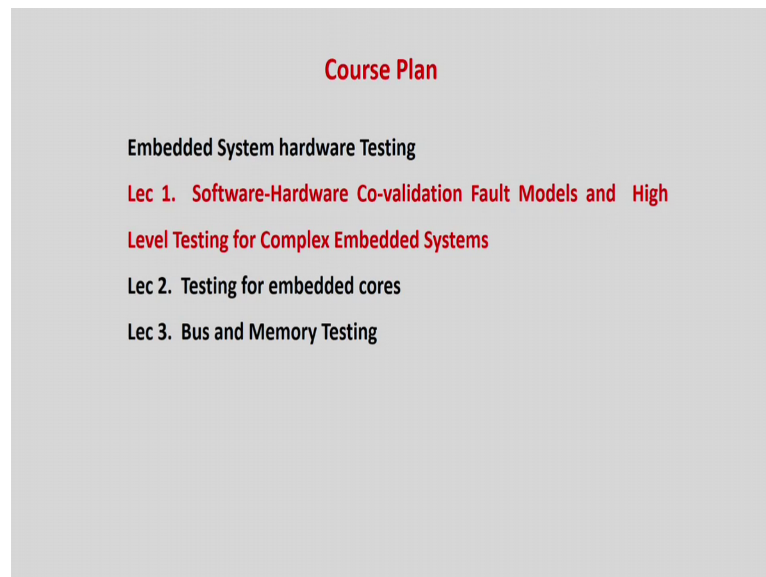
Embedded Systems – Design Verification and Test

Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 29
Part-3: Embedded System Testing
Software-Hardware Co-validation Fault Models and High Level Testing for Complex Embedded Systems

Hello everybody. So, as we are know that we are in the third part of the course which is on testing. Now, we will going to start the second module that is we have already seen the basics of tests of an embedded systems. Now, we are that is the basics of the prerequisites we have we have already discussed which is required to understand in depth of the testing of embedded systems.

(Refer Slide Time: 00:42)



Now, in the next part we are going to look at some of the concepts which are more specific to embedded system testing. First we are going to look at some kind of a fault models, which are motivated from hardware software design and we are going to look at testing at a higher level of abstraction. Then we look at because nowadays embedded systems are made of a multiple cores; that is who do not actually make all the cores by yourself you can input them from the vendors and put it in a large die which is the NOC

or an SOC. We will see how to test them and finally, all such complex embedded systems are connected by bus and memory. So, we will look at bus and memory testing.

So, now we will start with the our concepts on high level testing using the concept of software based fault models. Because as we know from on Prof. Arnab Sarkar discussion, Prof. Dekas discussion that embedded system is lot of hardware and software core design and core verification.

(Refer Slide Time: 01:41)

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, what we have seen that all faults all testing are basically done with fault models mainly started fault models and we called this structural test. So, fault model is nothing but an abstraction of the fault abstraction of the real defects, so the testing becomes very simple. And you can generate the test patterns easily. You can expirior test by applying test patterns fast because in number of test pattern should be small, but at the same time it is a very good correlation between the fault models and the real defects, so that quality can be ensured.

(Refer Slide Time: 02:13)

Complexity of Structural Test

Stuck at Fault Model:

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

And then basically what we have seen that we have already seen and details about stuck at fault models. Then what is the complexity there, that is if there is number of circuits or nets or lines are n there will be two in the number of stuck at faults we have to tested. And basically many of the case will require backtrack that is propagation and justification site that you have propagate the fault then you have to justify it, so they are may be errors or then be conflicts. So, you have to again backtrack. So, that backtrack is basically one of the main complexity of the circuit.

(Refer Slide Time: 02:37)

High-level fault modeling and RTL based Testing

- One of the most important issues of testing in modern deep sub-micron ES 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 ES testing

And in fact in modern embedded systems basically the circuit size the number of gates are in millions and there are multicore designs. As I told you an NOC or an SOC may have 1028 circuit course itself. So, even if you are as you have discussed the stuck at fault model is very efficient because I mean the number of stuck at faults are just to a not exponentially high, also the test patterns for most of the case first we go for us go for a random test pattern generation, and then for very difficult faults we go for sensitize propagate and justify approach where there can be conflicts.

But if you think about very large and particle embedded processors, the number of cores are so exterminately high that even stuck at fault detection ATPG becomes very very challenging and difficult. So, for such cases we require even a very faster method of test pattern generation. Again I am telling this lecture is not to minimize the test patterns as such; there is one of the main criteria of course. But main emphasis of or motivation of this kind of test pattern generation keep on the fact that even stuck at fault model base such an optimize scheme first stuck at fault model test could not be applied for large scale embedded processors.

So, there people have to think of something different stuck at fault model base testing will what very well for more morally mainly single core unit, but when we have gone to ultra large scale integration for embedded system such type of thing started fairly. So, then people started thinking can I do something in the abstract level or in a more soft level. Because in embedded system design first we generally as the software development, then you have a hardware partitioning, but then again the hardware is also written in a very log mode with Prof. Arnab Sarkar told you in brief. So, basically embedded system a lot of soft cores inside or software level descriptions.

So, the idea people thought is that can we have some kind of a hardware fault model and RTL mean register transfer level, can we try to test circuit is a very high level of abstraction. Testing cannot be vary at high level of abstraction because testing always happens in the gate level fabricated device. I should test plane can it be develop at a higher level of abstraction because if we have gate levels and more finer details basic what problem you are having is more number of wires in multi core SOCs extremely high number of gates and nets, and even with this very efficient test methodology is like structural test for stuck at faults these becomes very very complex to handle.

So, people though it can I plan test at a very higher level. And it should be possible in case of embedded systems because embedded system many modules are written in a software hardware code design methodology. So, we have very high level description models for many of the systems. So, most of the important issues of testing in modern deep sub sub-micron embedded system design is scalability as I was take on the multicore systems, because test schemes are generally logic level leading to sate explosion.

