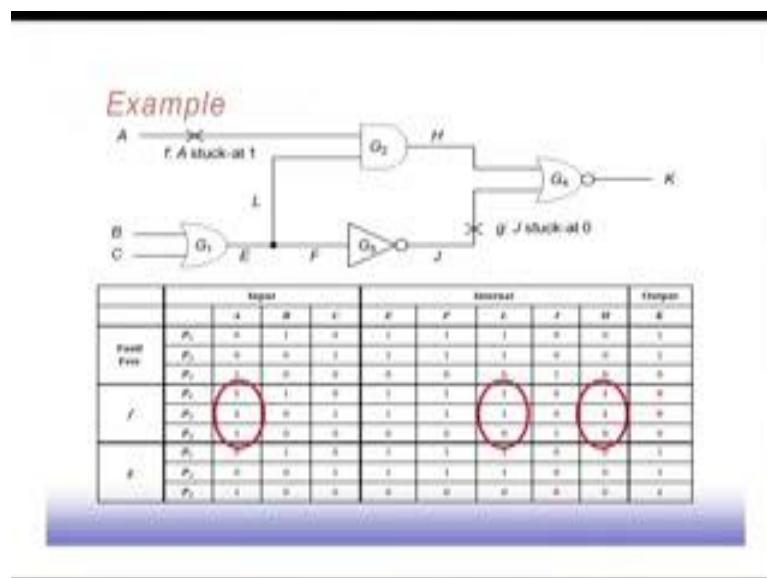


Digital VLSI Testing
Prof. Santanu Chattopadhyaya
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 15
Logic and Fault Simulation (Contd.)

As an example of this parallel fault simulation, parallel fault pattern fault simulation, so what we do is that we take this example and now we see these some circuit that we have discuss previously.

(Refer Slide Time: 00:25)



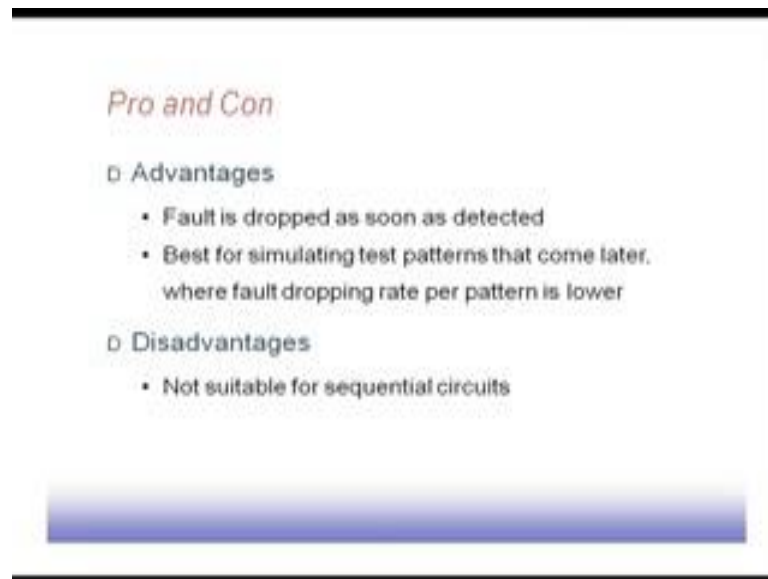
Now here this, fault free response for this P 1 P 2 and P 3, so this 3 pattern the fault free responses are like this that a b c being this. So, these are the fault free responses.

Now, when this fault F is there, what we are doing is in one short we are simulating 3 patterns simultaneously P 1, P 2 and P 3 then in the presence of fault, if we are again simulating the 3 patterns. So, when the fault F is present than this P 1 and P 2, these values are different from values of A, they are different in P 1 and P 2, but for P 3 it is same. So, it is 1, 1, 1. So, these P 3 it does not changes, this way if we do a simulation you see the finally, these P 1 and P 2 could detect these particular fault F particular fault F could be detected by them then we go to the fault G and again for 3 pattern. So, we simulate for fault G and see that this response is finally, turning out to be one here. So, we have got these output different from the good circuit. So, these fault getting detected.

So, this is the parallel pattern fault simulation in parallel fault simulation where we are trying to simulate for multiple faults for the same pattern.

Now, in case of parallel pattern fault simulation, we are trying to simulate multiple patterns for the same point.

(Refer Slide Time: 02:03)




Fault is dropped as soon as detected. So, this is one of the very important advantage because previously in parallel fault simulation the problem was that until unless all $w - 1$ faults are detected, we could not detect, we could not drop those $w - 1$ faults in none of those faults so, but in this case since the simulation is done for a single fault. So, we could detect if as soon as the fault is detected, we can drop that particular point. So, it is based for simulating test patterns that come later where fault dropping rate per pattern is lower. So, begin later on what will happen is that for dropping pattern will for dropping late will reduce as I said that initially even some random pattern.

Will detect a good number of faults, but as go beyond some point then this fault pattern, fault dropping rate will reduce therefore, newer patterns will not be detected will not be able to detect very complex faults or very we call it resistance faults. Disadvantage is not suitable for sequential circuits because sequential circuit it has to go over number of cycles. So, that is one problem. Next we will look into another fault simulation strategy which is known as deductive fault simulation. So, it is based on logic reasoning rather than simulation.

