

Optimization Techniques for Digital VLSI Design
Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Lecture – 18
High-level fault modelling and RTL level Testing

So, welcome to the last lecture on the module on VLSI testing, which is on high level fault modelling and RTL level testing.

(Refer Slide Time: 00:34)

Course Plan

Module 4: VLSI Testing

Lecture 1 and 2: Introduction to Digital VLSI Testing, Automatic Test

Pattern Generation (ATPG), Design for Testability

Lecture 3 and 4: Optimization Techniques for ATPG

Lecture 5: Optimization Techniques for Design for Testability

Lecture 6: High-level fault modeling and RTL level Testing

So, basically if you look at the previous series of lectures; So, we were discussing on different type of optimisation techniques like high level fault models, then how basically one type of high level fault model can actually go for coverage of stuck at fault model. Then we also have discussed very interesting theory on how we can compress the test patterns and then inside the circuit we can again decompress it. So, that the test volume remains smaller and then also we have seen some kind of design for optimisation techniques for design for testability, like how the hardware on the chips which are required for testing can be reduced like some kind of scan chain based optimisation etc.

But if you look at all the things basically were emphasizing on gate level design, that we know that structural fault model structural testing with the fault model is 1 of the defect to standard because, nobody could handle the complexity of functional testing. So, but

everywhere the concept was till date that we are going for a structural test with the fault model and everything was boiling down to a gate level.

So, now why gate level because I mean till the advent of some kind of suffix very sophisticated circuits like NOC's and SOC's more or less we could handle VLSI circuits testing and design etc, with quite of good level of optimisation and timing requirements or you can say about the cost required to implement the algorithms, time required to generate the test patterns time required to do the verification with quite reasonable at the gate level. But when this NOC's and SOC's have come into picture even the gate level with structural fault model has started blowing up.

So, today's lectures we will be basically concentrating on a the directions what people are thinking that how we have to move away from gate level, all analysis from gate level modelling to some level of abstraction like register transfer level. So, basic the motivation I mean before these NOC's and SOC's type of sophisticated designs, where we were looking mainly at the gate level.

Now why at the gate level because gate level basically has a good correlation with so called the transistor level and the basic device level and when you talk about structure basically we do not think about the functionality of the circuit and structure means basically if you tell me what is the structure of a digital circuit we will all tell that it is basically gates and interconnects and when you if somebody tells about what this structure of analogue circuit you will tell about transistors.

So, 2 for as basically the basic atomic level for digital design were gates, so if you and statistically it was also verified that if you can handle the fault models of the gate level, you can have a very very good correlation with the defects at the transistor or the silicon level. So, that is why we were all thinking in terms of the gate level design a gate level testing gate level verification etc. But when this number actually move to NOC's and SOC's gate level the number of gates and interconnects became so high that not even the structural test with most advanced optimisation techniques could give test pattern generation in a reasonable amount of time.

So, we had to go away from this lower level to higher level abstraction and. In fact, this is 1 optimisation basically which leads to slightly a quality of compromise in the quality of test patterns, but nothing we can do because if we start going for gate level based

testing in NOC's and SOC's the amount of test time generation will be in years and months basically which will give a very invisible way of solving the problem. So, to address it address the higher level of complexity we have to trade off slight level of quality of service to achieve a reasonable timing.

So, today's lecture plan is to give you an abstract idea of how people are now moving from gate level to a newer abstraction level, this is actually a paradigm shift and this has been started in last 5 to 6 years the major emphasis has been going from the gate level to more abstraction level.

(Refer Slide Time: 04:16)

Structural Testing with Fault Models

- Structural testing with fault models involves verifying each unit (gate and flip flop) is free from faults of the fault model.
- Fault model is an abstraction of the real defects in the silicon such that
 - the faults of the model are easy to represent
 - should ensure that if one verifies that no faults of the model are in the circuit, quality of test solution is maintained.

So, let us go about it, so till now what was the defect to standard that we have to go with structural test and fault models and what is a fault model fault model basically abstracts out the real defects, that is one option that is one idea and the second idea it should be very much correlated with the absolute defect level like you cannot have a very abstract fault model which has no correlation with the defect level.

The idea is I have repeating many times, but that is one thing which is to go into everybody's mind there are fault models to simplify testing, but at the same time it should have a good correlation with the defect level, so that you actual the quality of test solution is maintained.

(Refer Slide Time: 04:53)

Complexity of Structural Test

Stuck at Fault Model:

- Twice the number of nets ✓
- **Backtracking (propagation and justification at fault site)**

Bridging Fault Model:

- Number of nets C_2
- Backtracking (propagation and justification at fault site)
- **Additional Backtracking (Control at dominating point)**

~~Bridging Fault Model:~~ *At Speed Delay for*

- Complexity same as stuck at fault model
- Cost for at speed testing

But now if you look at the complexity of structural test from the view of VLSI circuits, which is slightly less complex than the NOC's and SOC's; So, if you go for stuck at fault model which is the defect to standard used everywhere. So, we know that generally it is twice the number of nets because there can be stuck at 0 and stuck at 1 so and in fact that there is lot of concepts like fault collapsing one pattern can detect multiple patterns. So, actually the number of faults actually comes down then the twice the number of nets it is much lower than that.

But what is the problem that is actually the backtracking. So, what happens already we have discussed that if you sensitised the fault, then you have to propagate the value to the output and then you have also justify it. So, by doing it many times you find out that 1 net should be 0 because for propagation and for justification it needs to be 1, so there is a conflict. So, conflict has to be resolved and you have to keep on doing it then as many paths from the fault setter possible to propagate, the value to the output till either the fault is successfully propagated sensitised propagated and justified or you find out that there is no more path available to do it.

So, the backtracking is the main complexity here, so if you think about very big circuit the number of nets will also be high the number of gates will also be high, of course the number of faults is not of a big concern this you have to remember because, due to fault collapsing by equivalence dominance like the like an and gate if you take an and gate a

stuck at 0 fault at the output and a stuck at 0 fault at the input are equivalent because, always you require a 1 and 1 2 1 one as the input to test it, but the problem is because of backtrack the more larger circuits have you have more larger.

The number of fan outs will be and more number of options will have to send the values of the fault effect to the output and similarly for the justification of the lines. So, more number of choices more number of different values you can put in the signals and more number of conflicts can arise and this has to be done till you get a successful drive to the a successful fault propagation and justification. So, backtracking is the major problem and if you think of a very big circuit it actually kills the timing requirements to go for the automatic test pattern generation, so that backtracking is the big problem apart from the number of faults.

Secondly we are looking at the bridging fault model, so bridging fault model again the number of faults are slightly larger because we take combination of 2. So, number of nets into 2 any two bits can be having a bridging fault, but anyway we try to minimize it or optimise it by making slightly smaller size windows, if we are lines are more congested and where you have you where the designer tells that there can be good chances of having bridging faults, so you can do that.

So, the number of faults are slightly larger again there is a backtracking issue because, already we have seen in the last classes that bridging fault is similar to a testing of our stuck at fault, but say for one line you go for stuck at fault testing; but some other line which 2 are couple you have a constraint that where the whether when you test the stuck at 0 fault over here may be this line you have to make one or something like that. Already we have seen define type of bridging fault model like and gate fault model and a or gate bridging fault model. So, the idea there we have seen that one line of the bridging fault has to be tested for stuck at 0 and stuck at 1, at the same time you should also have a constraint at the other line which is involved in coupling.

So, more constraints you put more conflicts you will have and additional backtrack will be required, so this control has the dominating points. So, 1 point is getting dominated where you are going for test the stuck at 0 and stuck at 1 fault, but the other line which is dominating it has to be put to a constraint; like as I told you in this case may be you are taking checking for stuck at 0 here you have to keep a 1. So, more number of such

control points you have more complexity stuffs have because of the backtrack. So, the main issue is are all happening in such cases because, of the backtrack issue.

So, if you have a very big circuit lot of choices and the complexity become exponentially prohibitive at because, looking at all path is an exponential problem even if the heuristic to find out with they are good path for propagating the fault effect etc. Still for a very big circuit like NOC and SOC such type of algorithms are prohibitively, prohibitably complex and you not get the test patterns in a reasonable amount of time also we have seen sorry, this is about the at speed test model. So, pardon it is at speed or delay fault it is about delay fault.

So, in delay fault actually basically this is the delay fault so. In fact, the delay fault basically is very similar to the stuck at fault complexity because, you have to just verify the in one line you have to find out whether it raises properly or it falls properly. So, there is no more additional constraint, but the only issue is that you have to apply at speed the value has to be applied at speed. So, it does not involve any kind of computational complexity as such, only thing is that you have should have a higher value higher speed ATE or sometimes you have to put the signals inside the built in self test mode on chip circuit which we will apply. So, that complexity you have already seen how it can be handled by DFT circuit.