To solve this issue, people try to move it to higher level of abstraction like register transfer level architectural level etcetera; but again if I moving to test at the higher level of abstraction you should have very high level fault models for the embedded systems. Can you have a high level fault model? So that is the motivation of this lecture that looking at the embedded system design from a software perspective and software models what can we do?

(Refer Slide Time: 05:38)

RTL based Modeling

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

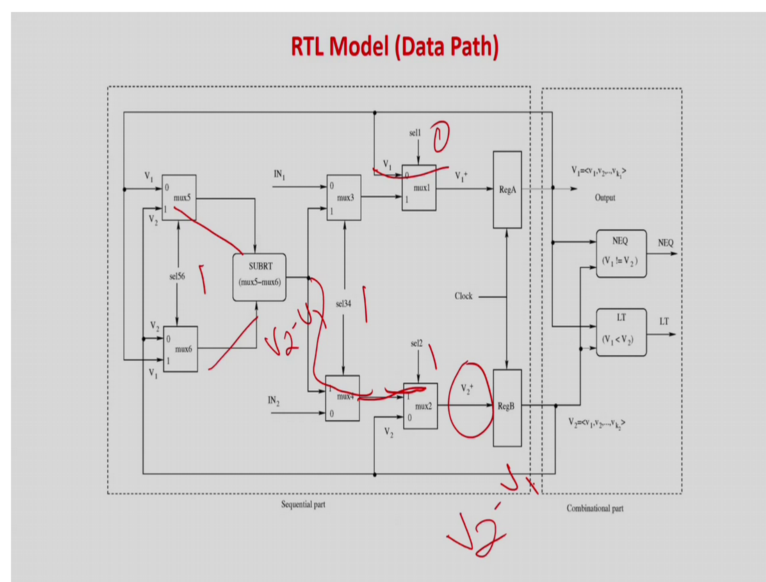
1: Begin
2: $V_1 \leftarrow IN_1$
3: $V_2 \leftarrow IN_2$
4: while $(V_1 \neq V_2)$ do
5: if $V_1 < V_2$ then
6: $V_2 \leftarrow V_2 - V_1$
7: else
8: $V_1 \leftarrow V_1 - V_2$
9: end if
10: end while
11: $Output \leftarrow V_1$
12: End

Before going to any elaborate theory I will take you an example. So, this is an example of a GCD computing hardware, assume that this is a embedded module which embedded of hardware which will do GCD calculation greatest common devisor. Again in embedded system or the most of the cases we also have a software description because an which is more emphasize in embedded systems because embedded system starts with the basic software description. Then we take some of the soft cores and some will

actually take at the hardware. So, always we have good deal of software description for such modules.

So, if you look at it is so this same as very standard algorithm we take the inputs and then we compare V 1 and V 2 whichever is larger we do this subtraction if V 1 is larger we do V 2 minus V 1 otherwise V 1, V 1 minus V 2, we keep on doing it still basically at till both of them become equals, and then the output is actually V 1 very standard algorithm.

(Refer Slide Time: 06:29)



And basically it is the hardware circuit that looks like but now it is different of much higher level of abstraction that is what has to be looked into. We will not going to gate level description. So, if you look at it, I will just give you the description of the circuit because this more or less a functional description not directly related to testing, not for the sake of completeness basically I will describe you the circuit.

So, if you look at it, so there is basically let me zoom it for you. So, there is basically one substractor involved. So, the substractor we will do V 1 minus V 2. If this multiplexer select five six line is basically equal to 0. So, it will be V 1 while coming over here and V 2 will be coming over here. So, answer will be V 1 minus V 2. If the select line is 1, then V 2 will be coming from here and V 1 will be coming from here to the multiplexer. So, it will (Refer Time: 07:14) be if this is a 1, it will be V 2 minus V 1 right similar is a substractor.

Again if you look at this part of the circuit, so there is a storage unit this is your registers where you store the values of variable V 1 and V 2. Of course, you require two comparators whether both the numbers are equal or not or whether whoever is greater than or less than. So, this greater than signal, basically will be feeding this select lines.

So, as I told you if V 1 is less than V 2, if this is true, it may generate the value 1. So, then if it is generating the value 1, so it will be actually caving this value V 2 over here and V 1 over here that is if this is a 1, so this one is a 1, so this one will go over here and this one will be so this one will be going over here. So, it one will be V 2 minus V 1, it will be V 2 minus similarly the other way. If this is 0, so this one be V 1 minus V 2 is you can easily look at the first part of the circuit and you will be explain.

And basically if you look at this is storage part, initial will be loading it. How will you load it initially? So, if you look at it, this is the input line, this is your input line. So, initially select three four is equal to sorry initially this will be equal to 0 initially. So, initially if it is 0, so basically it one will be IN 1 will be coming to here, I will also make select 1 equal to 0; I will also make select 2 equal to 0 sorry I can make select 1 equal to 1, sorry, sorry, sorry, I will make this as 1, and I will make this as 1; and initially this is 0.

So, what is the case? So, if it is 0, this IN1 will be coming over here it is 1. So, it will be going over here it will be stored in register a again this is 0. So, this one will be coming over here this is a 1. So, V 2 will be storing in register B. This corresponds to this state initial statement again this comparator business already I have explained you that depending on this value, how this part of the circuit is going to generate the answers.

Again and there will be one more thing which is the very important that this if you look at this point V 1 is assigned to V 1 and V 2 is assigned to V 2. So, if I make select 1 equal to 0, and select 2 equal to 0, the line the values of the variables are freezed that is they are not changed. Because in one case it is V 2 minus V 1; another case it is V 1 minus V 2. So, when this operation is happening, so V 1 should be freezed and vice versa. So, this part of the circuit that if select 1 equal to 0 and select 2 equal to 0 you see that this is a feedback.

