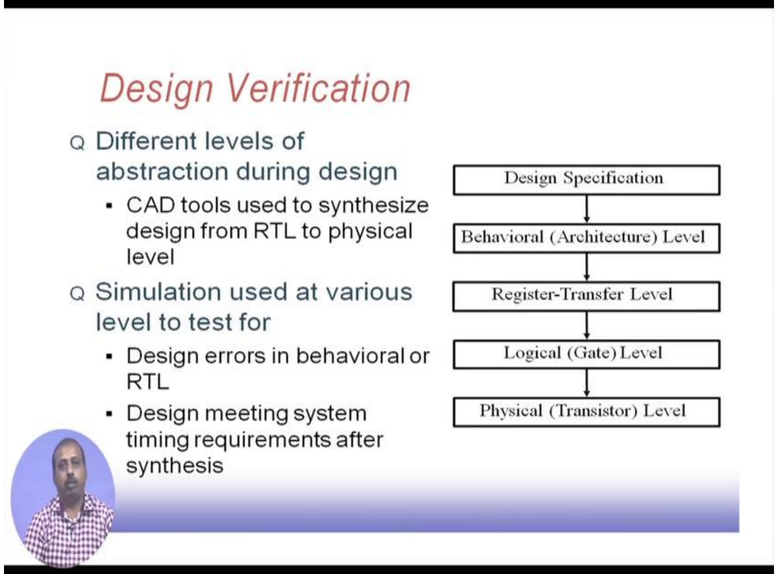
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 02
Introduction (Contd.)

(Refer Slide Time: 00:22)



So, the design verification stage, so what we do we verify whether the system has been designed properly or not. So, how do we verify? So, there can be 2 approaches for that, one is by means of some formal verification tool that will be proper be verifying some properties on the design system. And another easier technique for doing it is via simulation. So, simulation so this will be applying some stimulus to the description, and it will see whether the result that is produced is as per the desired one. So, simulation tools are available and simulation is much easier to do, because formal verification so we need some formal logic background, so that all the designer may not be having. But the simulation is more or less now the standard technique for doing this thing.

So, design errors in behavior or the register transfer level description, they can be decide found out by the simulation. So, at this behavioral level and the register transfer level, so they are actually we simulators are run, and the same stimulus is fade to both the stimulus both the stages as a result we can get the response from them, and check whether the responses do match between the 2 stages. So, that if the response is match

that means this translation from behavior to register transfer level that has been done correctly. If it is not then there is some problem in this translation process, so that has to be rectified. So, that way these design errors can be detected.

Similarly, after this rejecter transfer level. So, we have got this gate level description, where this if the total RTL description is flatting into gates and we have got a gate level net list for the system. So, for this gate level net list, so we can again the run the simulator and see whether the system behavior is matching or not. And finally, at the physical level the transistor. So, there are some transistor level simulator like spice, so that can be run at the lowest level, and then we can see whether it meets the requirements of the system or not. So, that way at every level the simulators are used wildly for getting confidence about correctness of the system. But still up to this much we have got only descriptions. So, it is at the physical level also this nothing, but a transistor level description of the system, this is not the actual chip that is manufactured.

(Refer Slide Time: 02:54)

Yield and Reject Rate

- We expect faulty chips due to manufacturing defects
  - Called yield $yield = \frac{number\ of\ acceptable\ parts}{total\ number\ of\ parts\ fabricated}$
- 2 types of yield loss
  - Catastrophic – due to random defects
  - Parametric – due to process variations
- Undesirable results during testing
  - Faulty chip appears to be good (passes test)
    - Called reject rate $reject\ rate = \frac{number\ of\ faulty\ parts\ passing\ final\ test}{total\ number\ of\ parts\ passing\ final\ test}$
  - Good chip appears to be faulty (fails test)
    - Due to poorly designed tests or lack of DFT

So, as a result when we go to the manufacturing stage at that point at after each level of manufacturing, we need to test whether the manufacturing process is correct or not. So, design this simulators they can catch the design errors, and this testing processes they can catch manufacturing errors. So, we have got some parameters like yield. So, it is defined as the number of acceptable parts to the total number of parts that are fabricated.

