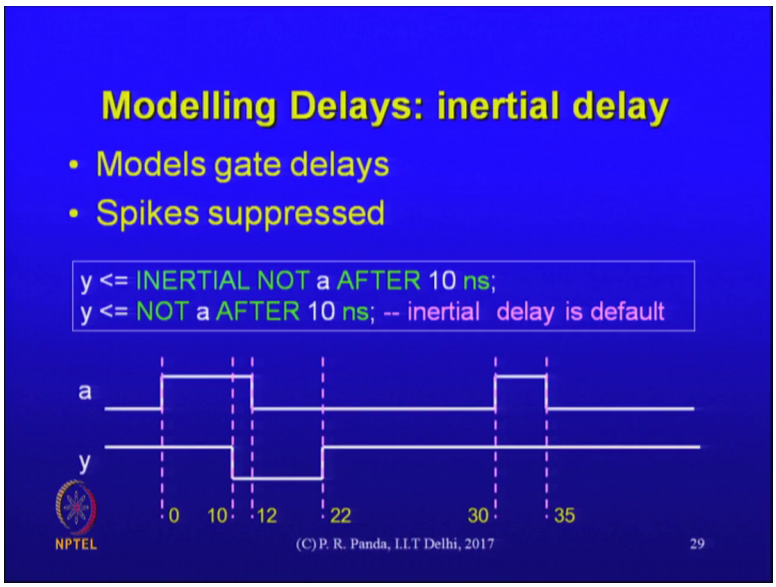
Synthesis of Digital Systems
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

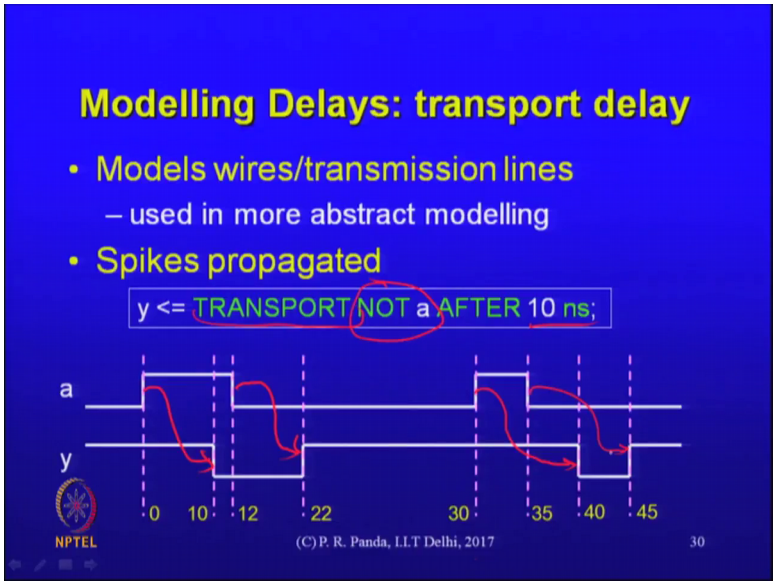
Lecture – 04
VHDL: Modelling Timing - Events & Transactions

(Refer Slide Time: 00:17)



We will continue with the other mode of delay modelling which is called transport delay. This is intended for a slightly different purpose from gate delay of the delays due to propagation delays due to transistors. But this kind of delay is useful in modelling delays through wires or transmission lines.

(Refer Slide Time: 00:42)



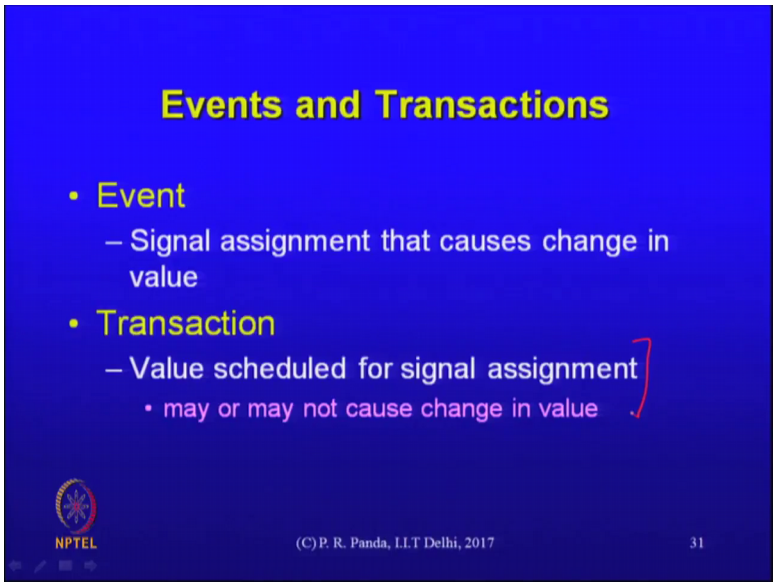
Or in general when there is an abstract model of a component that could be simple or complex. But there is a certain amount of delay that we associate with that component. The difference here being that the spikes are correctly propagated.

The essence of the inertial delay was that if the input is changing too fast considering what the propagation delay is, then the spikes are hidden. They are absorbed. You do not see those spikes in the output. But in transport delay those spikes are visible. So, this is an example you have y takes the value of transport. So, that is the keyword that is used, and functionality is not a after a delay 10 nanosecond. But the presence of the transport keyword makes this difference. So, if the input goes from 0 to 1 here. Then the output changes after 10. The input here changes from 1 to 0, and the output changes after 10. This is fine.

But in the other case where you have the input changing from one from 0 to 1, that leads to an output change for 10 nanoseconds later. But the input changed too fast, right. Input has changed before the propagation delay. Transport delay will still go ahead and schedule the next event for 10 nanoseconds from the time that the input changed. So, that is one fundamental difference between these 2 delay models. Which one to use depends on what is the kind of modelling that you really to do. Normally if it is CMOS gates that we would like to model the delay off then we would use inertial delay. But if it is

something more abstract little higher level of modelling, then we could use transport delay. It could also be that you have buffered, long buffered wires or something like that then to transport delay could be used for modelling.

(Refer Slide Time: 03:01)

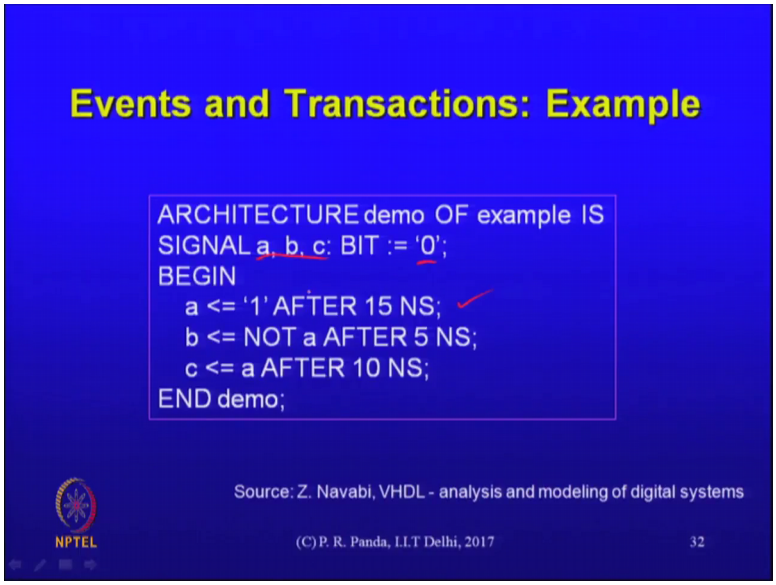


So, these are the 2 modes, with that much background about how the delay is modeled, it is worth taking a look at the internals of an HDL simulator. And understand a little bit from the simulators point of view what is happening, when we assign values when we change values on signals and ports and so on. So, as we do our modelling we need to have an understanding of how it will be interpreted by the simulator of course. So, 2 things, one is an event occurring other is a transaction being posted or scheduled within the simulators framework. Event is said to have occurred if we have a signal assignment that causes a change in the value of that signal. So, it does not matter what the type of the signal is, if the value is changing at a particular time then we say that an event has occurred at that time.

