

Advanced VLSI Design
Prof. Virendra K. Singh
Department of Electrical Engineering
Indian Institute of Technology- Bombay

Lecture - 37
VLSI Testing: Design for Test (DFT)

Welcome to the lecture series on advanced VLSI design. I am covering portion on VLSI testing and today will cover design for test techniques. In last lecture, we discussed about various difficulties in sequential test generation and we discussed that you need to unroll your circuit multiple timeframes to generate a test sequence.

(Refer Slide Time: 01:00)

Difficulties in Seq. ATPG					
<ul style="list-style-type: none">❖ Poor initializability.❖ Poor controllability/observability of state variables.❖ Gate count, number of flip-flops, and sequential depth do not explain the problem.❖ Cycles are mainly responsible for complexity.❖ An ATPG experiment:					
Circuit	Number of gates	Number of flip-flops	Sequential depth	ATPG CPU s	Fault coverage
TLC	355	21	14*	1,247	89.01%
Chip A	1,112	39	14	269	98.80%

* Maximum number of flip-flops on a PI to PO path

Advance VLSI Design 2

And so now if you look at the source of these difficulties, these difficulties are arising due to poor initializability of the circuit. So that means here it takes couple of cycles to initialize all the flip-flops to some specific state. It has poor controllability and observability of the state variable. So that means here you cannot directly assign any value to any of the flip-flops or you cannot read directly it again take couple of cycles to take out the value from the state variable from the output.

And then as the design is growing, the number of gates are increasing, number of flip-flops are increasing hence the sequential depth of the circuit is increasing and hence you need to unroll it multiple timeframes and your test generation would be more and more complex and if you look at the complexity of the test generation problem in that case, you will find that most of time cycles are main responsible to unroll circuit multiple timeframes.

Hence your circuit would be much more complex than your combination circuit and hence it would be difficult to generate test factors.

(Refer Slide Time: 02:09)

Benchmark Circuits				
Circuit	s1196	s1238	s1488	s1494
PI	14	14	8	8
PO	14	14	19	19
FF	18	18	6	6
Gates	529	508	653	647
Structure	Cycle-free	Cycle-free	Cyclic	Cyclic
Sequential depth	4	4	--	--
Total faults	1242	1355	1486	1506
Detected faults	1239	1283	1384	1379
Potentially detected faults	0	0	2	2
Untestable faults	3	72	26	30
Abandoned faults	0	0	76	97
Fault coverage (%)	99.8	94.7	93.1	91.6
Fault efficiency (%)	100.0	100.0	94.8	93.4
Max. sequence length	3	3	24	28
Total test vectors	313	308	525	559
GenTest CPU s (Sparc 2)	10	15	19941	19163

If you look at that the how difficult these things are. Like here these are the four different ISCAS89 circuits, so 1196 has 14 primary inputs, 14 primary output and 18 flip-flops and about 529 gates. As 1494 is almost similar kind of circuit, which has 8 primary inputs, 19 primary outputs and only 6 flip-flops and about 600 gates, but this is cyclic circuit, this is acyclic circuit.

If you look at the test generation, now you can achieve 100% fault efficiency for this acyclic circuit and fault coverage is 99.8 so that means your remaining faults are redundant faults or identified as redundant faults and it generates 313 test vectors 10 seconds whereas this circuit has lesser number of flip-flops, but this is cyclic and so it generates about 559 test vectors and the test generation time is something about 20,000 seconds and still you are not able to achieve 100% fault efficiency, here it is 93.

So now you can see the huge difference almost three order of magnitude difference between the test generation time for acyclic sequential circuit and cyclic sequential circuit. This can give you the fair understanding that if your circuit is cyclic then test generation time would be huge and this is for the circuit, which has only 6 flip-flops. In practice, we have millions of flip-flops.

You can imagine how much time it will take to generate a generate test. Then here now the question is if it grows like this what is the alternate? This may or may not be practical if we go for millions of flip-flops then what to do with this?

(Refer Slide Time: 04:43)

Difficulties in Seq. ATPG

- ❖ Poor initializability.
- ❖ Poor controllability/observability of state variables.
- ❖ Gate count, number of flip-flops, and sequential depth do not explain the problem.
- ❖ Cycles are mainly responsible for complexity.
- ❖ An ATPG experiment:

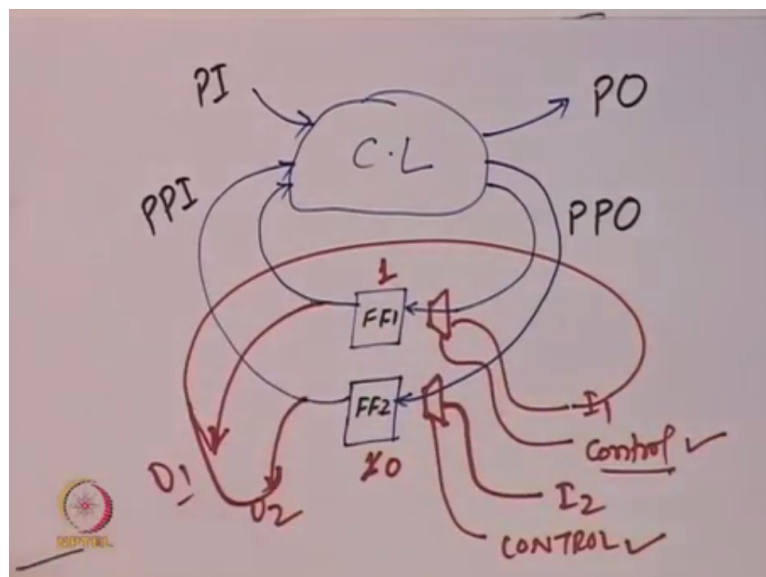
Circuit	Number of gates	Number of flip-flops	Sequential depth	ATPG CPU s	Fault coverage
TLC	355	21	14*	1,247	89.01%
Chip A	1,112	39	14	269	98.80%

* Maximum number of flip-flops on a PI to PO path

Advance VLSI Design 2

So now here look at the problems, problems are due to the poor controllability and poor observability. So that means here if we can enhance the controllability and observability of the circuit, we can make it better test evolve and hence we can reduce the test generation time and this is coming though.

(Refer Slide Time: 05:03)



Now here if you look at a sequential circuit, if you can look at a sequential circuit, it has a combinational logic, a couple of flip-flops. So now it has some primary inputs, some primary outputs and then here the input, which is coming from the flip-flop is known as pseudo

primary input and input which is going to the flip-flop is referred as pseudo primary outputs. Now if I say that if I can assign any random value in these flip-flops, in that case my test generation complexity would be almost similar to this test generation complexity of a combinational logic.

So if I can assign any value and if I can read any value from here again here so now when you can write any value here, you can read any value from these flip-flops at any point in time in that case here test generation is as simple as test generation for combinational logic. So the question is how I can do that? Is there any way? So one of the ways that we can use would be say if I place one multiplexer here and one extra primary input, so this primary input say I1.

So now here whenever I want to load anything here I can control this multiplexer and now using this control I can assign any value here. So this gives me enhanced controllability. I can control any value here in flip-flop. So that means here I need two additional inputs, but now here the controllability is not enough, I have to enhance the observability as well right. So now here how I can enhance the observability?

So in order to enhance the observability what I need to do is I have to take the output to the primary output. So now here I need to have additional primary output. This is for one flip-flop. Now here for another flip-flop I need to have another multiplexer and then one more test control. So this is I2 and then again I have to observe this O2. So now here what I need is I need three additional primary input output per flip-flop.

Now the question means is it a practical thing? So if you have say million flip-flops you need to have 3 million additional primary input output that is impractical. So now here what is the solution? But we want this kind of functionalities that we can load any flip-flop by any random value at any point in time and then we can observe any value, which is stored in the flip-flop. So now what is the solution?