(Refer Slide Time: 03:18)

Deductive Fault Simulation

- Based on logic reasoning rather than simulation
- Fault list attached with signal x denoted as L_x
 - Set of faults causing x to differ from its fault-free value
- Fault list propagation
 - Derive the fault list of a gate output from those of the gate inputs based on logic reasoning



We are not really simulating the circuit, but we are doing some logical reasoning over the test pattern and fault list. So, what we do if fault list L_x is attached with signal x . So, every signal line x has got a fault list L_x attached to it. So, what does L_x contain? It contains set of faults causing x to differ from its fault free value. So, this is basically the faults the circuit that could be detected if you look at point x . So, set of faults that cause x to differ from they are for from its fault free values and another thing is the fault list propagation. So, based on the fault list of input we tried to propagate the fault list to the output using logical reasoning. So, we will how this thing happens say we have got AND gate.

(Refer Slide Time: 04:22)

Fault List Propagation Rules

c : controlling value
 i : inversion value
 I : set of gate inputs
 z : gate output
 S : inputs holding controlling value

| | 0 | 1 |
|------|---|---|
| AND | 0 | 0 |
| OR | 0 | 1 |
| NAND | 1 | 0 |
| NOR | 1 | 1 |

- All gate inputs hold non-controlling value

$$L_z = \left(\bigcup_{j \in I} L_j \right) \cup \{z / (c \oplus i)\}$$
- At least one input holds controlling value

$$L_z = \left[\left(\bigcap_{j \in S} L_j \right) - \left(\bigcup_{j \in I-S} L_j \right) \right] \cup \{z / c \oplus \bar{i}\}$$

Z stuck-at c XOR i

(Refer Slide Time: 04:31)



When we have an AND gate suppose, we have got an AND gate now in this AND gate may be certain faults could be detected at line x. So, we have got L_x associated with it and. So, we have got y where it is L_y now depending upon these lists L_S and L_x and L_y and the values that we have here. So, it may so happen that an output some of these faults will propagate it will be possible to propagate some of these faults to the output. So, for example, for AND gate we know that the controlling value is c is 0 and the inversion value is i is also 0. So, this is basically as we have seen earlier this is a truth

table representation of these logic function is a more compact version. So, it is the controlling value and this is the inversion value.

If any of the inputs are having controlling value then the output is $c \text{ XOR } i$ and if the value is different if none of the gates are having controlling value none of the inputs are having controlling value then the value of a gate is $c \text{ bar XOR } i$.

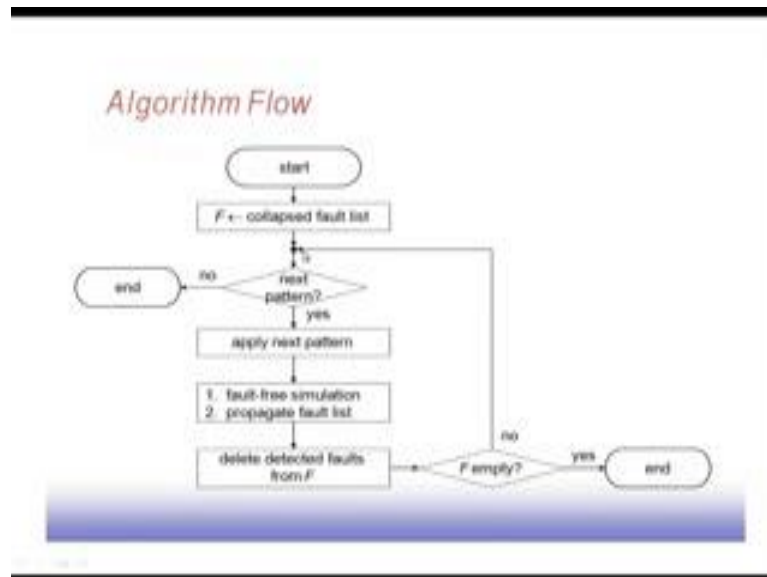
Now, this is the formula. So, if all gate inputs hold non controlling value that is none of the gate none of the inputs are stopping the other value to propagate like when you are talking about AND gate if none of the inputs are 0; that means, it cannot stop one input cannot stop the faults associate with other input to propagate to the output.

If that is the situation that all gate inputs, non controlling value then the list of faults associated with line z that could be observed at line z is given by union of these L_j where j belongs to i , i is the set of gate input. So, whatever faults we could have associated with the inputs of the gate. So, we take the union of them and that becomes a part of L_z and there is something more which is z stuck at $c \text{ XOR } i$. So, this particular symbol is used for representing z stuck at $c \text{ XOR } i$. So, as I was talking, as I was taking the example of and gate. So, assuming that both the inputs are 1, both the inputs are 1. So, they are non controlling value they are holding non controlling value. So, whatever is L_x and L_y they will come to the output plus we can check at this point the z stuck at z stuck at 0 fault. So, these fault we can also be, these fault also can be observe if we can look at the circuit at point z . Now, this is the case when all the inputs are holding non controlling value.

On the other hand, if at least one of the input holds controlling value then there is a problem because it will not be able to see all the faults. So, which faults will propagate, if I have got the S as the set of input holding controlling value then, you have to take inter section of these. So, which ever faults are common between all those controlling points that minus the union of all these faults that are at the gate input other than the controlling value other the faults at the non controlling value that are they are this minus this. So, this becomes the first part and the second part is z stuck at $c \text{ XOR } i \text{ bar}$. So, that can be detected. So, this way we can say, we can see what is these L_z that can be computed and it can be propagated through the circuit. So, this is you see that you are not doing any fault simulation here. We are not simulating the circuit logically, but we are