So, this ratio is called yield; naturally we will expect as in the manufacturing process thus that yield should be maximized.

Now; so, the number of chips that we discard that you had to discard as faulty, they should be minimum. Now there are 2 types of yield loss that can happen Catastrophic; catastrophic they are due to some random defect. So, it maybe some defect in the vapor at some point. So, that cannot be avoided. So, that is one type of problem and the parametric problem because some parts may not be fabricated properly as a result there is the gate of a transistor might not be the of exact link that we were looking for, as a result it leads to some problems. So, these are known as parametric variations. So, they are the process variation and they come under this parametric yield loss. So, the result undesirable result during testing. So, these are the problems; like if my testing process is not proper then these things can happen. If faulty chip may appear to be good; so if it is happens then a faulty parts, so that will be shipped as a good chip and when this chip will be put into field operation, at that point of time at system level it will give problem.

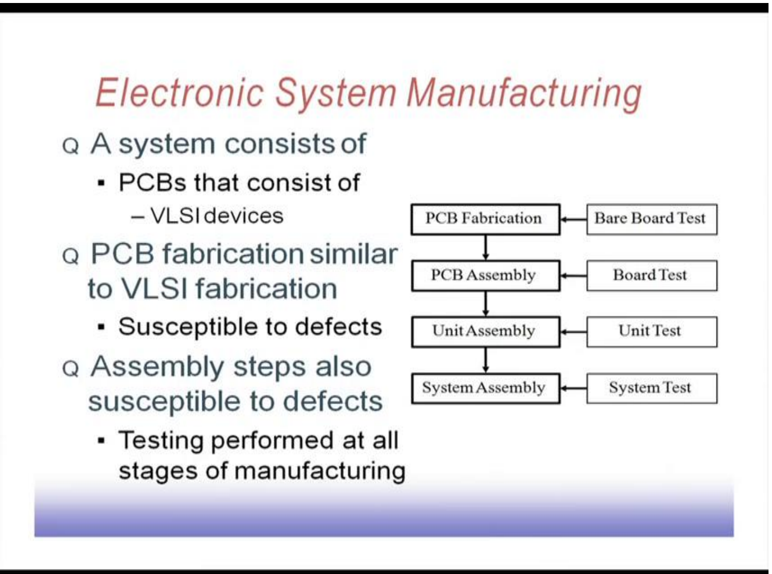
So, at a system level we have to take care and by the rule of tense we know that by the time the cost of this miss testing will be very high; because that will be our that may cause rejection of the full system. So, this is the reject rate. So, this number of faulty part passing final test that is they declared falsely as correct, and the total number of parts passing the final test. So, it is the number of faulty part that are passing, the ratio of faulty parts that are passed, that has passed the final test. So, that is the reject rate and another problem that can occur is some good chip that may appeared to be faulty.

So, chip is actually good, but they failed the test. It can happen due to various reason major problem is with power and temperature by issues like say maybe that it consumes good amount of power as a result that temperature also goes up and that delay variation occur. So, and also some points in the circuit if. So, it may not be very easy to test and we may try to taste it and we may fail. So, as a result we that may also be taken as some good chip appearing to be faulty; so this is also an undesirable thing. So, that should not happen.

So, this can happen due to poorly design test, maybe the test pattern that we test pattern that we apply. So, they are not sufficient to they can they have the problem with the good responses for some fault, even if the fault is not present. So, it sometime maybe creating

difficulty and lack of this design for testability mechanism. So, that may also cause some good chip to appear as faulty. So, these are the 2 things that we would like to avoid.

