Hardware Modeling [VU] (191.011)
– WS25 –
Simulation and Testbenches

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

Modified: 2025-12-16, 16:03 (f8a58e9)
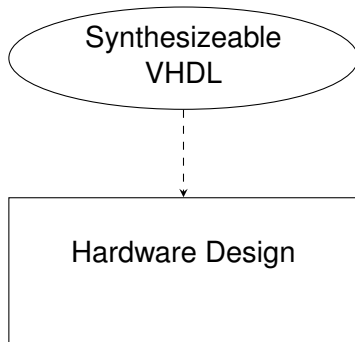
HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Introduction (cont'd)

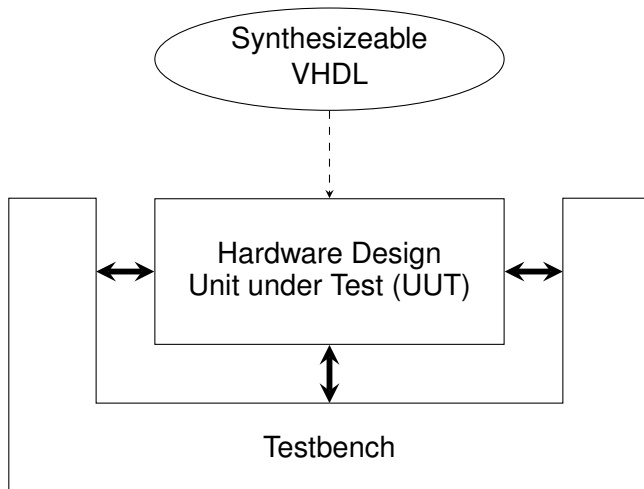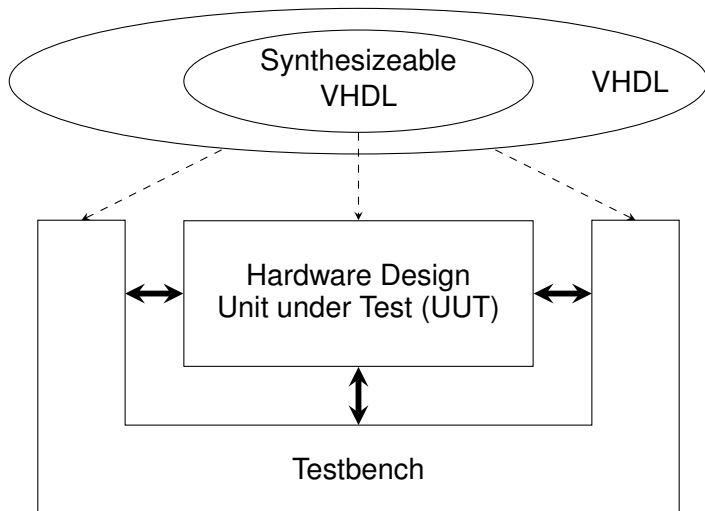- Testbenches are regular entities/architectures

- Testbenches are regular entities/architectures
- Testbench architectures
  - create the instance of the unit under test (UUT)
  - produce input signals for the UUT
  - check the outputs of the UUT for correctness

- Testbenches are regular entities/architectures
- Testbench architectures
    - create the instance of the unit under test (UUT)
    - produce input signals for the UUT
    - check the outputs of the UUT for correctness
- Testbench entities
    - have no ports
    - may have generics

- Testbenches are regular entities/architectures
- Testbench architectures
  - create the instance of the unit under test (UUT)
  - produce input signals for the UUT
  - check the outputs of the UUT for correctness
- Testbench entities
  - have no ports
  - may have generics
- Unit test for a hardware module

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Wait Statements

- Wait statements used so far
  - Unconditional "`wait;`" (at the end of a process)
  - Sensitivity lists (equivalent to a "`wait on [...];`" at the end of a process)

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

- Wait statements used so far
    - Unconditional "`wait;`" (at the end of a process)
    - Sensitivity lists (equivalent to a "`wait on [...];`" at the end of a process)
- Wait until a condition **becomes** true:
    `wait until condition;`
- Wait for a specific amount of time:
    `wait for expression;`