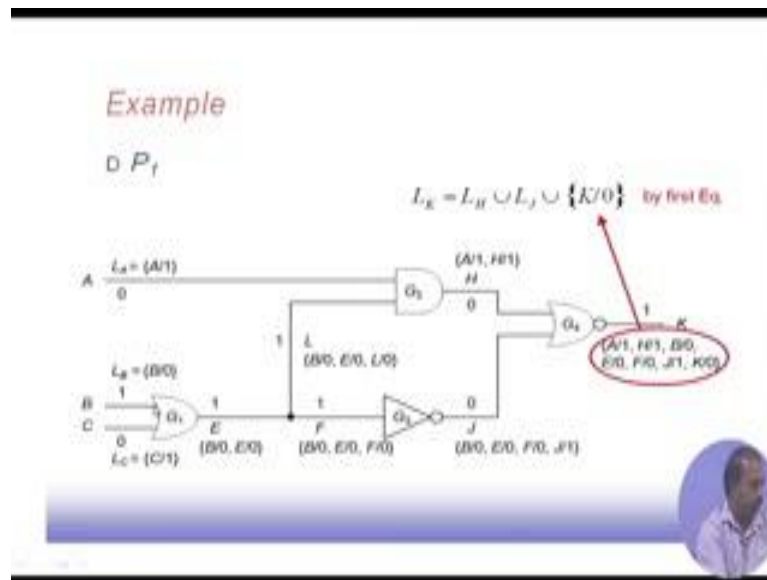
going a set of logic operations, we are doing some set operation and trying to construct the set of faults that could be detected at different point in the circuit. So, when these z becomes the primary output then at the primary output at that primary output we have got the set of fault that could be detected.

(Refer Slide Time: 08:41)



What is the procedure? We start with the collapsed fault list look at the next pattern. So, apply next pattern there is a fault free simulation propagate fault list and then we delete the faults from the faults, fault list and the faults becomes empty. So, there is than it will go back.

(Refer Slide Time: 09:06)

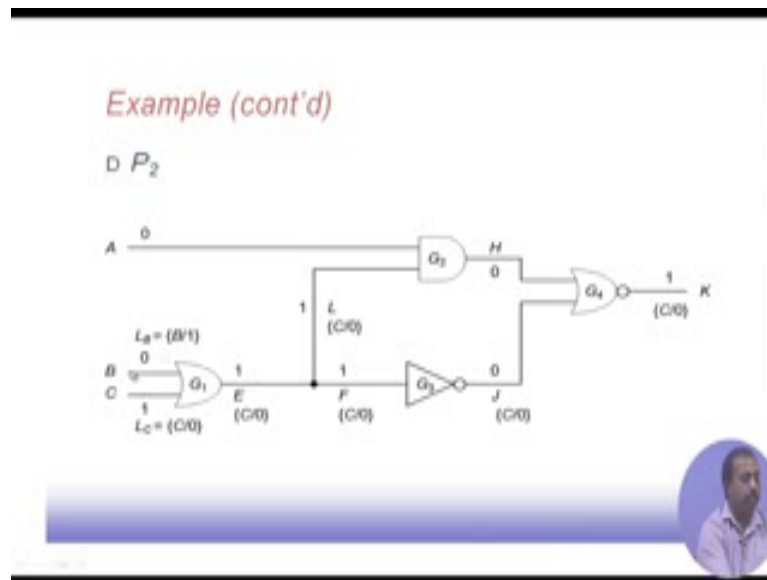


In this case these set P 1. So, P 1 is these L A, so, with a since it is the primary input. So, we have got the set of fault that can be detected here is since P 1 has got 0, 1, 0 has the pattern. So, if I can observe at these point, it is a circuit then the A stuck at 1 fault can be seen by these similarly these point, if I can observe then I can see B stuck at 0, this point I can see C stuck at 1.

Now, since this is an, this is an OR gate and OR gate to one of the input is 1. So, that is the controlling value. So, in this case, we get at the output, these B stuck at 0, E stuck at 0, these 2 faults, B stuck at 0 is coming from this controlling input and, there is only 1 controlling input. So, if will look into these formula. So, it is a inter section of these controlling inputs. So, that is nothing but there is since there is only 1. So, this is B stuck at 0 then we have got these then this minus this. So, that is nothing so nothing comes from their second part and then this E stuck at 0 because E is becoming 1 in the fault free circuit. So, E struggle, these becomes the fault test associated with line E.

This way it continues like say here this particular example, so what is happening is that this is a NOR gate, you see that if both the inputs are 0 and for a NOR gate, 0 is a non controlling input. So, we will get these L of K as L of H union, L of J union, K stuck at these 0. So, that value will be detected. So, this is the formula, if you combine these, use this formula. So, I ultimately these line K, it can be detect all these faults. So, that is detected.

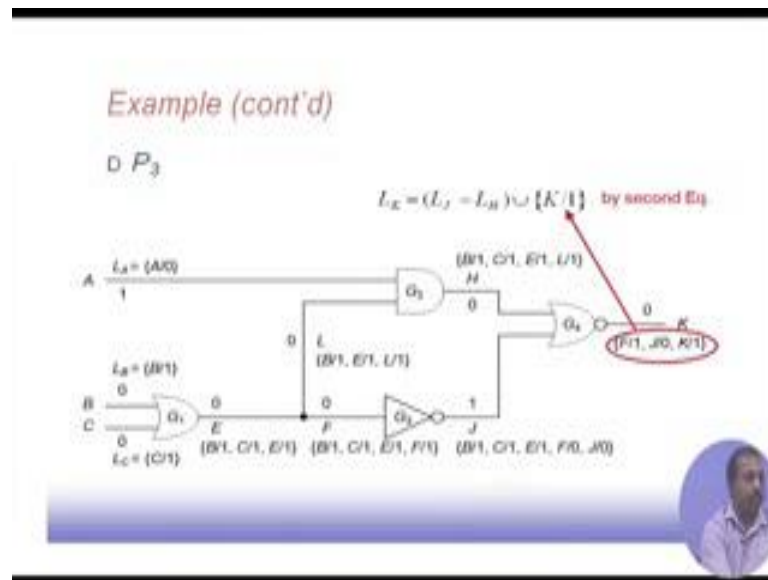
(Refer Slide Time: 11:02)



For pattern P 2, it is, it was 0, 0, 1. So, 0, 0, 1, it could detect these new fault B stuck at 1. So, which was not there in the previous cases since this a stuck at 1 H, stuck at 1 B faults are already detected. So, they are drops from the fault list. So, we have got these B stuck at 1 fault and these C stuck at 1 is also covered. So, C stuck at 0. So, these 2 are the remaining faults that could be detected by these particular pattern at these point accordingly we propagate and since is an OR gate and one is the controlling value.

