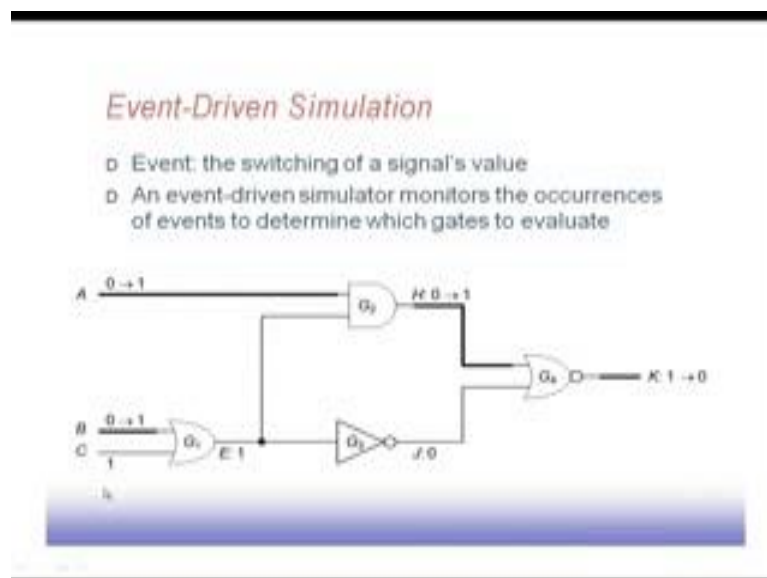


Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 13
Logic and Fault Simulation

Next, we will discuss with event driven simulation techniques. So, event means that when some switching of signals occur like; when a particular signal value changes from 0 to 1 or 1 to 0, so, then that is an event.

(Refer Slide Time: 00:25)



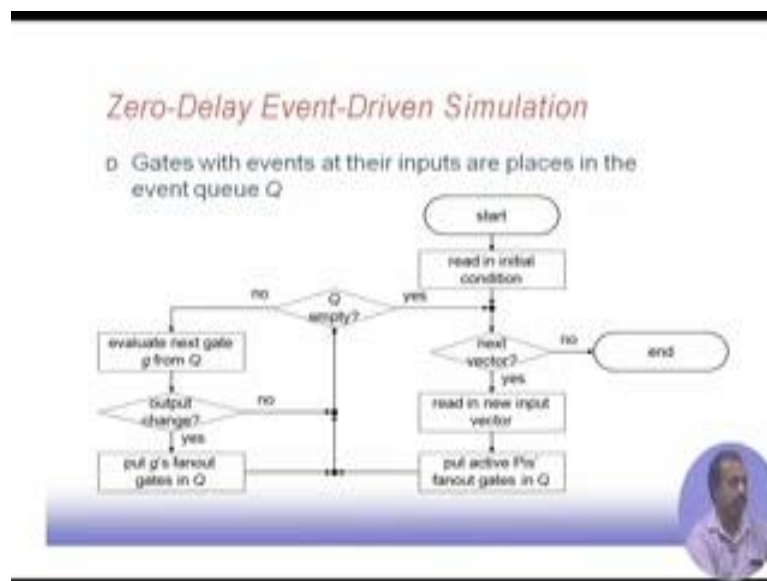
Actually since, it is a combinational circuit as long as input does not change, so, there is no requirement of a simulating the circuit again, because the output will remain unaltered. So, is simulation is necessary only when there is some event. So, if we can identify the events and based on the events we do the simulation and the amount of simulation that we need to do so, that will be much less. For example, in this particular circuit, we see when A, B and C they were at 0, 0 and 1 output K was evaluated to 1.

After that suppose, this A changes from 0 to 1 and B changes from 0 to 1 whereas, C remains unaltered. So, both A changing and B changing both of these are two different events, but for the sake of say understanding. So, let us consider the case that these two values are change simultaneously, that is A and B transitions are occurred together. So, what will happen, after this events have occurred now, this gate G₂ has to be simulated

again before that based on this topological sorting, we can see, we have to simulate G 1 first. So, G 1 will become 1. So, this remains unaltered, as a result if this remains unaltered. So, these inverters output, so, this does not change. So, this part also need not be simulated.

So, the parts that needs to be simulated is this gate, because its input has changed from 0 to 1. After we see that the output does not change, we do not need to simulate any gate to which is fans out. So, this gate G 3 has got input from E only. So, nothing has to be done whereas, from gate G 2 its input has changed A input has changed, so as a result, we need to do simulation of G 2. So, that changes H from 0 to 1 and then we find that for G 4 its input has changed, there is an event. So, accordingly we have to simulate G 4. So, this is a small circuit, so you see that only a few lines need to be they are (Refer Time: 02:31) for simulation for in a large circuit. So, there may be a good number of places where the simulation is not necessary.

