```vhdl
------------------------------------------------ half_adder

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity HA_E is
  port (I1, I2: in std_logic;
    S, Co: out std_logic);
end HA_E;

architecture HA_A of HA_E is
  begin
    S <= I1 xor I2;
    Co <= I1 and I2;
end HA_A;


------------------------------------------------ half_adder test bench

--pragma synthesis_off
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY ha_tb_e IS
END ha_tb_e;

ARCHITECTURE ha_tb_a OF ha_tb_e IS
  COMPONENT ha_e
    PORT (i1, i2 : IN  std_logic;
          s, co  : OUT std_logic);
  END COMPONENT;
  SIGNAL i1_s, i2_s, s_s, co_s : std_logic;
  CONSTANT t  : time := 20 ns;
BEGIN
  dut : ha_e
    PORT MAP (i1_s, i2_s, s_s, co_s);
  i1_s <= '0' AFTER 0*t,
          '1' AFTER 2*t;
  i2_s <= '0' AFTER 0*t,
          '1' AFTER 1*t,
          '0' AFTER 2*t,
          '1' AFTER 3*t;
END ha_tb_a;
```

```
------------------------------------------------ full adder

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY fa_e IS
  PORT (i1, i2, ci : IN  std_logic;
        s, co      : OUT std_logic);
END fa_e;

ARCHITECTURE fa_a OF fa_e IS
component HA_E
  port (I1, I2: in std_logic;
    S, Co: out std_logic);
end component;
    signal CO1_S, S1_S, CO2_S : std_logic;
BEGIN
HA_1 : HA_E
  port map (I1=>I1, I2=>I2, Co=>CO1_S, S=>S1_S);
HA_2 : HA_E
  port map (I1=>S1_S, I2=>Ci, Co=>CO2_S, S=>S);
  Co <= CO1_S or CO2_S;
END fa_a;
```

```
------------------------------------------------ full adder test bench

--pragma synthesis_off
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY fa_tb_e IS
END fa_tb_e;

ARCHITECTURE fa_tb_a OF fa_tb_e IS

component FA_E
  port(I1, I2, Ci : in std_logic;
    S, Co : out std_logic);
end component;

signal i1_s, i2_s, ci_s, s_s, co_s : std_logic;

BEGIN
  dut : fa_e
    PORT MAP (i1=>i1_s, i2=>i2_s, ci=>ci_s, s=>s_s, co=>co_s);
  i1_s <= '0' AFTER 0*t,
          '1' AFTER 4*t;
  i2_s <= '0' AFTER 0*t,
          '1' AFTER 2*t,
          '0' AFTER 4*t,
          '1' AFTER 6*t;
  ci_s <= '0' AFTER 0*t,
          '1' AFTER 1*t,
          '0' AFTER 2*t,
          '1' AFTER 3*t,
          '0' AFTER 4*t,
          '1' AFTER 5*t,
          '0' AFTER 6*t,
          '1' AFTER 7*t;
END fa_tb_a;
```

```vhdl
------------------------------------------------ adder structural

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY add_e IS
  GENERIC (nbits : natural := nbits);
  PORT (i1, i2     : IN  std_logic_vector(nbits-1 DOWNTO 0);
        ci         : IN  std_logic;
        s          : OUT std_logic_vector(nbits-1 DOWNTO 0);
        co         : OUT std_logic);
END add_e;

ARCHITECTURE add_a OF add_e IS
  COMPONENT fa_e
    PORT (i1, i2, ci : IN  std_logic;
          s, co      : OUT std_logic);
  END COMPONENT;
  SIGNAL co_s : std_logic_vector(nbits DOWNTO 0);
BEGIN
  g : FOR i IN nbits-1 DOWNTO 0 GENERATE
    addi : fa_e
      PORT MAP (i1(i), i2(i), co_s(i), s(i), co_s(i+1));
  END GENERATE;
  co_s(0) <= ci;
  co <= co_s(nbits);
END add_a;
```

```
------------------------------------------------- adder structural teast bench

-- 140 ns
-- pragma synthesis_off
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY add_tb_e IS
END add_tb_e;

ARCHITECTURE add_tb_a OF add_tb_e IS
  SIGNAL   ci_int_s, i1_int_s, i2_int_s,
           s_int_s, cos_int_s              : integer;
  SIGNAL   ci_s, co_s                      : std_logic;
  SIGNAL   i1_s, i2_s, s_s                 : std_logic_vector(nbits-1 DOWNTO 0);
  SIGNAL   cos_s                           : std_logic_vector(nbits DOWNTO 0);

  COMPONENT add_e
    PORT (i1, i2     : IN  std_logic_vector(nbits-1 DOWNTO 0);
          ci         : IN  std_logic;
          s          : OUT std_logic_vector(nbits-1 DOWNTO 0);
          co         : OUT std_logic);
  END COMPONENT;

BEGIN
  dut : add_e PORT MAP (i1_s, i2_s, ci_s, s_s, co_s);

  i1_int_s <= 1 AFTER 0*t,
              2**(nbits)-1 AFTER 5*t;
  i2_int_s <= 2            AFTER 0*t,
