

In this lecture we discuss the two commonly used pure and inertial delay models. We further introduce the support of VHDL for these two models via its delay mechanism.

HWMoD
WS25

Delay Mech.
Delay Types
VHDL

Hardware Modeling [VU] (191.011) – WS25 – VHDL Delay Mechanisms

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

In the previous lecture we already encountered the `after` keyword for delaying the assignments of signals. However, while this is certainly handy for creating certain waveforms, this is nothing we could not also achieve via the different `wait` statements we already covered in previous lectures. So why does VHDL include this feature? Is it just syntactic sugar?

Introduction

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL

- Previously: `after` for delaying the assignments of signals

Well, as we will discuss in this lecture, the possibility to delay signals is paramount for simulating the behavior of real hardware. In a real circuit, all its individual elements, for example gates and flip-flops, do actually only change after a certain delay. This should not be too much of a surprise though, as you already encountered this fact in the guest lecture about the synchronous design style. But what are the causes of this behavior? What support do we require from a hardware description language in order to properly model the delays of a real circuit?

Introduction

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL

- Previously: `after` for delaying the assignments of signals
 - Why is this needed? ⇒ Delays exist in real hardware

From an abstract point of view we can differentiate between basic delay models which have their origin in different physical phenomena. You should already be familiar with the first one, the so-called *pure* delay, which has its origin in the finite propagation speed of a signal. The other one, referred to as *inertial* delay, is likely less familiar to you. This kind of delay is the result of parasitic capacities that must be charged and discharged whenever a signal changes. We will discuss both in a bit more detail now.

Introduction

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL

- Previously: `after` for delaying the assignments of signals
 - Why is this needed? ⇒ Delays exist in real hardware
- Two types of delay
 - *Pure* delay: finite signal propagation speed
 - *Inertial* delay: (dis)charging of capacitance



In order to illustrate and discuss pure delay, let us start by assuming that we have a simple piece of wire as shown on the slide.

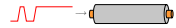
Pure Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



- VHDL Delay Mechanisms
 - Delay Types
 - Pure Delay**

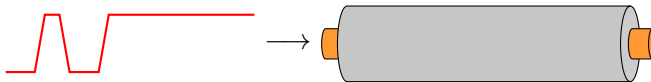


Next, we apply some input signal on the left side of this wire and assume that it follows the trace shown on the slide. If we now connect a measurement device like an oscilloscope to the right side of the wire, what will we observe?

Pure Delay

HWMMod
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL





Ideally we would of course assume that we are able to exactly observe the applied signal. However, this ignores the fact that the propagation of a signal is actually the physical process of charge carrying particles moving from one physical place to another. These particles can only move with a finite speed.

Pure Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL





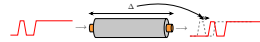
Hence, what we will observe is a delayed version of the applied signal. Still, the overall waveform observed to the right of the wire will be exactly the same as to the left, just delayed by a certain time. Due to the delayed waveform being of identical shape than the one fed into the wire, this kind of delay is called pure delay. This particular model has a single parameter, which is the amount of time by which the measured output lags behind the input.

Pure Delay

HWMMod
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



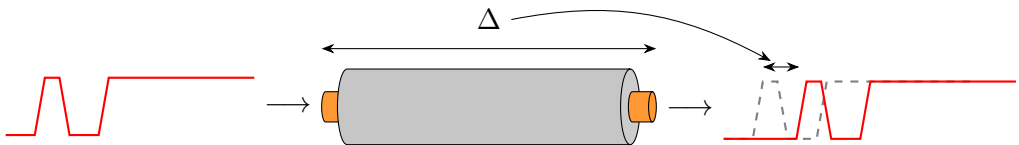


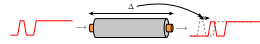
As illustrated on the slide, in the shown example this value is essentially depending on the properties of the wire such as the length.

