

Design Verification and Test of Digital VLSI Designs
Prof. Dr. Santosh Biswas
Prof. Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 10
Sequential Circuit Testing and Scan Chains
Lecture - 1
ATPG for Synchronous Sequential Circuits

Welcome to the and to a new module, module number 10 in the lecture series on VLSI testing. So, what we have dealt, till now in the test series in the discussion on test, we regarding combinational circuit. So, we are started with fault models, then we have seen for random test pattern generation, then we have seen fault collapsing, coscup algorithms, sensitize propagate and justify based algorithm as d algorithm and so forth, but most of them were limited to combinational circuits.

So, in the, in the new module 10, we will menu look about how the same things or all the technology's like stack at fault models collapsing etcetera, ATPG algorithms etcetera, whatever, we have seen for combinational circuits, how the macro sequential circuits

So, this lecture is determine, today we learn on automatic test pattern generation for synchronous sequential circuit. That means, you should also know that sequential circuits can be synchronal as well as synchronal, which you have learnt in our basic digital days. But actually here will be mainly dealing with synchronal sequential circuits with a single clock, I mean to make the course a bit simple, because this course is mainly on VLSI design verification and test.

So, we are going to have a overview of all most all the key aspects of VLSI design. So, rather than going in to depth of a synchronal circuit testing and so forth, we whether thought that, we will try to give you an broad overview. So, to limit the depth of sequential circuits ATPG, we will limit our scope to single clock synchronal circuit. And most of this digital designs, we do these days are basically driven by single clocks and it is mainly sequential in nature.

So, I mean this actually covers the main part of VLSI testing for sequential circuits and a very few limited circuits, you can think of run a synchronal. So, the moral as you will not for covering that for this does not have a too much impact on our VLSI testing study. So, basically what we will study.

(Refer Slide Time: 02:08)

Introduction

- VLSI testing, only from the context of combinational circuits (last 3 modules)
- ATPG for sequential circuits.
 - variations required in fault model, algebra and ATPG procedure
 - compare the ATPG Complexity
 - A special scheme called "scan chain" which "modifies a sequential circuit into a virtual combinational one".

So, test patterns for a sequential circuit with scan chain can be generated with slight variation in ATPG algorithms for combinational circuits (D-algorithm, for example).

NPTEL

So, I mean we have seen combinational circuits from last three days. So, now, we will today, we will see for the a sequential circuit, that is having a clock. So, what is the verification required. So, we will see what is required in the fault model. So, we will just see that the fault model is not is absolutely similar in case of combinational sequential circuits, but as 1 2 small assumptions may be there, that will point out.

So, we will see that, there will be a requirement of a higher order algebra for sequential circuit and we will see the ATPG procedure and we will see how different is ATPG in sequential circuits compare to combination circuit, that is what we will see, in the next lecture may be we will see something like a scan chine etcetera. So, which can help you in ATPG, so will show that, the ATPG procedure complexity in case of sequential circuit is much higher than the a combinational circuit.

So, following that, we will find out a technique design technique or scan chain, which will help you use the complexity and which will help in ATPG for sequential circuit that will be in a in the next two lectures may be. So, I mean so test pattern generation for a, I mean, we will see this scan chine and also test pattern generation for sequential circuit with scan chine can be generated with single ATPG algorithms D example and all.

So, this is about in details, we will see slowly, I mean in today's class what, we will discuss, we will mainly see, what is ATPG about different, why is ATPG different in

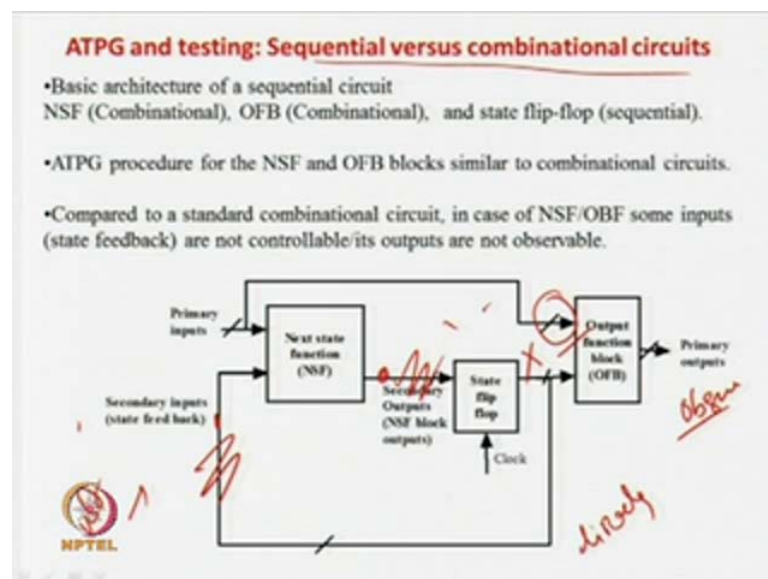
sequential circuits. And then we will see that in the next lecture and also we will point out that, ATPG for sequential circuit is more competence in combinational circuit.

So, followed by so, in next lectures, we will see that, how we can develop something call a scan chain, which will convert, you sequential circuit to a virtual combinational circuit. And then all algorithm, which are, which develop for combinational circuits can also be applied to sequential circuits with a scan chain without any much change.

So, in that way, we will handle the complexity of ATPG for sequential circuits, but that will be in a later half of the lecture. So, that is why it is say that, basic motivation of developing these 3 lectures or 4 lectures on sequential ATPG will be to convert, we complexity of ATPG for sequential circuit compare to combinational circuits and then convert the sequential circuits to virtual combinational once using scan chain.

So, that means, ATPG algorithm for combinational circuits like d-algorithm etcetera can be applied to sequential circuit with a variation. So, that is the basic agenda of our 3 or 4 lectures on sequential circuit testing, in this course. So, as we said that first (()) our main goal today or in today's lecture will be to see how combinational circuit ATPG is different or how sequential circuit ATPG is more complex compare to combinational circuit ATPG that much, we will see.

(Refer Slide Time: 04:35)



That is why we say that sequential versus combinational circuit ATPG and testing, what is the difference that is let us first study. So, if we look at this look at the figure this is the basic architecture of a digital circuit. So, here we have the next state function block, which takes input as the primary inputs as well as the feedback from the flip-flop. So, feedback from the flip-flops, we consider them a secondary inputs or virtual primary inputs, because they are not totally, primary inputs, they are virtual, because they are actually, feedback from the state registers or flip-flops.

The output of the next state function block actually, tells you it will be next state of the circuit, which are also call secondary output, because the primary outputs will be output from the output function block, which we will see later. But the output of the next state function block NSF will tell you, that this is your secondary output, because this is not directly output, but it is the secondary output, which will determine the next state of the circuits.

So, this also sometimes call the next state beats of the circuit and similarly and output of the state flip-flop of the present state. So, we all know that from the digital design. So, this comprises this present state and also they are actually, feedback to the next state function block. So, we call them as virtual primary input or secondary inputs and then there is a last block, which is actually, call the output function block, which takes as the input the present state as well as the input.

So, if we does, if we take this input call it the mini machine and if we does not take this primary inputs only it depends on the next state sorry, the present state. So, it is call the (()) machine. So, already we studied, they I mean the digital design.

So, any way make the things general the output function block will take the primary inputs as well as present state and it will generate the primary output. So, I mean, if you look at this block carefully. So, we have 3 main blocks, next state function block, state flip-flops and output function block, among them this is a combinational circuit this is a combinational circuit block and state flip-flops of the sequential circuit blocks.

So, now, if you look at this, from this sequential circuit basic block diagram of the combinational circuit point. So, you can see that, this is a combinational circuit this is a combinational circuit. So, you could have done ATPG using or D-algorithm or whatever we have learnt, but this is a small problem here.

So, in case of D-algorithm or automatic test pattern generation by random pattern generation, what we assume that in a circuit the primary inputs and the outputs are controllable and observable respectively, that is you can apply some patterns in the primary output directly and you can observe the patterns output directly.

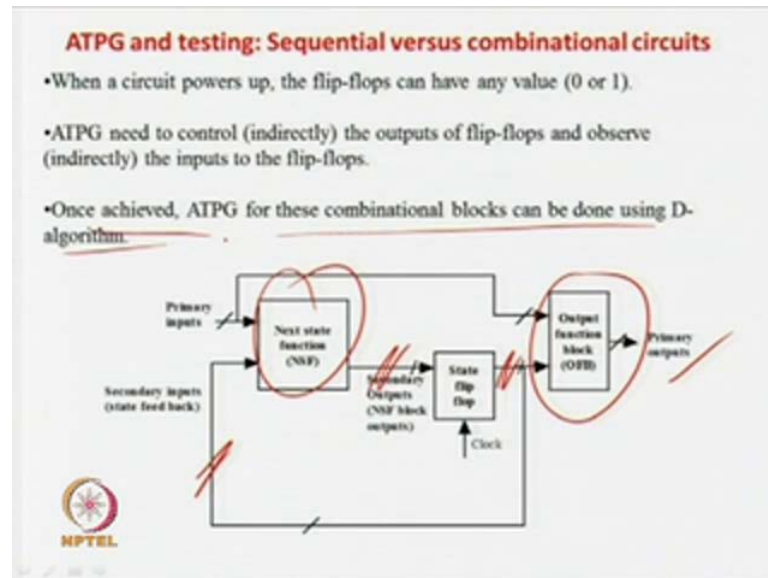
That is no need to for extra circuit to control this inputs are observe the primary output, you may require some circuits to control intermediate lines, what we have again discuss that approach. But, if you look at this very carefully, when you circuit will start so, output of the flip-flops are not known sometimes, they can be 0, sometimes they can be 1, you do not know exactly what will be the value.

So, generally call them as x , only you can control the primary input, similarly for the output function blocks this set of input that corresponds to the state flip-flop outputs are x , that is they depend on what will be the value when your circuit starts. So, if you look at it. So, this is a primary output, so very good. So, you can observe them directly. So, there is not a problem, but you can control only half or one-third or some or a fix fraction of your inputs of your circuit for this combinational block and the output function block.

Similarly, for this block, if you see you can control this portion of your primary inputs for this next state function block, for the secondary inputs of the virtual primary inputs, you cannot control it directly. Similarly, again another problem with this block is this outputs actually, also you cannot directly observe, because they are going to be fair to the next flip-flop, next flips I mean as the input of the flip-flop.

So, this cannot be observed directly observed directly, similarly this cannot be not control directly. So, that is actually the difference. So, had it been a purely combinational circuit testing algorithm then you should have been able to there should have been a decoupling over here. There should have been a decoupling over here. So, you should have been directly able to control these points and you should have been able to directly observe this points similarly, for this.

(Refer Slide Time: 08:34)




So, as these things are not possible, so there is what we require. So, we require two extra steps that is when you are going to test a sequential circuit, there are the two combinational blocks. So, I think you can directly use the combination algorithms like ATPG D-algorithm and all, but with a extra constrain that somehow, you should control the main directly and somehow, you should observe them indirectly.

Similarly, for the output block, you can observe directly for this part no problem or somehow, you should control this indirectly. So, that is what it says once achieved then ATPG for this combinational blocks can be done in the D-algorithm.

(Refer Slide Time: 09:06)

ATPG of sequential circuits: Assumptions

- Single clock sequential-circuits.
- Each flip-flop is treated as 1-bit memory element with ONE common clock.
- After a clock edge, the secondary output pattern (next state) is transferred to the output of the flip-flops (present state), which become new secondary inputs. Also, the primary outputs are updated. This activity occurs at each clock edge, and so it is called "synchronous" operation.
- Single stuck at faults are assumed in NSF and OFB.
- Internal faults of flip-flops are not modeled; their output and input faults are modeled as faults on input and output signals of the combinational blocks.
- No faults are considered in the clock signal.
- Most of the time, D-flip-flops are used in VLSI designs. So in this course whenever we refer to a flip-flop we essentially mean a D-flip-flop.



So, that is what we are going to see in today's lecture, how to do that that is, we apply D-algorithm, what is basically, we have to control the virtual primary inputs and say somehow, we have to observe the virtual primary outputs, this is what our job there our job will be done. So, ATPG for sequential circuits will be same as ATPG of the combinational circuits.

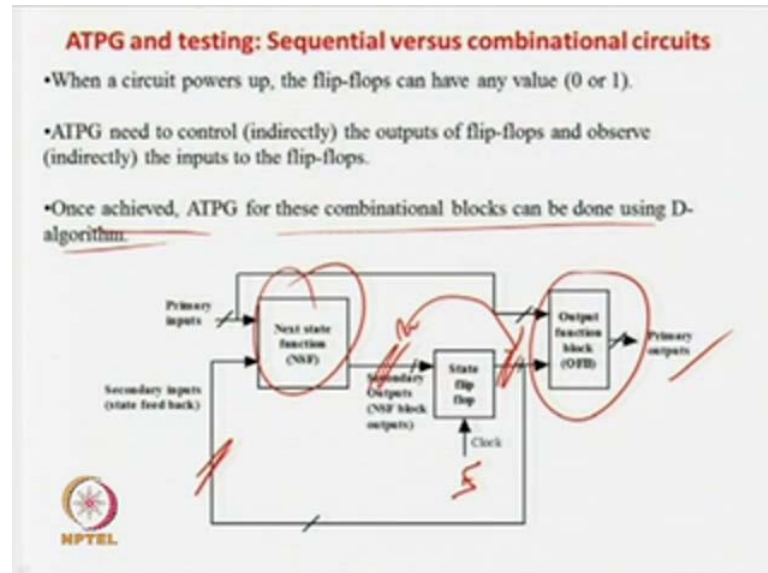
So, now we will see is there some assumptions, which will be taking. So, for virtual for ATPG of sequential circuit, that is some properties of the circuit, we have taken as assumption to make our, I means algorithm simple on the same way. So, this set of assumptions actually, apply for a very very high fraction of the digital circuit, which are being design.