- Wait statements used so far
    - Unconditional "`wait;`" (at the end of a process)
    - Sensitivity lists (equivalent to a "`wait on [...];`" at the end of a process)
- Wait until a condition **becomes** true:
    `wait until condition;`
- Wait for a specific amount of time:
    `wait for expression;`
- Control the flow of time in the simulator

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

```
1 entity wait_example is
2 end entity;
3
```

Wait Statements - Example

```vhdl
1 entity wait_example is
2 end entity;
3
4 architecture arch of wait_example is
5  signal x : std_ulogic;
6 begin
```

Wait Statements - Example

```vhdl
1 entity wait_example is
2 end entity;
3
4 architecture arch of wait_example is
5   signal x : std_ulogic;
6 begin
7   proc_a : process
8   begin
9     x <= '0';
10     wait for 2.5 ns;
11     x <= '1';
12     wait;
13   end process;
```

```
1 entity wait_example is
2 end entity;
3
4 architecture arch of wait_example is
5  signal x : std_ulogic;
6 begin
7  proc_a : process
8  begin
9   x <= '0';
10   wait for 2.5 ns;
11   x <= '1';
12   wait;
13  end process;
14
15  proc_b : process
16  begin
17   wait until x = '1';
18   report "now=" & to_string(now);
19   wait;
20  end process;
21 end architecture;
```

```vhdl
1  entity wait_example is
2  end entity;
3
4  architecture arch of wait_example is
5   signal x : std_ulogic;
6  begin
7   proc_a : process
8   begin
9    x <= '0';
10    wait for 2.5 ns;
11    x <= '1';
12    wait;
13   end process;
14
15   proc_b : process
16   begin
17    wait until x = '1';
18    report "now=" & to_string(now);
19    wait;
20   end process;
21  end architecture;
```

Simulation output

```
Note: now=2500 ps
```

4

Example: Full Adder Testbench

Entity

```
1 entity fa is
2  port (
3   a    : in  std_ulogic;
4   b    : in  std_ulogic;
5   cin  : in  std_ulogic;
6   sum  : out std_ulogic;
7   cout : out std_ulogic
8  );
9 end entity;
```

Example: Full Adder Testbench

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Entity

```
1 entity fa is
2  port (
3   a    : in  std_ulogic;
4   b    : in  std_ulogic;
5   cin  : in  std_ulogic;
6   sum  : out std_ulogic;
7   cout : out std_ulogic
8  );
9 end entity;
```

Testbench

```
1 entity fa_tb is
2 end entity;
```

Example: Full Adder Testbench

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Entity

```
1 entity fa is
2  port (
3   a    : in  std_ulogic;
4   b    : in  std_ulogic;
5   cin  : in  std_ulogic;
6   sum  : out std_ulogic;
7   cout : out std_ulogic
8  );
9 end entity;
```

Testbench

```
1 entity fa_tb is
2 end entity;
3
4 architecture tb of fa_tb is
5  signal a, b, cin, sum, cout : std_ulogic;
6 begin
```

Example: Full Adder Testbench

Entity

```vhdl
1 entity fa is
2  port (
3   a    : in  std_ulogic;
4   b    : in  std_ulogic;
5   cin  : in  std_ulogic;
6   sum  : out std_ulogic;
7   cout : out std_ulogic
8  );
9 end entity;
```

Testbench

```vhdl
1 entity fa_tb is
2 end entity;
3
4 architecture tb of fa_tb is
5  signal a, b, cin, sum, cout : std_ulogic;
6 begin
7  uut : entity work.fa
8  port map (
9   a    => a,
10   b    => b,
11   cin  => cin,
12   sum  => sum,
13   cout => cout
14  );
15
```

5

Example: Full Adder Testbench

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Entity

```
1 entity fa is
2  port (
3   a     : in  std_ulogic;
4   b     : in  std_ulogic;
5   cin   : in  std_ulogic;
6   sum   : out std_ulogic;
7   cout  : out std_ulogic
8  );
9 end entity;
```

Testbench

```
1  entity fa_tb is
2  end entity;
3
4  architecture tb of fa_tb is
5   signal a, b, cin, sum, cout : std_ulogic;
6  begin
7   uut : entity work.fa
8    port map (
9     a     => a,
10    b     => b,
11    cin   => cin,
12    sum   => sum,
13    cout  => cout
14   );
15
16   stimulus : process [...]
17
18  end architecture;
```