Pure Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



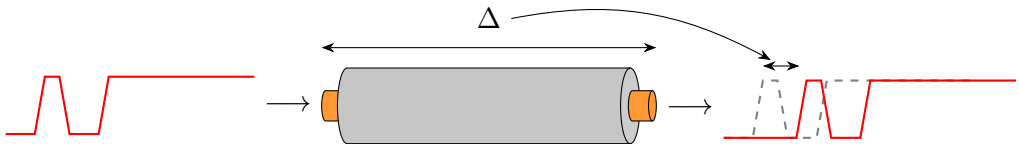


At this point you might be asking yourself if this is really relevant for the highly integrated circuits at nanometer scale we are ultimately after. However, due to the high clock frequencies permitted with modern manufacturing processes and the finite speed of light, even short wire distances in the nano to micrometer range have a significant impact on the overall timing of a design and must be accounted for. We also want to point out that the pure delay model is often not very accurate and designers must carefully evaluate if pure delay is applicable to their case. For example, for fiber optic cables this delay might be a sensible choice. Let us now come to the second kind of delay we are going to consider, inertial delay.

Pure Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL





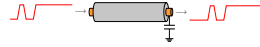
Let us start by considering the same scenario as before. That is, we again have a wire into which we feed the same signal trace. So far the output to the right of the wire will naturally be the same as in the previous case of pure delay.

Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



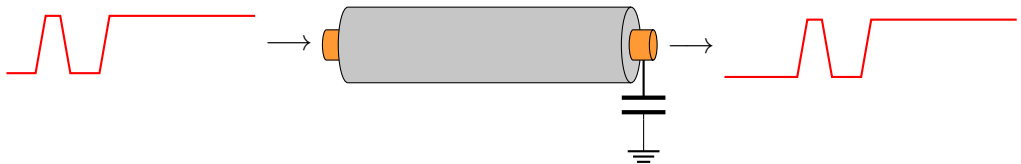


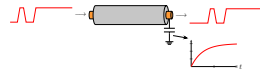
However, in order to better capture reality, we now add a capacitance to the output of the wire. This is symbolized by the capacitor symbol to the right of the wire. The wire capacitance is a parasitic effect and is always present in any electrical conductor. However, in reality the capacitance would actually be distributed across the whole length of the wire. If we think about the wire also featuring a capacitance we must reconsider which signal we expect to observe to the wire's right.

Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



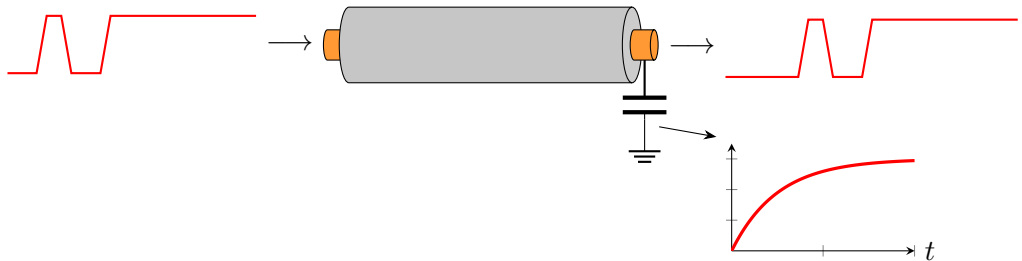


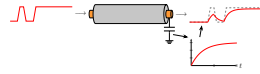
Recall from some basic courses about electronics that a capacitor is actually charged, respectively discharged, to the applied certain voltage level over time. The respective charging and discharging characteristics follow an exponential function which is depicted on the slide. This has a remarkable consequence on the signal we are able to observe.

Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



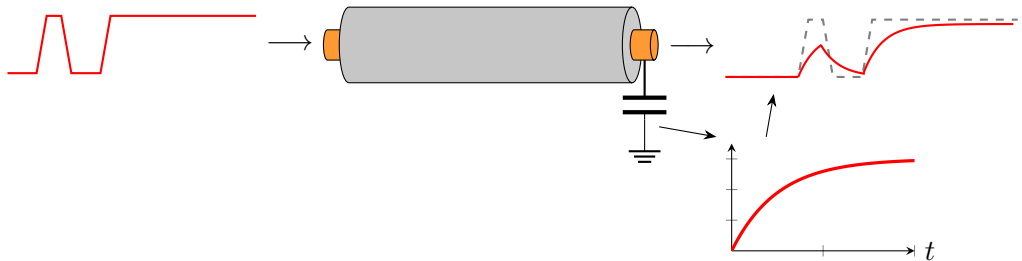


Instead of the linear slopes we assumed so far, we will instead observe the wire's capacitance being charged or discharged to the applied voltage. As we can see in the example shown on the slide, if a voltage is applied for a short time only, the capacitance might never even be completely charged to the applied input level.

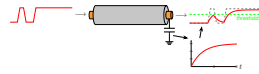
Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



- VHDL Delay Mechanisms
 - Delay Types
 - Inertial Delay**

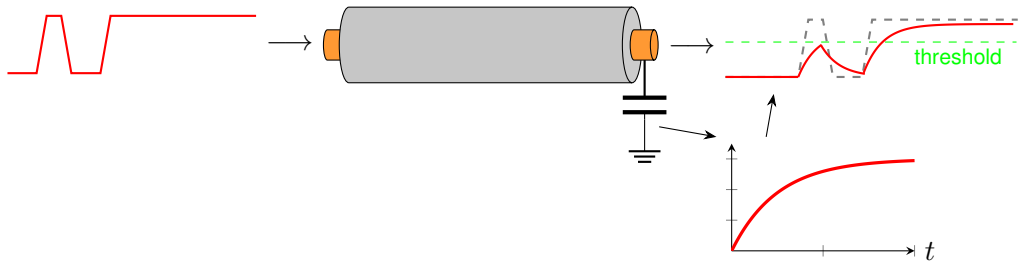


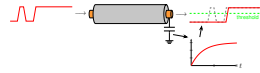
Now, recall that a digital circuit discretizes its analog input based on an internal threshold voltage as drawn on the slide. Considering this discretization voltage, what will a digital receiver observe?

Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



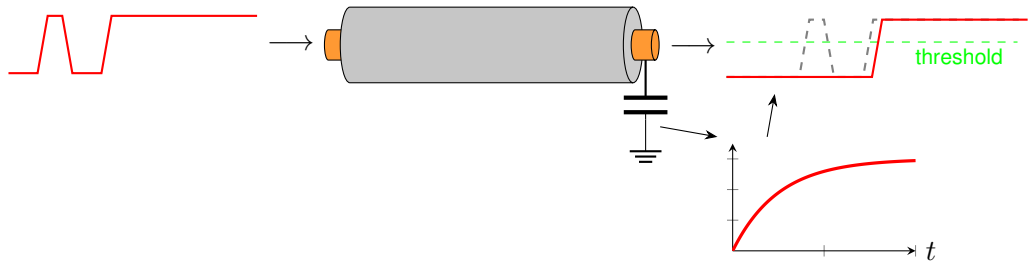


Well, essentially the digital receiver will be oblivious to all pulses that never cross its threshold. In our example this corresponds to the first two transitions being completely missed by the imaginary receiver circuit. In conclusion, inertial delay does not only encompass the effect of pure delay, but also model certain pulses to be filtered away during transmission of a signal. This is obviously in stark contrast to the behavior exhibited by pure delay alone.

Inertial Delay

HWMoD
WS25

Delay Mech.
Delay Types
Pure
Inertial
Comparison
VHDL



VHDL Delay Mechanisms

Delay Types

Pure vs. Inertial Delay

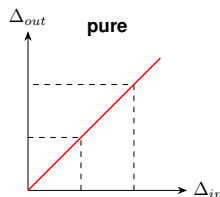
- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible



Let us now summarize what we just discussed. Assuming a pure delay means that an observed signal trace is basically an exact copy of the original trace shifted backwards in time by a constant amount. All transitions are thus propagated exactly as they are. In particular, pulses can be arbitrarily small and will still be faithfully observed by a receiver. This is illustrated by the plot on the slide where input pulse widths Δ_{in} are mapped to the corresponding pulse widths observed by a receiver, referred to as Δ_{out} . Since all pulses propagate we can observe an identity function. What makes this delay model charming is its simplicity, both conceptual and from the point of view of implementing it. However, it is only applicable in cases where the effects of capacitances can be neglected.

