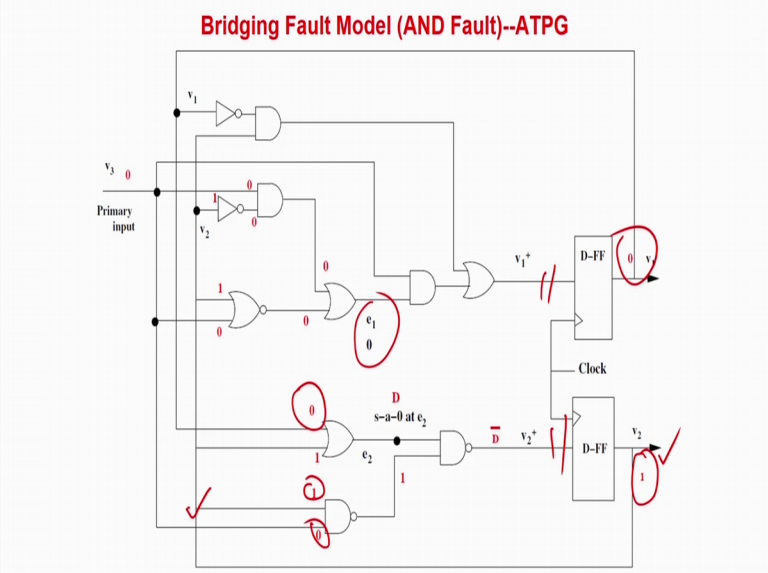
So, basically I am going for this level of justification and then again additionally I should have a 0 over here.

(Refer Slide Time: 38:03)



So, added been these stuck symbols stuck at 0 fault, these should have been done the job. So, now, basically if you look at it, I require basically some kind of values like precise and look at the exact values. So, if you see I say that I require a 1 over here and I require a 0 over here.

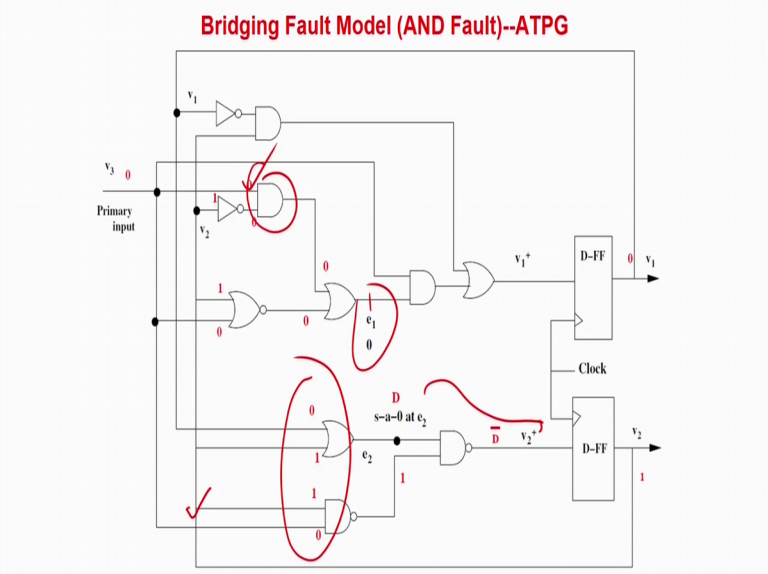
(Refer Slide Time: 38:21)



So, if you can find out this one basically this line is actually fed back from this clock from the clock number 2, I require a 1. So, again I assume that there all scan clocks. So, I will basically makes this line as a 1 and if you can look at the to justify this line basically if you look at the output of the second one. So, it is actually coming over here the output of this one is coming input of this and gate at e 2. So, basically I require a 0 over here.

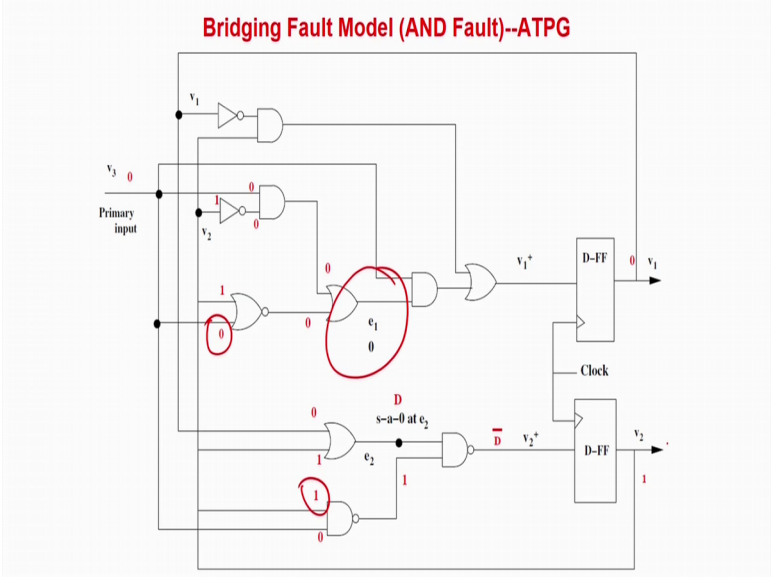
So, as how do I do this? Basically I have this all the flip clocks are scan clocks. So, I can just disconnect this two and basically I can make them 1 and 0 by the scan manner and then I can go for this approach. So, basically so, directly the flip flops are controllable that was what I have assumed.

(Refer Slide Time: 39:06)



So, basically this is what I propagate the value over here and this is basically the requirements over here. Additional constraints is something like this then for the and gate I require two zeros. So, basically this is one and gate, to make the output of this and gate output 0, I can just put a 0 over here my job will be done and similarly you can find out the from the figure let me just (Refer Time: 39:30) up that very easily you can propagate an justify again just re-emphasizing slightly that.

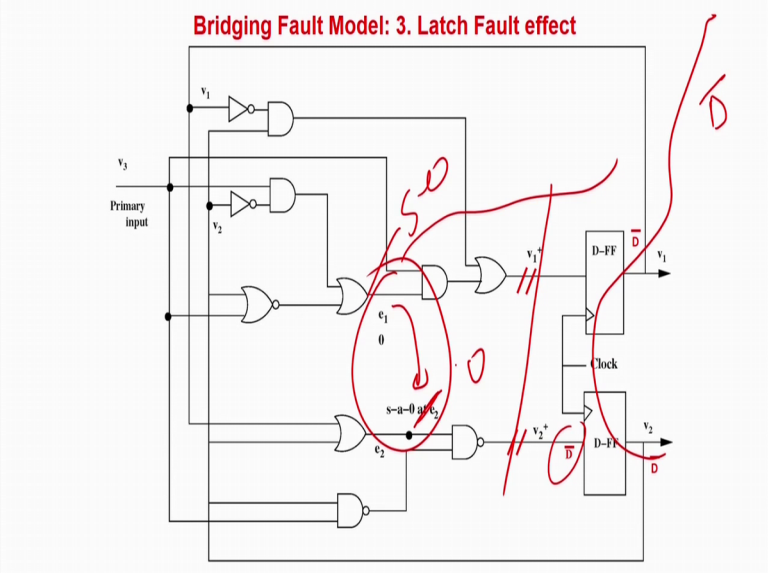
(Refer Slide Time: 39:32)



This additional stuck will require additional amount may require additional amount of in iterations, because more number of choices will be there and if you taking a wrong choice conflicts may happen let me do it in steps.

