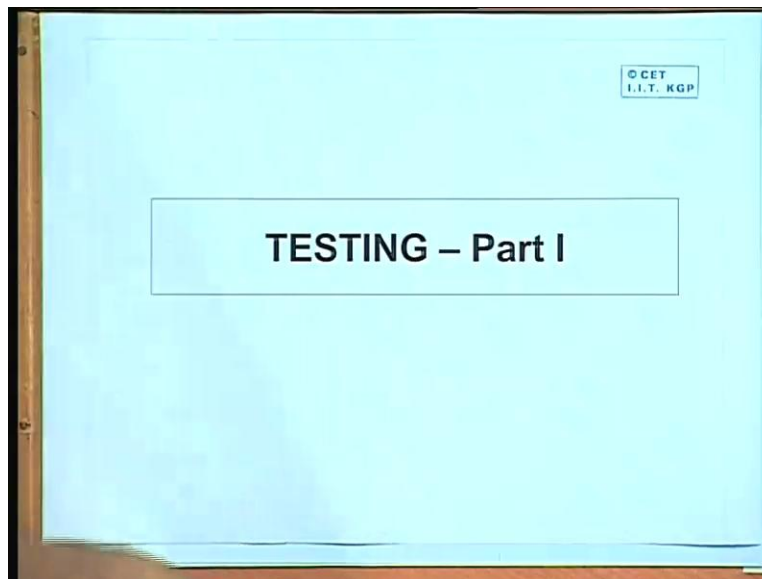


Electronic Design Automation
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No #30
Testing - Part I

In this course we have so far talked about various aspects of electronic design automation namely design, then synthesis, then the electronic physical design automation which is sometimes also called the backend design step in the VLSI design.

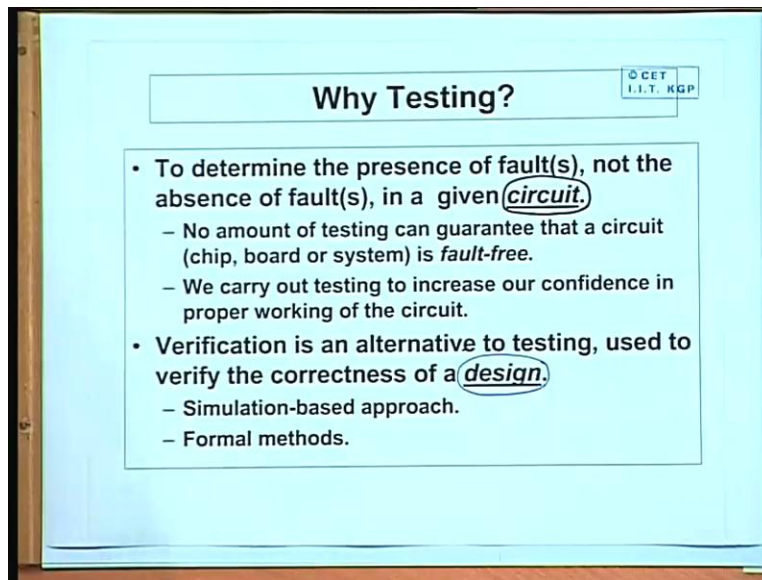
(Refer Slide Time: 01:11)



But now we would be talking about another very important aspect of design which is testing. So once we design a circuit well it can be at the level of a design. That means we have some kind of a design you can say design at a conceptual stage, it can be a behavior specification. It can be a resistor transfer level net list, it can be a gate level net list or it can even be at a much lower level like a transistor level net list or finally we can have the finished product in the form of a chip ASIC. So once we have a design or a finished product with us, the natural question is to verify whether it is functioning according to the intended specification. So the goal of testing is to do exactly this. So in the next few lectures we would be talking about the main emphasis that the

designers put on the various testability aspects of a design and what are the main you can say the rules and techniques which people use. So first let us try to talk about the basic need for testing why we need testing in a design environment.

(Refer Slide Time: 02:39)

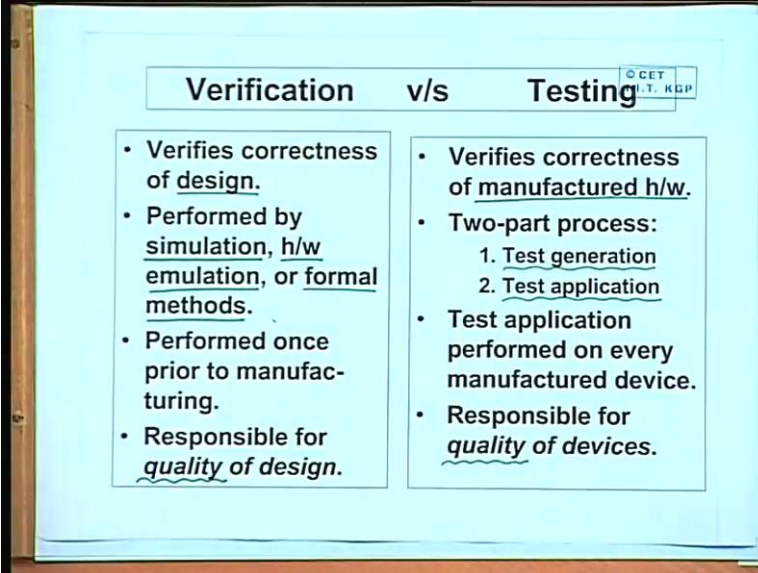


Well some people feel or think that the objective of testing is to ensure or guarantee the absence of faults. But I would like to emphasize that this is not true because no amount of testing can guarantee that the circuit is absolutely free of faults. So rather we can define testing as a process using which we are trying to determine the presence of faults ok. So as we go on doing testing and as we go on detecting faults if any we continue to increase our confidence in the proper working of the circuit. This is true because of the wide variety of the kinds of faults that are possible in a typical circuit. So this testing is one way to achieve this. So using testing we try to find out whether a circuit is behaving as per as our specification.

So here we are talking about a circuit the circuit maybe at the level of gates it can be at level of transistors or at any other level. But there is another alternative way we can go about doing this this is called verification. Verification is considered to be an alternative to testing, but the difference is that verification cannot handle a circuit rather it takes a design into consideration

and tries to find out whether the design is working correctly or not. There are a number of ways to carry out verification simulation based is the most popular but recently formal methods based on some mathematical tools are also increasing in importance. So first let us try to identify clearly the differences between verification and testing.

(Refer Slide Time: 04:49)



Verification	v/s	Testing
<ul style="list-style-type: none">• Verifies correctness of <u>design</u>.• Performed by <u>simulation, h/w emulation, or formal methods</u>.• Performed once prior to manufacturing.• Responsible for <u>quality of design</u>.		<ul style="list-style-type: none">• Verifies correctness of <u>manufactured h/w</u>.• Two-part process:<ol style="list-style-type: none">1. <u>Test generation</u>2. <u>Test application</u>• Test application performed on every manufactured device.• Responsible for <u>quality of devices</u>.

So this slide gives a comparative study of these two. First thing is that verifies correctness of design that is the task of verification and correctness of manufactured hardware this is the objective of testing. Now when we say manufactured hardware it can be a gate level circuit which you have fabricated on a breadboard it can be a field programmable gate array FPGA or it can be an application specific integrated circuit ok. So when we are testing we have a hardware which you can feel to which you can apply an input and we can observe the output we have such hardware and we want to verify the correct operation of that. But whereas verification works on a design which is on paper ok. So the hardware has not yet been manufactured.