Pure vs. Inertial Delay

- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible

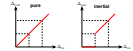


VHDL Delay Mechanisms

Delay Types

Pure vs. Inertial Delay

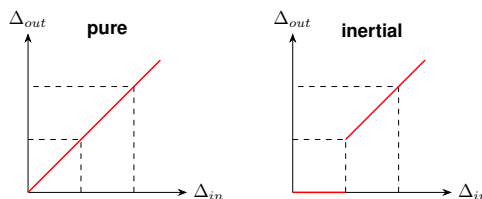
- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible
- Inertial delay
 - Output wave is input wave "shifted backward" in time plus pulse-width filter
 - Only pulses above a certain width propagate
 - Use where capacitance relevant



Inertial delay is to some extent a more specialized form of the pure delay. Pulses above a certain width, depending on the particular circuit, are propagated with a pure delay, but pulses below this width are not observed by a receiver. We can thus view inertial delay as pure delay with a preceding filter that removes pulses below a certain width. The plot on the bottom of the slide illustrates this. Input pulses below a certain width result in a pulse of width 0, meaning no pulse at all, at the receiving side. When compared to pure delay, inertial delay is conceptually somewhat more complex to understand than pure delay. Furthermore, it is a bit harder to implement. However, in case there is significant capacitance involved in the signal propagation, this delay model is much more accurate.

Pure vs. Inertial Delay

- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible
- Inertial delay
 - Output wave is input wave "shifted backward" in time plus pulse-width filter
 - Only pulses above a certain width propagate
 - Use where capacitance relevant

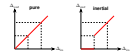


VHDL Delay Mechanisms

Delay Types

Pure vs. Inertial Delay

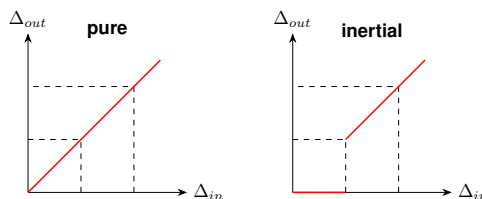
- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible
- Inertial delay
 - Output wave is input wave "shifted backward" in time plus pulse-width filter
 - Only pulses above a certain width propagate
 - Use where capacitance relevant
- Further, more specialized, models exist



Finally, we want to point out that there also exist further, more specialized, delay models. However, for our purpose the two presented ones suffice.

Pure vs. Inertial Delay

- Pure delay
 - Output wave is input wave "shifted backward" in time
 - Arbitrary small pulses are propagated
 - Use where capacitance negligible
- Inertial delay
 - Output wave is input wave "shifted backward" in time plus pulse-width filter
 - Only pulses above a certain width propagate
 - Use where capacitance relevant
- Further, more specialized, models exist



In the lecture about signal assignments we already mentioned that VHDL provides a dedicated delay mechanism as part of its signal assignments. We will now continue by discussing this language feature and how it is used.

Delay Mechanisms in VHDL

163

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- VHDL contains support for delays

```
target <= [ delay_mechanism ] waveform;
```

VHDL Delay Mechanisms

VHDL

Delay Mechanisms in VHDL

- VHDL contains support for delays
- target <= [delay_mechanism] waveform;
- Both presented delay models are supported

```
delay_mechanism ::=  
  transport  
  | [ reject time_expression ] inertial
```

At the bottom of the slide you can see the syntax of this delay mechanism. We can immediately note that it consists of two alternatives. These two alternatives can be used to describe pure, respectively inertial, delays.

