-Signal Assignments

Hardware Modeling [VU] (191.011) — WS24 — Signal Assignments Florian Huemer & Sebastian Wedemann & Dylan Baumann WS 2024/25

In this lecture we take a deeper dive into VHDL signal assignments and explore language features crucial for performing simulations requiring more complex signal timings. Understanding their semantics will also give us more insight into the event-based VHDL simulation in general.

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Hardware Modeling [VU] (191.011) - WS24 -

Signal Assignments

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2024/25

Modified: 2025-03-12, 16:33 (21636bb)

-Signal Assignments

Recap

We Work of the statements in processes

Concurrent signal assignments

Concurrent signal assignments

Conty covered very basic value assignments

Signal assignments were one of the first topics we covered in this course. We learned that there are two types of them, depending on whether they are used in a process or as a stand-alone concurrent assignment. We also intensively covered how signal assignments are fundamentally different from variable assignments. However, so far we have only seen and used very basic assignments. In most cases some target signal was assigned a constant value or some other signal. Beyond that we have only used conditional assignments, as a compact way to express if-ELSE-if statements.



−Signal Assignments └─Introduction └─Introduction

Plonge
 The type4
 The type4
 Adgreement and assignments
 Concurrent speed assignment speed
 Concurrent speed assignments
 Concurrent speed assignments
 Concurrent speed assignment
 Conconcent speed assignment
 Concurrent speed assignment
 Concur

In this lecture, we will take a more systematic and comprehensive look at the signal assignment syntax supported by VHDL. We will learn that for simulations signal assignments can be equipped with delay information and that it is even possible to describe complex waveform patterns with them. Exploring the semantics of these language features and the way they are handled during simulation, will also give us insight into the event-based VHDL simulation.



─Signal Assignments └─Assign. Statements └─Signal Assignment Statements (for Processes)

Signal assignments in processes Concurrent signals assignments have a similar syntax Covered separately at the end of the lecture

Let's start with discussing signal assignment statements in processes. To the extent to which we cover VHDL in this course you can consider concurrent signal assignments to use the same syntax. While there are some differences, they go beyond the scope of this introductory course. We will come back to concurrent assignments at the end of the lecture.

Signal Assignment Statements (for Processes)

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples

Signal assignments in processes

- Concurrent signals assignments have a similar syntax
- Covered separately at the end of the lecture

—Signal Assignments └─Assign. Statements └─Signal Assignment Statements (for Processes)

Signal assignments in processes ⊂ Concrete trading assignments have a similar systax ■ Conversite products assignment that are set of the lecture Signal assignment_statement ::= [label:] simple_signal_assignment [label:] setortia_inal_assignment

Consulting the VHDL reference reveals that there are three general classes of assignment statements, which are referred to as simple, conditional and selected assignments.

Signal Assignment Statements (for Processes)

HWMod WS24

Signal assignments in processes

- Concurrent signals assignments have a similar syntax
- Covered separately at the end of the lecture

Signal assignment statement syntax

```
signal_assignment_statement ::=
```

- [label :] simple_signal_assignment
- [[label :] conditional_signal_assignment
- [label :] selected_signal_assignment

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign.

└─Signal Assignments └─Assign. Statements └─**Signal Assignment Statements (for Processes)**

Bignal assignments in processes
 convert signals assignments have a similar systax
 Coverd subjectivy at the and of the lacture
 Bignal assignment statement systax • •
 assignment
 assignment
 assignment
 continue assignment

Note that, as all statements in VHDL, assignments can be equipped with an optional label.

Signal Assignment Statements (for Processes)

HWMod WS24

ssign. Statements

Signal assignments in processes

- Concurrent signals assignments have a similar syntax
- Covered separately at the end of the lecture

Signal assignment statement syntax

```
signal_assignment_statement ::=
```

- [label :] simple_signal_assignment
- [label :] conditional_signal_assignment
- [label :] selected_signal_assignment
- Optional label

—Signal Assignments —Assign. Statements —Signal Assignment Statements (for Processes)

Span backgommets in processes
 Conversion supports that a substrayed by the s

These example assignments show how this can look like in practice. You should be familiar with the semantics of these statements by now.

Signal Assignment Statements (for Processes)

HWMod WS24

ssign. Statements

Signal assignments in processes

- Concurrent signals assignments have a similar syntax
- Covered separately at the end of the lecture

Signal assignment statement syntax

```
signal_assignment_statement ::=
```

- [label :] simple_signal_assignment
- [label :] conditional_signal_assignment
- [label :] selected_signal_assignment
- Optional label

Examples

```
simple : x <= a xor b;
conditional : y <= a when c else b when d else c;
z <= ('1', others=>'0'); --no label
```

-Signal Assignments Assign. Statements Signal Assignment Statements - Multiplexer Example

Before we dig deeper into the syntax specification, let's have a quick look at the three assignment classes in action. Here we see a simple two-to-one multiplexer, with the data inputs A and B the control input C and the output O.

Signal Assignment Statements - Multiplexer Example

HWMod **WS24**

Assign, Statements



Conditional

Selected

```
1 process(all)
2 begin
   o \leq (a and not c) or 3 o \leq a when c = '0'
3
4
        (b and c);
5 end process;
```

Simple

1 process(all) 2 begin 4 else b; 5 end process; 5

```
1 process(all)
2 begin
3 with c select
     o \leq a when '0',
4
          b when others;
6 end process;
```





Using a simple assignment, the functionality of the multiplexer is directly described by a Boolean expression.

Signal Assignment Statements - Multiplexer Example

HWMod **WS24**

Assign, Statements

1 process(all)

5 end process;

2 begin

3

4



5 end process;

3 with c select 4 o <= a when '0',</pre> b when others; 5 6 end process;

-Signal Assignments Assign. Statements Signal Assignment Statements - Multiplexer Example

The conditional assignment allows to express the same behavior in a slightly more readable way, as we can simply list the different cases.

Signal Assignment Statements - Multiplexer Example

HWMod **WS24**

Assign, Statements

3

4



-Signal Assignments Assign. Statements Signal Assignment Statements - Multiplexer Example

The last possibility is a selected assignment. Where you can think of a conditional assignment as an if-ELSE-if construct, embedded into an assignment statement, the selected variant embeds a case statement. For the sake of brevity we don't go into further details on this assignment class. However, we wanted to include it such that you have seen it once and that you know it exists.

Signal Assignment Statements - Multiplexer Example

HWMod **WS24**

Assign, Statements



Conditional

Selected

1 process(all) 2 begin 3 4 (b and c); 5 end process;

Simple

```
1 process(all)
                      2 begin
o \le (a and not c) or 3 o \le a when c = '0'
                      4
                               else b:
                     5 end process;
```

1	process (a	11	L)	
2	begin			
3	with c	se	elect	
4	o <=	а	when	′°′,
5		b	when	others;
6	end proce	ess	5;	

─Signal Assignments └─Assign. Statements └─Simple Signal Assignments

Simple signal assignment syntax simple_signal_assignment ::aimple_averform_assignment | aimple_force_assignment simple_waveform_assignment ::target <- (folay_mechaniam | waveform ;

Now, let us turn to the simple signal assignment and see what the VHDL reference manual has to say about it. According to the manual such an assignment can either be a simple waveform assignment or a force or release assignment.

