VLSI Physical Design

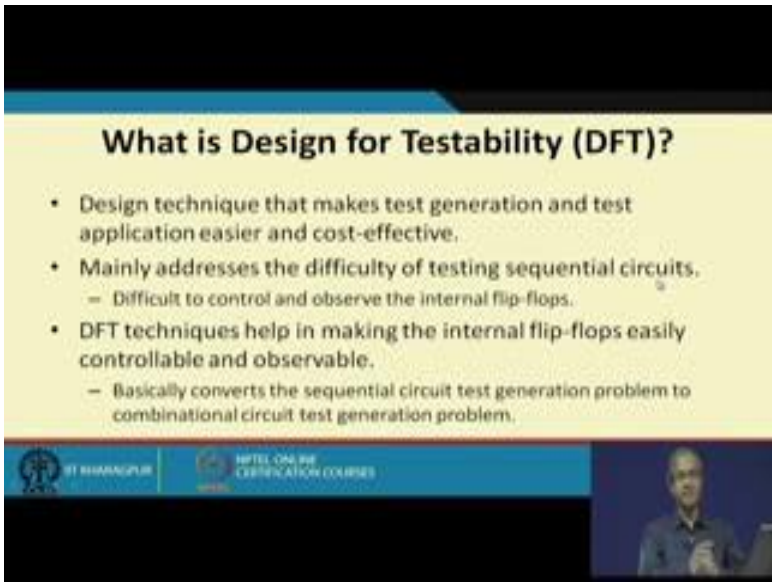
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 54
Design for Testability

So, in the last lecture we talked about test pattern generation, how we can generate tests. Now one comment we made towards the end we said that combinational ATPG is reasonably we have tools that can generate combinational test vectors for even large circuits, but sequential ATPG is a problem. Firstly, it takes lot of time. Secondly, it does not provide good fault coverage meaning it is often not able to detect tests for many of the faults, primarily because of the existence of the storage elements inside or the flip flops inside it is not so easy to initialize the flip flops from outside.

Now, unless we initialize flip flop, so when we apply input you cannot say what the output will be right. So, we have to go for some techniques called design for testability, because almost all the practical circuits are sequential in nature. So, we cannot use a sequential ATPG tool because they are very slow and not so good. So, we try to solve these problem in a different way let us see that.

(Refer Slide Time: 01:33)



What is Design for Testability (DFT)?

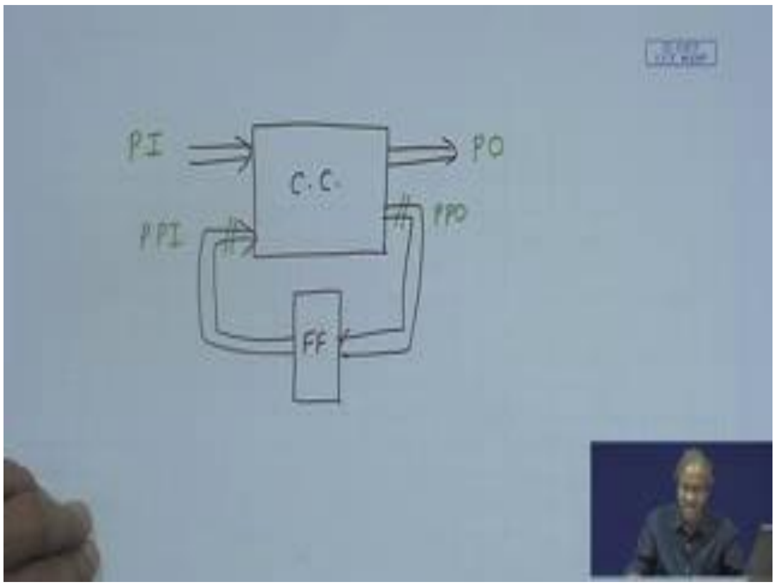
- Design technique that makes test generation and test application easier and cost-effective.
- Mainly addresses the difficulty of testing sequential circuits.
  - Difficult to control and observe the internal flip-flops.
- DFT techniques help in making the internal flip-flops easily controllable and observable.
  - Basically converts the sequential circuit test generation problem to combinational circuit test generation problem.

So, what is the basic motivation behind design for testability? DFT means they stand for a set of designed techniques that makes test generation and test application easier and cost effective. This is a very general statement, but how? Specifically the DFT techniques that are used in practice, they address the difficulty of sequential circuit testing. You see I mentioned that for sequential circuit the main hurdle are the flip flops, they are sitting inside the black box which is the circuit and I can apply inputs only from outside.

So, unless I initialize the flip flops I cannot run a test this is a major problem of sequential circuit. So, for any design for testability technique, I have to address and find the solution to this question how to initialize the flip flop rather easily, and how to observe the states of the flip flops whenever I want to. Normally both these questions are difficult in a general sequential circuit, but now you are saying we would be framing some design rules design for testability rules, which if we follow it will make our circuits easily testable right.

So, the main issue was difficult to control and observe the internal flop flops. So, essentially the DFT techniques which are used, so one of them we will be discussing here, they try to make the internal flip flops easily controllable and observable. Controllable means you can initialize them into any value you want; observable means we can see or read out their values whenever we want. So, if you can do this then you see we have a very interesting you can say corollary to this. So, we are converting the sequential circuit test generation problem to a combinational circuit test generation problem, how let me just show you schematically.