So, V 2 is basic it easily getting as a feedback from here. So, they are basically freezing the lines. So, these are design principle, you can easily look at this circuit and you can easily get a feel that how this circuit will be operating important point. So, this lines

basically the select lines are control lines. There is actually a controller circuit which I have not shown over here they will take the lines from here and here and they will respectively generate the control signals.

Generally that is a small finite test machine for which you can use any way of testing like it is stuck at fault model, but generally this is actually the data path. So, data path testing is very very complex. Why, because the number of bits like V 1 and V 2 may not be a single bit number, they can be 128 bit number, they can be 256 bit numbers, because you are computing the GCD of two integers. So, integers can be long ints, they can be even flouts, so it is quite large. So, you have to appreciate that this bits or this mark size or this subtractor size or this register size are not single bit they can be 256 bit registers they can be at 256 bit multiplexer they can be a 250 bit subtractor.

So, you can understand how large this circuit will be a single multiplexer has some kind of four or five gates or 256 bit multiplexer will have some 256 into 4 number of bits. Think about the subtractor, substractor will have very very large gate if is a 256 bit subtractor. And this is one small module of the ALU which can do GCD. So, if the ALU is 128 bit ALU or it can support 256 bit numbers, the data path which requires the mathematical computation becomes very very large in circuit because of the increasing the data width.

But this controller bits are always single bit, because they are taking some later less than greater than signals and they are just sending out the control values as the signals. So, which is nothing but a signal finite state machine or a controller which will take some of the output bits which are generally single bit single bit patterns like less than signal 1, greater than signal become 1, and they will respectively control the select lines of the multiplexers. Generally it happens that way. So, you can easily appreciate that this is the very small circuit.

So, for this people are not worried about how to test it, testing it very simple. You can take a single stuck at fault model and go for structural test. But people are mainly bothered that testing of the embedded hardware because of the large precision numbers which are considering such multiplexers or which is part of the data path. So, we will today see how to test this data path using software motivated fault models and testing at a higher level of abstraction.

And again emphasizing this is possible in embedded systems mainly because embedded designs mainly start from a software hardware code design. So, we will always have software specific models. Again for this again a design principle, I am not going into too much details on it just I will give you the idea that how the control path works.

So, as I told you this is the control path states. So, q naught may be the initial state. So, as already discussed that the initial state basically what should happen the initial state basically should load this in value to register 1 and in value 2 to register b. So, what it should happen sorry it should have 0 because it should flow in and select 1 should be 1 and select should be 1 right and there it should be loaded over here.

(Refer Slide Time: 12:43)

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

- 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, in state q 0 basically you are saying there is some reset line, which will actually be resetting the whole stuff, so that is the condition and don't care a less than or next this to less greater than less than not equal to not importance for putting d d. So, if reset equal to 1, the next state will be is q 1 because this is the q naught is the reset state; from here you are going to q 1 where some functionalities will happen. But in q 1 what are the control signal generated control signal generated as sel equal to 1 and select 2 equal to 1. So, it is going to make this as 1 and this as 1, and select 3, 4 should be 0.

So, as saw means yeah select 3, 4 basically in this case should be 0 right. So, select 3, 4 is 0. So, this is 0. So, basically each will be feed over here this one is feed over here and

this select 5, 6 is basically a don't care, because I am not bothering any computation right know that is basically your initial state.

Next state automatically it will go to something called a q 2, sorry it will go to q 1. So, if you look at q 1, so this is about your basically first state. Now, in this state q 1 if you look at, there are two parts of q 1. So, now, you have to make this reset equal to 0 because work has started competition has started. Now, in q 1 there can be two conditions, basically not equal to either q 1 this V 1 is equal to V 2 or V 1 is not equal to V 2 that is the condition that is being checked, at that same greater than less than equal to am not checking this is basically corresponding to this line of (Refer Time: 14:10) right.

So, now if you look at it, so if it is not equal to you are going to something to state q 2, because in q 2 it is if we look at not equal to you are going to state q 2. And in q 2 you will be to doing all these comparison etcetera right. And if it is equal to that is corresponding to this, then basically you will be landing in this state q 1 itself. It may be q 1 itself that is already equal to so basically you have to freeze the answer. So, this is your answer as if you look at it, this is your answer output equal to V 1, if V 1 is equal to V 2.

So, if you look at it, so if you it will so, it will go to q state q 2 and in this case basically select equal to 1 equal to 0 and both are equal to 0 0. So, you have to make all of them 0 0. So, select 1 will be 0 and select 2 will be 0 right. So, this one is going over here and this one is going over here. And we do not much bother about select 5, 6 in the state q 2 I mean we can analyze this later on right.

And in fact, in this case is both of them are equal you will be going to state you will remain in case state q 1 itself and the answers are 0 0. So, basically in both the cases if you look at in both the cases you look at this select 1 and select 2 lines are 0 0 in state q 1. So, what is that? This is 0 0 and this is 0 0. This corresponding to the fact than V 1 and V 2 are freezed, there no changes in value. But if it is greater than if you not equal to if V 1 is not equal to V 2, then you are going to do next state when lot of activities will be there.

But if you are if you if you are both of them here equal to you will be in this state q 1 itself, so you will be here is a sel proof and the value is freezed, so that is actually you have stopped over here and the output you can take from the V 1. But if it is not the case,

so we will go to state q 2 that is this functional state you are going they are not equal to. So, again there will be two q 2s right they are taking two q 2s. So, in this case again mainly I will be looking at the at the less than greater than, because the state q 1, I have already established that rather they are equal or not equal that they are may by main concern. In the next state q 2, my main concern will be later than later than or great less than or greater than.