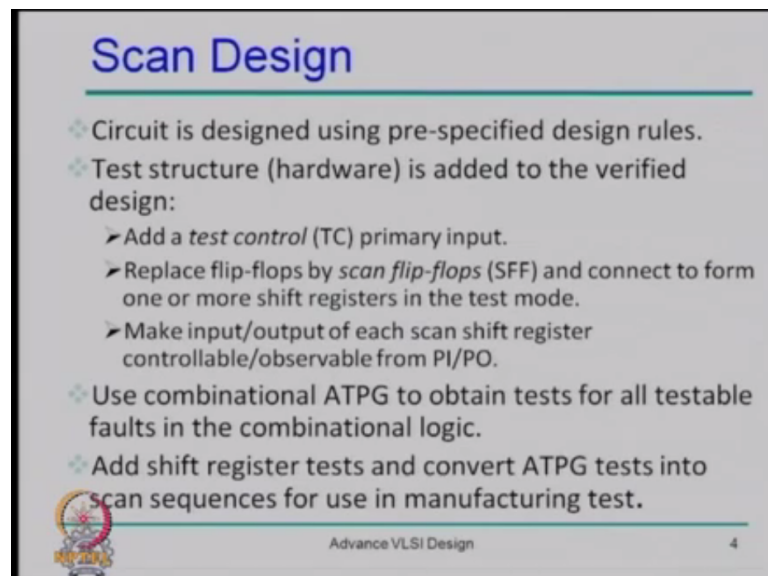
One thing that we can immediately look at is that here when I want to load in one flip-flop I can simultaneously load second flip-flop hence your test control input here we can share among the flip-flop. So that means your same test control can be used to control all these multiplexers. So now here I reduce this n number of control signals into single signal.

But now here still I need at least $2n$, where n is the number of flip-flop in the design, $2n$, number of primary input outputs. How I can reduce that number to a reasonable number? and one of the simplest approach that we can make use is that we can connect the output of one flip-flop as input to the another flip-flop. In other word what I can say is that here we can change these flip-flops like this.

This is here from here not from here. So now here I can change this so that means here what it will give me, it will make a shift register like flip-flop 2 and then flip-flop 1. So now if I want to load say value 10 in that case here in the first cycle I load value 1 that would be stored in flip-flop 2 and then in the next cycle here that value would be transferred here and then the another value I can load here in is 1 so now I can load any arbitrary value 10.

So now this is a shift register so whatever value it has that can be scanned out. This approach was first proposed by people from NEC in early 60s and that was published only in Japanese literature. So it was unknown to rest of the world until late 60s or 70s when IBM came up with the alternate scan architecture and this design is known as scan design. So now here this gives you flexibility or a way to load any arbitrary value in all the flip-flops.

(Refer Slide Time: 11:06)



Scan Design

- ❖ Circuit is designed using pre-specified design rules.
- ❖ Test structure (hardware) is added to the verified design:
 - Add a test control (TC) primary input.
 - Replace flip-flops by *scan flip-flops* (SFF) and connect to form one or more shift registers in the test mode.
 - Make input/output of each scan shift register controllable/observable from PI/PO.
- ❖ Use combinational ATPG to obtain tests for all testable faults in the combinational logic.
- ❖ Add shift register tests and convert ATPG tests into scan sequences for use in manufacturing test.

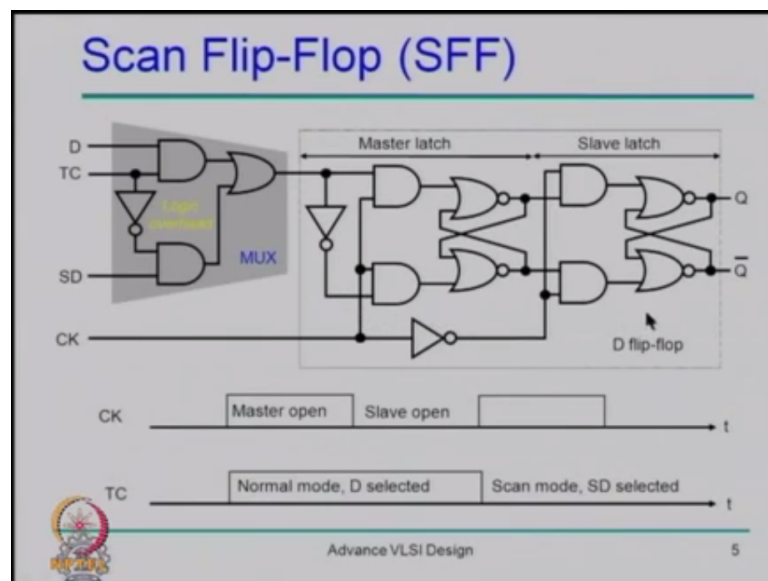
Advance VLSI Design 4

So now here I can load any arbitrary value, I can read any value from these flip-flops, but here what I need, I need at least one additional test control input that is the input that can control all these multiplexers and now here you have to replace all these flip-flops by scan

flip-flop, scan flip-flop I mean a flip-flop with a multiplexer and now here we make the input output as a scan shift register.

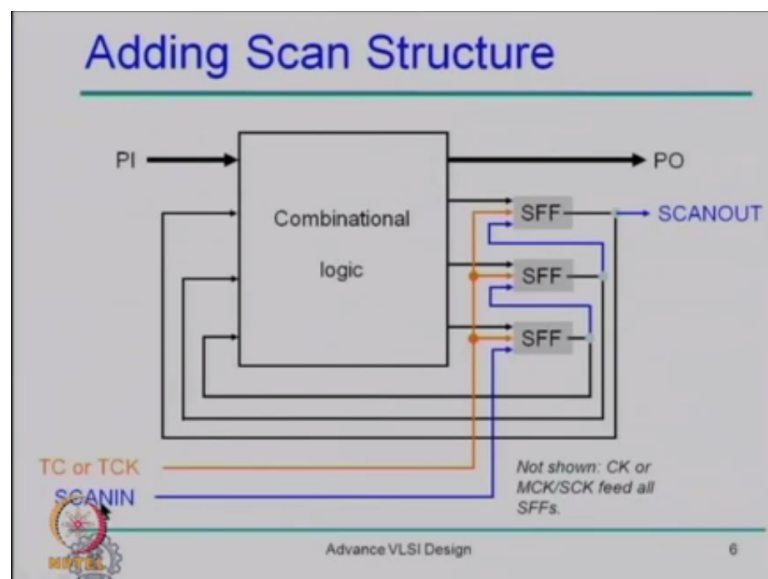
So now when you can arbitrarily load any value and read any value from the flip-flop now here when you need to generate a test vector, you need to consider only combinational logic not a sequential logic. Now the test generation problem or in other words the sequential test generation problem is reduced to the combinational test generation problem.

(Refer Slide Time: 12:03)



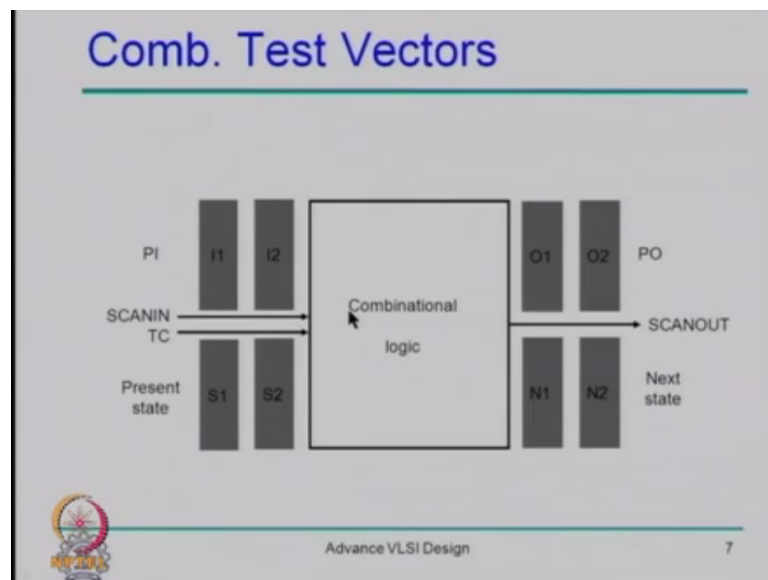
So this is the design of a scan flip-flop wherein we use a master slave D flip-flop and a multiplexer before that. So here you have the test control input, scan data input, this D input from the circuit and then this is your clock.

(Refer Slide Time: 12:27)



So now here as I said that we need three additional signals, one is the SCANIN, one is the SCANOUT and another is that the test control or test clock whatever we say. So now here this signal controls all the multiplexers associated with these flip-flops and then the SCANIN and SCANOUT, provide you the shift path. So now you can load any value and when you generate the test you need to generate test for the combinational logic.