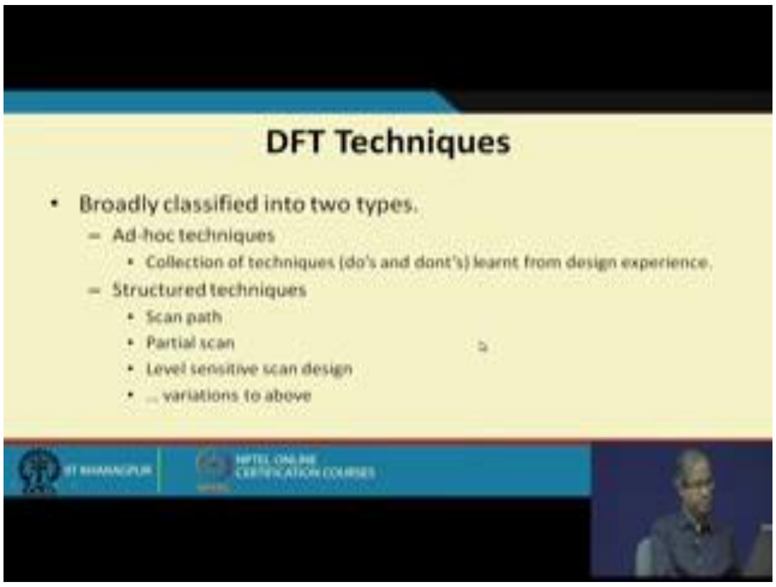
(Refer Slide Time: 04:00)



So, in a schematic sense a sequential circuit looks like this, this is the combinational circuit part, these are the flip flops, you have the inputs coming outputs going, and some outputs are coming to the flip flop, something like this.

Now, what we are saying is that you see these are my primary inputs, these are my primary outputs. Normally in a classical sequential circuit testing these are the only inputs and outputs I have. So, via this PI we try to initialize the flip flops that is why the problem becomes complex, but if we have a mechanism let us assume we have that we can initialize flip flops to any value you want easily, then you can say that well these inputs also can be easily set, we call them as pseudo primary inputs. Similarly when we say we can observe the values of flip flops easily, here as if we are saying that we can observe these outputs of combinational circuits also, because it is these outputs that are going to the flip flop, these are called pseudo primary outputs. So, you see once you have this then I can forget my flip flop. So, I can treat this whole thing as a combinational circuit where PI and PPI are my inputs PO and PPO are my outputs, and I can use a combinational circuit test pattern generator to detect all falls here. So, I am not worrying about the flip flops anymore, but of course, how to test the flip flops that we will have to address separately fine.

(Refer Slide Time: 05:49)

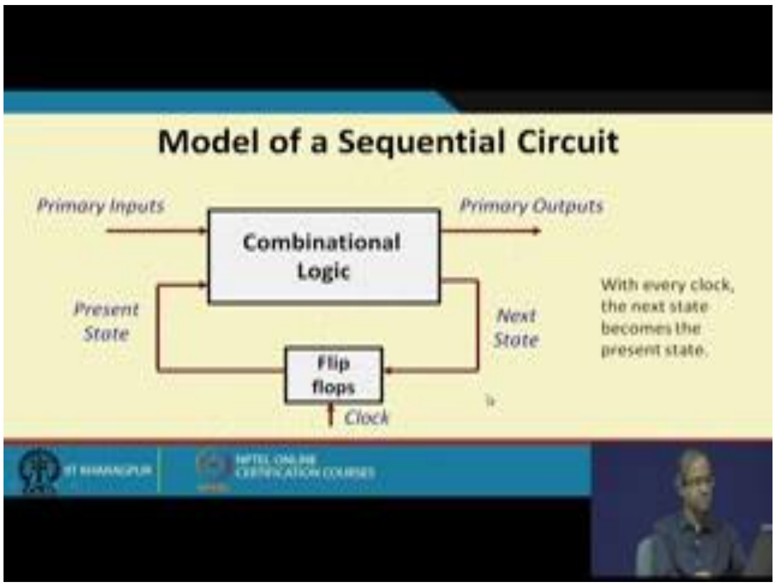


DFT Techniques

- Broadly classified into two types.
  - Ad-hoc techniques
    - Collection of techniques (do's and dont's) learnt from design experience.
  - Structured techniques
    - Scan path
    - Partial scan
    - Level sensitive scan design
    - … variations to above

So, talking about the DFT techniques broadly they can be classified as either Ad-hoc or structured techniques. Now Ad-hoc techniques as the name imply they are really Ad-hoc, they are a collection of techniques; that means a long list of do’s and dont’s. So, some designers experienced designers they have provided a list, that you should not get a clock you should not do this, you should not do that. So, if you follow those rules, then your circuit will be a little better from the testing point of view. But structure techniques can totally solve the problem as I have just now mentioned. There now here we shall be looking at the scan path approach, there are variations to the scan path approach partial scan, level sensitive scan designs and several others; but if you look at the basic idea behind scan path, you will get the feeling that exactly what you are trying to do.

(Refer Slide Time: 06:49)



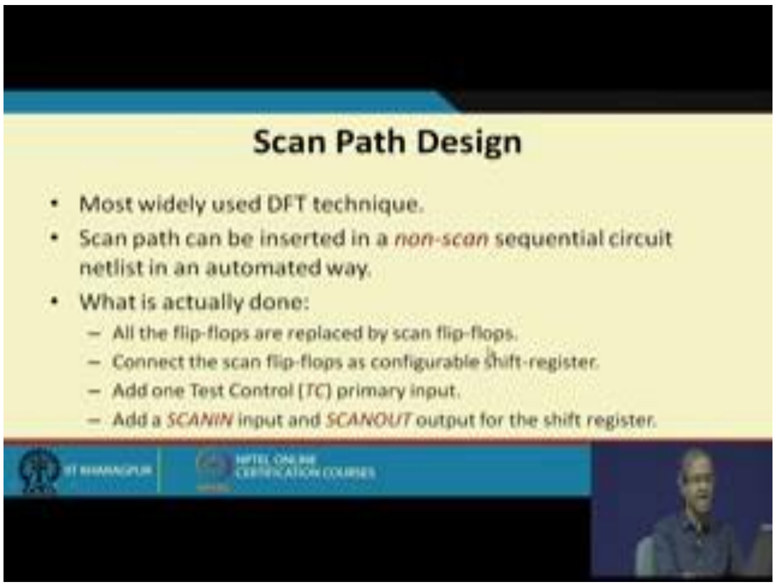
Model of a Sequential Circuit

With every clock, the next state becomes the present state.

So, this is the model of the sequential circuit that I showed just now, that you have a combinational logic where the primary inputs and primary outputs are coming and going, you have the flip flops which are fed by a clock. Now these outputs we refer to as the next state, and these inputs that are coming from the flip flop they are referred to as the present state.