Verification as I have told you that this is normally carried out either through simulation or through formal methods. Of course there are methods called hardware emulation which is primarily used to speed up the process of simulation ok. Simulation is just you can say some kind

of an accelerator to the simulation process. Testing in comparison it consist of two different steps first is called test generation next is test application. Now in test generation we try to find out that in order to verify the correctness of a circuit what are the inputs we need to apply ok and in the second step test application. We are actually applying those test vectors to find out whether the circuit output or responses are correct. Verification since it works on a design this is performed once prior to manufacturing.

So after we are verified design is correct we can go ahead with manufacturing. But the objective of testing is to find out defects during the manufacturing process. So it has to be applied on each and every manufactured device this is the difference ok. Verification is responsible for the quality of the design, because it will give you a confidence that the design is correct. In contrast the testing will be responsible for the quality of the devices because each and every finished product you are testing against manufacturing defects ok. So in this lecture we would mainly be concentrating on the problem of testing and not verification. Of course we would be talking about simulation but not about something like formal verification or hardware emulation ok. So first let us see that when you talk about testing.

(Refer Slide Time: 08:13)

Levels of Testing CCEET
I.I.T. KGP

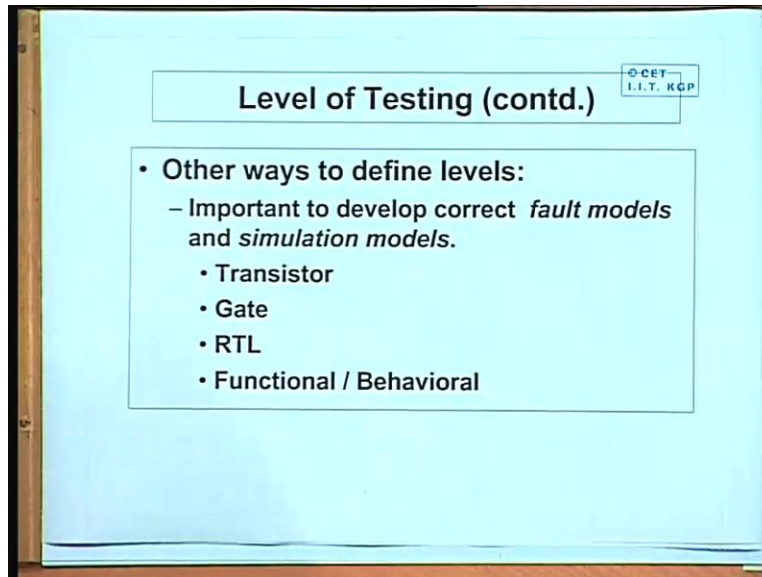
- Testing can be carried out at the level of
 - Chip ✓
 - Board ✓
 - System ✓

↓
- Cost :: Rule of 10
 - It costs 10 times more to test a device as we move to the next higher level in the product manufacturing process.

What are the different levels at which we can carry out testing. Now it depends that at which level we have the system or the hardware manufactured and available to us. It can be an individual chip it can be a printed circuit board which consists of a number of chips or it can be a system where a number of such boards are plugged in ok. So we can of course potentially carryout testing at any of these levels but we have to remember the implications of cost. There is a rule of ten this is a rule of thumb which is very much applicable to a hierarchy like this. This says that if you can catch a fault at the chip level then it will take you or it will incur about 10 times most cost to detect the same fault at the board level and again with respect to board level system level fault diagnosis would require 10 times more cost again.

So it says that it costs ten times more to test a device as we move to the next higher level in the product manufacturing process. So when you say next higher level we are actually moving in this direction from chip to board to system. So what this really means is that when the chips are manufactured we have to carryout testing immediately on the chips. So as the chips are used to manufacture the boards you must have a confidence that the chips are working correctly ok well right. So this is one way to identify or categorize the level of testing chip printer circuit board or system. There are other ways to categorize the levels also. It depends on the level of abstraction at which you are really viewing the circuit. Well a circuit can be viewed at a number of different levels of abstraction. So there can be another kind of categorization that depends on this level of abstraction.

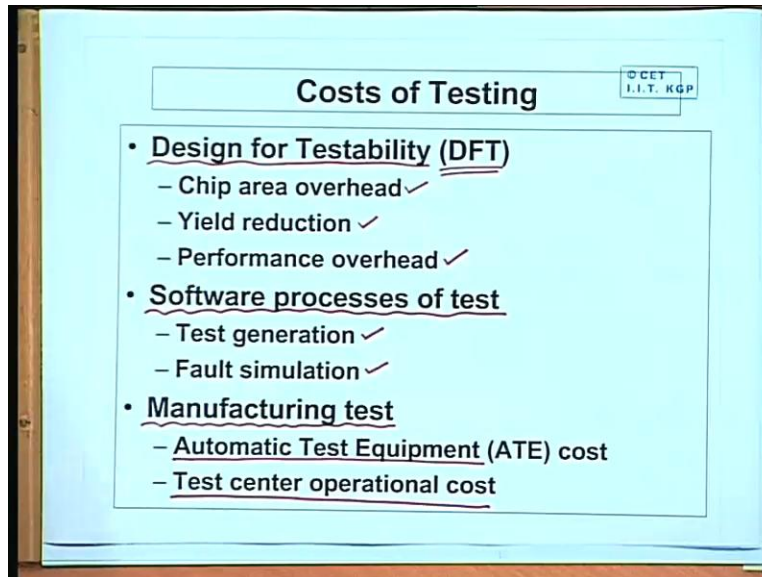
(Refer Slide Time: 10:39)



Well the levels of abstractions we can defined or as follows we can assume that the circuit is available as a net list of transistors typically MOS transistors or as a net list of gates or as a net list of register transfer level blocks like multiplexers, arithmetic logic units, registers, etcetera. And at the highest level we can have functional or behavioral model where we have very high level functional description of the blocks and their interconnection ok. So now it really depends that at which level out here that we are concentrating on.

Because if we are concentrating say for example on the gate level model of a circuit, then we will have to develop something called fault model and simulation model we will be talking about this later which will be pertinent to the gate level description of the circuit. If we move on to the other levels the fault model and the simulation model will also vary. So it really depends on the test engineer at which level he or she wants to carry out the testing and the fault modeling and the simulation modeling has to be done according to that and well we want to test the devices. Naturally in order to test the devices we have to incur some additional cost. So let us see what are the different components of the costs that we incur during the process of testing ok.

(Refer Slide Time: 12:38)



Here we introduce a term called design for testability this is a very important term and this is required for the following reason. Say if you look at the modern day devices, the VLSI chips which are coming out fabricated they contain millions of transistors or gates in a single package. Of course the number of pins that we have for that package is very much limited. So if someone gives you such a chip with a million transistors inside, so it will become very difficult for you to test that chip and say with a very high degree of confidence that this chip is working correctly.

So at the time of design itself you have to put in some additional effort so that the design will be such that during the testing phase it will be easier for you to carry out the testing. This in short is called design for testability. These things we will be discussing later in short this is called DFT. Now in DFT actually we introduce some additional hardware inside the chip with the objective that externally from the outside world we can test the chip much more easily. Since we are adding some external hardware of course we would be incurring some chip area overhead the number of transistors or gates in the chip will increase.