Transaction refers to a value being scheduled for signal assignment. This is internal to the simulator. not necessarily visible to us as the user or even as we observe the waveform of the simulation. It is not immediately obvious we have to do something to take a look at what this means also what is the difference let us try to understand. A value

is scheduled for signal assignment, it may or may not cause a change in the value of the signal itself.

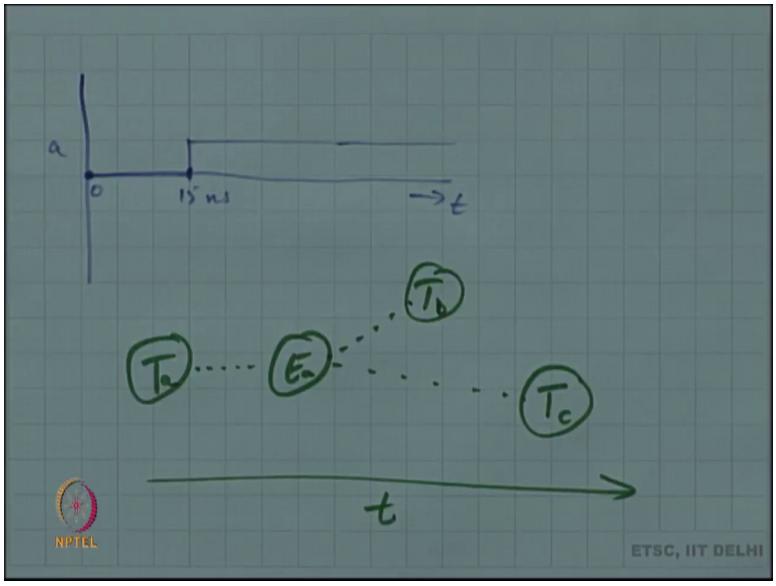
(Refer Slide Time: 04:47)



When that happens let us just take a look. But these are the 2 elementary concepts that are used for delay modelling. Let us understand the difference the way this works and finally, how it translates to interpreting a inertial delay and transport delay in an inertial simulator using a simple example like this.

I did not show the entity here, but it is not very important a b c are 3 signals that we have in this design. And these are initialized to 0, and I have 3 assignment statements a takes the value of one after 50 nanoseconds. What kind of waveform does that result for a?

(Refer Slide Time: 05:25)



So, you have the time 0, this is time. For signal a you just have so, remember it is initialized to.

Student: (Refer Time: 05:31).

To 2 0. So, what waveform are we talking about. Here when I say a takes the value of one after 15 nanoseconds.

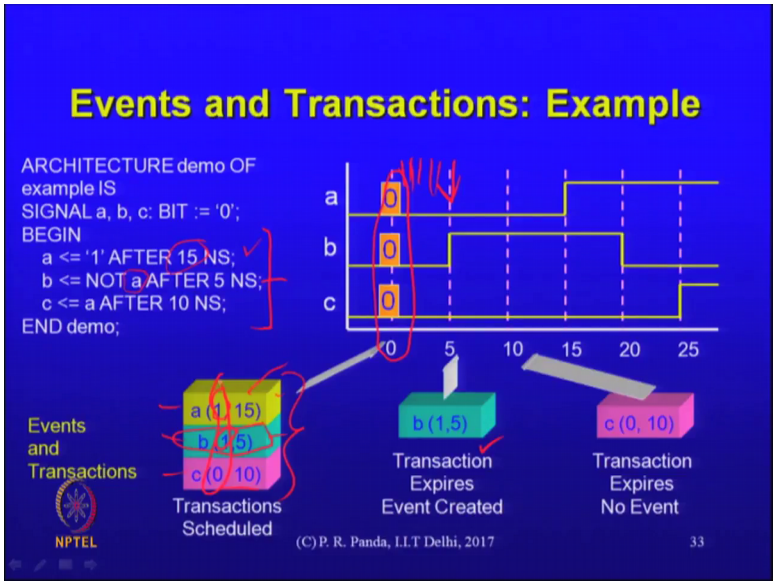
Student: (Refer Time: 05:39).

So, if this is 15, then it stays what it was it stays 0 for 15, but it takes the value for one out later on what happens?

Student: (Refer Time: 05:56).

It stays that way there is nothing else that happens. So, for the rest of the simulation time that is the waveform for it. B takes the value of not a after 5 nanoseconds. That also should be obvious 1 can infer it from the waveform form a itself. And then I have c takes the value of a after 10 nanoseconds. That is all that is there in the design. Let us just rise through the simulators waveform, but also in the process understand these differences. First of all, between events and transactions and then later on between inertial and transport delays through this example.

(Refer Slide Time: 06:34)



Here is my waveform. I have all the values being initialized to 0 at time 0. So, a as we saw takes the value of one. So, there is a transition at time equals 15 nanoseconds from 0 to 1 for a. But let us just understand it from the simulators point of view. We are assigning a value of 0 at time 0.

That too is a new value that too is a change in the value because there was no value before time 0, since simulation started at time 0 new transactions are scheduled for each of those signal. So, it means that at time 0 all of these statements are executed. This is consistent with the overall semantics. When will that signal assignment be triggered? We said that it will be triggered, if anything changes on the RHS of that assignment statement.

But of course, at time 0, this is a special case of if there is a variable or if there is a signal on the right-hand side, that value did change at time 0. Because there was nothing before and at time 0 there was some initialization. So, that is a change in value that should trigger the statements. not merely those statements in which some signal is there on the RHS. But also, other statements like this which are actually not sensitive to any signal.

But still all the signal assignments are executed once at time 0. So, that just to start off the simulation. Executed is interpreted as what actually happens when we say signal assignment is triggered. Here is what happens. Transactions are scheduled in the

simulator for all the 3 signals abc this notation here is just saying that a value of one is scheduled for a time 15 nanoseconds from wherever we are now. So, that 15 there is a relative time.

It is the same as when we say 15 here what we mean. It is 15 from wherever that execution is happening, wherever that statement is getting triggered. So, such a transaction a 1 15 is what is now stored in simulators b. The bunch of transactions are scheduled they are stored in the in the internal queue of a simulator. And so, this is one of them, a 1 15 is one transaction. B 1 5 is a transaction corresponding to that statement being executed at time 0. Does that make sense? A value of 1 being scheduled on to b after 5 nanoseconds based on what that statement is. Why is the value one being scheduled?

Student: (Refer Time: 09:24).

Currently a is 0. Write the new value has not yet appeared of course on a. So, it takes the current value whatever is there, that current value is remember is a new value because at time 0 everything is new not of 0 is 1. So, 1 is scheduled on to be after 5 nanoseconds, fine. C takes the value of a after 10, that results in that transaction 0 10. Current value of a is 0, and it is being scheduled after 10. So, these are the 3 transactions that are entered into the system at time 0.

So, when time changes, maybe some new events occur. Then we will subtract some transactions from there removed from transactions and add possibly new transaction, that is how the simulations working goes, yeah.

Student: (Refer Time: 10:12).

Events are created on a b c just by default because this is time 0. So, as of now we have only talked about transactions being scheduled event has not been created other than just those default events of everything being 0. So, since I have 3 transactions here, a being one at 15, b being 1 at 5, c being c in at 10. We just proceed from here. We forward the time this is an event driven simulator. The way the simulator works, is it keeps track of all the transactions currently that have been scheduled into the future. And picks up for processing which one?

Student: Earliest.

The one that is scheduled for the earliest, it works in a way that is intuitive, but it is an important mechanism for simulations. Specifically, what it means is this is 5 nanosecond, right. It seems as though out of our 3 transactions that are scheduled, we should be picking up this one for processing.

Next, but as a consequence of that you see what we are not doing is we are not evaluating the state of the system at all these intermediate steps, right. Conceptually this is what event driven simulation is about. Which is we do not divide time into fine grain intervals and trigger the simulator at every time unit. Whether it is every nanosecond or picosecond or femtosecond, we do not do that. We automatically forward the time to whatever is the next interesting event. And thereby retain all the old values for that duration. You could do that because this is a digital simulation. The all types of simulations do not work in this mode, but this kind of a simulation where you can forward.