So, till now. So, what we have gathered is the major problem of ATPG happens because, of backtracking and backtracking and of course, the secondary factor is the number of faults because that the number of nets increase. But the idea here is that if the circuit becomes very large the backtracking is becoming; So, complex that we have to from this and we have to go for some kind of abstraction.

(Refer Slide Time: 10:00)

High-level fault modeling and RTL based Testing

- One of the most important issues of testing in modern deep sub-micron design is scalability.
- This is because test schemes are designed at logic gate level leading to the state explosion problem.
- In order to solve this issue, i.e., to improve the scalability, test schemes need to be developed at higher description level e.g., Register Transfer Level, Architecture Level etc.
- High level fault models for RTL based testing .

So, then what actually came is nowadays called abstraction, so you cannot go to very low level like a gate level like gate or transition you have to go more away from it. So, 1 of the most important issues of testing in deep sub micron or very large NOC's and SOC's scalability this is because all the test schemes till last 20 to 30 years with all concentrating on gate levels and they were trying to optimise it which actually makes the problem exponential because, if you think that I have a 2 numbers to add. So, they are 32 bit numbers I have 2 adders.

So, may be the complexities around something of 30 order of 32, but then I make the numbers as 64 bit then eventually the complexity becomes enormously high. Then if can have a more higher precision numbers and the number may be say 128 bit. So, the data path actually the data width keeps on becoming increased increasing in nature because, of the precision etc and the complexity of testing becomes very high.

But if you think of some kind of abstraction that will consider these 2 32 bit numbers or 64 numbers as a single number then a plus b or a minus b you have to test then whether is a 64 bit number, whether it is a 32 bit number or whether it is 1028 bit number does not matter. So, this one level of abstraction, but where will where we are compromising we are not going for bit level testing and already statistic. So, for the years have verified that if you are going for gate level testing that is a bit level testing with fault model, then the accuracy or correlation is very very high but we do not know.

But exactly what is the correlation or how high it is if you are going for such kind of abstraction, that is instead of I mean considering a and b as bit level numbers, we are considering a and a and b as a whole number together and we are not going into the depth of the bit level. So, that nobody knows, but still that is the only way to solve the problem because jumping from 4 bit to 8 bit 8 bit to 32 bit and from 128 bit such numbers, the problem of testing the adder will become so complex in the structural level that ATPG solution will not be I mean possible within a logical amount of time.

This only about a adder, but think of a circuit with N number of IP codes, where all the 128 processors are running in a single code then you can think that adder is only a very small part of the design. So, it can be a hundreds of order of hundred times the adders or 200 times the size of the adder that complex circuits are there. So, there is no other option then to go for higher levels of abstraction.

So, that is why people try to model the whole test problem in the very high level of abstraction called register transfer level or something called the architectural level. So, you all know about register transfer level, that we have a blocks like adders multipliers subtracters multiplexers registers instead of having flip flops and gates and of course if you are modelling at such an high level of the circuit of course, your test fault model also should be at a very higher level of abstraction; like the stuck at fault model was very much I mean say consistent or was moving in hand in hand with the gate level because, we are saying that the input and the output of the gates should not have a stuck at 0 stuck at one.

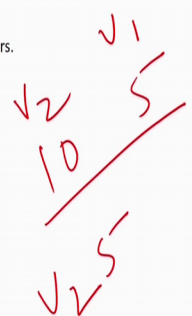
But such concept is very difficult to apply at the higher level of abstraction. So, not only the model is abstracted circuit model is abstracted also you have to have fault models, which are also at the same level of abstraction so that there is a good hand to hand correlation.


(Refer Slide Time: 13:06)

RTL based Modeling

Algorithm of Greatest Common Divisor (GCD) of two unsigned integers.

```
1: Begin
2:  $V_1 \leftarrow IN_1$ 
3:  $V_2 \leftarrow IN_2$ 
4: while  $V_1 \neq V_2$  do
5:   if  $V_1 < V_2$  then
6:      $V_2 \leftarrow V_2 - V_1$ 
7:   else
8:      $V_1 \leftarrow V_1 - V_2$ 
9:   end if
10: end while
11:  $Output \leftarrow V_1$ 
12: End
```





So, before going into all the algorithms we will start with an example, this is a GCD computation which is a very well known basically theory which we learned in high school. So, what we do, basically we take 2 numbers say V_1 and V_2 and we find out the GCD. So, what we do the well known algorithm is that till V_1 is equal to V_2 with we compare V_1 and V_2 and then we say V_1 is equal to V_2 minus V_1 . If V_2 is larger else we do the reverse and we keep on doing it till the numbers are equal and then whatever is the when they become equal the GCD, actually happens in the stored in the output of V_1 we already know that this a very simple algorithm like say we are taking say 10 and 5.

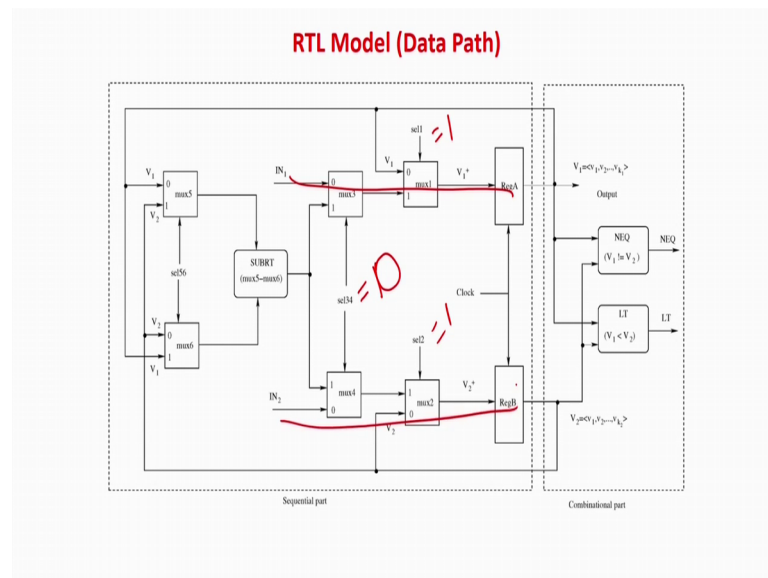
So, first step will subtract 10 and 5, so it will be 5 and this is say V_2 and this is V_1 . So, we will subtract 10 from 5 10 from 5 and then actually V_2 will be equal to 5 then V_2 is equal to V_1 in the next step. So, the GCD is 10 or 5 about 5 and 10 is 5 itself very well known high school mathematics. So, this is the actually algorithm say that we want to implement the circuit out of it.

So, if you look at the gate level it is going to have around say 100 kind of a gate, so implement a circuit because you require a subtracter you require a comparator and everything will depend on what is the size of the numbers in. So, if you are consider 32 numbers the size may be around 100 here 1 thousand gates, but if you consider in 1 and in 2 as 128 bit numbers the size will be extremely high number of gates. So, that will

make it very exponentially very complex to find out the ATPG solutions because, of this huge size of the circuit.

But if you look at the RTL level it is totally in material what is the size of the inputs of size of in 1 and in 2 it looks as a small smaller and nice problem. So, all this things can be possible if you are modelling at this level of abstraction that at the block level, which is called the register transfer level.

(Refer Slide Time: 14:55)



So, I am showing you the basic whole RTL diagram for the circuit, now it will go to see how it works basically and we are going to zoom the individual part of it. So, let me just show one part which is the required to do the computation, which is first part is computing basically NEQ that is whether V1 is equal to V2 and part is the comparator I am just showing you just keep it in mind because, that thing will not be required too much in discussion because for every iteration this a combinational circuit we will just calculate whether V1 is not equal to V2 and another block will compare which is greater.

But again you have to appreciate the size of this 2 blocks will heavily depend on the bit width of the numbers input and input 2, but in this case as you have very much abstracted is an RTL design we have abstracted V1 and V2 as a single number. So, actually if you look at this is actually the data path. So, all circuits have a data path and a control path. So, data path will actually talk about how the numbers are moving what are the arithmetic operations happening and the control path as you know will control that

which part of the multiplexer has to go in whether it will be $V1$ minus $V2$ or $V2$ minus $V1$ how the multiplexing has to be set, so that the control all will do.