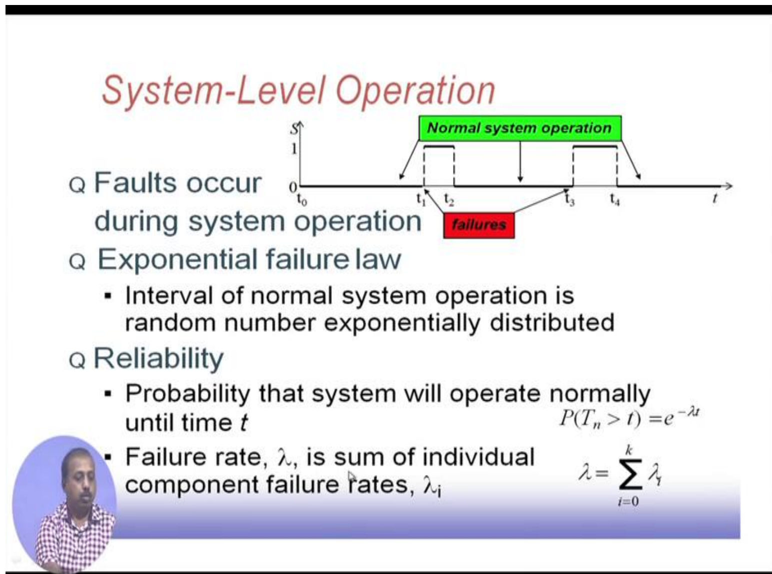
(Refer Slide Time: 06:39)



Now, in the system manufacturing process; a system consist of PCB's that consist the PCB will again contain this boards. So, that board will be they are actually the boards so on that board will have the VLSI devices of the chips. So, PCB fabrication is similar to VLSI fabrication, because in the VLSI fabrication we are putting several layers of polysilicon and diffusion and all that. So, here also we are doing same thing at different parts we are putting the chips, and then we are making copper connection between them. So, they are susceptible to defect like 2 lines running parallely on the PCB, so there may be cross talk. So, there may be interference between them. So, as a result it may create problems, so at some point of time that copper line that we are drawing, so there maybe some break etcetera.

So, this is this is necessary. Similarly assembly process so that also susceptible to defects, so testing process has to be there in all these stages. So, PCB fabrication level, PCB assembly level, unit level assembly and system level assembly at all this state stages we have to do this testing. So, at PCB level board test, PCB assembly level also we have got board test, now then at unit level unit test, and system level system test. So, if we can detect fault early then we can avoid the losses.

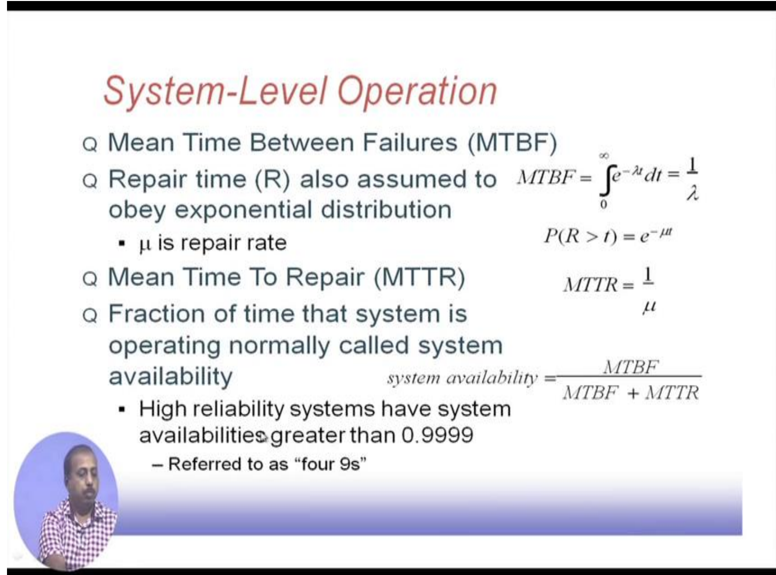
(Refer Slide Time: 08:11)



So, at a system level, so faults will occur during system operation. So, if say this is the; if we plot the time the lifetime of a system. So, over the lifetime maybe from 0 to time t 1, from t 0 to t 1 there was no problem with the system that is normal operation. Then a time t 1 some problem occurred and it required some time t 1 to t 2 for its recovery. So, it may recovery maybe from various angles, it may be some transient fault or it may be that there is some fault mitigation technique incorporated into the system so that takes care. Again from t 2 it starts operator or maybe the maintenance person, so take some measure by which this is corrected the system is corrected and then from t 2 to t 3 again it works normally then a time t 3 it fail. So, this is the; so these times t 1 to t 2 then t 3 to t 4, so these are the failure time, so at this time the system is not available.

