VLSI Physical Design

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

Lecture - 48
Testing of VLSI Circuits

So, welcome back. So far in this course we have been studying the various aspects of VLSI physical design like you recall we started with the basic VLSI physical design flow. We looked at the essential processes involved like the partitioning, floor planning, placement, routing and so on then we talked about the timing issues we said that the timing issues in high performance circuits are very important. So, we looked at the various kinds of timing analysis, timing aware placement, timing aware routing and so on.

So, in this week and also in the coming week we shall be looking at another aspects of the design when the chips are fabricated, you see when you are designing the chip you are taking all the basic things into consideration like of course, the performance the timing and also the area and the other requirements, but during fabrication there can be some faults because of the fabrication because of the design also, the source of the fault can be several. So, before you can send or ship a product you have fabricated to the market, you need to thoroughly test the product. So, testing of a fabricated device or a board whatever you say is very important in the context of the overall design flow of a VLSI system. So, the topic of our lecture today is Testing of VLSI circuits.

(Refer Slide Time: 02:05)

So, let us first try to motivate ourself why do we need testing. So, as I have just said that the fabrication processes are becoming more and more complex with the advent of the deep sub micron design technologies. So, now, the features which are there in a design there are coming closer and closer together, they are becoming smaller and thinner. So, the chances of two wires let say touching each other or a very thin wire breaking in between are becoming also pretty significant, these are the so called sources of errors or faults. So, there can be such errors that can creep in during the design and fabrication processes.

Not only that you see we are typically using some cad tools from some of the vendors' right. Now this cad tools themselves are very complex software programs, there is always the chance of some bugs existing in those programs. So, when you are translating a design specification finally, into your layout level specification, there can be some bugs which can creep in during this translation and also when you are means after fabricating when you are doing the packaging, there also there can be some faults because of incorrect or imperfect connections and so on. So, the bottom line is we need to test each and every chip before they can be shipped. So, this is what I have said earlier.

(Refer Slide Time: 03:49)

Basic Objective

- We use testing to determine the presence of faults in a given circuit / chip.
  - *Fallacy: Testing is used to guarantee that a circuit / chip is fault-free.*
  - No amount of testing can give this guarantee.
  - Using testing, we can increase out confidence in the correct working of the circuit / chip.
- We usually use verification along with testing.
  - Distinctly different objectives.

So, what is the basic objective of testing? Well, the truth is we use testing to determine the presence of faults, it can be a chip, it can be a circuit, which consists of several chips. So, when I have a such a system which is a single chip or a collection of chips, we want to determine the presence of faults, but there is a fallacy well sometimes we tend to think that we are using testing to guarantee that a circuit or a chip is free from any faults, but this is a false statement this is not true; why? Because you see mean you are doing testing how do I testing? Given a circuit you typically you apply some inputs you observe some outputs, and also there are some electrical characteristics like the delay switching and other characteristics also you test.

Now, the now issues, there are so many environmental variations possible like temperature, humidity, pressure, vibration and so on. So, who will guarantee that well I am carrying out testing in an environment where my ambient temperature is 32 degree Celsius, but if my temperature rises to 35 degrees, my circuits might fail because some transistors might not be working properly, may not be switching as per the specification. So, it is not really possible to test against all possible environmental variations and the possibilities, this is why we say; however, you elaborately we carry out testing, we can never guarantee that the product we are manufacturing is free of any faults.

So, what we are trying to achieve? We can only increase our confidence, in the correct working of the circuit and there is an auxiliary process which is also involved called

verification, we usually use verification along with testing to improve our confidence in the correct working of the circuits and devices. But verification and testing have very different objectives let us try to see what these are.

(Refer Slide Time: 06:14)

Difference between Verification and Testing

- Verification guarantees the correctness of the design.
- Performed once before the actual manufacturing of the circuits / chips.
- Primarily responsible for the quality of the design.
- Uses formal methods, simulation, etc.

- Testing tries to guarantee the correctness of the manufactured circuits / chips.
- Has to be performed on every manufactured device.
- Primary responsible for the quality of the devices that go to the market.
- Two steps involved: (a) Test Generation, (b) Test Application.