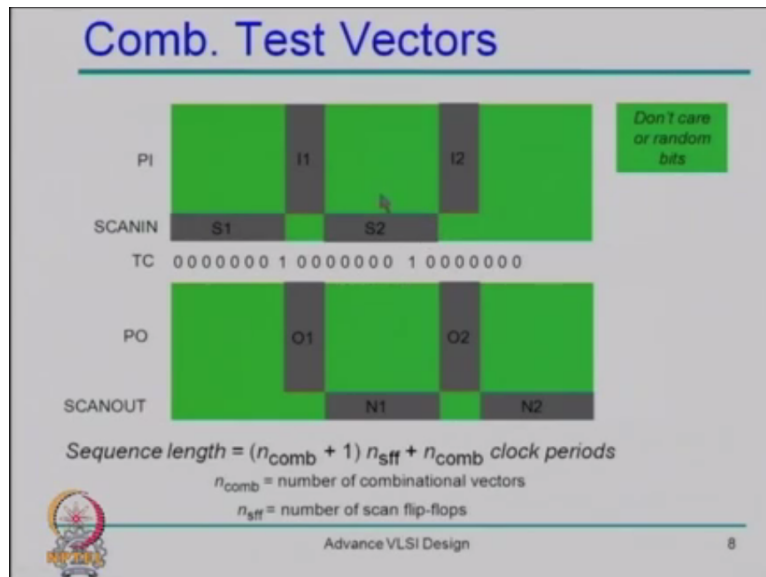
(Refer Slide Time: 13:03)



So now here how you apply test? First look at the simple combinational logic, how we apply test to a combinational logic? You have a combinational logic, you apply some input and then here you need to apply the value from the state variable. So that means your pseudo primary input and then primary input and then every cycle you are getting output then your output may be at the primary output or that may go to the pseudo primary output and you are reading that.

So this way we apply right. If it happens to be a plain combinational logic, but now it is not a plain combinational logic, it has memory elements or pseudo primary inputs means loading value to pseudo primary inputs take some time right because it is a shift register.

(Refer Slide Time: 14:00)



So now here in place of having this vertical S2, S1 now you need to make it horizontal so that that means here in couple of cycles you need to SCANIN the test vector in the flip-flops and then in one cycle you apply input from the primary input, capture the rest ones and then here you read the test response directly from the primary output of the circuit and then the response is stored in the flip-flops would be shifted out while you are shifting in the next vector.

So now here you have to shift out the response at the same time you can shift in the next test vector. So this way you can apply the test. So now here the question is how many cycles it will take?

(Refer Slide Time: 15:01)

$$(n+1) + n$$

$$n+1+n$$

$$k$$

$$(n+1)k + n$$

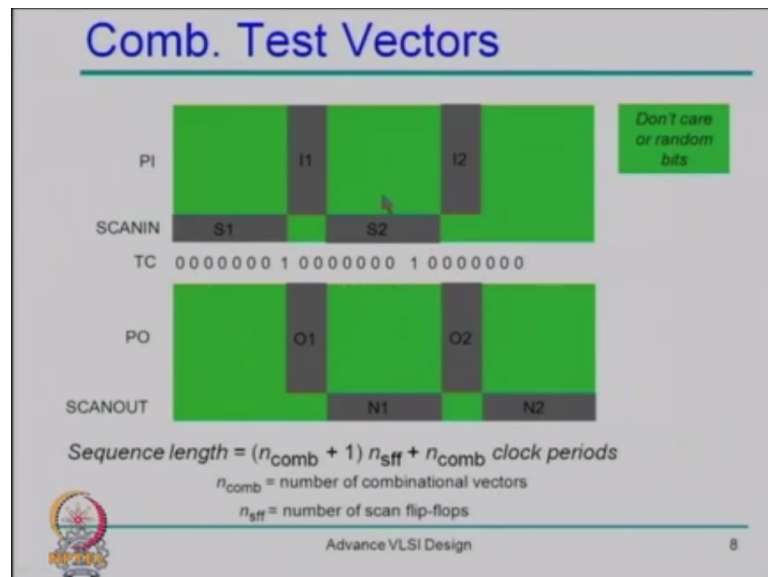
$$= (k+1)n + k$$

So if again I look at this problem say here if your scan chain length is n , so you need n cycle to load test vectors then in one cycle you have to go in the functional mode that means you have to change your test control signal from say 1 to 0 or 0 to 1 and then you capture the test response and you SCANOUT the test response. This is for the first vector.

For the second vector, you have to again load the test vector, go to the functional mode, apply primary input and capture the test response and then the captured test response from the three flops you have to SCANOUT. This way you keep on doing. Now because this is shift register in that case here you can do this operation in parallel so when you are scanning out the test response you have to SCANIN the test vector for the next test vector okay.

So now here if there are k vectors you want to SCANIN SCANOUT in that case here how many cycles it will take? It will take means for all vectors it will take $n+1$ cycle, so $n+1$ cycle multiplied by k and then here after application of the last vector you have to SCANOUT the response of the last vector, so this would be n read, so now here total test time you need is $k+1$ into n where n is the number of flip-flops and k is the number of test vectors.

(Refer Slide Time: 16:45)

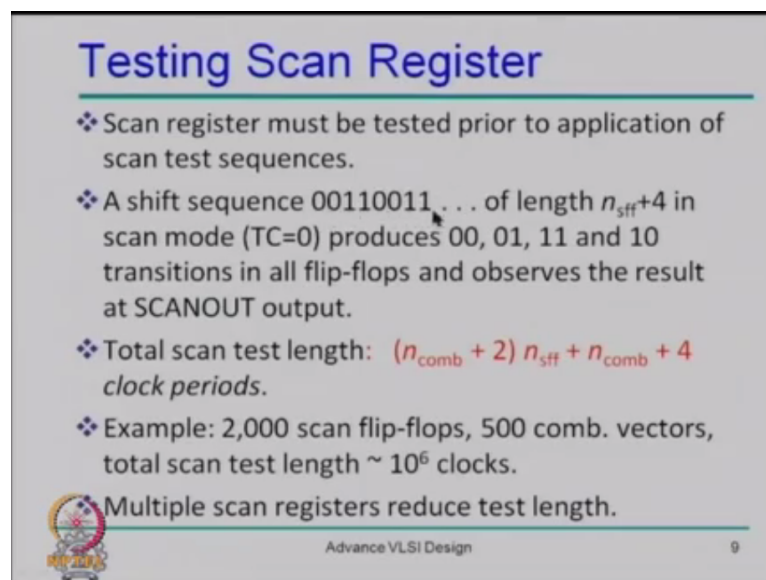


So now here if you look in this expression in that case here, this is n combinational logic so that means the number of test vectors + 1 into number of flip-flops + number of test vectors. These many cycles it will take, so it will elongate the test time. The elongation of test time directly results into increase in the test cost, but it greatly reduces the test generation effort.

And that is why though this scan design was proposed in way back in 60s till date it is surviving. So almost all the industries are using the full scan design wherein all the flip-flops are converted into scan flip-flop and then you apply test by loading that the test pattern in sequential fashion. So now here the other thing that we have to take into account, this way if you generate test for the combinational logic you can detect all the faults in the combinational logic.

But what if the flip-flop itself is a faulty circuit? So if it is itself is a faulty circuit, you have to test that as well. Now since flip-flop is a memory element it does a limited function hence in its place of having fault model like stuck-at kind of fault model, we can use the functional fault model for this and functional fault model is like it should either store 0 or store 1, so that means here we have to check whether it stores 0 or stores 1 or whether it provide you transition from 0 to 1 or 1 to 0.

(Refer Slide Time: 18:35)



Testing Scan Register

- ❖ Scan register must be tested prior to application of scan test sequences.
- ❖ A shift sequence 00110011 . . . of length $n_{sff}+4$ in scan mode (TC=0) produces 00, 01, 11 and 10 transitions in all flip-flops and observes the result at SCANOUT output.
- ❖ Total scan test length: $(n_{comb} + 2) n_{sff} + n_{comb} + 4$ clock periods.
- ❖ Example: 2,000 scan flip-flops, 500 comb. vectors, total scan test length $\sim 10^6$ clocks.

Multiple scan registers reduce test length.