So, there is an exponential failure law. So, it says the interval of normal system operation is a random number exponentially distributed. So, if we want to find the probability that the system will operate normally till time t, then we have got this probability expression that after till time t it will be this if time of failure is more than t. So, this is given by e to the power minus lambda t; by this exponential failure law, so the failure rate. So, since my system may consist of a number of chips. So, for the portal system, so I can say the failure rate is the sum of individual failure rates. So, total lambda is given by some of these lambda is. So, if I have got k number of components, so this sum of these k components.

(Refer Slide Time: 10:09)



Then there is mean time between failure and MTBF. So, we to find it, so we can say that this repair time is also assumed to be it will follow this now exponential distance. So, MTBF is given by this expression. So, it is in integration of that failure time, e to the power minus lambda TDT. So, if we integrate over 0 to infinite times. So, this is 1 by lambda. So, this meantime between failure is 1 by lambda; and repair times. So, this is also assume to be exponentially distributed, and if mu is the repair rate then the meantime to repair MTTR is 1 upon mu. So, system availability that how much time the system is available, so this is between basically the meantime between failure divided by the total time for the system. So, these MTBF divided by MTBF plus MTTR.

So, this is a fraction of time system is operating normally. So, that is system availability. So, high reliability systems, so they should have these availabilities more than 0.9999. So, it is a very high requirement that we have, that a system must be available almost 99.99 percent times. So, this is referred to as law of four 9s because 0.99999 that is why.

(Refer Slide Time: 11:31)

System-Level Testing

- Testing required to ensure system availability
- Types of system-level testing
  - On-line testing – concurrent with system operation
  - Off-line testing – while system (or portion of) is taken out of service
    - Performed periodically during low-demand periods
    - Used for diagnosis (identification and location) of faulty replaceable components to improve repair time

So, at a system level testing so testing is required to ensure that the system is available. Now there are the types of system level testing like online testing and offline testing. So, online testing is the when we doing the system the system is operating and simultaneously with we are doing the testing. So, we capture some data from the system operation and try to analyze and see whether the system is working correctly or not. So, that is the online system and offline testing. So, this is here also the system is operating, but we stop the operation temporarily for some time and then we do the tasting at that time.

So, when the system is not doing some high demand job it is doing some low demand work then we can suspend that low demand work for some time, and we can do this testing for that. So, this can be used for diagnosis; so of the faulty replaceable components that we can replace and that can improve the repair time. So, this system level testing is done both online and offline. So, there are test techniques for both of them.

(Refer Slide Time: 12:47)

Then comes the problem of test generation. So, what is a test? A test is define to be a sequence of test pattern that is a set of pattern, that they are also known as test vectors that are applied to the circuit under test; and we apply the pattern of a pattern set and then you see what is the output. So, for set of input patterns there will be a set of outputs, now the set of output so they will be checked with the correct responses.

So, if we find that the responses are correct the same as the called correct response or golden response, we say that the circuit under test is functioning properly, so this is tested. Now there can be many approaches by which we can do this testing one possible approach is exhaustive testing. So, we apply all possible test patterns to the circuit for if a circuit got 10 inputs and assuming that all these 10 are so they are binary input, then we have got all possible patterns. So, 2 to the power 10 different possible patterns that can come as input to the system. So, for all this 2 power 10 patterns. So, we check whether the output produced is as desired or not. So, that is known as exhaustive test testing. So, all possible now test pattern are applied to the circuit and the responses are checked.

Naturally the cost of such a testing is very high like for 10 input I can apply 2 power 10 patterns. So, so if I have got 100 inputs, I have to apply 2 power 100 pattern, the 1000 input 2 power 1000 patterns. So, that takes the test cost to be enormously high. So, we cannot go for that. So, exhaustive testing is normally done for small systems and which

are very critical. So, maybe number of input are less, but the operation is very critical. So, you can go for exhaustive testing.