So, here we show the differences between verification flow and the testing flow. Verification basically tries to guarantee the correctness of the design, design means we have not yet fabricated the circuit. So, our design is available either in the form of a high level description like in verylog or VHDL, or it is available in the form of some kind of a netlist, register transfer level or gate level netlist. So, we want to verify whether the design that is available at that level conforms to our desired specification, and this is performed only once, because we are performing the we are actually the evaluating the design not the fabricated chips. So, it is performed once before the actual manufacturing starts, and because we are assessing the correctness of the design this process assures quality of the design and typically methods like formal theorem proven techniques various formal methods, sat based techniques many are the available, and also simulation based techniques are used to carry out verification.

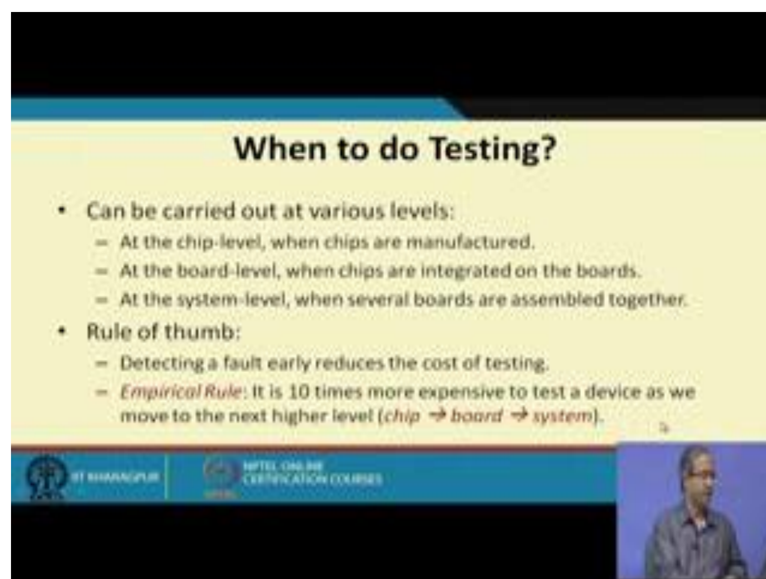
Now, in contrast the process of testing it tries to guarantee the correctness or the manufactured chips or circuits. Now see I have made a difference in the two step here, I mentioned verification guarantees and here I said testing tries to guarantee. Because as I have just now said using testing you can never guarantee 100 percent free from faults or

failures, but verification is a formal process it is some kind of a proof that we have given mathematically, whether your design is correct or not.

Testing naturally as I had mentioned has to be performed on each and every device that you are manufacturing, because each and every device can be faulty. And by doing testing you are improving the quality of the devices that you are selling to the market. During testing two steps are involved, one is done a single time this is called test generation. Test generation means given a circuit I want to find out what are the inputs I need to apply to the circuit, so that I can test it in the way I want that is called test generation.

Now, for a given circuit test generation is done only once, but for each and every chip you have to actually apply those test vectors, it is called test application; test application is done once for every device.

(Refer Slide Time: 09:22)



So, now we look into the issue that exactly when do you do testing; we do testing when the chips are fabricated or do we do testing when we have already put the chips on a board the boards in a system and the overall system is ready, we want to test at that level. So, there are implications.

So, we can actually do or carry out testing at various levels; like for instance you can do it at the chip level, while the chips are getting manufactured. So, you can also do testing

at the next higher level namely the board level, where several of these chips have been integrated on a printer circuit board and thirdly at an even higher level, when you are building a system there will be several such boards in the system. So, when all these boards are assembled together to form the system you can even carry out testing at that level.

Now, there is an empirical rule of thumb which exists, this is sometimes also called the rule of 10, this roughly says that if you are able to detect a fault early, it will reduce the overall cost of testing. And this empirical rule says that it is 10 times more expensive to test a device as we move to the next higher level for example, chip level to board level, board level to system level like for example, when you have manufactured a chip, if you give me that well I want to test this chip well I can tell you that you apply these inputs to the chip and see whether these outputs are coming. So, I can tell you some procedure for testing the chip.

But suppose you have a board where there are 10 such chips already soldiered, and you suddenly ask me that this board is not working, so how do I found out what the fault is. Now this fault can be in any one of these 10 chips right. So, it is much more difficult to identify and diagnose the fault, where the fault is located and what is the fault line right. So, if you talk about even higher level several such boards, the problem of testing and fault diagnosis is even more. So, sometimes what happens many of the manufacturers when they find some fault in a board, they simply replace the board by a new board instead of trying to find out where the fault is right, because the cost involved is pretty higher cost in terms to time and effort.