Delay Mechanisms in VHDL

163

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- VHDL contains support for delays
- target <= [delay_mechanism] waveform;
- Both presented delay models are supported

```
delay_mechanism ::=  
  transport  
  | [ reject time_expression ] inertial
```

The keyword `transport` is used to describe pure delay, which we will consider in more detail on the next slide.

Delay Mechanisms in VHDL

163

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- VHDL contains support for delays
- target <= [delay_mechanism] waveform;
- Both presented delay models are supported
 - Supports pure delay via `transport`

```
delay_mechanism ::=  
  transport  
  | [ reject time_expression ] inertial
```

```
■ VHDL contains support for delays
target <= [ delay_mechanism ] waveform;
■ Both presented delay models are supported
  ■ Supports pure delay via transport
  ■ Supports inertial delay via reject and inertial

delay_mechanism ::=
  transport
  | [ reject time_expression ] inertial
```

The other alternative of the `delay_mechanism` is for describing inertial delay. It consists of two keywords, `reject` and `inertial`, which we will also consider in more detail in this lecture. However, let us now consider VHDL support for pure delay.

Delay Mechanisms in VHDL

163

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- VHDL contains support for delays
`target <= [delay_mechanism] waveform;`
- Both presented delay models are supported
 - Supports pure delay via `transport`
 - Supports inertial delay via `reject` and `inertial`

```
delay_mechanism ::=
  transport
  | [ reject time_expression ] inertial
```

As mentioned, if you want to model pure delay, VHDL provides the `transport` keyword which can be used directly in front of a waveform declaration as shown on the slide.

Pure Delay in VHDL

164

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- Keyword `transport` before the waveform in an assignment
`target <= transport waveform`

By introducing this keyword the succeeding wave form is delayed by some constant amount of time - just as we discussed earlier.

Pure Delay in VHDL

164

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- Keyword `transport` before the waveform in an assignment
target <= `transport` waveform
- Signal assignment is simply delayed by some time

The delay time of this pure delay is given by the time expression of the `after` clause of the first element of the waveform. However, since an `after` clause is optional, there exists a default time. This default delay time is 0 nanoseconds. For the sake of completeness, let us now briefly look at an example.

Pure Delay in VHDL

164

- Keyword `transport` before the waveform in an assignment
target <= transport waveform
- Signal assignment is simply delayed by some time
 - Time given by waveform's `after` clause
 - Per default 0 ns



Consider the waveform of a signal `i`, as shown on the slide, assuming the signal is initialized to zero. For illustration, you can find a trace of this waveform right next to it.

Pure Delay in VHDL

164

- Keyword `transport` before the waveform in an assignment
target <= `transport` waveform
- Signal assignment is simply delayed by some time
 - Time given by waveform's `after` clause
 - Per default 0 ns
- Example

```
i <= '1' after 2 ns, '0' after 3 ns, '1' after 4 ns;
```





Next, consider the signal assignment shown on the slide. It assigns to signal `o` a waveform corresponding to a version of the one of `i` which is delayed by five nanoseconds. Observe how we require the `transport` keyword to explicitly express that we want to use a pure delay. Note that we again assume that `o` is initialized to zero.

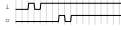
Pure Delay in VHDL

- Keyword `transport` before the waveform in an assignment
`target <= transport waveform`
- Signal assignment is simply delayed by some time
 - Time given by waveform's `after` clause
 - Per default 0 ns
- Example

```
i <= '1' after 2 ns, '0' after 3 ns, i
    '1' after 4 ns;
[...]
```



```
o <= transport i after 5 ns;
```

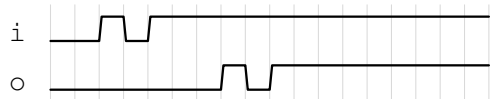


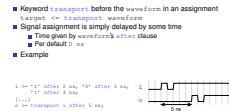
As a result, we can observe the trace shown on the slide.