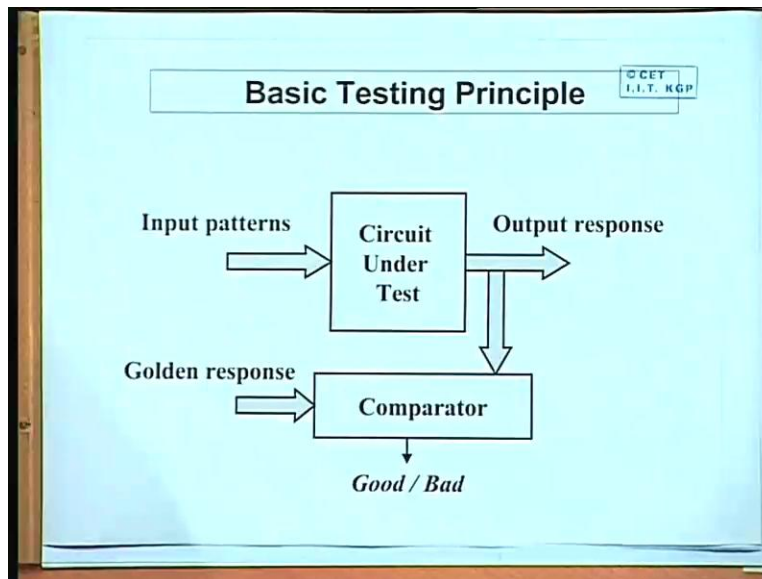
So the area will also increase and yield will reduce. Well yield is a factor which people use in connection with fabrication. So when a chip gets fabricated normally we do not fabricate one

chip at a time rather on a wafer which is typically of a circular shape many chips of the same type of a fabricated at row. Now yield is a measure of what is the success rate that means what fractions of the fabricated chips are found to be good. Now it has been found that bigger the chip more the number of components yield will tend to decrease and if we can somehow make the yield higher our cost per chip will also go down ok.

So yield reduction is a byproduct of chip area increase and since we are introducing some extra hardware some of this hardware will also be in the critical path of the circuit. So the performance overhead will also be there ok. These are the extra overhead for DFT well but during actual testing you will have to apply the test vectors and you will have to evaluate the test vectors. These are done through processes called test generation and fault simulation and these are software algorithms which are run offline on a computer. So you will also have to evaluate the cost of test generation and fault simulation which is done offline and finally when the chips are manufactured.

So you will have to test them by applying the tests which have generated earlier and by evaluating the response. This is typically done by using very expensive equipment called automatic test equipment. Since this is a very expensive equipment using this equipment will also incur additional cost and of course the test center which operates this ATE they are overrate cost is also have to be taken into account ok. So these are the different points you need to keep in mind when you are trying to evaluate the total cost you required for testing ok.

(Refer Slide Time: 17:08)

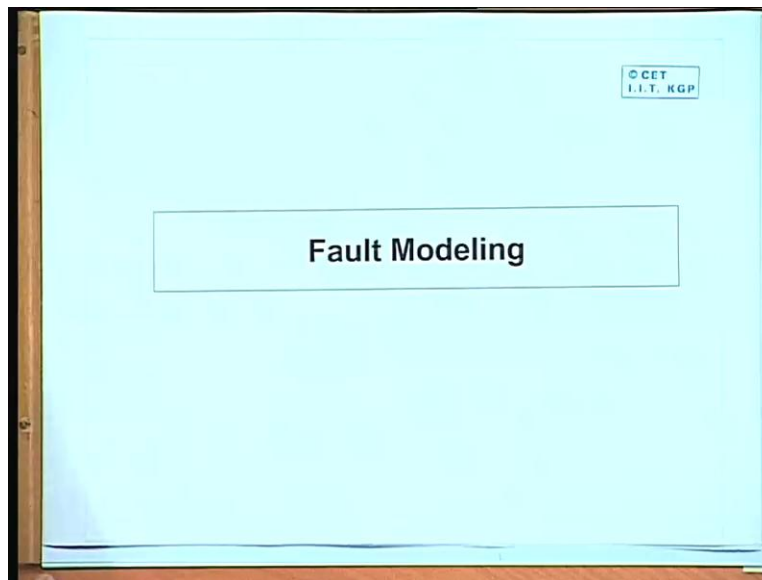


The basic testing principle in simplified form may be diagrammatically represented like this. Well you have a circuit under test this can be a combinational circuit also a sequential circuit it can be anything this is the block we want to test. We want to test means we have to excite it with a set of input patterns we have to apply some input patterns and we will have to observe the output response. This output response can be observed using some kind of a comparator this comparator can be done externally or it can be done inside the chip also where the output responses are compared against the desired response which is sometimes called the golden response. And if they match we announce that the circuit under test is good otherwise we announce it is bad. Now in this context from this diagram you try to understand one thing that the circuit you are trying to test, it can be a very large circuit.

Which means that the number of input patterns we are applying this can also be very large. Which implies that the number of output responses you are getting that is also very large. So you will have to keep a very large set of golden responses to compare with the output response. So in order to reduce, this overhead we will see later that in many cases we somehow compress the output response into a small value which is called a signature and we only maintain the golden signature. That is a very small number you can store and we compare the signatures to say or to

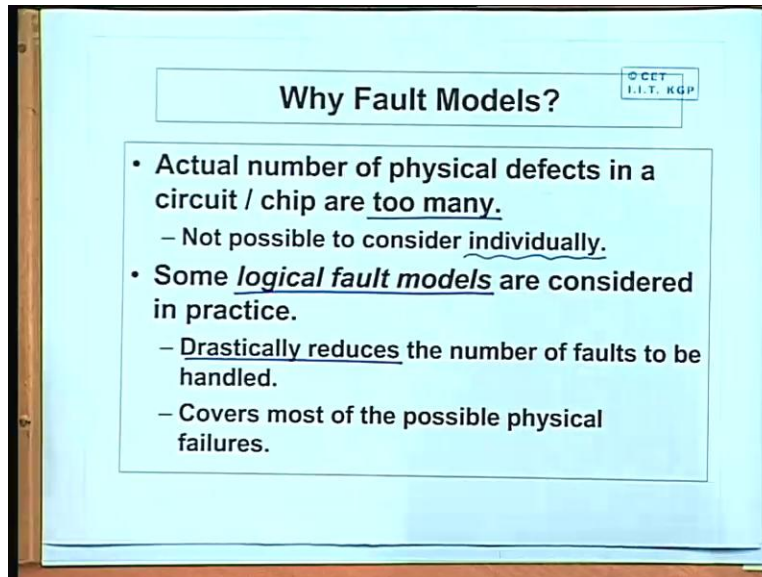
evaluate whether the circuit is good or bad ok fine. So before we move on to the different techniques that people use for test generation fault simulation and DFT. The first very important thing we need to talk about is something called fault modeling.

