```
library IEEE;
use IEEE.std_logic_1164.all;
use Work.ABRO_data_pkg.all;
use Work.ABRO_data_type_pkg.all;
package ABRO_data_sim_pkg is
   function check_data( check_data_0: in string) return boolean;
  function text_to_data( text_to_data_0: in string) return data ;
  function data_to_text( signal data_to_text_0: in data) return string;
  function "="( eq_data_0 : in data; eq_data_1 : in data) return boolean;
  function "/="( neq_data_0 : in data; neq_data_1 : in data) return boolean;
  function "<"( less_data_0 : in data; less_data_1 : in data) return boolean;
  function "<="( lesseq_data_0 : in data; lesseq_data_1 : in data) return boolean;
end ABRO_data_sim_pkg;
use Work.ABRO_data_pkg.all;
use Work.ABRO_data_type_pkg.all;
use Work.ABRO_data_sim_pkg.all;
use std.textio.all;
library IEEE;
use IEEE.std_logic_1164.all;


entity ABRO_abro_tb is
   constant Delay : Time := 10 ns ;
   constant AssertionFlag : boolean := true;
end ABRO_abro_tb;

architecture ABRO_abro_MixedView of ABRO_abro_tb is
   subtype esterelString is STRING ( 1 to 10);

   component ABRO
      port (
         comp_clk : in STD_LOGIC;
         comp_rst : in STD_LOGIC;
         comp_A : in STD_LOGIC;
         comp_Adata : in esterelString;
         comp_B : in STD_LOGIC;
         comp_Bdata : in data;
         comp_R : in STD_LOGIC;
         comp_O : out STD_LOGIC;
         comp_Odata : out data

      );
   end component;

   for DUT:ABRO use entity Work.ABRO(ABRO_RTL)
   port map(
     clk => comp_clk,
     rst => comp_rst,
     A => comp_A,
     Adata => comp_Adata,
     B => comp_B,
     Bdata => comp_Bdata,
     R => comp_R,
     O => comp_O,
     Odata => comp_Odata

   );

  signal stop : boolean := FALSE;
```

```
  signal sig_clk : STD_LOGIC := '0';
  signal sig_rst : STD_LOGIC := '0';
  signal sig_A : STD_LOGIC := '0';
  signal sig_Adata : esterelString := (NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL);
  signal sig_B : STD_LOGIC := '0';
  signal sig_Bdata : data := data_InitialValue;
  signal sig_R : STD_LOGIC := '0';
  signal sig_O : STD_LOGIC := '0';
  signal sig_Odata : data := data_InitialValue;

  file temporaryOutputFile : text is out "record_abro.eso";

   signal Assertion : severity_level;

   procedure write_formatted_string( L : inout line; value : in String) is
      variable index : positive := 1;
   begin
      loop
         exit when (index = value’length + 1) or (value(index) = character’(NUL)
);
         write(L, character’(value(index)));
         index := index + 1;
      end loop;
   end write_formatted_string;

   function "="( eq_data_0 : in data; eq_data_1 : in string) return boolean  is
   begin
      return eq_data_0 = text_to_data(eq_data_1);
   end;
   function "="( eq_data_0 : in string; eq_data_1 : in data) return boolean  is
   begin
      return eq_data_1 = text_to_data(eq_data_0);
   end;
   function "/="( neq_data_0 : in data; neq_data_1 : in string) return boolean  is
   begin
      return neq_data_0 /= text_to_data(neq_data_1);
   end;
   function "/="( neq_data_0 : in string; neq_data_1 : in data) return boolean  is
   begin
      return neq_data_1 = text_to_data(neq_data_0);
   end;
   function "<"( less_data_0 : in data; less_data_1 : in string) return boolean  is
   begin
      return less_data_0 < text_to_data(less_data_1);
   end;
   function "<"( less_data_0 : in string; less_data_1 : in data) return boolean  is
   begin
      return text_to_data(less_data_0) < less_data_1;
   end;
   function "<="( lesseq_data_0 : in data; lesseq_data_1 : in string) return boolean  is
   begin
      return lesseq_data_0 <= text_to_data(lesseq_data_1);
   end;
   function "<="( lesseq_data_0 : in string; lesseq_data_1 : in data) return boolean  is
   begin
      return text_to_data(lesseq_data_0) <= lesseq_data_1;
   end;
   function ">"( great_data_0 : in data; great_data_1 : in string) return boolean  is
```

```
   begin
      return text_to_data(great_data_1) < great_data_0;
   end;
   function ">"( great_data_0 : in string; great_data_1 : in data) return boolean  is
   begin
      return great_data_1 < text_to_data(great_data_0);
   end;
   function ">"( great_data_0 : in data; great_data_1 : in data) return boolean  is