So, in one case if V 1 is greater than V 2, it will go to some state; and in the other case it will go to a new state q 4. So, there will be a bifurcation and there will be going to state q 3 or state q 4. And in both of them the activities will be different. So, if you if I start looking at state q 3 say. So, in q 3 you will look at it you will see that select 1 is equal to 1; and select 2 equal to 0 at this one is a 1, so it is 0. So, this is this one is 0; this is 0; and basically this one is a one. So, in this case the V 1 is freezed. And in this case the new in this case the new value are V 1 will be coming out that will be nothing that will be and now right.

So, in this case basically so in this state if you look at we are looking at this state we are looking at in q 3 we are looking at the basic cases this one that is 0 and a 1 as the output. So, in this case if you look at, so this one is V 1 is freezed and V 2 is going to get a new value because select 2 is a 1. Now, you have to very much concerned about these two signals 3, 4 equal to 1; and 5, 6 equal to 1. So, 3, 4 equal to 1; and 5, 6 equal to 1. So, 5, 6 equal to 1 means V 2, and this one is sorry this also 1. So, it will be V 2 minus V 1 and this is also equal to 1; that means, this one is going to be fed to this. So, V 2 is now going to get the value of V 2 minus V 1. And why it was so because of the fact it was so because if you look at it, this is the case we are analyzing. So, in this case V 2 was greater than V 1.

Just simple digital electronics I am just given you the highlights of how it can be analyzed. So, you can easily trace out the signals and the next state and the graph. And you can easily trace how basically this whole functionality will work that is not the main emphasis of this course to understand the working of the circuit. Just I have somehow I elaborated for your understanding of the context of this lecture.

Important part is you have to understand that this one is a very simple hardware. If you try to implement as a circuit this one will be some 10 or 20 gates will be there involve,

irrespective of the bits size of V 1 and V 2. Say V 1 is 128 bit, 256 bit, circuit of this data path will changing entirely, because it requires substractor of high bits comparator of high bits less than equal to comparator of a higher bit orders, but irrespective of the bit size, these two tables the control path will be always same maybe a 10 not 20 gates. So, testing this controller is not at all a concern for embedded system, mainly I will be looking at testing the high level data path.

(Refer Slide Time: 19:04)

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, I have to do it as a higher level of abstraction. So, as already told that fault model is a very convenient way of representation. A fault models is a reliable if there is a good correlation between physical defects and the circuit faults. So, we all know that stuck at faults are very good correlation with transistance faults or transistor level faults or physical defects already well established

Now, next reliable fault model because it is said that transistor level faults are reliable. Physical level fault very reliable gate level and transition level are somewhat correlated. So, gate level also very good like stuck at fault model is a gate level fault model. We will already established very well this concept using statistics, but still now not very good results have been may could be established or statistically found that can some more higher abstraction level of faults can be taken into picture.

But recently some works have been coming out and very recent concept has come out in which case people have tried thinking fault models like this is defect physical defect,

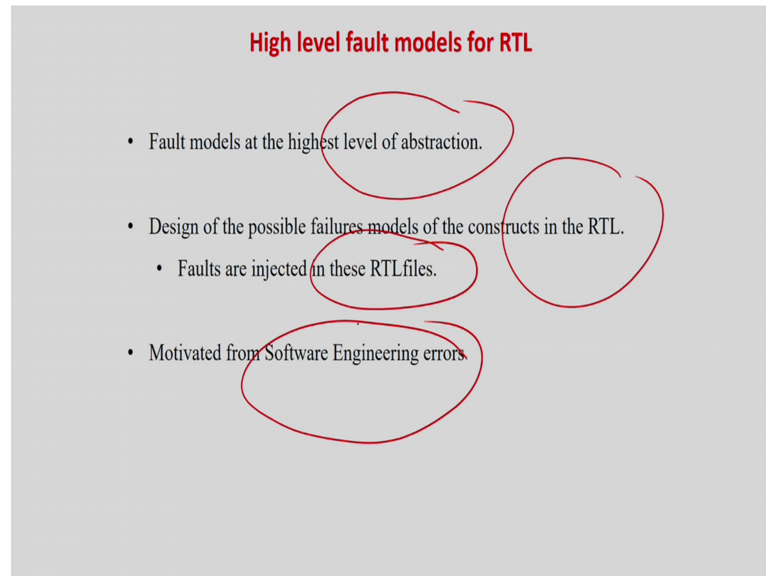
then transistor, then gate. And then people are thinking at the RTL or software level, then people have found out that this are all this path is already established. Now, still people are in the flow saying that we already know that RTL and gate level are quite related because RTL is transfer level and gate level is thus next level.

So, people have established there can be a good correlation at this level theoretically. And still the research is on that how much correlation exist between RTL and physical defects. People have lot of statistical measures are still going on and still the research is on to find out what is the correlation level. But still people have an estimate that as RTL as very good correlationship with gate and gate has very good correlation with defects. So, there should be a very good correlation

But people have still prone we have to still use RTL because I cannot go for gate level even if gate level is very good and very reliable, but for large embedded system gate level fault modeling is not a very good practice because of the complexity involved. So, even I know that gate level is very good, but because of the large side of the circuit I cannot go.

So, the current research is that how we whether we can apply any RTL level fault models, and test the testing at a higher level abstraction. And if it is so what how much would be correlation is. So, but the next generation testing will always be at the higher level. However, good is the stuck at fault model and the gate level we cannot go for the complexity. So, let us see how it gets modify.