(Refer Slide Time: 19:18)



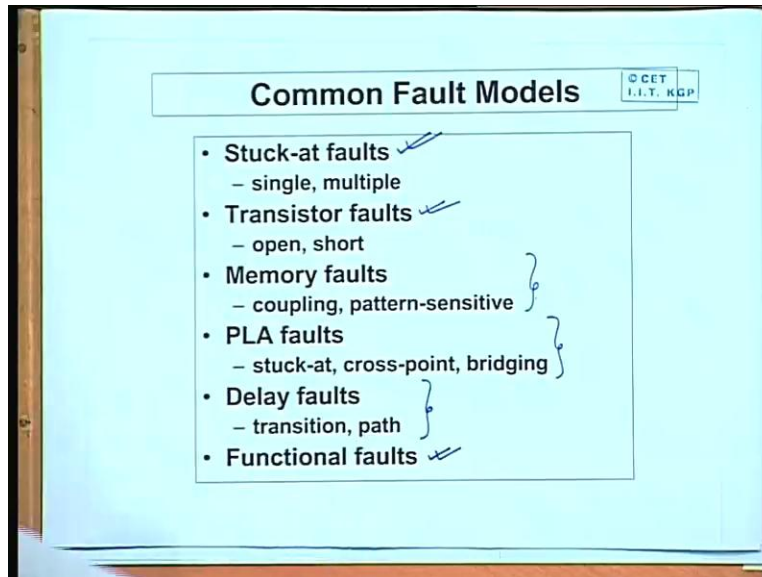
Now fault modeling what is that? See we are saying that we want to test a circuit ok but the question arises that we are trying to test fine but we are we are trying to test for what. Well if we say that we are trying to test for faults then the next question will be what kind of faults? If we talk about physical faults then the number of such faults can be infinitely many there can be physical parameter changes which can be continuous. There can be changes in voltages there can be changes in the wire thicknesses in environment temperature. There are so many different parameters which can change and the amount of change cannot be quantified. They are all continuous variables. So to make things simple what the test engineers do? They ignore the physical failures which are actually taking place but rather they look at the effects of the physical failure how they are affecting the behavior of the circuit. These are called the so called fault models or the logical fault models. So we would we would have to first fix up a logical fault model and using that we will have to frame our fault simulation, test generation, whatever scheme we want to use ok.

(Refer Slide Time: 20:45)



So as I had mentioned that the reason we use fault model is that the actual number of physical defects that are encountered in a chip or a circuit are too many because they are too many it is not possible to consider them each of them individually and address them individually so that you can say that all the faults have been tested for. So what is actually done is that some logical fault models are assumed. Logical faults models are you will see that these are some simplified fault models which are defined based on the circuit which you have. This faults models drastically reduces the number of faults so that the problem become much more manageable. But the good thing is that it has been found that even if we use a simplified logical fault model even then you will be covering most of the possible physical failures. This means even if we are using the logical fault models they are not bad they are good enough. You can use them in a practical testing scenario with a good degree of confidence. Now let us see what are the common fault models that people have used over the years.

(Refer Slide Time: 22:21)



Well the simplest and possibly the most widely used still today is something called stuck at fault model it can be single or multiple we would be talking about these things. Stuck at fault models typically concentrate on a gate level or a resistor level net list of the circuit. If you are talking about the transistor level circuit you can have some transistor level faults open and short there are some peculiarities in these fault lev fault models which cannot be captured in stuck at fault model we will see that. If we have some special circuit structures like memory or programmable logical arrays you have some fault models which are very specific to these but we will not be discussing these for now. Similarly when we are talking about delays in the circuit hazards there are something called delay faults which also come into the picture.

So there are fault models delay faults as well. Recently people have been talking about functional faults. But instead of looking at the circuit or the net list which you have rather you talk about the behavioral description of the circuit the input output behavior and what kind of test patterns you need to apply in order to verify the behavior ok. When you have a net list well you have that circuit diagram in your mind, you are now concentrating on the fault on each interconnection and all each of the component. But at the behavioral level everything is a black box ok. So the level

of testing is different fine. So let us first start with stuck at fault because this is the most important category of faults which is used in the industry.

(Refer Slide Time: 24:24)

Stuck-at Faults © CET I.I.T. KGP

- Some line(s) in the circuit are permanently stuck at logic 0 or logic 1.
- Two types:
 - Single stuck-at faults
 - Multiple stuck-at faults

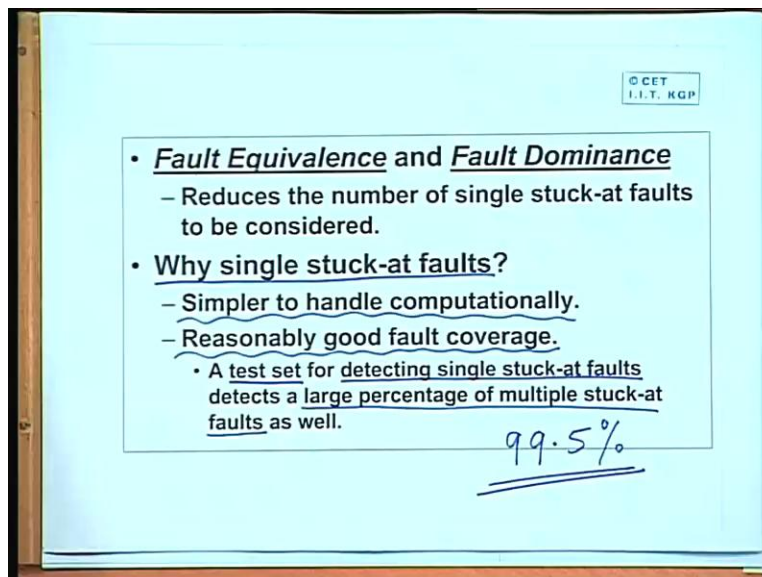
The slide contains two hand-drawn diagrams. The first diagram on the left shows a logic circuit with three AND gates. The top AND gate has inputs 0 and 1, and its output is labeled 's-a-1'. The second diagram on the right shows a TTL totem pole output stage with two transistors. The output node is labeled 's-a-0' and is connected to ground, representing a stuck-at zero fault.

Stuck at fault is very simple in concept. This says that some lines in the circuit are permanently stuck at logic zero or logic one say I have a circuit like this. Say there are three gates, I consider a gate level circuit. I can say that the output of this AND gate is stuck at one this means that irrespective of what input value I am applying to the inputs of this AND gate the output of this AND gate will always be at logic one. Now this is a logical fault model as I told you but you can also find an analogy from the physical fault module. For example if you think of a TTL gate, now we know that in a TTL in the output stage there are two transistors so a totem pole TTL get the output stage looks something like this. This is the output. Now due to some reason due to some fault in the circuit out here which is driving this transistor or some fault in the fabrication of this transistor itself if this transistor behaves is a short circuit then the output will always be short circuit to ground.

So this physical failure can be considered to be equivalent to an output stuck at zero fault ok. So this is just one example I have taken. Well almost all the physical faults which are possible in a

circuit can be modeled by an equivalent stuck at fault model of course there are some faults which cannot be modeled but most of them can. Now in the stuck at faults we can consider one stuck at fault at a time which is called the single stuck at fault model. We can consider more than one fault occurring at a time that is multiple stuck at fault model. Now again single stuck at fault model is more popular because of its simplicity. Well we will see how simple is it later. So the question is that if we use the stuck at fault model, then given a circuit we have so many lines these are 1, 2, 3, 4, 5, 6, 7 lines. So you can have 7 into 2, 14 possible single stuck at faults. Now we will have to detect all those faults.