(Refer Slide Time: 02:38)

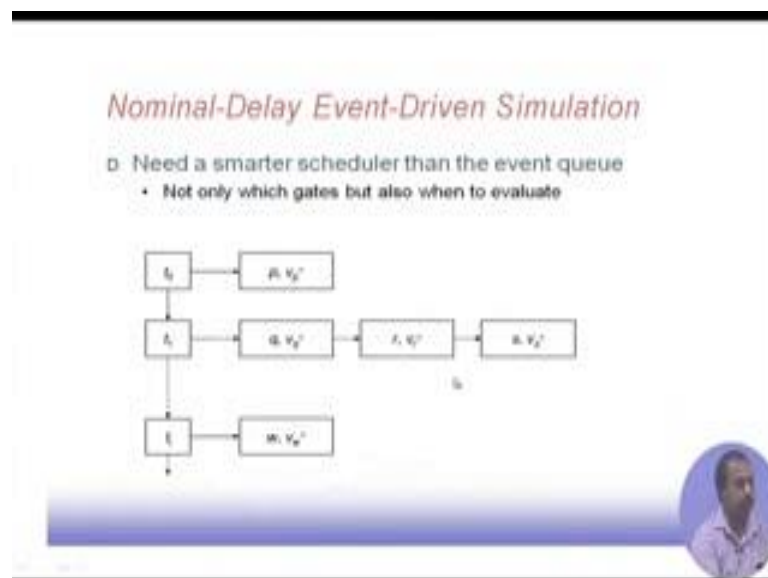


Zero-delay event simulation: so zero-delay means the gates are assumed to be of zero-delay. So, there is no delay. So, it is basically starting, reading the initial condition, then it is checking whether there is any more test vector. If there is any test vector, then it will read the new input vector, and after that the active primary input, they will be determined and their fan out gates will be the point to which this primary inputs connect to that their fan outs. So, they will be put into Q and then, say number of gates have been

put into Q. So, naturally the Q is not empty. So, this check is false. So, it comes to this side. So, evaluates are next gates from the Q, it gate takes the next gates G from Q and evaluate it. If there is output change then, we have to put the fan outs of G into the Q again, because those gates again need to be simulated otherwise, it goes back and check whether there is any more gate in the Q or not.

When this Q becomes empty then; that means, for this current test vector we have computed the simulation, then it has to go for the next test vector. So, in this way it continues. So, as long as there are vectors available in the stimulus file. So, it will go on doing the simulation. Once there is no more vectors. So, the simulation will end. So, that is the zero-delay event driven simulation policy.

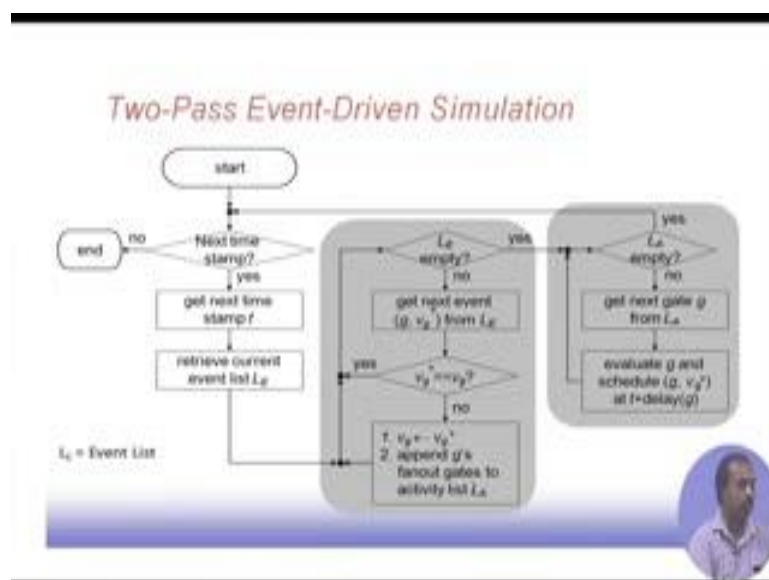
(Refer Slide Time: 04:06)



Then this much we had in the previous class we have seen some of this. Now, next we will consider the case where there is some delay associated with the gates. So, nominal delay event simulation; that means, every a gate it will be assume to have some fixed amount of delay and these delay is called the nominal delay. So, event driven simulation based on this nominal delay. Now, it is not that as soon as a gates input changes gates, gate has to be simulated and output will come immediately, but that will not happen because now the gate has got a finite delay. So, even if it is inputs are changed. So, it will take some finite amount of time for the output to change

So, what happens is that; we can think of a list of time stamps like t_0, t_1, t_i that goes in an increasing order. Now, at t_{naught} suppose the gate the line signal line p has to be evaluated and the new value is v_p plus. So, there is a new value of p in future. So, at time t_{naught} . So, this evaluation will take place at time t_1 . So, we have got three changes q, r and s . So, these are the three changes to occur. So, they are signal values will be changed to v_q plus v_i plus and v_s plus that way. So, you say that there is a definite delay associated with the gate. So, not only which gate. So, this is the point that it is not that we have to identify only the gates that we need to simulate, but also we need to identify the time at which the simulation has to be carried out.