(Refer Slide Time: 21:26)

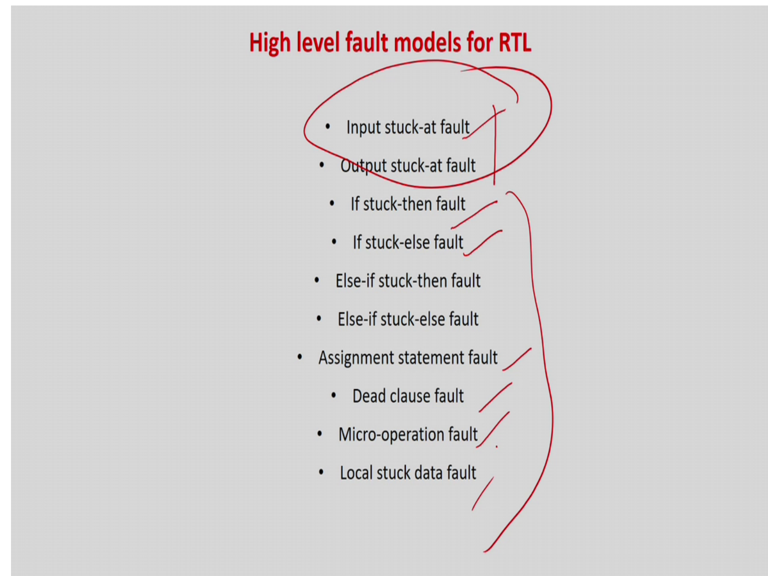


So, fault models in case of RTL are at the highest level of abstraction directly as like your software programming. So, and the fault models are such that you can insert them at the RTL. Of course, you cannot have stuck at fault model and insert at the RTL level that is not going to be do any kind of logic solving. Because RTL means you have if they nails like the code I have showed you like this is the code, you can insert stuck at fault like I can make all IN equal to 0, but that is not at all going to solve the purpose and people have seen that it has failed miserably because if you are thinking a circuit in this level the fault model also should be in sync with it.

So, the fault models are at the highest level and they are in an injected at the RTL level itself, like stuck at fault models gate level injection. Here RTL level high level faults are injected there. And they are mainly motivated from software engineering errors or as I told you it is very much feasible in embedded system because embedded system design start with software. And people have really found out started finding out good correlation and good level of course, the correlation cannot be as high as 99.99 percent, but people have found out more than 90 percent coverage in this case, but the great gain is that you can save a lot in the complexity and the number of test patterns if you go at a RTL level.

So, with increase complexity of circuits, people have to trade off that, because as I already told you test industries based on economics. So, if the circuit size is large, you have to slightly compromise. There is no other way of doing it.

(Refer Slide Time: 22:51)



So, what are the important fault models at the RTL level, very similar to software engineering. Input if input stuck at that is the all inputs will be stuck or output will be stuck anyway they are carrying that out, but there is only one of that. All others lot of others has to be done because first people try at that can I directly apply stuck at faults in RTL level, people failed miserably. But still they are into the picture all inputs will be 0 all outputs will be 0. Apart from that they are taking if stuck fault, else stuck fault, assignment fault, dead clause, micro operation, local data stuck, lot of these principles are from software engineering. I will give you an example to do becoming more concrete. This is the list.

(Refer Slide Time: 23:30)

High level fault models for RTL

Input stuck-at fault model

Code with no fault

```
------------------------------------
Primary input = X,Y; Primary Output = OUT1;
IF (logical_expression) THEN
OUT1 <= X;
ELSE OUT1 <= Y;
END IF;
```

Input Stuck-at-0 fault

```
------------------------------------------
Primary input=X,Y; Primary Output=OUT1;
IF (logical_expression) THEN
OUT1 <= "0";
 ELSE
OUT1 <= "0";
END IF;
```

So, I will take you an example. Inputs stuck at fault model. Like this is a code with no fault this is primary input x comma y this is the output if some logical expression out is equal to 1, else out equal to 0. You can make all output as default 0. So, this is the input and output stuck at fault model. Explicitly change the inputs and output to 0s that is very simple, we are not much interested.

(Refer Slide Time: 23:49)

High level fault models for RTL

If stuck then fault

Code with no fault

```
------------------------------------
IF (logical_expression1) THEN A <= IN1;
ElSE-IF (logical_expression2) THEN A <= IN2;
ELSE
A <= IN3;
END IF;
```

Code with fault

```
------------------------------------
IF (TRUE) THEN A <= IN1;
ElSE-IF (logical_expression2) THEN A <= IN2;
ELSE
A <= IN3;
END IF;
```

If then stuck fault, this is very important fault not at all was presently structural fault models. If you look at it is more linked to the functionality of a circuit and more related

to RTL level or software level, such type of fault models basically first appeared in software engineering. Because at the time we cannot call it as a fault basically this were design mistake, but again the emphasis I means started again coming or the motivation started coming from that fact.

So, what is the fault? If is logical expression then A equal to IN 1, else if IN 2 that is code. Now, that is the if stuck that is the only thing which is always going to be true, it will never come into else that is actually if equal to true that is the fault model in. That was always tried in software engineering because if you make certain errors in the coding it may happen that you will always not it will never go to else. People motivated from there they have taken a fault model like this. If true then this always true means it will you will never execute this.

(Refer Slide Time: 24:47)

High level fault models for RTL

Assignment statement fault

Code with no fault

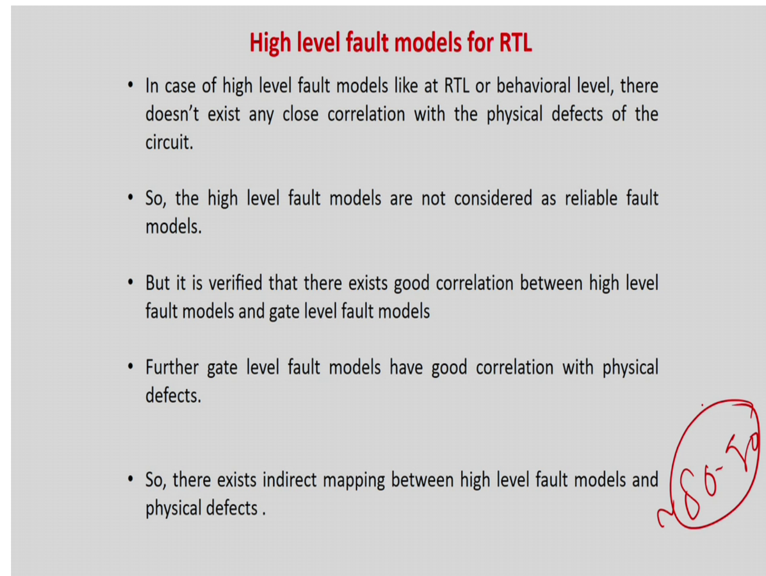
```
-----------------------------------
TYPE TRAFFIC LIGHT (GREEN, YELLOW, RED);
SIGNAL LIGHT: TRAFFIC_LIGHT;
IF(condition_1) THEN
    LIGHT <= GREEN;
ELSIF(condition_2) THEN LIGHT <= YELLOW;
ELSE LIGHT <= RED;
ENDIF
```