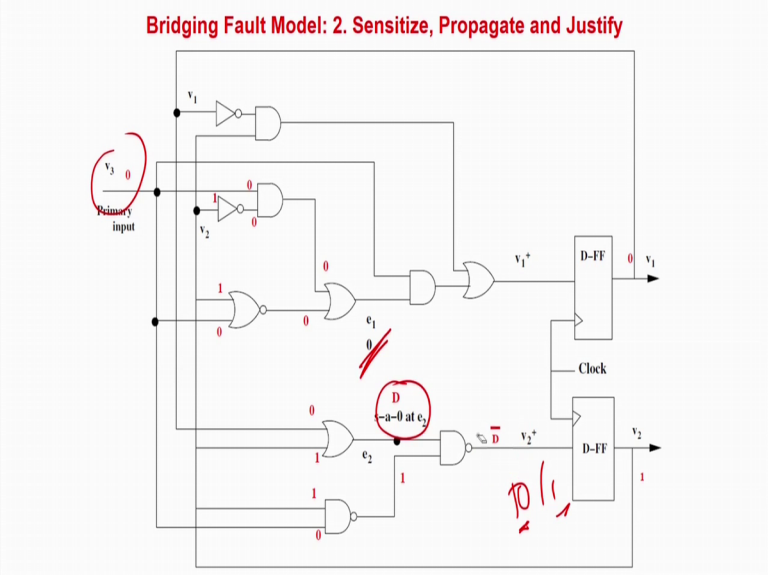
So, ATPG tells that basically I require a 0 over here and a 1 over here. So, I have to set these flip clocks (Refer Time: 39:49).

(Refer Slide Time: 39:49)



So, first step what you do basically is that you decouples the clocks and basically this has to be set by the scan. Next step state is basically if the clocks are already set and basically then you again we connect the clocks to a normal mode and then this is the test pattern input will be primary input will be 0 that you already you have found you can go through this slide in detail and you will be followed to find out.

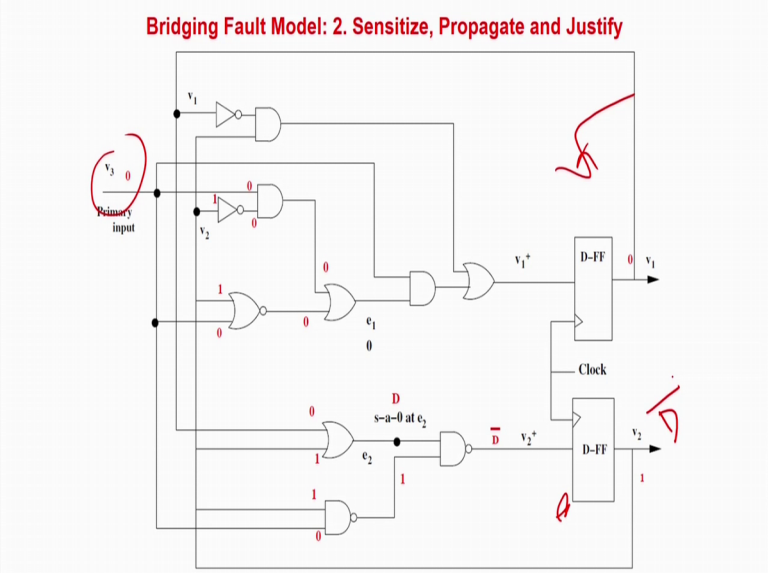
(Refer Slide Time: 40:07)



Then what is going to happen is that all the red marks signals will be validated D prime will be D bar will be appearing over here that is normal 1 sorry normal 0 faulty 1 will be the case and basically. So, if this there is this stuck at 0 fault over here rather in this case I should not call it as stuck at 0 fault over here because I am restricted this line to 0.

So, if there is a bridging fault over here you are going to get the answer 0 over here else the answer is a 1over here. So in fact, bridging fault is testing a stuck at fault with some additional constraints in some other line that is what is the idea then basically I am applying this fault very sensitize propagate and justify all these things are done and basically then you apply a clock.

(Refer Slide Time: 41:01)

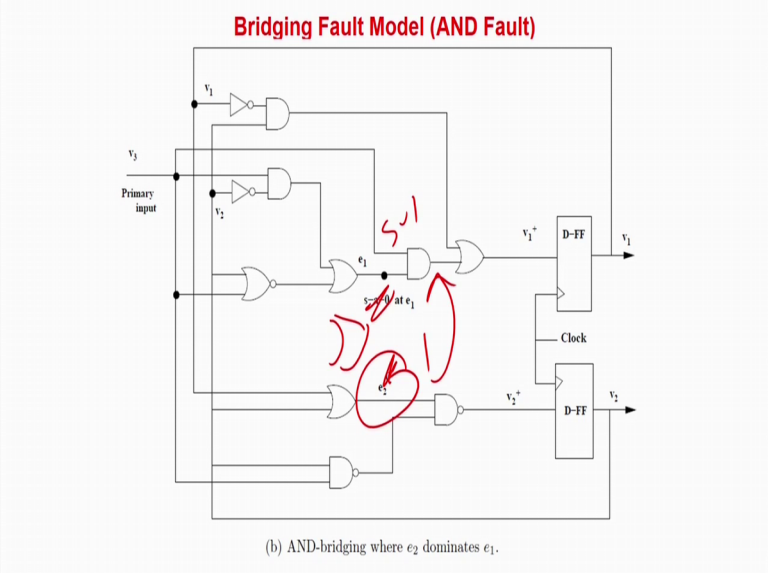


So, now the flip the flip flops already connected the value is applying over here you apply a clock pulse and the value will be d prime will be last over there. So, the D prime is last over here now again I have to cut out this flops because otherwise how do I get the values of this clocks flip clock value which has been fault values which has been lap last over here how do I get it out, I again make a scan connect and then I have actually propagate this value out of the scan chain that is D prime will be output and I can just find out that there is no or bridging fault over here or a or bridging fault sorry and bridging fault is present or not I can be take in which case e 1 is dominating the line e 2.

Similarly, you have repeat it for e 2 dominating e 1, which will be just dual of it in this case the fault propagation etcetera will be having over here and here we have to sorry this next case, basically will just reverse it out in the next case we just reverse out. So, in this case here you are going to have a stuck at 0 fault over here, you have to make this line explicitly 0 and the whole process will (Refer Time: 41:54) So, two steps you have

to do which are ex actually explicit mention over this figure this figure already we have seen.