Functional testing; so testing every truth table entry for a far combinational logic cart, so we can go for at the truth table entry, where it says that it is a 1. So, we apply those patterns only and see whether it gives work or not. Similarly over the when the truth table says it is 0. So, we see apply see whether it is giving a 0 or not. So, they are not practical. So, both this exhaustive testing, functional testing, so they are not partial. So, what is happing is functional testing we are looking at a functional level. So, it maybe not be true at a gate level operand. So, entire combinational logic so we are testing that way. But for exhaustive testing, so it may be testing each and every gate in the circuit exhaustively. So, that way it we there is a difference, but in both the technique. So, they are going to take exponential amount of time.

So, fault cover. So, what is the way out? So, way out is to apply only a subset of text pattern not all this exhaustive pattern. So, if we apply a subset of test pattern naturally we cannot expect that all that possible faults in the circuits they will get tested, only a part of it will get tested. So, that gives us a measure called fault coverage. So, it that is how many of faults can be deducted by the set of applied text vectors. So, if we want that we this fault coverage should be as close as 100 percent, but it is very difficult to achieve.

(Refer Slide Time: 16:18)

- Fault coverage for a given set of test vectors $\text{fault coverage} = \frac{\text{number of detected faults}}{\text{total number of faults}}$
- 100% fault coverage may be impossible due to undetectable faults

$$\text{fault detection efficiency} = \frac{\text{number of detected faults}}{\text{total number of faults} - \text{number of undetectable faults}}$$

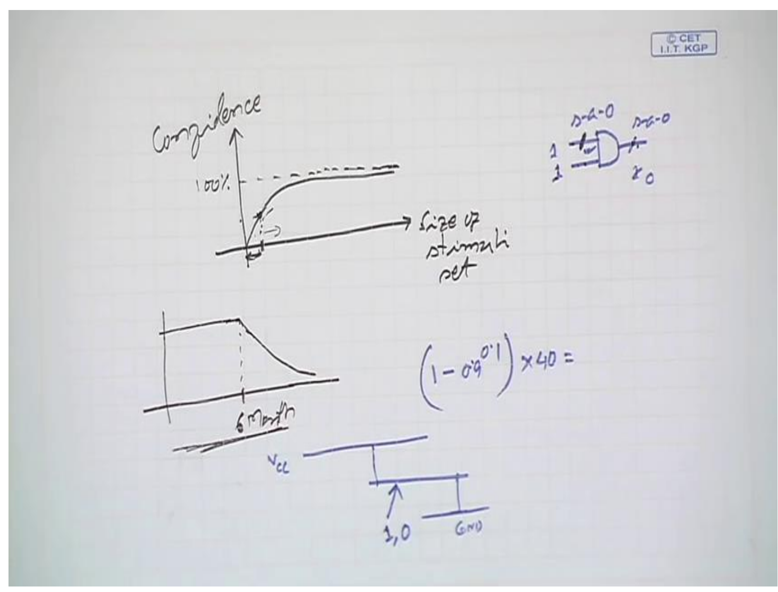
- *Reject rate* = 1 − *yield*$^{(1 - \text{fault coverage})}$
  - A PCB with 40 chips, each with 90% fault coverage and 90% yield, has a reject rate of 41.9%
    - Or 419,000 defective parts per million (PPM)

Next we look into that test generation processor. So, fault coverage as I was telling. So, it is defined as the number of detected faults divided by it a total number of faults. So, for a given test vector set, so this is the measure of fault coverage. So, 100 percent fault coverage maybe impossible due to some undetectable faults. So, we will see what is an undetectable fault essentially what it means is that due to some redundancy in the circuit, so some faults can never be excited by applying some patterns.

So, as a result so we cannot test those fault. So, that way there is a, it is we not get 100 percent fault converge, but even the circuit does not have redundancy. So, in those cases also it is very difficult to get 100 percent fault coverage, because it will require a very large test set. So, test vector set will be very large. So, that is why 100 percent is not achieved in most of the cases. So, fault detection efficiency. So, this is de define to be the number of detected faults to the total number of faults, minus number of undetectable fault. So, this is basically this denominated gives us the total number of faults that can be detected and this numerator is the number of fault detected by a particular test set. So, this ratio gives us the fault detection efficiency of a test set.