Simple Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign.

Simple signal assignment syntax

```
simple_signal_assignment ::=
```

- simple_waveform_assignment
- simple_force_assignment
- | simple_release_assignment

```
simple_waveform_assignment ::=
```

```
target <= [ delay_mechanism ] waveform ;</pre>
```

─Signal Assignments └─Assign. Statements └─Simple Signal Assignments

Simple signal assignment syste: simple_system.sasignment ::* simple_waveform_sasignment i simple_recesses assignment imple_waveform_sasignment target << [disp.genters.sasignment f Brocelesses segments @ Cha busch to works signal dvms N Konveel in the come

Force assignments can be used to forcefully drive a signal to a certain value, overriding other drivers on that signal. The release assignment is used to disable a previous force assignment. However, these concepts go beyond the scope of this lecture and we, thus, don't go into further detail.

Simple Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign. Simple signal assignment syntax
 simple_signal_assignment ::=
 simple_waveform_assignment
 i simple_force_assignment
 i simple_release_assignment
 simple_waveform_assignment ::=
 target <= [delay_mechanism] waveform ;</pre>
 Force/release assignments
 Can be used to override signal drivers
 Not covered in this course

−Signal Assignments └─Assign. Statements └─**Simple Signal Assignments**

Simple signed signment pythat
ill contained in a signment
ill contained in a signment
i simple_reviews_assignment
i simple_reviews_assignment
inconteduce assignment
reviews of dolary_mechanim | waveform ,
Forchelease assignments
waveform signment
for the s

What we are interested in, is the simple waveform assignment.

Simple Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign. Simple signal assignment syntax
 simple_signal_assignment ::=
 simple_waveform_assignment
 | simple_force_assignment
 | simple_release_assignment
 simple_waveform_assignment ::=
 target <= [delay_mechanism] waveform ;
 Force/release assignments
 Can be used to override signal drivers
 Not covered in this course
 Simple waveform assignments



As we already know, a signal assignment needs a target, which must obviously be a signal object.

Simple Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign. Simple signal assignment syntax
 simple_signal_assignment ::=
 simple_waveform_assignment
 | simple_force_assignment
 | simple_release_assignment
 simple_waveform_assignment ::=
 target <= [delay_mechanism] waveform ;
 Force/release assignments
 Can be used to override signal drivers
 Not covered in this course
 Simple waveform assignments
 Target signal

−Signal Assignments └─Assign. Statements └─Simple Signal Assignments

After the assignment arrow symbol follows the optional specification of the delay mechanism. Here, the delay model that should be applied to the assignment can be defined. However, as this is a separate topic on its own, there exists a dedicated lecture on it. For this lecture, we simply don't use this language feature.

Simple Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign.

- Simple signal assignment syntax
 simple_signal_assignment ::=
 simple_waveform_assignment
 | simple_force_assignment
 | simple_release_assignment
 simple_waveform_assignment ::=
 target <= [delay_mechanism] waveform ;
 Force/release assignments
 Can be used to override signal drivers
 Not covered in this course

 Simple waveform assignments

 Target signal
 Ortional drivers signal
 - Optional delay mechanism (covered in separate lecture)

─Signal Assignments └─Assign. Statements └─Simple Signal Assignments

Simple Apple adjoints of prices
 imple...setter internet...setter internet..

Then, the actual waveform, that should be assigned to the target signal, is specified. From your experience with VHDL so far, you can probably infer that in the most basic case, a waveform can just be a VHDL expression. However, as we will see shortly, in the general case waveforms are a list of value expressions and associated delay values specifying the exact times at which an assignment shall take effect. Thus, waveforms allow it to specify multiple subsequent signal transitions.

Simple Signal Assignments HWMod Simple signal assignment syntax **WS24** simple_signal_assignment ::= simple_waveform_assignment simple_force_assignment Simple Assign simple_release_assignment simple waveform assignment ::= target <= [delay mechanism] waveform ;</pre> Force/release assignments Can be used to override signal drivers Not covered in this course Simple waveform assignments Target signal Optional delay mechanism (covered in separate lecture) Waveform (a series of expressions with timing information)

−Signal Assignments └─Assign. Statements └─**Conditional Assignments**

m Conditional signal assignment syntax conditional_signal_assignment ::= target <= [delay_mechanism] cond_waveforms ; cond_waveforms ::= waveform when condition { else waveform when condition} { else waveform }

Before we look at the exact syntax specification for waveforms, let us have a quick look at the conditional signal assignment statement.

Conditional Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Simple Assign. Cond. Assign. Waveforms Examples Concurrent Assign.

```
Conditional signal assignment syntax
conditional_signal_assignment ::=
   target <= [ delay_mechanism ] cond_waveforms ;
cond_waveforms ::=
   waveform when condition
   { else waveform when condition}
   [ else waveform ]
```



 Conditional signal assignment syntax conditional.josal.panetasimpent 1:= Earpet <= [delay_mechanism] cond_waveforms ; cond_waveforms := waveform when condition [abservedors]; exceptions
 Earpet Optional doky mechanism

As with the simple signal assignment we need a target and can specify an optional delay mechanism.



Conditional signal subspinned repeats
 conditional, signal, subspinned to the subspin sector of the subspin

You can see that in the conditional waveforms block, each of the values that are assigned in a particular case are waveforms.

FWMod WS24 Fighal Assign Conditional signal assignment syntax conditional_signal_assignment ::= target <= [delay_mechanism] cond_waveforms ; cond_waveforms ::= waveform when condition { else waveform when condition { else waveform] Target Optional delay mechanism </pre>

Each of the expressions that are assignment are waveforms

Waveform syntax waveform ::= waveform_element { , waveform_element } lunaffected

At this point we can finally discuss the waveform itself, which is the actual core topic of this lecture. As you can see from the syntax specification, there are two basic cases of what a waveform can be.

Waveforms HWMod WS24 Signal Assign Maroduction Assign: Statements Examples Concurrent Assign: Image: Waveform syntax waveform ::= waveform_element { , waveform_element } | unaffected

Waveform syntax waveform ::= waveform_element { , waveform_element } | unaffected Comma-separated list of waveform elements

The first case is a comma-separated list of one or more waveform elements, which will be discussed in detail on the next slide.



wwwform syntax waveform ::= waveform_element { , waveform_element } | imaffected @ Comma-separated ist of waveform elements # pocial case: unaffected

The other case is the keyword "unaffected", which is basically used to indicate a no-operation.



 Wavdorm syntax waveform::: waveform_lement { , waveform_element } unaffected Comma-apparated ist of waveform elements Special case: unaffected; w mult; A << maffected; w mult;

Assigning unaffected to a signal is equivalent to executing a null statement. Hence, as you can see, such direct assignment to unaffected are largely pointless.