(Refer Slide Time: 12:00)

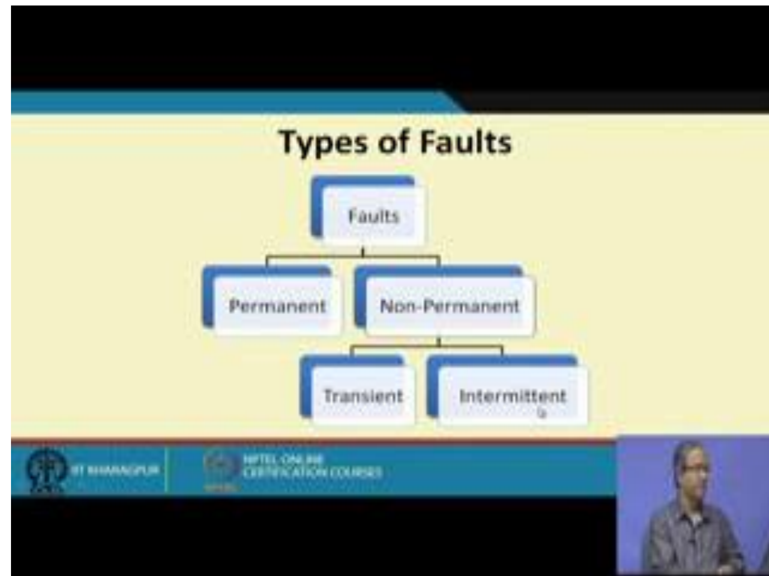
What are the Sources of Faults?

- Because of errors during the fabrication process.
  - Missing contact window, parasitic transistors, etc.
- Because of defects in the material(s).
  - Cracks or imperfections in the substrate, surface impurities, etc.
- Because of ageing.
  - Dielectric breakdown, electron migration, etc.
- Because of defects during packaging.
  - Contact degradation, disconnection, etc.

So, talking about the sources of faults, so we said that the faults can be because of errors during fabrication process; like we actually fabricate some rectangular patterns on the surface of the silicon. So, some of such rectangular patterns may be missing this is called missing contact window. Accidentally some diffusion polysilicon layers might overlap during to parasitic transistors and so on. There can be defects in the materials on which the chips or the layers are getting fabricated like the substrate silicon substrate, there can be some cracks or imperfections or some dust particles on the surface, which may lead to some imperfections in the layers which are fabricated on top of it.

Now, you may recall that the geometry of the features that you are fabricating that is very much comparable with the size of the smallest speck of dust. So, having a dustless environment during fabrication is extremely important. So, there can be some errors which may appear because of prolonged usage, which is called ageing, some dielectrics might breakdown there is a process called electron migration and so on and lastly there can be defects when you are putting a small chip inside a plastic package and connecting the pins of the chip to the pins of the package. So, there can be some imperfections in the contacts the wires are using to connect them. So, there can be so many sources of faults.

(Refer Slide Time: 13:52)



Broadly speaking this diagram shows how we can categorize the basic types of faults, like in the highest level the faults can be categorized as permanent or non permanent, and the normal non permanent faults on the other hand can be categorized as transient or intermittent, so let see what these are what are the characteristics.

(Refer Slide Time: 14:20)

- *Permanent Faults* change the functional behavior of a chip in a time-independent (permanent) way.
  - Design errors, incorrect connections, etc.
  - Easier to detect.
- *Non-Permanent Faults* occur randomly and at unpredictable times and for unpredictable time durations.
  - Difficult to detect.
  - The fault may not show up during testing.
  - On-line testing is a popular method.