(Refer Slide Time: 42:01)



We have just need a 0 over here explicit additional constraints, and there is a followed stuck at 0 similarly other way you have to make it a 0 over here and you have to find out the stuck at 0 fault over here, which is just the mirror image of this. And then again the other way you have to do you have to assume that there is a or gate over here and you have to just make the dual of it.

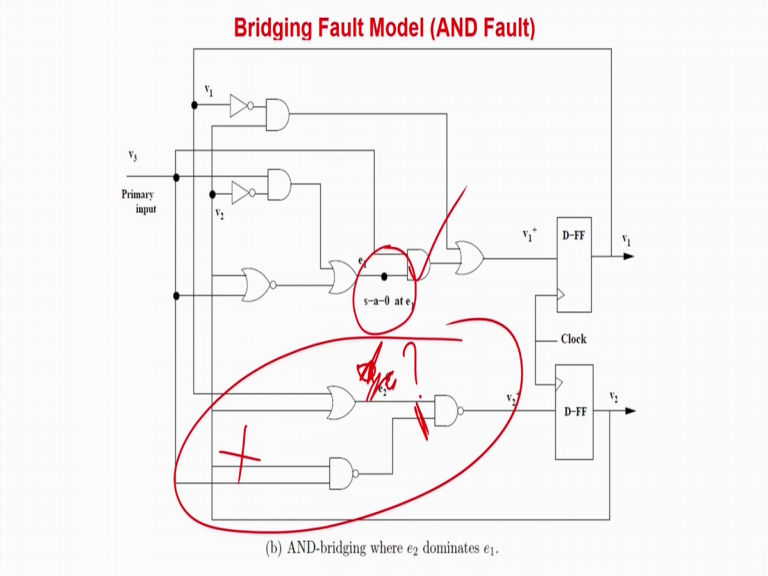
So, in this case you will be assuming a stuck at one fault over here and here you have to explicitly at the value one. So, that is a dual you can easily calculate. So, basically what happens you can observe again I am repeating the key terms here that, if you go for testing the bridging faults automatically we will find out that a lot of stuck at faults are already getting covered. So, what we do basically, we find out patterns and test for the bridging faults, then automatically which ever stuck at faults will be remaining we can go for the normal ATPG for stuck at.

So, that is a byproduct basically. So, you can you can easily find out by an intuition if I go for bridging fault test on the entire circuits stuck at faults will automatically get covered. But in fact, that test patterns generation a time in such a case will be large and in fact, there is not required also because not all parts of the circuits are generally highly consisted you can pin point at which are the regions of the circuit highly consisted and

which you case required to have a bridging fault where you can need to do a stuck at fault test etcetera.

So, that way the optimization happens in such a way, we go for the majority important within go for bridging fault automatically some stuck at fault will be tested and the remaining one, which you can go for the ATPG method first stuck at fault. But one interesting fact is that some people also try to find out in this manner that if I go for only stuck at fault testing.

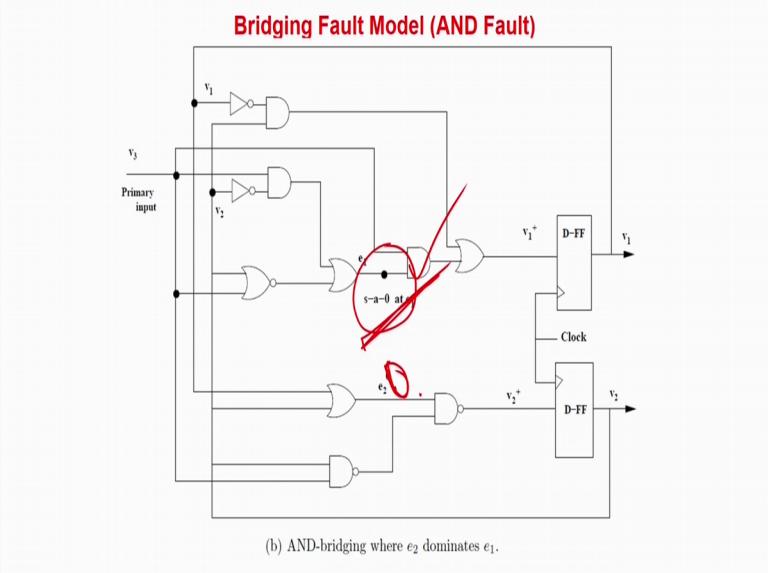
(Refer Slide Time: 43:37)



Say for example if I go for a stuck at fault test over here, some pattern will be applied over here of course, this is I will for me this using is x. In this case I do not bothered what is happening to the circuit, but by chance it may happen that this line basically has a 0 or as you are not bother about it may be either 0 or 1 so, by chance if it is a 0.

So, bridging fault is default tested with the stuck at 0 fault over here, but in fact, if it is not it is 1 then only the stuck at 0 fault over here.

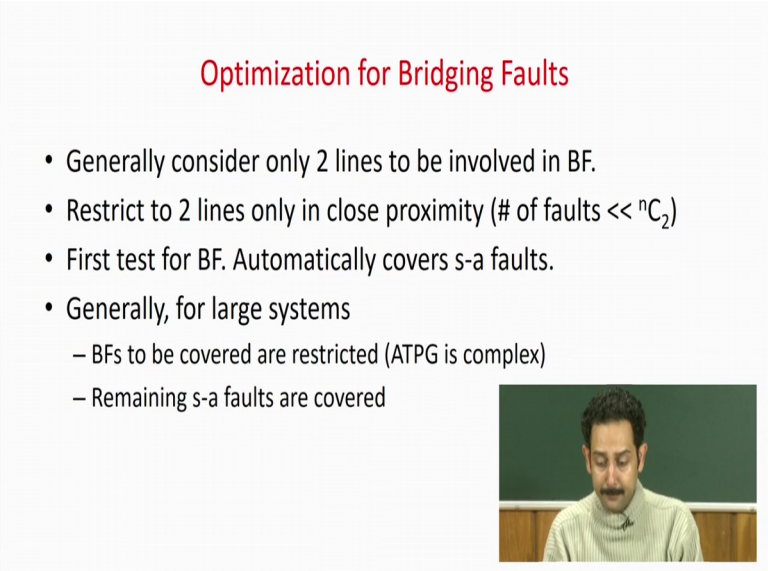
(Refer Slide Time: 44:09)



So, the test pattern basically which is resulting in a 1 over here and testing a stuck at 0 fault over here is called a bad test pattern or not a very good test pattern, because it is only testing a stuck at 0 fault over here.

But test pattern which is having a 0 over here and testing the stuck at 0 fault over here, we will call it as a good quality test pattern because it is actually testing both of them. So, our main goal in advance level of testing is to find out good quality test pattern. So, that it covers multiple faults at a single go or not only multiple faults, faults on advanced types in a single go.