Pure Delay in VHDL

- Keyword `transport` before the waveform in an assignment
`target <= transport waveform`
- Signal assignment is simply delayed by some time
 - Time given by waveform's `after` clause
 - Per default 0 ns
- Example

```
i <= '1' after 2 ns, '0' after 3 ns,
      '1' after 4 ns;
[... ]
o <= transport i after 5 ns;
```





Note how the trace is basically the same one as i. So far so good. Let us now get to VHDL's support for inertial delay.

Pure Delay in VHDL

164

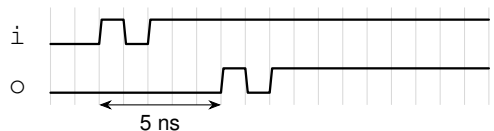
HWMoD
 WS25

Delay Mech.
 Delay Types
 VHDL
 Pure
 Inertial

- Keyword `transport` before the `waveform` in an assignment
`target <= transport waveform`
- Signal assignment is simply delayed by some time
 - Time given by `waveform's after` clause
 - Per default 0 ns
- Example

```
i <= '1' after 2 ns, '0' after 3 ns,
      '1' after 4 ns;
[...]
```

```
o <= transport i after 5 ns;
```



If you want to use the inertial delay model in VHDL, you can do this via the mentioned `inertial` and `reject` keywords. However, depending on the particular delay you are after this might not even be necessary as inertial delay is the default one in VHDL. Essentially, the `inertial` keyword is only strictly required if you want to make use of the optional `reject` clause which we will discuss shortly.

Inertial Delay in VHDL

164

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`

In essence, the VHDL support for inertial delay behaves like we discussed before. In particular, pulses narrower than a certain width are rejected and non-rejected pulses are delayed just as with a pure delay. The threshold width, referred to as *pulse rejection limit*, can be defined using the `time_expression` in the `reject` clause. If none is present, the limit is associated with the time of the first waveform element's `after` clause.

Inertial Delay in VHDL

164

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of `waveform`

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

Let us now consider the same example as before using an inertial rather than a pure delay.

Inertial Delay in VHDL

164

HWMoD
 WS25

Delay Mech.
 Delay Types
 VHDL
 Pure
 Inertial

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of waveform
- Example

```
i <= '1' after 2 ns, '0' after 3 ns,
      '1' after 4 ns;
```





In particular, we explicitly specify a pulse rejection limit of one nanosecond in the assignment to `o`.

Inertial Delay in VHDL

164

HWMoD
 WS25

Delay Mech.
 Delay Types
 VHDL
 Pure
 Inertial

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of waveform
- Example

```
i <= '1' after 2 ns, '0' after 3 ns, i
      '1' after 4 ns;
[...]
```



```
o <= reject 1 ns inertial i after 5 ns
```

- Default mechanism, explicit via `inertial` in an assignment
- Signal assignment is delayed and too short pulses are filtered out
- Minimum pulse width optionally defined by reject clause
- Per default time expression associated with first element of waveform
- Example

```

target <= [reject time_expression] inertial waveform
  
```

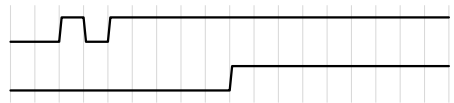
On the slide you can see the resulting trace. Note how it is clearly not the same as the one of the signal `i`.

Inertial Delay in VHDL

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of waveform
- Example

```

i <= '1' after 2 ns, '0' after 3 ns, i
      '1' after 4 ns;
[...]
o <= reject 1 ns inertial i after 5 ns;
  
```

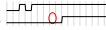


- Default mechanism, explicit via `inertial` in an assignment
- Signal assignment is delayed and too short pulses are filtered out
- Minimum pulse width optionally defined by reject clause
- Per default time expression associated with first element of waveform
- Example

```

1 i <= '1' after 2 ns, '0' after 3 ns, '1' after 4 ns;
2 i <= reject 1 ns inertial i after 5 ns;

```



We can note that the first, narrow, pulse is missing. The reason is of course that this pulse has a width of one nanosecond and is hence rejected due to the one nanosecond pulse rejection limit.