              3            AFTER 2*t,
              4            AFTER 3*t,
              2**(nbits)-1 AFTER 4*t;
  ci_int_s <= 0 AFTER 0*t,
              1 AFTER 1*t,
              0 after 2*t,
              1 after 4*t;

  s_int_s <= ci_int_s + i1_int_s + i2_int_s;
  i1_s <= conv_vector(i1_int_s, nbits);
  i2_s <= conv_vector(i2_int_s, nbits);
  ci_s <= conv_vector(ci_int_s);
  cos_s <= co_s & s_s;
  cos_int_s <= conv_int(cos_s);

  PROCESS
  BEGIN
    wait on i1_s, i2_s, ci_s;
    wait for 0.75*t;
    ASSERT cos_int_s = s_int_s
      report "*** Fehler ***" severity note;
    ASSERT cos_int_s /= s_int_s
      report "ok" severity note;
  END PROCESS;

END add_tb_a;
```

```
------------------------------------------------ adder algorithmic

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY accu_e IS
  PORT (i           : IN  integer_t;
        s           : OUT integer_t;
        load        : IN  std_logic;
        clk, reset  : IN  std_logic);
END accu_e;

ARCHITECTURE accu_a OF accu_e IS
  SIGNAL i_s, s_s : integer_t := 0;
BEGIN
  i_s <= i + s_s when load /= '1'
           else i;
  s <= s_s;
  reg : PROCESS(clk, reset)
  BEGIN
    if reset = '1' then
      s_s <= 0;
    elsif rising_edge(clk) then
      s_s <= i_s;
    end if;
  END PROCESS;
END accu_a;
```

```vhdl
------------------------------------------------ adder algorithmic test bench
--pragma synthesis_off

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY accu_tb_e IS
END accu_tb_e;

ARCHITECTURE accu_tb_a OF accu_tb_e IS
  COMPONENT accu_e IS
    PORT (i          : IN  integer_t;
          s          : OUT integer_t;
          load       : IN  std_logic;
          clk, reset : IN  std_logic);
  END COMPONENT;
  SIGNAL i_s, s_s        : integer_t :=0;
  Signal load_s          : std_logic := '0';
  SIGNAL clk_s, reset_s : std_logic := '0';
BEGIN
  dut : accu_e
    PORT MAP (i_s, s_s, load_s, clk_s, reset_s);
  PROCESS(clk_s)
  BEGIN
    clk_s <= NOT clk_s AFTER t/2;
  END PROCESS;
  reset_s <= '1' AFTER 0 ns,
             '0' AFTER 1 ns,
             '1' after 20 ns,
             '0' after 21 ns;
  load_s <= '0' AFTER 0 ns,
            '1' AFTER 5 ns,
            '0' AFTER 35 ns;
  i_s <= 0              AFTER 0 ns,
         integer_t'high AFTER 5 ns,
         integer_t'low  AFTER 25 ns,
         0              after 35 ns,
         1              AFTER 55 ns;
  PROCESS
  BEGIN
    WAIT FOR 1 us;
    ASSERT false;
  END PROCESS;
END accu_tb_a;
```

```
------------------------------------------------ ampel

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY ampel_e IS
  PORT (taste       : IN  std_logic;
        auto, fuss  : OUT ampel_t;
        clk, reset  : IN  std_logic);
END ampel_e;

ARCHITECTURE ampel_a OF ampel_e IS

  TYPE state_t IS (gruen, gelb_rot, rot_decr,
                   gelb_gruen, gruen_decr, gruen_gelb);

  SIGNAL state_s, next_s : state_t := gruen;
  SIGNAL i_s, s_s        : integer_t;
  SIGNAL load_s          : std_logic;
  SIGNAL auto_s          : ampel_t;

  COMPONENT accu_e
    PORT (i           : IN  integer_t;
          s           : OUT integer_t;
          load        : IN  std_logic;
          clk, reset  : IN  std_logic);
  END COMPONENT;

BEGIN
  accu : accu_e
    PORT MAP (i_s, s_s, load_s, clk, reset);

  auto <= auto_s;
  fuss <= NOT auto_s;

  comb : PROCESS(taste, state_s, s_s)
  BEGIN

  case state_s is
  when gruen =>
    auto_s <= gruen;
    load_s <= '1';
    i_s <= 0;
    if taste = '1' then
      next_s <= gelb_rot;
    else
      next_s <= gruen;
    end if;

  when gelb_rot =>
    auto_s <= gelb;
    load_s <= '1';
    i_s <= dauer;
    next_s <= rot_decr;
  when rot_decr =>
    auto_s <= rot;
    load_s <= '0';
    i_s <= -1;
    if s_s = 0 then
      next_s <= gelb_gruen;
    else
      next_s <= rot_decr;
    end if;

  when gelb_gruen =>
    auto_s <= gelb;
    load_s <= '1';
    i_s <= dauer;
    next_s <= gruen_decr;

  when gruen_decr =>
    auto_s <= gruen;