Advance VLSI Design 9

So if I scan a sequence of 00110011 all through all the flip-flops in that case here I can say that here means if I get out the same sequence from the scan chain I can say that it stores 0, it stores 1, and it also allows transition from 0 to 1 and 1 to 0. Otherwise if assume 1 flip-flop cannot store 1, that stores permanently 0 in that case after a while all the bit will be 1 because here the entire bit stream is going through that particular flip-flop and after a while you will get all bit as 0 so that means here that scan chain itself is faulty.

So in order to make sure you have to scan through a bit stream of 0011 and it will take the number of flip-flop + 4 number of cycles. So first you have to scan through a sequence of

0011 that is also known as the sanity check of scan chain, once it is done you can apply your test vector. So now here the number of scan chain + 4 number of test vector you are applying from here and then as we discussed earlier that in order to apply the test vector you need these many cycles.

So now here the total number of cycles it needs is the total number of test vectors + 2 into multiplied by the number of flip-flops + test vector + 4. So these are the total number of clock periods you need. So like here for example if you have a scan chain of 2000 flip-flops and you want to apply 500 combinational vectors, if it happens to be only combinational logic you need 500 cycles.

If it is a sequential circuit and you generate test using the sequential logic and assume that say every fault may need a sequence of 4 vectors maybe you may need 2000 vectors and then it may need 2000 cycles whereas if it is a scan design in that case, you can put this value 2000 and 500 in this formula you will get roughly about a million cycle. So you can see the impact in terms of test time.

So this will increase the test time, but it helps you greatly in reducing the test generation effort. Otherwise test generation may not be possible for a fairly large industrial design or if it is possible it may take enormously long time that may not be affordable. Now the question is so if you look at the various overheads then it comes with following overheads like you need a multiplexer with every flip-flop, so that result into additional area.

You need to have a test control signal routed all through the chip so that routing area overhead is also there. Then because you are placing a multiplexer in the functional path hence it will result into additional delay, hence your circuit may operate slow. So it has performance penalty. Then it should have three additional pins, one is SCANIN, SCANOUT and test control.

Now here because of the sequential loading of scan pattern it takes longer time. Now as I said that here for a small circuit, which has about 2000 flip-flops and 500 test vectors it take about a million cycle. Now what are the ways to reduce this time? Because this directly result into additional test cost. So one of the way that we can think of is you have very long scan chain, I can break that scan chain in multiple scan chains.

So like here say these 2000 flip-flops in one scan chain, I can chop off in say 10 scan chains and every scan chain will have now 200 flip-flop and I can load these 10 scan chains in parallel, but for that I need 10 additional primary inputs and 10 additional primary outputs, so that means here in all I may need 21 additional input output pins that we may not have. So then what is the solution?

One of the solution could be because when we are loading this scan pattern in the scan chain, we are not using primary input and primary output. So what we can do is we can multiplex these scan input and primary input and we can reduce this requirement. So that means here by multiplexing we really need one additional pin and that is your test control that can control all these multiplexers associated with the flip-flops.

So now here whenever you are in the test mode, it will not take input from the combinational logic or from the primary input and when it is in the functional mode, it will not shift the value from one flip-flop to another flip-flop. So now here that can solve our problem up to certain extent we can reduce. Then here the question is how many scan chain I can afford to and the answer of that is max number of inputs and number of outputs because here for every scan chain I need to have one input and one output.

So say your circuit may have 10 inputs and 8 output in that case here maximum I can have 8 parallel scan chains and that directly reduces test time by eight times okay.

(Refer Slide Time: 25:10)

ATPG Example: S5378		
	Original	Full-scan
Number of combinational gates	2,781	2,781
Number of non-scan flip-flops (10 gates each)	179	0
Number of scan flip-flops (14 gates each)	0	179
Gate overhead	0.0%	15.66%
Number of faults	4,603	4,603
PI/PO for ATPG	35/49	214/228
Fault coverage	70.0%	99.1%
Fault efficiency	70.9%	100.0%
CPU time on SUN Ultra II, 200MHz processor	5,533 s	5 s
Number of ATPG vectors	414	585
Scan sequence length	414	105,662

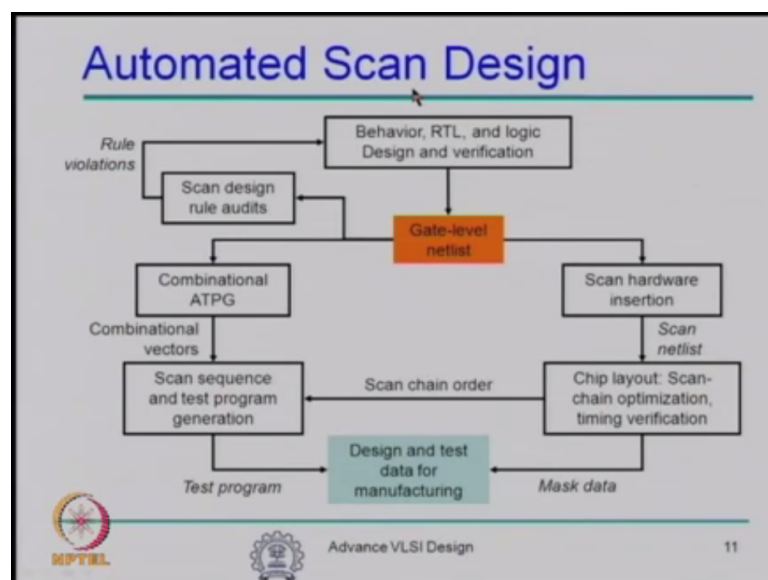


So if you look at the effort in that case here say you have circuit, which has number of gates about 3000 and number of flip-flops about 179 and so if you use the sequential test generation in that case you may achieve something like 70% fault efficiency in about 5500 seconds whereas if you cannot or flip-flop into scan flip-flop in that case here you need to generate test for combinational logic and it can achieve 100% fault efficiency in 5 seconds.

But now if you look at the test application, here if you generate these test vectors by using the sequential ATPG in that case you may generate say 414 test vectors and in order to apply this you need 414 cycles only, whereas here you generate 585 test vectors and in order to apply that you need about 1 lakh cycles and so now you can see the overhead in terms of time and this is recurring cost.

But as I mentioned you earlier that this can greatly help you the test generation here again we cannot directly compare this time this is almost 3 order of magnitude higher than this time, but if you look at the fault efficiency here we achieved 100% efficiency here, the fault efficiency is 70% only. If you go to 100% in that case here this time will exponentially grow and then this may go unreasonably large number or impractical.

(Refer Slide Time: 26:59)

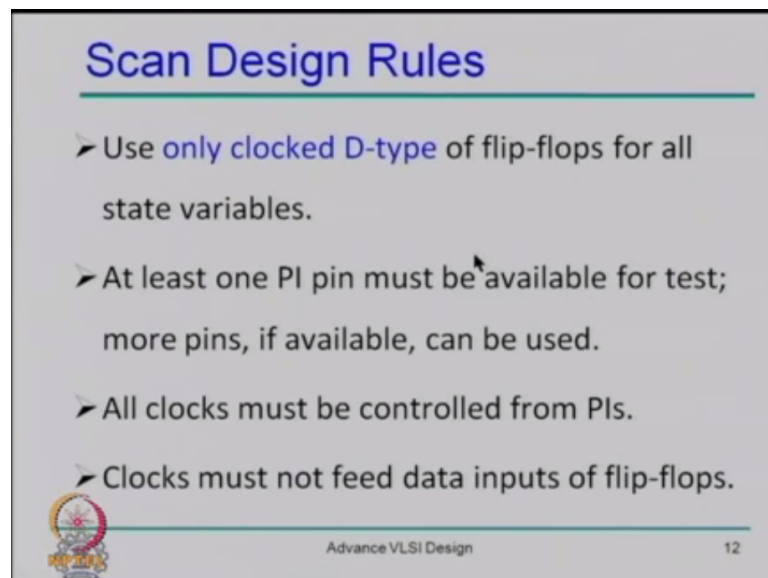


So now in general, the scan design flow is more or less automated today so when you start your design flow from RTL you synthesize your circuit to gate level netlist, once you have gate level netlist in that case here there are two things you have to do, one thing is because you have gate level netlist you have combinational logic in this, you start to generate the test vector from the combinational logic.

At the same time here you can insert the scan chain so insert a scan chain means here you have to convert all flip-flop into scan flip-flop and then you have to shift these scan flip-flop in a shift register. So now here once you have shift register in that case here you know the layout of the chip and you know that how this flip-flops are connected. Based on that, here you have to generate the test sequence.