These comes at the output and the C stuck at 0 and then these E stuck at 1 is already covered. So, E stuck at 0 is already covered. So, that is why these E stuck at 0 does not come at this point, this fault that though these pattern also has the capability of the detecting e struggle 0 at this point; however, that is not counted because of fault has been dropped. So, this way we proceed and then we can see that ultimately in the K, it can detect the fault C stuck at 0. So, this way we can figure out the faults that are detected by different patterns as we proceed through the pattern set.

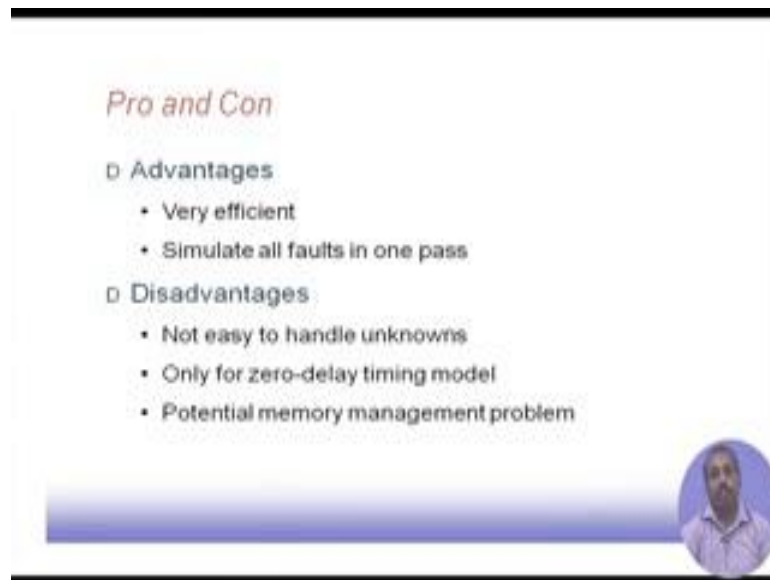
(Refer Slide Time: 12:18)



Then P 3; P 3 value was 1, 0, 0, then it was detecting a stuck at 0. So, so here we just, this A gate some function that 1 input is 0 another input is 1 and since this is a controlling input, this 1 is a controlling input for a NOR gate. So, I have to follow these formula L_j , the controlling input pattern set controlling input fault set minus non controlling input fault sets that is L_h , L_j minus L_h union K stuck at 1.

This F stuck at 1 j stuck at 0 and K stuck at 1, these are the faults that could be detected by pattern P 3. So, this way these deductive fault simulation works. So, this is set of, basically set manipulations and following a set of rules for deriving the output fault list for input fault list of individual logic gates.

(Refer Slide Time: 13:16)



The slide is titled "Pro and Con" in red text. It lists the following:

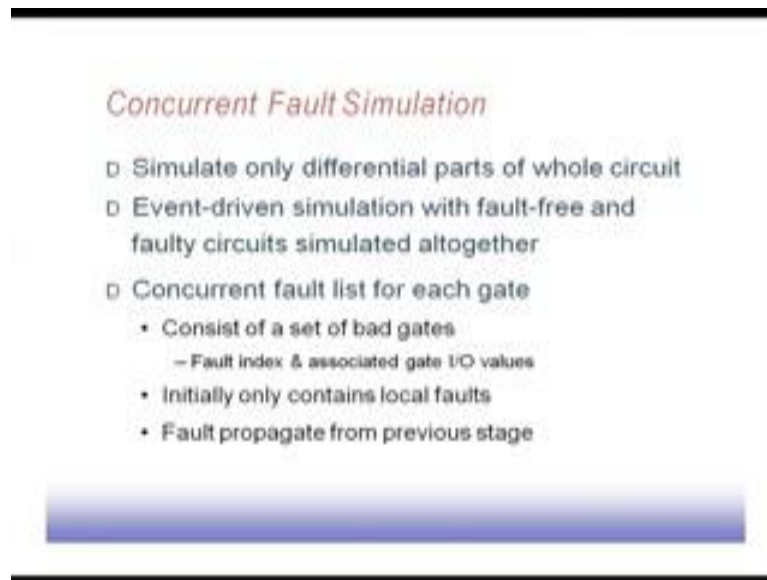
- Advantages
 - Very efficient
 - Simulate all faults in one pass
- Disadvantages
 - Not easy to handle unknowns
 - Only for zero-delay timing model
 - Potential memory management problem

A small circular portrait of a man is visible in the bottom right corner of the slide.

Advantage is very efficient because it is processing all the faults together and all that. So, it is a very good, I simulate all faults in 1 pass. So, it is not that we have to do it again and again, disadvantage not easy to handle unknown. So, this is another one problem because at some point if you get an u then it is not mentioned like how can you we handle and you situation unknown situation only for zero-delay model timing. So, basically if we have got some finite delays then of course, these faults will also take some time to propagate to the output, but that is very difficult to catch when exactly it has propagated. So, it has this deductive fault simulation, it will not help you that way and memory management problem is there because there at a set manipulation ideally for every line it can, it has the potential to detect all faults in the circuit.