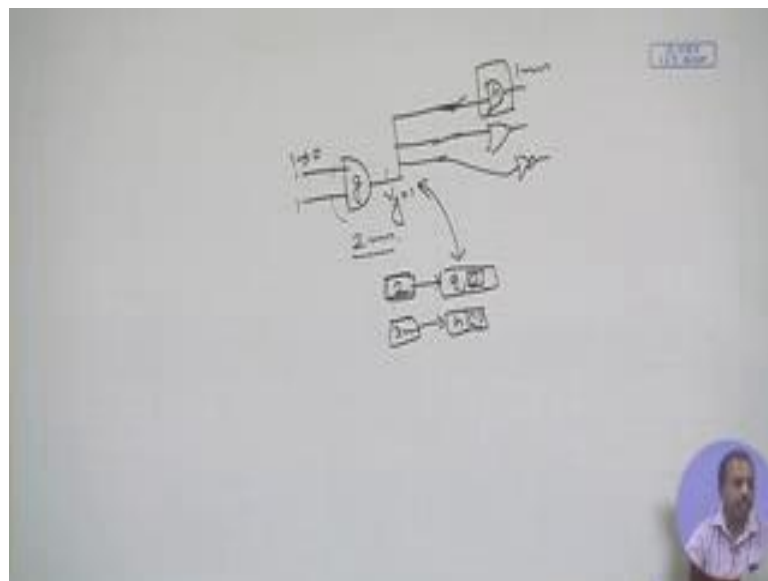
(Refer Slide Time: 05:49)



So, this is basically, a 2 pass event driven simulator that takes care of this time stamp as well. So, we start then there is a check with what is the next time stamp. So, if there are some more time stamp means if there are some events that to occur in future that rescheduled to occur in future. So, then this next time stamp will be have some value if it is yes then it will get the next time stamp t . So, in the previous list we have seen that is t_0, t_1, t_i . So, these times stamps are there. So, as long that list is not empty. So, it will try to get the next time stamp. If it gets the next time stamp, it gets the next time stamp t . So, it retrieves current event list L_E . So, L_E is the event list, in this particular diagram. Similarly, this is one event list. So, this has got a number of events associated with in that are supposed to occur at time t_0 or time t_1 etcetera.

Now, it comes to this point, if there is no event to occur at time stamp t , naturally, initially there will be a number of events, but as we are processing the events then number of events in that list will reduce. So, this is a , we check for this event list is empty or not. If it is not empty, then we get the next event $g \vee g \text{ plus}$ from $L E$. So, for get g ; and what is the next value that should come? So, we get this thing in $L E$. Now, there is a check. Actually, what happens is that maybe, I have got this gate, so, previously this gate was holding the value, this is the gate $v g$. So, this was holding the value $v g$.

(Refer Slide Time: 07:27)



Now, suppose this input changes from 1 to 0. So, previously both were 1. So, output were $v g$ was equal to 1. Now, this input changes from 1 to 0, then depending upon the delay of this gates say this delay is say 2 or 2 millisecond are so, so if I have started at that time 0 at time 2. So, we have got this event to be simulated, the gate g has to be simulated and this new value that will come is this it has gone from 1 to 0. So, this should become 0. Now we have to check whether this 0 that is mentioned here is different from the previous value of this gate. So, these two values will be compared. So, that is what is done here.

So, we check whether this $v g \text{ plus}$, that is the new value of $v g$ is different from the old value of $v g$ or not. So, if it is not different, if they are same, if $v g \text{ plus}$ and $v g$ are same

then naturally nothing has to be done from the fan out of this gate because this part to another successive fan outs parts they will not be changed. So, nothing has to be done

However, if these are not the same; that means, this output has also changed. So, naturally, so, now, we have to think about the fan out gates of this gate g . So, we have g 's fan out gates to the activity list L_A . So, this is the activity list that is created L_A . So, these are list of activity, like what to do? Like in this particular case, you see that if this gate fans out to say three different places feeding, three different gates. Then all these activities destination maybe it is feeding 1 and gate, this feeding 1 or gate this is feeding an inverter, so all of them need to be simulated again.

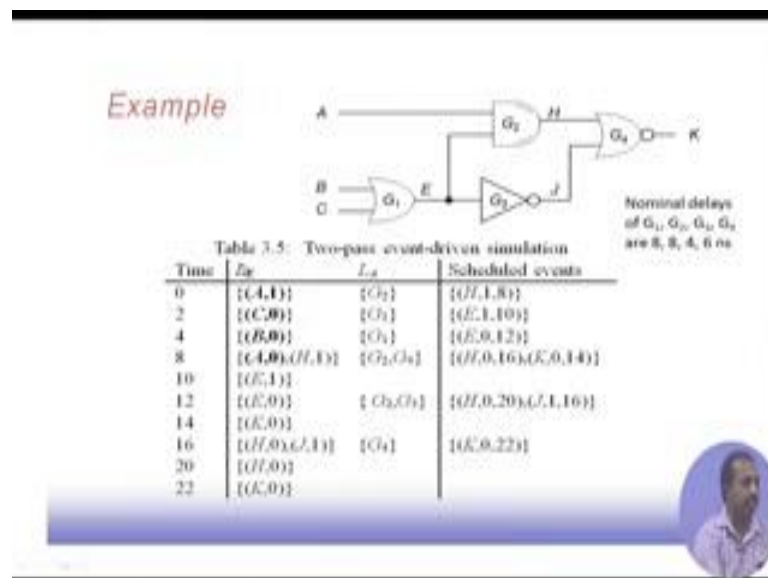
Now, in the activity list we just collect this information, that we have to do this activity, we have to simulate this three gates in future. So, these activities are added. Now, this then it goes to again this check, whether the list of event is empty or not. So, when this list of events has become empty; that means, we have got a set of activities to be done. So, these are the, we have identified the set of activity to be done, basically the gates to be simulated they are identified. So, we get the next node next gate g from this activity list L_A and evaluate the gate and schedule g at $t + \text{delay of } g$. So, as I was telling, now, this gate has to be simulated. Now, when this simulation will be done? If this delay is 2 millisecond; that means, after 2 millisecond time this gate should be simulated that events should occur after 2 millisecond.