 Woodsom sydta: exveform_slement (, vaveform_slement) Insafeted standom dements

 Special and an analysis of the standom dements
 exverse standom dements
 extended and analysis
 extended analysis
 exten

However, when employed in a conditional assignment, they can be used to disable one branch of the when-ELSE-when construct.



Waveform element syntax waveform_slement ::= value_expression [sfter time_expression] | null [sfter time_expression]

For the waveform element itself we again have two possibilities.

Waveforms (cont'd)

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Examples Concurrent Assign.

Waveform element syntax

```
waveform_element ::=
    value_expression [ after time_expression ]
    | null [ after time_expression ]
```

Waveform element syndax waveform_element ::= yalue_wogreasion [after time_expression] | null [after time_expression] ! Value expression evaluation to the type of the assignment target

The first, and arguably more important one, is given by a value expression followed by an optional after clause containing a time expression. The value expression can be any VHDL expression that evaluates to the type of the target signal of the assignment.

Waveforms (cont'd)

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Examples Concurrent Assign.

- Waveform element syntax waveform_element ::=
 - value_expression [after time_expression]
 | null [after time_expression]
- Value expression evaluating to the type of the assignment target

Waveform element system
 vareform_interment ::=
 value_expression [after time_expression]
 inil [after time_expression]
 indig after time_expression]
 value appression evaluating to the type of the assignment target
 Optional mme expression
 water to evaluate to a negative time value
 Mant of evaluate in succeeding window
 Mant of evaluate in succeeding window

The optional time expression introduced with the after keyword, specifies when the assignment of the value produced by the associated value expression should take effect. At this point you might already be able to see how multiple waveform elements can be used to describe complex waveforms. In any case, we will look at several examples over the remainder of this lecture to show you how this works. In the case of a value expressions without an after clause the VHDL standard defines an implicit after with a time of 0 nanoseconds. Another important point is that the time expression must not evaluate to a negative value, since it is not possible to specify events that happened in the past. Moreover, it must be ensured that the time expression values of consecutive waveform elements increase.

HWMod WS24 Sprat Assign Accelle accellent Accelle accellent Accelle accellent Concerned Accellent Concerne Concerne Concerned Accellent Concerned Concerned Conce

When denot sprats
 verificing_limits_if
 verificing_limits_i

Finally, a waveform element can also be null. This special case is used to completely turn off the driver to a signal. However, since this feature goes beyond the scope of this course, we will not cover it in any more detail. The only thing you should note is that an assignment to unaffected is not the same as a null assignment.

Waveforms (cont'd) HWMod **WS24** Waveform element syntax waveform element ::= value expression [after time expression] Waveforms null [after time expression] Value expression evaluating to the type of the assignment target Optional time expression Implicit default: after 0 ns Must not evaluate to a negative time value Must not decrease in succeeding waveform elements null waveform elements Used to turn-off drivers to a signal Not needed or covered in this course

This slides shows a few examples of how waveforms can look like in simple as well as conditional assignments. You may want to convince yourself that these code snippets indeed adhere to the presented grammar rules.

Hyped Waveforms (cont'd) Hyped Waveform assignment examples s <= t; s <= '0' after 1 ns; s <= not s after 1 ns; s <= not s after 1 ns; s <= '1' after 1 ns, '0' after 2 ns, '1' after 3 ns; Conditional signals assignment examples s <= '1' after 1 ns when c = '1' '0' after 2 ns when c = '0' else unaffected; s <= 42 after 1 ns, 1 after 42 ns when c = '1' else t;

standhiteoture arch of sim01 is
stgmal s = st_uologio == "0";
bogis
p0 = process
bogis
s bogis
s bogis
s = s == "1";
s = ond process;

Now that we have sufficiently covered the syntax of signal assignments and waveforms, let us turn to their semantics. We will do that by the means of concrete examples, for which we will show how they are evaluated by the simulator on a step-by-step basis. By the end of the lecture you should have acquired a "feeling" for the semantics of signal assignments, which you can deepen and extend with the quizzes in TUWEL. Before we go to more complex examples, let us first work through a basic assignment without any after clauses in order to establish some notation and fundamental principles. From your knowledge about VHDL it should be completely clear what happens in this architecture. The signal s is set to high at simulation time zero and stays at this value. However, how does the simulation actually arrive at this conclusion and what internal data structures are needed for that?

	Example 1
HWMod WS24	
Signal Assign. Introduction Assign. Statements Waveforms Examples Example 1 Example 3 Example 3 Example 4 Example 5 Concurrent Assign.	<pre>1 architecture arch of sim01 is 2 signal s : std_ulogic := '0'; 3 begin 4 p0 : process 5 begin 6 s <= '1'; 7 wait; 8 end process; 9 end architecture;</pre>

s acchisecture acch of sind) is a signal a sci_siopic i - '0'y begin sci_siopic i - '0'y begin sci_siopic acchieve begin sci_siopic acchieve sci_s

Internally the simulator not only stores the current value of every signal in a design but also keeps a list of events scheduled for the future. These can be seen in the table on the right side of the slide. Whenever a signal assignment statement is executed, only this list is changed and **not** the actual value. Variables, on the other hand, don't have event lists as their value is always assigned immediately.

Example 1 HWMod **WS24** s 1 architecture arch of sim01 is 2 signal s : std_ulogic := '0'; current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: Example 1 begin 5 active processes: 6 s <= '1'; completed processes: 7 wait; 8 end process; signal value scheduled events 9 end architecture; s ___ ___

architecture arch of sim01 is	4
begin	current simulation time: 0 ns
p0 s process	scheduled processes: p/l
begin	active processes:
wair.	completed processes:
end process; end architecture;	signal value scheduled events
	4 101

During initialization of the simulator, all the initial values of all signals in a design are derived from their declarations. Hence, in this example s is set to the value zero. Moreover, all processes in the design are scheduled for execution.

Example 1 HWMod **WS24** s 1 architecture arch of sim01 is 2 signal s : std_ulogic := '0'; current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: p0 Example 1 begin 5 active processes: 6 s <= '1'; completed processes: 7 wait; 8 end process; scheduled events signal value 9 end architecture; ' 0 ' s ___

architecture arch of simil is	4		
kegin	current a	inulation	time: 0 ns
pô a procesa	schedule	d proces	565
niged	active pr	COSSES.	pð
white	complete	d proces	808:
end process;	alasad	and an	and and deal as seen to
end architecturey		1.1.1	SCHOOLS WELL

Then the simulation starts to run the processes. As there is only one in our simple example, p0 is executed.

Example 1 HWMod **WS24** s 1 architecture arch of sim01 is signal s : std_ulogic := '0'; 2 current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: Example 1 begin 5 active processes: p0 6 s <= '1'; completed processes: 7 wait; 8 end process; scheduled events signal value 9 end architecture; '0' s ___

and the concerner and the family is a set of the set of

When the simulator reaches the assignment statement it adds the value one to the event queue of the signal s and schedules it to run at simulation time 0 nanoseconds.