So, you see whenever a clock comes the active edge of the clock comes. So, whatever is here that gets stored so the next state will become the present state right. So, with every clock this next state will come here this will become the present state, and the next output that comes out from here that will become the new next state this process continues.

(Refer Slide Time: 07:42)



Scan Path Design

- Most widely used DFT technique.
- Scan path can be inserted in a *non-scan* sequential circuit netlist in an automated way.
- What is actually done:
  - All the flip-flops are replaced by scan flip-flops.
  - Connect the scan flip-flops as configurable shift-register.
  - Add one Test Control (*TC*) primary input.
  - Add a *SCANIN* input and *SCANOUT* output for the shift register.

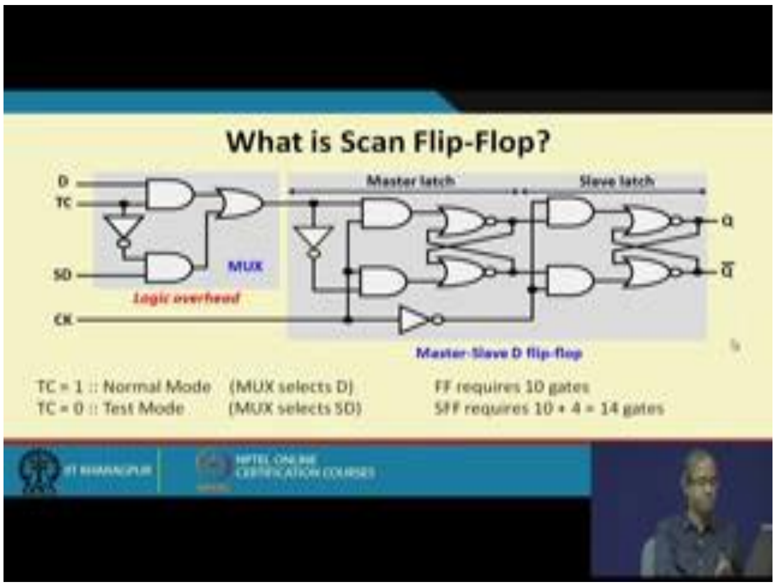
Now, we come to the scan path design for testability technique. Now as I said this is one of the most widely used technique. So, what we do? We take a conventional sequential circuit just like this to start with. So, here there is no concept of scan path, we call it a non scan sequential circuit. So, we can modify this circuit using an automated tool very easily to insert something called as scan path, this will see what is a scan path. So, what are actually done, see here we are saying that we are modifying this circuit, actually you are not modifying the first part you are modifying this part the flip flop side. So, what we are actually doing is, we are replacing the flip flops typically the D flip flops, they are replaced by a more complex version which is called as scan flip-flop.

So, a scan flip flop is essentially a D flip flop with a 2 to 1 multiplexer before it. So, a multiplexer followed by a D flip flop that is a scan flip flop. So, we interconnect this scan flip flops in a suitable way, so that we can configure it as a shift register if you want. We add one extra additional input from outside, we call it as the test control input and we add 2 more input and output lines called Scanin and Scanout as the input of the shift register and the output of the shift register.

So, the idea is quite simple, that I have a test control from outside. So, if I set it to one, then all my scan flip flops will be configured as a shift register. Now once it is configured as the shift register, I can feed some data from outside from my Scanin pin, and I can serially shift the data to all the flip flops. So, I can initialize them very easily.

Similarly if I apply clocks the shift register data will be shifted out from this scanout pin, so I can observe the states of the flip flops also. So, this is the basic idea how we are solving the controllability and observe ability problems.

(Refer Slide Time: 10:10)



What is Scan Flip-Flop?

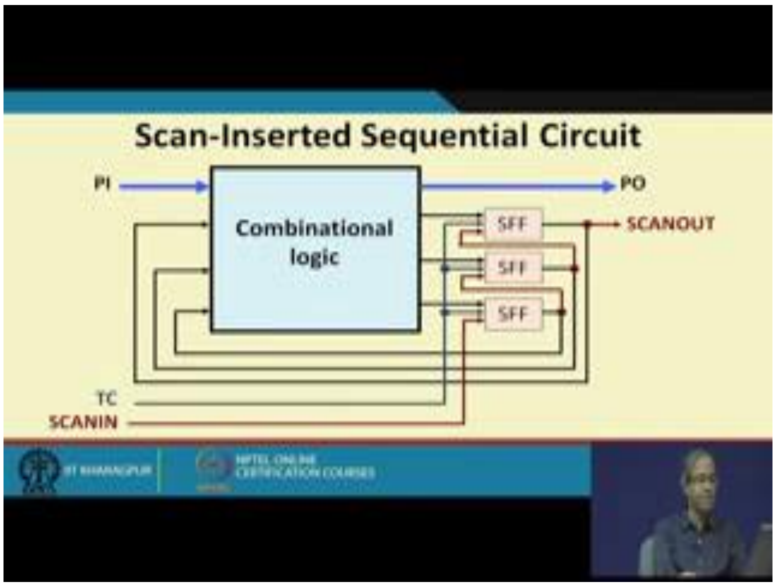
TC = 1 :: Normal Mode (MUX selects D)
TC = 0 :: Test Mode (MUX selects SD)

FF requires 10 gates
SFF requires 10 + 4 = 14 gates

So, as I said a scan flip flop is essentially a conventional flip flop with a 2 to 1 multiplexer before it. So, here I am showing a classical diagram using gates, but of course, in actual VLSI chip these will be implemented using MOS transistors, but let us try to do a quick evaluation using gates.

So, as you can see in the normal D flip flop masters left eye there are 10 gates, and for the multiplexer your logic over it will be 4 additional gates and this TC which is the select input of this multiplexer, if I set it to 1 this will be my normal mode. The D input will be going to the input of this flip flop multiplexer selects D, but if TC equals to 0 in this diagram which is my test mode, where I have another input I am calling it scan data or S D. This S D will come to this input of this flip flop so multiplexer selects S d. So, depending on the value of TC either D or S D is selected. So, you can see that the flip flop requires 10 gates, and this requires 14 gates so there is a overhead of 4 gates per scan flip flop.