5

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

```
1 stimulus : process
2 begin
3   a <= '0';
4   b <= '0';
5   cin <= '0';
```

```
1 stimulus : process
2 begin
3   a <= '0';
4   b <= '0';
5   cin <= '0';
6   wait for 1 ns;
```

Example: Full Adder Testbench (cont'd)

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

```
1 stimulus : process
2 begin
3   a <= '0';
4   b <= '0';
5   cin <= '0';
6   wait for 1 ns;
7
8   [...]
9
10  a <= '1';
11  b <= '1';
12  cin <= '1';
13  wait for 1 ns;
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

```vhdl
1 stimulus : process
2 begin
3   a <= '0';
4   b <= '0';
5   cin <= '0';
6   wait for 1 ns;
7
8   [...]
9
10  a <= '1';
11  b <= '1';
12  cin <= '1';
13  wait for 1 ns;
14
15  wait;
16 end process;
```

```
1  stimulus : process
2  begin
3    a <= '0';
4    b <= '0';
5    cin <= '0';
6    wait for 1 ns;
7
8    [...]
9
10   a <= '1';
11   b <= '1';
12   cin <= '1';
13   wait for 1 ns;
14
15   wait;
16 end process;
```

```vhdl
1  stimulus : process
2  begin
3    a <= '0';
4    b <= '0';
5    cin <= '0';
6    wait for 1 ns;
7
8    [...]
9
10   a <= '1';
11   b <= '1';
12   cin <= '1';
13   wait for 1 ns;
14
15   wait;
16 end process;
```

```vhdl
1  stimulus : process
2  begin
3    a <= '0';
4    b <= '0';
5    cin <= '0';
6    wait for 1 ns;
7
8    [...]
9
10   a <= '1';
11   b <= '1';
12   cin <= '1';
13   wait for 1 ns;
14
15   wait;
16 end process;
```

```vhdl
1 stimulus : process
2 begin
3   a <= '0';
4   b <= '0';
5   cin <= '0';
6   wait for 1 ns;
7
8   [...]
9
10  a <= '1';
11  b <= '1';
12  cin <= '1';
13  wait for 1 ns;
14
15  wait;
16 end process;
```

Assertions

Note

Checking waveforms is **hard** and **time-consuming**! It is completely infeasible to verify large designs this way.

Note

Checking waveforms is **hard** and **time-consuming**! It is completely infeasible to verify large designs this way.

Solution

The testbench validates the outputs programmatically, s.t. we don't have to look at the waveforms, using, e.g., assertions.

Note

Checking waveforms is **hard** and **time-consuming**! It is completely infeasible to verify large designs this way.

Solution

The testbench validates the outputs programmatically, s.t. we don't have to look at the waveforms, using, e.g., assertions.

However, ...

Simulation waveforms are still vitally important during development, especially when it comes to tracking down bugs.

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

- Can be viewed as "conditional report" statements
- Assertion syntax

```
assert condition
  [ report expression ] [ severity expression ];
```

- Can be viewed as "conditional report" statements
- Assertion syntax

```
assert condition
  [ report expression ] [ severity expression ];
```

- Severity level
  - Predefined enum type (standard package)

    ```
    type severity_level is
      (note, warning, error, failure);
    ```

  - Effect depends on the actual simulator and its configuration

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

- Can be viewed as "conditional report" statements
- Assertion syntax
  ```
  assert condition
    [ report expression ] [ severity expression ];
  ```
- Severity level
  - Predefined enum type (standard package)
    ```
    type severity_level is
      (note, warning, error, failure); IEEE SA
                                        OPEN
    ```
  - Effect depends on the actual simulator and its configuration
- Can be used in statement parts of entities, architectures, processes, subprograms, etc.

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Stimulus process with assertions

```
1 stimulus : process
2 begin
3  report "testing input 000";
4  a <= '0';
5  b <= '0';
6  cin <= '0';
7  wait for 1 ns;
8  assert cout = '0'
9    report "wrong carry" severity error;
10 assert sum = '0'
11   report "wrong sum" severity error;
12
13 [...]
14
15 wait;
16 end process;
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Stimulus process with assertions

```
1  stimulus : process
2  begin
3    report "testing input 000";
4    a <= '0';
5    b <= '0';
6    cin <= '0';
7    wait for 1 ns;
8    assert cout = '0'
9      report "wrong carry" severity error;
10   assert sum = '0'
11     report "wrong sum" severity error;
12
13   [...]
14
15   wait;
16 end process;
```