Then there is something called a reject rate it is 1 minus yield to the power 1 minus fault coverage. So, this major is known as the reject rate. So, for an example if a PCB has got say 40 chips, each with 90 percent fault coverage and 90 percent yield. So, if you 90 percent fault coverage. So, it is 0.9.

(Refer Slide Time: 18:10)



So, the reject rate will be given by for a particular chip will be 1 minus yield. So, yield is taken as 0.9; 1 minus 0.9 to the power 1 minus fault coverage that is 0.1. So, this whole thing I have got 40 such chips. So, if you calculate. So, it will turn out to be. So, per million so many chip of 419000 chips will become faulty. So, that is a huge number. So, you see we even with say 90 percent fault coverage we have got a very large number of defective parts, that are produced and that will get rejected after they have been shift. So, it is necessary that our fault coverage is even higher than 90 percent. So, that gives the challenge that we have.

(Refer Slide Time: 19:14)

Test Generation

- *Goal*: find efficient set of test vectors with maximum fault coverage
- Fault simulation used to determine fault coverage
  - Requires fault models to emulate behavior of defects
- A good fault model:
  - Is computationally efficient for simulation
  - Accurately reflects behavior of defects
- No single fault model works for all possible defects

So, values of all coverage say less than 95, 96 percent, so that is not going to be acceptable. So, what is this test generation process like the goal of the test generation process is to find efficient set of test vectors with maximum fault coverage that is the overall goal. So, there are several technique by which we generate test vector. So, we will see those in due course of time. Now how do we say that with the how many fault this particular test set can detect. So, for that purpose the sudden techniques are being developed which are known as fault simulation. So, fault simulation it will try to see like for a given set of patterns, what are the faults that this set of patterns can detect. So, that is that is the fault coverage what is the fault coverage.

So, naturally it will require the fault models to emulate the behavior of defects, as I was telling the physical defects are emulated by means of fault. So, this fault models will be

used. So, we will see that what type of different fault models can be there, but a good fault model it should be computationally efficient for simulation. So, it should not be very difficult to compute the values of different points in the circuit duty application of a test pattern, under that particular fault model. So, that is the one requirement, and second requirement it is that it should accurately reflect behavior of defect. So, the difference between the actual defect behavior and the actual fault behavior they should not be much different they should be as close as possible.

The difficulties that no single fault model works for all possible defect; so this is another catch, so as a result will see that different fault modules are to be used, and when you are doing the test generation. So, you have to assume this generation process on some fault model. So, therefore, they so one particular test generation process will not be able to cater to all possible fault models because that is not possible. So, they will be. So, that that we will have different fault test set generation process.

(Refer Slide Time: 21:21)

To give you an some idea about this fault models, a given fault models may have k different types of fault. So, k equal to 2 for most faults like say, you can say that if there is a signal line running in the chip in the system, then the signal line it were possible fault is that this line is permanently 1 or permanently 0. So, as if there is the V c c line running and there is a short between this line and this. So, as a result this line has to be treated as permanently one. So, this physical defect is model test permanently one, so that is known

as stack at 1 fault. Similarly if this line is sorted with same with the ground line, so we have got this another fault which is permanently 0.

So, when we are talking about say logic gates the inputs and outputs, then this model is a very ideal 1 because a logic gate has got some inputs and some output, and the fault that can occur is the line one of those lines becoming permanently one and permanent or permanently 0. So, that way every fault in this particular case they have got 2 types of faults. Now if a given circuit has got n possible faults sites then so depending upon the number of lines that we have in the system, there may be n number of lines and each of these lines may have the fault. So, as a result there are n possible faulty site. Now there can be multiple fault model like circuit can have multiple faults like in case those apart from single fault maybe a. So, if out of this n sites, maybe k of them are faulty, or say m of them are faulty. So, each of these faulty site it can say. So, each of these n faulty site it can have one of the k fault or the site maybe fault free; as a result if the line can be at a possible in any of this possible k plus 1 states.