So, this with this particular activity is added. Then after say; this new gate, gate becomes the current g . So, this gate has got a delay of say; 1 millisecond; that means, after 3 millisecond, at 3 millisecond, I have to simulate this new gate. So, this entry has to be added. So, this new gate let us call it H . So, H and the value of the gate whatever it is based on the other inputs of the gate. So, that will be evaluated and that will be added to the event list at $t + \text{delay of this new gate}$.

So, this way this 2 pass simulation will go on. So, this is called 2 pass, because in the first pass we are taking this, we are taking an event and then we are finding out all the future activities. Then in the next pass for all those future activities. So, they will again be updated they will be put into the list and then, when this activity list becomes empty; that means, corresponding to this event we have, so, corresponding to the current time stamp. So, based on that whatever event least we had, whatever events are to be

simulated. So, all of them have been completed, their activities have been identified and all that. So, simulation should advance to the next time stamp. So, when this L A becomes empty. So, it is counted here, it is coming back here. So, that the next time stamp is taken up and simulation will continue like this.

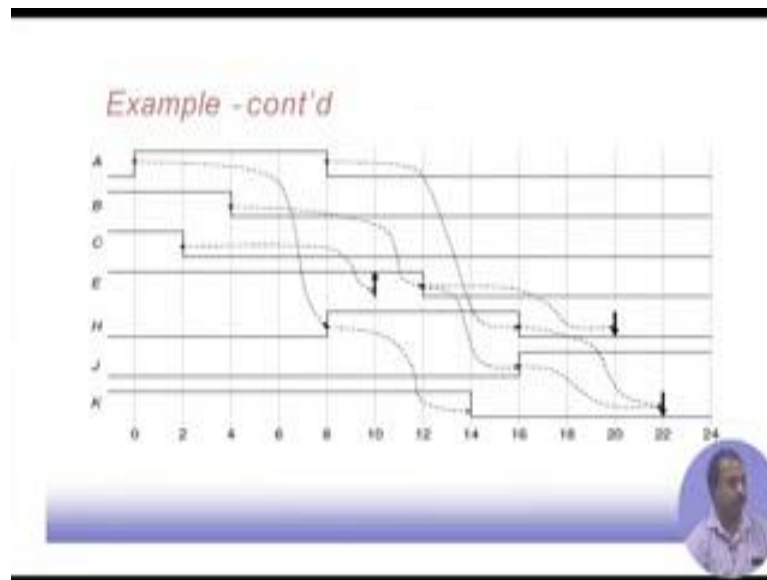
(Refer Slide Time: 11:57)



So, this is an example. So, in it is assume that the nominal delays for gates G₁ G₂ G₃ G₄ they are 8 8 4 and 6 nanoseconds. Now, at time 0, so, we have got this A B and C these inputs are assume to be 1 0 0. So, this is the event, that A becomes 1, B becomes 0, C becomes 0. Now since so, these are say; these are the times. So, A becomes 1. So, A becomes 1 at time 0 and then we need to the activity, corresponding activities is that the G₂ has to be simulated. So, the activity becomes G₂. Similarly, at time 2, C becomes 0. So, the activity for this is again we have to simulate G₁ and B means 0 at time 4. So, again activity is that G₁ has to be simulated.

Now, based on that, when these activities taken up. So, based on this activity, we find out that what is the current value of line H? So, line H is this is the 1. So, this has to be simulated and found that it is 1, but we know that G₂ delay is 8, 8 nanosecond so; that means, this scheduling of this particular event will occur at 8 nanosecond. So, this way this list will grow.

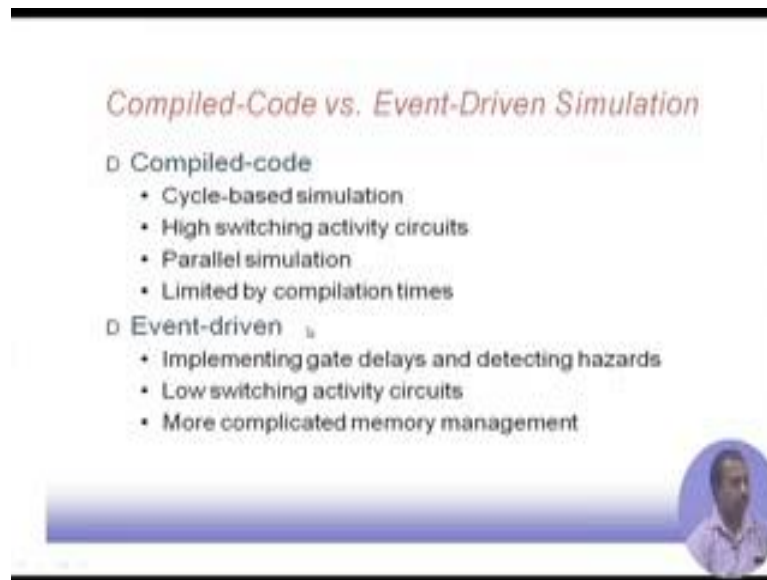
(Refer Slide Time: 13:28)



