Hardware Modeling [VU] (191.011)
– WS25 –
Signal Assignments

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Modified: 2026-01-10, 08:00 (f8a58e9)

Introduction

- Recap
  - Two types
    - Assignment statements in processes
    - Concurrent signal assignments
  - Fundamentally different from variable assignments
  - Only covered very basic value assignments
- This lecture
  - More systematic look at the signal assignment syntax
  - For simulations signal assignments
    - can be with equipped with timing information
    - can describe complex waveforms
  - Gain some insight into the simulator

Signal Assignment Statements (for Processes)

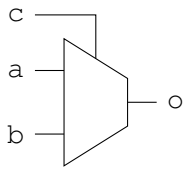
- **Signal assignments in processes**
  - Concurrent signals assignments have a similar syntax
  - Covered separately at the end of the lecture
- Signal assignment statement syntax

```
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 Assignment Statements - Multiplexer Example



Simple

```
1 process(all)
2 begin
3   o <= (a and not c) or
4         (b and c);
5 end process;
```

Conditional

```
1 process(all)
2 begin
3   o <= a when c = '0'
4        else b;
5 end process;
```

Selected

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

Simple Signal Assignments

- 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
  - Optional delay mechanism (covered in separate lecture)
  - Waveform (a series of expressions with timing information)

Conditional Assignments

- 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
- Each of the expressions that are assignment are waveforms

Waveforms

- Waveform syntax

```
waveform ::=
    waveform_element { , waveform_element }
  | unaffected
```

- Comma-separated list of waveform elements
- Special case: `unaffected`
  - `s <= unaffected;` ⇔ `null;`
  - `s <= unaffected when c else '0';` ⇔

```
if not c then
  s <= '0';
end if;
```

Waveforms (cont'd)

- Waveform element syntax

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

Waveforms (cont'd)

- Simple waveform assignment examples
  - `s <= t;`
  - `s <= '0' 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;
    ```

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;
```

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

signal	value	scheduled events
s	--	--

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

signal	value	scheduled events

Example 2


```
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;
```

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

signal	value	scheduled events
s	'0'	--

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

signal	value	scheduled events

Example 3


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

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

signal	value	scheduled events
s	'0'	--

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

signal	value	scheduled events

Example 4


```
1  architecture arch of sim04 is
2    signal s, t : 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      wait;
10   end process;
11
12   p1 : process(all)
13   begin
14     t <= s after 1 ns;
15   end process;
16 end architecture;
```

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

signal	value	scheduled events
s	'0'	--
t	'0'	--

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

Example 5


```
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 <= '0' after 2 ns;
12     wait;
13   end process;
14 end architecture;
```

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

signal	value	scheduled events
s	'0'	--

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

signal	value	scheduled events

Concurrent Signal Assignments

- 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."

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

```
s <= t after 1 ns;   ⟺   process(all)
                         begin
                           s <= t after 1 ns;
                         end if;
```

Lecture Complete!