(Refer Slide Time: 44:38)



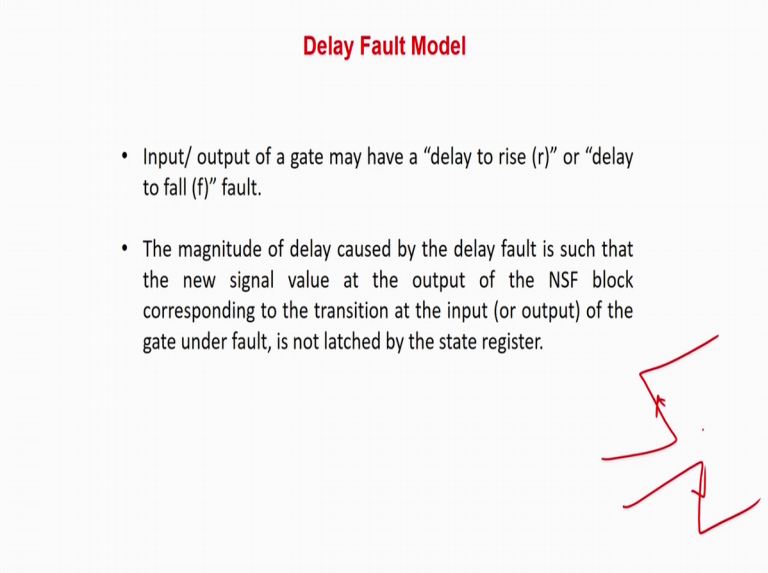
So, basically what as I was saying what are the basically optimization people do, they generally consider only two lines at a time and that also line to close proximity. Because otherwise if you are taking all combinations of faults n C 2 will be a very very large number. So, basically what you do you actually first test for bridging fault, automatic it will cover large number of stuck at faults.

So, basically for large system or complex system what we do? We cannot have a bridging fault of the entire part of the circuit we take some conflicts part or which design tells a more complicated as to be looked in to. So, we are going for a restricted; such in that case and remaining for we go for a stuck at fault models to be covered.

So, but this is you cannot assume that this is a the wide place or a buyable kind of a rules for a all kind of test circuit or test industry. If you will have the different rules and where people solve it is also a very secret for that industry that will not reveal it to everybody, but this is something call some kind of golden rules you people try to apply that taking only two lines restricted to region, basically you go for the bridging for certain regions and for stuck at fault you will going for the other.

But I mean I cannot say that no some venders in which case is circuits are highly compact in certain a one to apply bridging fault for the entire part of the circuit. In that case testing of the stuck at faults is the (Refer Time: 45:53) So, this is not a very golden or a buyable kind of rule, but these are basically some kind of thumb rule as I tell you with some generally say that two lines are considered for, but for many cases it will takes smaller window, but in smaller windows basically they take more number of lines we assume to be happening a coupling fault.

(Refer Slide Time: 46:14)



So, that so, another very important fault model is the delay fault model.

So, what is the delay fault model because now-a-days circuits are operating and a gigahertz level. So, you should know that not only the circuit is operating properly there is no any coupling fault, there is not even stuck at fault, but the answer should come in a predefined amount of time. The and gate should compute the value 0 0 2 0 or 1 1 2 1 and so, both for all the gates in a predefined amount of time that is the delay from one into the other end of the circuit should happening.

So, that it can make the speed requirement. So, delay fault model is the very important fault model for most kind of model circuits. So, almost all circuits now-a-days also cover delay faults test, but as we slightly see that now, the delay fault generation is also complicated as well as the fault application it was a very high cost ATE.

Because here patterns have to be applied to the gigahertz level; because if I am going for a fault what I am saying if I going for a delay fault test that is your apply some pattern and then you apply the next pattern and you should know that the and gate is computing the results at the near activate time.

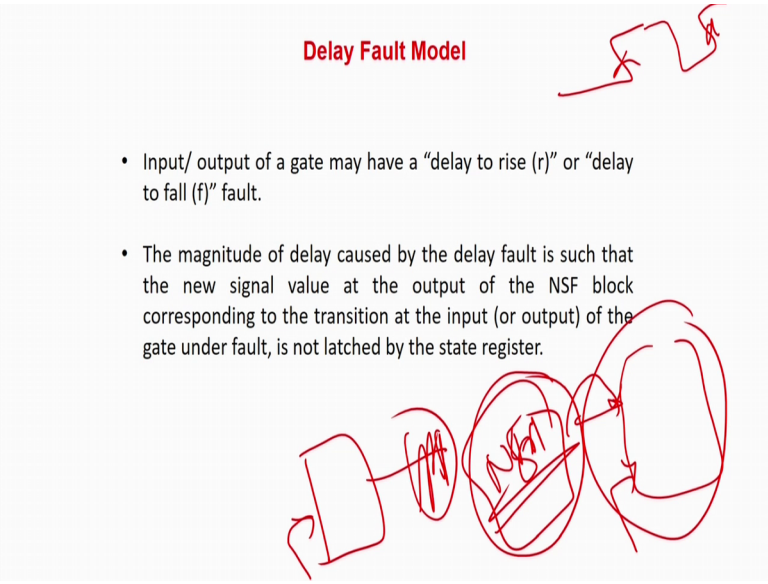
So, in that case; obviously, test pattern has to be applied as very first place, and also the data has to be latch very first place at the gigahertz level, which is typically the operating range of the circuit. If not able to do that then basically delay fault has no meaning. So, therefore, ATE is the very very costly in such cases and we will actually now-a-days applied for what do I call design for testability methods, that they do not exactly equal

and external equipments to do this a they put some internal circuits to apply those test pattern which will be looking at into the next few lectures, which you call design for testability optimization. But for today class basically we will try to see what the fault modeling what has to be applied and what is expected.

So, we say that the input output of a gate may have a delay to rise or delay to fall, basically all gates will be showing this or this rise or fall that is what is require and you should able to do it in the as speed required.

If it is, if you are able to do that and there is no stuck or bridging fault basically it is logically is proper and there is no up and down for such cases everything is in time and then you can know the circuit is having proper functionality and also it will be able to meet your speed requirements. So, the magnitude of delay cause by delay fault is such that the new signal value at the output of the n s the next function block, corresponding to the transition of the gate under fault is not latched by the state register.