So, we are I mean actually, targeting the majority of the designs. So, single clock sequential circuits, we are taking then a clock has a single circuit and it is sequential. So, each flip-flop is has a 1 bit memory element, that is obvious and 1 common clock that is 1 clock is being, I mean given to all the flip-flops, so all the flip-flops, which in synchronic with the single clock.

So, what it say that, next assumption is after the clock edge that, you can say that can assume that everything is what positive way, that is your secondary primary output, that is your next state is transfer to the output of the flip-flops, that is present state. They and

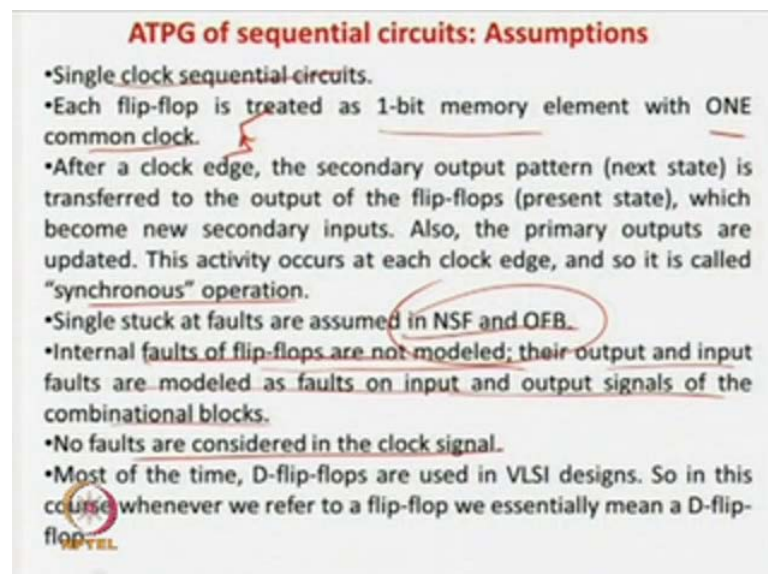
it becomes a new secondary outputs, also the primary outputs are updated, this activity occurs at the clock edge. So, it is called a synchronic operation.

(Refer Slide Time: 10:23)



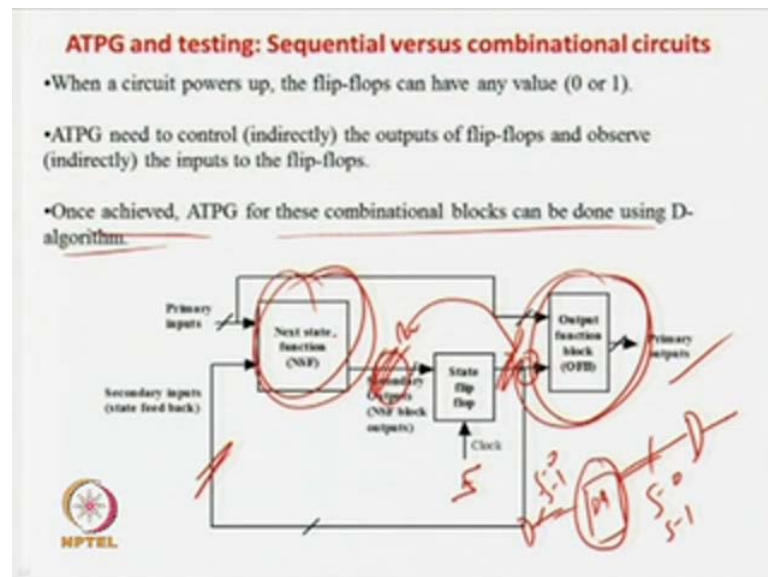
Basically, what have what they have said is that. So, there can be some input here. So, now, you give a positive clock after the positive clock edge, this output will come at this 1 and it will happen for all the state flip-flop in one goal for this for the positive edge of the clock. So, that is all changes that is all changes, in this all changes, we are actually called in secondary output, secondary output will become secondary inputs.

(Refer Slide Time: 10:45)



At the positive edge of a clock and it will happen for all the flip-flops circuit. So, it is called a synchronic operation. So, that is what is the idea, next important point is that, single stuck at faults are assumed in the NSF block and the output block.

(Refer Slide Time: 10:58)



So, we assume that these stuck at faults are at this nets and this circuits and the gates at this net and also gates at this net. So, I mean here, you are actually having some, you can say some D flip-flops are there. So, D q some D or j k or whatever some flip-flops are there generally, in case of circuits, we use flip-flops.

So, we are assume that all the D flip-flops are available, in this state flip-flops, now what happens, we are saying that say this flip-flops will be driven by some gate of this next state function block. So, you can assume that, this is the stuck at 0 fault here and stuck at 1 fault over here, similarly this is driving some other gate in the output functional block, also output function block or also some gate in the next state function block.

So, again you can have a stuck at 0 fault and stuck at 1 fault here, but internal to the D flip-flops, there other some other gates like feedback and all those things as we all know the internal structure of a D-flip-flop. So, it is assume that the faults are not present in those internal gates.

So, I mean if you this second assumption says that stuck at faults are the output of the NSF block and output block internal faults of the flip-flops are not modeled, their output

and input faults are modeled and faults of the input output singles are the combination blocks.

So, that what is being say that, we models faults here, we also models faults here, that is the input pins of the flip-flops and the output pins of the flips-flops will have faults model. Because, you are considering flip-flops, you are considering faults at the outputs of the gates of the NSF block as well as also the inputs of the gates at the output function block, just like I have given an example.

So, from gate here will be driving, this which will have a fault, which we can think of having a fault, similarly will be some gate whose input will be taken from here. So, this will also, you are considering as a fault. So, faults at the input and output pins of the flip-flops will be considered, but we are not considering the flip-flops faults at the internal of the flip-flop gates.

So, I mean this is to keep the thing simple, because if you start considering flip faults at the flip-flops then, you can we can show that the circuit may not remain a sequential circuit at all. So, also it may 1 flip-flop may start having multiple number of states and all those much more complex modeling will come into picture.

So, the order of complexity will be very very high. So, in the very first few lecture, we have say that, what is the beauty stuck at fault model. So, we are considering stuck at fault model, because it is simple to model, you can do collapsing and all number of test patterns require a very very less at the same time, that give you assurance that 99.9 percent cases. If, you are assuming stuck at fault, you can be assure 99.9 percent, that there is no defect, if your circuit does not have a stuck at fault.

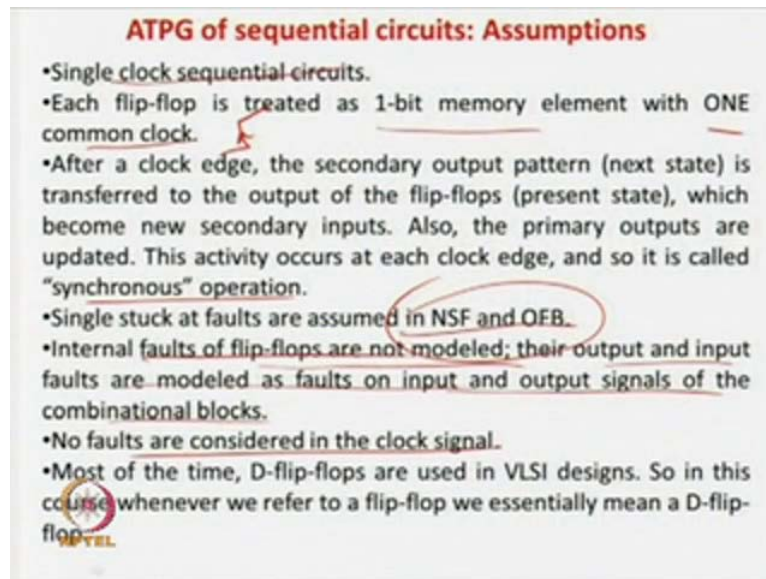
So, same thing logic also actually, holds for a sequential circuit, if you consider faults only at the input and the output of the flip-flops and forget about the internal fault faults of the flip-flops. Because, if you consider a faults are the internal of the flip-flop may be your accuracy improvement will be 99.9999 plus 0. 0 0 0 1 plus 0. 0 0 0 0 1 may be there, but to improved that the amount of complexity, that will come into the algorithm will not make any help.

Unnecessarily, you will find that the circuit will have more number of states, the circulate may become a combinational circuit from a sequential circuit to handle all these

complexities. The ATPG algorithms will be so, complex and the number of test patterns require may be so, complex that, we may loss what we are going to gain by adding this.

So, and people are followed experimentally just even, if you forget the internal faults of the flip-flop still your coverage assurance or your assurance that, you proves cover the stuck at fault, there is no real defect that assurance between this correlation between these stuck at faults and the defects then is very very high.

(Refer Slide Time: 14:20)



ATPG of sequential circuits: Assumptions

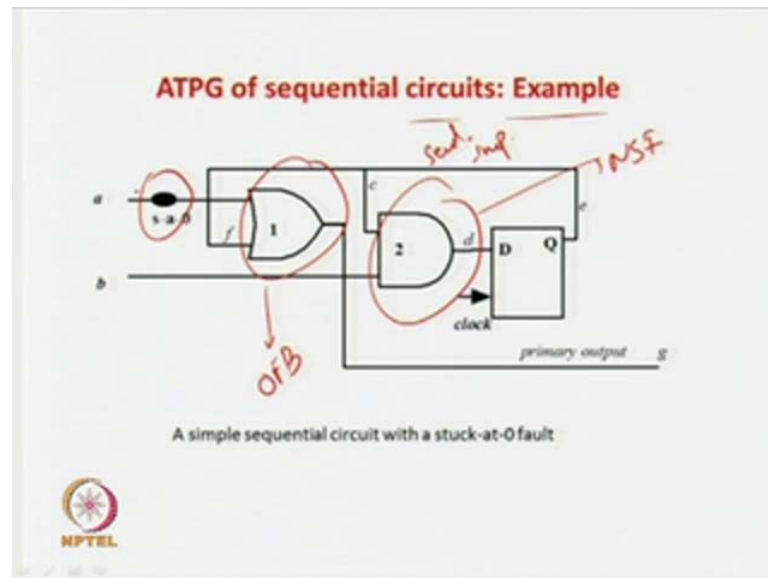
- Single clock sequential circuits.
- Each flip-flop is treated as 1-bit memory element with ONE common clock.
- After a clock edge, the secondary output pattern (next state) is transferred to the output of the flip-flops (present state), which become new secondary inputs. Also, the primary outputs are updated. This activity occurs at each clock edge, and so it is called "synchronous" operation.
- Single stuck at faults are assumed in NSF and OEB.
- Internal faults of flip-flops are not modeled; their output and input faults are modeled as faults on input and output signals of the combinational blocks.
- No faults are considered in the clock signal.
- Most of the time, D-flip-flops are used in VLSI designs. So in this course whenever we refer to a flip-flop we essentially mean a D-flip-flop.

So, that is why we can assume that, faults are only at the N S F block and output and we can fault, we can forget about internal faults of the flip-flops. Similarly, no faults are considered as the clock, because if you take faults at the clock things will be very very complex like a sequential circuit may not be a combinational circuit. Some of the circuit may become as synchronies, because for some of the parts of the circuit clock may reach for some part of the circuit clock may not reach, making a synchronies circuit and algorithm will be terribly complex.

And there will be also, we will not gain much in assurance level. So, people have experimentally, found out that, you can forget about faults in the clock, you will also forget about flops in the internal gate of the flip-flop still the (()) stuck at faults and whatever, you call the real defects are very very high. So, we will consider only flops in the inputs and outputs of the N F S block as well as the internal gates of the N F S block and the output block.

And also mainly our as I told you VLSI circuit are mainly mainly consider, what you called are the D flip-flops there than j k flip-flops or j flip-flop, because one flip-flop can be converted to other you all know, but in the case of digital circuits, we mainly consider about your D flip-flops. So, in our assumption is that all the circuits will be taking, hence for are made up of D flip-flop.

(Refer Slide Time: 15:29)



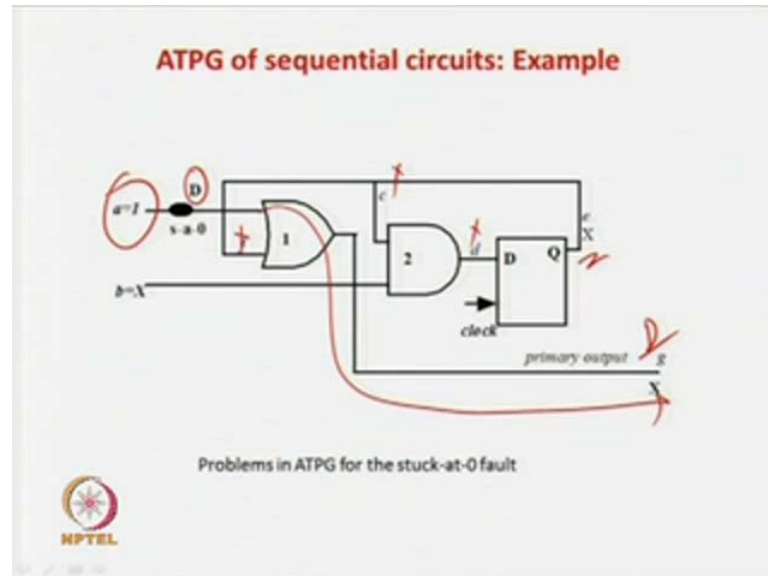
So, now again let us go back go to an example to tell what about the complex discussing. So, you may be lost, so let us see the example of the complex circuit, I mean is very simple example circuit. So, you can see that, if you look at this is the simple sequential circuit, now you can see that this gate actually, depends on the one input of the flip-flop and the primary input and it drives the primary output.

So, it will become your output function block, you can map it to the old diagram and find out. So, what this gate map belongs to it depends on the primary input D and also it depends on the output of the flip-flop, this is actually, the secondary input. So, you know that the block, which depends on the secondary input as well as the primary input is called the next state function and the output of the next state function blocks goes to the flip-flop as input and will return the next state.