Example 1

HWMod WS24

Assign. Statements Waveforms Examples Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Waveforms Examples Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
	Example 1

```
1 architecture arch of sim01 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1';
7 wait;
8 end process;
9 end architecture;</pre>
```

S

current simulation time: 0 ns scheduled processes: active processes: p0 completed processes:

signal	value	scheduled events
S	' 0 '	'1' @ 0 ns

i architecture arch of simil is
 signal a : std_ulogio := '0';
 segis
 cores
 begin
 secis
 s

Next, since the simulation ran into an unconditional wait statement it marks the process as completed and will never execute it again. Afterwards, the simulation goes through all the event lists associated with the process and checks whether there are pending assignments.

Example 1

HWMod WS24

Example 1	

```
1 architecture arch of sim01 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1';
7 wait;
8 end process;
9 end architecture;</pre>
```

s	

current simulation time: 0 ns scheduled processes: active processes: completed processes: p0

signal	value	scheduled events
S	' 0 '	'1' @ 0 ns
architecture arch of sized is sized as at at_blacks - rdr, begin bogin a <- rdr, mod processes a <- rdr, mod processes a strontecture, begin a <- rdr, mod processes a strontecture, begin a <- rdr, begin a strontecture, begin a <- rdr, begin a strontecture, begin a <- rdr, begi

As s is scheduled to transition to one at 0 nanoseconds, the assignment is popped from the event queue and executed. Note that the simulation time is still 0 nanoseconds so far, as none of the encountered statements did consume any "physical" time.

Example 1

HWMod WS24

Example 1

```
1 architecture arch of sim01 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1';
7 wait;
8 end process;
9 end architecture;</pre>
```

current simulation time: 0 ns
scheduled processes:

active processes:

s

completed processes: p0

signal	value	scheduled events
S	11	



The simulator now recognizes that there are neither scheduled events for any signal, nor any scheduled processes to run. Thus, it concludes that no further signal transitions are possible. Hence, the simulation time does not need to be advanced anymore.

Example 1 HWMod **WS24** s 1 architecture arch of sim01 is 2 signal s : std_ulogic := '0'; current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: Example 1 begin 5 active processes: 6 s <= '1'; completed processes: p0 7 wait; end process; 8 signal value scheduled events 9 end architecture; 111 S ___

Simulation End

No scheduled events or processes \rightarrow no further signal transitions possible.

Indefined are subject of stable to the st

If we would still forcefully simulate beyond this point we would see in the timing diagram that, indeed, no further signal transitions happen. Before we continue to the next example, we want to point out that the explanations in this lecture use a slight simplification. The simulation actually maintains an event list for every driver of a signal. Hence, if there are two processes writing to the same signal there are also two event lists.

	Example 1	
HWMod WS24		
Signal Assign. Imroduction Assign. Statements Waveforms Examples Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	<pre>1 architecture arch of sim01 is 2 signal s : std_ulogic := '0'; 3 begin 4 p0 : process 5 begin 6 s <= '1'; 7 wait; 8 end process; 9 end architecture;</pre>	s current simulation time: 5 ns scheduled processes: active processes: completed processes: p0 <u>signal value scheduled events</u> s '1'

Simulation End

No scheduled events or processes \rightarrow no further signal transitions possible.

Let's now look at a slightly more interesting example. Here we have an architecture that looks very similar to the one from the previous slide. The only difference is that the waveform now uses an after clause. The semantic of this construct is not hard to guess. The signal assignment should not happen at simulation time 0 but after 1 nanosecond. Hence, we would expect a timing diagram where s is low for 1 nanosecond, then transitions to high and stays there.



1 architecture arch of sim01 is 2 stgmal a rate ulogic i= "0" 3 begin 4 p0 : process 5 begin 6 4 <- "1" After 1 may 7 unity 8 eed arcoses; 9 eed arcotecture; a current simulation time: 0 ns scheduled processes: p0 sche processes: completed processes: signal value scheduled -----

The initial state of the simulator is the same as for the previous example.

	Example 2	
HWMod WS24 Signal Assign. Introduction Assign. Statements Waveforms Examples Examples Example 1 Example 3 Example 3 Example 4 Example 5 Concurrent Assign.	<pre>1 architecture arch of sim02 is 2 signal s : std_ulogic := '0'; 3 begin 4 p0 : process 5 begin 6 s <= '1' after 1 ns; 7 wait; 8 end process; 9 end architecture;</pre>	s

inchirecture arch of simd2 is
j signal a : st_uluitogio := '0';
j begin
i poi
i begin
i begin
i a co 'i' after i rag
i a co 'i' after i rag
i a co 'pootes;
i e cd architecture;

 current aimulation time: 0 ns scheduled processes: active processes: p0 completed processes: __signal__value__acheculed events

As before, the simulator then executes the single process p0. When reaching the assignment statement, the simulator schedules an appropriate event. However, compared to the previous example, the simulator now schedules this event not for 0 but rather for 1 nanosecond simulation time. The reason is of course the 1 nanosecond after clause.

Example 2 HWMod **WS24** s 1 architecture arch of sim02 is 2 signal s : std_ulogic := '0'; current simulation time: 0 ns 3 begin Example 2 4 p0 : process scheduled processes: 5 begin active processes: p0 6 s <= '1' after 1 ns; completed processes: 7 wait; end process; 8 signal value scheduled events 9 end architecture; '0' '1' @ 1 ns s

samplifecture areah of simble is sleqis Current simulation time: 0 ns scheduled processes: advise processes: completed processes: p0 signal value scheduled events

At the wait statement, the process is marked as complete but no actual value update to the signal s takes place.

Example 2

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Example 8 Example 9 Example 3 Example 4 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim02 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1' after 1 ns;
7 wait;
8 end process;
9 end architecture;</pre>
```

S

current simulation time: 0 ns scheduled processes: active processes: p0

signal	value	scheduled events
S	' 0 '	'1' @ 1 ns

sample are are of sholl is 'a' signal a static logic is '0' y
seque a pois process schedule pr
bela 's' after i ray schedule pr
s wity complete pr
s wity schedule pr
s wity schedule pr
s with schedu

The simulator then goes through all the scheduled events and advances the simulation time to the point in time of the next event.

Example 2

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Example 3 Example 3 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim02 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1' after 1 ns;
7 wait;
8 end process;
9 end architecture;</pre>
```

s _

current simulation time: 1 ns scheduled processes: active processes: p0

signal	value	scheduled events
S	' 0 '	'1' @ 1 ns

sample of a state of similar is a state of a state

 initiation time: 1 ns scheduled processes: active processes: completed processes: p0

Then the value assignment is executed.

Example 2 HWMod **WS24** 1 architecture arch of sim02 is s signal s : std_ulogic := '0'; 2 current simulation time: 1 ns 3 begin Example 2 p0 : process 4 scheduled processes: 5 begin active processes: 6 s <= '1' after 1 ns;</pre> completed processes: p0 7 wait; 8 end process; value scheduled events signal 9 end architecture; 11' s ___

anotherese acts of status is a signal as a status of status is a status of status is a status in the status is a status is a status in the status is a status is a status in the status is a status is a

Again, there are no further events or scheduled processes and hence no more signal changes. The simulation thus terminates, and we can go to the next example.

