```
--------------------------------------------------------------------------------
-- TestBench(ent_top)
--
-- File name : test.vhd
-- Purpose   : Generates stimuli
--           :
-- Library   : IEEE
-- Author(s) : Luca Fanucci
-- Copyrigth : CSMDR-CNR 2002. No part may be reproduced
--           : in any form without the prior written permission by CNR.
--
-- Simulator : Synopsys v. 2000.05, on SUN Ultra 10 with Solaris 2.8
--------------------------------------------------------------------------------
-- Revision List
-- Version      Author  Date            Changes
--
-- 1.0          LFanu   10 May 2002    New version
--------------------------------------------------------------------------------

use STD.textio.all;
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_textio.all;


entity ent_test is

end ent_test;


architecture arch_test of ent_test is

file INGR    : TEXT is in  "/hpusers/fanucci/ESER_2001/TESTFILE/filein.tv";
file USCITA  : TEXT is out "/hpusers/fanucci/ESER_2001/TESTFILE/fileout.tv";

component ent_top
   port(X      : in std_logic;
        CLK    : in std_logic;
        RESET  : in std_logic;
        ENLOAD : in std_logic;
        LOAD   : in std_logic_vector (7 downto 0);
        COUNT  : out std_logic_vector (7 downto 0));
end component;

-- I N P U T     S I G N A L S

signal SEQ, CLOCK, RST, EN_LOAD : std_logic;
signal LOADIN : std_logic_vector(7 downto 0);

-- O U T P U T     S I G N A L S

signal  COUNT_OUT : std_logic_vector(7 downto 0);

begin

   I0: ent_top
      port map(X      => SEQ,
               CLK    => CLOCK,
               RESET  => RST,
               ENLOAD => EN_LOAD,
               LOAD   => LOADIN,
               COUNT  => COUNT_OUT);

-- Generates clk

   clk: process
      VARIABLE clk_cycle: INTEGER:= 0;
      begin
```

```
         loop_clk : loop
            CLOCK <= '0';
            wait for 10 ns;
            CLOCK <= '1';
            wait for 10 ns;
            clk_cycle:= clk_cycle + 1;
         end loop;
      end process;

   stimulus_in : process
      variable BUF : line;
      variable A, B, C : std_logic;
      variable D : std_logic_vector(7 downto 0);
      begin
         while not ENDFILE(INGR) loop
            READLINE(INGR,BUF);
            READ(BUF,A);
            READ(BUF,B);
          READ(BUF,C);
          READ(BUF,D);
            EN_LOAD <= A;
            RST <= B;
          SEQ <= C;
            LOADIN <= D;
            wait for 20 ns;
            end loop;
            wait;
      end process;

   stimulus_out : process
      variable BUF : line;
      begin
         loop
            wait on CLOCK;
            WRITE(BUF,now,right,10);
            WRITE(BUF,SEQ,right,10);
            WRITE(BUF,COUNT_OUT,right,10);
            WRITEline(USCITA,BUF);
         end loop;
      end process;

end arch_test;
```

```
library IEEE,CSX_HRDLIB;

use IEEE.std_logic_1164.all;
use CSX_HRDLIB.Vcomponents.all;

entity ent_sequen is

   port( X, CLK, RESET : in std_ulogic;  Z : out std_ulogic);

end ent_sequen;

architecture SYN of ent_sequen is

signal CURRENT_STATE : std_ulogic_vector(2 downto 0);

signal NEXT_STATE : std_ulogic_vector(0 downto 0);

signal n56, n57, n58, n59, n60, n61, n62, n63, n64 : std_ulogic;

begin

   U29 : NO2 port map( A => n56, B => CURRENT_STATE(0), Q => n64);
   U30 : NA2 port map( A => n62, B => CURRENT_STATE(1), Q => n56);
   U31 : AND32 port map( A => n59, B => n60, C => CURRENT_STATE(2), Q => Z);