So, the controller is always generated a small finite state machine which is smaller in size generally in compact 1 RTL design path. So, if it if the bit width of the numbers increase like in one is 32 bit to 128 bit the size of the data path will be anonymously large in terms of the gates. But the controller will always be the same because controller takes the number like $V1$ is $V2$ $V1$ not equal to $V2$ that is the output of this combinational blocks it will take and give the appropriate signals to control the multiplexers. So, basically even in very large circuits or very very complex circuits we are more concern with the testing of the optimisation for testing of the data path, controller more or less remains similar if you are doing a single bit number or 128 bit number.

So, I am not saying that it is very simple to test the data path, control path rather but actually the with the increase in the complexity of circuit the data path problem become it is much much higher or more involved in the number of gates compared to the controller path. So, basically will in this work and most other works people always try to optimise the data path mainly, but again in this work we will show you both.

So, let us try to look at the first part so basically what happens sorry let me just concentrate on this. So, there is one multiplexer mux 3 and 1 multiplexer is mux 4. So, what we so this signal which is called select 34 this is some signal the controller will give the value. So, it will be first made 0 so that the in one goes to mux 3 and in 2 goes to mux 4, then you make the second line if you see select one also this is select one will be equal to 1 and select 2 also will will be equal to 1. So, in this case what I have told you that select 3 4 is equal to 0, so that the in 1 and in 2 will be going to the mux respective mux s then the select 1 and select 2 will be equal to it will be equal to 1, so that these value this in 1 will reaches.

So, just as you look so what I was saying that 32 we are making with a select 34 equal to 0 sel 1 sel 2 mux values and making it to 1, so that input will go via this and will be registered will be locked into the register; that means, now we are reading the value of IN 1 and IN 2 in 2 registers that is REG A and REG B. Again telling notifying it is very important to understand that when you are talking about the register transfer level

modelling, the REG 1 and REG 2 are not single flip flops they are actually register back; that means, if A IN an IN2 IN1 and in 2 are 32 bit numbers, then the register a and register b will be 32 bits or they will have 32 bit registers, but if it is 128 bit REG A and REG B will have 128 and 128 bit 128 different flip flops to last the values.

So, but here we are modelling it as a single block that is the duty of abstraction in the register transfer limit, now what we are losing here we will be losing the deep level value or it is of the test, we will be testing register A as a whole we will be testing register B as a whole, but we may not be able to test the at the very granular level that the bit level of all the flip flops involved in the register A and B, but again the complexity is coming down from 128 bit to 1 bit. But again the this backtracking problem is an exponential problem order is exponential because, you have multiple number of path from 1 node to the another node and all path combinations is exponential in the number of nets and which is obviously, very proportional to the number of gates.

So, if you reduce the number of flip flops or gates from 128 to 1 obvious the complexity drastically comes down exponentially. So, that is why the RTL level actually models everything in terms of blocks. So, that they can huge the bring down the complexity level in the drastically the level is brought down. So, that the ATPG etc becomes very very simple not of but here actually as I told you the slight compromise in quality because, we cannot test at the granular level and all the statistics that good correlation between stuck at faults at gate level real defects exist for the gate level.

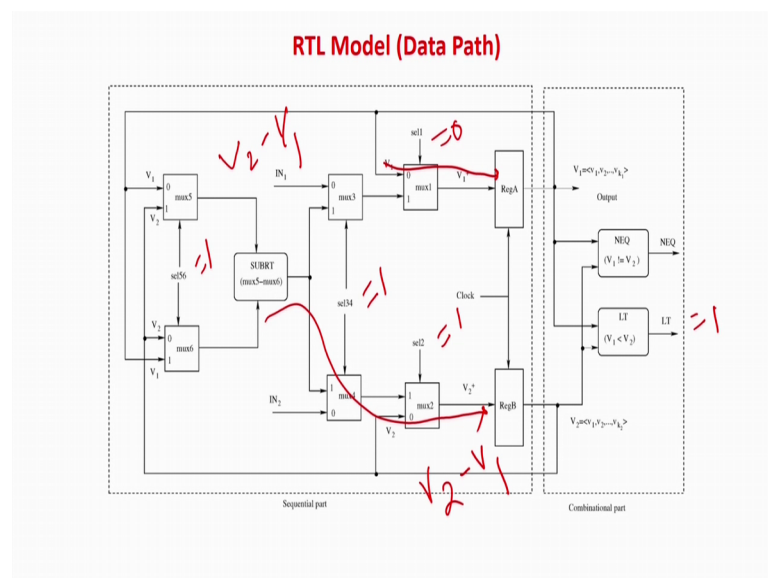
But for the RTL level which is a very new concept people are still struggling to find out what is the complexity what is the correlation? Obviously, complexity is very low but what is the correlation with real defects people have found out it is not bad it is quite good in fact, but still it is a currently under study as statistical data is still being still being studied and gathered to find out very good quality RTL fault models anyway coming back to the our discussion. So, on the first step basically as we have seen the in 1 and in 2 will be lashed at the 2 registers which is nothing basically this in 1 and in 2 will be registered at V1 and V 2. So, v register 1 actually have V1 and register actually have V 2, so this type is done.

So, this is standard algorithm, but I am just going to give an highlight, so that you understand what is basically register transfer 1 then what we will do then as I told you

once this values V1 and V2 are latched this NEQ and less than equal to this 2 values will be fed to the controller. So, if V1 is not equal to V2 that the loop has to go, but if the if basically this becomes sequel and the data will be latched from register RA.

So, whenever this one is equal to 0, that means they are not equal to become 0; that means, they are equal in that case the output guy who is going to take the output he will capture the value from register 1 that is the end of iteration, but anyway before that let us assume that the lt becomes 1, that means V2 is greater than V1. So, if V2 is greater than V1 then what should happen basically if V2 is greater than v1 then we should have we should have the operation called V2 minus V1 So, if we should have the operation called V2 minus V1 we are assuming that this less than this is less than equal to is a 1.

(Refer Slide Time: 21:22)



So, basically that is V1 is less than V2 this value is equal to one; that means, V2 is greater than v1. So, this operations would happen so how this operation would happen? So, the controller will get the value of 1 less than equal to equal to 1, so that means, it knows that V2 is greater than V1. So, you have to arrange the multiplexers in such a manner that it happens basically. So, you can see this is the subtracter which will take the value from mux file and it will subtract the value from mux 6.

So, I will the control will make 5 6 equal to 1 the value of this control will be equal to 1. So, let us see what happens basically. So, if it happens then basically here 1 means the

V2 will be coming sorry the V2 will be coming out from this guy to here and also this 1 is connected. So, is also equal to 1, so V1 will be connected from here to here it is V1.

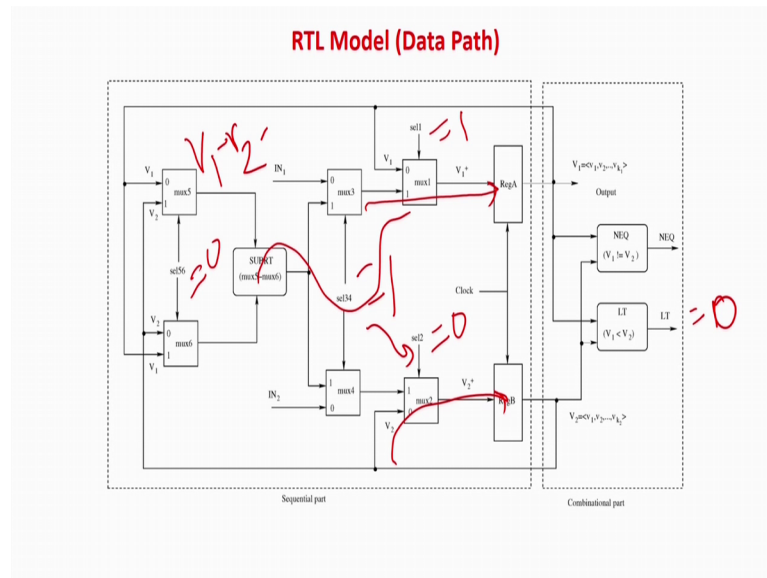
So, in subtracter you will have the value of V2 minus V1 if I make 5 6 equal to 1, now who will make it the controller will make it how the controller will decide it will decide because the 1 t value is equal to 1 this block is saying that is the case. Now what now again V 2 minus V1 will be latched to register 2 because v V2 minus V1 will be registered to V 1 because if you look at the algorithm, so V2 minus V1 will be latched to v 2, so now again I ma coming back. So, now again the multiplier I mean the multiplexer arrangement will be something like. So, this part I should not do anything basically. So, select will be equal to select 1 will be equal to 0 and this 1 will be equal to 1.