So, one actually, is the output function block and this is the next state function block, if you just map it to this 1, you will find out. So, let us consider a flop as a I mean sorry,

faults stuck at 0 at 1 input of your output function blocks. So, now, let us see how you can do the testing.

(Refer Slide Time: 16:29)



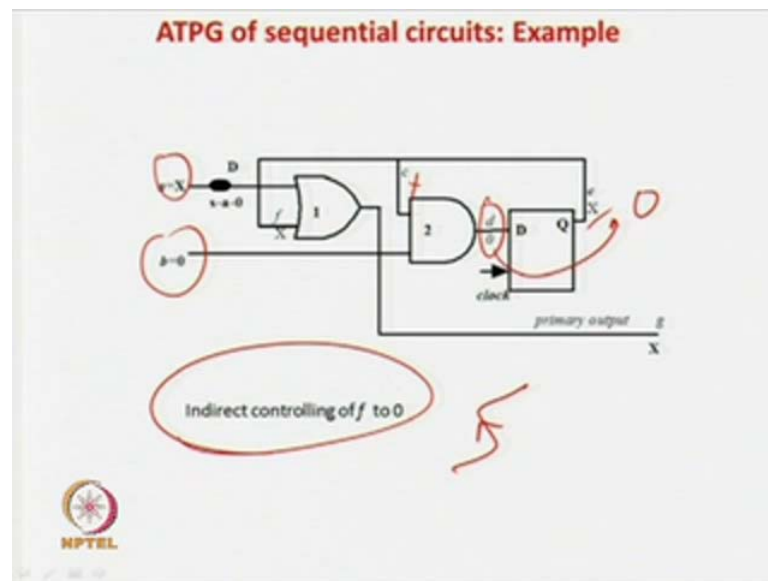
So, this is the case, so only you have to remember that this is the output function block and 2 is the next state block. So, now when this circuit starts up as you all know that initially, every all the gates all the nets are having a value of x as for D-algorithm. So, now, this is a stuck at 0. So, you have to apply a 1. So, it will be D, now 1 thing is that. So, there is actually this is the D.

So, there is only 1 path through, which the fault affect can be propagated that is actually, called a singular singular D path. So, you have to a fault affect should be there. So, our target is that. So, gate 1 is in the deferent here, because 1 input is D and the other output is a x and there is only one path through, which the fault can be propagated this called the singular D path.

So, this should be a D after propagation of the fault affect again, now it may 1 may be now it is D output is D. So, this f is unknown x. So, by the j algorithm or the j concept, we know that f has to be a 0, for the fault affect to be propagated. So, we require that 1, but now you can see, because the circuit, because now you cannot directly control f, that is the deferent between a sequential circuit testing as well as combinational circuit test. Had it been a combinational circuit then what would you have to be done, f could have been directly controlled and you could have apply 0 at this.

But, now is not directly controllable, because this the output of a flip-flop and the output of a flip-flop will totally depend on what is the value when it gets started. So, now, it is x. So, if you require it to be a 0, what now it is a x. So, we are in a problem, because we do not know how to make at this 1. So, we have to know do something. So, that we get f equal to 0 and once f equal to 0 then, you can apply this 1, to get this to get the fault testing.

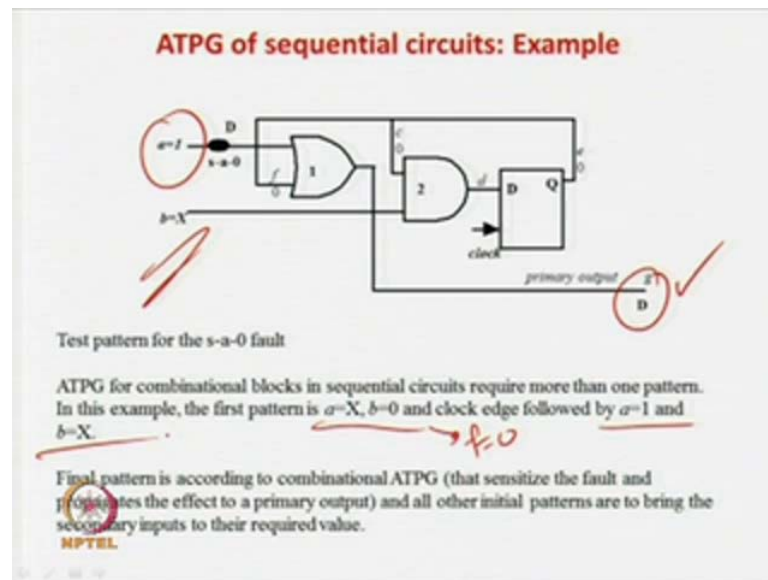
(Refer Slide Time: 17:58)



So, our main job is now to get a 0 at f. So, actually it is called indirectly control of f to 0, because you cannot directly this do this 1. So, how can you do that, let us say that we put b equal to 0. So, if b equal to 0, this is x because output. So, if you apply b equal to 0 then irrespective of what is the x, x can be 0 or 1, we do not know whatever already told, you the x means either 0 or 1, we do not know because the 1, the power of the circuit, we can have any value at x. So, even if it is 0 or whatever be the case, but if you put D a 0. So, D will be a 0.

Now, you put b equal to 0 x, I do not bother at the time b and you apply a positive edge of the clock. So, once you do this, this 0 will be coming now over here right. So, you apply a b equal to 0 and apply you get a 0 and apply a clock pulse.

(Refer Slide Time: 18:41)

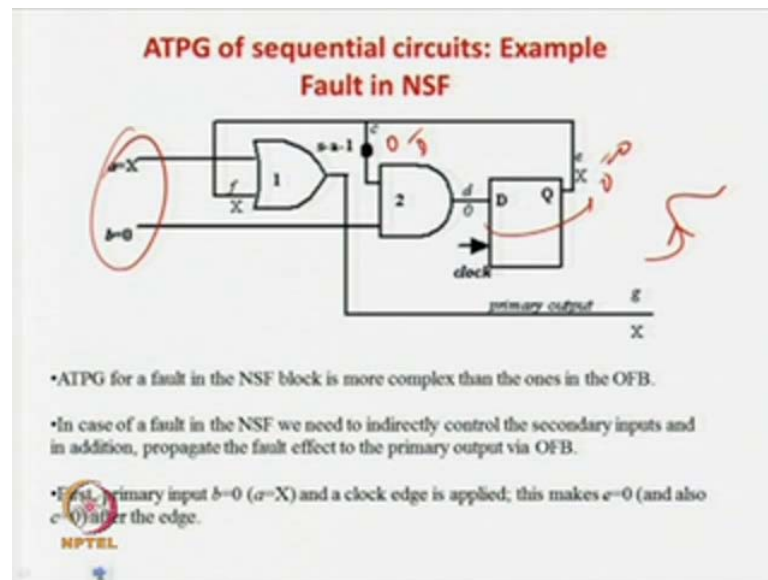


So, once that is done. So, what you are going to get, you are going to get a 0 over here, you are going to get a 0 over here. Now, you indirectly, you have f equal to 0, now you can put a equal to 1, you can totally forget for the time b and then, you will get this value of the output and circuit.

So, now, in case of sequential combinational circuit, what we have seen, there are only 1 pattern can test your fault correct. So, you find out by sensitize propagate and justify or whatever by a random pattern or but, 1 test pattern can detect a directly a with a there is a fault in the circuit or not. But in this case, we require two patterns as you can see first, we have to apply a x b 0, a is we do not know b equal to 0.

So, which will determine make that, f equal to 0 indirectly. So, once f is equal to 0, now you apply a equal to 1 and b equal to x . So, that your circuit will tested, so this is the basic fundamental difference between sequential circuit testing and combinational circuit testing. So, in sequential circuit testing, what we have done.

(Refer Slide Time: 19:34)



In sequential circuit testing, we require that, first the virtual primary outputs or secondary inputs whatever sorry, the virtual primary inputs or secondary inputs whatever, you call they have to be indirectly controlled. Because, there the outputs of the flip-flops, they have to indirectly controlled to some value, that is require for ATPG.

So, you may in this case, there is only 1 flip-flop. So, you require only 1 step. So, slowly we will see how many steps are required to make, this make the secondary inputs controllable. So, you will require 1 1 pattern 1 clock then 1 paternal clock some amount of patterns and a clock. So, in this cases only 1 pattern required and a clock has to be applied.

So, it will virtually or indirectly controlled your virtual inputs, now once the virtual inputs are controlled then you can directly controlled your primary inputs and then you can see, what is the ATPG is done. So, in this case, we require two patterns do it.

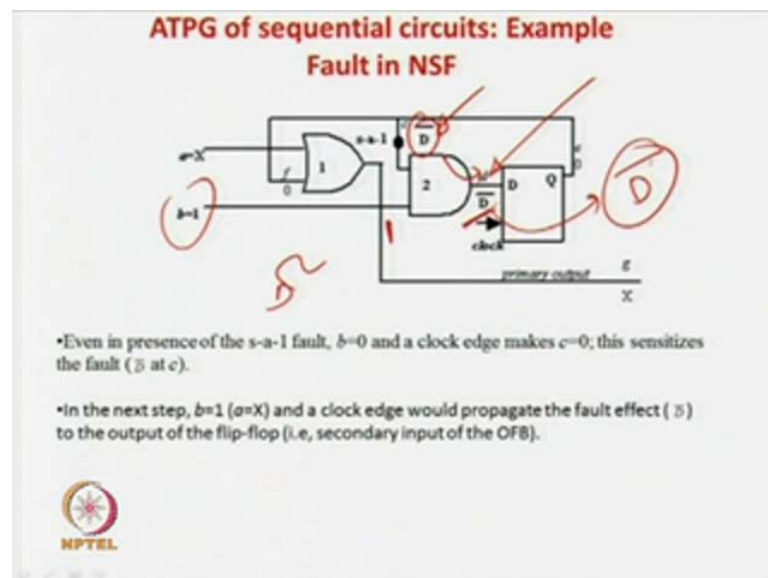
Now, let us see that, but in this case, we considered a fault, in this outs nested function block this was NSF. So, in case of sorry, this was the output function block sorry, this was the output function block. So, we consider a fault here. So, somehow we required to control this net to 0. So, we have done is directly, but the output was directly observable. So, we could have seen done this in 2 pattern, but now we will take the another fault in this block, which is actually your next state function block.

So, here like this be more difficult, because you have to directly control this sorry, indirectly control this secondary input as well as the output of the next state function block is not directly observable. So, you have to somehow make it indirectly observable. So, faults at testing faults at the next state function block is what more complex then, you are what do you call, this output function block.

Because, in case of output function block that the primary output is directly observable, but in this case it is not. So, stuck at 1 fault, so what do you require for stuck at 1 fault, so obviously, you require a 0 over here correct. So, now, this requires a 0 means, it is D prime over here, so but to get a D prime over here. So, you have to get a 0 over here, but as you know this is the output of a flip-flop. So, we have to do that indirectly. So, how you can do that indirectly.

So, you do not forget about x a b you put as 0. So, inexpertly everything, you get a 0 over here and then you get a clock pulse. So, once you do this, you will get this as a 0 over here, So, this is what is being done. So, if you make D equal to 0, apply a clock pulse then, you get x equal to 0 and your fault get sensitize.

(Refer Slide Time: 21:49)



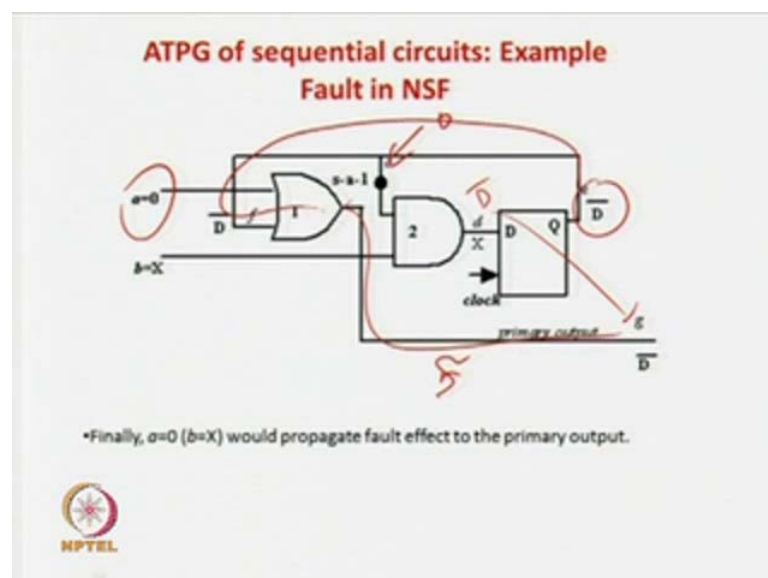
This is what you do in the first step is indirect control correct. So, you get this 1, now next time what you have do, so you get a D prime over here. So now, what you know. So, you fault the sensitize. So, previous case b equal to 0 was the previous case. So, now, what you have to do. So, you know that, if this is the fault, this is the only and gate. So,

therefore, this is becomes D front here and unique, D front here or there is only a unique x path in this case.

So, you have to somehow propagate, the value of D prime over here. So, for that you require to have a 1 over here, So, in the next stage, what you do you may forgot about x make b equal to one and then. So, it will be a 1. So, this D prime D bar will be floated over here and then you give a clock pulse. So, once you give a clock pulse over there. So, d prime will appear at the output it, now you have the observe that even, if D prime has come to the output D, that is the virtual primary output, but still you cannot observe the directly unlike your next state function block, because it is going to the input of a D flip-flop.

So, that is more that is why more complex fault should be tested in the what you called output function blocks sorry, fault excuse me fault, which are in the next state function block is more difficult to the tested, because not only you have to done indirectly control some lines, but also to indirectly observe some lines. Which was not in the case of gate 1, which is the output function here. So, somehow you have brought this here, but not you cannot observe it directly. So, you have to somehow make this D prime appear over here, so you have to the third step.