   U32 : IN1 port map( A => X, Q => n61);
   U33 : AND32 port map( A => n60, B => n62, C => CURRENT_STATE(0), Q => n63);
   CURRENT_STATE_reg_0_label : DFA2 port map( C => CLK, D => NEXT_STATE(0), Q
                           => CURRENT_STATE(0), QN => n59, RN => RESET);
   CURRENT_STATE_reg_1_label : DFSA2 port map( C => CLK, D => n64, Q =>
                           CURRENT_STATE(1), QN => n60, RN => RESET, SD => n63,
                           SE => X);
   CURRENT_STATE_reg_2_label : DFSA2 port map( C => CLK, D => n57, Q =>
                           CURRENT_STATE(2), QN => n62, RN => RESET, SD => n58,
                           SE => n56);
   U34 : NO2 port map( A => X, B => n59, Q => n57);
   U35 : LOGIC0 port map( Q => n58);
   U36 : ON32 port map( A => n61, B => CURRENT_STATE(1), C => CURRENT_STATE(0),
                           D => n57, E => n56, Q => NEXT_STATE(0));

end SYN;

library IEEE,CSX_HRDLIB;

use IEEE.std_logic_1164.all;
use CSX_HRDLIB.Vcomponents.all;

entity ent_counter_DW01_inc_8_1 is

   port( A : in std_ulogic_vector (0 to 7);  SUM : out std_ulogic_vector (0 to
         7));

end ent_counter_DW01_inc_8_1;

architecture SYN of ent_counter_DW01_inc_8_1 is

signal carry : std_ulogic_vector(7 downto 2);

begin

   U1_1_1 : HA12 port map( A => A(6), B => A(7), CO => carry(2), S => SUM(6));
   U1_1_3 : HA12 port map( A => A(4), B => carry(3), CO => carry(4), S =>
                           SUM(4));
   U1_1_2 : HA12 port map( A => A(5), B => carry(2), CO => carry(3), S =>
                           SUM(5));
   U1_1_5 : HA12 port map( A => A(2), B => carry(5), CO => carry(6), S =>
                           SUM(2));
   U1_1_4 : HA12 port map( A => A(3), B => carry(4), CO => carry(5), S =>
                           SUM(3));
```

```
   U5 : EO1 port map( A => A(0), B => carry(7), Q => SUM(0));
   U1_1_6 : HA12 port map( A => A(1), B => carry(6), CO => carry(7), S =>
                           SUM(1));
   U6 : IN1 port map( A => A(7), Q => SUM(7));

end SYN;

library IEEE,CSX_HRDLIB;

use IEEE.std_logic_1164.all;
use CSX_HRDLIB.Vcomponents.all;

entity ent_counter is

   port( CLK, RESET : in std_ulogic;  LOAD : in std_ulogic_vector (0 to 7);
         ENLOAD, ENCOUNT : in std_ulogic;  COUNT : out std_ulogic_vector (0 to
         7));

end ent_counter;

architecture SYN of ent_counter is

   component ent_counter_DW01_inc_8_1
      port( A : in std_ulogic_vector (0 to 7);  SUM : out std_ulogic_vector (0
            to 7));
   end component;

   signal COUNT_TMP, COUNT_TMP119, sum165 : std_ulogic_vector(7 downto 0);

   signal n395, n396, n397, n398, n399, n400, n401, n402, n403, n404, n405,
      n406, n407, n408, n409, n410, n411, n412, n413, n414, n415, n416, n417,
      n418, n419, n420, n421, n422, n423, n424, n425, n426, n427, n428, n429,
      n430, n431, n432, n433, n434, n435, n436, n437 : std_ulogic;

begin
   COUNT <= ( COUNT_TMP(7), COUNT_TMP(6), COUNT_TMP(5), COUNT_TMP(4),
      COUNT_TMP(3), COUNT_TMP(2), COUNT_TMP(1), COUNT_TMP(0) );

   U116 : AND22 port map( A => ENLOAD, B => LOAD(0), Q => n397);
   U117 : IN1 port map( A => ENLOAD, Q => n428);
   U118 : IN1 port map( A => ENCOUNT, Q => n429);
   U119 : OR32 port map( A => n399, B => n400, C => n397, Q => COUNT_TMP119(7))
                          ;
   U120 : NO2 port map( A => ENLOAD, B => ENCOUNT, Q => n395);
   U121 : NO2 port map( A => ENLOAD, B => n429, Q => n396);
   U122 : NO2 port map( A => LOAD(0), B => n428, Q => n398);
