```
-------------------------------
-- cla
-------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity cla is
  generic(
    n_g: integer :=4    -- count of bits
  ); port(
    x_p   : in std_logic_vector(n_g-1 downto 0);
    y_p   : in std_logic_vector(n_g-1 downto 0);
    cin_p : in std_logic;
    cla_p :out std_logic_vector(n_g-1 downto 0);
    cout_p:out std_logic
  );
end cla;

architecture beh of cla is
  signal cg: std_logic_vector(n_g-1 downto 0);
  signal cp: std_logic_vector(n_g-1 downto 0);
begin
  cg <= x_p and y_p;
  cp <= x_p or y_p;

  process (cg,cp)
    variable cla: std_logic_vector(n_g downto 0);
  begin
    cla(0) := cin_p;
    cla(1) := cg(0) or (cp(0) and cin_p);
    for i in 1 to n_g-1 loop
      cla(i+1) := cg(i) or (cp(i) and cla(i));
    end loop;

    cla_p  <= cla(n_g-1 downto 0);
    cout_p <= cla(n_g);
  end process;
end beh;

-------------------------------
-- adder_cla
-------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity adder_cla is
  generic(
    n_g: integer :=4    -- count of bits
  ); port(
    x_p   : in std_logic_vector(n_g-1 downto 0);
    y_p   : in std_logic_vector(n_g-1 downto 0);
    cin_p : in std_logic;
    s_p   :out std_logic_vector(n_g-1 downto 0);
    cout_p:out std_logic
  );
end adder_cla;
```

```
architecture struct of adder_cla is
  signal cla: std_logic_vector(n_g-1 downto 0);
begin
  l1: entity work.cla(beh)
    generic map(n_g)
    port map(x_p,y_p, cin_p,cla,cout_p);

  s_p <= x_p XOR y_p xor cla;
end struct;

-------------------------------
-- adder_cla_t
-------------------------------

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity adder_cla_t is
  constant n:integer :=4;
begin
end entity adder_cla_t;

architecture test of adder_cla_t is
  signal x,y,s: std_logic_vector (n-1 downto 0);
  signal cin,cout: std_logic;

  procedure printStatus is
    variable l:line;
  begin
    write(l, now, right, 10, ns);
    write(l, string'("␣in:␣"));
    write(l, x);
    write(l, string'(","));
    write(l, y);
    write(l, string'(","));
    write(l, cin);
    write(l, string'("␣out:␣"));
    write(l, s);
    write(l, string'(","));
    write(l, cout);
    writeline(output, l);
  end procedure printStatus;

begin
  uut: entity work.adder_cla(struct)
    generic map(n)
    port map (x,y,cin, s,cout);

  test: process
    constant td:time := 200 ns;
  begin
    cin <='1';
    x <= "1010";
    y <= "0110";
    wait for td;
```

```
    printStatus;

    x <= "1011";
    y <= "0110";
    wait for td;
    printStatus;

    wait;
  end process test;

end architecture test;
```

```
------------------------------
-- mul_4
------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity mul_4 is
  port(
   A0 : in  std_logic;
   A1 : in  std_logic;
   B0 : in  std_logic;
   B1 : in  std_logic;
   C0 : out std_logic;
   C1 : out std_logic;
   C2 : out std_logic;
   C3 : out std_logic);
end mul_4;

architecture table of mul_4 is
   signal ab : std_logic_vector (3 downto 0);
   signal c : std_logic_vector (3 downto 0);
begin
   ab <=  A1 & A0 & B1 & B0;

   mul : process (ab) begin
      case ab is
        when "0000" => c <=  "0000";
        when "0001" => c <=  "0000";
        when "0010" => c <=  "0000";
        when "0011" => c <=  "0000";
        when "0100" => c <=  "0000";
        when "0101" => c <=  "0001";
        when "0110" => c <=  "0010";
        when "0111" => c <=  "0011";
        when "1000" => c <=  "0000";
        when "1001" => c <=  "0010";
        when "1010" => c <=  "0100";
        when "1011" => c <=  "0110";
        when "1100" => c <=  "0000";
        when "1101" => c <=  "0011";
        when "1110" => c <=  "0110";
        when "1111" => c <=  "1001";
        when others => c <=  "XXXX";
      end case;
   end process;
   (C3,C2,C1,C0) <= c;
end table;

architecture k of mul_4 is begin
  C0 <= A0 and B0;
  C1 <= (A0 and not A1 and B1) or
        (A0 and not B0 and B1) or
        (not A0 and A1 and B0) or
        (A1 and B0 and not B1);
  C2 <= (A1 and B1 and not B0) or
        (A1 and not A0 and B1);
  C3 <= A1 and A0 and B1 and B0;
```

```
end k;

-------------------------------
-- mul_n
-------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_bit.all;
use ieee.numeric_std.all;

entity mul_n is
  generic(
    n_g: integer :=4    -- count of bits
  ); port(
    a_p: in std_logic_vector(n_g-1 downto 0);
    b_p: in std_logic_vector(n_g-1 downto 0);
    c_p: out std_logic_vector(2*n_g-1 downto 0)
  );
end mul_n;

architecture beh of mul_n is
  signal a,b: integer range 0 to 2**n_g-1;
  signal c:   integer range 0 to 2**(2*n_g)-1;
  signal ctmp:unsigned(0 to 2*n_g-1);
begin
  a <= to_integer(unsigned(a_p));
  b <= to_integer(unsigned(b_p));
  c <= a * b;

  ctmp <= to_unsigned(c, 2*n_g);
  c_p <= std_logic_vector(ctmp);
  --c_p <= std_logic_vector(to_unsigned(c, 2*n_g));
end beh;

-------------------------------
-- mul_n_t
-------------------------------

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity mul_n_t is
  constant n:integer :=4;
begin
end entity mul_n_t;

architecture test of mul_n_t is
  signal x,y: std_logic_vector (n-1 downto 0);
  signal res: std_logic_vector (2*n-1 downto 0);

  procedure printStatus is
    variable l:line;
  begin
    write(l, now, right, 10, ns);
    write(l, string'("␣in:␣"));
```

```
    write(l, x);
    write(l, string'(","));
    write(l, y);
    write(l, string'("␣out:␣"));
    write(l, res);
    writeline(output, l);
  end procedure printStatus;

begin
  uut: entity work.mul_n(beh)
    generic map(n)
    port map (x,y, res);

  test: process
    constant td:time := 200 ns;
  begin
    x <= "1010";
    y <= "0110";
    wait for td;
    printStatus;

    x <= "1011";
    y <= "0110";
    wait for td;
    printStatus;

    wait;
  end process test;

end architecture test;
```

```
------------------------------
-- fib
------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity fib is
  generic(
    n_g: integer :=4    -- count of bits
  ); port(
    reset_p : in std_logic;
    clock_p : in std_logic;
    out_p:   out std_logic_vector(n_g-1 downto 0)
  );
end entity fib;

architecture beh of fib is
    signal fprev : natural;
    signal fcurr : natural;
    signal fnext : natural;
    signal tmp:    unsigned(n_g downto 0);
begin
    fnext <= fcurr + fprev;

    process (clock_p, reset_p, tmp) begin
      if reset_p = '1' or
         tmp(n_g) = '1'
      then
        fprev <= 0;
        fcurr <= 1;
      elsif rising_edge(clock_p) then
        fprev <= fcurr;
        fcurr <= fnext;
      end if;
    end process;

    tmp <= to_unsigned(fprev, n_g+1);
    out_p <= std_logic_vector(tmp(n_g-1 downto 0));
end architecture beh;



------------------------------
-- fib_t
------------------------------

library ieee;
use ieee.std_logic_1164.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity fib_t is
end entity fib_t;

architecture test of fib_t is
  constant td: time := 50 ns;
```

```
  constant n:  integer := 4;

  signal reset: std_logic;
  signal fout:  std_logic_vector(n-1 downto 0);
  signal clk:   std_logic := '0';
  signal done:  std_logic := '0';
begin
  uut: entity work.fib(beh)
    generic map(n)
    port map(reset,clk,fout);

  pclock: process
  begin
    while (done = '0') loop
        wait for td/2;
        clk <= not clk;
    end loop;
    wait;
  end process;

  ptest: process
    file fvector: text;
    variable lin,lout: line;
    variable testvec: std_logic_vector(n downto 0);
  begin
    file_open(fvector,"tfib93.vec",READ_MODE);

    while not endfile(fvector) loop
      readline (fvector, lin);
      read (lin,testvec) ;
      reset <= testvec(n);

      wait until rising_edge(clk);
      wait until falling_edge(clk);

      if (testvec(n-1 downto 0) /= fout) then
        assert false
            report "Test␣vector␣failure!"
            severity note;
      end if;

      write(lout, now, right, 10, ns);
      write(lout, string'("␣expected:␣"));
      write(lout, testvec(n-1 downto 0));
      write(lout, string'("␣obtained:␣"));
      write(lout, fout);
      writeline(output, lout);
    end loop;

    file_close(fvector);
    done <= '1';
    wait;
  end process;

end architecture;
```