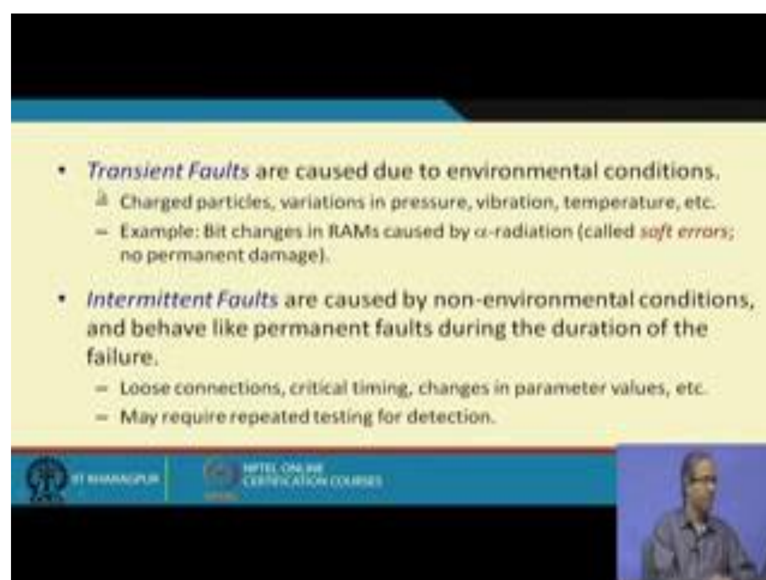
The permanent faults as the name implies they are permanent, means they change the behavior of a circuit or a chip in a way which is not dependent on time which is permanent. This can happen due to design errors maybe your fabrication is perfect, but

your design was wrong, because of some incorrect connections and so on. These kinds of faults are much easier to detect, because when you are carrying out the testing it is guaranteed that this kind of faults will be present because it is permanent.

But in contrast non permanent faults do not appear at all times. They occur randomly, they show up at unpredictable times, and when they show up they will remain present for unpredictable durations. So, you see there is no there is no guarantee that when you are actually testing a chip, this kind of non permanent fault will show up. Maybe right after the testing is over you declare the chip is good some of these faults show up. So, these faults are relatively much more difficult to detect, and for this kind of faults there is a methodology called online testing which is quite popular.

Now, what is online testing very broadly speaking? Here we are using some kind of codes, meaning just by looking at the output of a circuit I can say that whether this circuit is a valid output or an invalid output; like a simple example I can say that the number of ones in the output will always be odd, this is how I design my circuit. So, if during operation I find that some of the output is coming with odd not odd even number of wants, then I can declare immediately that there is a fault somewhere. So, this checking I am doing using hardware continuously during normal circuit operation that is why this is called online testing, right.

(Refer Slide Time: 16:40)

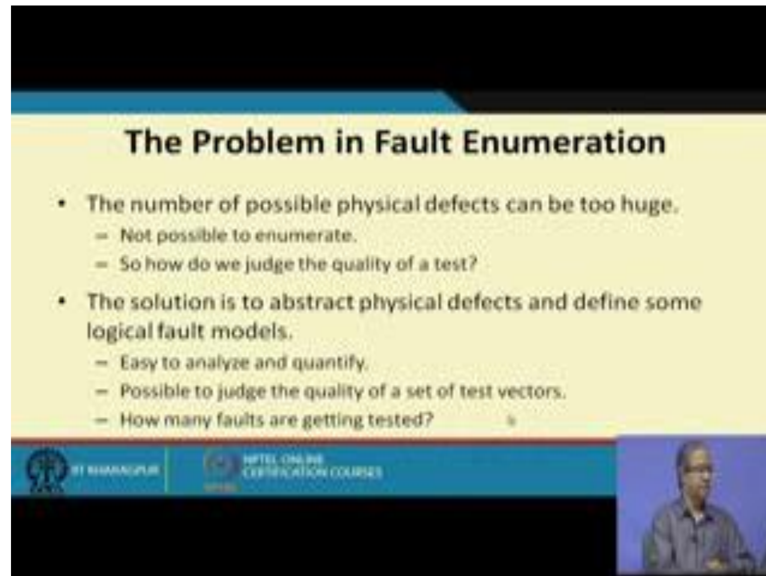


Now, under the non permanent faults transient faults occur due to some environmental conditions, which sometimes happen sometimes they do not happen like you may be working in an environment where there are lot of charged particles, there can be variations in pressure, vibration, temperature; like one classic example is this happens whenever we use some computer systems, let us say in an aircraft or in a satellite or in an some kind of a space machine and so on. So, you know that when todays computer system we use the memory systems, memory units using something called dynamic ram dynamic memory. So, in a dynamic memory we store the information not as a flip flop, as in a classical statics storage device, but as the charge stored on a tiny capacitor.