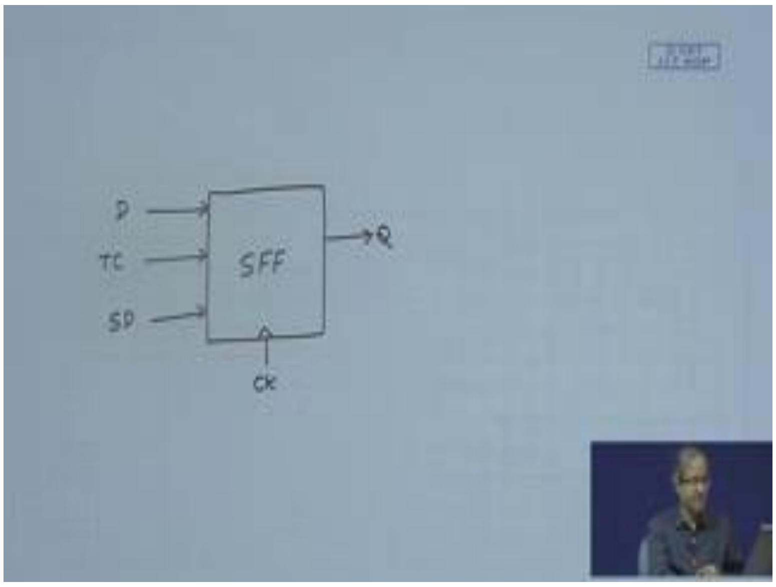
(Refer Slide Time: 11:42)



Scan-Inserted Sequential Circuit

Now, this diagram shows how we have modified our circuit to configure them as a scan flip flop; let us go back once more to this circuit you see this circuit where the inputs are D, TC, S D clock and Q bar is normally not required so only Q.

(Refer Slide Time: 12:09)



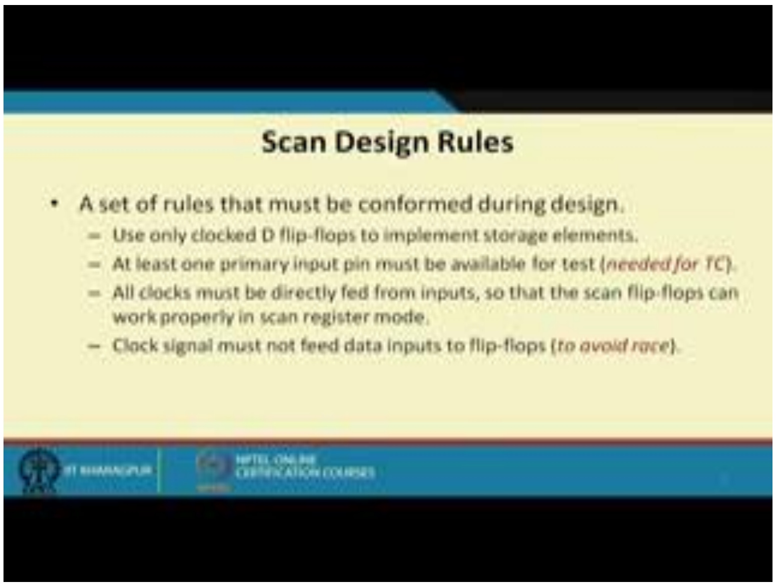
So, if I can show this as a schematic like here let say I show it as a black box, this is my so called scan flip flop. So, what are the inputs of my scan flip flop? I have a test control input, I have a D input, I have an S D input and I have a clock, in the output let us say I or Q, this is my schematic of a scan flip flop right.

Now, using this schematic I will be explaining next diagram fine. So, here I am showing an example where there were 3 flip flops, normally in the non scan version this flip flop outputs were directly coming to the input; like in the previous diagram I showed let me go back this diagram. The output of the flip flop are coming as the input to this flip flop, this was the normal version of the flip flop, now I am making some changes here.

So, what kind of changes you see? The test control input is going to the TC input of all these scan flip flops. The combinational circuit output is going into the D inputs all of them and scan flip flop output is going back to the input of the combinational logic just like it was. But the additional thing we have done is that we have added a scanin input that is going into the S D input of the first flip flop, and the fa in the output of this one is going to the S D input of the second flip flop, this is going to the S D of this and finally, this output is also going out as scanout.

So, if I say TC equals to 0, then in the input multiplexer my this S D line will be selected. So, now, my circuit will be a shift register you see this part will be selected. So, by configuring as shift register I can shift in any data from scanin. So, I can initialize my flip flop or I can observe the states of the flip flop. So, this is exactly what we are trying to do in the scan based design, we are trying to means as I said we are trying to reduce the complexity of sequential test generation problem into a combinational test generation problem ok.

(Refer Slide Time: 14:40)



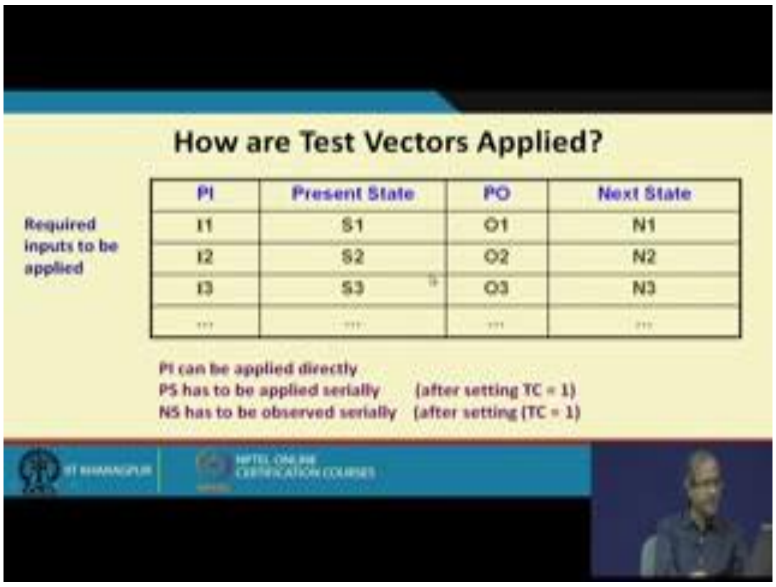
Scan Design Rules

- A set of rules that must be conformed during design.
  - Use only clocked D flip-flops to implement storage elements.
  - At least one primary input pin must be available for test (*needed for TC*).
  - All clocks must be directly fed from inputs, so that the scan flip-flops can work properly in scan register mode.
  - Clock signal must not feed data inputs to flip-flops (*to avoid race*).