So, if you consider the simultaneous if you consider multiple fault occurring simultaneously, then the total number of multiple faults is given by this expression k plus 1 to the power n, minus 1; this where this 1 is the good circuit. So, the fault free circuit. So, that is excelled. So, this is the number of multiple fault that can occur. Naturally this is impractical. So, if we have got for accepting small circuits where this n the number of lines in the circuit is very small. So, this becomes impractical. So, normally what is done the single fault model is used that is circuit has got only 1 fault, and it is said that though there n potential faulty sites, it is assumed that only one of those sites may become faulty at any point of time.

So, that way, so each of these n locations they may have a fault or may not have a fault as a result this only one of them may become faulty. So, I have got k into n number of single faults that are possible in the system, because this out of this n only one point will be faulty and that one point may pick up any of the k faults. So, total number of possibilities is k into n.

So, this type of fault model the single fault model though it is very simplistic, but this helps in generating developing algorithms for doing this testing efficiently, and it has been observed that good single fault coverage it generally implies would multiple fault

coverage. So, this is some experimental result that shows that if you are if you do not use a multiple fault model where a circuit can have multiple fault. So, if you do not use that model even in that case. So, we can have this if you use a single fault model and we have got the test pattern set are generated using the single fault model. So, for multiple fault also that particular test set can be applied and if we apply that test set. So, it gives reasonably good level of detection of multiple faults as well. So, that is why in most of our discussion, so will never be talking about this multiple fault model or multiple faults occurring in a circuit simultaneously. So, will most of the time will be talking about single fault model that is we will assume that the entire circuit has got a single fault.

(Refer Slide Time: 26:03)

Now, there are some faults which are equivalent; that is one or more single fault that have identical behavior for all possible input patterns. So, some cases it is not possible to detect such fault, because they are be becoming equivalent like see if I have got an and gate if I have got an and gate, now you see that suppose this line I want to check whether this is stuck at 0 or not. So, what will happen? I will apply to check whether this line is stuck at 1, so I will apply a pattern 1 1 and I will see the output. If at the output I get a 1, I know that this line is not stuck at 0 so, but if I get a 0, so assuming that these 2 lines are correct because under that single fault model only 1 line can be faulty. So, knowing that these 2 lines are correct, so I will conclude that this line has got a stuck at 0 fault by applying the pattern 1 1 I get it.

Now, considered the situation that this line is ok, but this line is I want to check whether this line is stuck at 0 or not. In this case also I have to apply the pattern 1 1 and see what is the response. So, if I get a 1, I will say that they know the line is not stuck at 0, but if I get a 0 then I will say that this line is stuck at 0. Now for both the faults like this line stuck at 0 and this line stuck at 0 the response of the AND gate is same for all the test cases. So, as a result I cannot distinguish between these 2 faults; a same is true for the other input as well. So, one or more single faults that have identical behavior for all possible input patterns. So, they become equivalent fault. So, we cannot have a test generation process that can distinguish between occurrences of these 2 faults.

So, from this entire set I need to consider only one fault knowing that all others are equivalent. So, they need not be tried out. So, that any test pattern that can detect this fault will also detect the other faults in that equivalent set; so way I do not need to have separate test generation process for each fault in this equivalent class. So, that is one thing. So, this is the definition of an equivalent fault. So, now, knowing that there are equivalent faults, so we can do we can do fault collapsing, fault collapsing means we want to reduce the number of faults that we need to consider for the text generation process, we remove this equivalent fault except for one to be simulated. So, or we keep only a single fault from the entire equivalency class and remaining all we collapse into that particular faults. So, this will reduce the total number of fault as a result this will reduce the fault simulation time, it will also reduce the test generation time.

So, for this is the first state that is done under the particular fault model, so we will find out the equivalent faults for a circuit; for each of the equivalent class we will take only one fault, and that they will do for collapsing and then the set of faults that we get finally, they are all nonequivalent faults and for those nonequivalent faults so we have to run our test generation process, and our fault simulation process. So, we will continue this in the next class.