So, we find out the activities and then accordingly, find out the events and they are put into this case and simulation proceeds further. This is the same information put in a timing diagram for. So, when A has occurred. So, A makes a transition from 0 to 1 at time 0. So, that triggers this particular event.

So, these are actually, all these edges are all events. So, this triggers the event that H will make a transition from 0 to 1 at time 8 then. So, every word says C makes a transition from 1 to 0 at time 2; that means this will be triggered. This will trigger at this point, but it may not have any effect. So, here it does not have any effect, it does not change. So, E remains unaltered though this simulation has to be done and this has not triggered any new action whereas, in this 8 changes, it has triggered a new action, but since here 8 E did not change. So, E could not trigger any new action.

(Refer Slide Time: 14:21)



Compiled-Code vs. Event-Driven Simulation

- Compiled-code
 - Cycle-based simulation
 - High switching activity circuits
 - Parallel simulation
 - Limited by compilation times
- Event-driven
 - Implementing gate delays and detecting hazards
 - Low switching activity circuits
 - More complicated memory management

The slide features a light blue background with a dark blue header and footer. A small circular inset image of a man is located in the bottom right corner of the slide content area.

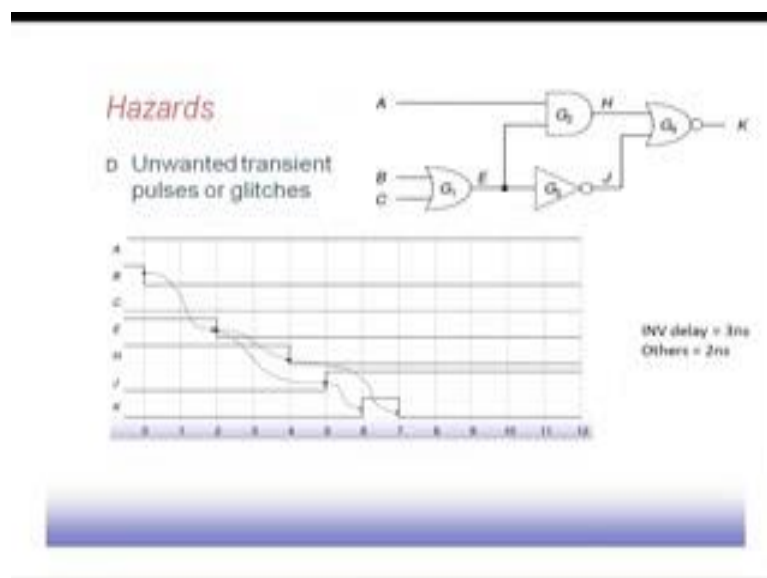
So, if you just compare side by side this compiled code and this event driven simulation. In case of compiled code, it is a cycle based simulation. So, naturally high switching activity circuits will come into picture, because every cycle we are going to simulate the circuit. So, if the circuit has got lot of switching in it then, it is better to go for a cycle based simulation, because if we try to generate the event list then event list will also be very large. So, it is there is no point maintaining additionally this data structure. So, that will increase the complexity of simulation.

So, if you have got high switching activity circuit. So, it is better that we do not go for this event driven simulation whether we do a code base simulation. Say, third important thing is that is code way simulation. We have seen that it can do parallel simulation. So, multiple stimulus they can be simulated parallelly, because taking help of this what size of the processor. So, that way it can help us in running the simulation faster. However, compilation times are necessary, so naturally it will be limited by the compilation time. So, we have to give extra time for doing compilation and anything you change in your circuit design. So, again the code has to be compiled and particularly, if you are doing on a platform, which is if you are doing the simulation on a platform, which is different from where you have compiled it, basically the cross compiler sort of things. Then naturally, we have to again go back to the cross compiler and do that compilation and again come back to the target machine and do the simulation.

So, that way it becomes sometimes conversion. On the other hand if you take event driven simulation the advantage is that it implements gate delay and thus it can help in detecting hazards. So, compile code simulation, since it is a cycle based simulation. So, every time it is changing. So, it will just show that information, but whether any hazard is occurring, like there is no delay actually this compiled code version was basically a zero-delay version. So, we could not get the hazard into picture whereas, in case of event driven simulation since every transition is taken as an event. So, if there is a hazard in the circuit. So, that can that will also can detected and it is particularly suitable for low switching activity circuits, because low switching activity circuit if you are going for a cycle duration cycle base simulation then every cycle will simulate the circuit, but which is unnecessary, because ultimately the circuit inputs or internal gates, they do not change their state.