(Refer Slide Time: 27:25)



© CET
I.I.T. KGP

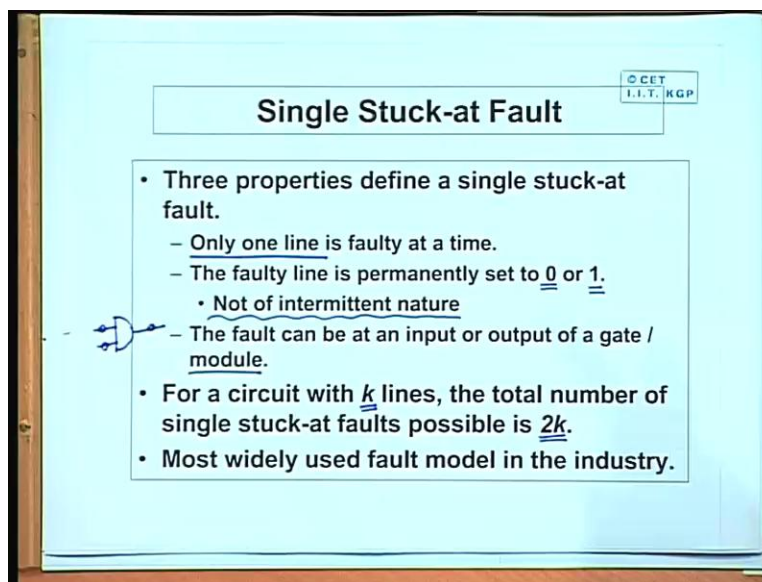
- **Fault Equivalence and Fault Dominance**
 - Reduces the number of single stuck-at faults to be considered.
- **Why single stuck-at faults?**
 - Simpler to handle computationally.
 - Reasonably good fault coverage.
 - A test set for detecting single stuck-at faults detects a large percentage of multiple stuck-at faults as well.

99.5%

So there are techniques called fault equivalence and fault dominance which we would be talking about very shortly which can reduce the number of single stuck at faults to be considered. Right now we are assuming that we have a circuit which is a gate level net list and we are concentrating on the single stuck at fault model ok. And we are trying to find out a way using which we can reduce the number of faults that we can consider. Now let us try to address this question once more why single stuck at fault. Well obviously as compared to multiple stuck at faults the number of single stuck at faults will be much less in number so it will be simpler to handle computationally.

Not only that it has been found that fault coverage which means the number of faults that are actually getting detected is reasonably good for single stuck at fault. This says that if you generate a set of input vectors which is called a test set, which can detect all single stuck at faults in a circuit it can be proved that this test set can also detect a large percentage of multiple stuck at faults. Well when I say large percentage of multiple stuck at faults it can be as high as 99.5 percent. So this is not a very bad figure. Because of this single stuck at fault has become so popular and most of the industry people they indeed use single stuck at fault model for testing purposes ok.

(Refer Slide Time: 29:18)



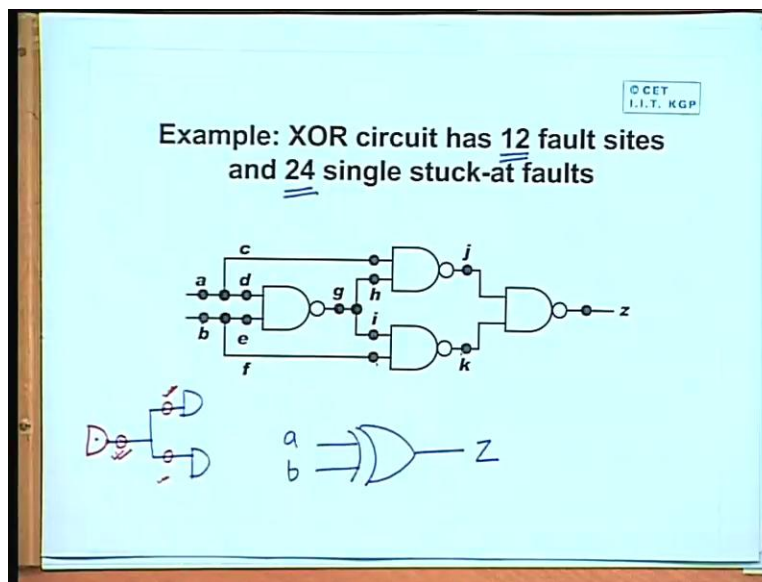
The slide is titled "Single Stuck-at Fault" and includes a logo for "CET L.I.T. KGP" in the top right corner. It contains a bulleted list of properties defining a single stuck-at fault. A small circuit diagram of an AND gate is shown to the left of the list, with a blue arrow pointing to its output line.

- Three properties define a single stuck-at fault.
 - Only one line is faulty at a time.
 - The faulty line is permanently set to 0 or 1.
 - Not of intermittent nature
 - The fault can be at an input or output of a gate / module.
- For a circuit with k lines, the total number of single stuck-at faults possible is 2k.
- Most widely used fault model in the industry.

So let us now concentrate on single stuck at fault. Single stuck at fault as I told you there are three properties. This I have already mentioned since it is single fault only one line is faulty at a time. Now when you say faulty, since it is stuck at the line is permanently set to 0 or to 1. So when I say permanently it means that the fault is not of intermittent nature intermittent means it appears sometimes it does not appear sometime. So we assume that whenever we apply a test vector the fault is present ok. So it can be detected anytime we applied it as went and the faults can occur at the input or the output of a gate.

Suppose we have a gate like this, we can have a stuck at fault on this input on this input or on the output. Now it can be a gate or it can be it can be a module at a higher level also ok, in general it can be a module. So we can easily see that if there are k number of inter connecting lines in a circuit the total number of single stuck at faults will be $2k$. Because each line can be stuck at 0, stuck at one and one fault occurring at a time there are $2k$ possible faults. Now this is most widely used as I had mentioned. So let us take an example.


(Refer Slide Time: 30:53)



So here we have a NAND gate implementation of a two input XOR gate. This is the NAND implementation of a two input XOR. Now as a matter of convention you see in this circuit here or also here there is some places where we have a fan out from a line you have two branches going out to two different gates ok. So if you have a scenario like this, then these three are considered to be three different lines. The reason we consider them to be different is as follows. There can be a fault inside this gate so that the output is always stuck at zero or stuck at one. So if a fault like that happens that fault will equally affect both the gates. So this can be considered to be equivalent to a fault here. But if there is a fault in the input of this gate for example there is a disconnection. Well in a TTL gate you know if a line is open circuit it is treated as logic high.

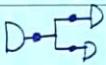

So in that case the fault will appear here but not here or here similarly here. So whenever there is a fan out because of the physical nature of the fault we have to consider these three faults separately. So in this circuit you can see that there are so many fault sites 1, 2. There is a fan out so 3, 4, 5, 6, 7 again a fan out, 8, 9, 10, 11 and 12 so there are 12 fault sites and since we can have stuck at zero, stuck at one, in all this fault sites we can have 24 single stuck at faults in this circuit ok. Well now we will see that that out of this 24, for example in this example circuit we can eliminate many of the faults because they are not required to be tested. We can reduce the number of faults to be tested from 24 to a smaller number. Now we come to the notion of a concept called fault equivalence.

(Refer Slide Time: 33:33)



Fault Equivalence

- Number of fault sites in a gate-level circuit
= #PI + #gates + # (fanout branches)
- **Fault equivalence:**
 - Two faults f1 and f2 are *equivalent* if all tests that detect f1 also detect f2.
 - Example:
 - An input line s-a-0 and output line s-a-0 in an AND gate.
- If faults f1 and f2 are equivalent then the corresponding faulty functions are identical.