Now, if such a chip is exposed to external charge particles radiations like alpha particles let us say. So, these particles can penetrate the surface of the chip, they can go inside the chip, they can hit the capacitors and the charge might get discharged. So, some bit which was stored as 1 might become 0; these are sometimes called soft errors because these are not because of any hardware damages or faults, but because of a temporary situation let say some alpha radiation or a alpha particles are hitting, but after some time these will go out and again the circuit will start working correctly right.

In contrast intermittent faults they are caused due to non environmental conditions like loose connections, the timing is very critical; sometimes it is meeting sometimes it is not meeting, changes in parameter values of transistors resistances over time. So, this kind of faults are very difficult to detect, this may require repeated testing for detection. So, the kind of faults that we talked about in our subsequent discussions, we will be assuming that there are permanent in nature. For handling the non permanent faults as I said some kind of online fault testing is done, which is a little beyond the scope of our discussion in this course.

(Refer Slide Time: 19:10)



So, there is another issue we talk about fault enumeration, like can we count the number of possible physical defects? We cannot because they are too huge, I can say that the value of a resistance is changing, the resistance is supposed to be 100 ohm, it has changed. But that is not a single fault, I can say the 100 ohm has become 101 ohm; 100 ohm has become 100.11 ohm, 100.111 ohm. So, there are infinite such variations which are possible in terms of the faults, we really cannot count or enumerate the total number of such faults which has happened right these are physical defects.

So, now the question arises. So, if we accept the fact that number of defects can be infinitely large. So, how do we judge that the testing we carry out is good or bad, because anyway we cannot address all possible defects right? So, what we normally do is, we abstract these physical defects in some way and define something called logical fault model, which is some kind of an abstraction, which is much simpler to understand and analyze; means simpler to analyze here you can count how many faults are possible in terms of this fault model, then you can tell that well I have a test in that test 90 percent of these faults are getting detected, this is possible here you can judge the quality of a set of test vectors, you can actually count that for a given set of test vectors. So, how many of these faults are getting tested.

(Refer Slide Time: 21:05)

Some Terminologies

- *Fault Coverage*: Percentage of the total number of logical faults that can be tested using a given test set *T*.

$$FC = \frac{\text{Number of detected faults}}{\text{Total number of faults}}$$

- *Defect Level*: Fraction of shipped parts that are defective.

$$DL = 1 - Y^{(1-FC)}$$

where *Y* is the yield.

Now, some terminologies are defined here, the first is something called fault coverage; this determines the quality of a set of test vectors, this is defined as the percentage or the fraction of the total number of logical faults that can be tested using a given test set; that means, given set of test vectors. So, it is defined as the ratio, number of detected faults, divide by the total number of faults. So, if you want to express it as a percentage, multiply this by 100. Defect level is another term which is also used sometimes, this says fraction of the shipped parts that are defective; that means, we are manufacturing the chips we are testing them, then we are sending them to the market, now what fraction of those chips can be detective defective?

Now, there is a factor called yield. So, if also depends on the yield wield, yield actually tells you that what is the fraction of the chips that are fabricated which are good. So, out of them 1 minus FC will be those fraction of the chips which faults are not being covered. So, this is a factor which tells you that what is the proportion of the chip which you cannot detect a fault, but you have declared it as good. So, if FC is 1 which means all the faults have been detected you see DL will be 1, defect level fraction of shipped parts that are defective will be means 1 minus 1 0 right. So, if FC equal to 1 this will be y to the power 0 1. So, this is sometimes used to measure this fraction of ships parts that are defective.

(Refer Slide Time: 23:01)

Why Testing is Considered Difficult?

- Consider a combinational circuit with *N* inputs. Suppose we use a naive way of testing where we apply all 2^N inputs and verify the truth table.

N	Number of Test Vectors
25	33.55×10^6
50	1.13×10^{15}
100	1.26×10^{30}

Not feasible as N increases.

- Such naive approach is not feasible as circuits increase in size.

Now, another thing testing is not an easy problem, let us try to justify this ourselves first that why testing is difficult? We first look at a very naive approach, we take a combination circuit there are N inputs. So, in the simplest way we can test the circuits by verifying the truth table, verifying the truth tables means we can apply all two to the power N inputs and check whether the output is coming to be correct or not.

But the problem is you see as N increases 25, 50, 100 well in practical circuit N is much higher I have shown only upto 100 and here the value of 2 to the power N is shown, you can see for 25 it is about 33 million it goes to means 1 into 10 to the power 30 for N equal to 100. So, clearly this is not feasible for large values of N, as N increases we cannot use this kind of naïve approach of verifying the truth table.