As a result, if we have following a simple implementation than you have to, have some huge associated with each line if we are using some link list base manipulation of these faults that can be detected at a particular line then that link list manipulation is again other issue. So, this way we have got both advantage and disadvantages of this deductive fault simulation.

(Refer Slide Time: 14:40)



Concurrent Fault Simulation

- Simulate only differential parts of whole circuit
- Event-driven simulation with fault-free and faulty circuits simulated altogether
- Concurrent fault list for each gate
 - Consist of a set of bad gates
 - Fault index & associated gate I/O values
 - Initially only contains local faults
 - Fault propagate from previous stage

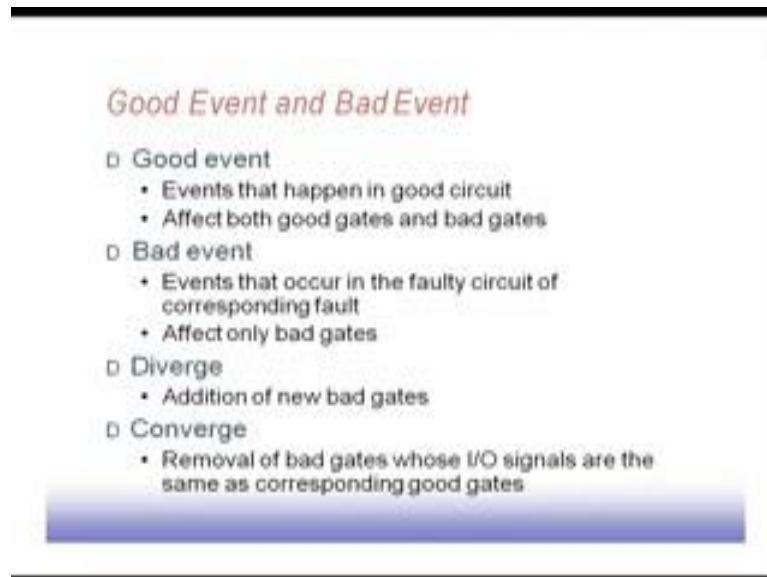
Next we look into another policy which is known as concurrent fault simulation. So, it says that simulate only differential parts of the whole circuits. So, as some sometime will logic simulation we have seen that if we are from one pattern to the next pattern. So, we identify the portion that is changing and for that part only we will do the simulation. So, here also the same condition, if we see that some part of the circuit is going to change their values then know that part only will be simulated.

It is an event driven simulation with fault free and faulty circuits simulated altogether. So, event driven simulation means the if the circuit changes some part changes then only the output will be output of that gate will be simulated and it is also, it is concurrent because fault free and faulty circuit they will be simulated together. So, it clubs the advantages of parallel simulation and this event driven simulation policies. So, concurrent fault list for each gate consist of a set of bad gate that is if something went wrong with the circuit then what happens to these gate.

It may not be due to some problem at the input of these gate alone if may be problem due to some problems somehow else in the circuit as well, but if due to that fault they this gates behavior changes then that will call a bad gate, along with good gate. So, we remember all the bad gates for every gate that we have in the circuit. So, initially we have got only the local faults, but as we propagate the faults from previous stages these set of faults will increase as a result the set of bad gates that we have. So, that is going to

increase. So, all those gate will be simulated parallely. Again the events they can be classified into a number of cases good events and bad events so events that happen in good circuits, so they are called good event.

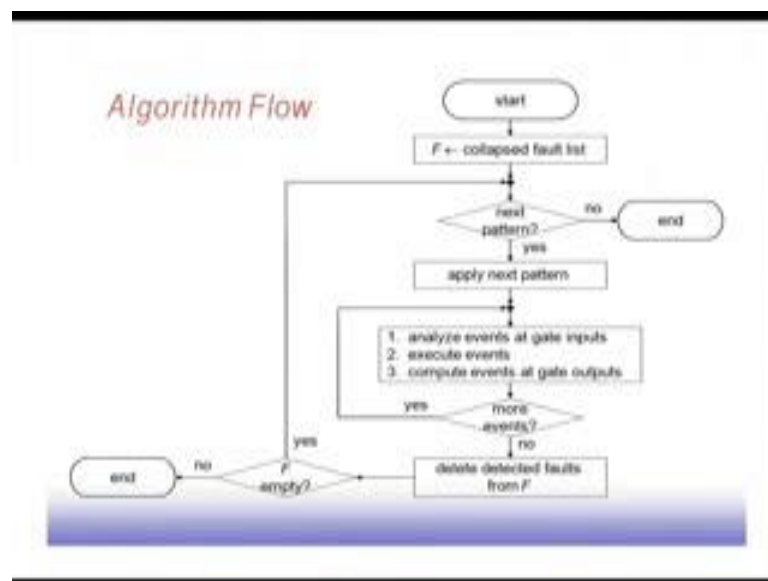
(Refer Slide Time: 16:34)



Basically some input pattern changes due to their some gate inputs will change. So, that is called good events. So, this good events they will affect both good gates and bad gates because the input pattern is going to is go changing then there is some bad event. So, events that occur in the faulty circuit of corresponding fault. So, if due to some fault some event has occurred some output has changed from 1 to 0. So, for example, if certain gates has been assumed to be stuck at 1, as a result the output input of some other gate has changed so that other gate that we are talking about. So, that needs to be simulated.

So, that way, that is the events that occur in the faulty circuit of the corresponding fault. So, that is a bad event and it will affect only bad gates. So, good gates need not be simulated against diverge. So, addition of new bad gates, if new bad gates get created in the process that is called diverge and converge is the removal of bad gates whose I o signals as same as corresponding good gates. So, if you find the some of the gates becomes similar as good gates then there is no point simulating that simulating that output further. So, that will be that will call converge as if the bad gates have converge on to a good gate. So, that bad gate need not be considered.