Time and the assumption is that all the values are retained during that entire period is an important part of the event driven simulation semantics. Why do we do it this way? Of course, this is an efficiency issue from the simulators point of view. If it had done all the intermediate evaluations, would it may have made a difference?

Student: (Refer Time: 12:24) processing time.

It would have made a difference to the processing time; it would not make a difference to the waveform. You would get the same values. In fact, there is nothing that is happening in that system in that intermediate stage. Of course, this is set up in a way that it is to move forward in time. So, if it was not nanoseconds, but these values were seconds would the simulation be faster or slower.

Student: (Refer Time: 12:48) slower no difference.

No difference because the simulator is working the speed of the simulator is working is a function of, what is a function of what? Even what transactions we are putting into the event queue? That is what would be called an event queue. And you pick up transactions

from the event queue and process them. That is only dependent on the frequency at which those events are getting in and getting out of that event queue. In fact, it would make no difference. If all those units were seconds or years or something conceptually it is the same thing that is happening. Then we pick up that transaction b 1 5 and a process it, yeah.

Student: (Refer Time: 13:35).

Yes.

Student: We can know the present state of a (Refer Time: 13:39).

To pick up the b 1 5 operation, do we need to know the present value of a?

Student: (Refer Time: 13:51).

Right, so, but to pick it up do we need to know? No, why?

Student: (Refer time: 13:57) we already.

Actually, that value we have already entered here, right. So, the current value of a will you mean that the current value of a at 5 nanosecond is actually not relevant to this simulation. What well the value that is relevant is; the values that we evaluated for b at the time that the transaction was created. So, that is being scheduled. So, one is being scheduled where did that one come from it came from not a, the well we took the value of a at time 0. That is the transaction that has been scheduled.

So, here it is only the values that we are putting there. We are not putting what function is it of other signals. That function was actually computed when we evaluated the signal at time 0. That is how the simulation proceeds. So, we do not need to look at the current value of a. not that it should not matter electrically, and when it does matter let us say what we need to do about it. But, in fact, the fact that there was no other transaction, no other event and so on between 0 and 5 gives us the guarantee that it is to pick up that old value of a.

A is value has not changed since that time. That is what we are guaranteed therefore, we are able to make that assumption. So, when we pick this up for processing, we say that the transaction expires. So, this what is maintained here is an event queue. And what we picked up from there is the first element, the earliest element that is scheduled in the event queue. So, after I take that away, what would remain in the event queue are the other 2 transactions a and c, the b 1 5 transaction is taken out, yeah.

Student: sir if the value of k would have changed say at 2.5 seconds.

Right.

Student: 2.5 nanosecond.

Right.

Student: When would we have been put the value of b have been evaluated?

If the value had of a had changed in the middle somewhere at 2 nanoseconds, then would this transaction still be evaluated in this way. In fact, what does intuition tells us?

Student: No.

No, but we did not see that happening here. Here actually we are guaranteed, right. That it would not happen the reason we skipped 2.5 nanosecond, was that the transactions were scheduled in a way that such an event could not have occurred. But when it does occur we do need to do something about it. But, in fact, the our specification of inertial delay was that we would be sensitive to such things happening, right where the spike should be suppressed that is what we had said.

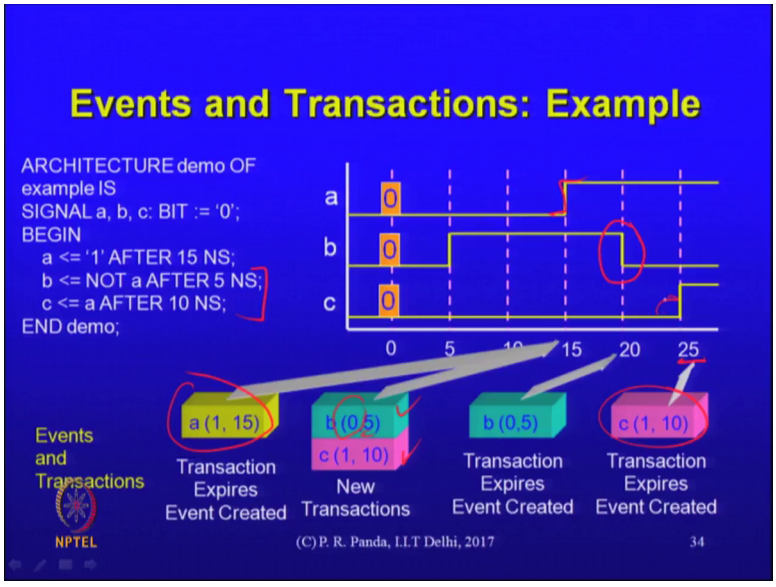
So, we should be able to do that, but how we do that let us just proceed along the waveform we will see an example of that happening. So far this is not happening. And all we do is we remove that b 1 5 transaction, and that is it in the event queue so far. After that I have 2 transactions still remaining to be processed. So, I just pick up whatever is the earliest transaction, which is that c 0 10, because that 10 is earlier than 15 and therefore.

That is the transaction that is picked up if there were one thing I forgot to mention is that when the transaction is expiring. Because we picked it up we, now posted the new value as scheduled in that transaction onto the signal. Now if the value of the signal is changing, then we say that an event has been created. So, this resulted in an event being created on b.

Let us move to the next transaction. I have a 0 value being scheduled on to c at time 10. I pick it up because that is the earlier signal anyway. Earlier transaction to be scheduled I post the transaction. What happens to the value of c? There is no change, because the current value is 0 anyway. This is an example of a transaction. That is scheduled that is processed because it expired it is time has come, but it did not result in the creation of an event, right because the value of c did not change.

It did not result in c changing, as a result if there was some other signal that was sensitive to c. Right it had a c on the right-hand side of the assignment. Then that signal assignment would not be triggered event driven simulation means that, you triggered signal assignments only when the appropriate condition has been met; but here it is okay. So, transaction expires no event is created because nothing changes in the value of c number 1.

(Refer Slide Time: 18:30)



This is the last transaction that is still there in the queue. It was scheduled for 15 the value of a was supposed to be 1. And so, that is what has happened here. And as a result, an event is created on a. Because the current value of a is 0, and the value of a has changed to 1. Let us see what happens later on. Because a has changed, now some new transactions may be created, right. The simulator has to look out for the possibility of new transactions being created, resulting in new statements being executed whenever any event takes place.

So now, that an event has occurred on a, what might happen in the other 2 statements. In fact, both of these statements have a on the right-hand side of the assignment statement right. So, they are candidates. So, new transactions may need to be scheduled for both of them. So, we will evaluate b, the value of 0 is scheduled for b for 5. So, this 5 is a relative time like I was saying it is 5 from the time that we are currently processing.

So, a value of b 0 5 is scheduled that is one transaction and similarly c 1 10 is another transaction. 2 new transactions enter the event queue. Then what happens? 5 nanoseconds later, their transaction expires right on b, is that is the earlier transaction. So, this expires and as a result there is an event that is created on b. So, this is the only other transaction that is remaining it would be expiring at time 25 and the value of c this time does change and as a result, and event is created on c.

So, that is how that simulation waveform at an elementary level goes. Is that clear? The interplay between the transactions on the events, transactions are created when some event occurs on a signal that is on the right-hand side of an assignment statement. It is a possible change in value. We do not yet know whether it is it will actually lead to a change in value or not, you could argue from this example that there was the first transaction that did not even result in an event because it did not change the value of c. So, why did we even bother to create the transaction? And we at the time that we posted, the transaction did we know or not that.

Student: (Refer Time: 20:59).

Right, but in general, you post a new transaction. And at that time, it seems as though it is value is the same as the current value, should you post the transaction or not?

Student: Yes, we should.

Why?

Student: An entry has to be made right for every transaction, so.

Does it help us to post a transaction on a waveform when that value is the same as what is currently there?

Student: Yes (Refer Time: 21:28) in between and that again.

Yeah in general I need to be careful about how I need. So, this is one example it is not here from this example whether it was worth it or not. But let us see other examples related to the correct handling of inertial delays, which did not necessarily come up right here. But this waveform is clear this example is clear hopefully, yeah.

Student: (Refer Time: 21:51) sir if there is a glitched (Refer Time: 21:53).