So the test sequence is generated and now here the mask and the test program you have to send to the fab wherein they applied at. So now when you have the gate level netlist, it has to follow certain rule in order to insert the scan cell in the circuit.

(Refer Slide Time: 28:26)



Scan Design Rules

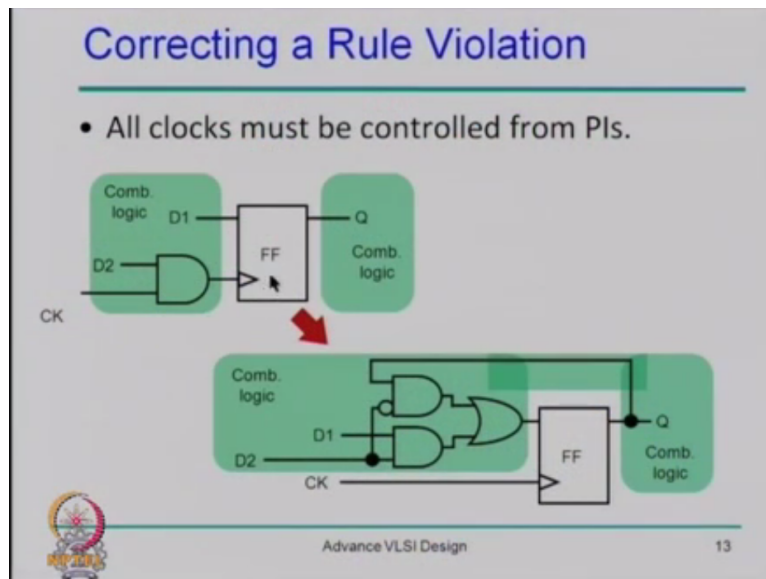
- Use **only clocked D-type** of flip-flops for all state variables.
- At least one PI pin must be available for test; more pins, if available, can be used.
- All clocks must be controlled from PIs.
- Clocks must not feed data inputs of flip-flops.

Advance VLSI Design 12

So what are those rules which this need to follow? Some of the rules are like here we have to use only clocked D-type flip-flops. We discouraged to use any other type of the flip-flop like SR flip-flop, JK flip-flop, T flip-flop. So now practically all the circuits are built or designed using clocked D-type flip-flop. At least you should have one primary input pin available for the test that is that the test control pin, which is used to control the multiplexers associated with these D flip-flops.

And third thing is that all clocks must be controlled from the primary input so that means here the clocks should not be gated. Clocks are controlled by primary inputs only and clocks must not feed data input to the flip-flops. So these are some of the design rules you have to follow if you want to insert scan. So that means here once it is followed, you can directly replace the flip-flop by scan flip-flop that means a flip-flop with a multiplexer.

(Refer Slide Time: 29:45)

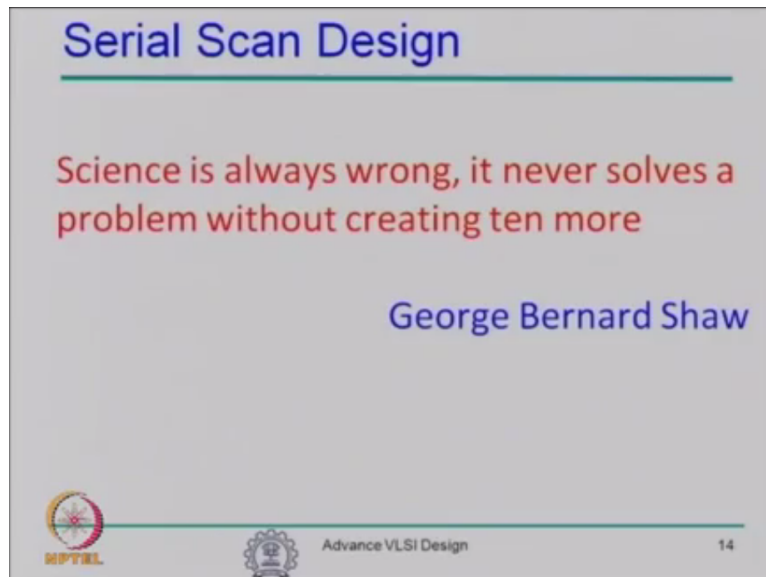


So like here for example if this is the circuit wherein you have one flip-flop and then the clock is gated and sometimes it is very easy to design this kind of gated clock circuit. So now here clock will depend on the clock and D input so that means here whenever D input is high and clock is high, it allows the value to change in this flip-flop. Now when I do not want this or we do not allow gated clock what we need to do?

We have to convert this gated clock input into non-gated clock input, so now here the same functionality we can achieve by this circuit. So you have to convert if your circuit is designed by like this you have to transform your circuit or convert your circuit into this design wherein the clock is clean that is controlled by primary input that is not controlled by the data input. Sometimes though here means it is discouraged to use the gated clock.

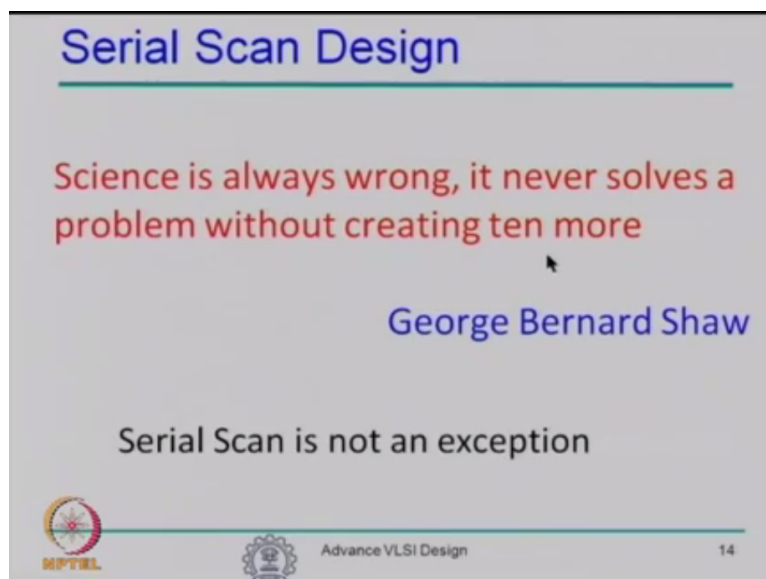
But sometimes in order to optimize your circuit you feel it is better to do the clock getting rather than optimizing using any other way. So one must be careful means while you are designing if you want to go for a scan design. Now here as I said that the scan design comes with some of the overhead those are like here additional area, additional performance penalty, then additional pins, but it reduces test generation time and test generation effort significantly.

(Refer Slide Time: 32:04)



So this was wonderful technique, which was ever proposed in VLSI test and that is why it is surviving for last five decades, but there is a famous quote by George Bernard Shaw, which says that science is always wrong, it never solves a problem without creating ten more.

(Refer Slide Time: 32:10)



And scan design is not an exception to this. This solves a very important problem of sequential test generation, but it creates some additional problems and then we have to solve those problems. In some sense, it gives you some additional problems to solve. So now here what are those additional problems we may have?

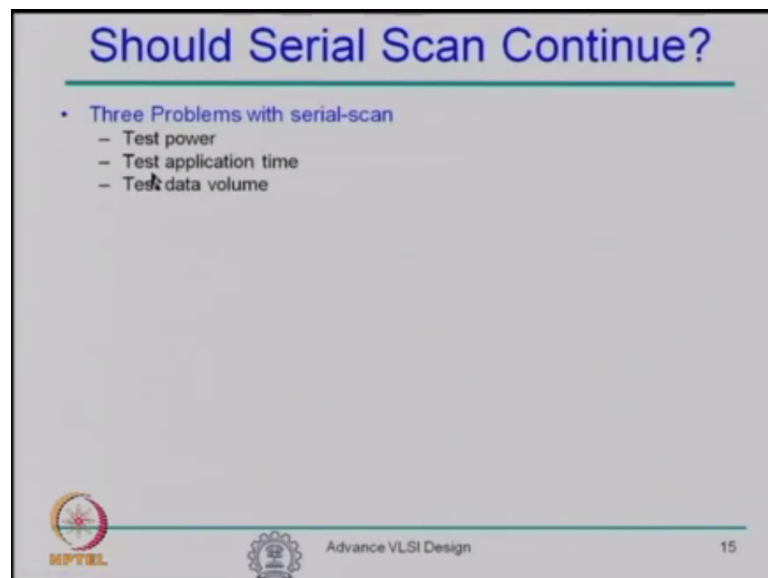
So one of the problems that I mentioned you earlier is that here your test application time will significantly increase and it will directly translate to the test cost. Another problem is test data volume. What does this mean? Like here if you apply sequential test in that case here you

need to store only a few vectors and then you have to apply those from the primary input and number of primary inputs are pretty small.