(Refer Slide Time: 23:10)



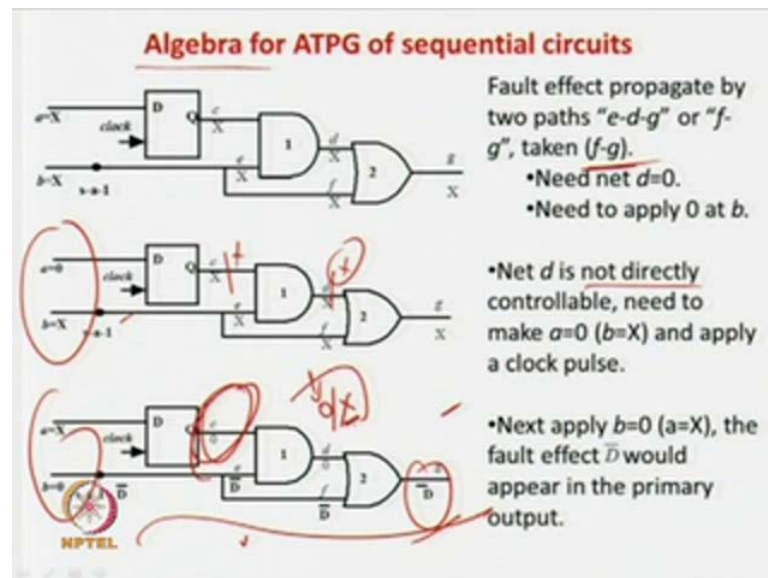
So, last we could have done everything, in the seam 2 steps in this first, we controlled f 2 some value and then we fault effect was propagated here But, now in this case 3 steps are

require first, you sensitize this to 0 then you propagated the fault value to here and then as this is not directly observable somehow, you have to bring this value to here.

So, next step, so in the second part what we have done. So, we got this 1. So, second stage what we have done. So, it is 0, so sorry it is 1. So, this fault effect from here, it has come over here, now you apply a clock pulse. So, if you apply a clock pulse over here, this is a third step you apply a clock pulse over here. So, your D prime will come here, so now, your D prime is here. So, that is in the third phase you get a D prime over here, but now again this has to be observed by this 1.

So, you have to make a equal to 0. So, we are indirectly applying 3 clock pulses, if you see first a equal to x, b equal to 0. So, you are indirectly, you are controlling this value this 1. Next you are applying a x equal to 1, x equal to 1, b equal to 0. So, your fault effect is propagate into the output, but the output is cannot observe. So, third step what we do again, you put a clock age. So, that it comes out indirectly via your primary output.

(Refer Slide Time: 24:18)



So, that is a more difficult, if you are considering faults at the next state function block compare to output function block. But for all the cases unlike combinational circuit, you have to indirectly control for virtual m secondary inputs and indirectly observe the secondary output, that is what is the basic conceptual deference. So, in case of sequential circuit more than 1 test patterns may be required to test a fault.

So, indirectly so, broadly speaking the complexity of a sequential circuit testing is very more difficult compare to a combinational circuit testing. So, in this case, we you have to observe that, we are not considering the cases where, there can be inconsistency and when there can be back track. So, you can understand that how the complexity will blow up, if you start considering this. So, it may use that D-algorithm to find out that, this can be a good process of propagation and all.

So, now, instead of 1 pattern, we have generated 10 or 20 patterns, which are required to indirectly control, this virtual inputs and indirectly, observe the virtual outputs. But, now after that, you could have found out that for some step, there is a inconsistency and we have to do a lot of back tracking. So, in combinational circuit, there was only 1 pattern for a fault, but now there are 10 patterns or 11 patterns or x number of patterns for a fault.

So, again if you remember the if there is a back track in the final phase. So, how much amount of competition work has been lost. So, that is why sequential circuit testing is highly more complex than a combinational circuit. So, that is what we have going to pocus, today and in the next class, we will see how we can solve this complexity.

So, now, we will see that, what kind of algebra in Roth's 5 value as algebra, we have seen for combinational circuits, but now we are going to study here do we require some extra kind of logic or extra kind of algebra. We are asking the question a whether some extra kind of logic is required for ATPG for sequential circuit or whether 5 valued logic would suffice or not.

So, let us just see that, we will try to use a 5 valued logic and see what is the gain and what is the loss and then we try to see, we can go for a higher order logic. So, let us see that this is the circuit. So, this is stuck at 1 fault, so obviously, stuck at 1 fault over here this is a stuck at 1 fault over here. So, a stuck at 1 fault here means, you have to apply a 0 over here. So, now, there is 2 paths.

So, 1 path is the fault can be propagated here and other path can be here. So, there are 2 x paths here. So, for the time being let us see that, if this path is smaller. So, by any heuristic, if you apply a over D-algorithm. So, we can take that E N F may be a very good path for propagate the path, because the other path like e d and g is the longer path correct.

So, stuck at 0 0 means, it will be a D prime over here. So, you can propagate the value of D prime over here, d prime over here, but immediately, we know that D has to be a 0 over here, so how do you get a 0 over here. So, you know that directly, we cannot get a 0 over here. So, better what we can do this. So, how do we get a 0 over here. So, what we can do. So, net D actually already told that net D is not directly controllable. So, we have to make this net D equal to 0, because actually, this net depends on a output of a flip-flop.

So, this gate is not directly connect to the output of a flip-flop, but it indirectly dependent on net c which is the output of a flip-flop. So, indirectly we have to control this D to 0,

so if you just look at it, we can even is the do that, you can make a equal to 0. So, if you just make a equal to 0, now what you do, you get give a clock pulse. So, just if you give a clock pulse a equal to 0, then this 1 will come here and this is x. So, x and a 0 will be a 0 correct, that is what is our idea. So, that is this is how we were doing by D-algorithm.

So, just let us quickly see once more, what we have done. So, this is the fault stuck at 1. So, we apply a 0 over here, we need to apply a over 0 and this is the path propagation. So, we may get a D over D prime over here. So, this is the path and to the to get this 1, we require a D over here. So, D has to be a 0, over here, both you get fault this 1, but not these not directly, connected to the output of a flip-flop.

But, it is connected to c, which is the output of a D flip-flop and this can be directly controllable, see you may note that it is nothing, but a primary secondary input sorry, secondary output, because it is the output of flip-flop not directly controllable.

Somehow, you have to make this has to be 0. So, if want to make this 1 to be 0, because D to be 0, somehow you have to make this to be 0, because you know that 0, x is a singular cover. So, to make x equal to 0, what we can do we have to make a equal to 0 and give clock pulse. So, immediately D will equal to 0 and your D prime will be propagated to the output. So, what are the 2 test patterns to test the fault 1 test pattern is a is 0 b x and next pattern is a x and b 0. So, this will test your fault. So, that 2 pattern required to testing.

Now, let us just do a high freq of Boolean algebra and c can be achieve something better. So, let us for totalize just see this 1 and forget about all this. So, we can keep it as an x

for the time b keep it an x , now let us see what we can do. So, let us see, it is stuck at 1 and we apply a 0. So, normal case 0 fault case 1, we can write like this correct. So, now, in this case, you can see that, this is also D prime, we just write out this.

This we require correct, this is what we require something over here and here, we write also this is 0 slash 1, now just have a very quick look at the circuit just sorry, just look at this point. So, this point is very important, this net e , if you look very carefully. So, what is the idea. So, if just net D is 0 and a 1, now let us look sorry, net e is directly, you can net details about net e , you can directly get from this net because, it is the fan-out for this state, now let us look at net D very carefully.

So, net D is in fact, x by D -algorithm or Roth's 5 value algebra, now you can see, because 1 value is x and 0 1. So, if you do something like this. So, you can will, you will get 0 and x kind of a thing, you will get at the output so, but 0 and x is not a part of your Roth's algebra. So, let us see in this case. So, we get D is equal to 0 1 and this is an x , but now look at D very carefully.

So, now what happens say less, we have forget about this being 0, let as sum that, it is x we do not know the output of the this one. So, if fault is not there, what if fault is not there. So, what is the input over here, it is a 0 and what is the value of this it is x , because we are not consider, we do not know the case or for because, we do not know the output of this 1. So, what is the output of D at fault. So, if the fault is not there normal. So, output will be a 0, because if the fault is not there stuck at fault is not there. So, you apply a 0 over here. So, you will get a 0 over here.

So, the fault your D is going to be 0 and other things are fine, now if there is a fault then what will be the case, if there is a fault then this stuck at 1. So, you are going to get 1 over here and this is a x because, we are not applying anything for the timing. So, it will be a x . So, it is 0 and x at D , if you are not controlling this for the timing correct. So, now, the output of e is $D D$ prime that is 0 slash 1 and D , you can write as 0 slash x , if the fault is not there you apply a 0. So, the output will be 0, but if the faults stuck at 1, fault is there, you do not know what is the output.

So, it can be a x , now you just f , we know is 0 1. So, if you do a oaring of this 1. So, the output of g is what. So, it is 0 or 0 is 0 1 or x is a 1, which is nothing, but a d prime. So,

is very surprising cases happened here, that even without controlling this gate to 0 or this gate to 0 indirectly, which is required D algorithm or Roth's 5 value algebra.

So, if you have a element in this, which we are calling it as 0 slash x, because if you remember Roth's algebra, there is no concept of 0 slash x, we have x slash x 0 1 1 0 0 0 0 1 sorry, 0 0 normal 0 fault 0 1 1 normal 1 fault 1 x x, we do not know 0 1, that is D prime 1 0 that is D.

But, we did not have any concept of 0 slash x, but if you have a concept of 0 slash x then only 1 pattern can detect this fault, this is a very special case, in which case, I mean if this is not require to control an all. But, many most of the times, we may have to require this, but for some of the circuits, you can find out that, if you have the concept of the 0 say that is point e is 0 slash x then only 1 pattern x a equal to x and b equal to 0 with test your circuit.

But, if you are directly following D-algebra that is we do not have any concept, that is there is only 1, that is in the Roth's 5 value algebra, there is only 1 concept, that it is x. So, x be series x. So, it is x that means, x x. So, this is also x x something like this you cannot write 0 slash x or x slash 0.


So, in that case, you would directly follow the D-algebra. So, you require one pattern this 1 to make this D equal to 0 and then you require another pattern to propagate the fault value. But, if you write some, if you there is a concept or writing 0 slash x, which is the basically the case, because if the circuit fault is there only the output is not known, which will be dependent on the virtual primary output that is D, but if the fault is there sorry, if the fault is not there then you apply a 0.

So, at all does not dependent on the output of the flip-flop, that is at all dependent on the primary virtual, primary output or the secondary output then, you get the value 0, directly and by doing this way you can get a D prime over here without controlling this. So, this is a very special case of an example where, you can test a circuit sequential circuit without requirement of controlling the primary secondary primary output, because of the fat.

(Refer Slide Time: 34:00)

Algebra for ATPG of sequential circuits
Nine value algebra

Symbol	Implication	Normal Circuit	Faulty Circuit
0 ✓	(0/0)	0	0
1 ✓	(1/1)	1	1
X ✓	(X/X)	X	X
D ✓	(1/0)	1	0
D' ✓	(0/1)	0	1
G0	(0/X)	0	X
G1	(1/X)	1	X
F0	(X/0)	X	0
F1	(X/1)	X	1



But, now for that you require, some special symbols, which we are this was about the theory, which we are discussing that higher order algebra, we required a symbol like 0 x and all these things. So, higher order algebra come back. So, by adding 1 extra symbol, that is actually, 0 slash x, we got that in case of sequence ATPG only 1 pattern or at least 1 level of competition was less required.

So, 9 value algebra are use this sequential circuit, that is 0 1 x D and D prime, these are from Roth's 5 value algebra. But, now we have added some more 0 x, that is normal case 0 fault case unknown g 1 normal case 1 fault case unknown x 0 normal case x fault case 0, normal f 1 is normal case unknown fault case 1.

(Refer Slide Time: 34:49)

Algebra for ATPG of sequential circuits

- In the example, higher order algebra is used, as a net is marked as 0/X, which is not available in 5 value algebra.
- Higher order algebra improves efficiency of ATPG of sequential circuits. As higher order algebra reduces the number of input (primary and secondary) lines to be controlled, there is reduction in the number of steps (in terms of clock edges and test patterns) to control the secondary inputs (or make the NSF block outputs observable via OFB). However, it does not guarantee that ATPG will not require controlling the secondary inputs and propagating the fault effect to the primary output via OFB. This example was a special case where controlling the secondary inputs were not required.
- Higher order algebra will also reduce the number of lines to be controlled in ATPG of a combinational circuit. However, it is not applied as computational complexity rises with increase in order of the algebra and inputs are easily controllable in combinational circuits.

NPTTEL

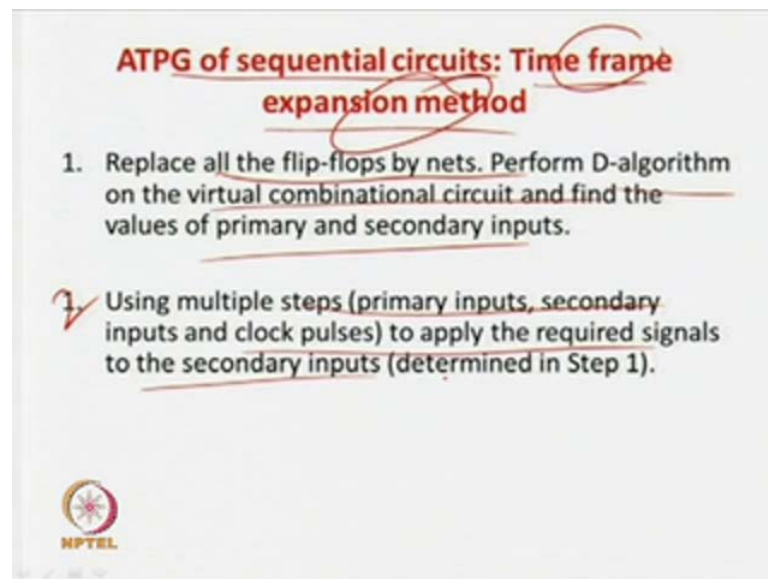
So, with this addition of these things, you can see that for many times, you will be requiring less amount of computation to solve your problem. So, that is what is required in the last example higher order algebra this one. So, which is required to reduce the number of computation or only one test pattern can do it and it is not available in a 5 valued algebra. So, that is why we sometimes require to go for a high order algebra that is 9 value algebra.