Simulator Output (QuestaSim)

```
# ** Note: testing input 000
#    Time: 0 ps  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 001
#    Time: 1 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 010
#    Time: 2 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 011
#    Time: 3 ns  Iteration: 0   Instance: /fa_tb
# ** Error: wrong carry
#    Time: 4 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 100
#    Time: 4 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 101
#    Time: 5 ns  Iteration: 0   Instance: /fa_tb
# ** Error: wrong carry
#    Time: 6 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 110
#    Time: 6 ns  Iteration: 0   Instance: /fa_tb
# ** Error: wrong carry
#    Time: 7 ns  Iteration: 0   Instance: /fa_tb
# ** Note: testing input 111
#    Time: 7 ns  Iteration: 0   Instance: /fa_tb
# ** Error: wrong carry
#    Time: 8 ns  Iteration: 0   Instance: /fa_tb
#  quit
# End time: xx:xx:xx on xx xx,xx, Elapsed time: xx
# Errors: 4, Warnings: 0
```

9

```
1 stimulus : process
```

Further Automation

```
1 stimulus : process
2   variable v : std_ulogic_vector(2 downto 0);
3   variable h : natural;
4 begin
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Further Automation

```
1 stimulus : process
2   variable v : std_ulogic_vector(2 downto 0);
3   variable h : natural;
4 begin
5   for i in 0 to 7 loop
```

Further Automation

```vhdl
1 stimulus : process
2  variable v : std_ulogic_vector(2 downto 0);
3  variable h : natural;
4 begin
5  for i in 0 to 7 loop
6   v := std_ulogic_vector(to_unsigned(i, v'length));
7   a <= v(0); b <= v(1); cin <= v(2);
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Further Automation

```vhdl
1 stimulus : process
2   variable v : std_ulogic_vector(2 downto 0);
3   variable h : natural;
4 begin
5   for i in 0 to 7 loop
6     v := std_ulogic_vector(to_unsigned(i, v'length));
7     a <= v(0); b <= v(1); cin <= v(2);
8     report "testing input " & to_string(v);
9     wait for 1 ns;
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

```
1 stimulus : process
2   variable v : std_ulogic_vector(2 downto 0);
3   variable h : natural;
4 begin
5   for i in 0 to 7 loop
6     v := std_ulogic_vector(to_unsigned(i, v'length));
7     a <= v(0); b <= v(1); cin <= v(2);
8     report "testing input " & to_string(v);
9     wait for 1 ns;
10
11     h := 0;
12     for j in v'range loop
13       if v(j) = '1' then
14         h := h + 1;
15       end if;
16     end loop;
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Further Automation

```vhdl
 1 stimulus : process
 2  variable v : std_ulogic_vector(2 downto 0);
 3  variable h : natural;
 4 begin
 5  for i in 0 to 7 loop
 6   v := std_ulogic_vector(to_unsigned(i, v'length));
 7   a <= v(0); b <= v(1); cin <= v(2);
 8   report "testing input " & to_string(v);
 9   wait for 1 ns;
10
11   h := 0;
12   for j in v'range loop
13    if v(j) = '1' then
14     h := h + 1;
15    end if;
16   end loop;
17   assert std_ulogic_vector(to_unsigned(h, 2)) = cout & sum
18    report "wrong output!" severity error;
19  end loop;
```

HWMod
WS25

Sim. & TBs
Introduction
Wait Statements
Full Adder Testbench
Assertions
Further Automation

Further Automation

```vhdl
1  stimulus : process
2   variable v : std_ulogic_vector(2 downto 0);
3   variable h : natural;
4  begin
5   for i in 0 to 7 loop
6    v := std_ulogic_vector(to_unsigned(i, v'length));
7    a <= v(0); b <= v(1); cin <= v(2);
8    report "testing input " & to_string(v);
9    wait for 1 ns;
10
11   h := 0;
12   for j in v'range loop
13    if v(j) = '1' then
14     h := h + 1;
15    end if;
16   end loop;
17   assert std_ulogic_vector(to_unsigned(h, 2)) = cout & sum
18    report "wrong output!" severity error;
19   end loop;
20   wait;
21  end process;
```

10

Lecture Complete!