Example 2

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Example 8 Example 9 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim02 is
2 signal s : std_ulogic := '0';
3 begin
4 p0 : process
5 begin
6 s <= '1' after 1 ns;
7 wait;
8 end process;
9 end architecture;</pre>
```



current simulation time: 5 ns scheduled processes: active processes: p0

signal	value	scheduled events
S	' 1'	

i architecture arch of sim01 is
 signal s : std_ulogic := '0';
 z Begis
 p0 : process
 s Degin
 s A <= 1' after 1 ns,
 s A <= 1' after 1 ns,
 s and process
 unit;
 edi process;
 s edi process;
</pre>

Again, the code is very similar, but this time the signal s should transition back to 0 after 2 nanoseconds.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is signal s : std_ulogic := '0'; 2 3 begin 4 p0 : process 5 begin s <= '1' after 1 ns, 6 '0' after 2 ns; 7 8 wait; 9 end process; 10 end architecture;

1 schitecture arch of slud) is
2 signal a siste ulogic (- '0')
3 begin
4 p0
6 s <- '1' after 1 ms,
7 vali;
8 end process;
8 end process;
9 end process;

a current simulation time: 0 ns current simulation time: 0 ns active processes: completed processes: signal value scheduled event

The initialization is again the same as in the previous examples.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: p0 Example 3 begin 5 active processes: 6 s <= '1' after 1 ns,</pre> completed processes: 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '0' s ___ 10 end architecture;



current simulation time: 0 ns scheduled processes: policie processes: p0 completed processes: signal value scheduled events

Executing the assignment statement now schedules 2 events for the signal s.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: Example 3 5 begin active processes: p0 6 s <= '1' after 1 ns,</pre> completed processes: 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '1' @ 1 ns, '0' @ 2 ns '0' s 10 end architecture;

isochisecure arch of simil s signals = std_ulogic = begin s begin s begin s s < 'i's after 1 ns, s begin s similar s s < 'i's after 2 ns; s contexts s begin s s < begin s s < 'i's after 2 ns; s contexts s s < begin s s
s s
s
 current simulation time: 0 ns scheduled processes: completed processes: completed processes: p0 tigral value scheduled events

At the wait statement the simulator marks the process as complete and

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 0 ns 3 begin 4 p0 : process scheduled processes: Example 3 begin 5 active processes: 6 s <= '1' after 1 ns,</pre> completed processes: p0 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '1' @ 1 ns, '0' @ 2 ns '0' s 10 end architecture;

Lurrent simulation time: 1 ns
cheduled processes:
completed processes: p0
signal value scheduled events

then advances the simulation time to the next scheduled event, which is at 1 nanosecond.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 1 ns 3 begin 4 p0 : process scheduled processes: Example 3 5 begin active processes: 6 s <= '1' after 1 ns,</pre> completed processes: p0 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '1' @ 1 ns, '0' @ 2 ns '0' s 10 end architecture;

: architecture anch of simbling : angula a : acd_uiogio := 'd : begin : begin : begin : begin : begin : a <= '1' sites 1 ns, : 'd' after 2 nsj : waitj : e of processj : e of processj

After executing the assignment the simulation again looks for the next scheduled event,

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 1 ns 3 begin 4 p0 : process scheduled processes: Example 3 begin 5 active processes: 6 s <= '1' after 1 ns,</pre> completed processes: p0 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '1' '0' @ 2 ns s 10 end architecture;

: suchitecture arch of simil : signal a : std_ulogic :-: begin : process : begin : a \sim '1' after 1 sa, : '0' after 2 saj : wait; : est process; : est process; Lin
 Long
 Long

advances the simulation time accordingly and executes it.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 2 ns 3 begin 4 p0 : process scheduled processes: Example 3 5 begin active processes: 6 s <= '1' after 1 ns,</pre> completed processes: p0 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '0' s ___ 10 end architecture;

a current simulation time: 5 ns scheduled processes: active processes: completed processes: p0 algnal value scheduled event

Since, there are no scheduled events in the event queue, the simulation ends here.

Example 3 HWMod **WS24** 1 architecture arch of sim03 is s signal s : std_ulogic := '0'; 2 current simulation time: 5 ns 3 begin 4 p0 : process scheduled processes: Example 3 begin 5 active processes: 6 s <= '1' after 1 ns,</pre> completed processes: p0 7 '0' after 2 ns; 8 wait; scheduled events signal value end process; 9 '0' s ___ 10 end architecture;

 $\label{eq:constraints} \begin{array}{l} \mbox{architestrainswareh of simplify a simplify a star of a simplify a simplify$

In this example, we want to show how multiple processes are handled and how the sensitivity list is used to schedule new process executions.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign: Statements
Waveforms
Example 1
Example 2
Example 3
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
2
    signal s, t : std_ulogic := '0';
3 begin
    p0 : process
4
5 begin
6 s <= '1' after 2 ns,
7
           '0' after 4 ns,
           '1' after 6 ns;
8
9
    wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
    t <= s after 1 ns;
14
    end process;
15
16 end architecture;
```

In the beginning the signals s and t are set to 0, and both processes are scheduled for execution.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```

s t

current simulation time: 0 ns scheduled processes: p0, p1 active processes: completed processes:

signal	value	scheduled events
S	' 0 '	
t	' 0 '	



Let's say process p0 is run first. Executing the assignment statement schedules three events for the signal s.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Examples
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
8
            '1' after 6 ns;
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
      t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```

s t

current simulation time: 0 ns scheduled processes: p1 active processes: p0 completed processes:

signal	value	scheduled events
S	' 0 '	'1' @ 2 ns, '0' @ 4 ns,
		'1' @ 6 ns
t	' 0 '	



Then the unconditional wait statement terminates the process p0.

Example 4

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Example 1 Example 3 Example 3 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
      wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
     t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```

s t

current simulation time: 0 ns scheduled processes: p1 active processes: completed processes: p0

signal	value	scheduled events
S	' 0 '	'1' @ 2 ns, '0' @ 4 ns,
		'1' @ 6 ns
t	' 0 '	

<pre>i architecture arch of al i aigeal a, t : arciulo 3 begin 4 p0 : process 5 begin 6 a <- 1' after 2 ns 7 40' after 4 ns 8 1' after 6 ns 8 1' after 6 ns</pre>	A IA A A CURRENT A CURRENTA A CURRENTA A CURRENT A CURRENT A CURRENTA	nulation I proces cesses: I proces	time: 0 ns mas: pi mas: pi
to end processy	signal	value	scheduled events
11 of a process (all)	4	101	11 8 2 se, 10 8 4 se,
13 begin			111 8 6 ma
te - a after 1 may			.0.61.00
is end processy			