Yeah.

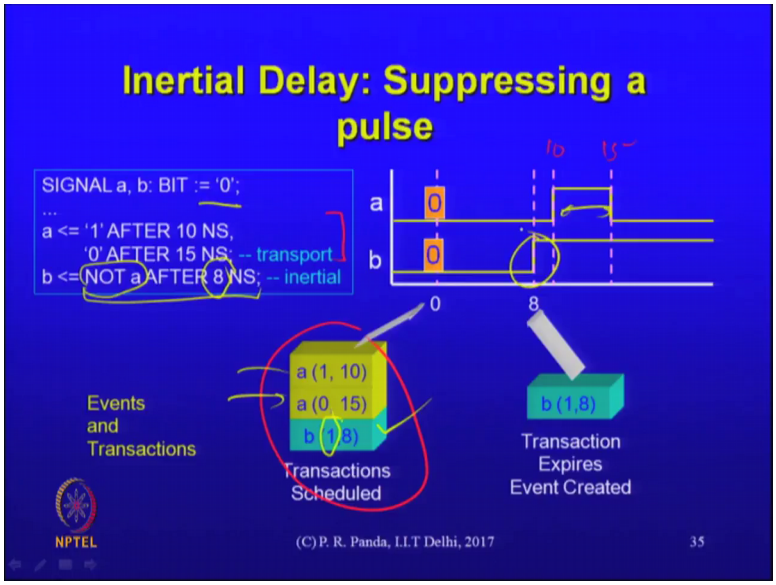
Student: It will go un noticed or

We will handle the glitch situation, right. Here the glitch has not occurred in this example as of now. So, this is sort of the simplest waveform you can think of. But this still illustrates the difference between an event and end transaction; that is important hopefully we understand, that all transactions do not necessarily result in an event, although all events are necessarily created because of trend chance that we post into the simulator, so.

Student: Sir.

Yeah.

(Refer Slide Time: 22:25)



Student: (Refer Time: 22:26) we say that an event was created.

Yeah event like we said is occurring when the value is changing on a signal. That is the simple definition. As this is a simulation specific meaning of course, but that is what an event is.

Student: is it might be possible at when a transaction expires 2 variable one of which is having it is value changed.

Yeah.

Student: One of the other one is constant. So, in that case also an event would be created.

When a transaction expires 2 different assignments are sensitive to that same signal, right.

Student: But transaction expires one signal due to one variable, it is value changes the other variable it value it is value remains constant.

Right.

Student: So, an event would be created.

So, you are saying that a transaction is created; well the transaction is posted first. Later on, it leads to an event being created, As a result of that event maybe 2 other transactions. So, maybe let us let me say call it a transaction on a, event is created on a as a. Result of it some transaction is created on b is posted. Both the transactions are posted as of now. And some other transaction is posted on c where this is our time. Both of these need not result in events on b and c. That is all right the event has occurred on a, because it is value changed. Now at that time we will see that as of now we are actually inserting all the transactions into the event queue.

And if it turns out that the value that we had posted is the same as the current value, then there is no event. That happened in b then there would be no event on b. If the value of c changed here, then there is an event here maybe they say that there is no event on b that that is all right the, but the way this process hopefully is clear. That at any point in time we maintain a bunch of such transactions that we had here.

That is our event queue. And we simply pick up for processing the earliest transaction that is scheduled process event, right. So, but of course, we have to handle the spikes or pulses in an appropriate way. Particularly we already understand how that inertial delay works. The way we are processing the transactions and events has to be consistent in a way that we are able to handle a inertial delay, we are also able to handle transport delay. But the underlying mechanism for handling both of these is the same; it is just transactions and events.

But the way we are interpreting them would be different. So, here is a statement. That says a takes a value of one after 10, and 0 after 15, resulting in a waveform like this. It goes up at 10, right. That is 10, and at 15 it goes down. That is my waveform specification.

The commas could be given to essentially specify a waveform on a particular signal. I gave 2 values here you could give any number of values and any number of clauses could be introduced there effectively leading to a very complex specification of a waveform as part of a single signal assignment statement. But by definition that kind of a waveform specification refers to a transport delay. Inertial delay remember means that if

you have a bunch of values being scheduled the last one is what is picked up the older ones would be thrown out.

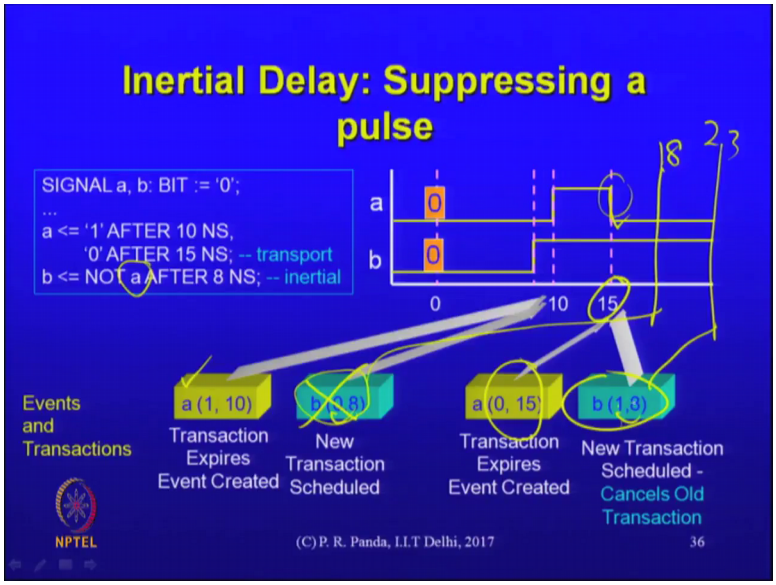
You can not specify a waveform if you are constrained by inertial delay. So, if you are specifying a waveform it means that even if you did not write transport, it is actually treated as transport delay. Otherwise an entire waveform would not appear. The other one though, I have b takes not a after 8.

This is a normal inertial delay specification, but you see the way this example is designed such that this propagation delay is more than the width of that spike there. So, the issue that we had referred to with respect to inertial delay and how it is handled should be correctly handled now. This is a test case that should exercise that. How do we go about doing it? We have 2 transactions at time 0, we have initialized the values to 0. Even if you did not initialize it, the default value for the bit data type is 0. So, we have 0s on both a and b at time 0.

Then I have one transaction a equals 1 being scheduled at 10, and another transaction a equals 0 being scheduled at 15. So, both of these are entered here to be processed at different times. That is how we would handle transport delays. But as of now it is just a specification, and these I have these 2 transactions being entered into the queue. And I have one other transaction on b, because I have not a being evaluated at time 0.

At time 0 all the statements are evaluated. So, not a is evaluated. So, back value of one is being scheduled on to b at time 8, right. So, 8 nanoseconds from time 0 so that is 8. So, that is my event queue as of now. These are the 3 transactions that are there. That is the first transaction that is picked up for processing. So, that results in a transaction expiring a 8 and an event being created on b at time 8. Then what should happen?

(Refer Slide Time: 28:07)



After that I have the other 2 transactions remaining in my queue still. So, a 1 10 is the next transaction that would expire, every time an event is created we have to look at our statements to find out which subset of the statements may need to be evaluated, right. And possibly transactions might be posted on them. So, because a changed and I have a occurring on the RHS of that statement, I should have a new computation and possibly a new transaction being created. So, I have a b 0 8 that is because the new value of a is 1. A value of 0 is computed for b. And that is my new transaction b 0 8, it will be processed whenever it is time comes.