So say you have 10 inputs and it means there are 100 test vectors you want to apply, so in that case here you may need to store only 100 bits. Now assume that you have 10 inputs and you have 1000 scan cells and you may want to apply 100 inputs, so now here how many bits you want to store? You may need to or you must store $1000 + 10$ input per vector. So that means here it is the number of bits that you may need to store would be 100 multiplied by $1000 + 10$.

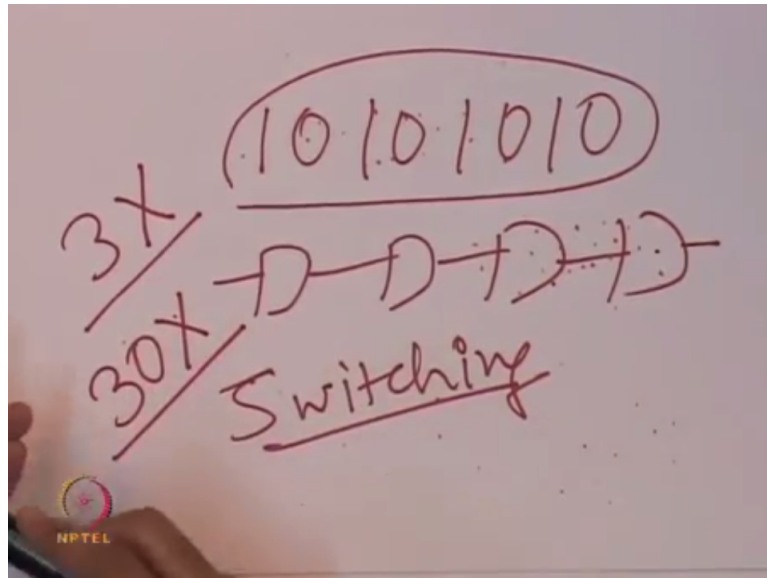
And so you may need to store now here almost 1000 times additional test vectors that needs the memory on the tester. In the same way, you also need the golden response from the primary output as well as the values in the flip-flops. So now here your test data as well as golden response will also increase several order of magnitude and assume you have a circuit with millions of flip-flops and several millions of test vector.

(Refer Slide Time: 35:10)



You need enormous used memory so that means the storage of the test data is also a problem. So that is additional problem that we have. The very important problem nowadays is the test power, which was not there earlier. Though here the test data volume was also not very big problem initially because at the time the circuit complexity was not that much or the circuit was too small.

(Refer Slide Time: 35:32)



Now here what is the source of test power? So say easily you may generate a test vector wherein you may need to load a test data like here 101010 sequence because this sequence you are generating from the combinational logic. Now when you load this sequence in flip-flops what will happen because there are different adjacent bits, so in every cycle when these bits will shift every flip-flop will toggle.

So that means here large number of switching will happen in the flip-flops and these switching will also reflect in the combinational logic and see this happens only in the test mode because here in functional mode you may not need to shift this test vector, but because here in the test mode I want to load these bits in the sequence cells shift fashion here it creates switching in old flip-flops in all the cycles.

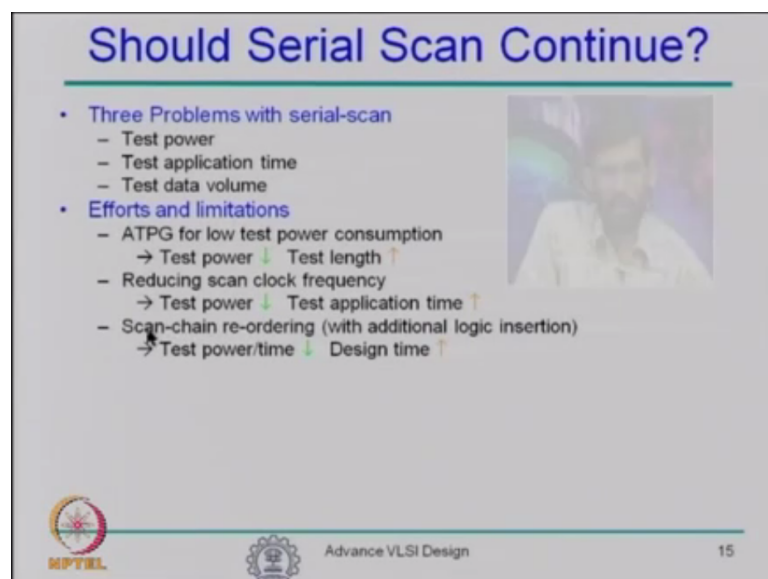
And that switching also reflects to the combinational logic, combinational logic will also switch and now here the total power that would be dissipated during this period may go very high and practical observation says that in some of the cases peak the average power can go three to four times than the normal power and peak power may go 30 times than the normal power.

What are the implications of this? If average power exceeds beyond the limit then your chip may burn out, so that means here you may damage a good chip and that will result into yield loss because of your bad test methodology. If peak power exceeds beyond certain limit, your circuit will start to drive large current all of sudden from your power grade. Power grade may not support that and hence there would be a drop in VDD.

If there is a drop in VDD then transistor start to switch slower, hence your response may not be able to propagate or the effect of application of one test vector may not be able to propagate to the output within the given clock period and hence here you may classify your good chip as bad chip because you will receive a random response out. So again here it will lead to the yield loss.

So these are the important problems those appeared in or past 2000 so now here there must be some solutions to these problems if you want to continue with the serial scan. These problems appear due to serial scan shift nature of the test vector. So now here what are the solutions? I may describe a couple of solutions in brief. You can look at literature there is large body of literature available targeting these problems.

(Refer Slide Time: 39:28)



Should Serial Scan Continue?

- Three Problems with serial-scan
 - Test power
 - Test application time
 - Test data volume
- Efforts and limitations
 - ATPG for low test power consumption
 - Test power ↓ Test length ↑
 - Reducing scan clock frequency
 - Test power ↓ Test application time ↑
 - Scan-chain re-ordering (with additional logic insertion)
 - Test power/time ↓ Design time ↑

The slide includes a small portrait of a man in the top right corner. At the bottom, there are logos for MPTEL and a university emblem, along with the text 'Advance VLSI Design' and the page number '15'.

So like here some of the efforts, which are being made like one of the effort is while you are generating the test you can generate test because as we discussed earlier that one fault may have multiple tests. So you can select a test, which can dissipate less power and when you are generating test, it generates large number of access so that means here the propagation of fault effect is not impacted by those primary inputs or pseudo primary inputs.

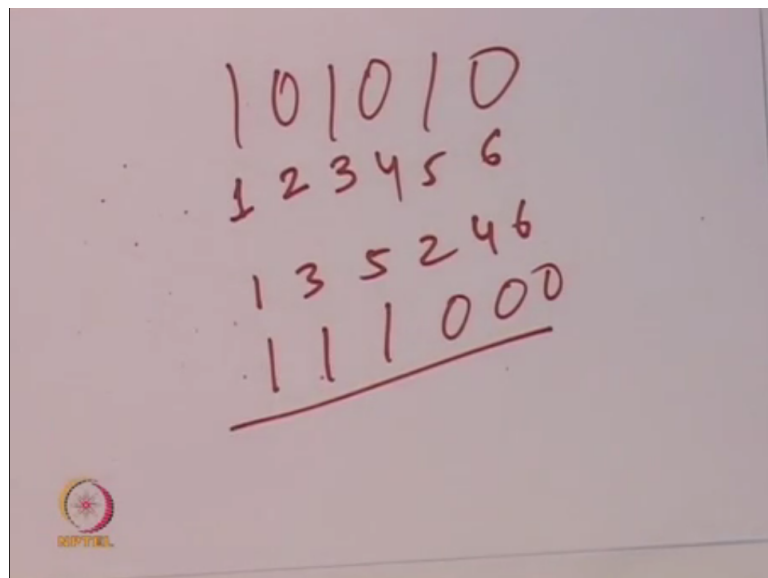
So now you can feel these access by any value and typically people use three approaches, one is all access are filled by 0s or all access are filled by 1s or all access are filled by looking at the adjacent specified bit. If it is 0 in that case it should be filled by 0, if it is 1 in that case it

should be filled by 1 that is known as minimum transition. So you want to reduce the transition.