As a result, it is just wastage of CPU type of simulation and naturally the memory management will be more complex. So, we have got those link list; the event list and activity list. So, maintaining them becomes a bit cumbersome. So, dynamic memory based policy has to be used and it has got naturally its own difficulties. So, this is an example of hazards. So, hazard is an unwanted transient pulse or glitch.

(Refer Slide Time: 17:24)



So, what happens is that you see; in this particular case A B and C, their values are 1 1 and 0, so 1 1 and 0. So, now suppose this B changes from 1 to 0. So, 1 1 0 and B it might

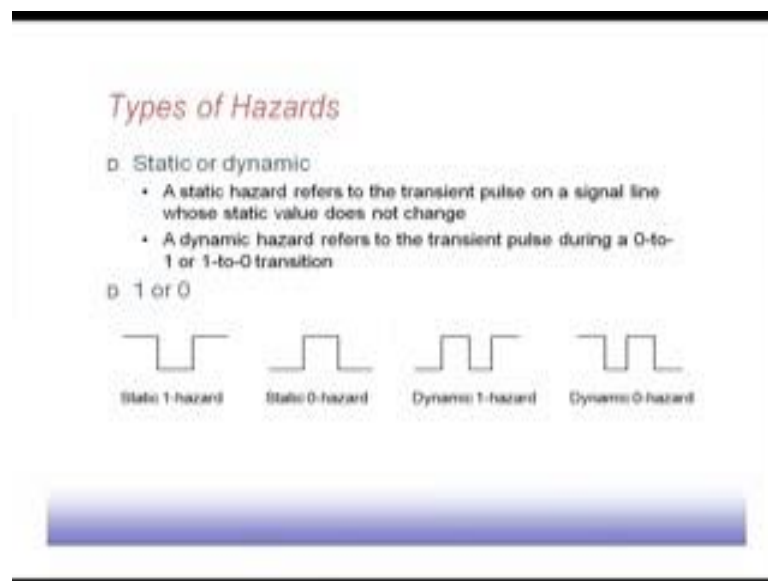
changes to 0. So, as if we changes to 0 what will happen. This will become 0 as a result this H will become. So, let us see what it was previously, it was A 1, B was 1 and C was 0, so 1 1 and 0. So, it was. So, this value was 1, these value is also 1. So, H value is 1 and. So, this nor gate output was 0.

Now, if you when this B makes a transition from 1 to 0 then what happens? This B can goes from 1 to 0. So, as well as, but C complex C is at 0. So, this value becomes 0 as a result is this G value becomes 1. So, NOR gate will continue to remain at 1 K will continue to remain at 0, so this is the behavior. However, if you consider that the inverter has got a delay of 3 nanosecond and other gates had got delay of 2 nanosecond what will happen? This inverter, so, when this B value changes B value changes from 1 to 0. So, this E value change it occurs after 2 nanoseconds, because this or gate G 1 has got a delay of 2 nanoseconds after 2 nanoseconds, E value is changed to E value is changed to 0 and when E value changes to 0 this inverter has got a delay of 3 units. So, J it is changes its level after 3 time units. So, J becomes 1 at this time

Now, this OR gate, so what happens is this or gate has got a delay of 2 nanosecond. So, for some time it finds that both the inputs are 0. So, this is H this is also 0 and J is also 0. So, it becomes equal to 1, this NOR gate value becomes equal to 1 and when it becomes it after sometime of course, this inverter will rise as a result this will fall back and it will become 0 again. So, this is a hazard. So, in this circuit that is present. So, if we have A if you are seeing zero-delay model then we cannot identify this type of hazards, but if we have got, which is quite practical that gates will have some finite delay. In fact, wires will also have some finite delay. So, if you consider those delays into a picture then of course this hazards will come.

So, if this hazards occurred during the circuit operations, then there may be problem, because the value of the signal maybe sample, when that hazard has occurred. So, as a result it will get an improper value. And second thing is that this due to this hazards there will be unwanted transitions in the circuit. So, that will increase the power consumption of the circuit. So, it is better that at the design stage we resort to resolve these hazards, but first of all to identify we have to identify the hazard. So, this way we can identify the hazard.

(Refer Slide Time: 20:44)



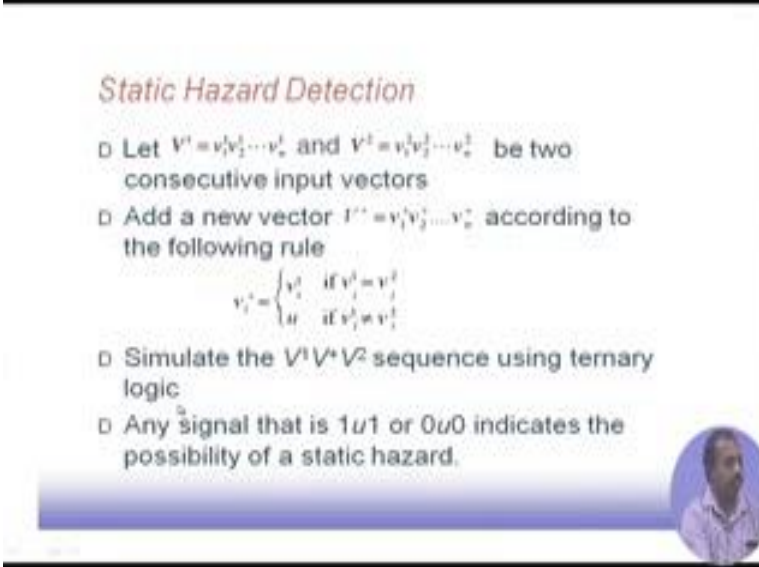
Now, there are mainly two types of hazards; static hazard or dynamic hazard, static or dynamic, where a static hazard is transient pulse on a signal line, whose static value does not change. So, there can be a static 1 hazard or static 0 hazard. So, static 1 hazard means the line was remaining 1, there is a hazard. So, it comes to 0, and then it again after sometime it goes back to 1. So, there is a high to low transition. So, there is a small high to low pulse occurring, which is on a single line which should, otherwise we continually high