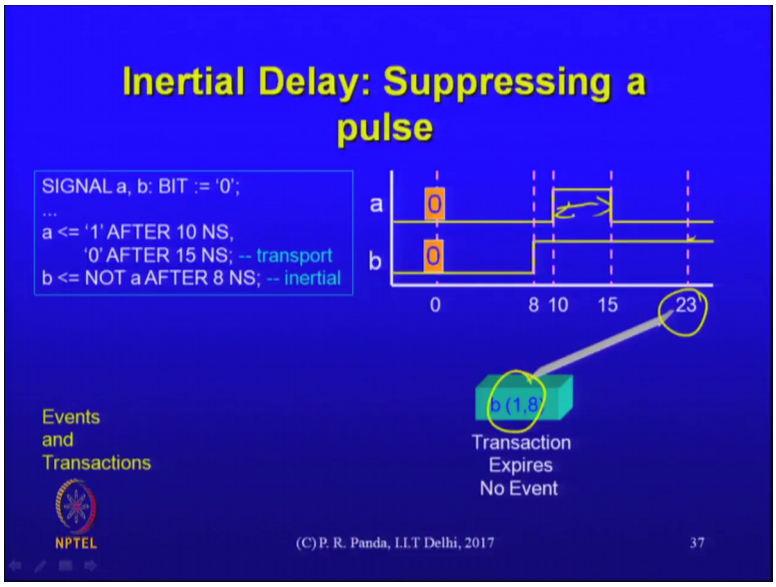
So, since our current time is 10, the scheduled time for it is processing is at 18. But what has happened is in the meantime, before we could reach 18 the value of a has actually changed, back to. So, this is the transaction 15. That expires first right that b is value that was scheduled at 18, does not expire, right.

This one expires first. This is an interesting situation what should happen here; that transaction on b is still there it is pending to be processed, but the older transaction on a expired and a new event is created which would lead to, what, which would lead to that statement being evaluated again. So, b 0 it was there, but now a new transaction is sought to be entered into the event queue, right. Here is a situation that we need to resolve in some way because a value of 0 is scheduled for the future.

But the current value of the input requires us to schedule something else into the future. That is further away this one 8 is further away from the 0 8, because we are at 15 now, right. So, there is something that scheduled at 18 that is this one. But this one would be scheduled for 23 8 from here.

But this is a conflict and the inertial delay seeks to resolve this conflict. In what way should we resolve it? We will just cancel the old transaction. That way that spike does not get propagated to the output. So, we will say that this transaction that was there is actually canceled. And the b 1 8 is what is still retained that one 8 remains in the queue.

(Refer Slide Time: 30:48)



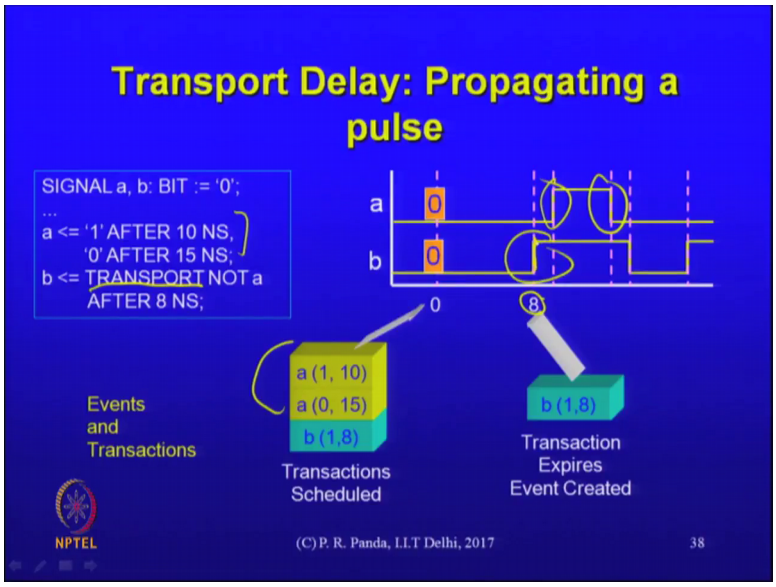
So, that is scheduled here at 23. And in fact, the value of b was already 1. Remember, it never went down because the earlier transaction was cancelled, right.

So, we have 23 when this transaction expires. In fact, the value is already, what it is it does not result in an event being created. Neither the earlier event was created as a result of a going up. Nor is the new event created as a result of a going down. This is the mechanism the simulator uses to suppress a spike.

So, that essentially this 5-nanosecond pulse on a is considered the equivalent of a spike, because it is small compared to the propagation delay of that gate at which it is the input. So, that is the simple mechanism the simulator uses to handle inertial delay, which is that

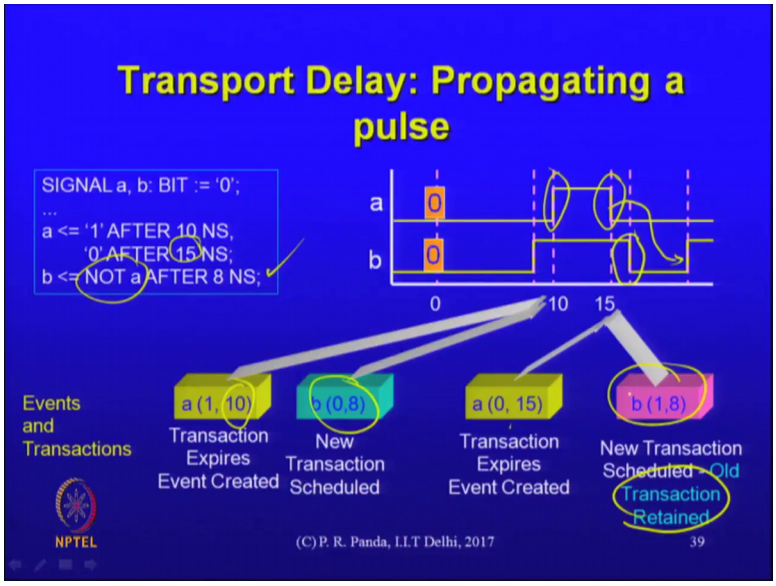
if a new value is scheduled on a signal for which a transaction is already posted in the event queue, then we just cancel the old transaction and we enter the new transaction. Any questions on how this is handled? This is a simple and elegant way to handle.

(Refer Slide Time: 32:03)



The inertial delay certainly. What if I had b equals transport? So, the earlier delay was an inertial delay and we know how to deal with that. Let us quickly go through this waveform this is no different in the earlier stages. Actually, for a nothing changes in the processing of a is the same it is the same statement. So, I have this transaction being scheduled you would both of these transaction being scheduled onto a at time 0. B 1 8 is scheduled for time 8, that is the same as earlier.

(Refer Slide Time: 32:43)



At time 8 that transaction expires, right the transaction for b expired first. And the value of b is changed. So, I have an event being created on b. Let us move on. There is a transaction expiring on a at time 10. And an event is created at time 10. Now because an event is created on signal a at time 10, what happens is that statement gets triggered.

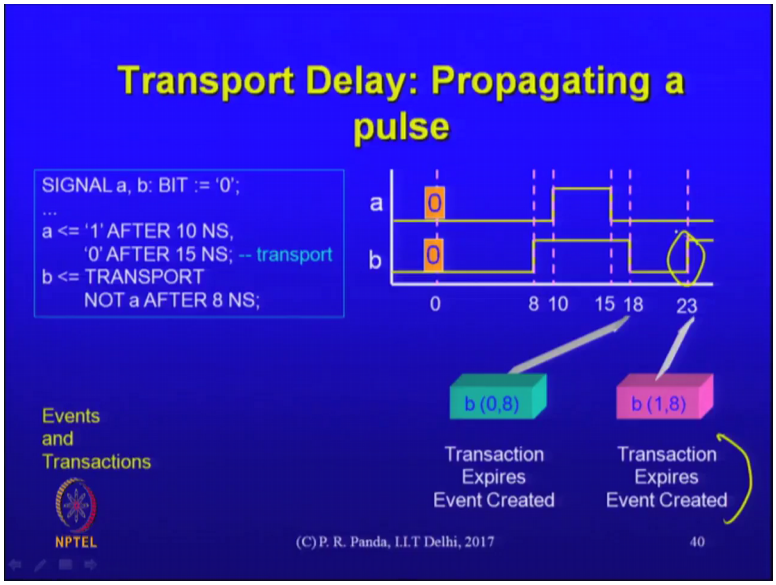
Because of the event so that statement is triggered and a new transaction b 0 8 is scheduled; 8 units of time for 8 nanoseconds from 10 which is at scheduled to be processed at 18. We move on. So, I have 15. So, at time 15 I have the other event occurring on a. And because of that event I would have this statement being executed again, which means that my new transaction would again be processed just as earlier. But this time what we say that is that we will retain the old transaction, we will not cancel the old transaction. Through the simple mechanism, I will just permit the old transaction to be processed at the time that it was scheduled which was 18, right. So, at time 18 we pick up b 0 8 for processing and the value of b changes, event is created on b.