Fault equivalence is a mechanism using which we can find out true faults whose effects on the circuit are identical. Now if there are two faults whose effects are identical we can delete one fault from on list we can concentrate on only one ok, so this is the idea. Now in a gate level circuit as I had said the number of fault sites are number of primary inputs plus number of gates plus the number of fan out branches as I had just ex explained because a gate will contribute one fault site if there are two fan outs. There will be two more fault sites ok so number of fan out

branches will get added ok. Now fault equivalence the definition says that two faults f_1 and f_2 they will be set to be equivalent if all test patterns that detect f_1 also detect f_2 .

Here now let us try to clarify one point what do we mean by test that detect a fault. Suppose I have a circuit and I am applying a test pattern to the circuit suppose in the absence of a fault the output value is zero. Now a particular fault f_1 occurs in the circuit now if the fault f_1 is present the output is one then I say that this particular test vector will be detecting the fault f_1 which means that the output of the circuit the good value and the faulty value must be different only then we say that the particular fault will get detected by the test vector we are applying ok. Now here we take an example of an AND gate lets see lets take a three input AND gate a b c output is f. Well this statement says that the input line stuck at zero and the output line stuck at zero is equivalent. Why?

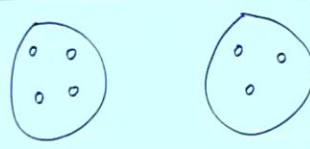
Because if you if any of this input lines are stuck at zero, the outputs will obviously one. Similarly if the output is stuck at zero, then also it will be sorry the output will be zero. Similarly if the output is stuck at zero then also it will be zero. So if any of these four faults occur the output value will be the same. So instead of considering these four faults we take any one of these four say for example. We only consider f stuck at zero fault and we delete the stuck at zero, b stuck at zero and c stuck at zero from our consideration ok. So in this example this a zero, b zero, c zero and f zero this four faults are equivalent ok. So if we find out such sets of equivalent faults the next step we carry out is something called fault collapsing.

(Refer Slide Time: 37:10)

Contd.

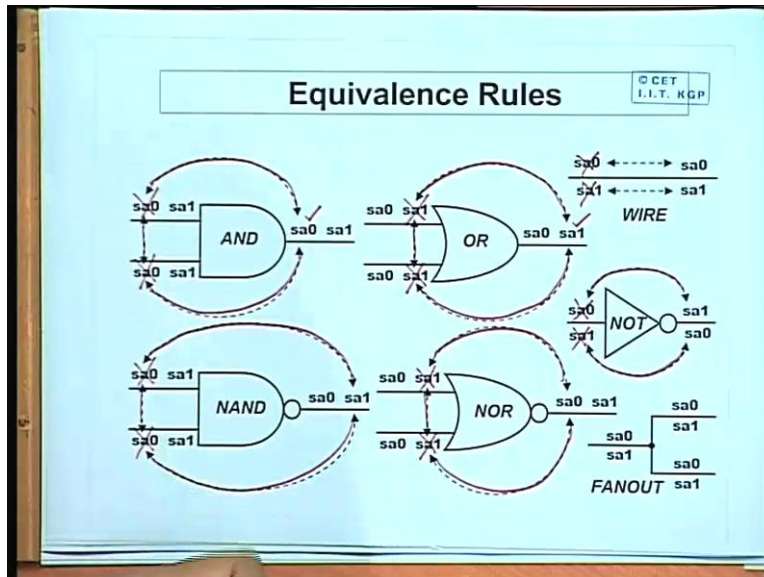
© CET
I.I.T. KGP

- **Fault collapsing:**
 - All single faults of a logic circuit can be divided into disjoint equivalence subsets.
 - All faults in a subset are mutually equivalent.
 - A collapsed fault set contains one fault from each equivalence subset.



Now fault collapsing says that we have the equivalent subsets say we have one subset where for example these four faults are identical. We can have another subset where we can find out there are three faults which are identical. Now what we say that you take each of these equivalent subsets retain one delete the others. It means you collapse all the faults that belong to one subset into anyone representative member of that subset. So a collapsed fault set will contain one fault from each equivalent subset. So fault collapsing is a preprocessing step and using fault collapsing you can drastically reduce the number of faults that need to be considered. Because reducing this number of fault will help you in carrying out fault simulation and also test generation and also doing testing. So the number of test patterns also gets reduced ok. So let us try to frame some rules using which you can carryout this kind of fault collapsing ok.

(Refer Slide Time: 38:35)



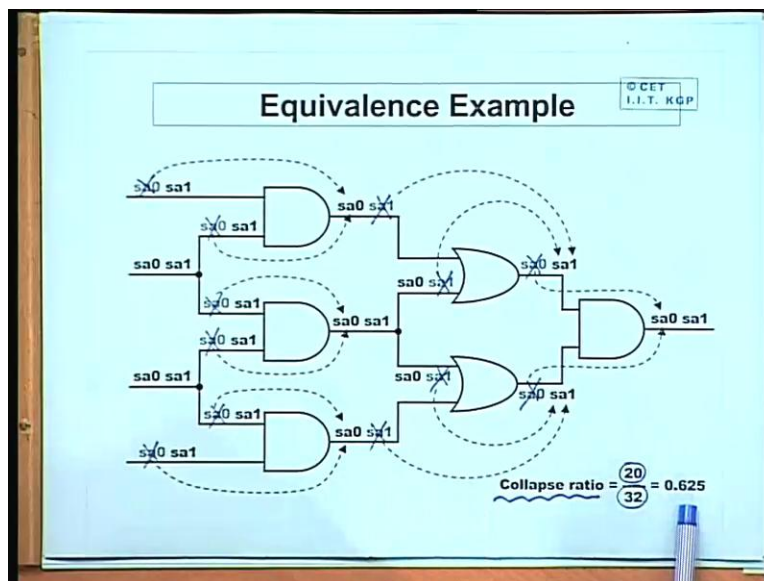
So I have a slide out here which shows all the popular gates and their impacts with respect to fault equivalence. Well without loss of generality we are considering two input gates. Well you can easily extend them to gates of any number of inputs. Ok first let us consider an AND gate. Well AND gate input can be stuck at zero, stuck at one, this line also stuck at zero, stuck at one, output also stuck at zero, stuck at one. So there are six possible faults in this AND gate. Now as this dotted line shows these three faults are equivalent. So in a AND gate you can delete any two of them and retain only one. Say for example we can delete this stuck at zero, we can delete this stuck at zero, and we can retain only the output stuck at zero. So in an AND gate from 6 we have got down to 4 faults 1, 2, 3 and 4.

Similarly for our OR gate the input stuck at one and the output stuck at one are equivalent. Because if any of the input lines are at one the output will also be at one ok. So, just in a similar way as at this AND gate here also these three faults will belong to the equivalent subset. And you can delete any two of them and you can keep one ok. Similarly for a NAND gate the stuck at zero in the input and stuck at one in the output are equivalent. Because if the inputs are stuck at zero, the output will be one ok. So here again you can define this equivalent subset and you can delete these two. Similarly for a NOR stuck at one in the input and stuck at zero in the output are

equivalent. So you can delete the stuck at ones in the input. So what the rule says rule says for AND and NAND you can ignore the stuck at zero faults in the inputs.