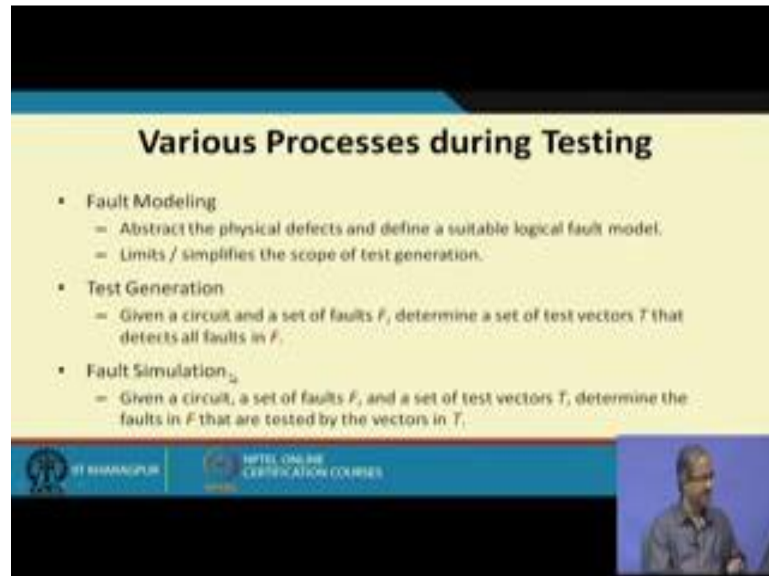
(Refer Slide Time: 24:06)

- Now consider a synchronous sequential circuit with *N* inputs and *S* state variables.
  - For any given input, the output will depend on the internal state.
  - 2^N possible inputs and 2^S possible states → complexity of $O(2^{N+S})$.
- Thus, verifying the state table of the sequential circuit for testing in infeasible.
  - Need some mechanism to enhance the controllability and observability of the state variables.
  - *Design for Testability* is a standard option that is used in practice.

So, if you now consider a synchronous sequential circuit, the problem is even more difficult. There are N number of inputs there are S number of state variables flip flops. Now in the sequential circuit when you apply a input, we cannot predict what the output will be because the output will be dependent on this state of the flip flops. So, if there are S number of flip flops they can be in 2 to the power S possible states. So, for each of the 2 to the power S is possible states, the output can be different, right. So, you must be sure which state the flip flops are then only you can apply the input and in a normal sequential circuit is not so easy always to initialize the flip flops to known state right. So, 2 to the power n possible inputs, coupled to 2 to the power s possible states, this results in a complexity of 2 to the power n plus s.

You compare this to the combination circuit where this second part was not their here only two to the power n was there right. So, verifying the state table of the sequential circuit for again infeasible because extremely large. So, we need some mechanism to control and observe the states of this internal flip flops; these are called controllability and observability of the state variables. There are some techniques called design for testability, which is used primary to address this concern. So, how to make this internal state variables of flip flops? Easily controllable or easily observable.

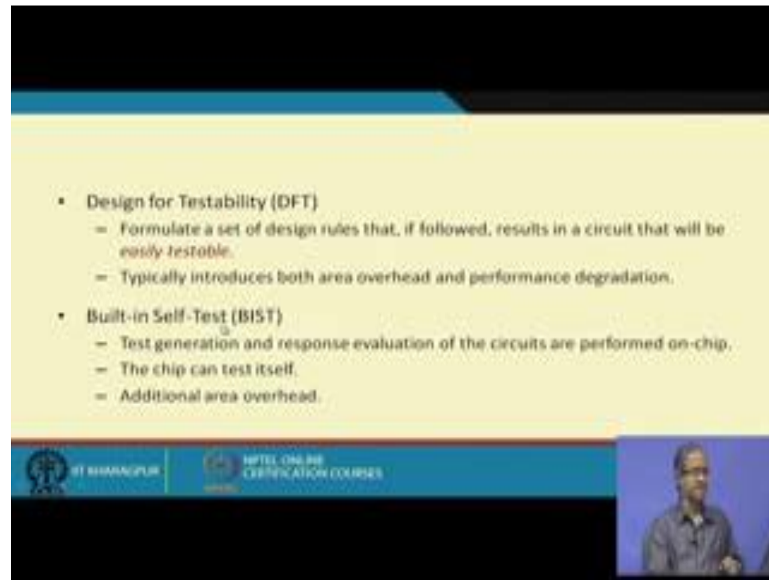
(Refer Slide Time: 25:53)



Now, let us quickly look at what are the different processes of testing some of these we shall discussing our subsequent lectures, first of course, you already mentioned fault modeling, because the number of physical defects can infinitely large, here we abstract the physical defects and define some logical fault model. By doing this we can say simplifies or limit this scope of test generation, because now you can say that want to test only faults that belong to this fault model, we have were restricted our selves.