Now let us see and of course, this value will be equal to 1 now let us see how it happens. So, if you look at so now this 1 is having the value of V2 minus V1 select 4 equal to 1 I have made equal to 1. So, basically in fact from both this transistors both this multiplexers 3 and 4 I am sending the value of V2 minus V1. Now basically output of mux 3 and mux 4 both is now having the value of V2 minus v one. So, now you see register a that is equal to V1 should remain constant now why it should remain constant because I am doing V2 minus V1 so should retain.

So, I am making this select 1 equal to 0 control will make it because 1 t is equal to one. So, in that case select 1 will be equal to zero. So, old value of V2 V1 will be fed back which is constraint. So, V1 will remain as it is, but in this case if I make select equal to 1 mux 4 output will go mux 4 is having the value of subtracted output is V2 minus V1, so V2 will now have the value of V2 minus V1. So basically this 1 will be coming over here you will going to have the value of V2 minus V1 and they are the old value of V2 V1 will be there and it will be restrained and it will have the old value.

Similarly, whenever the value of 0 the it will be 0 in the other case, so in this case just the reverse thing is going to happen in this case basically you should have the value sorry. So, in this case you will have the have value of 0 , so in this case basically it will be V1 minus V2 ok, this 1 will be have to be made 1.

(Refer Slide Time: 24:23)



So, basically V_1 minus V_2 will be flowing to both these parts in this case just the reverse has to be done select should be 0. So, if it 0 the old value of V_2 will be fed over here, but in this case select 1 will be equal to made 1. So, in this case the value of V_1 minus V_2 will be fed over here. So, this way the multiplexing arrangement is possible and it actually works in this manner.

So, this is very easy and basically now 1 more thing I have to say the 2 emphasis thing is that one has you have seen this is the data path. So, what is the data path data path actually takes care of the fact that how the V_1 and V_2 are moving, what are the controller doing the controller based on the value of V_1 and V_2 subtraction greater than less than is setting the value of the multiplexers.

So, that the exact operations are happening and the data is flowing in the correct path in the data and again the complexity highly increases in all circuits because, if the number widths are increasing the data path size blows up, but controller will more or less remain the same as you can easily appreciate that even if I have a $A1 \vee V_1$ and V_2 as 4 bit numbers or they are about 1028 bit numbers the controller will remain same only the size of this data path will become enormously large.

So, we always target to test the I mean data path in a very abstract manner. So, in this case as we are easily appreciate that even if the V_1 and V_2 becomes 128 bit numbers, the size of this circuit in this terms will remain the same.

So, knowing you are testing circuit at this level of abstraction we are how many blocks 1 2 3 4 5 6 7 8 9 10 11. So, only eleven blocks are there or you can say that eleven RTL blocks are there, so they are synonymous to gates in the gate level testing. So, only a circuit with eleven gates are there I have to test it this is very simple, but if I have a 128 bit numbers in this are data path a number of gates will be around 10s of 1000s.

So, you can find out what is the complexity and how it will be difficult to generate the test patterns at that depth in the gate level, now you to also appreciate the fact that this RTL model for this GCD you will be only a very 0.0001 percent of the original circuit in an NOC and SOC because, in a NOC and SOC are 100s and 100s of I p codes sorry 100s of IP codes in a single dye and in which case the arithmetic part of this course will have 100s and 1000s of circuits which whose size will be equivalent to the GCD. so very difficult problem in that way so we go for this abstract level testing.

(Refer Slide Time: 26:41)

RTL model (Control Path)

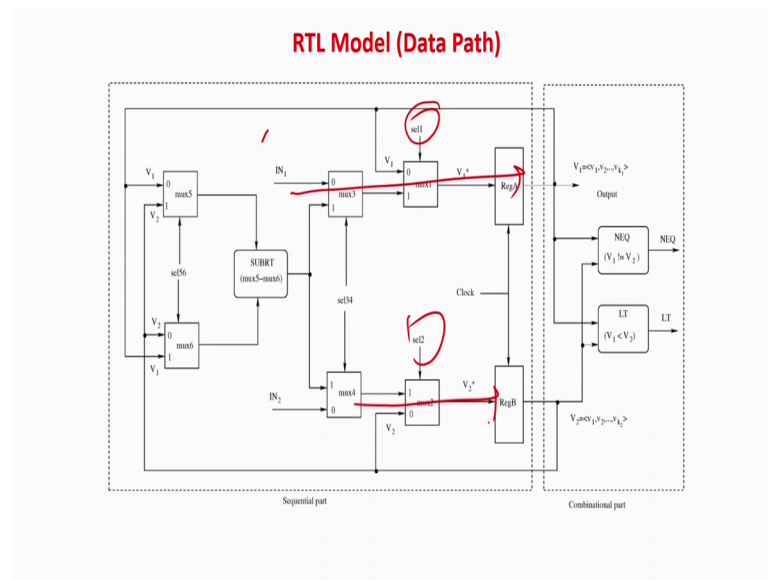
Table 1: FSM represents the control part of GCD algorithm

Present state	Conditions			Next state	Control signals			
	Reset	NEQ	LT		sel1	sel2	sel34	sel56
q0	1	d	d	q1	1	1	0	d
q1	0	1	d	q2	0	0	d	d
q1	0	0	d	q1	0	0	d	d
q2	0	d	1	q3	0	0	d	d
q2	0	d	0	q4	0	0	d	d
q3	0	d	d	q1	0	1	1	1
q4	0	d	d	q1	1	0	1	0

- q0 where reset signal is 1 -----sel1 = 1, sel2 = 1, sel34 = 0, sel56 = d.
- sel1 = 1, sel2 = 1 load the registers with input values which are selected by sel34 = 0 in both mux3 and mux4.
- q0 the control signal sel56 has no impact, thus, it contains don't care (d).
- Values of two registers (regA(= V1) and regB(= V2)) remain unchanged, and are used for checking equality and less than conditions in states q1 and q2, respectively.

So, I am not going to discuss about this controller path, so basically it is the control path. So, just I can tell you within gist what happens basically whatever control signals I told you is generated is shown over here; like for example, if I am the start state of the controller then reset is equal to 1 this some reset signal and basically select equal to select 1 equal to 1 and select 2 equal to 1; that means, what these 2 guys are equal to 1.

(Refer Slide Time: 27:01)



So, that the values and cell 4 also equal to 1 this 2 are 1 and select 4 equal to 0. So, select 4 equal to 0 means the in values will be last over here, so basically this the controller structure then automatically you go to the next state.

(Refer Slide Time: 27:14)

RTL model (Control Path)

Table 1: FSM represents the control part of GCD algorithm

Present state	Conditions			Next state	Control signals			
	Reset	NEQ	LT		sel1	sel2	sel34	sel56
q_0	1	d	d	q_1	1	1	0	d
q_1	0	1	d	q_2	0	0	d	d
q_1	0	0	d	q_1	0	0	d	d
q_2	0	d	1	q_3	0	0	d	d
q_2	0	d	0	q_4	0	0	d	d
q_3	0	d	d	q_1	0	1	1	1
q_4	0	d	d	q_1	1	0	1	0

- q_0 where reset signal is 1 ----- sel1 = 1, sel2 = 1, sel34 = 0, sel56 = d.
- sel1 = 1, sel2 = 1 load the registers with input values which are selected by sel34 = 0 in both mux3 and mux4.
- q_0 the control signal sel56 has no impact, thus, it contains don't care (d).
- Values of two registers (regA(= V_1) and regB(= V_2)) remain unchanged, and are used for checking equality and less than conditions in states q_1 and q_2 , respectively.

In the next state basically you again do kind some kind of computation and at many cases you can see that always I am looking for the less than signal, sometimes it will depend whether the less than signal is 0 on 1 whether it will go to state Q_3 and Q_4 in state Q_3 you may do V_2 minus V_1 and in state Q_4 you will do V_1 minus V_2 . So,

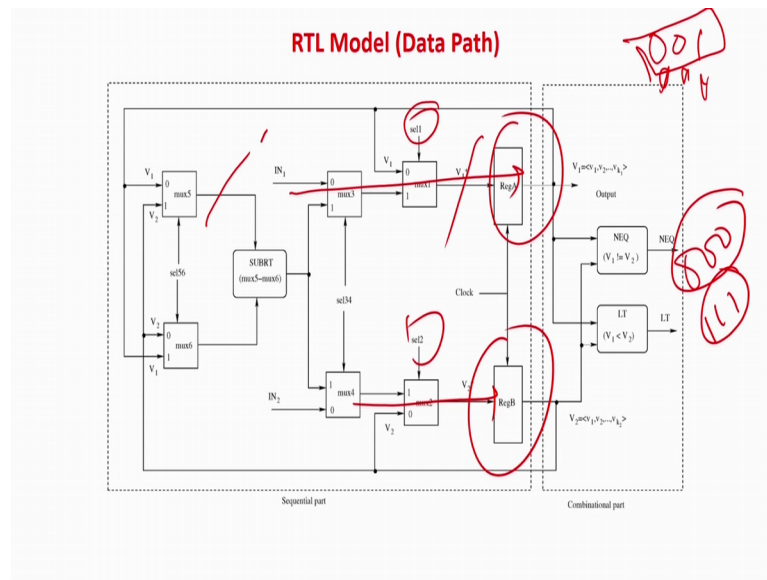
whatever I have told you the control signal which will be generated are represented by this final state machine of the GCD algorithm; that means, all the explanation I have done how the signals are generated etcetera is depicted in this table and the text below. So, you can just have a reading through this which will make your idea very clear though these are the entire explanation of the steps.