So there are couple of approaches. So now here if you want to select a test vector that can consume or dissipate lesser power that means it has to create lesser transitions or activity in the circuit. If it creates lesser activity in that case here that may not be able to detect multiple faults. Hence you may need more number of test vectors, hence this can solve your problem of test power, test power can be reduced, but here test length will increase.

Other possibilities when you are scanning in you can reduce the test clock, so that means here you can operate your circuit slower hence you can reduce the average power. In practice, scan shift operation happens almost five times slower than the rated clock frequency. So now here you can reduce the test power, but now when you reduce the clock frequency in that case your test application time will increase, hence your test cost will increase.

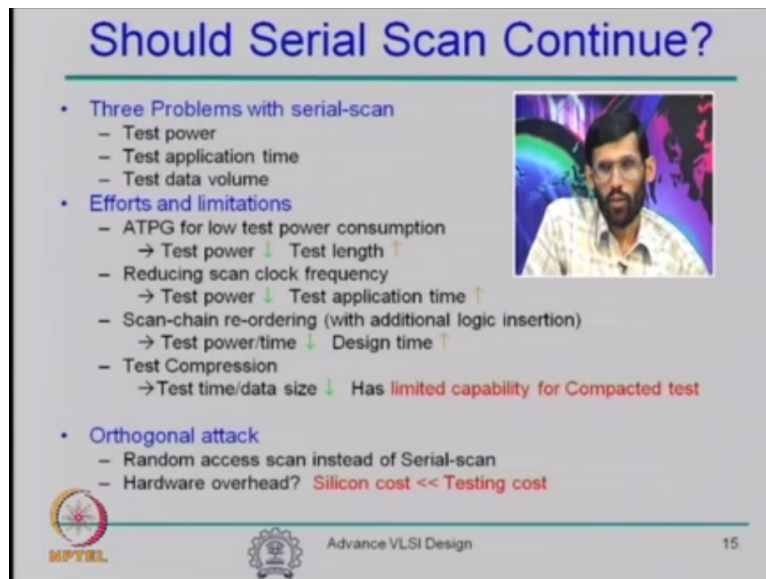
(Refer Slide Time: 42:16)



The other way is like here you can reorder these scan chain and reordering mean here say if you want to SCANIN the pattern 101010 and so now here you have 1 in flip-flop 1, 0 in flip-flop 2, 1 in flip-flop 3, 0 in flip-flop 4, 1 in 5, 0 in 6. If I can stitch this scan chain in different way like here 135246 in that case here how I need to SCANIN, I need to SCANIN 111000 and hence I can reduce the number of transitions because now in every cycle there would be only one transition here.


So this can reduce the test power and test time, but here now you need to modify the scan stitching and that needs some design effort and hence design time will increase.

(Refer Slide Time: 43:05)



Should Serial Scan Continue?

- Three Problems with serial-scan
 - Test power
 - Test application time
 - Test data volume
- Efforts and limitations
 - ATPG for low test power consumption
 - Test power ↓ Test length ↑
 - Reducing scan clock frequency
 - Test power ↓ Test application time ↑
 - Scan-chain re-ordering (with additional logic insertion)
 - Test power/time ↓ Design time ↑
 - Test Compression
 - Test time/data size ↓ Has **limited capability for Compacted test**
- Orthogonal attack
 - Random access scan instead of Serial-scan
 - Hardware overhead? **Silicon cost << Testing cost**

MPTEL  Advance VLSI Design 15

Then in order to reduce the test time or test data size here you can do the compression at the primary input and at the primary output. Compaction or compression is the widely used technique, but here again it has limited capability so these are some of the effort, which were proposed by people to take care of the additional problems, which are coming from the serial scan design.

There is a alternate approach that we can get away from the serial scan and then now in place of serial scan we can have random access scan like we have random access memory, so that means we can load or unload any flip-flop whenever we want. We do not need to serial shift everything that can also help you in targeting all these three additional problems. You can look at various papers published for random access scan.

Okay so here I complete the scan design portion. I discussed about why we need scan design and what are the advantages of scan design and what are the overheads it comes with, what benefit it gives you and what are the additional problems we are getting due to serial scan in current designs and what are various solutions we have. So in nutshell, I can say that here if we have serial scan design, we can convert every flip-flop into a scan flip-flop.

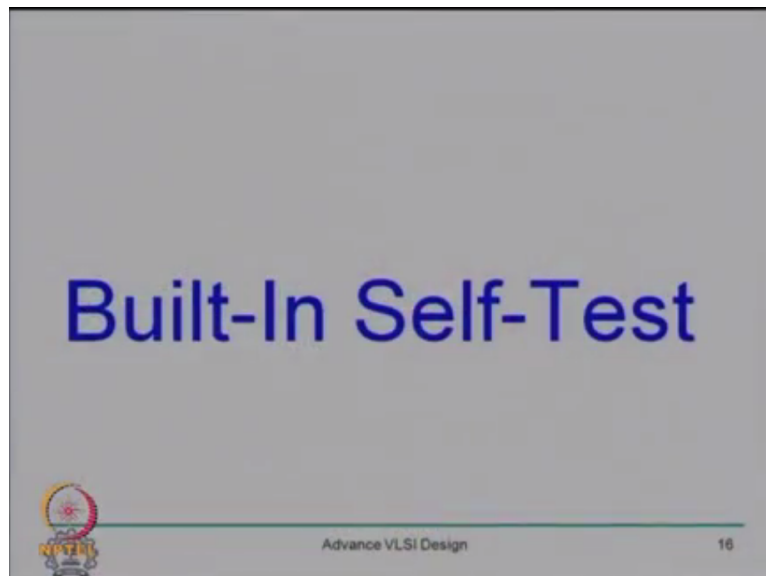
And then we can go to ATPG generate test using a combination of the ATPG and then we can apply the test, but you need an expensive tester to apply these test vectors and you can test

these shifts just after manufacturing or at the fab house or in the design house if you have expensive tester. Some of the devices are very safety critical devices and you would like to monitor the health of those devices while they are operating in the field as well. So that cannot be done if you are testing a chip using external very expensive test equipment.

If it happens to be a small equipment or very inexpensive equipment, you can still test it. So now if you want to build capability of field test you have to think about some other alternate that is one of the problems. Second as I said that here it has some cost implications in terms of like when you put your cheap on tester it cost you somewhere 5 to 10 cents per second. So it is expensive.

Then it was thought that can we build a test capability in chip itself, so that means is it possible that chip can test itself that can help you in testing your chip at speed so that means here you do not need to slow shift your test vector, apply the test in the functional mode and SCANOUT the rest ones.

(Refer Slide Time: 47:26)



That can also give you the flexibility to diagnose your chip and so that technique is known as built-in self-test. It may not sound well if you say that chip test itself so that means here I need to have all these kind of test resources on chip that means here you have to have a test generator on the chip, you have to have a test response collector on the chip, you have to have some mechanism that can check with the golden reference, the collected reference and collected response and then make a decision whether chip is good or bad.

(Refer Slide Time: 48:03)

BIST - Motivation

- ❖ Useful for field test and diagnosis (less expensive than a local automatic test equipment)
- ❖ Software tests for field test and diagnosis:
 - Low hardware fault coverage
 - Low diagnostic resolution
 - Slow to operate
- ❖ Hardware BIST benefits:
 - Lower system test effort
 - Improved system maintenance and repair
 - Improved component repair
 - Better diagnosis




But as I said that here field test is important for some of the devices like nowadays when you power on your laptop it generally do some kind of self-test for the memory, sometimes for the processor as well in order to monitor the health of your system whether it is good. Generally for that the diagnoses here we use the software test, but if you use the software test in that case here the hardware fault coverage is pretty low that maybe something like here 20%, 30% after application of large number of hardware test vector like here you boot the operating system or you run some application on that.

Diagnostic resolution is also very low, it is very slow process so that means here if you use the hardware built-in self-test in that case here you may have lower system test effort you can use the same built-in self-test after manufacturing and then in the field, it can improve the system maintenance and repair. It can improve the component repair and then it gives you the better diagnosability.