So, some of the scan design rules means have been formulated which a designer needs to follow. So, the rules you can easily correlate with the just previous diagram this diagram that we had shown. The first one says that only clocked D flip flops should be used. So, you should not use j k or S r flip flops. Now in fact this is true for VLSI circuits, because in VLSI it is much easier to implement D type flip flops as compared to means other type of flip flop that S r or j k. So, this is an assumption which is followed almost universally today, and as I said we need at least one additional primary input pin the test control, and all the clocks must be directly fed; that means, it should not be a gated clock like here you see here although you have not shown the clocks, where the clocks are going directly into the 3 scan flip flops, so that they can work in a shift register mode.

So, clock signal must not feed any data input to the flip flop, this is one of the design rules this is used to avoid race condition, because depending on clock skew and other problems the data that is getting stored in the flip flop maybe wrong.

(Refer Slide Time: 16:08)



How are Test Vectors Applied?

Required inputs to be applied

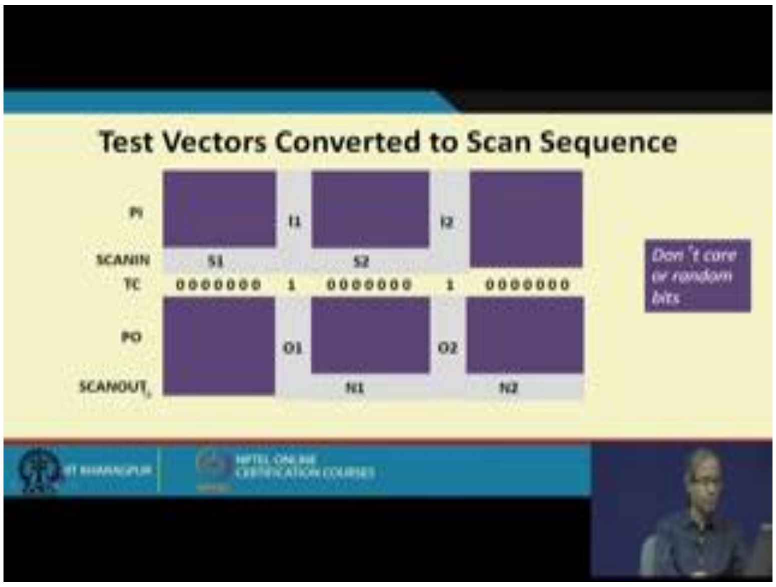
PI	Present State	PO	Next State
I1	S1	O1	N1
I2	S2	O2	N2
I3	S3	O3	N3
...	...	...	...

PI can be applied directly
PS has to be applied serially (after setting TC = 1)
NS has to be observed serially (after setting (TC = 1)

So, here I am giving showing an example for example, suppose this circuit as I said that it now looks like a combinational circuit. So, now, the inputs will be P I and present state the outputs will be PO and next state. Suppose I use a combinational ATPG tool to generate my test vectors. So, these are my inputs these are my outputs. So, let us say the ATPG tool has given me these test vectors this I 1, S 1 is the input and O 1 N 1 is the expected output.

So, I 2, S 2 is my second input, O 2 N 2 is my second expected output and so on. But the problem here is that the way we have configured like here this circuit, the P I part you can feed directly. This is our present state this is our next state. Now whatever is my present state I will have to load the flip flops with those values that will come here right. So, that can be done parallely, that has to be shifted in serially in shift register mode. So, the present state has to be applied serially after setting this should be TC equals to 0 because in diagram TC 0 means test mode.

(Refer Slide Time: 17:37)



Test Vectors Converted to Scan Sequence

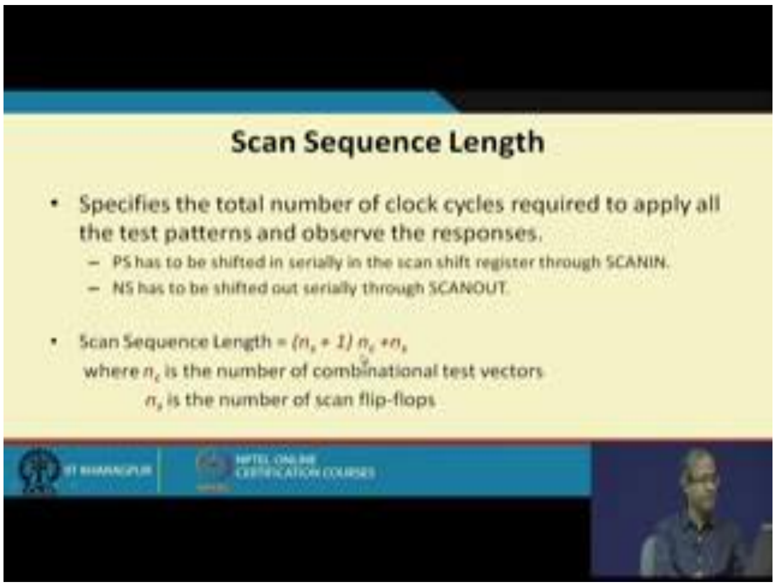
Similarly, next step has to be observed serially after setting TC equals to 0. So, this diagram shows in the excess of time what happens? From left to right this is the excess of time. So, I am showing the inputs and outputs of the circuits on one side, P I are my inputs scanin is a single input, TC is a single input, PO are my outputs scanout is another output.