So, there will be actually 4 states, so 4 state means it will 2 bit machine. So, basically the number of flip flops will be 2, 4 state means a sorry 5 state q_0 to q_4 . So, number of flip flops required will be free to implement the circuit. So, now again important part is this very standard digital design you can easily go back to your higher means b tech level digital design and you can easily find out how state machine can be implemented.

But again reemphasising the fact that the control path is very simple in terms of number of gates compared to the data path, data path as I told you will blow with the number of size of the inputs input bit width, but you can see the controller will remain same and the control size is how much very very small because only 5 state; 5 state means 3 flip flops and number of gates to represent this final state machine will not be more than 8 10s or maximum the number of gates will be 10 or 20 not more than that.

If you make a combinational circuit using your standard higher, I mean digital design fundamentals we will find out the size will not be more than 3 flip flops and around 10 to 20 kids. So, test is of the control path and the abstraction level is possible, but not a real emphasise at this level really people are facing problem with this data path. So, we will be mainly concentrating on the data path in this case and also 1 more assumption we have that all this registers there is the flip flops and also the flip flops will be used to represent this states are also scan enabled that is obviously, possible very easily possible.

(Refer Slide Time: 29:23)



Now, as I told you can very easily appreciate that what I will do with the stuck at fault over here, you can say that I can only have stuck at fault only inputs and outputs you can do that, but we will have found out statistically that that is a very devastating fault model no correlation can be found not even 20 percent correlation could be found. If you directly apply a stuck at fault model over is not that people have not tried people have found out that stuck at fault model well accepted, I have very abstract model let us try to put some stuck at faults in this place and try to see what happens.

So that means, you are considering this whole line together whole V1 together; that means, either the number is in 1 or the number is 0 basically. But say for example, I am taking a 3 bit number so; that means, take it 0 0 1 so these are number. So, in a stuck at fault model in the gate level you will see whether this line is stuck at 0 1, this line is stuck at 0 1 this line stuck at 0 1, but in the RTL level this whole number is considered as a single number. So, either the number is 0 0 0 0 0 1 itself that is the original number or it will be all zeroes or all ones because, the whole number is taken together stuck at 0 and stuck at 1, so you do not have the individual granularity.

(Refer Slide Time: 30:35)

High level fault models for RTL

- A fault model is a convenient representation of the effect of the physical defects or failures on the operation of the circuit.
- A fault model is said to be reliable if there exists good correlation with the physical defects of the circuit, i.e., it can detect most of the physical defects.
- The transistor level fault models (transistor stuck-on and stuck-off faults) are most reliable fault models because the defects and the fault model both are at the same level, i.e., transistor level.
- The next reliable fault model is gate level fault model (stuck-at-1, stuck-at-0, bridging, etc.) because the physical defects like interconnect wire short or open can be directly detected by the gate level stuck-at faults.
- Similarly the physical contact between two or more interconnect wires can be detected by bridging fault model at gate level.

So, if you do it statistically it is found that stuck at fault model at the RTL is a very bad fault model because, the correlation is very very low, so you have to go something beyond that. So, people started thinking about high level fault models for RTL and again I have to tell that this is a not a very old concept. So, still statistical data is being collected people are trying to find out which is a good fault model, which is slightly complicated, but still you can fix with the RTL and still defect correlation is very good. So, as you all know fault model is a convenient representation of physical defects or failures to the operation of the circuit can the circuit is at the RTL level and physical level is a transistor.

So, lot of gap in the levels from a transistor level to a basically RTL level initially, it was very good the RTL level sorry it was the gate level and the transistor level there is a 1 half correlation, but now the half of correlation as increased very high. So, therefore the correlation make from a transistor level to RTL level is very difficult and we always know that a fault model is good because, if there is good correlation between the physical defects and the fault model.

Now, how people have started thinking about it, so people know that transistor fault level fault models are very good fault model because, transistor level fault model are very at the defect level. Now the next reliable fault level is gate level that is the stuck at fault level transistor level anyway you could not go. But that is the most this is the best

possible fault model you can say is the device level fault model, but anyway I mean beyond physics it is the if you go slightly more abstract from that, then the transistor fault level fault model will be very good because, there will be 1 half correlated with the defects again that is more very complex. So, we all accepted the stuck at 0 stuck at 1 fault model which was at the gate level.

Now, the gate level is directly correlated with transistors, so therefore people have found good correlation, now people have started finding out that we should find out fault models are the RTL level we should have a very good correlation with the gate level because , we cannot have a correlation between RTL level and transistor level it will be very difficult to find out and mathematically show or statistically find out.

Rather people have trying to find out such fault models which are good correlation with the gate level and the RTL level and they are thinking of a indirect transitive relation. That is from the gate level very good correlation with this transistor level and if we make good fault models which are very good correlation with the gate level may be there should be a good correlation with the transistor level and defect level. So, that is the idea of the high level fault models which is applicable for RTL.

Now, we are going to see some of the fault model which are being proposed for the RTL level and but still as I told you this is a very new area may be which are still time to find out good fault models with the very good correlation this is under study basically.

(Refer Slide Time: 33:03)

High level fault models for RTL

- Fault models at the highest level of abstraction.
- Design of the possible failures models of the constructs in the RTL.
 - Faults are injected in these RTLfiles.
- Motivated from Software Engineering errors

So, these are the fault models which should be at the high level of abstraction and the design of fault model should be very much coherent with the constructs of the RTL, as the faults will be injected in the RTL file; that is should not have a kind of gate level or transistor level business when you are talking about RTL level model and testing, which should be very similar to the faults which can be embedding the RTL. Also RTL as you see are very much similar to something kind of software.


So, these are software basically nothing but is a c code or a code either very translated it to ac code. So, people are try to get motivated from the fault model from the software engineering error, because is a huge subject in software engineering is software testing. So, what type of fault models they take people are tried to adopt those models into the RTL model because, RTL is nothing but the code.

Then they will are trying to find out how the correlation can happen at the real gate level and the transistor level that is the idea of research when you are thinking about testing of very complex circuits and the fault model at those level. So defend fault models input stuck at output stuck at.

(Refer Slide Time: 33:58)

High level fault models for RTL

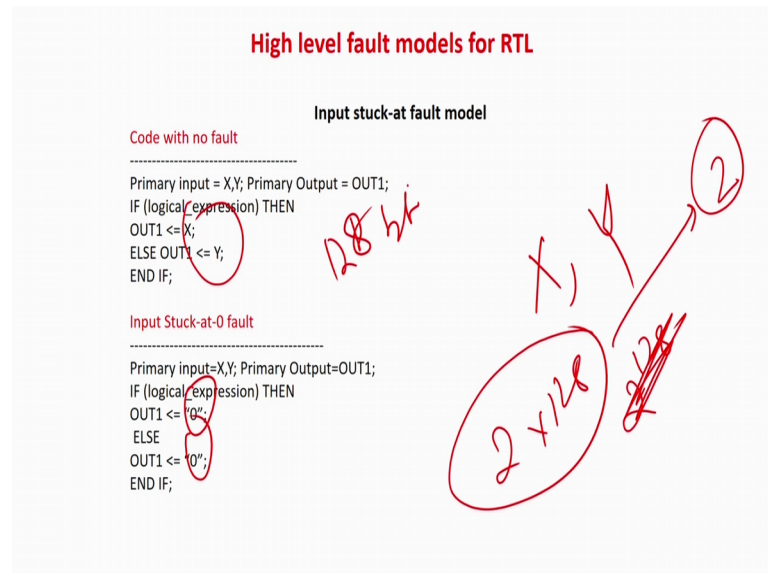
- Input stuck-at fault
- Output stuck-at fault
- If stuck-then fault
- If stuck-else fault
- Else-if stuck-then fault
- Else-if stuck-else fault
- Assignment statement fault
 - Dead clause fault
 - Micro-operation fault
 - Local stuck data fault



So in fact, they are again emphasizing that is the whole input taken together, so all the bits will be 0 or all the input bits will be one, we cannot have any intermediate chains in between. If then stuck else if stuck assignment failure dead clause failure and so many fault models are there I will discuss most important of them once, but again they are all

motivated very much from the software engineering perspective that is from the code because, RTL itself is a code in that manner so input stuck at fault.