(Refer Slide Time: 49:20)

Costly Test Problems Alleviated by BIST

- ❖ Increasing chip logic-to-pin ratio – harder observability
- ❖ Increasingly dense devices and faster clocks
- ❖ Increasing test generation and application times
- ❖ Increasing size of test vectors stored in ATE
- ❖ Expensive ATE needed for over 1 GHz clocking chips
- ❖ Hard testability insertion – designers unfamiliar with gate-level logic, since they design at behavioral level
- ❖ Shortage of test engineers
- ❖ Circuit testing cannot be easily partitioned



Advance VLSI Design 18

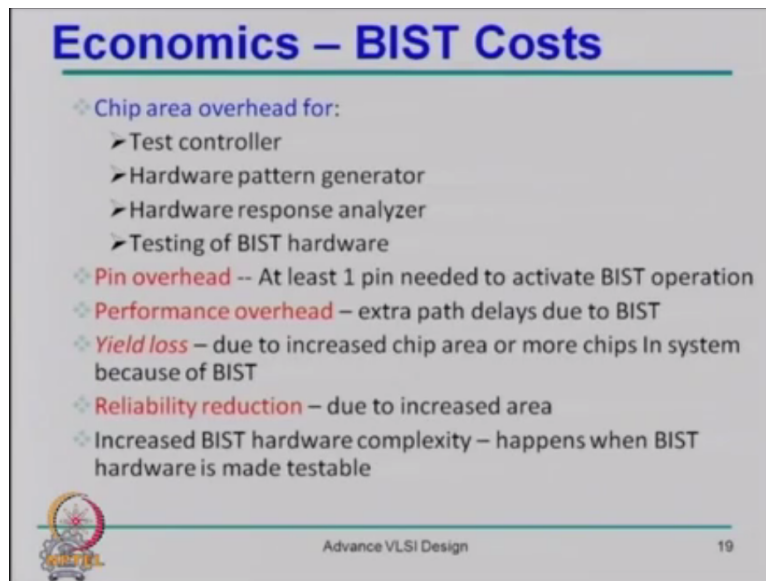
It also alleviate some of the test problems, which are coming from the expensive tester so some of the problems are like here in today's design the logic-to-pin ratio is very flash so that means here you have flash logic and then there are only few pins so the pins are not increasing as per more slow, but here logic is doubling almost every 18 months, so now here it is very hard to observe some internal points.

So observability is very difficult then the density of device is increasing and then here clock is becoming faster the test application time because the number of flip-flops are increasing then test application time is increasing, the number of test vectors are increasing and like here more slow the test vectors are doubling almost every 13 to 16 months and you need expensive test vector and then here it is very hard to insert some test points in the circuit because here this may impact the timing of the circuit.

Of course here industry faces the shortage of test engineer so that is why also we need built-in self-test kind of test mechanism. Other thing is like here for the design point of view, it is easy to partition the circuit and do design in parallel, but here it is very, very or extremely difficult to partition the circuit for the test point of view. So these are some of the problems, which are alleviated by built-in self-test.

So now here built-in self-test means a chip is supposed to test itself. So that means here you need to have a hardware pattern generator, you need to have a response analyzer or compactor that can compact, collect the response, compare with the golden response and say go now go.

(Refer Slide Time: 51:32)



Economics – BIST Costs

- ❖ Chip area overhead for:
 - Test controller
 - Hardware pattern generator
 - Hardware response analyzer
 - Testing of BIST hardware
- ❖ Pin overhead – At least 1 pin needed to activate BIST operation
- ❖ Performance overhead – extra path delays due to BIST
- ❖ Yield loss – due to increased chip area or more chips in system because of BIST
- ❖ Reliability reduction – due to increased area
- ❖ Increased BIST hardware complexity – happens when BIST hardware is made testable

Advance VLSI Design 19

And then here in order to control all these activities here you must have a test controller that can control that means that can excide the test better and generator and analyzer at different point in time and then here you need to have some methodology to test that built-in self-test hardware. It also comes with some pin overhead so that means here at least you must have one additional pin that can say you that here now it is the time for built-in self-test.



So now you have to run the test, so you need additional pin for that because here we are inserting additional hardware for the test pattern generator and analyzer and then test controller in that case here some of the additional gates may be being inserted in some critical path and due to that here performance may reduce because we are putting additional infrastructure in the circuit.

Then the area will increase if area increases in that case here the number of faults may likely to increase hence the yield may be low and then because of the additional area the reliability of the system also reduces and then here the built-in self-test hardware complexity will also increase when you want to make that additional circuit as a testable.

(Refer Slide Time: 53:06)

BIST Benefits

- ❖ **Faults tested:**
 - Single combinational / sequential stuck
 - Delay faults
 - Single stuck-at faults in BIST hardware
- ❖ **BIST benefits**
 - Reduced testing and maintenance cost
 - Lower test generation cost
 - Reduced storage / maintenance of test patterns
 - Simpler and less expensive ATE
 - Can test many units in parallel
 - Shorter test application times
 - Can test at functional system speed

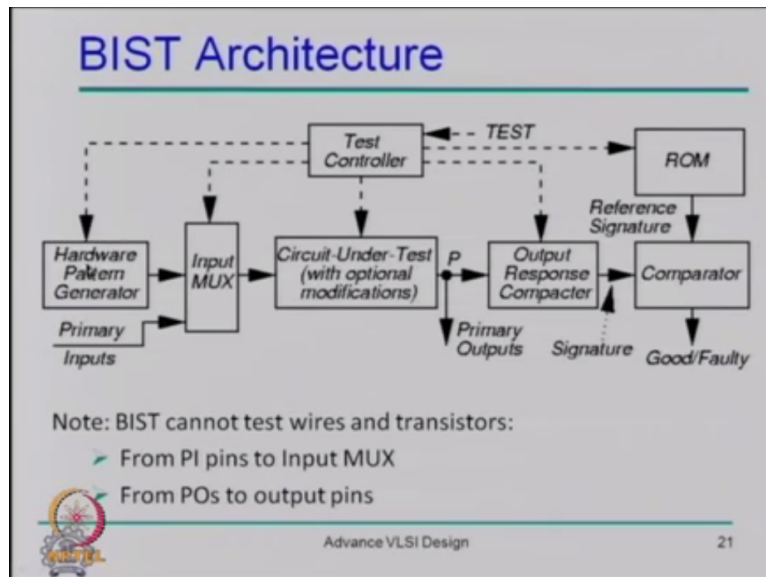


Advance VLSI Design 20

So now here if you look at what kind of faults your built-in self-test can detect? These are like here all single faults in the combinational or sequential circuit, all the delay faults, single stuck-at faults in the built-in self-test hardware or architecture. What are the benefits of this? It reduces the maintenance cost, it generate the test at lower cost, it reduces the storage or maintenance of test vectors, it is simpler and less expensive, automatic test equipment is needed.

Because here you need to say only that here now you start the test and now you stop the test that is it. It can test as many units as possible in parallel because now you need a very small board that can tell a chip that now you should start to test the chip and these are all very inexpensive board. So you can test large number of chips in parallel and so now here the test application time would be shorter and now here you can apply test at the circuit speed in that case, your at speed test is enable.

(Refer Slide Time: 54:26)



So if you look at the built-in self-test architecture what you need, you need a hardware pattern generator, you need hardware response analyzer, you need to have some place where you store or the golden reference input and then you need to have a comparator that can compare you or collected response with the golden response and it will say that chip is good or chip is bad.

Now here this is your circuit under test so now where you need to apply the test, you need to apply the test at the primary input right, so that means here you have to multiplex the output of hardware pattern generator with the primary inputs and you have to take output from the primary output of the circuit and in order to do all these things you need to have a test controller that can generate the control signal for all the test patterns.

So now here this approach can test all the faults, which are in the circuit under test, all the faults which are in the built-in self-test structure except the faults, which are present at the primary input of this multiplexer because here when you are testing your chip you are not exercising this path and primary output so now here if there is a open in that case here you may not be able to exercise that and you may not be able to test.

So these are the faults, which may remain uncovered if you use built-in self-test. So now describing this built-in self-test, I stop here today, we will continue with the various components of built-in self-test in the next lecture. Thank you very much for patience for listening. Good day.