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
```

Similarly, assignment level fault. Some if then else are there something is the light equal to green, else light equal to yellow else this. But in this case what is the error instead of green, the assignment is different. So, they are all software engineering faults, but still people try to find try to incorporate them in the RTL design. And they have found out quite a good correlation with reliable defects. Although the motivation came from software engineering, now statistical result show that correlation is not as high as 99 percent like stuck at fault, but still about 90 percent correlation, but you can you can save a lot in the design and fault generation time and the number of faults.

(Refer Slide Time: 25:22)



So, high level fault models. So, in case of high level fault models that does not exist any close correlation with the defects. So, high level fault models are not considered very reliable, but it is verified that there exist good correlation between high level fault model and the gate level. Further gate level fault models have good correlation physical defects. So, there exist indirect mapping between high level software based fault model and the physical defects.

And people have find out approximately 80 to 90 percent correlation people have found out in most of the cases, and saving of complexities very very huge. So, this is this where you have to go in the next generation. If we as again repeating if stuck at fault may be very very good, but still you cannot do anything about it because of the complexity.

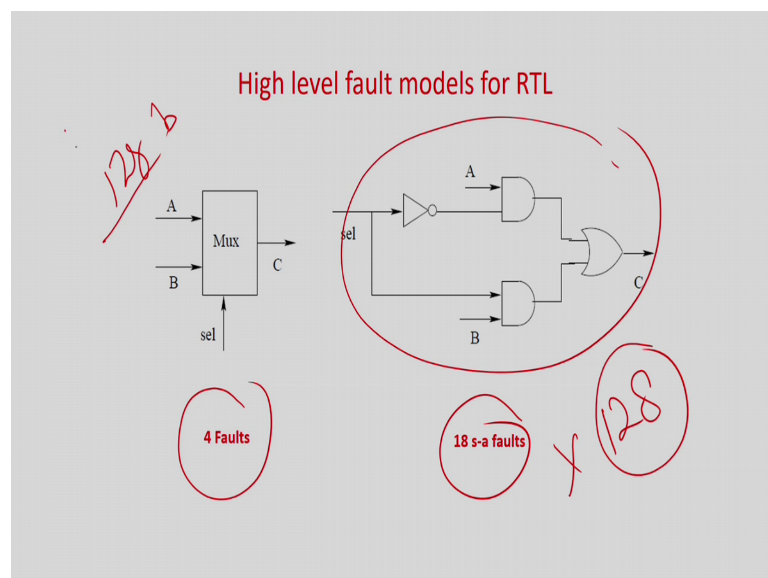
(Refer Slide Time: 26:02)

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

Now, I will take an example again I will come back to the example of the GCD. So, this is the normal behavior and we are going to take a fault in a mask. So, this is the normal behavior of the marks if condition input 1 input 2; faulty behavior 1, the get they get swapped. So, instead of 1, it will become 2 and 1 constantly. Input for both the cases the input output is input 1 and the second case the output is input 2. People have found out that if you take this fault model for a mask, then more than 95 percent plus correlation has been established for the multiplexer.

(Refer Slide Time: 26:31)



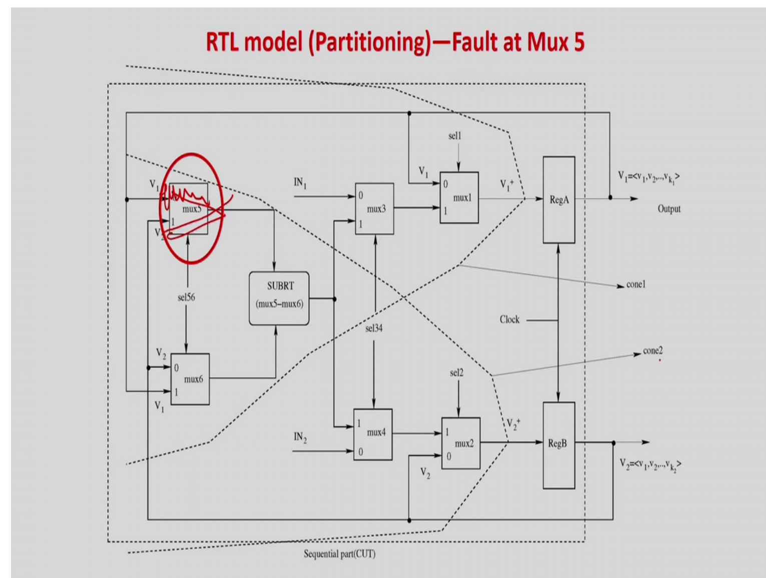
This is the figure I have shown you. So, in a single bit multiplexer the number of gates are this many one, two, three, four, 4 gates are there. And if you look at it, there will be 18 faults. But in the high level fault model this multiplexer has only four faults. Now, I will bring to the (Refer Time: 26:47) clock soft assuming that is 128 bit multiplexer, there how many it will be state 128 in 18 into 120 number of faults will be there as simple as that. But you have to appreciate that for this marks at this RTL level even if it is 128 bit 256 bits the number of faults will be just this four that is what is the driving factor of this RTL base or high level test of embedded systems.

So, even with the rise in the number of bits the number of faults and if the number of faults are less also the number of test patterns will be less and that is why you are going to be the key. So, as I again repeating that now test pattern generation time is important, but not as important as the number of test patterns, because number of test patterns are to be applied in all these chips which will be coming to the 80. So, if you have less number of faults of course, there will be less number of patterns very direct relation. So, in this case, you are actually double (Refer Time: 27:38).

Firstly, at the number of faults are getting reduced at the RTL level, so the test generation time will be less. And if the number of faults are less the number of test patterns will (Refer Time: 27:45). So, you will be a big gain in overall. Problem is that the correlation. So, people have found out is not very bad for this symbol marks 90 more than 90 percent correlation has been obtained for many general cases around 85 to 90 or 80 to 90 percent correlation is there. But gain is time huge reduction in time, and basically the less very very less number of test patterns.