(Refer Slide Time: 34:24)



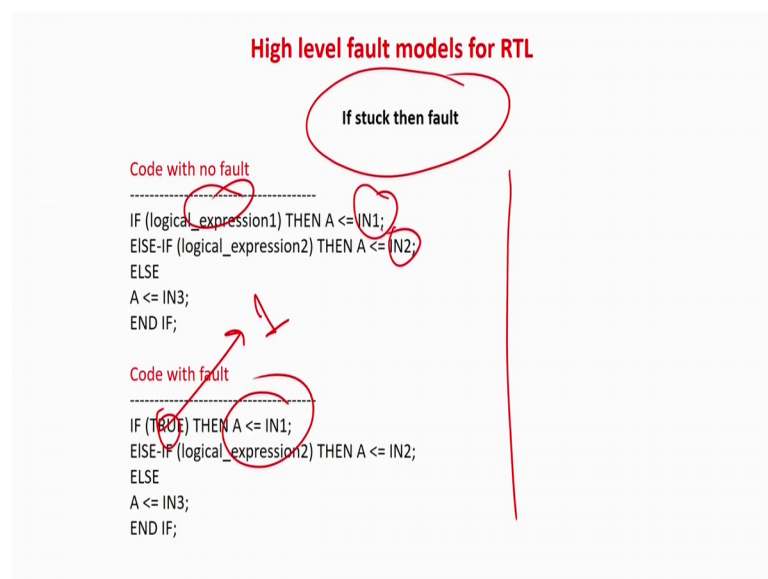
So, I have a very simple code which says that primary inputs x and y output is out 1, then it says that if logic expression something then out equal to x this 1. Now x and y may be 128 bit numbers, but then what is a stuck at 0 fault you will say that out equal to 0 and this equal to 0 that means, x and y all the inputs are basically stuck at 0, importantly all the whole x if it is 128 bit numbers all the 128 inputs will be 0 and all the 128 bit input will be 1 in 2 different cases.

So, instead of having 2 to the power 128 different combinations or I should not call it 2 to the power 128, in fact 2 into 128 because each it will be either 0 or 1 so 2 into 128 is now converted to only 2.

So, huge number drop is there so basically if you are taking a gate level models. So, all the bits has to be individually considered 0 1 0 1 like that, but in this case I am clubbing all the bits together and it is either all zeroes and all ones. So, there is a compromise here because, already it was proved that if you are taking each individual bit of this 128 x and y numbers, then very good correlation happens with the transistor level. But now we are just taking 2 numbers all zeroes and all ones.

So, among this 128 2 into 128 stuck at faults I am just taking 2 all zeroes and all ones. How much correlated they are so people have found that the correlation is not bad may be around 70 to 80 percent, but not as high as 99.9 percent, but still people say that at the huge high level of this design complexity of NOC's and SOC's we cannot go for 128 bit level. So, but this is only 1 fault model that is we have to apply all these and even many mores, so to get around 90 to 95 percent more coverage complexity or coverage or correlation level.

(Refer Slide Time: 36:15)



So, they may be this 1 is going to give you some 30 percent correlation, then we have if stuck at fault it means what is a logic expression we says that if some expression then a equal to in 1 else if a equal to in 2 some code is there. Then basically what is the if start that means this 1 will be stuck at 1 that is always a will be equal to in 1 because, always the position will be true I can also have if stuck at.

That means, always the if clause will be false, these are all the fault model which are taken from basically software engineering perspective and this we can very easily fit to all the RTL. This actually calls from if then else mean some multiplexer this fault model will directly fit to a multiplexer this 1 will directly fit to assignment values. So, so all these models has to be taken together to go for RTL testing then the correlation will be as acceptable level, but again the numbers will be very very less compared to the gate level.

(Refer Slide Time: 37:08)

High level fault models for RTL

Assignment statement fault

Code with no fault

```
TYPE TRAFFIC_LIGHT (GREEN, YELLOW, RED);  
SIGNAL LIGHT: TRAFFIC_LIGHT;  
IF(condition_1) THEN  
    LIGHT <= GREEN;  
ELSIF(condition_2) THEN LIGHT <= YELLOW;  
ELSE LIGHT <= RED;  
ENDIF
```

Code with fault

```
TYPE TRAFFIC_LIGHT (GREEN, YELLOW, RED);  
SIGNAL LIGHT: TRAFFIC_LIGHT;  
IF(condition_1) THEN  
    LIGHT <= YELLOW;  
ELSIF(condition_2) THEN LIGHT <= YELLOW;  
ELSE LIGHT <= RED;  
ENDIF
```

Because always we are considering this in and in 1 and in 2 irrespective of their bit size and there is something called assignment failure. So, we are saying that is a code in this case light equal to green but all the inputs may be stuck to yellow, then whatever condition you give everywhere the assignment will be yellow. So, this is some kind of a constant assignment failure.

So, this is another fault model, so in that way N number of different fault models exist, which I am not going to cover elaborately in this they are all mainly coming from the software engineering perspective. So, some 20 to 30 different fault model people have proposed mainly from the software engineering like if stuck else stuck some case condition is there some case condition is stuck assignment is stuck etc and so forth dead clause that means dead code etcetera etcetera.

So, all there if you take together then you may get in acceptable level of coverage, but not by the individual models; individual model the coverage or I mean what do I say correlation is extremely low. But always they are having a good level of abstraction and the complexity actually comes down heavily down considering the fact that the circuits are very large, then there is no other option to go for this abstraction in case of high level fault model like RTL or behavioural that does not exist any close relation with defect levels in the circuits.

(Refer Slide Time: 38:08)

High level fault models for RTL

- In case of high level fault models like at RTL or behavioral level, there doesn't exist any close correlation with the physical defects of the circuit.
- So, the high level fault models are not considered as reliable fault models.
- But it is verified that there exists good correlation between high level fault models and gate level fault models
- Further gate level fault models have good correlation with physical defects.
- So, there exists indirect mapping between high level fault models and physical defects .

So, they are not considered as reliable as reliable fault models, so 1 fault model is not very reliable. So, you have to take lots of fault models together test it and then only we can find out statistically the reliability is quite high. But not as much as the gate level stuck at bridging and delay fault, but still we have no other choice but to go in this direction since last 4 to 5 years I was telling you people all the test areas started moving to in this direction because, we have left with no other solution then finding out more and more accurate RTL fault models and go at the higher level of abstraction.

But as I told you but it is verified that there exist good correlation between high level fault model and the gate level models, gate level models they can have good correlation with physical level, so there is an indirect jump. So, we actually believe in or trying to find out that we have to test at the RTL level what are the good fault models at the RTL level.

(Refer Slide Time: 39:00)

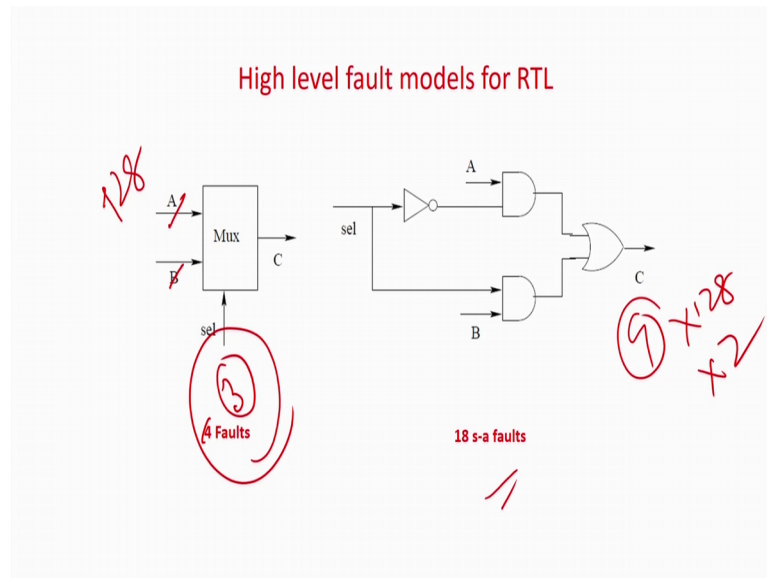
High level fault models for GCD example

Normal behavior	Faulty behavior 1	Faulty behavior 2	Faulty behavior 3
begin	begin	begin	begin
if(condition)	if(condition)	if(condition)	if(condition)
Output=input1;	Output=input2;	Output=input1;	Output=input2;
else	else	else	else
Output=input2;	Output=input1;	Output=input1;	Output=input2;
end	end	end	end

So, now again I will now again I will come back to the GCD example, take a real fault do an a t p g and then tell you the procedure. So, here I am taking an example of a if then else fault, so this is an if then else statement then 1 fault is that it is a swiping condition. So, output 1 if some condition is true else input 2 in this case it is just reversed. So, instead of the input 1 the position is 2, if some condition holds the output will be input 2 and test the output is input 1, but now they are flipped; this is constraint output that whatever may the condition input 1 will go and the other condition is whatever may be the condition the input 2 will go to the output.