   U123 : AND2 port map( A => n396, B => sum165(7), Q => n399);
   U124 : AND2 port map( A => n395, B => COUNT_TMP(7), Q => n400);
   U125 : MU2 port map( I0 => LOAD(2), I1 => LOAD(2), Q => n401, S => LOAD(1));
   U126 : MU2 port map( I0 => LOAD(3), I1 => LOAD(3), Q => n402, S => LOAD(2));
   U127 : MU2 port map( I0 => n402, I1 => n402, Q => n403, S => LOAD(1));
   U128 : MU2 port map( I0 => LOAD(4), I1 => LOAD(4), Q => n404, S => LOAD(3));
   U129 : MU2 port map( I0 => n404, I1 => n404, Q => n405, S => LOAD(2));
   U130 : MU2 port map( I0 => n405, I1 => n405, Q => n406, S => LOAD(1));
   U131 : MU2 port map( I0 => LOAD(5), I1 => LOAD(5), Q => n407, S => LOAD(4));
   U132 : MU2 port map( I0 => n407, I1 => n407, Q => n408, S => LOAD(3));
   U133 : MU2 port map( I0 => n408, I1 => n408, Q => n409, S => LOAD(2));
   U134 : MU2 port map( I0 => n409, I1 => n409, Q => n410, S => LOAD(1));
   U135 : MU2 port map( I0 => LOAD(6), I1 => LOAD(6), Q => n411, S => LOAD(5));
   U136 : MU2 port map( I0 => n411, I1 => n411, Q => n412, S => LOAD(4));
   U137 : MU2 port map( I0 => n412, I1 => n412, Q => n413, S => LOAD(3));
   U138 : MU2 port map( I0 => n413, I1 => n413, Q => n414, S => LOAD(2));
   U139 : MU2 port map( I0 => n414, I1 => n414, Q => n415, S => LOAD(1));
   U140 : MU2 port map( I0 => LOAD(7), I1 => LOAD(7), Q => n416, S => LOAD(5));
   U141 : MU2 port map( I0 => n416, I1 => n416, Q => n417, S => LOAD(4));
   U142 : MU2 port map( I0 => n417, I1 => n417, Q => n418, S => LOAD(3));
   U143 : MU2 port map( I0 => n418, I1 => n418, Q => n419, S => LOAD(2));
   U144 : MU2 port map( I0 => n419, I1 => n419, Q => n420, S => LOAD(1));
   add_52 : ent_counter_DW01_inc_8_1 port map( A(0) => COUNT_TMP(7), A(1) =>
```

```
                           COUNT_TMP(6), A(2) => COUNT_TMP(5), A(3) =>
                           COUNT_TMP(4), A(4) => COUNT_TMP(3), A(5) =>
                           COUNT_TMP(2), A(6) => COUNT_TMP(1), A(7) =>
                           COUNT_TMP(0), SUM(0) => sum165(7), SUM(1) =>
                           sum165(6), SUM(2) => sum165(5), SUM(3) => sum165(4),
                           SUM(4) => sum165(3), SUM(5) => sum165(2), SUM(6) =>
                           sum165(1), SUM(7) => sum165(0));
   COUNT_TMP_reg_0_label : DFA4 port map( C => CLK, D => COUNT_TMP119(0), Q =>
                           COUNT_TMP(0), QN => n430, RN => RESET);
   COUNT_TMP_reg_1_label : DFA4 port map( C => CLK, D => COUNT_TMP119(1), Q =>
                           COUNT_TMP(1), QN => n431, RN => RESET);
   COUNT_TMP_reg_2_label : DFA4 port map( C => CLK, D => COUNT_TMP119(2), Q =>
                           COUNT_TMP(2), QN => n432, RN => RESET);
   COUNT_TMP_reg_3_label : DFA4 port map( C => CLK, D => COUNT_TMP119(3), Q =>
                           COUNT_TMP(3), QN => n433, RN => RESET);
   COUNT_TMP_reg_4_label : DFA4 port map( C => CLK, D => COUNT_TMP119(4), Q =>
                           COUNT_TMP(4), QN => n434, RN => RESET);
   COUNT_TMP_reg_5_label : DFA4 port map( C => CLK, D => COUNT_TMP119(5), Q =>
                           COUNT_TMP(5), QN => n435, RN => RESET);
   COUNT_TMP_reg_6_label : DFA4 port map( C => CLK, D => COUNT_TMP119(6), Q =>
                           COUNT_TMP(6), QN => n436, RN => RESET);
   COUNT_TMP_reg_7_label : DFA4 port map( C => CLK, D => COUNT_TMP119(7), Q =>
                           COUNT_TMP(7), QN => n437, RN => RESET);
   U145 : AO221 port map( A => n398, B => LOAD(1), C => LOAD(1), D => n397, E