Now you refer to this table, for the first is vector I have to apply I 1 S 1, I am expecting this O 1 N 1 as the output right. So, what I do first I set test control to 0 and serially I shift in S 1 through scan in, this S 1 was my this part present state part. So, I am serially shifting in S 1 while setting TC to 0. Now during this time P I can be do not care, this solid blue region indicates do not care random bits, but after I have shifted in what is the status. Now this S 1 is already there in the flip flops, now I can apply I can do 2 things together test control I set to 1, now it is a normal mode I apply this I 1 to the primary input directly.

So, now I can observe the primary output of the combinational circuit delay. You see here after we apply the inputs, so some of the outputs are going to the primary output and some of the outputs are going to the N 1, but it has not yet stored in the flip flop. So, after the next clock comes so it will get stored. So, what I do? I again set TC to 0, I apply clocks and through the scanout the bits of N 1 will come out.

So, I am applying S 1 serially, applying I 1 parallelly and observing O 1 parallelly, observing N 1 serially. Now one thing while N 1 is being shifted out in parallel I can also shift in the present state of the next vector; like here while I am observing N 1 I can parallelly shift in S 2, because it is a single shift register there is an output there is also an input. So, why the input line scanin, I can start shifting the next one. So, in this way it can proceed ok.

(Refer Slide Time: 20:19)



Scan Sequence Length

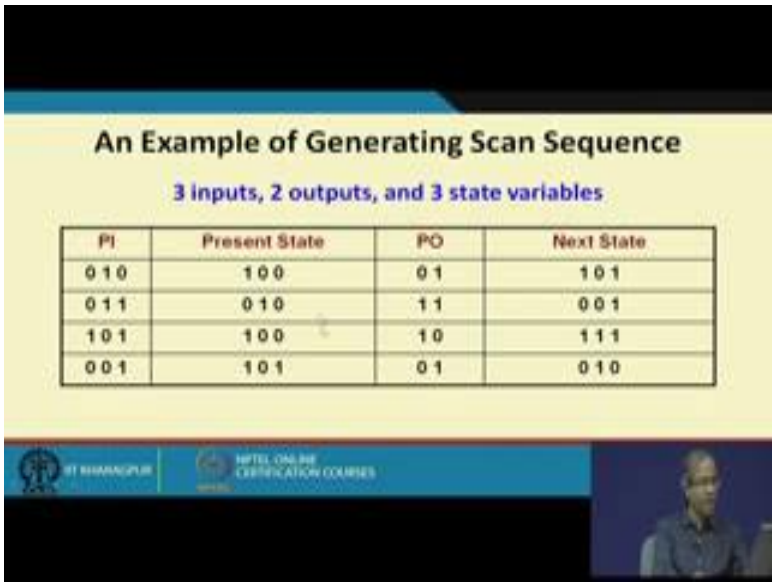
- Specifies the total number of clock cycles required to apply all the test patterns and observe the responses.
  - PS has to be shifted in serially in the scan shift register through SCANIN.
  - NS has to be shifted out serially through SCANOUT.
- Scan Sequence Length = $(n_s + 1)\ n_c + n_s$
  where n_c is the number of combinational test vectors
  n_s is the number of scan flip-flops

So, if you just look at a simple calculation. So, means you can determine the so called scan sequence length. So, what is scan sequence length? It tells you how many total number of clock cycles are required; like here in terms of the time total how many clock cycles are required. Just to look one thing you observe for the last let us say pattern. So, when you are observing the N for the last pattern, so for that this scanin there is nothing to shift in, this part will be empty.

So, here I am showing it only a small snap shot. So, this already I have mentioned. So, the total scan sequence length if you make a simple calculation to be n s plus 1 into number of test vectors, then n s plus 1 why? This is n s this 1 2 3 4 5 6 7 this is n s, this 1 for here, and you repeat this for all the test vectors and for the last one you need another n s to shift out the last n. So, this is my expression.

So, n c is the number of test vectors you have generated, and n s is the number of flip flops. So, it is roughly proportional to the product of n s and n c.

(Refer Slide Time: 21:42)



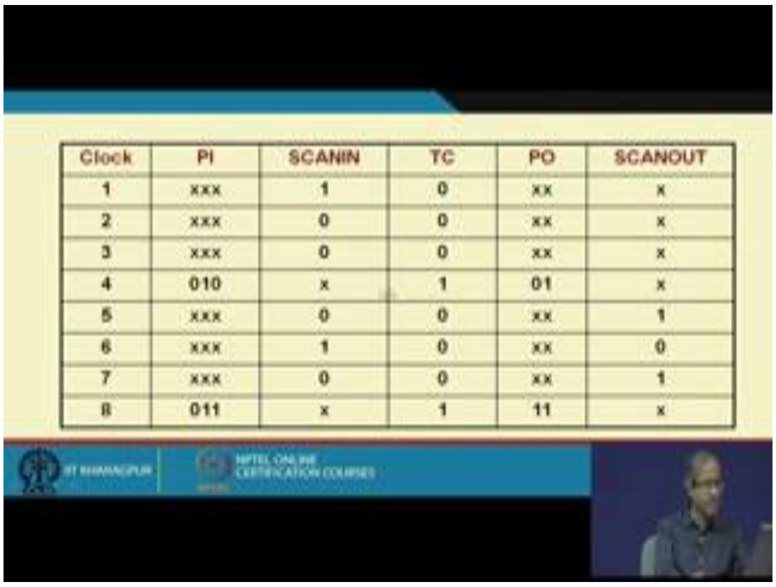
An Example of Generating Scan Sequence

3 inputs, 2 outputs, and 3 state variables

PI	Present State	PO	Next State
0 1 0	1 0 0	0 1	1 0 1
0 1 1	0 1 0	1 1	0 0 1
1 0 1	1 0 0	1 0	1 1 1
0 0 1	1 0 1	0 1	0 1 0

So, here I am showing a small example with actual numerical values, let us say we have a circuit with 3 inputs 2 outputs and 3 flip flops or state variables.