So, basically they are the 4 fault models which are very much appropriate for a multiplexer. So, people has found out that if you test for all this faults in a multiplexer the correlation is very very high with the gate level model. So, what we do 1 is the swipe in and 1 is the constraint input and 1 is the constraint output.

(Refer Slide Time: 39:54)

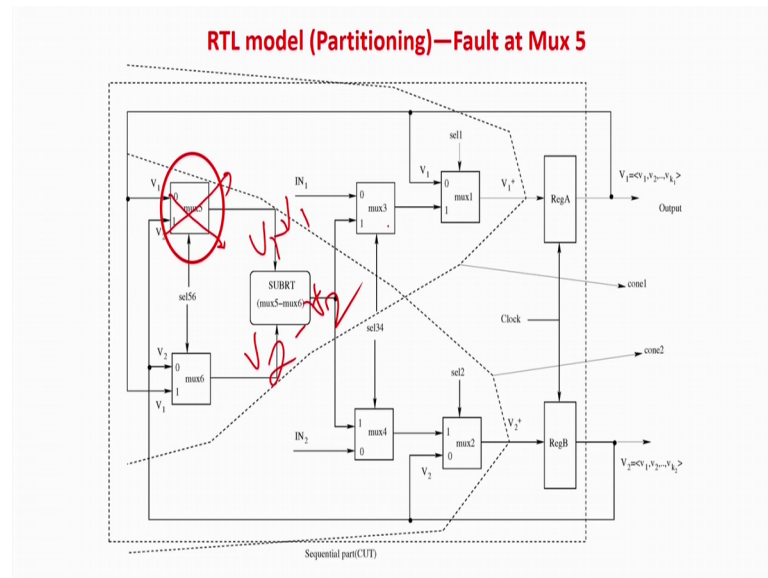


That is 1 a will always go b will always go and the other is there will be a swipe in that are the 3 fault models basically 1 is the normal basically. So, I am I should not call it actually 4 basically 3 1 is actually the normal condition. So, 3 faults will be considered over here and in this circuit if you take which is the equivalent gate level for the mux. So, there are actually 9 lines over here so 9 into 2 will be 18 stuck at faults.

So, from 18 I have come down to 9, but now you think that if A and B are 128 bit numbers. So, in this case still the number will faults will be 3, but in this case it will be 9 into 128 into 2. So, with the raise in the data bit the testing of the RTL data path becomes simpler well at the gate level it actually blows up like anything. So, therefore to make the data path complex testing complexity simple people have 2 or must go for RTL or some abstract level of testing there is no other solution in the current time, because nowadays our bit precisions are also high and a bits of the numbers are increasing in width, so therefore, that is the only way.

So, this example actually directly shows an example that from 3 fault basically we are coming down to sorry from 18 faults we are coming down to 3 faults if the mux is the single bit multiplexer, it will the gap will raise then the bit width will increase.

(Refer Slide Time: 41:15)



So, now basically this is the RTL same RTL circuit we are thinking, now I am going to take you to the example how to go for a test. We are assuming that the mux 2 of this mux 5 as a fault let me zoom over here, so it will be easier for you. So, this are the fault. So, now what is the fault I am taking the swipe fault so that means, what if in this case if select 5 is equal to 0 so V1 will go to the output. If this 1 is equal to 1 V2 will go to the output but now because of the fault, what is going to happen there is a swipe; that means, if select is equal to 1 basically here we will be going to have the answer as V1 and if select is equal to 0 the output will be V2, so that is the reverse just look at it what will be the effect of the fault.

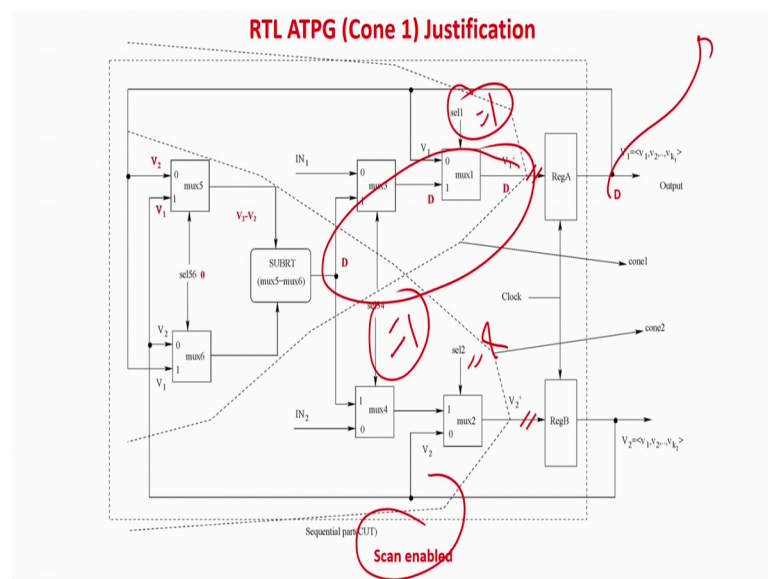
So, rub it off and say I say that select is equal to 0. So, if select is equal to 0 V2 will be coming out of mux 6 because, again still we are going always going for the assumption that single fault model nobody has been able to go for 2 or more higher level fault models, because that will make the complexity of NC1 NC2 and will blow up till now nobody is trying in this direction or a very few group is trying to find out what is the advantage and disadvantage. But as far as I know people are founding more disadvantages compared to advantages if you are taking 2 faults at a time 3 faults at a time.

Anyway so 0 of this means the V1 will come as the output here what is going to happen, here V2 means V1 should come, but because of the fault is a criss cross and this is going

to come as the output which is become V2. So, the answer will be V2 minus V2 which is be actually to 0 right, that is the effect of the fault. Now again if I take the other side if I take remove everything and if I take mux is equal to 1. So, if I take the mux 5 6 equal to 1, V1 will be coming at the output.

Here equal to 1 that means basically V2 should come out, but it will not V1 is going to come out basically and because of this swipe and again it will be V1 minus V2 sorry V1 minus V1 because, 1 here is the flop is a reverse. So, this is going to come out V1 minus V1 the answer is 0. So, you can highly easily appreciate that because of this fault this criss cross fault erase it and this is basically a criss fault. So, what is going to happen I am going to have the answer here V1 minus V2 V1 minus V1 or V2 minus V2 instead of V1 minus V2 or V2 minus V1, so that is the problem that is going to happen so that is actually the fault over here.

(Refer Slide Time: 43:49)



So, I have to move again for the same business sensitised propagate and justify same philosophy will apply let us see how it happens let me zoom it for you so this is where is the case. So, I have to sensitise, so how I will sensitise you know that general case it is V1 minus V2 or V2 minus V1. So, first thing I have to apply is that as the input I have to apply 2 numbers V1 and V2 and they should not be equal because, if they are equal the fault will not be sensitised as simple as that. So, basically I am applying I have to first

enable this scan why I have to enable the scan basically because, if you see this V1 and V2 are going as a feedback from this 2 register back.

So, if I do not set this before at the scan mode then, I then actually I have to use the old circuitry and I have to make some numbers at registers a and some numbers I have to register b which should not be equal and so forth. So, we do not go by that instead use the scan chain and there will be if the number is say 128 bit each. So, this scan chain length is 128 into 2 and you feed the values into the scan. So, that this 2 are 2 non equal numbers that is maybe you are setting the value of 16 here and may be you are setting the value of 1 121 here 2 different values I put to this numbers. So, that V1 and V2 is not equal that I actually I will do by scan, so initial I scan chain will be enabled and I will do it.