                           => n421, Q => COUNT_TMP119(6));
   U146 : AO221 port map( A => n398, B => n401, C => n401, D => n397, E => n422
                           , Q => COUNT_TMP119(5));
   U147 : AO221 port map( A => n398, B => n403, C => n403, D => n397, E => n423
                           , Q => COUNT_TMP119(4));
   U148 : AO221 port map( A => n398, B => n406, C => n406, D => n397, E => n424
                           , Q => COUNT_TMP119(3));
   U149 : AO221 port map( A => n398, B => n410, C => n410, D => n397, E => n425
                           , Q => COUNT_TMP119(2));
   U150 : AO221 port map( A => n398, B => n415, C => n415, D => n397, E => n426
                           , Q => COUNT_TMP119(1));
   U151 : AO221 port map( A => n398, B => n420, C => n420, D => n397, E => n427
                           , Q => COUNT_TMP119(0));
   U152 : AO22 port map( A => COUNT_TMP(6), B => n395, C => sum165(6), D =>
                           n396, Q => n421);
   U153 : AO22 port map( A => COUNT_TMP(5), B => n395, C => sum165(5), D =>
                           n396, Q => n422);
   U154 : AO22 port map( A => COUNT_TMP(4), B => n395, C => sum165(4), D =>
                           n396, Q => n423);
   U155 : AO22 port map( A => COUNT_TMP(3), B => n395, C => sum165(3), D =>
                           n396, Q => n424);
   U156 : AO22 port map( A => COUNT_TMP(2), B => n395, C => sum165(2), D =>
                           n396, Q => n425);
   U157 : AO22 port map( A => COUNT_TMP(1), B => n395, C => sum165(1), D =>
                           n396, Q => n426);
   U158 : AO22 port map( A => COUNT_TMP(0), B => n395, C => sum165(0), D =>
                           n396, Q => n427);

end SYN;

library IEEE,CSX_HRDLIB;

use IEEE.std_logic_1164.all;
use CSX_HRDLIB.Vcomponents.all;

entity ent_top is

   port( X, CLK, RESET, ENLOAD : in std_logic;  LOAD : in std_logic_vector (7
         downto 0);  COUNT : out std_logic_vector (7 downto 0));

end ent_top;

architecture SYN of ent_top is

   component ent_sequen
```

```
      port( X, CLK, RESET : in std_ulogic;  Z : out std_ulogic);
   end component;

   component ent_counter
      port( CLK, RESET : in std_ulogic;  LOAD : in std_ulogic_vector (0 to 7);
            ENLOAD, ENCOUNT : in std_ulogic;  COUNT : out std_ulogic_vector (0
            to 7));
   end component;

   signal Y : std_ulogic;

begin

   Usequen : ent_sequen port map( X => X, CLK => CLK, RESET => RESET, Z => Y);
   Ucount : ent_counter port map( CLK => CLK, RESET => RESET, LOAD(0) =>
                           LOAD(7), LOAD(1) => LOAD(6), LOAD(2) => LOAD(5),
                           LOAD(3) => LOAD(4), LOAD(4) => LOAD(3), LOAD(5) =>
                           LOAD(2), LOAD(6) => LOAD(1), LOAD(7) => LOAD(0),
                           ENLOAD => ENLOAD, ENCOUNT => Y, COUNT(0) => COUNT(7)
                           , COUNT(1) => COUNT(6), COUNT(2) => COUNT(5),
                           COUNT(3) => COUNT(4), COUNT(4) => COUNT(3), COUNT(5)
                           => COUNT(2), COUNT(6) => COUNT(1), COUNT(7) =>
                           COUNT(0));

end SYN;
```

```
--------------------------------------------------------------------------------
-- (Behavioral)
--
-- File name : ent_top.vhd
-- Purpose   :
--           :
-- Library   : IEEE
-- Author(s) : Luca Fanucci
-- Copyrigth : CSMDR-CNR 2002. No part may be reproduced
--           : in any form without the prior written permission by CNR.