10000
00001
00001
00010
00011
00101
01000
01101
00000

```
---------------------
-- mux_no_ff
---------------------

library ieee;
use ieee.std_logic_1164.all;

entity mux_no_ff is
  port (
    a,b,c : in std_logic;
    sel: in std_logic_vector (2 downto 0);
    z:  out std_logic);
end mux_no_ff;

architecture beh of mux_no_ff is
begin
  process (a,b,c,sel) begin
    if sel(0) = '1' then
      z <= a;
    elsif sel(1) = '1' then
      z <= b;
    elsif sel(2) = '1' then
      z <= c;
    else  --- izbjegavanje bistabila u sintezi
      z <= 'X';
    end if;
  end process;
end beh;


-- bistabil se generira za svaku varijablu
-- koja bi se mogla procitati prije pisanja:

-- process(clk)
--   variable tmp: std_logic_vector(0 to 1);
-- begin
--   if (rising_edge(clk)) then
--     if (e_p) then
--       out_p <= tmp;
--     else
--       out_p <= "00";
--     end if;
--     tmp := in_p xor "10";
--   end if;
-- end process;
```

Literatura:

* http://en.wikipedia.org/wiki/VHDL
* http://www.acc-eda.com/vhdlref/
* http://www.vhdl-online.de/~vhdl/TUTORIAL/
* http://www.altera.com/support/examples/vhdl/vhdl.html
* http://tams-www.informatik.uni-hamburg.de/vhdl/models/recursive/
* http://www.cs.ucr.edu/content/esd/labs/tutorial/