(Refer Slide Time: 22:08)

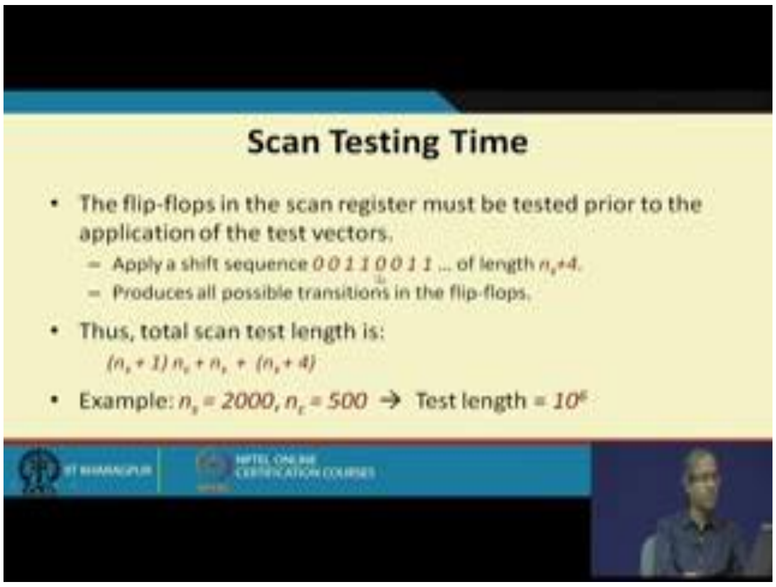


Clock	PI	SCANIN	TC	PO	SCANOUT
1	xxx	1	0	xx	x
2	xxx	0	0	xx	x
3	xxx	0	0	xx	x
4	010	x	1	01	x
5	xxx	0	0	xx	1
6	xxx	1	0	xx	0
7	xxx	0	0	xx	1
8	011	x	1	11	x

Suppose a test generator has generated these 4 test patterns so I am showing a part of this diagram, this is called scan sequence. I am showing only a part of this scan sequence in the formal table with the clockwise. You see for the first pattern the present state is 1 0 0. So, let us first shift this serially. So, for this we are setting TC to 0, we are shifting to scanin 1 0 0. During this time primary inputs are do not care, and P O and scanout we do not know or this can be empty.

After this I set TC to 1, and I apply this corresponding P I 0, 1 0, I apply 0 1 0. So, here when you apply 0 1 0, now the expected primary output 0 1 will appear after some delay. So, PO will get 0 1, but now to observe the other part the next state which is 1 0 1 what I do? I again set TC to 0 apply clocks, so our scanout I will get 1 0 1. Now while I was doing this I can shift in also the next present state 0 1 0, this also we had been through scanning 0 1 0 so these 2 things are getting overlapped. So, this will continue I am only shown a small snap shot of the process.

(Refer Slide Time: 23:24)



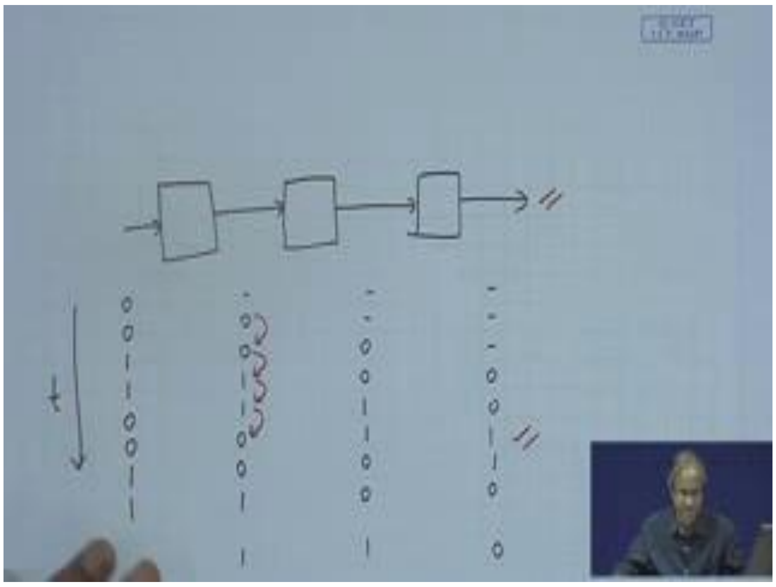
Scan Testing Time

- The flip-flops in the scan register must be tested prior to the application of the test vectors.
  - Apply a shift sequence *0 0 1 1 0 0 1 1* ... of length n_s+4.
  - Produces all possible transitions in the flip-flops.
- Thus, total scan test length is:

  $(n_s+1)\, n_c + n_s + (n_s+4)$
- Example: $n_s = 2000, n_c = 500$ → Test length = 10^6

Now, talking about the scan testing time, well we have only talked about how to apply the combinational test vectors, but what about the flip flops the shift registers. Now there is a standard way for testing shift registers, so this says that you apply a shift sequence 0 0 1 1 0 0 1 1 like this.

(Refer Slide Time: 23:50)



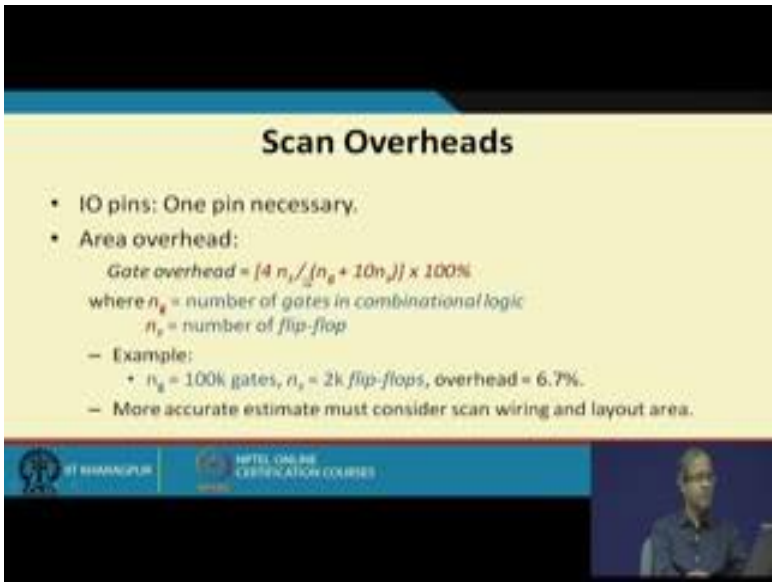
Let us take an example and show let us say what this means. Let us say I have 3 flip flops, they are configuring as shift registers. So, in time I am just applying let us say 0 I am showing like this, this is my axis of time 0 0 1 1 0 0 1 like this, this pattern I am applying. So, in the output of the first flip flop after one bit delay this bits will appear and this one will come out, then again after one bit delay this bit will appear here, 1 1 0 0 again the next one will go out. So, here again after one bit delay this will come, this 0 will come out.