--
-- Simulator : Synopsys v. 2000.05, on SUN Ultra 10 with Solaris 2.8
--------------------------------------------------------------------------------
-- Revision List
-- Version      Author  Date            Changes
--
-- 1.0          LFanu   10 May 2002    New version
--------------------------------------------------------------------------------

library ieee,work;
use ieee.std_logic_1164.all;


entity ent_top  is
   port(X      : in std_logic;
        CLK    : in std_logic;
        RESET  : in std_logic;
        ENLOAD : in std_logic;
        LOAD   : in std_logic_vector (7 downto 0);
        COUNT  : out std_logic_vector (7 downto 0));
end ent_top;


architecture arch_top of ent_top is

component ent_sequen
   port(X     : in std_logic;
         CLK   : in std_logic;
         RESET : in std_logic;
         Z     : out std_logic);
end component;

component ent_counter
   port(CLK     : in std_logic;
        RESET   : in std_logic;
        LOAD    : in std_logic_vector(7 downto 0);
        ENLOAD  : in std_logic;
        ENCOUNT : in std_logic;
        COUNT   : out std_logic_vector(7 downto 0));
end component;

signal Y : std_logic;

begin

   Usequen: ent_sequen
      port map(X     => X,
               CLK   => CLK,
               RESET => RESET,
               Z     => Y);

   Ucount: ent_counter
      port map(CLK     => CLK,
               RESET   => RESET,
               LOAD    => LOAD,
               ENLOAD  => ENLOAD,
               ENCOUNT => Y,
               COUNT   => COUNT);

end arch_top;
```

```
--------------------------------------------------------------------------------
--
--
-- File name : funct.vhd
-- Purpose   :
--           :
-- Library   : IEEE
-- Author(s) : Luca Fanucci
-- Copyrigth : CSMDR-CNR 2002. No part may be reproduced
--           : in any form without the prior written permission by CNR.
--
-- Simulator : Synopsys v. 2000.05, on SUN Ultra 10 with Solaris 2.8
--------------------------------------------------------------------------------
-- Revision List
-- Version      Author  Date            Changes
--
-- 1.0          LFanu   10 May 2002    New version
--------------------------------------------------------------------------------

library ieee, work;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all;


package UNSIGNED_CONVERSION is

   function vect2int (ARG: std_logic_vector; SIZE: natural) return natural;

end UNSIGNED_CONVERSION;


package body UNSIGNED_CONVERSION is

   function vect2int (ARG: std_logic_vector; SIZE: natural) return natural is

   variable I: natural range 0 to (SIZE-1);
   variable A: natural range 0 to (2**(SIZE)-1);

      begin
         A := 0;
         for I in 0 to (SIZE-1) loop
           if (ARG(I) = '1') then
              A := (2**I)+A;
           else
              A := A;
           end if;
         end loop;
         return A;
      end vect2int;

end UNSIGNED_CONVERSION;
```

```
--------------------------------------------------------------------------------
-- (Behavioral)
--
-- File name : counter.vhd
-- Purpose   :
--           :
-- Library   : IEEE
-- Author(s) : Luca Fanucci
-- Copyrigth : CSMDR-CNR 2002. No part may be reproduced
--           : in any form without the prior written permission by CNR.
