```
entity teller is
   port (m, klok, reset: in bit;
         output: out bit_vector(2 downto 0));
end teller;

architecture behavior of teller is
   type state_type is (S0, S1, S2, S3, S4, S5, S6, S7);
   signal current_state, next_state: state_type;
begin
   combin: process(current_state, m)
   begin
      case current_state is
         when S0 =>
            output <= "000";
            next_state <= S1;
         when S1 =>
            output <= "001";
            if (m = '0') then
               next_state <= S2;
            else
               next_state <= S3;
            end if;
         when S2 =>
            output <= "010";
            if (m = '0') then
               next_state <= S3;
            else
               next_state <= S6;
            end if;
         when S3 =>
            output <= "011";
            if (m = '0') then
               next_state <= S4;
            else
               next_state <= S2;
            end if;
         when S4 =>
            output <= "100";
            if (m = '0') then
               next_state <= S5;
            else
               next_state <= S0;
            end if;
         when S5 =>
            output <= "101";
            if (m = '0') then
               next_state <= S6;
            else
               next_state <= S4;
            end if;
         when S6 =>
            output <= "110";
            next_state <= S7;
         when S7 =>
```

```
            output <= "111";
            if (m = '0') then
               next_state <= S0;
            else
               next_state <= S5;
            end if;
      end case;
   end process;

   sync: process(klok, reset)
   begin
      if (klok'event and klok = '1') then
         if (reset = '1') then
            current_state <= S0;
         else
            current_state <= next_state;
         end if;
      end if;
   end process;
end behavior;

entity teller_tb is
end teller_tb;

architecture test of teller_tb is
   component teller
      port (m, klok, reset: in bit;
            output: out bit_vector(2 downto 0));
   end component;
   signal m, klok, reset:bit;
   signal output: bit_vector(2 downto 0);
begin
   lbl1: teller port map (m, klok, reset, output);
   klok <= '1' after 50 ns when klok /= '1' else
           '0' after 50 ns;
   reset <= '1' after 0 ns,
            '0' after 125 ns,
            '1' after 1225 ns,
            '0' after 1325 ns;
   m <= '0' after 0 ns,
        '1' after 1200 ns;
end test;

configuration teller_tb_cfg of teller_tb is
   for test
      for all:teller use entity work.teller(behavior);
      end for;
   end for;
end teller_tb_cfg;
```

```
entity equal_bit is
   port (in1: in  bit;
         in2: in  bit;
         res: out bit);
end equal_bit;

architecture fast of equal_bit is
begin
   res <= not (in1 xor in2) after 2 ns;
end fast;

architecture slow of equal_bit is
begin
   lbl1: process(in1, in2)
   begin
      if (in1 = in2) then
         res <= '1' after 10 ns;
      else
         res <= '0' after 10 ns;
      end if;
   end process;
end slow;


entity and_gate is
   port (ina:    in  bit;
         inb:    in  bit;
         result: out bit);
end and_gate;

architecture fast of and_gate is
begin
   result <= (ina and inb) after 5 ns;
end fast;

architecture slow of and_gate is
begin
   result <= (ina and inb) after 20 ns;
end slow;


entity comparator is
   port (a:  in  bit_vector(1 downto 0);
         b:  in  bit_vector(1 downto 0);
         eq: out bit);
end comparator;

architecture structural of comparator is
   component equal_bit
      port (in1: in  bit;
            in2: in  bit;
            res: out bit);
   end component;
   component and_gate
```

```
      port (ina:    in  bit;
            inb:    in  bit;
            result: out bit);
   end component;
   signal bit0, bit1: bit;
begin
   lbl1: equal_bit port map (a(0), b(0), bit0);
   lbl2: equal_bit port map (a(1), b(1), bit1);
   lbl3: and_gate  port map (bit0, bit1, eq);
end structural;

architecture behavioural of comparator is
begin
   lbl1: process(a, b)
   begin
      if (a = b) then
         eq <= '1' after 1 ns;
      else
         eq <= '0' after 1 ns;
      end if;
   end process;
end behavioural;


entity comparator_tb is
end comparator_tb;

architecture test of comparator_tb is
   component comparator
      port (a:  in  bit_vector(1 downto 0);
            b:  in  bit_vector(1 downto 0);
            eq: out bit);
   end component;
   signal aa, bb: bit_vector(1 downto 0);
   signal eq:     bit;
begin
   lbl1: comparator port map(aa, bb, eq);

   aa <= "00" after   0 ns,
         "01" after  50 ns,
         "10" after 100 ns,
         "11" after 150 ns,
         "00" after 200 ns,
         "01" after 250 ns,
         "10" after 300 ns,
         "11" after 350 ns,
         "00" after 400 ns,
         "01" after 450 ns,
         "10" after 500 ns,
         "11" after 550 ns,
         "00" after 600 ns,
         "01" after 650 ns,
         "10" after 700 ns,
         "11" after 750 ns,
```

```
         "00" after 800 ns;

   bb <= "00" after   0 ns,
         "01" after 200 ns,
         "10" after 400 ns,
         "11" after 600 ns,
         "00" after 800 ns;
end test;

-------------------------------------------------------------------
-- voltooi de volgende configuratie zodanig, dat de 'behavioural'
-- architecture van comparator wordt gebruikt bij simulatie.