(Refer Slide Time: 48:32)

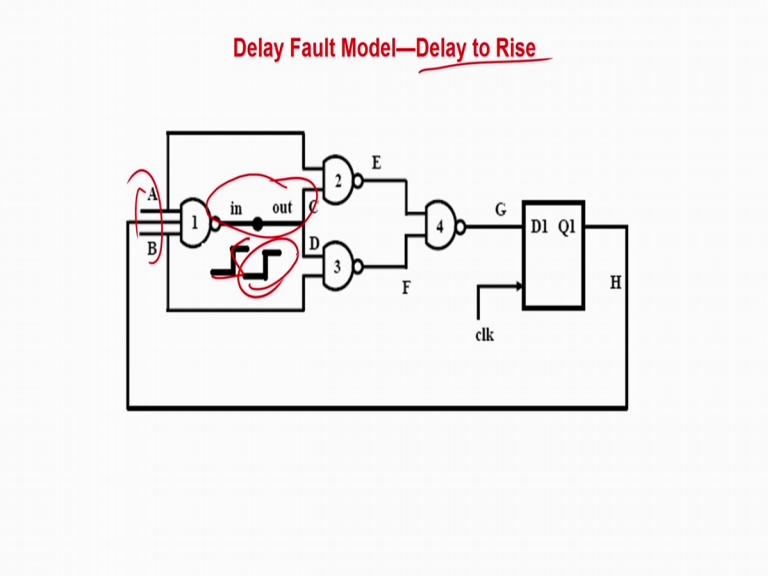


That means, what generally we have a flip clock yeah we have lot of combinational circuits over here we have another combination circuit over here. The magnitude of delay fault such that the new signal, this is actually called the NSF we may call it next state function block or some (Refer Time: 48:45) block. So, you have apply some inputs over here then basically this is your clock this is you clock. So, basically before the arrive.

So, some new some patterns comes at this clock at this clockage some input pattern comes over here. Then this NSF block of the combination clock will compute the values, but before the arrival of the next clock pulse this data should be available over here before the set up time of the flip clock so, that the exact value can be last. If you are not able to do that; that means, there is a delay failure in some gates or some paths of the nested function clock.

So, that is the actually a delay fault will take an example.

(Refer Slide Time: 49:17)



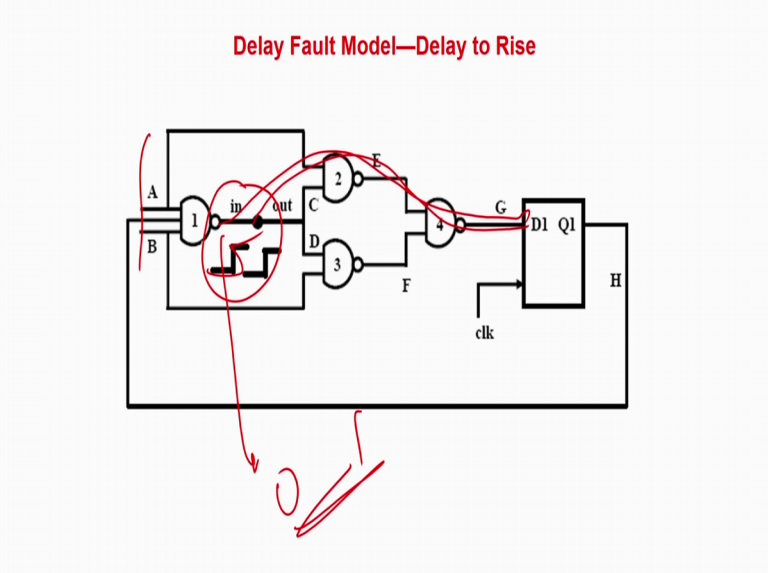
So, this is an example of a circuit we are assuming lot of example can be provided, but I am mainly this course we are trying to give you the basic philosophy. References will be provided in the course material or the course pages in the site you can easily beat for the advance part because this is course; we slightly require some advances which we are basically made into practice for two title modern day circuits and the modern day failures. So, we will try to give you idea of the main clocks main basically clocks of the how to do, and the other parts you can very easily read and you can we can do it.

So, here we are going to talk about delay to rise. Delay to fall will just be the mirror image of it. So, what happens delay to rise means when you are assuming that only this line because we have to in fact, try this delay to rise and fall, we have to do it for all the gates. So, we are assuming for gate 1 there is delay to rise fall; that means, when you are

change the inputs such that the output of the gate should change some 0 to 1 it is not happening in this proper time, there is some slightly delayed and it is happens over here.

So, this is basically a delay to delay fault module or delay to rise fault, we have to fault model will be very similar thus the reverse of it. So, we have to verify this. So, of course, very basic philosophy we have till learn about around 2 to 3 hours of lecture in test till now. So, this is and from the intuition also you should have some you should apply first 0 over here and after sometime you should able to apply a 1 over here that is the sensitization path and of course, the value of the propagated to the output.

(Refer Slide Time: 50:35)

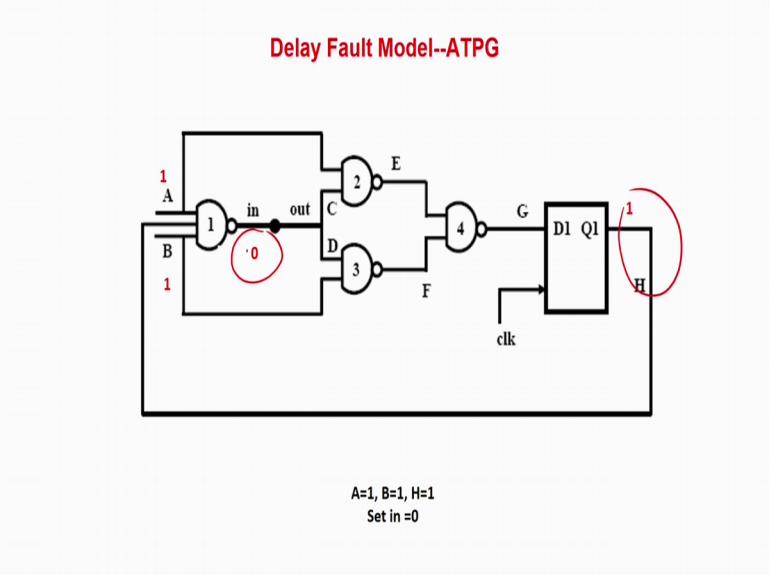


So, it has to be latch that means, but the change from 0 to 1 at this has to be done in a very fast manner. So, first test pattern should be such that 0 is applied over here very quickly of we have to another test pattern. So, it should sorry. So, it should become a one and that should happen in a quick amount of time reasonable amount of time by gate number one because we are assuming the single fault model in all cases by the way most of the fault models assume a single fault; that means, only in this in outline we have a delay fault.

So, if there is a not by the delay, the value will be propagated over here and you are able to capture the value in the right manner in clock otherwise there is an delay fault which can be captured.

So, 0 and immediately we have to make it a 1.