So, basic concept actually always remain same that is scan chain will remain same sensitise propagate and justify will remain same, only thing is that we are going to higher level of abstraction the basic remains same only the concept changes whenever we are going for abstraction level, but the basic philosophy of algorithms always remain same.

So, basics of computer science will never change basically. So, what we are doing we are enabling this scan V1 is not equal to V2 for sensitising it and actually we have to make select we are making select equal to 0, you can even make select 56 equal to 1 that also is possible because, if you are making select 56 equal to 0 then what happens basically from here V2 is going to come and from here again V2 are swap again the V2 is going.

So, basically what I am doing I am making the select equal to say 0, so in this case it will be actually V2 minus V2 will be the sorry this 1 actually this is the swapped values. So, if you look at it this is the swapped values basically so you are going to get the value of V2 minus V2. In fact, this 1 V1 and V2 is the normal case, but now it is swapped is V2 and V1, so this is was V1 V2 and by swapping it has become V2 V1.

So, we are going to get the value of V2 minus V2 minus V2. In fact, basically you could have also taken the value of select sel56 equal to 1 in that case you would have got the value of V1 and V1 minus V1. So, that is the material you can take anything now 1 point how do I make sele52 equal to 0 or 1 directly. In fact, sel56 is fade by the controller that is again a circuit. So, again the controller circuit will also have scan chain that is as I have shown you in the table that is the this table if you look at it. So, this table is very

easily implement as the digital circuit from your undergraduate digital fundamentals, that circuit will also a flops you can again set the scan flops. So, that actually it will directly make the value of V56 is equal to 0 1 or 1 as you like right, so that is done.

So, now I have made it 5 6 from the control circuit by using again a scan business. So, this is 0 and we are going to get V2 V2 minus V2 over here, again you could have also made 5 6 equal to 1 then it will be V1 minus V2 anyway for the example I have taken this. So, it is scan enabled rub it and make it clear for you. So, what I am having scan is enabled I am making V1 and V2 2 different numbers of my choice, but they are not equal to this is 0, so but basically the job is done and getting V2 minus V2 so the fault is sensitise. So, here we are going to get the value 0, but in case of a normal circuit the value will be equal to non 0 because V1 is not equal to V2.

So, now again let me zoom the relevant part of it, so the answer here is a D that is normal case the fault is non is 1 should not actually d, here is slightly different it is non 0 verses 0. So, in the normal case this are value is non 0 and the failure place the value is are all zeroes

But again I am telling you not a single bit number it is a bit width. So, may be V1 and V2 are 128 bit numbers. So, this all the lines are actually 128 bits in nature, here D means non 0 verses 0, again I am I am I am selecting this line for the propagation which any do it will be again D the multiplexer will be D. So, of course, you have to make sel34 equal to 1 because I have to this value and I am just propagating the value through this and basically so scan is disabled I am propagating the values from DD and this is the selection part and to justify what you have to do basically, similarly I am also showing the justification this is the propagation path.

So, justification here is very simple you have to make 3 equal to 1 and basically you also also have to make select equal to 1 and this can be X. So, this is the propagation and 2 is very easily you can justify the value, so D will be latched over here so. In fact, that is basically you can say that this is your actually the justification values this is basically equal to 1 and select is equal to 1 and this equal to x not required. So, basically if I say make it clean propagation path and basically these 2 are the justification part, your job is done now D is latched over here.

So, what you do D is latched over here, now you will be using a scan again you enable the scan and put the value at the output and just check whether it is non 0 number or 0 number, if 0 number this mux is having the problem of fault type else reverse if else reverse or it is a normal circuit. So, concept is extremely similar, but basically instead of a considering all these numbers as a higher bit numbers higher bit width, we are considering as a single block single line.

So, you can easily appreciate that even if the N 1 and N 2 are 128 bit or whether they are 256 bit or 1128 bit the numbers will remain very very limited to 11 blocks in this case. But if you are taking a gate level it will actually blow up. So, what do you have achieved basically.

So, the number of RTL faults in this whole circuit if you consider is something around 19, now I have also synthesised the circuit then the number of stuck at faults and I have simply considered the inputs and outputs as only 8 bit numbers in and in 2 are only 8 bit numbers.

(Refer Slide Time: 50:25)

What we achieve ?

- Number of RTL faults 19
- No. of s-a faults 844
- Minimization of test pattern generation time
- Low number of Test patterns

- Accuracy may be lower
- Bridging faults may not be covered
- Delay Testing is not ensured

Then the number of stuck at faults is 844 around including fault collapsing etc. So, you can see that is jump is so high and if I consider the number from 1 to 18. I means a 8 bits to 128 bits the number of faults will be something around 10 lakhs or 20 lakhs, but the RTL faults will never change it will be always 19 in nature and it is not only about this mux faults, I have taken all the relevant fault model for the multiplexers subtracted

registers less than equal to greater than equal to all the existing fault models which comes from the RTL fault model very like some which have already discussed.

Now, are all concerned considered in all the blocks basically then the number of faults is 90 and here the number of faults it was synthesising the whole circuit in terms of gates for 8 bit inputs is 844. So, you can easily see that I am going to lower the test pattern generation time like anything so and that is the thing you have to go if for going for a NOC and SOC kind of circuit, there is no other way other than going for abstraction and there very very important point is basically low number of test patterns because, you have 19 faults the number of test patterns to be 19 or even less than that.

But if they are more number of faults of course, the number of test patterns will be higher as I told you always telling you that there are 2 types of complexity 1 is test generation type that is not a primary importance, but more important is how many test patterns you are applying because that is to applied to all the circuits. So, here with the lower number of RTL faults the number of test patterns also comes down. So, testing of each circuit will also come down, so that is what is a big achievement we have.

So, therefore people are all moving towards RTL abstract level of testing in advance circuits because there is no other solution, but what we lose your accuracy may be lower people have till now with the all the fault models we are considering people do not know that how many bridging faults are covered, basically they are till now trying to mainly find the correlations with stuck at faults as slightly few words may be appearing which is for the bridging faults.

But as of now people mainly are looking at correlation between RTL and stuck delay is no longer ensured by this methods, means they have no must studies on those areas, but people have still find out that these good correlation between stuck and RTL and there is slightly quality compromise or the accuracy. But still as you can appreciate there is a huge jump in complexity like from the first lecture on testing, you might have found out that if I go for functional test it is 2^N and if you go to structural test it becomes $2N$ or even lower than that.

So, from complexity to very much linear kind of algorithm we are coming, but here also if the number of N is also very large that the numbers of inputs number of gates or nets are also very high, then again you have to even bring down the value of N there is no

other option other than to go for abstraction like RTL and RTL fault models. So, even from 2^N we have got to something like N by may be some 10s of 1000 or something like that we have brought down. If any 10 lakhs the number of an RTL if are going at the block level it may be drastically brought down by the number of bit size and may be 128 or 256 that is by the approximate by the bit width we are taking down the number of nets or the gates. So, the test pattern number of test pattern test time actually comes down.

So, basically this actually brings us to the end of the test methodologies for advanced circuits. So, that what we have covered basically in this module we have basically given you a very brief idea of testing and then we have seen that for typically complex VLSI circuits structural fault models like delay, bridging etc was still at the gate level. But when we are moving towards more advance circuits like NOC's and SOC's we have to consider the RTL or more abstracted view and about the fault models people have now moving from structural faults from stuck at to bridging delay and more sophisticated faults.

Also we have discussed how optimisation should be done at the level of the design for testable circuit. So, that all the in built circuits in this in the on chip circuits for testing should be lower in number, like how can you minimise the scan size how can you optimise scan type and how test pattern can be compressed, but these are all directions which are coming for advance test or advance test concepts for modern circuits.

So; in fact in this course basically is a is not a it is a very advance course which gives you pointers towards the newer paradigms of testing, by no means like a classical cad for VLSI I can cover all the algorithms. But what I have tried to give I have tried to give you ideas that these are the opened research avenue what people are thinking, what people are trying to develop the test methods and fault models which will be applicable in the modern day concept. So, and we will upload many references in the website for which you can use to perceive research in the direction of testing and towards and in the next module basically we will be looking at verification, in test what we do we take a circuit give input and we see whether it happens properly or not.

So, everything is based on the inputs we give, but can we mathematically guarantee that will work for all conditions, it is not basically it will actually make the problem very difficult like an exhaustive testing, but still for some cases we require mathematical

guarantee that whatever design you have is mathematically proper that is actually called formal verification. So, formal verification is a more complicated problem than testing because, we have to guarantee entire correctness of the behaviour. So, in the next module we will try to give how people have thought of optimising the optimising the verification algorithm for very complex circuits.

Thank you.