--
-- Simulator : Synopsys v. 2000.05, on SUN Ultra 10 with Solaris 2.8
--------------------------------------------------------------------------------
-- Revision List
-- Version      Author  Date            Changes
--
-- 1.0          LFanu   10 May 2002    New version
--------------------------------------------------------------------------------

library ieee,work;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.UNSIGNED_CONVERSION.all;

entity ent_counter is
   port(CLK     : in std_logic;
        RESET   : in std_logic;
        LOAD    : in std_logic_vector(7 downto 0);
        ENLOAD  : in std_logic;
        ENCOUNT : in std_logic;
        COUNT   : out std_logic_vector(7 downto 0));
end ent_counter;


architecture arch_counter of ent_counter is

signal COUNT_TMP : natural range 0 to 255;


begin

   counter : process(CLK, RESET, ENCOUNT, ENLOAD)
      begin
         if (RESET = '0') then
            COUNT_TMP <= 0;
         elsif CLK'event and CLK='1' then
            if (ENLOAD = '1') then
               COUNT_TMP <= vect2int(LOAD, 8);
            elsif (ENCOUNT = '0') then
               COUNT_TMP <= COUNT_TMP;
               else
                  COUNT_TMP <= COUNT_TMP+1;
               end if;
            end if;
   end process;

COUNT <= CONV_STD_LOGIC_VECTOR(COUNT_TMP, 8);

end arch_counter;
```

```
--------------------------------------------------------------------------------
-- (Behavioral)
--
-- File name : sequen.vhd
-- Purpose   :
--           :
-- Library   : IEEE
-- Author(s) : Luca Fanucci
-- Copyrigth : CSMDR-CNR 2002. No part may be reproduced
--           : in any form without the prior written permission by CNR.
--
-- Simulator : Synopsys v. 2000.05, on SUN Ultra 10 with Solaris 2.8
--------------------------------------------------------------------------------
-- Revision List
-- Version      Author  Date            Changes
--
-- 1.0          LFanu   10 May 2002    New version
--------------------------------------------------------------------------------

library ieee,work;
use ieee.std_logic_1164.all;


entity ent_sequen  is
   port(X     : in std_logic;
        CLK   : in std_logic;
        RESET : in std_logic;
        Z     : out std_logic);
end ent_sequen;


architecture arch_sequen of ent_sequen 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, X)
   begin
      case CURRENT_STATE is
         when S0 =>
            Z <= '0';
            if (X = '0') then
               NEXT_STATE <= S0;
            else
               NEXT_STATE <= S1;
            end if;
         when S1 =>
            Z <= '0';
            if (X = '0') then
               NEXT_STATE <= S0;
            else
               NEXT_STATE <= S2;
            end if;
         when S2 =>
            Z <= '0';
            if (X = '0') then
               NEXT_STATE <= S3;
            else
               NEXT_STATE <= S1;
            end if;
         when S3 =>
            Z <= '0';
            if (X = '0') then
               NEXT_STATE <= S4;
            else
               NEXT_STATE <= S1;
            end if;
```

```
         when S4 =>
            Z <= '1';
            if (X = '0') then
               NEXT_STATE <= S0;
            else
               NEXT_STATE <= S1;
            end if;
         when S5 =>
            Z <= '0';
            NEXT_STATE <= S0;
         when S6 =>
            Z <= '0';
            NEXT_STATE <= S0;
         when S7 =>
            Z <= '0';
            NEXT_STATE <= S0;
      end case;
   end process;

   sincr: process(CLK, RESET)
      begin
         if (RESET = '0') then  CURRENT_STATE <= S0;
          elsif  (CLK'EVENT and CLK = '1') then
             CURRENT_STATE <= NEXT_STATE;
          end if;
   end process;

end arch_sequen;
```