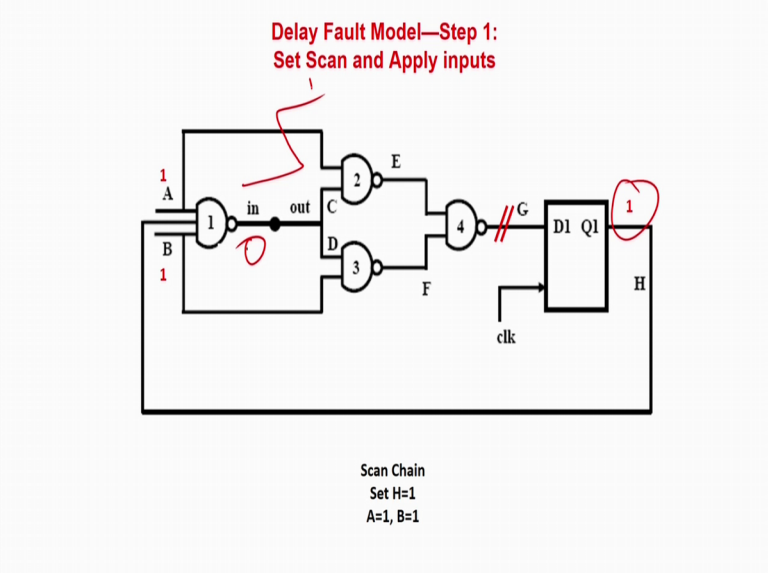
(Refer Slide Time: 51:21)



So, first have to make the value at this line as a 0. So, how can I do it? This is an and gate. So, an and gate output is 0 only if everything is a 11 and one. So, you can see a and one are primary inputs, which I can easily make 1, but this latch which is the feedback of this clock also has to be made 1.

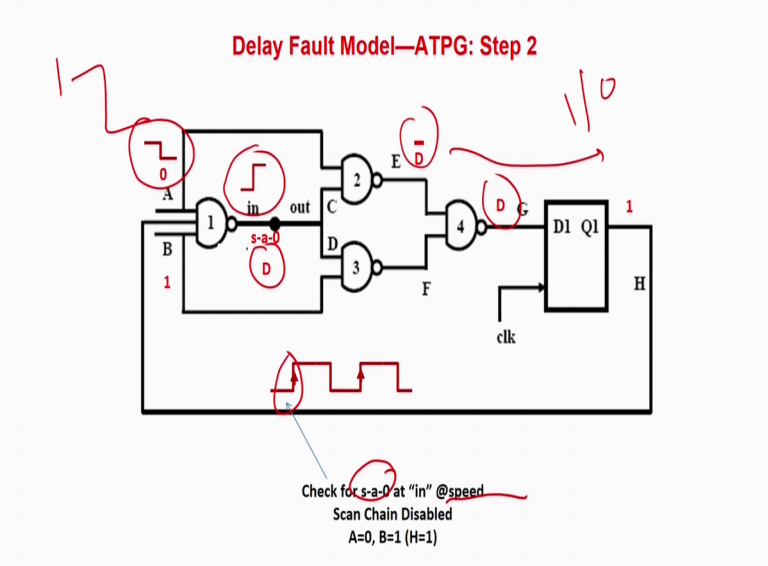
So, what I have to do is that I am first going for the ATPG. So, I know that this is what the first stuck in be applied and for that basically I require the flip clock to be having a value 1 which I can easily do by a scan chain.

(Refer Slide Time: 51:45)



So, which I cut the value cut the flop by a scan and then I i for this scan chain I actually make the value one and apply 11 and 1. So, of course, and gate 1 1 and 1 it basically a 0 over here.

(Refer Slide Time: 51:59)



Now again, I trouble the clock with the circuit and then what I do? In the this is the first case 1 1 clock scan chain and this is as 0. So, 0 is over here then what I do basically I reconnect the clock, then I actually make the what actually call it is test for stuck at 0 because; obviously, we are going for test for this. So, first you make the 0 and then you apply a one and this one should propagate properly; that means, what you are testing a stuck at 0 fault, but at speed that is what is very important.

You are testing a stuck at 0 fault at the line, but at the very very high speed. So, what I do? At this clockage at one clockage I will make the line a equal to 0 from 1 to 0. In fact, you can also do same for b. So, if you do this after certain amount of delay the in will become a one because an and gate. So, 1 0 1 0 0 1 whatever if any one of the line is a 0. So, the an and gate will give the answer as a 1.

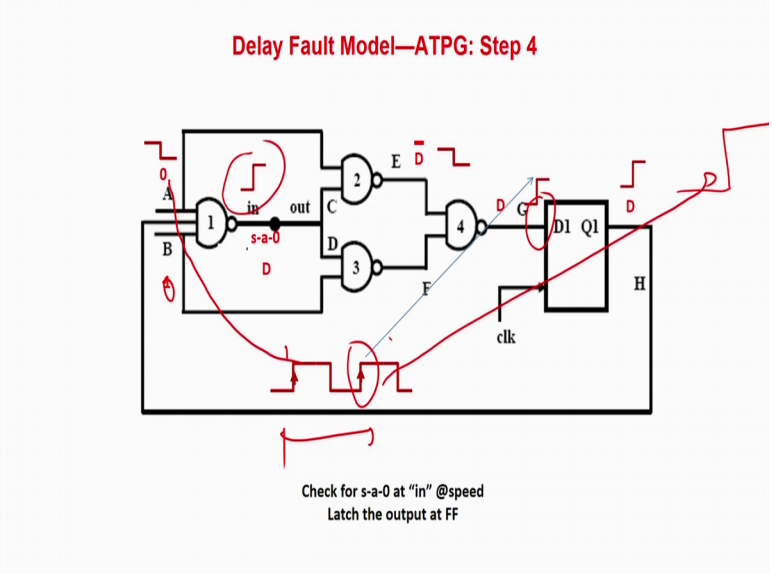
So, there should be a rise. But if this is so, much delayed; that means, this is stuck at fault. So, if it is it the it the right happen and in the proper time then the value will be propagated to the out to the basically clock and it will be captured, but it cannot rise; that means, the they slightly delay and then what is going to happen? This one it will happen it is not the line stuck it is not line is not permanently stuck it will rise, but the rise time will be so high that basically it will not be last in the second clock period it will be lost it will be lost last at the third clock pulse or even less even a letter. So in fact, for that case basically it will assume that this is the stuck at 0 the rises happen. But it will so delayed let it cannot be less by the flip clock.

So, basically it is nothing, but a stuck at 0 fault. So, what I have to do? Stuck at 0 fault means I have to apply a 1, which I can do by having A equal to 0, B equal to o1 and H equal to 1. So, this is D and I have to assuming this line as propagating the value. So, inversion will be D prime and again have the inversion will be D. So, basically what is the answer this is 1 in normal and for (Refer Time: 53:57) this is what is to be expected.

Now, this scan chain is disabled I am making A equal to 0, but I am doing extremely fast that is from 1 to 0 this a and doing the level of (Refer Time: 54:05) So, that is that is why I require a very fast ATE do that. So in fact, external test are having such high frequencies very expensive. So, what we try to do many times you put the built in silk set circuit inside the circuit itself, which you are calling bits architecture that will actually give the in pulse which will see later.