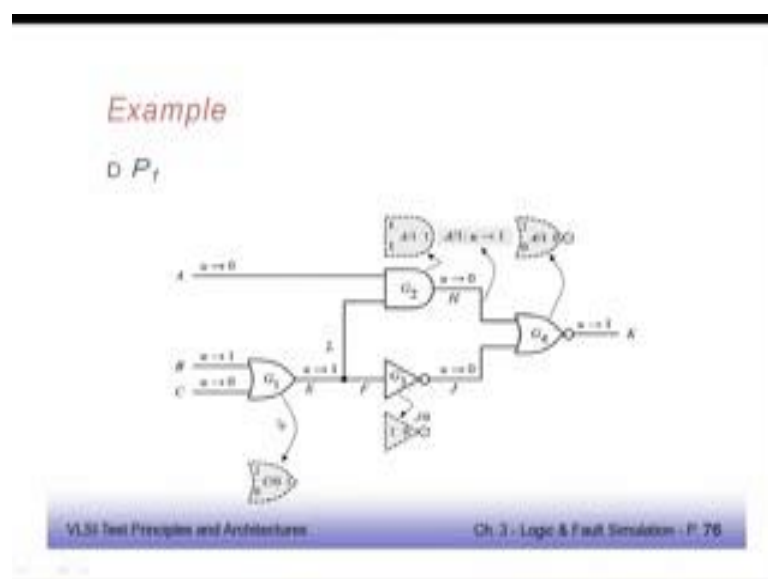
(Refer Slide Time: 18:06)



The flow of the algorithm is as simple. So, it again starts with the collapsed fault list get the next pattern.

Apply next pattern in analyze events at gate inputs executive events compute events at gate outputs more events. So, then it will again analyze events. So, that way it goes; however, as I was telling that there are a number of gates associated with each physical gate in the circuit some one of them is a good gate and others are bad gates, how that thing happens?

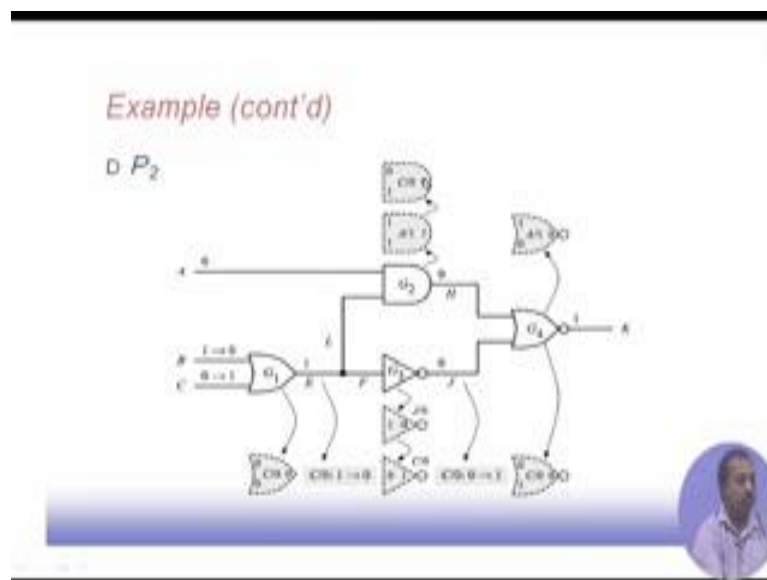
(Refer Slide Time: 18:37)



Let us see this example, suppose we have got this circuit, now initially A, B, C, all of them are unknown, now we apply the first test pattern P 1 which is 0, 1, 0, now when we have got 0, 1, 0, then this input this input may change if this A is stuck at 1, if this A stuck at 1, fault is there then this gate will behave differently like this. So, from u equal to 0, sorry yeah, if this gate pattern is if these fault, is if this input is stuck at 1 then this gate will behave in a different fashion. So, these becomes a bad gate for this case.

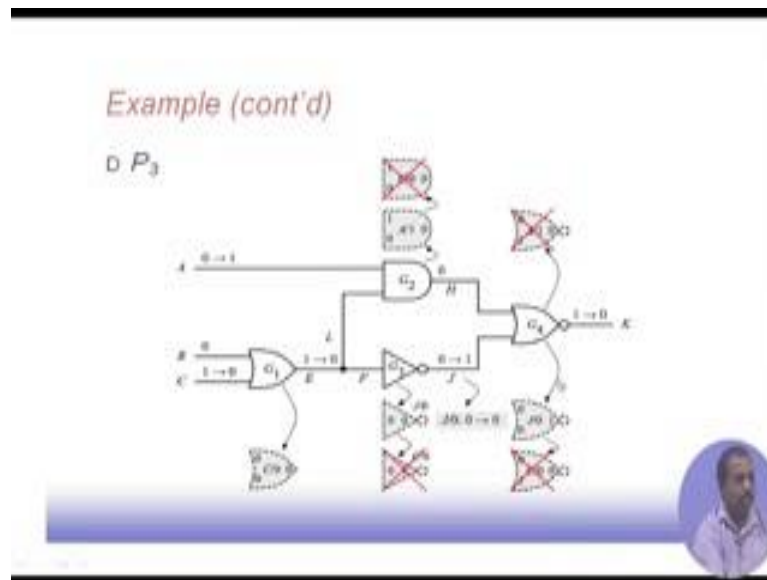
Now, similarly these or gate when it is C stuck at 0, fault will be coming. So, then this will behave in a bad way. So, that is another bad gate for this now this inverter. So, this is J stuck at 0, if J stuck at 0 occurs then this point becomes; this is these becomes bad gates. So, this way it identifies the set of bad gates for patterns P 1.