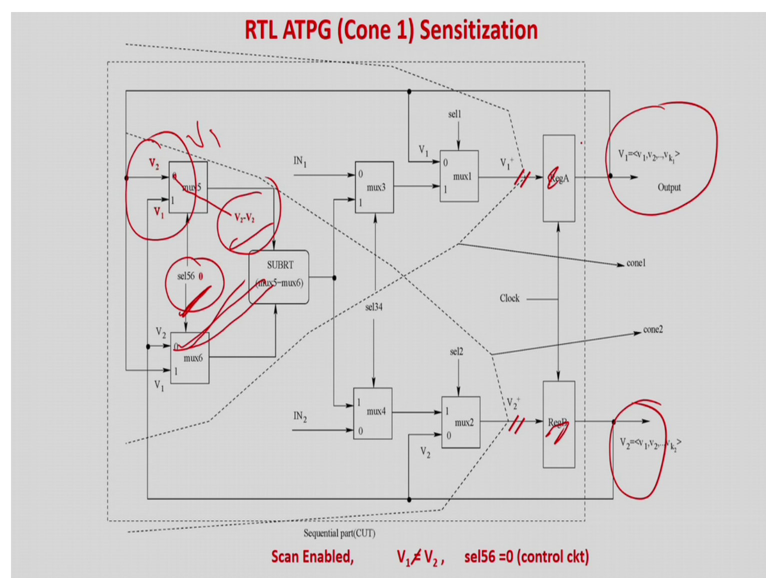
Because as I told you we are mainly worried about this data path for this because more the bit increase in the data path the bits width of the multiplexer, the bit width of the adders the bit width of the computer computing units will start becoming larger. Control design is not much of a concern, because even if the number of bits width increase the controller size more or less remains same. Because they take some of the comparator values and they generate some signals which have to control the multiplexers. So, the size of the controller more or less remain same we are more concerned about the data path right.

(Refer Slide Time: 28:35)



Now, I am going to test you how generate how to test this circuit. We are taking this, this circuit as the as this multiplexer as a failure multiplexer. So, what I have to do, of course, you know that this is a problem. So, how the fault will be sensitized of course, you should ensure that that means, you are taking a case where there is a swap of values or something. So, it is very obvious that you cannot take V 1 equal to V 2; if V 1 equal to V 2 none of the fault will be detected.

(Refer Slide Time: 28:58)



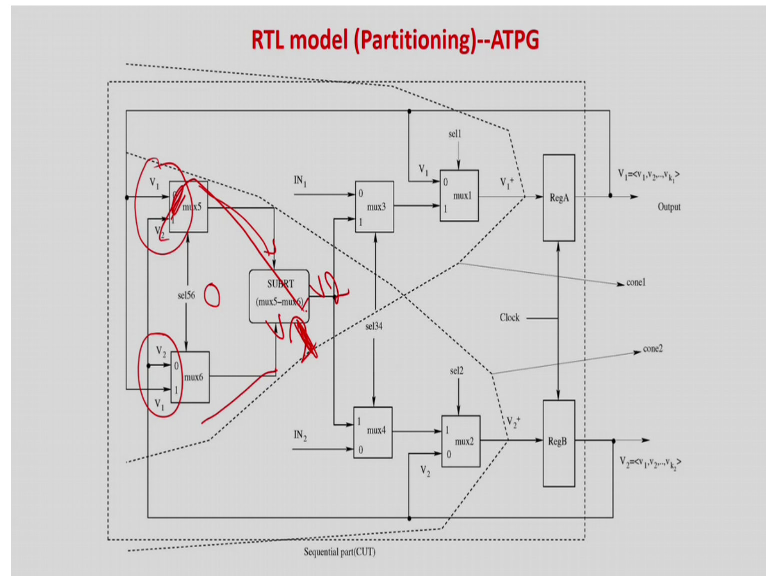
So, first what you will to do, because I have to put some values as V 1 and V 2. So, I will they are also having a scan chain. So, I will tag this scan chain. And what I will try to do is that, I will load the values of register A and register v such that V 1 is not equal to V 2. Of course, you can very easily appreciate the fact that if your taking V 1 equal to V 2, fault will not be sensitize. The fault take any something like this one like whatever be the case you are going to get the same input at the output. This is the case the same input will be coming at the output that whatever be the case this one and this one will be very, very similar.

We are always going to take the V 2 as the output; this one is not going to be coming at the output. Whatever be the case the V 2 will always come at the marks, that is something like this, this is the fact fault model we are taking this one always the input will be coming at the output. So, these whatever be the case if you look at, so always the V 2 will be coming over here. So, of course, to sensitize the fault, V 1 should not be equal to V 2 very obvious. So, why we are making this V 1 is not equal to V 2, and select line we are making as 0.

So, what you are going to get, the output should be here the output should be basically equal to if I zoom it, the output should be V is 0. So, in this case basically V 2 will be coming from sorry V 2 the 0. So, V 2 should be coming from here and in this case basically as it is 0, so V 1 should have been coming from here yeah. So, right.

So, in this case it should be V 2 sorry you should be V 2 minus V 1, but in this case due to swapping of the values of the multiplexer, basically from here also V 2 is going to come and here also V go value of V 2 should be coming from here. If you let me just look at the normal circuit we just look at the normal operation, then it will be more clear normal and is the normal behaviour of the circuit let us look at.

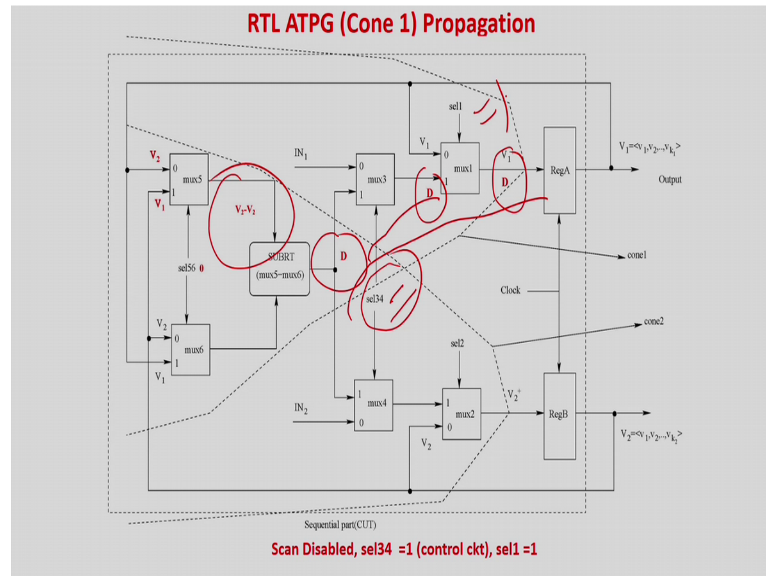
(Refer Slide Time: 30:44)



So, if we look at it is V 1 and V 2 and this is V 2 V 1 ok. And we are saying that our case will be basically the select 5, 6 will be 0. So, in the select 5, 6 equal to 0, so it should be V 1 minus V 2 should be there. But actually the error which you are going to take as the V 2 will also the coming out of here; V 1 will not be going to come out because of the fault. So, you are going to get V 2 minus V 2 in the case of error which is shown over here, this is the error right. So, now, you have looked at it. So, it is 0 and 0. So, V 2 is again coming over here and V 2 is coming over here.