For the time being let us assume that such a fast signal can be give it, but assume that you are also know that applying first a fast signal is a from an ATE to a t cost is very high. So, we do that. So, what basically it me just remove the write as from the slide. So, that it become clear. So, that is what is the case and basically at this clockage all the stuff is happening at this clockage.

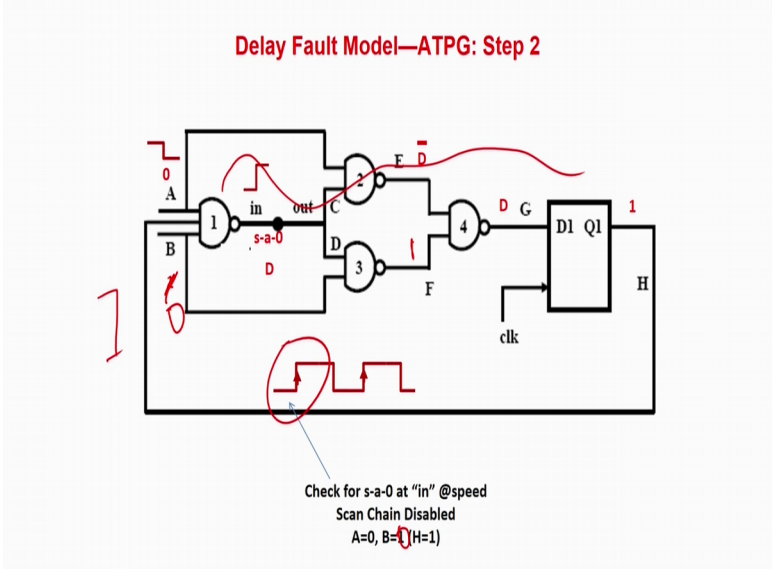
(Refer Slide Time: 54:48)



Next what happens if everything is proper by the next clockage you are going to get the value of a one it will very fast. It will go from 0 to 1 from here it will be falling from 1 to 0 and again basically it should come over here and in fact, basically you should get this rise over here. So in fact, in the second clockage so, basically if it is a an and gates I should get a 1 over here so.

In fact, slight slightly I should backtrack over here. So, this is the value is propagated to be justified, I required a value of 1 over here to require a value of 1 over here I i should I cannot have a 1 over here because that will create a problem in fact I should make a justification I forgot to do.

(Refer Slide Time: 55:23)



So, just it justify you make the value of 0 before the an and gate. So, it will be 1 so. In fact, I should say that in fact, b should be equal to 0. In fact, b was if you make b equal to 1 that will be a conflict basically. So, I make b equal to 1. So, basically this answer is a 1. So, this is a proper solution.

So, the this cases 0 then actually I have to make A equal to B equal to 0 then I have to connect, but in fact, this B may not be as speed that is what is the idea because generally I will having testing for the propagation of this path, the B is from 1 to 0 again this can be made (Refer Time: 55:59) slow pattern mainly I am trying to heat the circuit on this on this path. Because I am assuming that this is what is the path B followed for the as speed testing. So, anyway so, this line is for just justification and line a e in out e and g is for propagation and sensitization do each. So, that I with a 0 over here and value is a 1 over here.

So, basically what happens? So, in this case, this rise should happen and this should happen before the arrival, that this is the duration this is duration. So, this change actually should happen just after this because this point is triggering it and basically I am sorry I should say that this is actually trigger point. So, this is basically trigger point. So, this is trigger point and this change is by the delay has been targeted.

So, this should happen so, fast that all this thing should happen before the arrival of this clock puls. So, if the clock before the arrival of clock pulse everything is in proper, then you are going to getting the value of d over here and that will be actually last by the flip clock over here; that means, at this clockage; that means, the value of normal it is one failure it is 0.

So, if the d is appearing and it is the one that raise has happened and it is lag by the second it; that means, this path this fault in this path in fact, actually has a no delay in doing it. In fact, you can also find out that, if this is true then this is also having no fault and this is also having no fault; that means the gate e does not have a delay to fall fault and again the gate four does not have a delay to rise fault.

So, by the sudden many faults delay to rise and fall fault delay to fall faults have been detected. Basically this fall the delay to rise delay to fall delay to rise would be easily detected by this pattern, but in fact, anyway I must mean we are just highlighting on this. So, let us look at it. So, basically what happens it is not a stuck at fault. In fact, basically

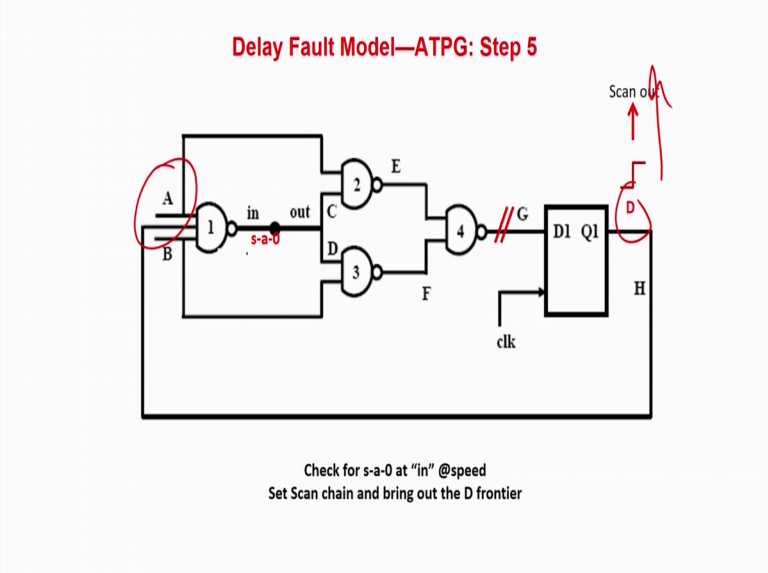
what is going to happen here is that after certain amount of delay basically even if the delay fault is there this is the rise, but it will rise much after this one so in fact, this will assume that there is a stuck at fault.

So, basically rise will be there, but still it will the d flip clock will assume, it as stuck at for because it will only latches value at the clockages and in fact, that is require because all computation have should have to (Refer Time: 58:11) two clockages. So in fact, what is the delay for model? It is very similar to testing of a stuck at fault model or bridging fault model, in which case you should have in the fault path you have to give a sensitization.

Sensitization is nothing, but you have to first delay to rise means you have to apply a 0 and next part you have to make a 1, but that one has to be done basically in a very fast manner at the level of peak, and then you have to latch at the value at the flip clock also at the same speed. But anyway that is not a problem because that is the normal part of the circuit. So, it will be generally operating at the gigahertz level, only problem is there this has to be applied at a very fast rate that is why ATE cost becomes high and so forth. So, if you are able to latched at the value as the output, there is the no delay fault for rise delay fault otherwise there will be a delay fault.