Since, there are still processes scheduled, the simulator does not yet advance the simulation time, but executes p1 first. The single assignment statement in p1 specifies that the signal t shall get the current value of s after 1 nanosecond. Hence, an appropriate event is scheduled for simulation time 1 nanosecond and the process execution is completed.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Example 1
Example 3
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
8
            '1' after 6 ns;
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;
14
    end process;
15
16 end architecture;
```

s t

current simulation time: 0 ns scheduled processes: active processes: p1

completed processes: ${\tt p0}$

signal	value	sche	du	led	l ever	nts				
S	' 0 '	' 1 '	g	2	ns,	' 0'	Q	4	ns,	
		' 1 '	g	6	ns					
t	' 0 '	' 0 '	g	1	ns					

 An information with a field if is a

Next, the simulation time advances to the next event and the associated assignment is executed. However, since the value of signal t is already 0, nothing changes.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
     signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
8
            '1' after 6 ns;
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```

-

s

t

current simulation time: 1 ns scheduled processes: active processes:

completed processes: p0, p1

signal	value	scheduled events
S	' 0 '	'1' @ 2 ns, '0' @ 4 ns,
		'1' @ 6 ns
t	' 0 '	

1 2 3 4 8 6 7 8 8 7 8 8	architecture ands of stabl is signal a, t : std_uliqie := '0'; begin begin d : process begin < '1' after 2 sa, '1' after 4 sa; '1' after 4 sa; '1' after 4 sa; '1' after 4 sa;	t current s schedule active pr complete	inulation d process coesses d proces	50mm: 2 mm mme: p1 mme: p0
10	end process;	signal	value	scheduled events
		4	114	"0" 8 4 14, "1" 8 6 K6
13	p1 + process(s11)	τ.	101	
13	Degin			
14	t <= a after 1 nay			
18	end process;			
16.1	end architectures			

The next scheduled event is at 2 nanoseconds and sets the signal s to high. Notice that since s is in the sensitivity list of process p1, and it changed its value, the process needs to be rescheduled for execution.

Example 4

HWMod WS24

ignal Assign. Introduction Assign. Statements Waveforms Examples Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim04 is
     signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```



current simulation time: 2 ns scheduled processes: p1 active processes: completed processes: p0

signal	value	scheduled events	
S	11	'0' @ 4 ns, '1' @ 6 ns	
t	' 0 '		

A strategy of the strategy of the

Thus, in the next step p1 is executioned again, which in turn schedules an event for the signal t at simulation time 3 nanoseconds.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Example 1
Example 3
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
      t <= s after 1 ns;
14
    end process;
15
16 end architecture;
```



current simulation time: 2 ns scheduled processes: active processes: p1

completed processes: p0

signal	value	sche	du	led	l ever	nts				
S	11	' 0 '	g	4	ns,	'1'	g	6	ns	
t	' 0 '	' 1 '	Q	3	ns					



After advancing the simulation time this assignment is actually executed and ${\rm t}$ is set to 1.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Examples
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```



current simulation time: 3 ns scheduled processes: active processes:

completed processes: p0, p1

signal	value	scheduled events	
S	11	'0' @ 4 ns, '1' @ 6 ns	
t	' 1'		

Implementary stars of classics in a star in

The next few steps should be quite obvious, as they are just a repetition of the events that we already showed. We therefore only show them as animation steps.

Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign: Statements
Waveforms
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
     signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
8
            '1' after 6 ns;
9
     wait;
    end process;
10
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
    end process;
15
16 end architecture;
```



current simulation time: 4 ns scheduled processes: p1 active processes: completed processes: p0

signal	value	scheduled events
S	' 0 '	'1' @ 6 ns
t	' 1'	



Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;
14
15
    end process;
16 end architecture;
```



current simulation time: 4 ns scheduled processes: active processes: p1

completed processes: p0

signal	value	scheduled events	
S	' 0 '	'1' @ 6 ns	
t	' 1'	'0' @ 5 ns	



Example 4

HWMod WS24

```
introduction
Assign. Statements
Waveforms
Examples
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
15
    end process;
16 end architecture;
```



current simulation time: 5 ns scheduled processes: active processes:

completed processes: p0, p1

signal	value	scheduled events
S	' 0 '	'1' @ 6 ns
t	' 0 '	



Example 4

HWMod WS24

```
Signal Assign.
Introduction
Assign. Statements
Waveforms
Examples
Example 1
Example 3
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns,</pre>
6
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
14
       t <= s after 1 ns;</pre>
    end process;
15
16 end architecture;
```



current simulation time: 6 ns scheduled processes: p1 active processes: completed processes: p0

signal	value	scheduled events
S	' 1'	
t	' 0 '	



Example 4

HWMod WS24

```
introduction
Assign. Statements
Waveforms
Examples
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;
14
15
    end process;
16 end architecture;
```



current simulation time: 6 ns scheduled processes: active processes: p1

completed processes: p0

signal	value	scheduled events
S	' 1'	
t	' 0 '	'l' @ 7 ns



t arret simulation time: 7 ns checklied processes: completed processes: p0, p1 signal value scheckled event

Example 4

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Example 1 Example 2 Example 3 Example 3 Example 4 Example 5 Concurrent Assign.

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
15
    end process;
16 end architecture;
```



current simulation time: 7 ns scheduled processes: active processes:

completed processes: p0, p1

-

signal	value	scheduled events
S	11	
t	'1'	



Example 4

HWMod WS24

```
introduction
Assign: Statements
Waveforms
Example 1
Example 2
Example 3
Example 4
Example 5
Concurrent Assign.
```

```
1 architecture arch of sim04 is
    signal s, t : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
7
            '0' after 4 ns,
            '1' after 6 ns;
8
9
     wait;
10
    end process;
11
    p1 : process(all)
12
    begin
13
       t <= s after 1 ns;</pre>
14
15
    end process;
16 end architecture;
```



current simulation time: 8 ns scheduled processes: active processes:

completed processes: p0, p1

signal	value	scheduled events
S	11	
t	'1'	

: a subdispectrue and h of sind(is) signal + of subdisplate - "O"y begin i point i poi

Finally, in our last example we demonstrate what happens when a value is assigned to a signal that already has scheduled events in its event queue.

Example 5

HWMod WS24

Example 5

1	architect	ure	arch (сf	sin	105	is	
2	signal	s :	std_u	log	gic	:=	' 0'	;
3	begin							
4	p0 : p1	codes	ss					
5	begin							
6	s <=	' 1'	after	2	ns,			
7		' 0 '	after	4	ns,			
8		' 1'	after	6	ns			
9		' 0 '	after	8	ns;			
10	wait	for	3 ns;					
11	s <=	' 1'	after	2	ns;			
12	wait;	;						
13	end pro	ocess	5;					
14	end archi	itect	ure;					

inditional and alarge in any inditional inditatinal inditional inditional inditiona

As before, the signal ${\rm s}$ is initialized to be low, and the single process ${\rm p0}$ is scheduled for execution.

Example 5

HWMod WS24

Assign. Statements Waveforms Examples Example 1 Example 2 Example 4 Example 4 Example 5 Concurrent Assign.	
Waveforms Examples Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Examples Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Example 1 Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Example 2 Example 3 Example 4 Example 5 Concurrent Assign.	
Example 3 Example 4 Example 5 Concurrent Assign.	
Example 4 Example 5 Concurrent Assign.	
Example 5 Concurrent Assign.	
	Example 5

1	architect	ure	arch	of	sim	105	is	
2	signal	s :	std_u	llog	gic	:=	' 0 '	;
3	begin							
4	p0 : pr	oces	ss					
5	begin							
6	s <=	'1'	after	2	ns,			
7		' 0 '	after	4	ns,			
8		' 1 '	after	6	ns			
9		' 0 '	after	8	ns;			
10	wait	for	3 ns;					
11	s <=	' 1 '	after	2	ns;			
12	wait;							
13	end pro	cess	;					
14	end archi	tect	ure;					

current simulation time: 0 ns
scheduled processes: p0
active processes:

completed processes:

s

signal	value	scheduled events		
S	' 0 '			
	architecture arch of sim05 is signal s : std_ologic :- '0'; begin p0 : process begin	4 current si schedule	mulation d proces	time: 0 ns
----	--	-----------------------------	----------------------	---
2	a <- '1' after 2 na, '0' after 4 na, '1' after 6 na	complete signal	d proces value	ses: scheduled events
10	usit for 3 may a <- '1' after 2 may usity end processy end architecturey	1	.0.	'1' 0 2 20, '0' 0 4 20, '1' 0 6 20, '0' 0 8 20

Next, the process becomes active and the simulator reaches the first statement, highlighted on the slide. The waveform that is assigned to s in this example describes two 2 nanosecond pulses. The fist pulse is scheduled at 2 nanoseconds of simulation time, while the second one is scheduled for 6 nanoseconds. Hence, the simulator schedules four separate events, ordered by their respective time, and continues.

Example 5

HWMod WS24

Example 5	