So, higher order algebra improves the efficiency of sequential circuit testing, because you require less number of test for the computational, but only 1 thing, you have to observe that always, it will not be possible to even, if you are using a 9 value algebra that directly, you did not required to indirectly control any virtual primary outputs or virtual primary inputs kind of a thing. So, special case when it happened, but only when assure, you can assume that found on experimentally that, if you are using higher order algebra then the number of computation.

Generally reduces, because you have some more flexibility like x 0 1 x 0 x 1 x something like this. So, more the number of excess in the circuit, we have seen that more number of flexibility is there. So, that is why it will improve it, improve your efficiency, because in case of combinational circuit testing 1 pattern was used to detect this test and, but here you require more than one pattern. So, lot of computations are required. So, you want to a more flexibility in the circuit. So, you are using some other higher order algebra.


So, you can also question me that, higher order algebra would have also helped in combinational circuit the answer is true, because in the last lecture with an example, we have shown that, if you have more number of excess in your circuit then it is true that there will be more number of flexibility and ATPG may succeed better. So, if you have 9 valued algebra in combinational circuit, you are correct, you will get better solution, but having a 9 order algebra is also combinational difficult.

(Refer Slide Time: 36:23)



ATPG of sequential circuits: Time frame expansion method

1. Replace all the flip-flops by nets. Perform D-algorithm on the virtual combinational circuit and find the values of primary and secondary inputs.
2. Using multiple steps (primary inputs, secondary inputs and clock pulses) to apply the required signals to the secondary inputs (determined in Step 1).

 NPTEL

And it has been found out that the amount of gain, you are going to get while using 9 valued algebra is not that much in a combinational circuit by the amount of inconsistency that, you may reduce. That is because in sequential circuit, there are more number of test patterns, more number of patterns are required to find out a test pattern. So, we want more flexibility.

So, you use this extra 4 symbols like 0 8 x 0 and all this things, but same thing, if you bring out in a combinational circuit and make a 9 valued algebra for combinational circuits. So, some advantage, you will get that is defiantly, because more number of execs will be there in the circuit, but the amount of complexity will incur there will not help you to gain, because in combinational circuit one pattern is enough to detect the circuit.

So, I mean inconsistency more less compared to sequential circuits. So, that is what so, that is what about the higher order algebra and we have seen the motivation. So, now,

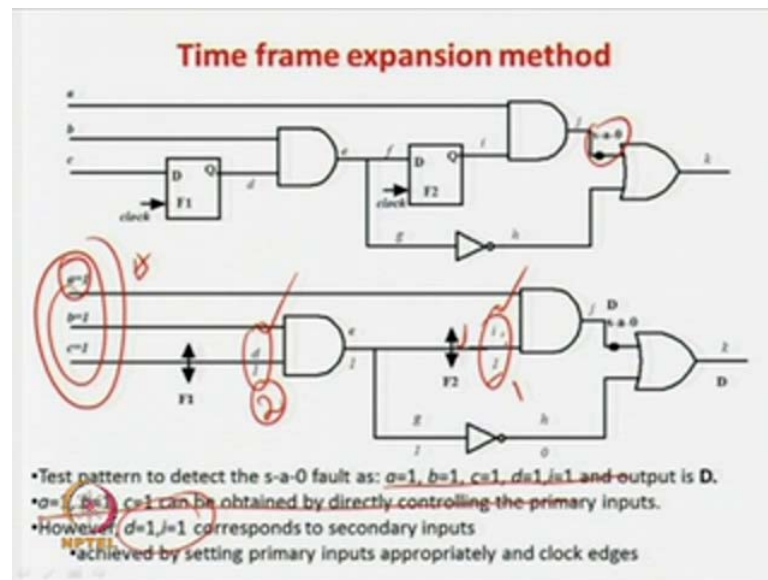
what we are going to do. So, we are going to see ATPG for sequential circuit using time frame expansion method that is like D-algorithm, we have seen. So, in D-algorithm what was the idea, that is a formal algorithm to find out the test pattern for combinational circuit.

So, in this case this time frame expansion method is another formal method for what you can say it is the formal method like D-algorithm, which can find out test patterns for sequential circuit. So, let us see this in details with an example. So, what is there first step replace all the flip-flops by nets. So, first there will be sequential circuits will be there. So, there will be some flip-flops over will be there. So, as purely sequential circuit is a singular clock is the assumption. So, then you remove all the flip-flops, that is the first job and perform D-algorithm on the virtual combinational circuit.

So, when ever if you remove flip-flops so, what will happen, the output of the flip-flops will become your virtual primary inputs and your that is what. So, the we know the and the outputs of your and the inputs of the flip-flops will become your virtual primary output, see what we have done, we are removing all the flip-flops. So, if you remove the all the flip-flops what is happen the outputs of the flip-flops will become virtual primary inputs. Now, you can also consider them as the inputs and the outputs of your sorry and the inputs of your flip-flop will become the virtual primary output.

So, which you seen in the example, that is what is going to happen. So, virtually you are going to get a combinational circuit, now you use D-algorithm on that and you would have to use. So, this step 2, you have to do multiple steps will be requiring to control the secondary inputs and observe the secondary output.

(Refer Slide Time: 38:38)



So, already we have seen that this would be the that for controlling the virtual primary inputs and observing, the virtual primary outputs you may require multiple steps. So, that may be require, but.

So, let us see with an example, because there was the mathematical definition kind of a thing. So, this let this is be your circuit to be tested and we have a fault over here. So, there are 2 flip-flops. So, you can consider, this as your virtual secondary input, because this output of a flip-flop, this is secondary input you can say similarly, you can also see this is the output of the flip-flop, this is also a secondary input, because you cannot control this directly secondary input, you cannot con observe this directly correct.

Now this is 1 input to the flip-flop and this is a special case. So, this is a primary input as well as the input to the flip-flop. So, you can say that, this is also a both a primary input as well as a virtual input and, but in this case you can see that, this is a output of this gate is going to the input of this gate. So, this is actually a virtual input, this is virtual primary input or secondary input, because you cannot control this directly, that is what is our idea.

So, this is virtually primary input, this is virtually primary output, you can say or secondary output, similarly this is output of the flip-flop, but you cannot see that directly. So, you can consider that this virtual primary output kind of a thing. So, that is what is a idea and also, you can say that this is again going to the input of another gate, because

this is also this is output as well as it is inherit mean directly coming out then we could have virtual primary output, because or a primary output something like that, but you can see that also it is going as a input (()) gate.

So, that can also be cannot be directly controlled observe. So, you can say that, this is the virtual primary input or virtual primary output, that is secondary input output. So, it is not directly observe to or not directly (()).

So, somehow we have to do this indirectly. So, you can easily map, this gates to, which is the N F S block and which is the output function. So, that can easily done, but anyway that is not much of concern right now. So, this is the stuck at fault over here. So, what is the first step, first step is remove this flip-flops. So, we have remove this flip-flop here, we have remove this flip-flop here and we are shorted this. So, this is what is your idea what is we easily remember that, you cannot control this directly a an also, you cannot control this directly.

That is even, if this is connection over there you, if you want a 0 say or if you want a 1 over here say you apply a 1 1 or 1 here, you cannot get the 1 here directly, because it is (()) by a flip-flop then assumption, you have to know that similarly, you cannot get a 0 here, directly or 1 here, may be not possible be even if, we have shorted this.

So, that is has to be remember, but it is now a so, but for the D algorithm [fo/for] for next step is that is now the with this small assumption in mind, that is not a very explicit assumption here. So, what do you what do, what we have seen that is not a very explicit assumption, but we just know in our mind that directly cannot be controlled our observe.

Now, this is a virtual combination, that was I was saying that is a virtual combination circuit, now in the virtual combinational circuit, you apply combinational D algorithm that is it. So, now, you just see what is the combinational D algorithm over here. So, we know that with the 0. So, we have to apply a 1 here. So, it is a D over here. So, it is the propegative.

So, it will be a D, now the other input should be a 0 or be this is 1 over here, you require and next is that you [ap/apply] apply 1 over here. So, apply a 1 here means a will has to be a 1 and this input also has to be a 1 correct.

Next what is the case, so this has to be 1. So, the output has to be a 1. So, 1 means the and gate this has to be a 1. So, if you apply a D algorithm. So, you you could have directly say that, a 1 b 1 and c 1, if you give and the output is the. So, your circuit would have been directly tested, which is the combination circuit, but here the big (()) over here.

So, you require a 1 over here, you require a a 1 over here, but as I already told you, that in internal you have to remember this big (()) that, this directly not controllable, this also not directly not controllable, that is a big that, you have to remember. So, how you can get a 1 over here, you have to do that indirectly.

Similarly, how to get a 1 over here, that also you have to do indirectly, because these 2 are output of the flip-flops, which are going to your input of some gates. So, they are your virtual primary input, they are not your direct primary input, there is a virtual primary inputs any how to conclude then virtual that is the idea.

Similarly, if some continues to observe, this let us that also not be possible, because they are either internals or you can also even if, they are feeding it some outputs of the flip-flops or even if there the going as a inputs (()), they cannot be done directly, because of this virtual primary output and virtual primary input kind of the behavior.

So, now let us forget about this. So, I mean let us see about is our target. So, our target is that, this you have to apply eventually, to they do this part before that, we require a 1 over here, because sorry, we require a 1 over here for that, we require a 1 over here. Similarly, that is that is done. So, this is our requirement number 1, this our requirement number 2 and once this this requirement is done. So, you get a 1 over sorry, this is not your immediate requirement. So, this is your requirement number 1, this is requirement number 2.


So, if you can get a 1 over here, if you can get a 1 over say 1 over, this point then your circuit testing is done by applying this. So, somehow you should get these 2 things indirectly. So, that is it is same. So, this this 1 is your input pattern a 1 can be applied directly, but D equal to 1 and i equal to 1 D equal to 1, I have to we have to do by second by some other mechanism.

(Refer Slide Time: 43:53)

Time frame expansion method Definitions

Definition 1: Sequential depth of a flip-flop:

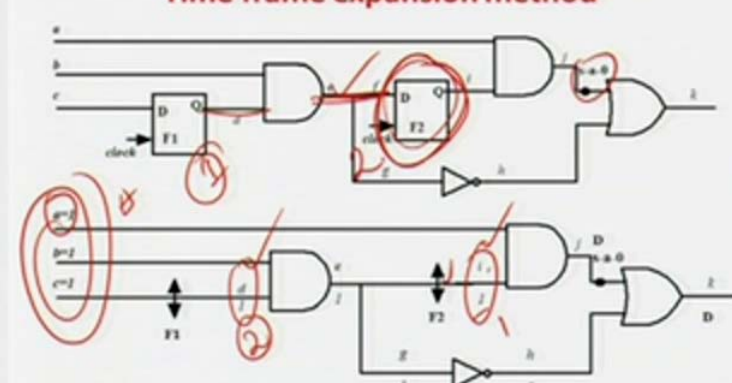
If the output of a flip-flop can be controlled by only primary inputs (and a clock pulse) it has sequential depth of 1. In the circuit of Figure 10, flip-flop F1 has sequential depth of 1 as it is controllable by primary input(s) c . A flip-flop has a sequential depth of d_{seq} if its output is dependent on primary inputs and at least one flip-flop of depth $d_{seq}-1$. In the example, flip-flop F2 is dependent on primary input b and output of flip-flop F1 (via net e); so sequential depth of F2 is 2.



So, that is what is we must required and we will see how to do that by time frame expansion method. Now so, just 2 very simple definition. So, one is call the sequential depth of a flip-flop. So, the output of a flip-flop can controlled by only primary inputs and a clock pulse then the sequential depth of 1.

(Refer Slide Time: 44:12)

Time frame expansion method



- Test pattern to detect the s-a-0 fault as: $a=1, b=1, c=1, d=1, i=1$ and output is D.
- $a=1, b=1, c=1$ can be obtained by directly controlling the primary inputs.
- However, $d=1, i=1$ corresponds to secondary inputs
 - achieved by setting primary inputs appropriately and clock edges

That is just look at this flip-flop. So, it can be controlled directly by the primary input. So, it is depth is 1, now if you look at this flip-flop. So, it depends on what it depends on this input, this input lying e , which is again dependent on this flip-flop, which is having

a. So, this flip-flop is dependent on sorry, so this flip-flop actually is dependent on this input correct and this input is dependent the output and our flip-flop. So, and this is having a sequential depth of 1, because this is directly contribute with the primary input. So, depth of this 1 will be equal to 2.

(Refer Slide Time: 44:47)

Time frame expansion method
Definitions

Definition 1: Sequential depth of a flip-flop:

If the output of a flip-flop can be controlled by only primary inputs (and a clock pulse) it has sequential depth of 1. In the circuit of Figure 10, flip-flop F1 has sequential depth of 1 as it is controllable by primary input(s) *c*. A flip-flop has a sequential depth of d_{seq} if its output is dependent on primary inputs and at least one flip-flop of depth $d_{seq}-1$. In the example, flip-flop F2 is dependent on primary input *b* and output of flip-flop F1 (via net *e*); so sequential depth of F2 is 2.

NPTEL

So, what is the definition just, if you look at it. So, if the output of a sequential flip-flop is direct, this 1 is, this is 1 then the circuit of figure last figure, if you take then it is a sequential depth of 1 and it is directly controlled by *c* that is what so, directly controlled by *c*, so, depth is 1. Now, this flip-flop, this is saying that a flip-flop sequential depth these sequence, if it output is dependent on primary inputs and at least 1 flip-flop of depth n minus 1.