So, normal case if you look at here it was V 1 minus V 2 the normal case, but because of the error it is V 2 minus V 2, it should be a 0. So, this is what the showing you the circuit. So, this is the first case scan chain is done; and you are going to give the value of this one equal to 0. So, instead of V 1 minus V 2, it is basically V 2 minus V 2, this is of the swap. So, for the sensitized.

(Refer Slide Time: 31:47)



Now, we again decouple you can connect the scan chain values. So, in this case, if you look at this V 2 minus V 2, so this is where the value of fault I am propagating over here. Select 3, 4, I make it as the one same ATPG principal. So, this value will be passed over here the value will be passed over here, and I also make select line equal to one all others are don’t care. I can do this because I am doing a standard ATPG base solution

So, first thing is that again I am repeating what I am doing. So, the fault of the multiplexer already I have discussed. So, generally it should be V 1 minus V 2, but it is becoming V 2 minus V 2 because of the error, because V 2 is coming over here although V 1 should come over here. So, I have sensitize this fault by making f 6 equal to 0. So, V 2 minus V 2, and I am also making V 1 an because I am having a scan chain control. So, I am applying the value of V 1 and V 2 I can control. So, I am giving unequal value the fault is sensitized.

So, instead of V 1 minus V 2 the answer is 0 over here the fault is chain scan chain is disable. So, the value is propagated by this path for this I have to make select 3, 4 equal to 1; and select basically this one also select 1 has to be a 1; and select 3, 4 also has to be a 1 as this is the path. So, now it will be last in the flip flop this scan chain register A, and you will get the value of clear the output. And again you are going to test the fault a very simple. So, I am not going to concentrate more on the ATPG, because already we have study elaborately for stuck at fault.

(Refer Slide Time: 33:18)

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

Now, just we need to see what we have achieved. So, the number of RTL faults is some 19. If you are take for this circuit for a practical circuit like this, we have to considered some 8 bit inputs. The number of RTL faults were only 19 the stuck at faults were 844 for a simple 8 bit GCD computer. Minimization of test pattern time and very less number of test patterns, because the number of RTL faults 19 means in the worst case you will have only 19 patterns. Number of stuck at faults 844means it will be high number of test patterns.

So, by using an RTL model, for the 8 bit GCD, you can will have you again see from the example then we have reduce test patterns and also reduce the test pattern generation time. But what are the cost we are paying that is accuracy is lower and many other fault models like bridging and delay which will see later are not covered by this. Because stuck at fault model is very good. It has lot of correlation with other advance fault models. So, all this things we are losing.

But still we should go for it? The answer is yes, because in the next generation embedded system hardware with multiple course nobody will have time to generate because is only for a single 8 bit process 8 bit GCD you can think and a processor we have hundreds of such GCD units and the saving will be much much higher if I going for a RTL model with some at lower lowering of accuracy. But we will in spite of this you have to go for it, you there is no other solution.

So, this lecture, we have try to motivate that and very interesting fault model which is against the concept of structural test, but still it is coming into picture of embedded systems because most of the designs are in software. So, very you can easily you can have the software description of your system and you can put lot of software fault models and test patterns can be generated very similarly like in case of ATPG like the sensitize propagating and justify approach. But the interesting thing we have seen over here is the interesting fault model, which is motivated from software code design.

Thank you. And in the next class as I told you we will be trying to look at testing of embedded codes, the bus and the memory which are using the mainly embedded system design.

Thank you.