```
1 architecture arch of sim05 is
     signal s : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns.
6
            '0' after 4 ns.
7
            '1' after 6 ns
8
9
            '0' after 8 ns;
       wait for 3 ns;
10
       s <= '1' after 2 ns;</pre>
11
12
       wait;
    end process;
13
14 end architecture;
```

S

current simulation time: 0 ns scheduled processes: active processes: p0 completed processes:

signal	value	scheduled events						
S	' 0 '	'1' @ 2 ns, '0' @ 4 ns	,					
		'1' @ 6 ns, '0' @ 8 ns						

	architecture arch of sim05 is signal s ; std_ulogic :- '0'; begin	4 L	mulation	time: 2 ns
	po i process	scheduled	s proces	see: po
	a c- 'l' after 2 pa.	active pro	COSSES.	
	'd' after 4 ns,	completes	d proces	848.
	'if after 6 ns			
	101 after 8 pag	signal	value	scheduled events
10	wait for 3 may		101	111 8 2 may 101 8 4 may
	a ce 'if after 2 may			111 8 6 may 101 8 8 ma
12	waity			
13	end process;			
	end architecture;			

Now the simulator encounters the wait for statement. This statement results in the process being suspended until three nanoseconds of simulation time have passed.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
    p0 : process
4
    begin
5
6
       s <= '1' after 2 ns,</pre>
            '0' after 4 ns.
7
            '1' after 6 ns
8
            '0' after 8 ns;
9
       wait for 3 ns;
10
11
      s <= '1' after 2 ns;</pre>
      wait;
12
    end process;
13
14 end architecture;
```

current simulation time: 2 ns scheduled processes: p0 active processes: completed processes:

s

signal	value	scheduled events						
S	' 0 '	'1' @ 2 ns, '0' @ 4 ns	,					
		'1' @ 6 ns, '0' @ 8 ns						

<pre>i architecture arch of sim05 is i signal s i std_ulogic i= "0";</pre>	- <u></u>
<pre>> begin 4 p0 + process 5 begin 6 a <= '1' after 2 ns, 7 '0' after 4 ns,</pre>	current almulation time: 2 ns scheduled processes: p0 adlive processes: completed processes:
 '1' After 6 ns '0' After 8 nsj whit for 2 nsj a <- '1' After 2 nsj 	signal value scheduled events 6 '1' '0' @ 4 may '1' @ 6 may '0' @ 8 may '1' @ 6 may '1' @ 6 may '1' @ 6 may
13 whity 13 end processy 14 end architecturey	

However, in the meanwhile the simulator will go through the event queues, encounter the assignment scheduled at 2 nanoseconds for s, and actually execute it. Note that the process still remains suspended.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
6
       s <= '1' after 2 ns,</pre>
            '0' after 4 ns.
7
            '1' after 6 ns
8
            '0' after 8 ns;
9
       wait for 3 ns;
10
11
       s <= '1' after 2 ns;</pre>
12
       wait;
    end process;
13
14 end architecture;
```

current simulation time: 2 ns scheduled processes: p0 active processes: completed processes:

s

signal	value	scheduled events							
S	'1'	' 0 '	g	4	ns,	'1'	Q	6	ns
		' 0 '	Q	8	ns				

i architecture arch of sladi is
i architecture arch of sladi is
i architecture arch of sladi is
po a prises
i po a prise
i po a

The simulator finally advances the time to 3 nanoseconds which wakes up the process again.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
4
    p0 : process
    begin
5
6
      s <= '1' after 2 ns,</pre>
            '0' after 4 ns.
7
            '1' after 6 ns
8
            '0' after 8 ns;
9
      wait for 3 ns;
10
11
     s <= '1' after 2 ns;
      wait;
12
    end process;
13
14 end architecture;
```

s		

current simulation time: 3 ns scheduled processes: active processes: p0 completed processes:

signal	value	scheduled events							
S	'1'	' 0 '	g	4	ns,	'1'	Q	6	ns
		' 0 '	Q	8	ns				

architecture arch of sim05 is aignal a ; and plogic ;- '0';		Ļг	
begin p0 s process	current s schedule	imulation d proces	time: 3 ns ses:
begin did offer dies	active processes: p0		pô
"O" after 4 ns,	complete	d proces	505:
"1" after 6 na "0" after 8 nas	signal	value	scheduled even
wait for 3 may	4	.1.	10.64 me*
a <- "1" after 2 nay			.0.68 = 0
unit; end process;			

The next statement is another assignment to s. The simulator will schedule this assignment which takes place at 5 nanoseconds. However, now something particularly noteworthy happens. Every time a new assignment to a signal happens in the same process, all old transactions that are projected to occur at or after the time at which the earliest new transaction is projected to occur are deleted. Note that the latter part of the last sentence is a direct quote from the VHDL reference.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
     signal s : std ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns.
6
            '0' after 4 ns,
7
            '1' after 6 ns
8
9
            '0' after 8 ns:
      wait for 3 ns;
10
       s <= '1' after 2 ns;</pre>
11
12
       wait;
    end process;
13
14 end architecture;
```

S	

current simulation time: 3 ns scheduled processes: active processes: p0 completed processes:

signal	value	sche	du	led	l ever	nts			
S	'1'	' 0 '	g	4	ns,	'1'	Q	6	ns
		' 0 '	Q	8	ns				

1 2 3 4 8 6 7	architecture arch of simd5 is signal s : ord_ulogic := "O"; begin p0 : process begin s <= "1" after 2 ns, "O" after 4 ns,	 current si schedule active pro complete 	mulation d process d process	5me: 2 p2 565:	0 na			
	the after a sec	signal	value	sche	duled	events		
- 10	whit for 3 new	4	111	.0.	8.4	ne, 11	. 8.5	ĉ
	a c- "1" after 2 may							
12	waity							
	end process;							

For our concrete example this means that every event at or after 5 nanoseconds is deleted from the event queue. Thus, the transition to zero at 4 nanoseconds is kept, while every thing else is replaced by a new event at 5 nanoseconds setting s to one again. Note that you have made use of this behavior already when you used default assignments to mitigate latches, or other overriding assignments.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
     signal s : std_ulogic := '0';
2
3 begin
    p0 : process
4
5
    begin
       s <= '1' after 2 ns.
6
            '0' after 4 ns,
7
            '1' after 6 ns
8
9
            '0' after 8 ns:
       wait for 3 ns;
10
       s <= '1' after 2 ns;</pre>
11
12
       wait;
    end process;
13
14 end architecture;
```