Then at time 23 which is as a result of this yet as a result of that 0 15 transaction. I have the new event being created on b. So, the difference was that earlier, we canceled the old transaction we process the new transaction, but the new transaction did not result in a change in the value of a because the value was the same. Here we just retained the old transaction as a result of that there were 2 changes 2 events that occurred on b. It is a

simple mechanism to handle inertial versus transport delay. So, the mechanism is almost exactly the same. The way the events and the transactions are processed they are common, but depending on whether their signal that is being assigned to is an inertial delay assignment or whether it is a transport delay assignment. We would make that simple change in the way it is processed in the signal queue. So, when the new transaction is posted.

In the inertial case, we will first look at the queue to find whether there is a pending event already on that signal. If it is there then we cancel the pending event. In the transport delay case we do not bother. What is there we just go ahead and post a new value anyway. That is it that is the very simple way to discriminate between the 2 kinds of delay modules. So, what you see on the output waveform is very different, but the way it is handled is just through a simple change of semantics.

(Refer Slide Time: 35:32)



Of course, at the end, the final transaction that is processed is this 1 at 23. I have that event being generated and that is the last transaction for this example. But this should give you an idea of how the simulator works. It is like an infinite loop in which the status of the event queue continues to change depending on what events are occurring. So, the moment an event occurs, maybe it triggers a bunch of statements each of those statements, that event would sensitize possibly, those are all executed.

And as a result of the execution, new transactions may be posted. At different times in the future depending on what those after clauses where whether it is 10 or 15 or 8 and so on. Then so, we just modify the status of the event queue we pick up 1 transaction to process from the event queue.

That leads to possibly a bunch of different signal assignment statements being evaluated. As a result of that, a bunch of new transactions would be created. All of them are again put back into the event queue for processing. And what would the final step be. Well the final step would be that you pick up the event; you process all of them there may be multiple events that are expiring at our current time right. So, after that the all you have to do is you just forward time to the next scheduled event, right.

Student: Sir.

Yeah.

Student: 2 transactions are simultaneously scheduled, right. Both of them are scheduled after 8 nanoseconds or such kind (Refer Time: 37:09).

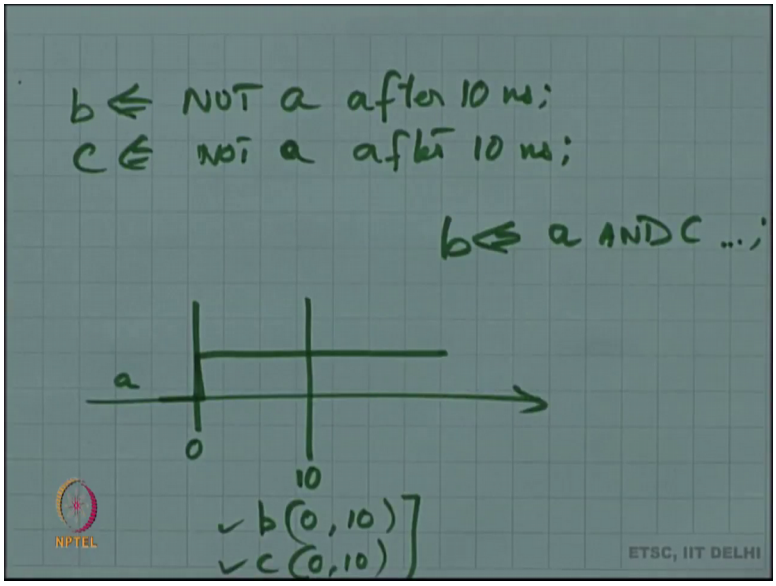
Student: In that case how will it organize in the event queue.

Let us see. You I have 2 transactions being simultaneously scheduled. How will that come about? Let us see, what kind of statements would lead to 2 transactions being scheduled at the same time.

Student: B is equal to not a after 10 nanoseconds.

okay.

(Refer Slide Time: 37:24)



Student: After 10 nanosecond and c is equal to not a after 10 nanoseconds

C equals not a a after 10 nanoseconds fine. So, this time goes from left to right. And at some time, I have a changing let us say at 0 time. The value of a let us say I made it from 0 to 1.

And as a result of that 10, nanoseconds later, I have those 2 transactions b being. So, I will say a change to 1. So, I have b being 0 10, and c being 0 10 right. So, both of these are these 2 transactions are there. And they have to be processed the order is not very important. Well, from the correctness point of view the order is you could pick up this one first for processing. A more detailed treatment of this situation, where time is not changing, but maybe multiple iterations might happen we will get to. But as of now you can just assume that we will pick up one transaction. Does not matter which one and process it. And then we will move to the next one and process it. For now, this is what would happen, yeah.

Student: Now these are 2 events (Refer Time: 38:55).

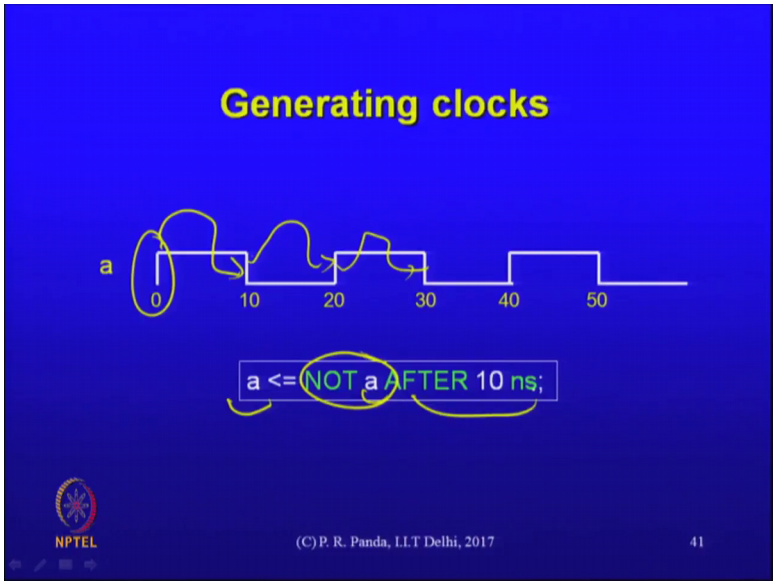
These are different there is an event on b, there is another event on c. Event is associated with a signal.

So, we say an event has occurred on signal b. And another event has occurred or has not occurred on signal c. So, 2 events are created, 2 transactions are posted. In the general case both of them might result in event. So, that is what might happen here.

Student: But in a different case suppose, the event on b effects c.

Why do not we get to that later? I think you are referring to a situation where the event on b let us say you have not a. You could have a statement that says b equals a and c or something. Like that and both of them are scheduled. It is possible that multiple transactions may need to be scheduled on to the same signal without time moving forward. That is also possible does not introduce any inconsistency, but we will look at a.

(Refer Slide Time: 39:50)



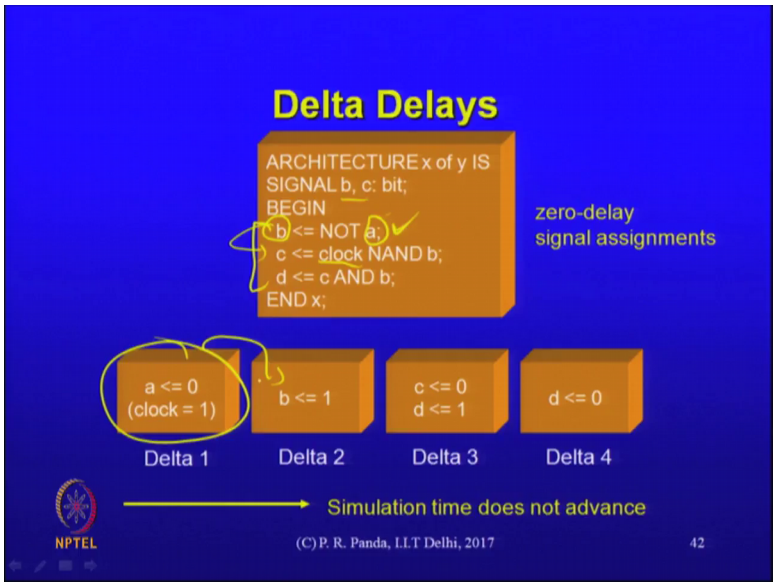
Simple application of those delays, of course, is a kind of statements where you are generating clocks. There would be a very simple way to just say a takes the value of not a after a delay.