Similarly static 0 hazards means, the line should ideally be at 0 level, but due to the presence of this hazard, it makes a transition and it goes to 1 for some time in between. Now this is static hazard, because the initial value and the final value of the signal line they are same. Now if the situation is like this, that the initial value and the final value of the signal line are not same, in that case, it is called a dynamic hazard. So, what was happened in this case? You see the initial value of the signal line is 0, final value of the signal line is 1, but it should make a ideal, it will it should make a transition from 0 to 1, but what has happened is it is made a transition from 0 to 1, but instead of being settle there. So, it comes back to 0 and then again makes a transition to 1.

So, this low to high then again high to low, this particular transition this is basically a dynamic 1 hazard, because it is dynamic, it is going to 1 the signal value is going to 1. So, it is called a dynamic 1 hazard. On the other hand, so we have got dynamic 0 hazard also,

so that is the initial value was high, and final value is low and in between it see this type of transition, the single line goes down to 0, then it becomes 1 and then again it goes down to 0. So, these are the different types of hazards that we can have.

(Refer Slide Time: 22:41)



Static Hazard Detection

- Let $V^1 = v_1^1 v_2^1 \dots v_n^1$ and $V^2 = v_1^2 v_2^2 \dots v_n^2$ be two consecutive input vectors
- Add a new vector $V^+ = v_1^+ v_2^+ \dots v_n^+$ according to the following rule

$$v_i^+ = \begin{cases} v_i^1 & \text{if } v_i^1 = v_i^2 \\ u & \text{if } v_i^1 \neq v_i^2 \end{cases}$$
- Simulate the $V^1 V^+ V^2$ sequence using ternary logic
- Any signal that is 1u1 or 0u0 indicates the possibility of a static hazard.

Now, how do we detect these hazards? So, if we see that suppose v_1 and v_2 . So, they are two consecutive input vectors. So, I have got a circuit that has got n input, and these v_1 and v_2 they are n bit vectors. So, I have been $v_1^1 v_2^1 \dots v_n^1$. So, these are the individual bits of the vector. So, if we want to see that if there can be a hazard between these two. So, what we do? So, we add a new vector v plus, which is given by v_1 plus v_2 plus etcetera, how is it defined? So, it is defined like this that v_i plus v_i plus is equal to v_i^1 , if v_i^1 and v_i^2 they are same.

So, for example, if v_1^1 and v_1^2 are same, then will be called they are same. So, we will keep the same value; otherwise if these two values are differing, if this two input values are differing then we say that we do not know the value. So, this is unknown. So, v_i plus is set to unknown. So, in this way we said this particular vector v , then we simulate this particular sequence of patterns $v_1 v$ plus, first v_1 then this v plus this vector that you have created and then the v_2 , now if. So, this simulation since we have got u into picture so we have to do a ternary simulation.

So, once you are doing this ternary simulation. So, if we find that any signal value has got this type of thing 1u1 or 0u0; that means, there is a possibility of a static hazard,

because in between the value of the signal is becoming unknown. So, it may be 0 it maybe 1, if it is 0 during operation then there is of course, that there is a hazard, if it is 1 then there is no problem, but at the time of simulation we do not know whether this value is a exactly 1 or exactly 0. So, that is why simulator has given you and given it as an u. So this presence of u in between so that identifies the hazard, possibility of hazard.

(Refer Slide Time: 24:52)

Multi-Valued Logic for Hazard Detection

- o 6-valued logic for static hazard detection
- o 8-valued logic for dynamic hazard detection
- o Worst case analysis

Table 3.6: Multi-valued logic for hazard detection

Symbol	Interpretation	6-valued logic	8-valued logic
0	Static 0	{000}	{0000}
1	Static 1	{111}	{1111}
R	Rise transition	{001,011}=0u1	{0001,0011,0111}
F	Fall transition	{100,110}=1u0	{1110,1100,1000}
0*	Static 0-hazard	{000,010}=0u0	{0000,0100,0010,0110}
1*	Static 1-hazard	{111,101}=1u1	{1111,1011,1101,1001}
R*	Dynamic 1-hazard		{0001,0011,0101,0111}
F*	Dynamic 0-hazard		{1000,1010,1100,1110}