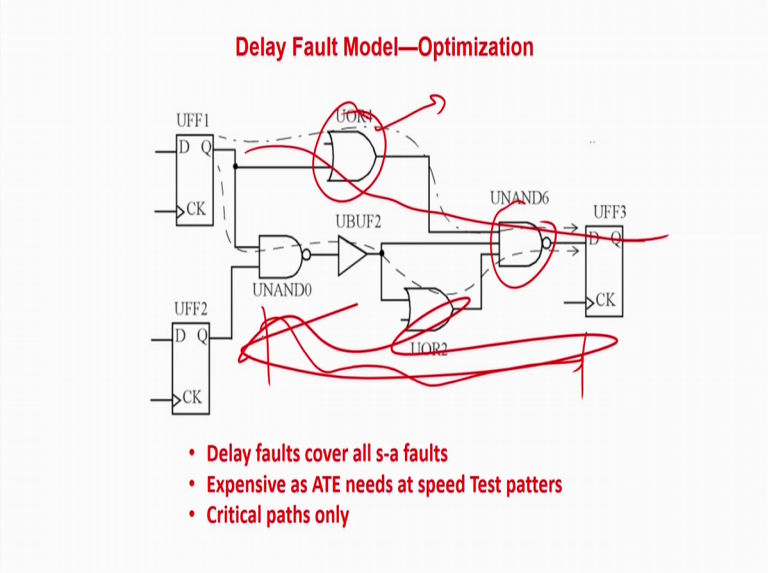
So, you can also see over here, that if you are going for delay fault model raw the stuck at 0 and stuck at faults will be by defect to covered.

(Refer Slide Time: 59:06)



So, that this is the other part of it. So, basically the value of b was less standing on putting into scan out, which you can again do it a slower piece. So, only problem is that the application of test pattern as the input as a very fast rate, but all others like the d front you are taking a output scan can be very slow approach. So, only this is where the cost comes.

(Refer Slide Time: 59:24)



But anyway we are talking about the optimization. So, basically what? Here also you have see the beauty that we are going for the testing of a bridging fault, whether you are testing for a delay fault basically stuck at fault automatically starts getting covered, but not the not the other way.

That if you want doing only test for cover for a stuck at fault this is for may be covered or may not be covered delay fault, will not be covered because you are not applying test pattern as speed. But if you are going for the advance fault model test, which are required for modern day circuits because they are manufactured in this micron level our stuck at fault automatically start getting covered, but what is the optimization for delay faults.

Because optimization at the bridging fault already we have seen restricting the line distinguish the regions, but here we are not in case of delay fault you understand that it is a very costly affair because patterns have to be a applied at speed. So, what we do? Basically we try to apply test pattern only in critical paths. Critical path means the

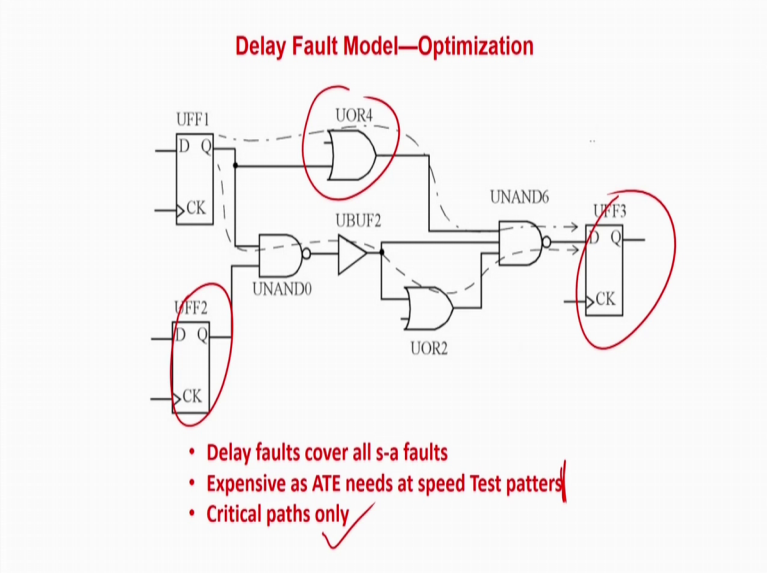
longest path. If you look at this circuit this is the path which is the longest path means more number of gates are involved. This is the path which is the slightly shorter.

So, if this gate has certain delay this is not that much of a concerned because generally speaking, the delay of this gate will not be such high that we will undertake all the delays of the gates and still it will be more higher with the fault; that means, if there is a delay fault here, delay fault here, delay fault here or here will be more concerned because the path is long it may easily evaluate the timings, but this gate can have a delay fault, but the timing will delay will not be so, high that will actually take up all the long path delays into account.

So, generally we try to find out something called the critical paths which are the longest path and then we try to see whether the delay faults are happening there or not. So, this is an again an optimization because if you are trying to take all paths it is neither the require because I told you in this example, whatever fault might be there in this path in this gate as this path is shorter it cannot be higher than all the gates taken to in the other path.

So, you taken in the long the longest path which is designer will be tell you as the most critical paths or near critical paths on those go for the delay fault test. So, that your test time and your test cost also remain lower because delay fault test is expensive, because a t has to apply faults are very very fast ways patterns at very very fast way. So, we restrict it to very specialized regions of the circuit, where you know that the delay increases the latching of data.

(Refer Slide Time: 61:39)



From this flop to this flop may be delayed or it will be hampered and so on stuff will has to be organized in this manner.

So, that is why the optimization is that we do only for critical paths because ATPG ATP ATE cost is very very high and of course, as you have seen that only testing of the delay faults will easily covered stuck at fault. So, what we seen in this? So, you have basically seen that advance fault models are very much required if we are going for testing of circuits at deep sub micron level which is the de facto standard of fabrication for NOCs and SOCs interestingly.

If you go for testing of the advance fault model automatically some stuck at fault models will falls stuck at faults will be tested. What we needs to be done you go for the remaining at the later phase and so, that you for a 100 percent coverage. So, this for idea go for advance fault model test in a restricted manner and in a optimize manner. So, that do not you do not over space on testing and then find out by de facto what are the stuck at automatically get in tested and remaining one you do.

So, this is how basically the modern test pattern generation optimization happens by finding out good quality test pattern test. In fact, everything every test pattern whether it is a stuck fault delay fault etcetera nothing by the some input patterns at some peak. So, among the said whose are the good one, the good ones are those which will covered multiple different type of so, such patterns has to be selected and in this lecture basically you have try to give you an idea on this.

This is actually a very new research area that how such in important fault models can be covered how they can be optimize you have given idea of the very basic and how people start about. So, in the differences will try to give you more ideas on that which you can read through and get more advance how people are dealing in the modern days.

In the next lecture basically we will be trying to look at some aspects of test pattern type of compression, that if you are by huge data how can we compress it and try to make the test data size limited. And then we will go for see how additional structures can be put in the circuit so that the test time and test optimization can be developed.

Thank you.