So, that is you have a flip-flop here. So, this input is dependent on a flip-flop depth of these sequence minus 1 and some other primary inputs it can be there. So, that is indirectly, this is a flip-flop, whose input is dependent on another flip-flop, whose sequential depth is these sequence minus 1 then you have to add a 1, for this flip-flop then it will be these sequence.

(Refer Slide Time: 45:32)

Time frame expansion method Definitions

Definiation2: Non-cyclic Circuit

A sequential circuit is non-cyclic if there is no flip-flop whose input is dependent on its output. For example, in the circuit, input of F1 is dependent on only primary inputs. On the other hand, input of F2 is dependent on primary inputs (net *b*) as well as the output of F1 (net *d*). So the circuit non-cyclic



So, that is what so, first sequence you consider with the dependent directly primary input, that will be value 1, then you find out the next set of flip-flop, which is dependent on primary inputs and all those flip-flops whose sequential depth is 1. So, their their value will be 2 and you can keep on doing this.

(Refer Slide Time: 45:47)

Time frame expansion method

- Test pattern to detect the s-a-0 fault as: $a=1, b=1, c=1, d=1, i=1$ and output is D.
- $a=1, b=1, c=1$ can be obtained by directly controlling the primary inputs.
- However, $d=1, i=1$ corresponds to secondary inputs
 - achieved by setting primary inputs appropriately and clock edges


So, in our example, this is having a depth 1 and this having a depth 2. So, that is depth definition, we are got right.

(Refer Slide Time: 45:52)

Time frame expansion method
Definitions

Definiation2: Non-cyclic Circuit

A sequential circuit is non-cyclic if there is no flip-flop whose input is dependent on its output. For example, in the circuit, input of F1 is dependent on only primary inputs. On the other hand, input of F2 is dependent on primary inputs (net *b*) as well as the output of F1 (net *d*). So the circuit non-cyclic


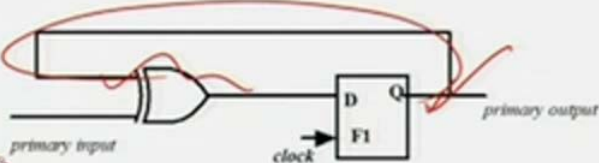


So, now this is the definition of cyclic cycle circuit. So, what is the cyclic. So, this is definition says that a sequential circuit is non cyclic, if there is no flip-flop, whose primary input is dependent input is dependent on its output.

(Refer Slide Time: 46:05)

Time frame expansion method
Definitions

In this case input of F1 is dependent on its own output (and primary input). It may be noted that in cyclic circuits we may not be able to control the secondary inputs; this concept will be elaborated in an exercise problem. A flip-flop whose input is dependent on its own output is called a cyclic flip-flop.



For example, if this consider this circuit is the sequential circuit, because its output is dependent on it is this one, you just look at the feedback. So, it is output is dependent on it is owning. So, that is actually a cyclic definition, but if you look at this flip-flop, this is not a cyclic, this is not a cyclic this one, because this input is not dependent on this.


So, that is a definition of a sequential what do you call cyclic circuit. So, non cyclic circuit means, the output or the input of the flip-flop should not depend on it is output this one the circuit is not definition. So, this is the cyclic circuit and the other example is a non cyclic circuit.

(Refer Slide Time: 46:44)

Time frame expansion method
Property

Property 1. The secondary inputs of a cycle free sequential circuit of depth d_{seq} can be brought to controllable value is at most d_{seq} primary input patterns and clock pulses.

Proof Idea: The proof is obvious. All flip-flops with $d_{seq}=1$ can be controlled by setting primary inputs and a clock pulse. Now, as all flip-flops with $d_{seq}=1$ have been set, we can control flip-flops with $d_{seq}=2$ by setting primary inputs and a clock pulse. In this order, if there are flip-flops with $d_{seq}=n$, we require at most n primary input patterns and n clock pulses.

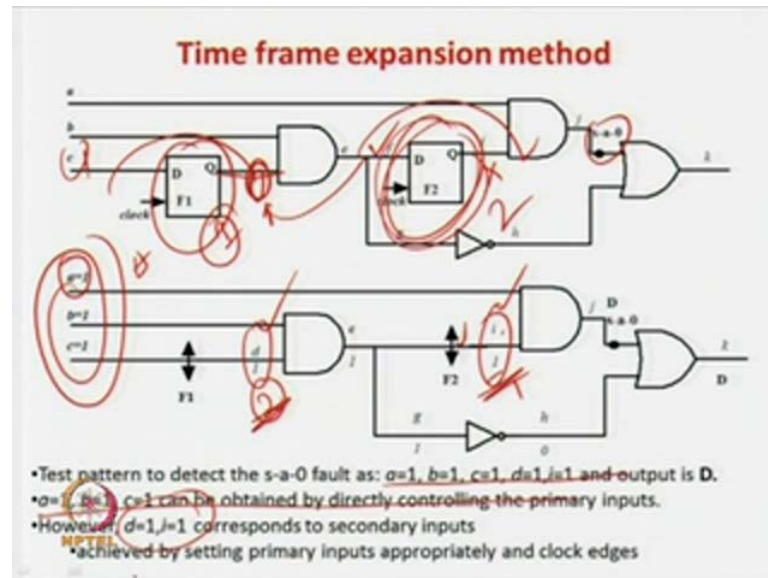


So, now we are going to very quickly see a property, it is a time frame expansion the property, first property the secondary output if if the circuit is cycle free, if the cycle is having a a, if the cycle is having a if the cycle is having a or what you call, if the circuit is having a cycle then we will see in the in that no testing can be done or it is very difficult to do testing for that kind of a circuit. Because, it will be there be continuing a oscillations like in this case the output is depending own input the input is depending own output and there will be oscillation and you cannot control any (()).

So, circuit falls cannot be tested in that circuit, it is very difficult to do that, but we will see the, but for the circuit there is no such cyclic property then what we can do is that, we can control all the virtual primary inputs and observe all the virtual primary outputs.

That is the property is saying the secondary outputs, that is your virtual primary input sorry, the secondary input that is your virtual primary inputs of a cycle free sequential circuit depth this one, can be brought under control in at most like this one, in the sequence number of primary input clock pulses. So, what we have seen in the last 3 example that, we require some nets like in this case, if you look at it.

(Refer Slide Time: 47:48)



So, we require to control this to one and also this to one. So, we now can analyze that how many close pulses and patterns will be require to do this the property says that, if the maximum depth is 2. So, you can do with the 2 clock pulses and 2 inputs, now how it can be done, it is very simple the idea is very simple, because this is sequential depth 1. So, sequential depth once means what it is directly directed to a primary input or a set of primary input. So, give some whatever pattern is required at the output this.

In this case it is a 1, so you can apply a 1 in c and 1 clock pulse will directly setting, that is in other words all the what you can call all the such flip-flop, which are in sequential depth 1 can be directly controlled by setting the primary inputs and apply the clock pulse, because the directly dependent on the on the primary inputs.

Now, after 1 clock pulse and 1 pattern all the sequences, all the flip-flops of sequential depth 1 has been set, now you can consider for the flip-flops in sequential depth 2. So, now, we know that after 1 clock pulse 1 pattern all the flops having a sequential depth 1 has been set, now flip-flops will sequential depth to a dependent on 1 primary inputs and output of flip-flops, which are sequential depth 1.

So, now, indirectly you can say that, 1 more pattern and 1 more clock pulse will setting, because after one pattern, this ghi is set, because this sequential depth 1 and this ghi is dependent on primary inputs and only on this output, that is only on flip-flops of sequential depth is 1, because this sequential depth is 2. So, now, you can apply another


clock pulse and this value will be the output. So, by second clock pulse all the flip-flops having a cycle all the flip-flops having a what, you call sequential depth 2 will be set.

(Refer Slide Time: 49:26)

Time frame expansion method
Property

Property 1. The secondary inputs of a cycle free sequential circuit of depth d_{seq} can be brought to controllable value is at most d_{seq} primary input patterns and clock pulses.

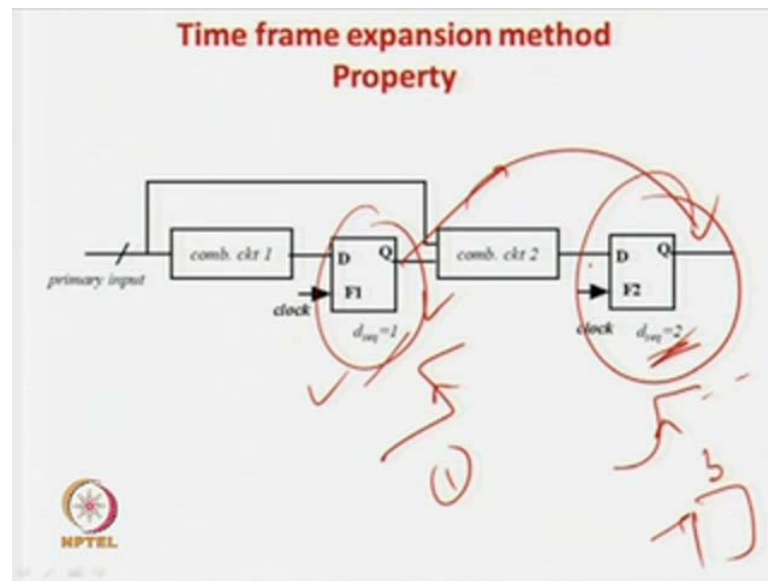
Proof Idea: The proof is obvious. All flip-flops with $d_{seq}=1$ can be controlled by setting primary inputs and a clock pulse. Now, as all flip-flops with $d_{seq}=1$ have been set, we can control flip-flops with $d_{seq}=2$ by setting primary inputs and a clock pulse. In this order, if there are flip-flops with $d_{seq}=n$, we require at most n primary input patterns and n clock pulses.


NPTEL

Similarly, if you have a sequential depth flop with 3 then, you have to have a third clock pulse, because by second clock pulse all the primary inputs, you can set it and all the flip-flops having sequential depth to have been set by the second clock pulse. Now, in the third clock pulse all the flip-flops or sequential elements with depth, he will be said because in the second pulse all it inputs are ready, because inputs our flop with the sequential depth c are ghi are flops with sequential depth through, which has already been said in the clock pulse tool. So, another clock pulse you apply your third level flops or sequential depth third clock will be said and so on.

So, the prove is very obvious which we have discuss and we can see this pictorially, this way these the sequential circuit of depth 1. So, the complex circuit so, primary clock, we apply, apply a clock this is set for the depth 2. Now in depth 1 (()) has been set, now in the depth 2, you require another clock pulse, because now this, because in first phase all this is set and incase of second in case of the clocks, we depth level of 2 all the flip-flops depth 1 has already has been set. So, you just apply another clock pulse and this output will be control.

(Refer Slide Time: 50:04)



Similarly, dot dot dot now, if there is another flip-flop whose depth is what you required the third pulse, because of this second flip-flop all the flip-flops of depth 2 have been set and flip-flops with level 3 are dependent on the output of the flip-flops with gate 2.

(Refer Slide Time: 50:51)

Time frame expansion method Property

Property 1. The secondary inputs of a cycle free sequential circuit of depth d_{seq} can be brought to controllable value is at most d_{seq} primary input patterns and clock pulses.

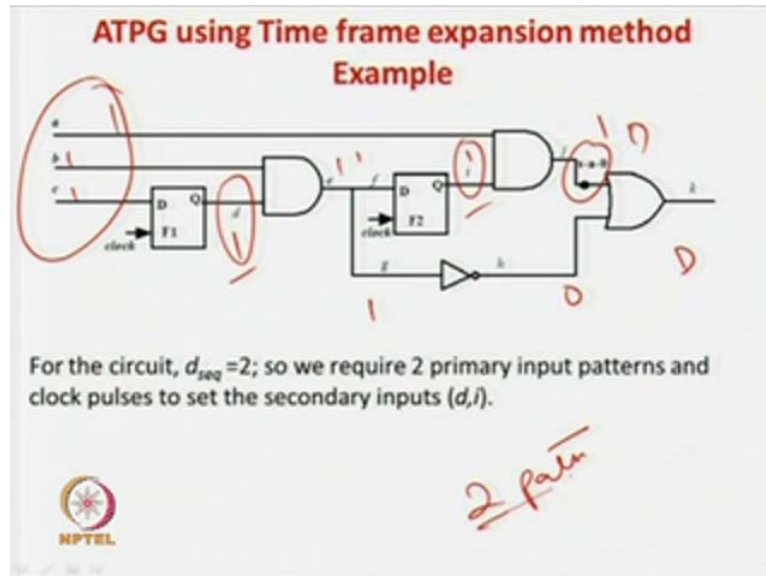
Proof Idea: The proof is obvious. All flip-flops with $d_{seq}=1$ can be controlled by setting primary inputs and a clock pulse. Now, as all flip-flops with $d_{seq}=1$ have been set, we can control flip-flops with $d_{seq}=2$ by setting primary inputs and a clock pulse. In this order, if there are flip-flops with $d_{seq}=n$, we require at most n primary input patterns and n clock pulses.

The NPTEL logo is visible in the bottom left corner.

So, third clock pulse is required so, very obviously, it is said that the secondary input of cycle pulse sequential circuits of depth, this can be brought under control in at most D sequential primary inputs and a clock pattern. So, therefore, if you have a cycle circuit with 10 flip-flops having sequential depth 10. So, you required 10 clock pulses and

10 patterns to set the virtual primary inputs and in the 11th period, you required to give the primary inputs and that will solve your problem.