(Refer Slide Time: 19:55)



Now, to take a pattern P 2 and then we see that we when you propagate we see that many new bad gates have got created these effect of these line C, it was C at the C stuck at 0. So, that effect that bad gate effect has propagated through this line and come to G 2 as a result this one G 2. So, this creates another bad gate C stuck at 0, in that case one input is 0 other input is 1 and C stuck at 0 fault is there. So, it generates 0 as the output, so that is the faulty output. So, this way it can for pattern P 2 we can identify the set of or gates which are going to be bad in this new situation.

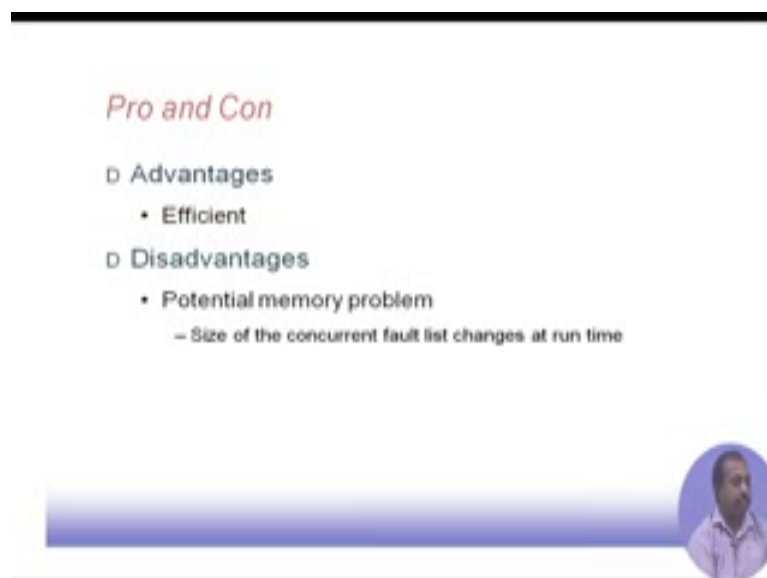
(Refer Slide Time: 20:40)



For P_3 , we see that this is propagating further and you see that at this point we have got these C stuck at 0 coming to the primary output. So, we drop C stuck at 0 from the set that is wherever C stuck at 0 gate was there.

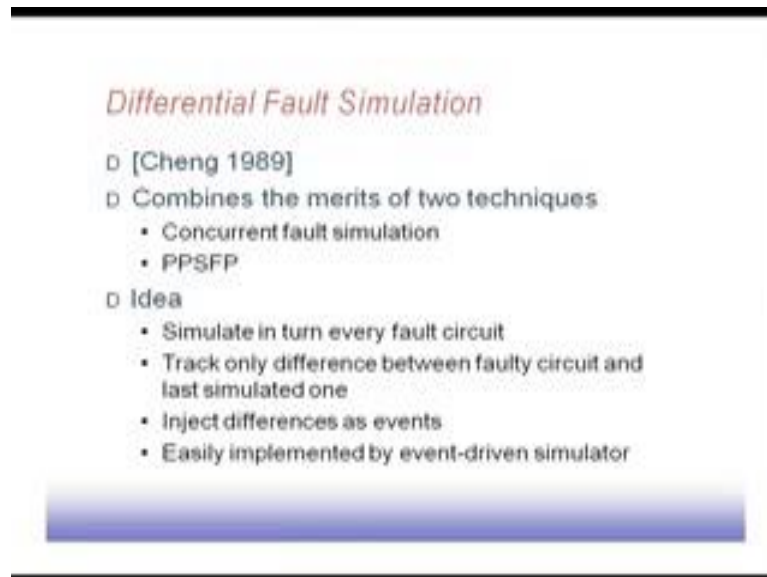
All those bad gates are dropped because they are detected at gate G_4 ; however, similarly this A stuck at 1. So, this is also dropped this because this is detected at primary output. So, this is also dropped. So, we have got this thing that while P_3 simulating P_3 .

(Refer Slide Time: 21:27)



We have found that these faults or these bad gates propagated up to these K. So, those bad gates can be deleted advantage this is efficient definitely for disadvantage is the memory problem because concurrent fault what is happening is that I need to remember a number of bad gates for every good gate.

(Refer Slide Time: 21:48)



Differential Fault Simulation

- [Cheng 1989]
- Combines the merits of two techniques
 - Concurrent fault simulation
 - PPSFP
- Idea
 - Simulate in turn every fault circuit
 - Track only difference between faulty circuit and last simulated one
 - Inject differences as events
 - Easily implemented by event-driven simulator

When we are trying to do that I have to remember a more number of gates. So, more amount of a memory and run time complicity then there is differential fault simulation which combines the merits of these concurrent fault simulation and parallel pattern single fault propagation techniques. So, both the techniques are combined together and we get these differential fault simulation. So, it is a; says that simulate in turn every faulty circuit and track only difference between faulty circuit and last simulated one.

That is why it is using that term differential. So, inject differences as events and then use an event driven simulator for doing the simulation. So, that is a differential fault simulation. So, concurrent and this parallel pattern single fault propagation technique.

