VHDL

Ganz super: Xilinx XST User Guide; enthält Erklärungen und jede Menge Beispiele

*www.**xilinx**.com/itp/**xilinx**10/books/docs/**xst/xst**.pdf*

gute Einführung für Anfänger http://wwwlrh.fh-bielefeld.de/vhdl_vor/VHDL_V_B.htm

VHDL Tutorial: Learn by Example-- by Weijun Zhang, July 2001

die Beispiele laufen mit erweiterten ghdl Schaltern: ghdl -a --ieee=synopsys -fexplicit tb_ALU.vhd

GHDL (ein freier VHDL Compiler)

unter Linux problematisch nur im x86 Mode; funzt nicht mit wine

```
$ ghdl -a adder_tb.vhdl                 …  compilieren/analyze
$ ghdl -e adder_tb                      … link/elaborate entity

$ ghdl -r adder_tb                      … run

$ ghdl -r adder_tb –vcd=adder.vcd       … for graphical output
$ gtkwave adder.vcd
```

adder.vhd

```
 1      entity adder is
 2        -- i0, i1 and the carry-in ci are inputs of the adder.
 3        -- s is the sum output, co is the carry-out.
 4        port (i0, i1 : in bit; ci : in bit; s : out bit; co : out bit);
 5      end adder;
 6
 7      architecture rtl of adder is
 8      begin
 9         --  This full-adder architecture contains two concurrent assignment.
10         --  Compute the sum.
11         s <= i0 xor i1 xor ci;
```

```
12          --  Compute the carry.
13          co <= (i0 and i1) or (i0 and ci) or (i1 and ci);
14       end rtl;
```

$ ghdl -a adder.vhd

adder-testbench.vhd

```
--  testbench
 2       entity adder_tb is
 3       end adder_tb;
 4
 5       architecture behav of adder_tb is
 6          component adder
 7            port (
 8                  i0, i1 : in bit;
 9                      ci : in bit;
10                       s : out bit;
11                      co : out bit
12            );
13          end component;
14
15          for adder_0: adder use entity work.adder;
16          signal i0, i1, ci, s, co : bit;
17       begin
18          --  Component instantiation.
19          adder_0: adder port map (i0 => i0, i1 => i1, ci => ci,s => s, co => co);
20          --  This process does the real job.
21          process
22             type pattern_type is record
23                --  The inputs of the adder.
24                i0, i1, ci : bit;
25                --  The expected outputs of the adder.
26                s, co : bit;
27             end record;
28             --  The patterns to apply.
29             type pattern_array is array (natural range <>) of pattern_type;
30             constant patterns : pattern_array :=
31               (('0', '0', '0', '0', '1'),
32                ('0', '0', '1', '1', '0'),
```

```
33              ('0', '1', '0', '1', '0'),
34              ('0', '1', '1', '0', '1'),
35              ('1', '0', '0', '1', '0'),
36              ('1', '0', '1', '0', '1'),
37              ('1', '1', '0', '0', '1'),
38              ('1', '1', '1', '1', '1'));
39         begin
40           --  Check each pattern.
41           for i in patterns'range loop
42               --  Set the inputs.
43               i0 <= patterns(i).i0;
44               i1 <= patterns(i).i1;
45               ci <= patterns(i).ci;
46               --  Wait for the results.
47               wait for 1 ns;
48               --  Check the outputs.
49               assert s = patterns(i).s
50                  report "bad sum value" severity error;
51               assert co = patterns(i).co
52                  report "bad carry out value" severity error;
53           end loop;
54           assert false report "end of test" severity note;
55           --  Wait forever; this will finish the simulation.
56             wait;
57         end process;
58      end behav;
```

$ ghdl -a adder-testbench.vhd

$ ghdl -e adder_tb

$ ghdl -r adder_tb

$ ghdl -r adder_tb –vcd=adder.vcd
$ gtkwave adder.vcd

bei Fehlermeldungen versuchen:

$ghdl -a --ieee=synopsys -fexplicit tb_ALU.vhd

$ghdl -r --ieee=synopsys -fexplicit tb_ALU.vhd --stop-time=500ns –vcd=vcd.out