    load_s <= '0';
    i_s <= -1;
    if taste = '1' then
        next_s <= gruen_gelb;
    else
      if s_s = 0 then
        next_s <= gruen;
      else
        next_s <= gruen_decr;
      end if;
    end if;

  when gruen_gelb =>
    auto_s <= gruen;
    load_s <= '0';
    i_s <= -1;
    if s_s = 0 then
      next_s <= gelb_rot;
    else
      next_s <= gruen_gelb;
    end if;

  end case;
  END PROCESS;

  reg : PROCESS(clk, reset)
  BEGIN
    IF reset = '1' THEN
      state_s <= gruen;
    ELSIF rising_edge(clk) THEN
      state_s <= next_s;
    END IF;
  END PROCESS;

END ampel_a;
```

```
------------------------------------------------ ampel test bench
--pragma synthesis_off

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.add_pack.ALL;

ENTITY ampel_tb_e IS
END ampel_tb_e;

ARCHITECTURE ampel_tb_a OF ampel_tb_e IS

  COMPONENT ampel_e IS
    PORT (taste       : IN  std_logic;
          auto, fuss : OUT ampel_t;
          clk, reset : IN  std_logic);
  END COMPONENT;

  SIGNAL taste_s         : std_logic;
  SIGNAL auto_s, fuss_s : ampel_t;
  SIGNAL clk_s, reset_s : std_logic := '0';

BEGIN
  dut : ampel_e
    PORT MAP (taste_s, auto_s, fuss_s, clk_s, reset_s);
  PROCESS(clk_s)
  BEGIN
    clk_s <= NOT clk_s AFTER t/2;
    IF rising_edge(clk_s) THEN
      ASSERT auto_s /= fuss_s
        REPORT "*** beide grün oder rot ***"
        SEVERITY error;
      ASSERT NOT(auto_s /= fuss_s)
        REPORT ";-)"
        SEVERITY note;
    END IF;
    END PROCESS;
  reset_s <= '1' AFTER 0 ns,
             '0' AFTER 1 ns;
  taste_s <= '0' AFTER 0*t,
             '1' AFTER 1*t,
             '0' AFTER 2*t,
             '1' AFTER 14*t,
             '0' AFTER 15*t,
             '1' AFTER 16*t,
             '0' AFTER 17*t,
             '1' AFTER 21*t,
             '0' AFTER 22*t,
             '1' AFTER 23*t,
             '0' AFTER 24*t;
  PROCESS
  BEGIN
    WAIT FOR 1 us;
    ASSERT false;
  END PROCESS;
END ampel_tb_a;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

PACKAGE add_pack IS
  TYPE integer_t IS RANGE -8 TO 7;
  TYPE ampel_t IS (rot, gelb, gruen);
  FUNCTION conv_int (arg : std_logic_vector) RETURN integer;
  FUNCTION conv_vector (z, size : integer) RETURN std_logic_vector;
  FUNCTION conv_vector (z : integer) RETURN std_logic;
  FUNCTION "/=" (l, r : ampel_t) RETURN boolean;
  FUNCTION "not" (l : ampel_t ) RETURN ampel_t;
  CONSTANT nbits  : integer;
  CONSTANT dauer  : integer_t;
  CONSTANT t      : time;
END add_pack;

PACKAGE BODY add_pack IS
  CONSTANT nbits         : integer   := 8;
  CONSTANT dauer         : integer_t := 4;
  CONSTANT t             : time      := 20 ns;

  -- functions not for general use (insecure)

 FUNCTION conv_vector (z, size : integer) RETURN std_logic_vector IS
   VARIABLE bv : std_logic_vector (size-1 DOWNTO 0) := (OTHERS => '0');
   VARIABLE hz : integer;
 BEGIN
   hz := z;
   FOR I IN bv'low TO bv'high LOOP
     IF (hz MOD 2) = 0 THEN
       bv(I) := '0';
     ELSE
       bv(I) := '1';
     END IF;
     hz := hz / 2;
   END LOOP;
   RETURN bv;
 END conv_vector;

 FUNCTION conv_vector (z : integer) RETURN std_logic IS
   VARIABLE b : std_logic := '0';
 BEGIN
     IF z = 0 THEN
       b := '0';
     ELSE
       b := '1';
     END IF;
     RETURN b;
 END conv_vector;
```

```
  FUNCTION conv_int (arg : std_logic_vector) RETURN integer IS
    VARIABLE temp : integer := 0;
  BEGIN
    FOR i IN arg'range LOOP
      IF arg(i) = '1' THEN
        temp := temp*2+1;
      ELSE
        temp := temp*2;
      END IF;
    END LOOP;
    RETURN temp;
  END;

  FUNCTION "/=" (l, r : ampel_t) RETURN boolean is
  BEGIN
    if ((l = rot) and (r = rot)) or
       ((l = gruen) and (r = gruen)) then
      return false;
    else
      return true;
    end if;
  END;

  FUNCTION "not" (l : ampel_t ) RETURN ampel_t is
  begin
      if l=rot then return gruen; end if;
      if l=gelb then return gelb; end if;
      if l=gruen then return rot; end if;
  end;
END add_pack;
```