So, now, you see you look at each of the flip flops, look at all the flip flops. See the flip flops are changing from 0 to 0, 0 to 1, 1 to 1 and also 1 to 0. So, I am exciting or testing the flip flop for all possible transitions, that is how we test a flip flop that it can store each logic value and it can also have all possible transitions. So, just by applying this pattern serially and if I observe that this same or the expected pattern is coming out here, so I can be sure that all these flip flops and also their interconnections they are working correctly right. This is how this shift registers testing is done or carried out.

So, the total number of clock cycle (Refer Time: 25:53) or the number of bits (Refer Time: 25:55) will be n s plus 4, n s will be number of flip flops and minimum 4 bits are required for this 0 0 1 1; so 0 0 1 1 plus the number of flip flops that many clocks are required. So, now, the total scan test time or the scan sequence length will be not only this, plus this additional n s plus 4 will also get added. So, just a small example if we have in a circuit 2000 flip flops and 500 test vectors, so test length will be approximately the product of n s and n c this is the dominant term, so it will be about a million, So, testing will become slower no doubt and means another thing also you should remember, but we are not testing the circuit at the rated clock frequency, we are shifting in the data in the flip flop then you are applying the input and seeing the output.

So, it is not that this that the circuit is supposed to work at 1 gigahertz, we are continuously applying inputs and 1 gigahertz and checking the output not like that. So, we are not doing something called at speed testing, we are doing it in a slower rate.

(Refer Slide Time: 27:09)



Scan Overheads

- IO pins: One pin necessary.
- Area overhead:

  Gate overhead = $[4\, n_s / (n_g + 10 n_s)] \times 100\%$

  where n_g = number of *gates in combinational logic*

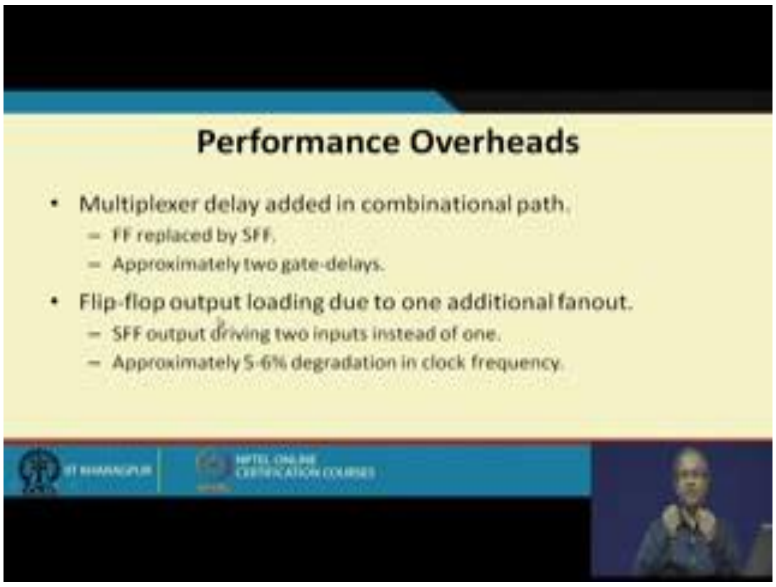
  n_s = number of *flip-flop*

  - Example:
    - n_g = 100k gates, n_s = 2k *flip-flops*, overhead = 6.7%.
  - More accurate estimate must consider scan wiring and layout area.

So, talking about the scan overheads, so as I said at least one additional pin for the test control is necessary. Now regarding this scanin and scanout, we can share them with the P I and P out pins, because if you had seen that diagram you can see that we never use scanin and P I together or we never use P O and scanout together. Only in the test mode I use scanin and scanout we do not use P I and P O. So, we can share those pins for scanin and scanout, but at least one pin necessary for TC.

Gate overhead will be this n g is the total number of gates in combinational logic, plus if n s is the number of flip flops. So, originally there were ten gates in every flip flop, now after you added the multiplexer there were 4 additional gates. So, 4 n S divide by this, this will be the percentage gate overhead. So, just a realistic figure if your number of gates is about a lakh in 100 k, number of flip flops is about 2000, 2 k then your overhead will come to about 6.7 percent.

(Refer Slide Time: 28:39)



Performance Overheads

- Multiplexer delay added in combinational path.
  - FF replaced by SFF.
  - Approximately two gate-delays.
- Flip-flop output loading due to one additional fanout.
  - SFF output driving two inputs instead of one.
  - Approximately 5-6% degradation in clock frequency.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

But of course, we have ignored the overheads due to interconnections and other things that will also need to be incorporated here. And regarding the performance we have added a multiplexer in the combination path. So, that will be approximately 2 gate delays, so the circuit will be slowed down. Similarly in the flip flop earlier the output of a flip flop was coming straight to the input of a of the combinational circuit, but now the output of every scan flip flop will be coming not only to the input of the circuit, but also to the input of the next scan flip flop. So, the fanout becomes 2. So, increasing fanout means more delay. So, the clock will also get degraded. So, typically 5 6 percent degradation this occurs.

So, in this lecture we have seen that we are using some additional hardware and also additional time, significantly additional time to address the problem of testing sequential circuits. But you see these additional time and performance degradation is acceptable or you see must be acceptable, because as otherwise it would be impossible to test these circuits of the type that we manufactured today. So, designs for testabilities are very important technique which is part and partial of chip design today.

Thank you.