So, how do you simulate this multi valued logic for hazard detection? So, for this static 0 and static 1, this 6 valued logic is sufficient. So, because static 0. So, this static 0 is in a 6 valued logic, so we have got this 6 symbols static 0, static 1, rise transition, fall transition, static 0 hazard and static 1 hazards. So, up to this symbol 0, 1, R, F, 0 star and 1 star. So, they are called, they are the symbols for 6 valued logic. Now they are actual values are been assigned in such a fashion, that if we do and or operations with them. So, we will be getting the hazards. Like here, say this, if a line is permanently 0 static 0 so that will be coded in a 6 value logic as 0 0 0. Whereas, 1 in 6 valued logic something that is 1 is actually represented by 1 1 1, a triple 1

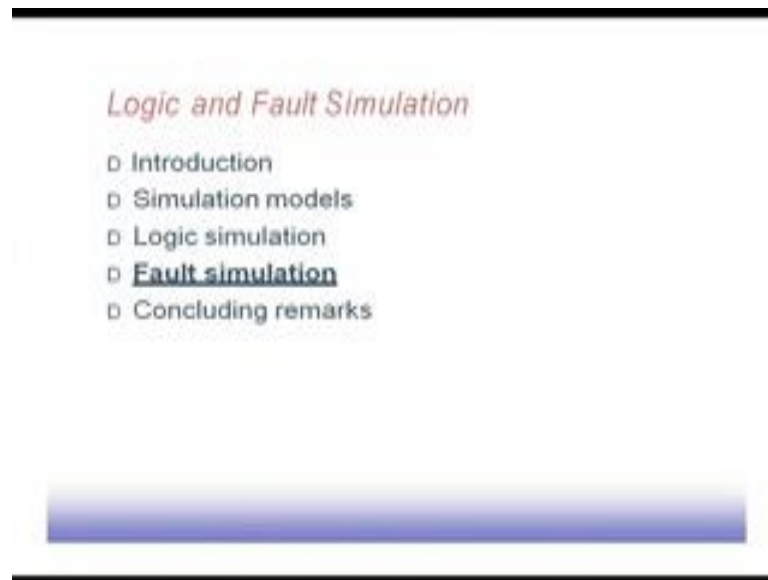
A rise transition is that then it is 0 to 1, it is going from 0 to 1, and in between we have said that since in between if you sample. So, you may get a it 0 or you may get a 1 it depends on the time at which your sampling the signal. So, that is coded as 0 u 1, because in between we are uncertain about the value of the line. Similarly a falling transition, so depending upon the point at which you sample the line, you may at the

beginning you will definitely get a 1, at the end you will definitely get a 0, in between you may get a 0 or you may get a 1. So, that is coded as 1 u 0.

So static 0 hazard means you are getting a 0 at the beginning, getting a 0 at the end, but in between you may get a 0 or you may get a 1. So, when you get this 0 u 0 that represent a static u hazard, and this 1 u 1. So, that is that represents a static 1 hazard. So, first one is static 0 hazard, and this is static 1 hazard. So, same thing if you extend to 8 valued logic then it is like this that 0 is coded as a 4 0s, 1 is coded as 4 1s, a rise transition is represented by couple like this. So, it can be 0 0 0 1 0 0 1 1, because there is a transit rise. So, it is going from 0 to 1. So, you see in all the cases it is a signal line is going from 0 to 1, in between if you sample you may get a 0, you may get you may get 0 0 0 1 or 1 1. So, that way it is coded. So, in this way this value, this logic value have been assigned to different symbols. So, that if you do AND OR operations over them. So, you will be getting the corresponding symbols.

For example if you are doing say and of 2 static 0 and static 1 lines, it is a twice ending. So, you get all 0. So, that is 0. So, 0 and 1 is 1. On the other hand if you are ending with say 0 and this falling transition this one. So, it is basically given by individual bit wise ORing or of this 0 and this rise a falling transition. So, you have to do bitwise ORing of all of them. Naturally we will get this falling transition itself, because ORing 0 will give you 0 only. So, all the three symbols have to be odd. So, as a result you will get again this particular (Refer Time: 28:19) containing three symbol so; that means, falling transition or static 0. So, that is giving as a falling transition. So, this way all this symbols have been assigned in such a fashion, that this AND OR NOT rules, they work properly,

(Refer Slide Time: 28:32)



Next we will go towards a fault simulation. So, logic simulation is ok, we can when we are doing at logic level evaluating the circuit etcetera, but under some existence of faults in the circuit, we need to know what is the output pattern, and if we get that information. So, if that we will check whether it is different from the good one or not, and if it is different from good one, then definitely there we have got the fault detector. So, this fault simulation will actually try to give a figure of merit to like how good or bad is given test and all that.

(Refer Slide Time: 29:14).



So, the way will proceed is like this we will start with an introduction. Then we will see different styles of this fault simulation techniques, serial fault simulation which is the simplest one, then we have parallel fault simulation deductive fault simulation, concurrent fault simulation, differential fault simulation, fault detection, then will also see some comparison between this fault simulation techniques and alternative to fault simulation; finally, we will see some concluding demands.

We will continue with this fault simulation in the next class.