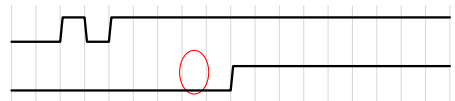
Inertial Delay in VHDL

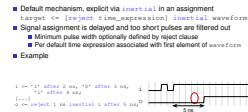
- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of waveform
- Example

```

i <= '1' after 2 ns, '0' after 3 ns, '1' after 4 ns;
[... ]
o <= reject 1 ns inertial i after 5 ns;

```





In addition to that, we can again observe a pure delay by the time specified in the `after` clause of the first waveform element. Finally, to end this lecture, we want to remark a few things about the delay mechanisms in VHDL.

Inertial Delay in VHDL

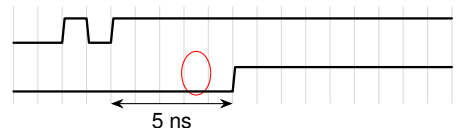
164

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

- Default mechanism, explicit via `inertial` in an assignment
`target <= [reject time_expression] inertial waveform`
- Signal assignment is delayed and too short pulses are filtered out
 - Minimum pulse width optionally defined by reject clause
 - Per default time expression associated with first element of waveform
- Example

```
i <= '1' after 2 ns, '0' after 3 ns, i
      '1' after 4 ns;
[... ]
o <= reject 1 ns inertial i after 5 ns; o
```



First, we really want to stress that per default VHDL assumes an inertial delay, even if you do not use the `inertial` or `reject` keyword. If you are not aware of this, it can have unexpected side effects, especially considering that both the pure and the rejection limit are per default associated to the first `after` clause.

Remarks

- Default is inertial delay!

Another thing we want to mention is that the rejection limit of the inertial delay must be positive and smaller or equal than the time specified by the first `after` clause of the respective waveform.

Remarks

- Default is inertial delay!
- Pulse rejection time positive and smaller or equal the `after` time

HWMoD
WS25

Delay Mech.
Delay Types
VHDL
Pure
Inertial

To emphasize on the fact that inertial delay is the default one and that the reject limit is optional, consider the three delayed signal assignments on the slide, which are actually equivalent to each other. In particular, we want to emphasize that the first statement does not correspond to a pure delay.

Remarks

- Default is inertial delay!
- Pulse rejection time positive and smaller or equal the `after` time
- All equivalent
 - o `<= i after 1 ns;`
 - o `<= inertial i after 1 ns;`
 - o `<= reject 1 ns inertial i after 1 ns;`

Also note that any pulse shorter than the limit is rejected in an inertial delay setting. In particular, we want to point out that this also includes low and not just high pulses.

Remarks

- Default is inertial delay!
- Pulse rejection time positive and smaller or equal the `after` time
- All equivalent
 - `<= i after 1 ns;`
 - `<= inertial i after 1 ns;`
 - `<= reject 1 ns inertial i after 1 ns;`
- Any pulse shorter than the limit is rejected

Finally, note that each signal assignment comprising a pure delay can be converted in an equivalent inertial delay assignment. This is achieved by explicitly setting the pulse rejection limit to zero.

Remarks

- Default is inertial delay!
- Pulse rejection time positive and smaller or equal the `after` time
- All equivalent
 - `<= i after 1 ns;`
 - `<= inertial i after 1 ns;`
 - `<= reject 1 ns inertial i after 1 ns;`
- Any pulse shorter than the limit is rejected
- Pure delay: can convert to equivalent `inertial` delay
 - `<= transport i after 10 ns;`
 - `<= reject 0 ns inertial i after 10 ns;`

Thank you for listening! We recommend you to immediately take the self-check test in TUWEL, to see if you understood the material presented in this lecture.

Lecture Complete!