(Refer Slide Time: 22:32)

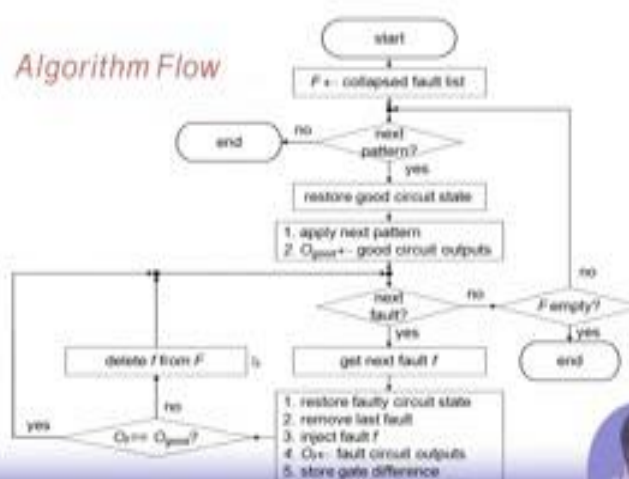
Simulation Sequence

| | P_1 | P_2 | ... | P_i | P_{i+1} | ... | P_n |
|-----------|-------------|-------------|----------|-------------|---------------|----------|-------------|
| Good | G_1 | G_2 | ... | G_i | G_{i+1} | ... | G_n |
| f_1 | $F_{1,1}$ | $F_{1,2}$ | ... | $F_{1,i}$ | $F_{1,i+1}$ | ... | $F_{1,n}$ |
| f_2 | $F_{2,1}$ | $F_{2,2}$ | ... | $F_{2,i}$ | $F_{2,i+1}$ | ... | $F_{2,n}$ |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| f_k | $F_{k,1}$ | $F_{k,2}$ | ... | $F_{k,i}$ | $F_{k,i+1}$ | ... | $F_{k,n}$ |
| f_{k+1} | $F_{k+1,1}$ | $F_{k+1,2}$ | ... | $F_{k+1,i}$ | $F_{k+1,i+1}$ | ... | $F_{k+1,n}$ |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| f_m | $F_{m,1}$ | $F_{m,2}$ | ... | $F_{m,i}$ | $F_{m,i+1}$ | ... | $F_{m,n}$ |

First these; the fault the first for the good circuit, we simulate this for this pattern P_1 , P_2 , up to P_n . So, these G_1 , G_2 is a first for P_1 , G_1 is simulated then we insert fault F_1 then it finds out what are the faulty responses for every fault. So, this is done using a parallel pattern simulator then out of these some of the faults they are they will be in some of the change while doing this simulation, it is done in a differential fashion. So, that is why this; that is if the portion of the circuit changes then only it will do.

(Refer Slide Time: 23:28)

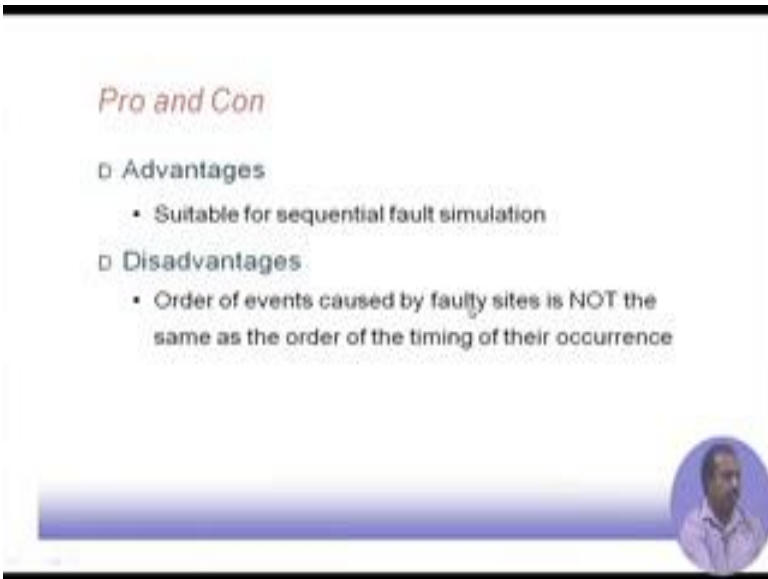
Algorithm Flow



Then after that it will go back to this, it will go back to the again it will start with the next P 2 and start with the good circuit and in insert faults. So, one after other inject fault one after other and that way it continue. So, start with the collapse fault list gets the next pattern. So, restore good circuit state because we know the good circuit value. So, that is there you apply the pattern apply the next pattern. So, we will have the good circuit output. So, this much is fine then if you see if we have any more fault to be simulated. So, if some fault is there it will get the next fault restore faulty circuit state. So, now, this fault has been injected into the circuit and remove last fault. So, whatever fault we have simulated last time. So, that is removed. So, this fault is getting injected into the circuit.

Now, O f is the faulty circuit output response and store the gate differences store the gate difference means that wherever the gate outputs are differing. So, those input those gates are only remember and then we see that whether this raise for the output response is same as good response or not if it is; if they are not same; that means, this fault F has been detected. So, it goes here takes of the next fault and continuous. So, in the simulation process this gate difference value is there. So, that is actually going to help us in doing the simulation less when the fault list becomes empty then if the fault list is there is no more fault then it will be coming to this one and it will see if we pick up the next pattern and it will try to simulate for the remaining faults in the circuit. So, this way it continues.

(Refer Slide Time: 24:57)



Pro and Con

- Advantages
 - Suitable for sequential fault simulation
- Disadvantages
 - Order of events caused by faulty sites is NOT the same as the order of the timing of their occurrence

Advantage: It is suitable for sequential fault simulation disadvantage the order of events caused by faulty sites is not same as the order of the timing of their occurrence. So, what we mean is that this order of timing of their occurrence means they have got some order in the circuit when they, when this circuit, when it try to simulate one particular gate, this is must have been simulated and all that, so that is not happening. So, it is putting wherever get differences it coming. So, it is noting that sometimes create problem in the simulation. So, simulation has to be much more careful because of those bad gates that the difference gates that are created. So, this differences are going to be differences are going to be again sorted based on time and then the simulation has to be done.

We will continue in the next class.