For OR and NOR you can ignore the stuck at one faults in the inputs. But in NOT, well NOT is also simple stuck at zero in the input is equivalent to stuck at one in the output. This is one equivalent set and stuck at one here and stuck at zero. This is another equivalent set. So from this equivalent set you can delete one from this set again you can delete one. So from NOT you can delete all faults from the inputs. If there is a fan out there are no equivalence here so for a fan out you cannot delete anything. But if it is a simple interconnecting wire trivially these faults will be equivalent. So you take the faults at any one end of the wire they are the same. So using the simple equivalence rules you can reduce the number of faults that need to be considered in a circuit. So let us take an example to illustrate this.

(Refer Slide Time: 42:23)



Suppose we have a circuit like this the circuit consists of only AND and OR gates. Now just from the rules that we had mentioned we can start doing the thing say. This is an AND gate we can delete the stuck at zero faults in the inputs, this is another AND gate stuck at zero, stuck at zero you delete this is another AND stuck at zero stuck at zero. These are OR so the stuck at one

faults you delete stuck at one, stuck at one, stuck at one, stuck at one. This is AND stuck at zero, stuck at zero, these are fan out branches you cannot delete anything. So in this circuit how many lines where there 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16. So there where 32 faults total. Out of them we have deleted 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

So we are left with 20 faults this ratio is sometimes called the collapse ratio that using fault equivalence we can reduce the number of faults to about 62 percent of the original ok. So this fault equivalence is a very useful technique which is used to reduce or decrease the number of faults that need to be considered in a circuit. There is another technique which can also used to I mean say you can say remove some of the faults fault equivalence is one technique we had just mentioned there is another method called fault dominance. Fault dominance here the concept is that one fault dominates the other. Now if there is domination relationship then one of the fault can be removed how let us see.

(Refer Slide Time: 44:39)

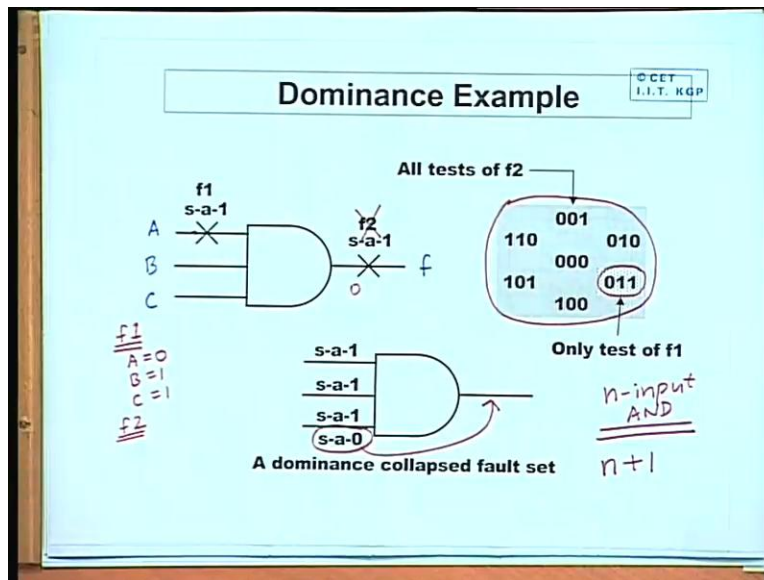
Fault Dominance

- If all tests for some fault f_1 detect another fault f_2 , then f_2 is said to *dominate* f_1 .
- ***Dominance fault collapsing:***
 - If fault f_2 dominates f_1 , then f_2 is removed from the fault list.
- In a tree circuit the primary input faults form a dominance collapsed fault set.
- If two faults dominate each other then they are *equivalent*.

The definition of dominance says that suppose we have two faults f_1 and f_2 . If all test vectors that detect fault f_1 also detect fault f_2 then we can say f_2 dominates f_1 so what does this mean? If we generate any test vector to detect fault f_1 that test vector will also detect f_2 . So why not we

remove f2 from consideration we can delete f2 we take only f1 and we try to find out a test for f1 ok. So dominance fault collapsing says if a fault f2 dominates f1 then f2 can be removed from the fault list ok. Well this point I will come to later. But first let us try to give an example of this.

(Refer Slide Time: 45:47)



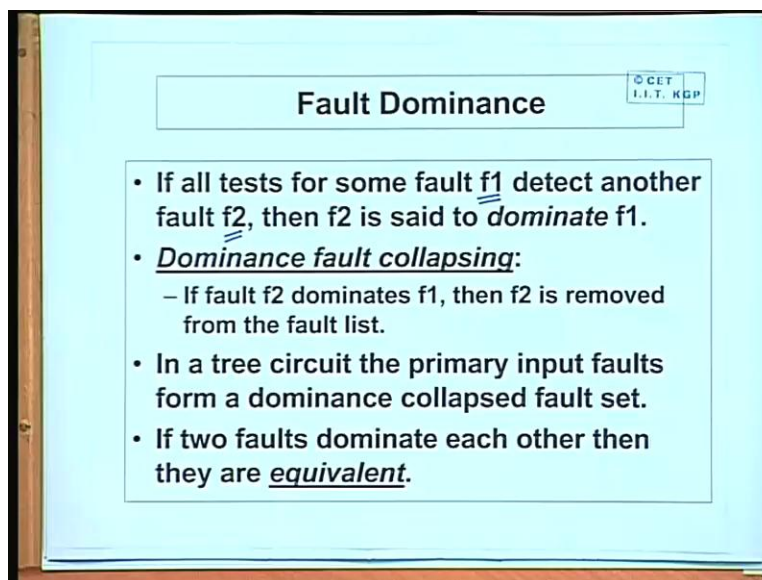
Let us take a simple example of a three input AND gate. Three input AND gate with two faults f1 and f2 say f1 is this these are a b c the inputs this is the f. So this f1 is a stuck at one and f2 is output f, this stuck at one. Now let us try to find out that what are the test vectors that are used to detect fault f 1. Let us try to understand this first for fault f1 what input we need to apply. First thing you observe that since the fault is an f1 stuck at one in order to detect it we have to apply a reverse logic value to a so a must be zero ok. Regarding b and c since this is an AND gate if either b or c is zero the output will always be zero. So we cannot detect whether fault a, is present or absent. So b and c must be one in fact for f1 stuck at one this is the only test vector which you can apply but for f2 what is the test vector you need f2 is the output stuck at one.

So somehow if you can apply a test vector which can put a zero on the output. Then you would be able to detect this fault because under fault free condition to be zero if the fault is there it will be one. For a three input AND gate all the 6, all the in fact all the seven combinations other than

the all one combination will give an output zero. So any of these can be use at the test set. This is shown pictorially here these are the different test vectors. Out of this this is the test vector for f1 as I had mentioned and all the seven test vectors excluding all ones 111 these will act as the possible test vectors for f 2. Now in this example who dominates whom? See I can say a test for f1 will also act as a test for f 2. But not the reverse which means f2 dominates f1 and I can remove f2 from the fault list ok.

So for an AND gate we can well total number of faults that we need to consider are only four stuck at one in the input and stuck at zero either in one of the inputs or you can also have stuck at zero on the output. So total 4 you need to consider. Because if you do not consider dominance it would be 5, 4 for three input gate three here and one here four. But for dominance 5, 2 here stuck at zero, stuck at one. But if we also use dominance the output stuck at one, also gets removed. So you have reduce it to 4. In general for an n input AND gate if we have an n input AND gate the number of stuck at first that need to be considered are the number of primary inputs plus one. This aim is true for OR, AND, NAND and NOR gates ok. So let us come back to the slide.