Do you see that it creates an infinite sequence of events, right? So, as a result of that statement executed at time 0, we would schedule a transaction at time 10, in that transaction expires at time 10, and it creates an event on a. There is nothing that is stopping us from including the same signal on both the left-hand side and right-hand side it is it is consistent. So, since the value of a changed, now that statement would be

executed, again because a also occurs on the right hand side. So now, because I am inverting the value of a, a new value which is one would be scheduled as a result of this, I have a new value being scheduled. As a result of this event I have a new value being scheduled. So, this goes on forever. And that in fact, would be a standard way of creating clock kind of statement.

This is a simple clock, but you can by varying these after clauses. You can have multiple after clauses remember. You can have clocks of different, where the on and off portions need not be equal right.

(Refer Slide Time: 41:13)



That brings us to one final aspect which is called delta delays. It handles this situation that was pointed out where multiple triggering of statements might happen without time being forwarded, right. So, let us say I have a bunch of these 3 statements. I have b c d being assigned. So, these are count concurrent signal assignment statements. Let us assume that well b c are signals internal to the architecture. And I have statements like this which has 0 delay signal assignments. All are 0 delay signal assignments, but what is interesting about this sequence is that; you see as a result of a changing, I have that statement being triggered.

But that other statement c is being triggered on clock changing which is fine. Remember, if any signal changes in that RHS expression then we should be reevaluating it. So, c

may be triggered, not merely because that input clock is changing, but also because one of the internal signals b is changing. And in fact, time might not have forwarded between the 2 triggers.

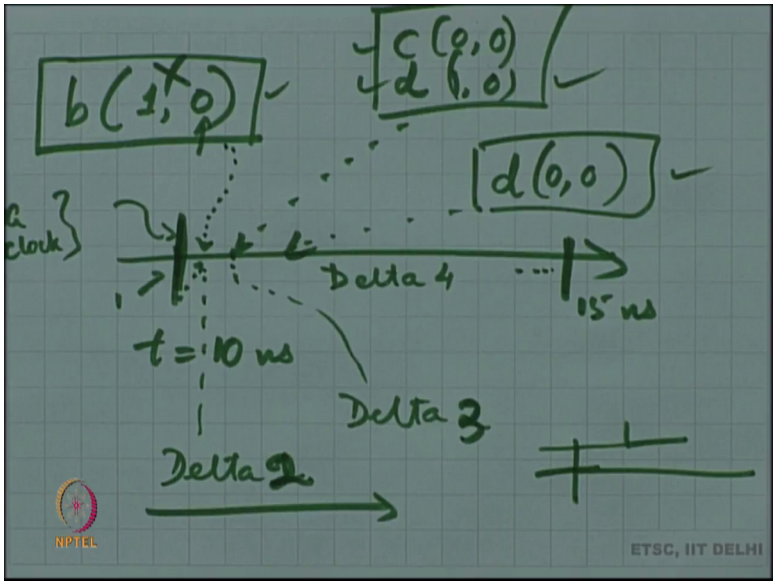
So, that is why it is in some ways an interesting case that needs to be separately looked at. So, let us say it at a particular time, I have these values on the inputs. Both are input ports a and clock are input ports. A is taking the value of 0, and let us say the value of clock was already 1 at that time. Because the value of a is changing, I have this statement being triggered. So, a value of one is being scheduled on to b.

A transaction is posted, but that transaction is posted with a 0 delay, right because I have to process that statement without time forwarding. That is the specification and I have to observe that whether that is reality or not is a different matter it is a practical, reality or not in practice nothing happens in 0 time. But still 0 delay has important modelling related advantages. It is worth writing 0 delay signal assignment statements. For understanding functionality, but also more important like we had said earlier, these the kind of statements can be input to a synthesis tool at a time when we have not decided the details of the circuit right.

Student: (Refer Time: 43:41).

No, no, no clock is an input port of this entity which is not showed here, fine. So, I have to post a transaction b equals 1.

(Refer Slide Time: 43:56)



That I have a b value of 1, but delay 0 has to be somehow inserted into my event queue for processing.

So, actually it is now different from the standard the general algorithm that we had looked earlier but of course, because this transaction has a 0 delay. This is the one that would be picked up when I conceptually forward my time, the time is actually not forwarding. But to be forward it through a mechanism called a delta delay.

These are the iterations that we go through in the simulator and the internally this event queue is being processed. But time is actually not forwarding, but we are of course, guaranteed that this transaction will be processed immediately, right because it is scheduled for immediate processing, even though all the other transactions are there in the event queue. This is the transaction that will be actually converted into any event. That is it because the time is not forwarded; we divide the time into what are called delta cycles. So, I have my time let us say is 10 nanoseconds.

At 10 nanoseconds all of these a changed and so, these inputs changed at 10 nanosecond. As a result of this what happens is that a sequence of triggering happen, but the immediate consequence of a changing is just that one statement is evaluated. All these statements are not evaluated, because they are not sensitive to a. What I meant here is that the clock is not changing. The value of clock is stable at 1 only a changed. As a

result of a changing only b can change, others cannot change. That is the algorithm of the similar. So, what we will do is we schedule that transaction, right. We post that transaction at what is called the next delta cycle of a simulator. This is a simulator, is internal concept that is the way they treat this.

So, delta means that; the time is still 10 nanoseconds, right the time has not changed. But we just move to what is called the next delta cycle. So, in the first delta cycle the values of. So, a changed at this a and clock, both of these changed. This resulted in some processing. So, it is resulted in that first statement getting evaluated and that transaction being posted for b at the first delta. So, this is the first delta.

But still this the first delta is still earlier than all the other transactions that are pending on the queue. So, we just pick up the transaction that we just entered, that is the transaction that would be picked up for processing. So, we process b 1 0, then what happens? Because of that the value of b changes, because the value of b changes, these other 2 statements assigning to c and d will actually be evaluated, right. Now it will be evaluated without time being forward. So, this is processed right.

But in the process what happens is 2 more transactions will be entered here. One is that c I have a value of 0, what is the time? Still 0 is the relative time from now for d the value is 1. But a time is (Refer Time: 47:18) both of these events would now get created and inserted into the event queue in the normal way that the event queue anyway works. Then what should happen? So, then we say that we are done with the first delta. We move to the next delta. So, that is my delta 2.

So, I move to the next delta. So, actually the way I wrote this up, you can think of maybe, this is the first delta. This is the second delta, and that is my third delta right. So, at that third delta these are taken up both of those transactions are taken up for processing. So, they result in new as event. So, new events may be created on c and d. And then what happens? As it happens, that assignment to d actually has c on the right-hand side. Now since an event was created on c that may result in a new transaction.

So, the others are b sensitive to a that is fine c is sensitive to clock and b that is fine. But because c changed, now I may need to change d. And therefore, what happens is a value

of 0, new value of 0 is posted on to d with the time being 0 still all of these are 0 delay. So, that is my new transaction; that would be posted here for processing in my delta 4. So, that may lead to a new event being created on d.

That should lead to a stable situation as far as the time t equals 10 nanoseconds is concerned. No more events to be processed, no more transactions to be created or events to be created at time 10 nanosecond. After that we move to; so, since we are done with t equals 10, we now look at the event queue to see what other events might be there, and picked up the next one maybe the next event is at 15 nanosecond per amps.

But this is the sequence of delta delays that is important from the point of view of handling the 0-delayed transaction. So, point is the same assignment statement might be triggered multiple times. You see that there were multiple transactions being posted on to d. Without time changing and it has happened in a normal way, if it happens that multiple transactions are inserted. You process all of them.