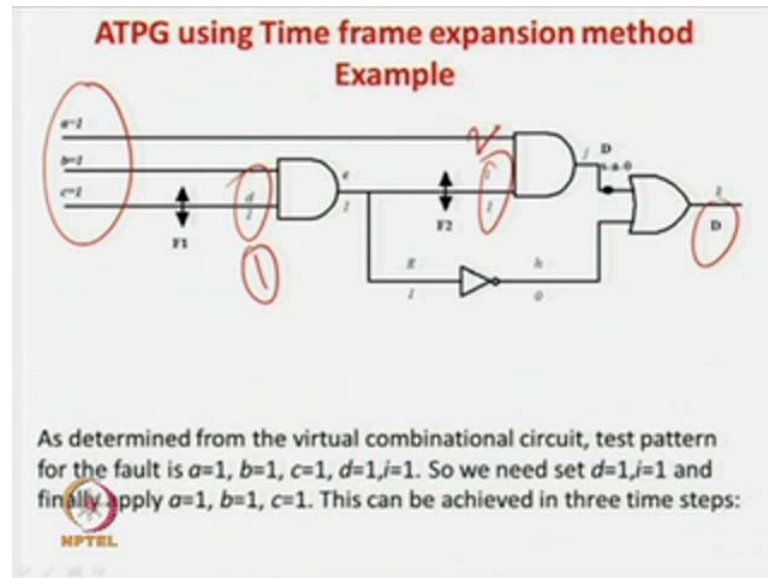
(Refer Slide Time: 51:16)



So, again coming back with this property will see, how this time frame expansion method can be applied to the circuit. So, you know that this is a stuck at 0 fault over there, sequential depth was 2. So, here we require 2 patterns 2 bring, we are already seen that, we require a 1 over here, we require 1 over here correct to do this testing, because we are going to apply a 1 and this will be D this will be d, we require a 0 over here, we require 1 over here, we require 1 over here, 1 1 this is also required 1 over here, 1 over here, 1 over here, something like this, I have already seen.

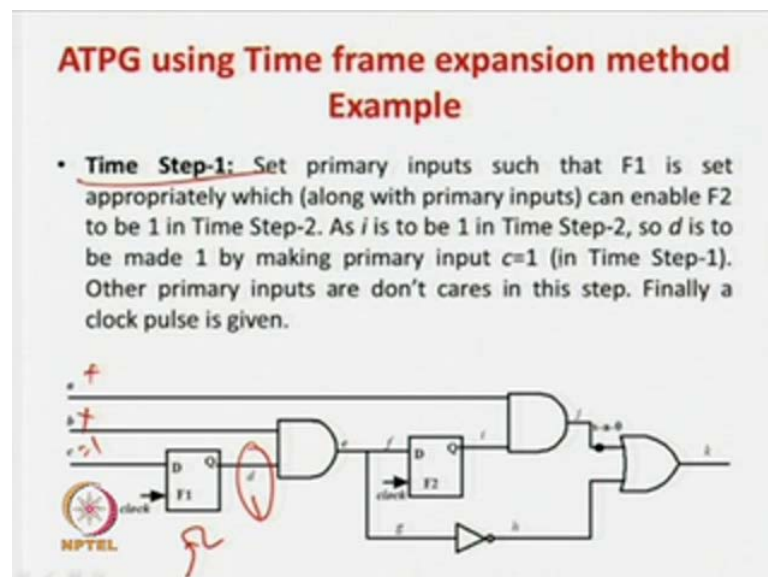
So, this we need to control to 1 and this, we equal to control to 1. So, and sequential depth is 2. So, we required 2 clock periods to make this 1 to 1 and this 1 to 1 and finally, we have to apply a equal to 1 b equal to 1 and c equal to 1 to test the for then (()) explicitly, how it can be done.

(Refer Slide Time: 52:01)



So, two patterns to set and the third pattern to test it. So, what we do, so first these what is the case. So, you require a this is the test pattern as I told you require a D equal to 1, i equal to 1 to do this testing. So, we require to control this at level 1 clock, because this level 1 flow and this will require a do the second clock pulse, because is the level 2 clock. So, this what is required.

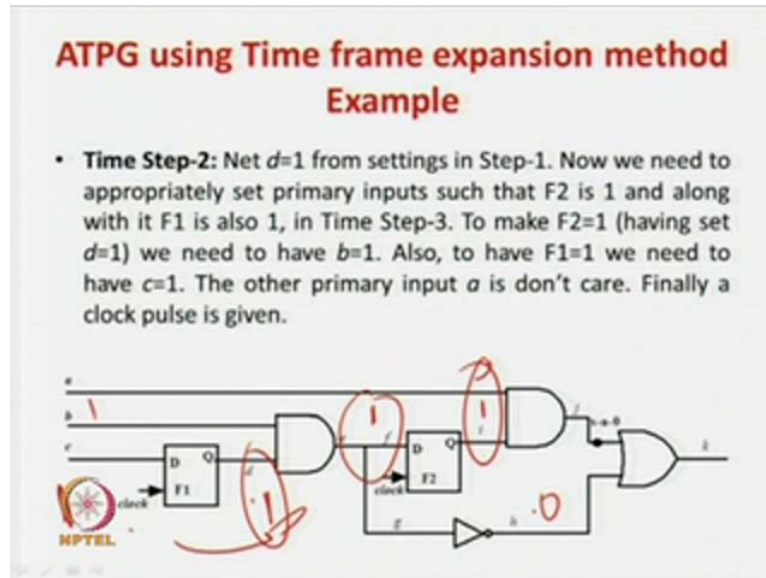
(Refer Slide Time: 52:22)



Now, you see this is time step 1. So, how to do that so, time step 1 what we do, we require a 1 over here. So, what we do we make c equal to 1 (()) do not require at this set

all and we apply a clock pulse. So, what is going to happen, we are going to get D equal to n. So, these you can think that your clock pulse is set right. So, that we have done in the step 1, other primary inputs are do not care finally, a clock pulse is given c equal to 1 is required that is shown we do this. So, you get a 1 over here and D is done.

(Refer Slide Time: 52:52)



Now, (()) now so, what is happening now so, you can know that now D equal to 1 already and this x, this is x we require 1 over here, this is very important that is second case v 1. So, this is require 1, this is already done now, now what we have to do. So, now, we require a 1 over here and if you apply a clock pulse then we can get it 1 over here, because this already done in the last phase. So, now, what we have to get a 1 over here what we do need, we need a 1 over here and we need a 1 over here.

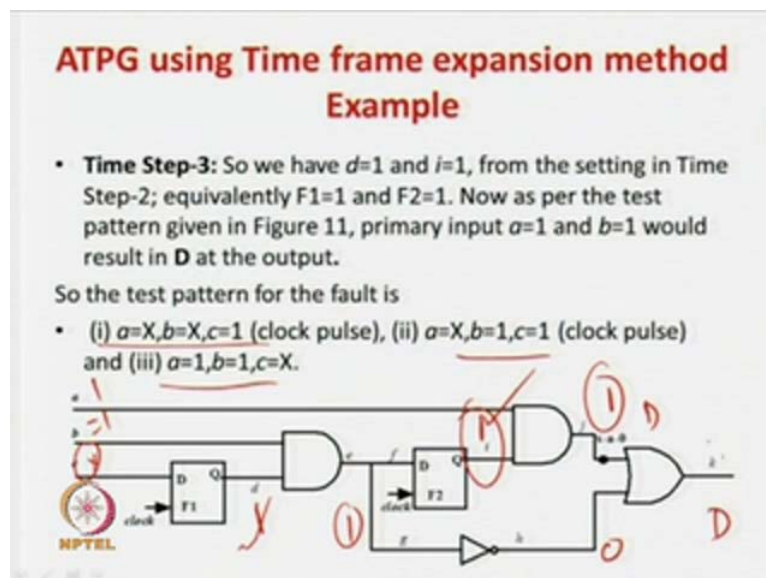
So, 1 over here is already done in the last step. So, if we apply a 1 and 1 over here and we apply the second clock h then what is happen this 1 will be reflected over here, but one more point, we have to remember that this this 1 and 1 will be reflected over here, but the same clock is going here and same clock is going here. So, it it is not that that, if you want to transfer this from here to apply, you apply a clock over here and the clock not coming here, the clock will also be coming over here. So, it will be very careful that when this 1, this one and this 1, together with this one is transferred over here.

So, this should remain a 1, because in the end of second phase, we require 1 over here and 1 over here. So, you apply a 1 over here. So, what we have done. So, indirectly what

we have done over here, this is a first phase, we have a 1 over here, in the second clock pulse, we put 1 over here. So, 1 and 1 you get a 1. So, with these as apply the clock pulse, you will get this 1, but you have to very careful that, after the second clock pulse also 1 is to be here. So, you also put c equal to 1. Now, we apply a clock pulse same clock pulse will go both.

So, 1 and 1 1 will be coming over here and we 1. So, this 1 will be maintained over here. So, now, we get after the second clock pulse, we get a 1 here, a 1 here. So, this one will gets 0 over here and 1 over here sorry, 0 over here, that is this what we are any after the second clock h, we get a 1 over here, we get a 1 over here, we get a 1 over here and A 0 over here. So, these that we have got after your second clock pulse.

(Refer Slide Time: 54:48)



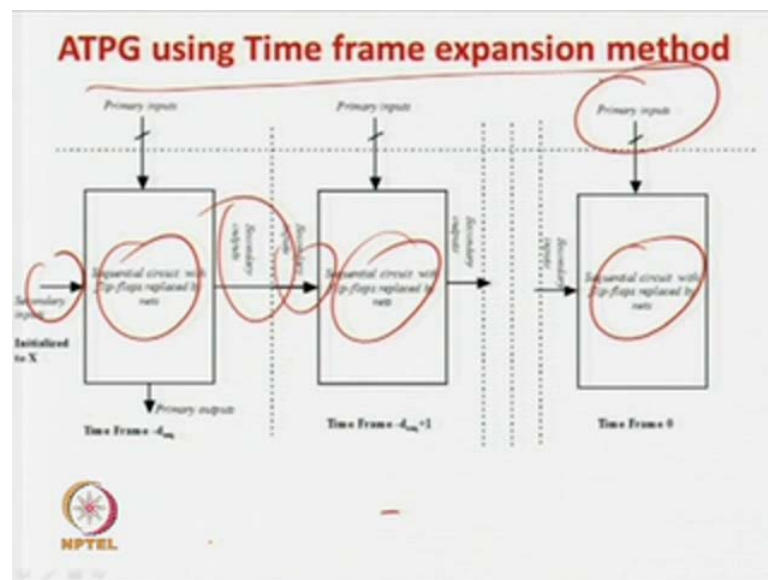
And the time step 3 finally, we have to apply the clock pattern and you will be done. So, you are getting a 1 over here, you are getting getting 1 over here, that is already done in the first 2 step. So, now, what we require, we require 1 over here, that will be get a D then you have to propagate this, this should be 0, this should be a 1 and you can do the testing.

So, to do this what you require, you get a 1 over here, we require 1 over here, that is already set by these are visual virtual primary input. So, you have already said by the clock pulses, you just apply D equal to 1. So, you will get a 1 over here, 0 over here and this is done. So, now, also you require 1 over here to sensodyne the fault, for that this is

1, which you already set using 2 clock pulses, because this was the virtual primary input and a is the primary input. So, you want to 1 here. So, apply a a equal to 1 over here, you get and this is done c equal to x, you do not require this.

So, 3 clock periods. So, first we apply c equal to 1 said this as 1, second clock pulse, we apply b equal to 1 c equal to 1 D equal to 1 and c equal to 1. So, you set this 1 and third clock pulse will apply D equal to 1, a equal to c, do not get and you do the test it. So, if that is what we have done in.

(Refer Slide Time: 55:48)



This is actually called a time frame expansion method, because in 3 clock periods, we have tested the circuit, in the first 2 clock period what we have done, we have said the non primary inputs or the virtual primary inputs or secondary inputs to 1 and 1. And in the third clock pulse, we apply the primary inputs and the this thing has been done.

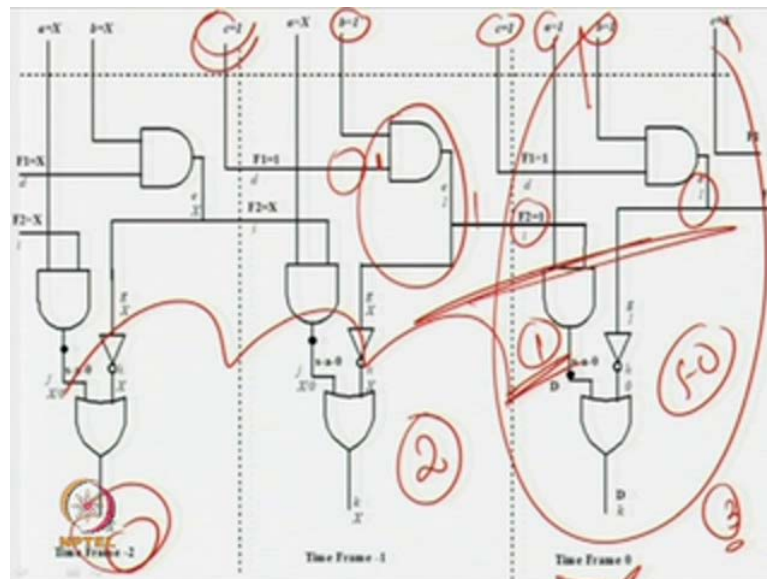
So, if you, if I want to illustrate this in a as a figure. So, these what is being done. So, if there are, if the sequential depth is D then we require 1 time frame, which is which we call minus D then, we actually called minus D plus 1 dot dot dot dot till D. So, in this case, we had a frame expansion depth of 2. So, these was actually, 2 minus 1 is equal to depth 1, then this will be depth 0 and this is actually depth 1 kind of a thing.

So, it is sorry, so our sequential depth was 2. So, it was minus 2 then minus D, D was 2 in this case. So, it was minus 2 then minus 2 plus 1, that is 1 and final it was 0. So,

finally, applied a equal to 1 b equal to 1, which would be testing in this case, we applied c equal to 1, we said the value first output of the first, we flop to 1 and in this case, we applied b equal to 0 and c equal to 0, then it actually said the value of the next flip of 2 1. And finally, we applied this pattern and we get this testing, that is how we we represent this this circuit, I mean this is a way of representing your time frame expansion method, that is we replace the flip-flops with a replace by a n x.

So, flip-flops are eliminated to this what we do and this is your secondary inputs not the primary inputs, this is secondary outputs and these how we represent is your primary inputs and these how we control.