So, next step will be to actually generates the tests given a circuit and a set of faults F may be under logical fault model, here we are trying to determine a set of test vectors, that can detect all the faults in this fault set F. Sometimes we may want to check that will given a set of vectors how many faults are getting detected. So, there is another process is called fault simulation which is used there. Here the set of faults is given, set of test vectors is given, we determine the faults that are tested by the vectors that are given. So, we can also identify that these are the fault which is still not tested; this is done through fault simulation.

(Refer Slide Time: 27:25)

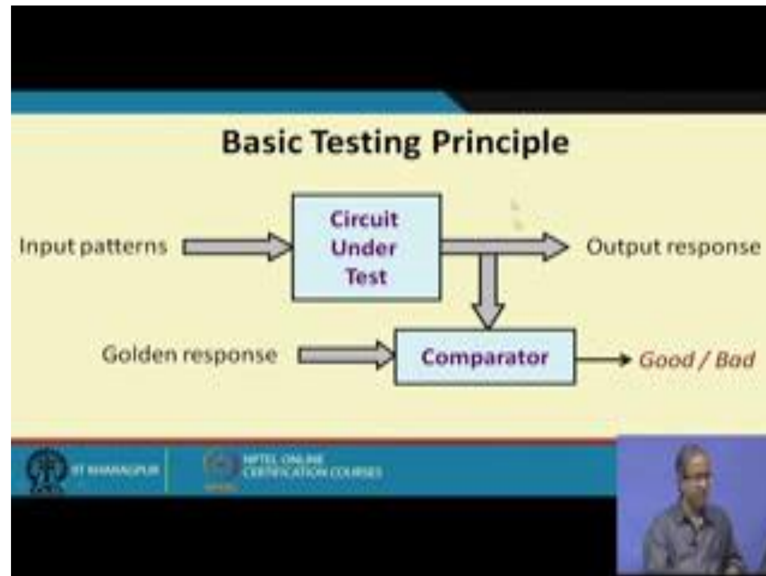


Design for testability I just now mentioned, basically these consists of a set of design rules which the designer must follow in a religious way, and what is the end result? If they are followed we will be getting a circuit that will be easier to test, but nothing comes free in order to do or achieve this, we will have to introduce some extra hardware; that means, extra area overhead and also some of this hardware will following in the critical path, which will be slowing down the block which means a little bit of performance degradation as well.

Now, the extreme case we can go for something called built in self test. See in the ideal scenario it will be it will be very fantastic to have a chip, which can test itself and it can tell us that will I am good or I am bad. So, I do not have to do any testing from outside, this principle or philosophy is called built in self test. So, some kind of test generation and response evaluation are carried out on chip. Now quite naturally this chip this test generation and response evaluation should be such that they can be implemented with very nominal hardware overheads.

So, if I say that I need a big memory to store my test patterns inside the chip that can be too much of an overhead. So, if we can do this then the chip can test itself well here of course, this test generated and response evaluated must be there inside the chip, and the additional control circuit. So, this will also incurred some area overheads.

(Refer Slide Time: 29:19)



So, this diagram gives you the overall testing principle like given a circuit which we want to test we have to apply some inputs, the outputs are coming and we have to compare this outputs again some golden response; and after this comparison experiment is over, we can declare that the weather is circuit is probably good or it is definitely bad; good we cannot say 100 percent confidence, right.

Now, this process the different steps that involved shown in diagram, this can be done inside the chip like built in self test, the pattern application and this comparison can be done outside the chip. So, there are several levels in which you can do the whole thing, but this diagram gives you the overall picture.

So, with this we come to the end of the lecture in the next lecture, we shall be looking into some aspects of fault modeling because that is the first step before you can proceed towards the other aspects of testing.

Thank you.