S		-		
currer	ıt simı	ulatio	on tim	ie:

scheduled processes: active processes: p0 completed processes:

signal	value	sche	du	led	l ever	nts			
S	' 1 '	' 0 '	g	4	ns,	' 1 '	9	5	ns

3 ns

	architecture arch of sim05 is signal a ; std plogic ;- '0';		Ļг	
	pd s process	current s schedule	imulation d proces	577.0 685
	begin	active pr	DC08888	
	for after 4 za,	complete	d proces	505:
	"1" After 6 ns	signal	value	sch
ŝ	wait for 3 map	4	.1.	*0
	a <= '1' after 2 may			
	unit;			
	end process;			
	and anobitedfures			

Finally, the unconditional wait statement is reached which completes the process. As in previous examples, the simulator will now go through its list of scheduled events and advances the simulation time to the one of the next event.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
    p0 : process
4
    begin
5
6
       s <= '1' after 2 ns,</pre>
            '0' after 4 ns.
7
            '1' after 6 ns
8
            '0' after 8 ns;
9
       wait for 3 ns;
10
11
       s <= '1' after 2 ns;</pre>
       wait;
12
    end process;
13
14 end architecture;
```

3								
current simulation time: 3 ns								
scheduled processes:								
active processes:								
completed processes: p0								
signal value	signal value scheduled events							
s '1'	' 0'	Q	4	ns,	' 1'	Q	5	ns



As a result the simulation time is set to 4 nanoseconds, and $_{\rm S}$ is set to zero.

	Example 5	
HWMod WS24		
Signal Assign. Introduction Assign. Statements Waveforms Examples Example 1 Example 3 Example 3 Example 4 Example 4 Example 5 Concurrent Assign.	<pre>1 architecture arch of sim05 is 2 signal s : std_ulogic := '0'; 3 begin 4 p0 : process 5 begin 6 s <= '1' after 2 ns, 7 '0' after 4 ns, 8 '1' after 6 ns 9 '0' after 8 ns; 10 wait for 3 ns; 11 s <= '1' after 2 ns; 12 wait; 13 end process; 14 end architecture;</pre>	s current simulation time: 4 ns scheduled processes: active processes: completed processes: p0 signal value scheduled events s '0' '1' @ 5 ns



a current simulation time: 5 ns scheduled processes: active processes: completed processes: p0 signal value scheduled events

The last event happens at 5 nanoseconds, which sets s back to one.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
4
    p0 : process
    begin
5
6
  s <= '1' after 2 ns,
7
           '0' after 4 ns.
            '1' after 6 ns
8
            '0' after 8 ns;
9
10 wait for 3 ns;
11 s <= '1' after 2 ns;</pre>
     wait;
12
    end process;
13
14 end architecture;
```

S			

current simulation time: 5 ns scheduled processes: active processes: p0

signal	value	scheduled events
S	'1'	

	architecture arch of sim05 is signal s : std_ulogic := "0";	•
	begin	current simulation time: 10 ns
	pů s process	scheduled processes:
	begin	and the second second
	a <- '1' after 2 ms,	active processes.
	101 After 4 zs,	compresed processes: po
	'l' after 6 ns	
	101 After 8 paul	signal value scheduled events
10	wait for 3 may	4 11 ¹
	a ce 'i' after 2 may	
12	waity	
13	end process;	
14	end architectures	

Since there are no more scheduled events or processes the simulation terminates. Hence, further advancing the simulation time does not lead to any more signal changes.

Example 5

HWMod WS24

Example 5

```
1 architecture arch of sim05 is
    signal s : std_ulogic := '0';
2
3 begin
4
    p0 : process
    begin
5
6
    s <= '1' after 2 ns,
7
           '0' after 4 ns.
           '1' after 6 ns
8
           '0' after 8 ns;
9
  wait for 3 ns;
10
11
     s <= '1' after 2 ns;
      wait;
12
    end process;
13
14 end architecture;
```

s				
current si	mulation	time: 10 ns		
scheduled processes:				
active pro	cesses:			
complete	d proces	ses: p0		
signal	value	scheduled events		
s	' 1 '			

−Signal Assignments └─Concurrent Assign. └─**Concurrent Signal Assignments**

Before we close this lecture, let us also briefly cover concurrent signal assignments. As already mentioned in the beginning of the lecture, for the scope of this course, you can consider concurrent assignments and assignments in processes as using the same syntax rules. However, the question remains, how they should be interpreted in simulation.

Concurrent Signal Assignments

HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Examples Concurrent Assign. Largely use the same syntax as assignment statements in processes

Largely use the same syntax as assignment statements in processes

─Signal Assignments └─Concurrent Assign. └─Concurrent Signal Assignments

Largely use the same syntax as assignment statements in processes
 VHDL Reference
 "For any concurrent signal assignment statement, there is an equivalenprocess statement with the same meaning."

The VHDL reference manual states that for any concurrent signal assignment statement, there is an equivalent process statement with the same meaning. It then goes on, presenting a detailed transformation algorithm that is guaranteed to preserve the meaning of the original assignment.

Concurrent Signal Assignments



HWMod WS24

Signal Assign. Introduction Assign. Statements Waveforms Examples Concurrent Assign.

- Largely use the same syntax as assignment statements in processes
- VHDL Reference

"For any concurrent signal assignment statement, there is an equivalent process statement with the same meaning."

─Signal Assignments └─Concurrent Assign. └─Concurrent Signal Assignments

 Langhy load the same syntax as assignment statements in processes
 WFUR, Holewoor Processing construction of the same meaning- process statement with the same meaning-process statement with the

However, to the extent we covered concurrent assignments in this course you can simply think of them as being embedded in a process with an all sensitivity list. If there are only constants on the left-hand-side than the assignment then the process is only executed once.

Concurrent Signal Assignments

- Largely use the same syntax as assignment statements in processes
- VHDL Reference

Concurrent Assign

"For any concurrent signal assignment statement, there is an equivalent process statement with the same meaning."

Think of them as embedded in a process with an all sensitivity list

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.



Signal Assign. Introduction Assign. Statements Naveforms Examples Concurrent Assign.

Lecture Complete!

Modified: 2025-03-12, 16:33 (21636bb)