As a result of processing each of those transactions another bunch of transactions may need to be scheduled and they are scheduled into the next delta cycle. You can have any number of delta cycles, but the treatment of the simulator would be the same as what we only saw, yeah.

Student: Sir if we if we make the transaction on d transition on d.

Right.

Student: If we could transport delay, then what would happen? Because this is delta function delta d right.

If it was a transport delay instead of inertial delay, this statement the assignment to d was a transport delay. So, it is possible that as a result of these sequence of events that are treated without time being changed. You end up in a situation where the same signal may have values that go up and down without time progressing. It is possible, right. If there was one another event let us say was a scheduled on to d.

In a future delta, it is possible that you go up and you come back without time progressing. The mechanism for distinguishing between inertial and transport delays is

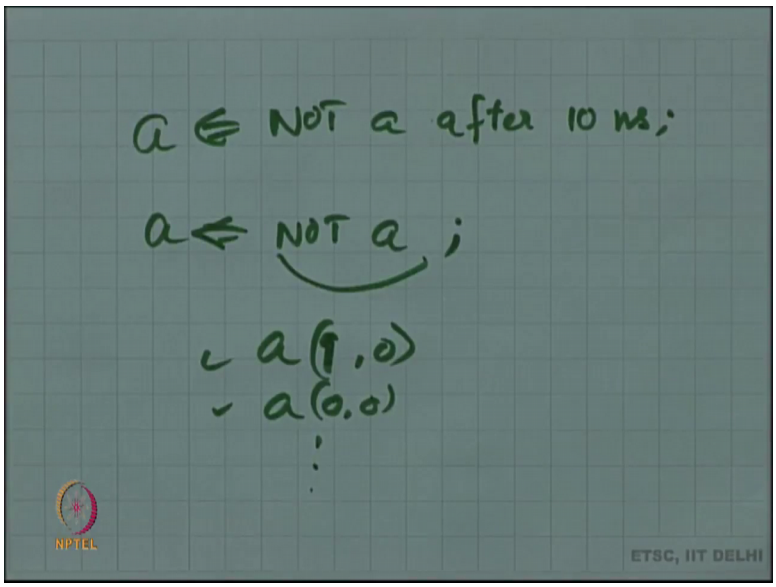
still the same. So, if they happened in sequence, then they show up like this. It may not be visible in the waveform that you see. These are internal states really this goes up and down. Simulator might give you a mechanism for expanding the view at a particular time to observe deltas.

By default, the deltas are not visible in the normal simulation. This is just there to illustrate and for us to understand what is going on internally. But the treatment is no different of than the standard transport delay, which is if it happens maybe we insert multiple transactions into the event queue if it is transport. The way this happened here it does not matter even if it was inertial delay it would still the spikes would show up, but only in the deltas which are meaningless anyway. This as a simulation concept this is not the real waveform, yeah.

Student: Sir what if there is a loop back (Refer Time: 52:04).

The bunch of statements that you write here could have a cycle in terms of dependency, right. I can have a depends on b, b depends on c, c depends on a. This is considered a correct VHDL statement there is nothing wrong in it. In fact, it could represent correct hardware also.

(Refer Slide Time: 52:37)



We said that the standard mechanism for creating a clock was a takes the value of not a after 10 nanosecond. What if you had written a takes the value of not a is this correct language? Syntactically, is it correct? VHDL.

Student: Yes.

It is correct there is nothing that is preventing us from writing like this. So, the compiler would go through. Will generate a design that at simulation time what happens? At time 0 what happens?

Student: (Refer Time: 53:06) internal state will change (Refer Time: 53:09).

Internal state will change, how many times?

Student: (Refer Time: 53:12).

Think in terms of the delta cycle, every time you have a 0 delay, if you have to think in terms the processing would be in terms of the delta cycles, what happens? At time 0 at first delta let us say we initialized everything. In the next delta we are scheduling that event a. So, let us say one is scheduled at time 0. In the next delta what happens?

Student: (Refer Time: 53:31)

So, a 0 is scheduled, how many deltas do we go through?

Student: Infinite.

An infinite number of times simulators normally will warn you that, the way you have written your statement simulation will not even proceed right. So, that is what. So, it is legally correct you can check it out if you want to, but there may be users creating statement like this is the perfect use of introducing a cycle. Since you can do it in one statement you could also do introduced that cycle through a sequence of statements and it would be fine.

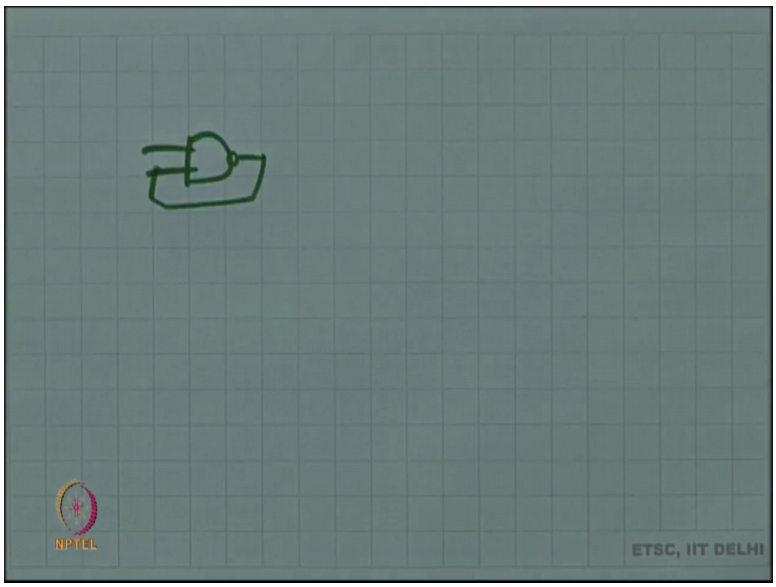
But it is fine. So, just like in sequential design, you can have output of one stage feeding back to the input of another stage. So, the cycles are they are permitted. We should

understand of course, what we have written. Hopefully the cycle should not be combinational cycles, right. That is when there may be some issue.

Student: Sir one more question. If 2 signals so, it is a multi-input gate like (Refer Time: 54:27) or some other function. And both the signals are reaching at the same time. Different values will come. And it is a race kind of condition if I use first one as the delta, then my output will be different. If I choose the other one as my first delta, then the output will be different depending on I choose a (Refer Time: 54:50).

So, there is some such feedback from output to the input.

(Refer Slide Time: 54:42)



Student: (Refer Time: 54:54) from the previous portion of the circuit, but both the signals have arrived at the same time exactly same time. So, how does simulator choose to?

At the heart of the simulators algorithm there is no intelligent resolution that is being made the algorithm is indeed what we just said and when there are a bunch of transactions expiring at a particular time, the simulator will pick them up in an order that is unspecified. The language does not insist that you pick up in this order or the other order.

Student: It is a implementation (Refer Time: 55:27).

It is left to the implementation of the simulator. And in fact, the designer has to be careful to make sure that the expected functionality does not depend on the order in which the simulator is going to pick up the events for processing.

Student: Sir these are posted for, but when are these processed that processes should happen at 0 time.

Yeah. So, all the events, all the transactions that are posted with 0 time, would be picked up for the next delta. All of them would be taken up for processing.

Student: So, it is does not matter how many deltas are there (Refer Time: 56:07).

You do not know how many deltas are going to be there. See, the processing of events at a particular delta may create 7 new events all of them are processed further are posted for the next delta. So, when it comes to next delta in some order I am going to process all of them maybe it will result in 5 new events being created.

Those 5 new events, again may lead to maybe 8 new transactions. All of them are posted for the next delta. So, it is only one delta at a time that we are processing. We do not post something for 3 deltas from now, because it is a 0 time we are only posting it for the next delta. But we do not know when we are posting it how many times I need to iterate because that is a that depends on the functionality.

Student: But graphically we will see the final values we will not see these transaction (Refer Time: 56:50).

Graphic; so, that the default waveform will only give us the final values, but I think simulators may give you an option to also look at the delta transitions. At a particular time, you can say that show me the expand the deltas and they might I think, yeah. But the standard waveform will not show it you here.