(Refer Slide Time: 57:25)

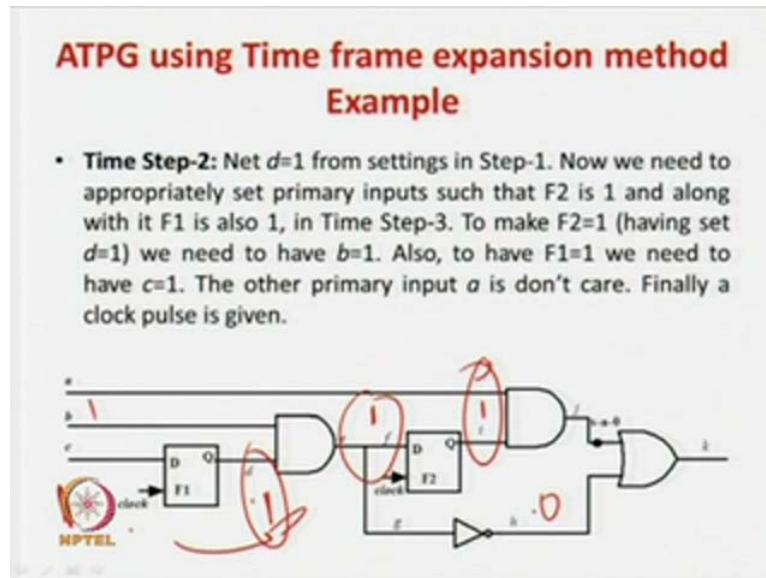


So, now just whatever we discussed in that. So, let us just put it in the time frame method and see what we have done. So, this was your circuit, if you remember, this circuit, this part of the circuit, we have just done drawn in a horizontal with this circuit, we have drawn in a horizontal way sorry, vertical way and this thing had been shorted and this thing has been shorted, that is only thing, we have done.

So, the c this is your flip-flop, these are the way entered. So, let us just mark this S say we can calling it gate number 1 and gate number 2. So, just you have to remember that input of gate 1 is b and D and input of gate number 2 is i and a, but this i is actually, a virtual input and this D is fine and this for this case, D is a virtual input. So, you can just see what we have done, this is say we can think that a see. So, this gate is and this and

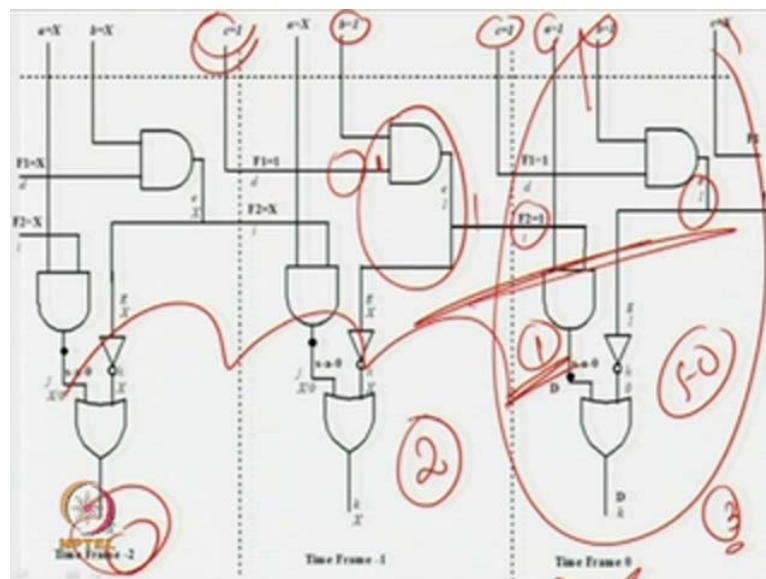
this gate is 1 and this gate is 2 kind of a thing. So, you see 1 gate 1, it is directly connected from a sorry, this is gate 2 and this is gate 1 sorry, this is sorry, this is gate 2 and this is gate 1, this is gate 2, gate 1, gate 2, gate 1.

(Refer Slide Time: 58:38)



So, see just this gate 2 is dependent on a and the output of this second flip-flop. So, this how we represent it.

(Refer Slide Time: 58:47)



So, this gate gate 2, this gate 2 is dependent on a a that is primary input and output of the second flip-flop, that is f 2 and gate 1, if you remember was dependent on b as well as the output of the first flip-flop.

(Refer Slide Time: 59:04)

ATPG using Time frame expansion method Example

- **Time Step-3:** So we have $d=1$ and $i=1$, from the setting in Time Step-2; equivalently $F1=1$ and $F2=1$. Now as per the test pattern given in Figure 11, primary input $a=1$ and $b=1$ would result in D at the output.

So the test pattern for the fault is

- (i) $a=X, b=X, c=1$ (clock pulse), (ii) $a=X, b=1, c=1$ (clock pulse) and (iii) $a=1, b=1, c=X$.

So, this is your gate 2, this is your a gate 2, so which is dependent on b as well as the output of the first flip-flop. So, this is how it is dot. So, now, you can call this 1 2, this can be called as 4, this can be called as 3.

(Refer Slide Time: 59:17)

The diagram illustrates the fault propagation through three time frames:

- Time Frame -2:** Shows the initial state with inputs $a=X, b=X$ and flip-flop outputs $F1=X, F2=X$. The fault is at node $c=1$.
- Time Frame -1:** Shows the fault propagating to node $f=1$ and $F1=1$.
- Time Frame 0:** Shows the fault propagating to node $D=1$ and $F2=1$.

So, this can be called as 4, this can be 3 whatever you like, these are not that much important, these only thing is important. So, gate 1 is dependent on b and the output of the first flip-flop gate 2 is dependent on a n this 1.

So, we write it this way and these are stuck at fault 0. So, now so, we write this is c about that the D algebra, which we have written extended D algebra. So, it is stuck at 0. So, normal fault case, it is 0 normal cases are known.

So, all other things, we have (()) x. So, just this one you can vary easily see. So, the all this things are x. So, only how we have written the circuit is that instead of this gate number 2 and gate number 1. So, instead that this is connected to the output of a flip-flop, we eliminate the flip-flop and we say that f 2 output. Similarly, for gate 1, it is depended on the f 1 output. So, we have said the flip-flop is not there, it is directly connected this 1. So, now, you see what do you require.

So, we require to test this circuit. So, what do you require, you can go to this last level of the circuit, this is 2 and a 1. So, we require a D over here. So, D has to be propagated over here. So, we require a 0 over here, just D 1 got the to 0 means, you require a 1 over here, you require a 1 over here, you require a 1 over here, you require a 1 over here, similarly. So, this is stuck at 0. So, you require a 1 over here. So, you require a 1 over here and you require a 1 over here. So, this is what we are requiring over this.

Now, this this will actually tested circuit. So, this is the last kinds of so a market, now to get a 1 over here. So, this is this is the (()) or the zeroed time step. So, this gate is correct to the output of second flip-flop, so to get. So, this to get this 1, in the output of flip-flop 2 1 primary input can be set at the 0, at time step there is done, but other input 1 has to be come from the output of flip-flop number 2. So, we have to write a output of the 1 from this (()) similarly, for this gate number 1, you require a 1 over here and you require a 1 over here.

So, this is the primary input can be controlled, but the other input, that is c equal to 1, that is and the gate that is actually, also you can know that, it is the output of the flip-flop number 1. So, you have to also know that, the flip-flop number 1 has to be a 1, you make this a 1 at time step 0.

That is in other words, we say that when is c f, that is the out flowing a flip-flop 1, that output has to be a 1, in the previous step to get a 1 in the next step. So, you write that flip-flop 1 is to be a 1 over here, flip-flop 2 is to be a 1 over here, but that is at the time step 0 and the requirement. And when you can achieve it, this flip-flop 2 1 can be achieved a time step minus 1 and this also has to be, I mean achieved at the previous time step.

So, now, in the previous time step, if I want to apply f equal to 1. So, what you have to do we have to make c equal to 1 and apply a (()) first, because the out input of the flip-flop 1 is directly depended on c. So, this can be done very easily by applying c equal to 1, in the previous time step.

Now, but you require flip-flop output 2, output flip-flop number 2 to also to be 1, but how to achieve this, that is some was not very simple, because again the which which 1 of the input is primary input. So, you can directly set, it at time step minus 1, but another input of this end gate, which is deriving flip-flop number 2 is depended on the output of flip-flop number 1. So, that has to be controlled in and that time step, which is 1, I had before.

So, to get 1 at this, which will actually make this flip-flop 1 and the (()) time step. So, what you have to do, we have to get a b equal to 1, in the second step that is 5, but again another input another input, that is actually, the of this end gate number 1 is currently the flip-flop number 1. So, it has to be controlled at the time state number minus 2. So, in minus 2, if you make set a equal to 1. So, what will happen that flip-flop output 1 will be 1, in the (()) 1 minus (()), that is what actually, we are doing.

So, indirectly what we are requiring just very quickly look at this snap shot what we have been done, we require a a 1 over here, a 1 over here, which is the flip-flop number 1. So, to get a 1 over here, we required this end gate to be a 1 by the primary input 1 fine, but another input of the end gate is actually, output of the flip-flop number 1. So, we also have to be a 1 over here. So, (()). So, this is as this f 1 is actually, a flip-flop, which is controlling gate 1.

So, it has to be controlled in the previous step. So, in step minus 1 time step minus 1, we apply c equal to 1 and we get in time step 2 again end gate is there. So, we require a 1 over here, another one is by this primary input a equal to 1.

So, this you can very easily get, in the time step 0 or the third step or the third step, but another input, this input has to be 1, it is the output of flip-flop number 2. So, this has to be achieved that previous times step, the time step minus 1. So, together 1 over here, you require a 1 over here. So, this can be done in the second step or time step done in the second stage, but again this and gate is dependent on flip-flop number 1, which also cannot be done at the current instant.

So, we have to go 1 more stage before and we have to make c equal to 1. So, if we make c equal to 1, in this time step minus 2 and then you make b equal to 1, the time step minus 1 then what you are going to is that, you are going to get a output of 1, in the second stage, which can be directly applied to the third stage. And if you are making c equal to 1, in this and step minus 1, on the time and ah step number 2 and what we are going to get in the third stage, we are going to get a 1 over here.

So, indirectly seen what we have already seen, in the long example before, we are just writing it in a formal way that is just some requirements, the are 1 over here, a 1 over, a one over here, they are the 2 requirements at the time stage number times step 0 or sorry or step 3, this step 3, you require this.

So, this requirement this can be achieved by making b equal to 1 and another input as to be a 1. So, which we can make achieve by making c equal to 1 in times state step 2, you can say or time say minus 1. This requirement of 1, that is again, these are actually, virtual primary outputs sorry, virtual primary inputs, which are direct controlling. So, know again this 1 is required. So, so get this as a 1, we require this 1 is a 1.

So, we can which we can easily get it time frame to a step 3, you can say, but another input to get is the 1, we required at this stage, we will go to this stage and make this a 1. To make this as a 1, we require b equal to 1 in time frame minus 1 and also another input is there, which is know depended on the time frame minus 2, that is actually, because this gate is depended on 1 flip-flop level 2 and 1 flip-flop level 1. So, and if again go back to time frame minus 2 and you have to say c equal to 1.

If you do that then this 1 will also become a 1, in the second stage, we will get a 1 and finally, we are going get your requirement. So, this is nothing, but whatever, we have discussed is nothing but this three stages these three stages, the stage 1 the where you are making c equal to 1 and apply a clock plus that is the 1, this one the next is a making 1

here one here applying here. And the third stage is nothing but applying this one and testing this one, that the 3 stages, which we have done, we are just representing pictorially in this way.

So, these are very formal way of ah this saying describing of time prow expansion method and you can solved your this thing, so that that that about time frame expansion method based testing. So, in this case you have to observed that to test a single stuck at fault what we have to do in case of at ATPG for combination circuit only 1 step would have solved your problem, but in case of sequential circuit, we require much more number of stages. And number of stages are equivalent to nothing, but your number of sequential depth of the free flow.

So, that is why you can say that sequential ATPG is combination ATPG star sequential that, that is why it is a very very complex procedural and you have to also remember that any stage, you may have back tracks leading into a big problem. So, in the next lecture, we are going to see, how we can make this very complex problem a simple 1, but before that lets very quickly see a question.

(Refer Slide Time: 01:07:21)

Questions and Answers

Question
Can ATPG and testing be performed for any fault in the XOR gate of circuit given below

Answer:
The circuit is cyclic as input of F1 depends on its own output. When the circuit starts, output of F1 is X. Now, input of F1 is inversion of the primary input value, if F1 at startup was 1. Similarly, input of F1 is the primary input value if output of F1 at startup was 0. So it is obvious that as X is unknown we cannot set F1 according to need for ATPG. So ATPG and testing cannot be performed for any fault in the XOR gate of the circuit.

So, this is a cyclic circuit and we have not considered any cyclic circuit. So, now, if I ask you that can you test any fault in this circuit. So, the question is can an ATPG and testing the perform for any fault in the so orate bellow, the answer is no why just, we will see see for example, we know that the output is always a x, it was the circuit is starting.

So, we will get the x over here. So, know if we have the primary input as 1. So, your answer will be x, because we know that in case of XOR gate, if the input is 1, the other is inverter and know if you primary input is 0, we are going to get an x. And again this x or x frame will be fad back and this will be loop. So, at no point of time, you can be able to get the value of 0 or 1 over here, if you get the primary input 1 will be explain, if you get a 0, it will be some other thing.

So, at no point of time, you can get any controlled decision or controlled value or together decision over here, because the input is depended, because this input is depended is own out. So, there for as we cannot set the flip-flop, so if we have a circuit like this no point of time, you can convert this x to any other stuck and fault cannot be this thing.

So, this we come to the closing of this lecture and the next lecture, we how we can solve the problem of such a huge number of test patterns required for a sequential stuck at testing. So, we have to somehow reduce the number of test patterns required or the number of sequences required for testing a fault in a sequential circuit.

Thank you.