(Refer Slide Time: 50:19)

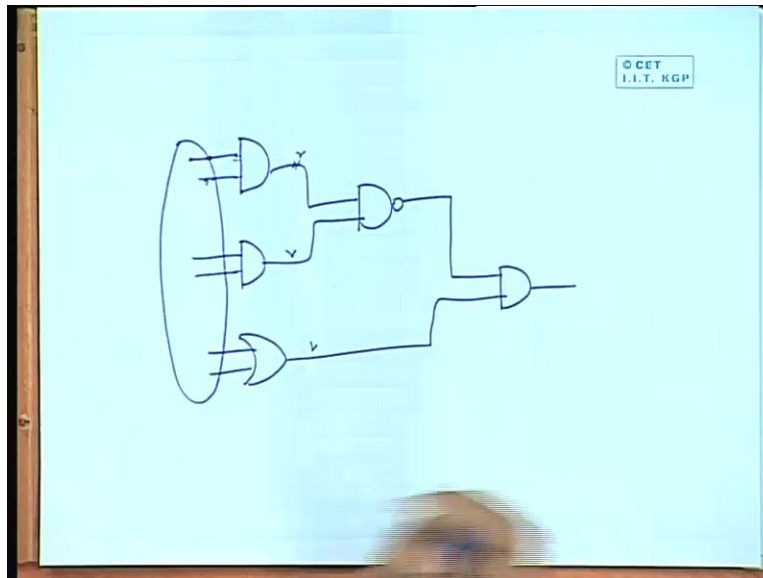


Fault Dominance O.C.E.T.
I.I.T. KGP

- If all tests for some fault f1 detect another fault f2, then f2 is said to dominate f1.
- Dominance fault collapsing:
 - If fault f2 dominates f1, then f2 is removed from the fault list.
- In a tree circuit the primary input faults form a dominance collapsed fault set.
- If two faults dominate each other then they are equivalent.

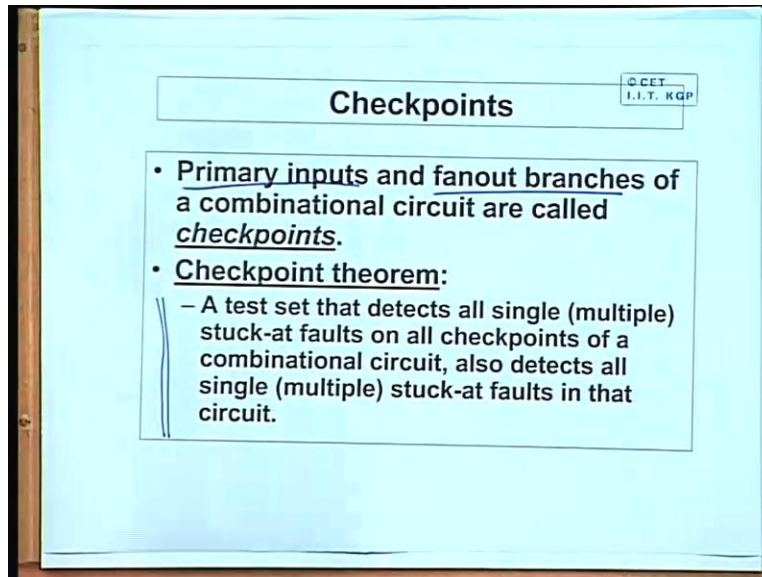
Here it says that in a tree circuit the primary input faults form a dominance collapsed fault set. This is an interesting result. Let us try to explain what this means. See a tree circuit is one where there is no fan out.

(Refer Slide Time: 50:37)



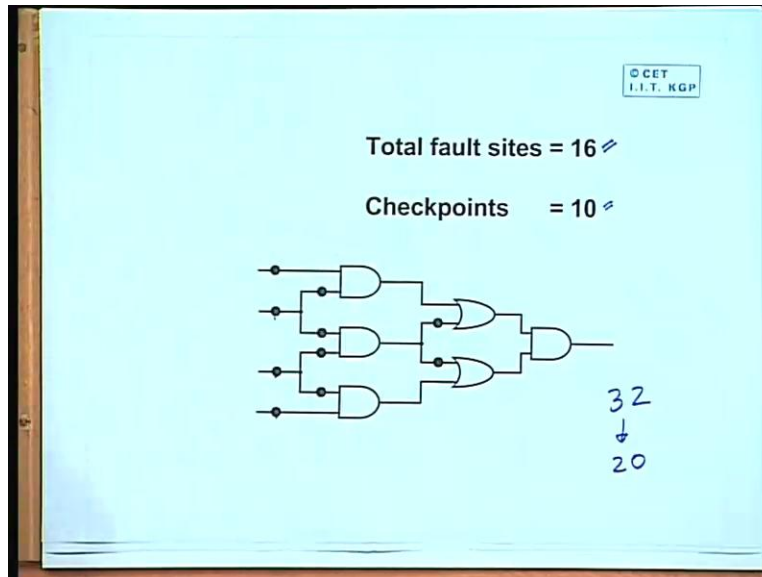
Suppose I have a circuit like this. This is an example of a tree circuit there are no fan outs. Well here what it says that if you consider only stuck at faults on the inputs primary inputs. This will constitute the primary input faults form a dominance collapsed fault set. Because you can easily see that such since there is no fan out. So any stuck at zero or stuck at one, fault on the primary input will be dominated by a fault on the primary input that can be removed. Similarly this can be removed this can be removed and this can be recursively applied and from each primary input there is a single path to the primary output. So this can be easily shown that if you have a circuit like this you need to consider only the primary input faults and this faults will constitute or this will cover the faults for the whole circuit. Now in fact there is an interesting theorem which helps us in concentrating on which faults to be considered for the test generation this is based on a concept called checkpoints.

(Refer Slide Time: 52:13)



Now checkpoint says that the primary inputs of a circuit and also the fan out branches these are called checkpoints. And the check point theorem says that a test set that detects all single. Well single for the time being is not multiple, now all single stuck at faults on all checkpoints also detects all single stuck at faults in that circuit. So it says that if we concentrate only on faults on the checkpoints that will cover the other faults. This is a very interesting theorem which helps us in reducing the number of faults in a very easy way without have to wit without having to analyze circuit in a very complex. So we can very easily do this. Now a very simple example to illustrate this.

(Refer Slide Time: 53:06)



Suppose we have a circuit like this these are the primary inputs, these are the fan out branches these are the fan out branches. So there are ten total check points but total fault sites where 16. So straight away using checkpoint theorem you have reduced the fault site number of fault sites from 16 to 10. So in terms of the faults you have reduced it from 32 to 20. But you should remember that using fault dominance and equivalence you can possibly reduce the number of faults even less than 20. But this checkpoint theorem is a useful starting point. Well so you can very quickly reduce the number of places you need to check before we apply the equivalence and dominance to reduce the number of faults. So in this lecture we have talked about the basic problem about testing and some of the fault models. So we would be continuing with our discussion in the next lecture where we would be looking at some other fault models multiple stuck at faults. Then the transistor level fault models and then we will be moving on towards the problems like fault simulation test generation and so on. Thank you.