-------------------------------------------------------------------
configuration comparator_tb_beh_cfg of comparator_tb is
   for test
      for all:comparator use entity work.comparator(behavioural);
      end for;
   end for;
end comparator_tb_beh_cfg;


-------------------------------------------------------------------
-- voltooi de volgende configuratie zodanig, dat de 'structural'
-- architecture van comparator wordt gebruikt bij simulatie, en binnen
-- deze architecture de 'fast' versies van equal_bit en and_gate.
-------------------------------------------------------------------
configuration comparator_tb_fast_cfg of comparator_tb is
   for test
      for all: comparator use entity work.comparator(structural);
         for structural
            for all:equal_bit use entity work.equal_bit(fast);
            end for;
            for all:and_gate use entity work.and_gate(fast);
            end for;
          end for;
      end for;
   end for;
end comparator_tb_fast_cfg;

-------------------------------------------------------------------
-- voltooi de volgende configuratie zodanig, dat de 'structural'
-- architecture van comparator wordt gebruikt bij simulatie, en binnen
-- deze architecture de 'slow' versies van equal_bit en and_gate.
-------------------------------------------------------------------
-
configuration comparator_tb_slow_cfg of comparator_tb is
   for test
      for all: comparator use entity work.comparator(structural);
         for structural
            for all:equal_bit use entity work.equal_bit(slow);
            end for;
            for all:and_gate use entity work.and_gate(slow);
            end for;
          end for;
```

```
      end for;
   end for;
end comparator_tb_slow_cfg;
```

```
library IEEE;
use IEEE.std_logic_1164.all;

entity counter is
   port (clk: in std_logic;
         rst: in std_logic;
         val: out integer);
end counter;

architecture behaviour of counter is
   signal tmp: integer;
begin
   val <= tmp;
   lbl1: process(clk, rst, tmp)
   begin
      if (clk'event and clk = '1') then
         if (rst = '1') then
            tmp <= 0;
         elsif (tmp = 9) then
            tmp <= 0;
         else
            tmp <= tmp + 1;
         end if;
      end if;
   end process;
end behaviour;

library IEEE;
use IEEE.std_logic_1164.all;
entity segm_dec is
   port (val: in  std_logic_vector(3 downto 0);
         code: out std_logic_vector(6 downto 0));
end segm_dec;

architecture behaviour of segm_dec is
begin
   lbl1: process (val)
   begin
      if    val = "0000" then code <= "0111111";
      elsif val = "0001" then code <= "0000110";
      elsif val = "0010" then code <= "1011011";
      elsif val = "0011" then code <= "1001111";
      elsif val = "0100" then code <= "1100110";
      elsif val = "0101" then code <= "1101101";
      elsif val = "0110" then code <= "1111101";
      elsif val = "0111" then code <= "0000111";
      elsif val = "1000" then code <= "1111111";
      elsif val = "1001" then code <= "1101111";
      end if;
   end process;
end behaviour;

library IEEE;
use IEEE.std_logic_1164.all;
```

```
use IEEE.numeric_std.all;

entity cntr_plus_dec is
   port (clk, res: in std_logic;
         code:     out std_logic_vector(6 downto 0));
end cntr_plus_dec;

architecture structural of cntr_plus_dec is
   component counter
      port (clk: in std_logic;
            rst: in std_logic;
            val: out integer);
   end component;
   component segm_dec
      port (val: in  std_logic_vector(3 downto 0);
            code: out std_logic_vector(6 downto 0));
   end component;
   signal s_bits:std_logic_vector(3 downto 0);
   signal s_int:integer;
begin
   s_bits <= STD_LOGIC_VECTOR(TO_UNSIGNED(s_int,4));
   lbl1: counter port map (clk, res, s_int);
   lbl2: segm_dec port map (s_bits, code);
end structural;

configuration cntr_plus_dec_orig_cfg of cntr_plus_dec is
   for structural
      for all: counter use entity work.counter(behaviour);
      end for;
      for all: segm_dec use entity work.segm_dec(behaviour);
      end for;
   end for;
end cntr_plus_dec_orig_cfg;

library IEEE;
use IEEE.std_logic_1164.all;
entity cntr_plus_dec_tb is
end cntr_plus_dec_tb;

architecture test of cntr_plus_dec_tb is
   component cntr_plus_dec
      port (clk, res: in std_logic;
            code:     out std_logic_vector(6 downto 0));
   end component;
   signal clk, res: std_logic;
   signal code: std_logic_vector(6 downto 0);
begin
   lbl1: cntr_plus_dec port map (clk, res, code);
   clk <= '1' after 500 ns when clk /= '1' else
          '0' after 500 ns;
   res <= '1' after 0 ns,
          '0' after 2 us;
end test;
```

```
configuration cntr_plus_dec_tb_cfg of cntr_plus_dec_tb is
   for test
      for all: cntr_plus_dec use configuration
work.cntr_plus_dec_orig_cfg ;
      end for;
   end for;
end cntr_plus_dec_tb_cfg;
```