   begin
      return great_data_1 < great_data_0;
   end;
   function ">="( great_data_0 : in data; great_data_1 : in string) return boolean  is
   begin
      return text_to_data(great_data_1) <= great_data_0;
   end;
   function ">="( great_data_0 : in string; great_data_1 : in data) return boolean  is
   begin
      return great_data_1 <= text_to_data(great_data_0);
   end;
   function ">="( greateq_data_0 : in data; greateq_data_1 : in data) return boolean  is
   begin
      return greateq_data_1 <= greateq_data_0;
   end;

   procedure WriteLogOutputs( temporaryOutputFile: out text; SIGNAL sig_O: in ST
D_LOGIC; SIGNAL sig_Odata: in data) is
      variable temporaryLine: line;
   begin
      if sig_O = ’1’ then
         write( temporaryLine, STRING’("% Output: O = """));
         write_formatted_string( temporaryLine, esterelString’(data_to_text(sig_
Odata)));
         write( temporaryLine, STRING’(""""));
         writeline( temporaryOutputFile, temporaryLine);
      end if;
   end WriteLogOutputs;




begin
DUT: ABRO port map (
   comp_clk => sig_clk,
   comp_rst => sig_rst,
   comp_A => sig_A,
   comp_Adata => sig_Adata,
   comp_B => sig_B,
   comp_Bdata => sig_Bdata,
   comp_R => sig_R,
   comp_O => sig_O,
   comp_Odata => sig_Odata

   );

CLOCK:
   sig_clk <= not sig_clk after Delay / 2 when not stop else ’0’ after Delay / 2
;

SCENARIO: process
```

```
      variable temporaryLine: line;
   begin
      -- -----
      write( temporaryLine, STRING’("% "));
      write( temporaryLine, STRING’("-----"));
      writeline( temporaryOutputFile, temporaryLine);
      -- File record_abro.eso generated from Esterel module ABRO.
      write( temporaryLine, STRING’("% "));
      write( temporaryLine, STRING’("File record_abro.eso generated from Esterel module ABRO.")
);
      writeline( temporaryOutputFile, temporaryLine);
      -- -----
      write( temporaryLine, STRING’("% "));
      write( temporaryLine, STRING’("-----"));
      writeline( temporaryOutputFile, temporaryLine);

-- RST --
      sig_rst <= ’1’;
      wait until sig_clk’event and sig_clk = ’1’;
      write( temporaryLine, STRING’("! reset;"));
      writeline( temporaryOutputFile, temporaryLine);

      sig_A <= ’1’;
      sig_Adata <= esterelString’("1" & (NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL));
      write( temporaryLine, STRING’("A = ""1"""));
      writeline( temporaryOutputFile, temporaryLine);
      sig_rst <= ’0’;
      sig_B <= ’0’;
      sig_R <= ’0’;
      -- Sync on CLK rising edge --
      WAIT UNTIL sig_clk’EVENT AND sig_clk = ’1’;
      write( temporaryLine, STRING’("% Cycle 1"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Log outputs to file --
      WriteLogOutputs( temporaryOutputFile, sig_O, sig_Odata);
      write( temporaryLine, STRING’(";"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Outputs checking --


-----------------------------------
-- CLK cycle number: 2 --
-----------------------------------
      -- Inputs initialization --
      sig_B <= ’1’;
      if not check_data( STRING’("premier tirage")) then
         stop <= true;
         write( temporaryLine, STRING’("bad user type value checked, in file abro.esi, line 2"));
         writeline( temporaryOutputFile, temporaryLine);
         Assertion <= FAILURE;
      else
         sig_Bdata <= text_to_data( STRING’("premier tirage"));
      end if;
      write( temporaryLine, STRING’("B = ""premier tirage"""));
      writeline( temporaryOutputFile, temporaryLine);
      sig_rst <= ’0’;
      sig_A <= ’0’;
      sig_R <= ’0’;
      -- Sync on CLK rising edge --
      WAIT UNTIL sig_clk'EVENT AND sig_clk = '1';
      write( temporaryLine, STRING'("% Cycle 2"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Log outputs to file --
      WriteLogOutputs( temporaryOutputFile, sig_O, sig_Odata);
      write( temporaryLine, STRING'(";"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Outputs checking --


-----------------------------------
-- CLK cycle number: 3 --
-----------------------------------
      -- Inputs initialization --
      sig_A <= '1';
      sig_Adata <= esterelString'("2" & (NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL,NUL));
      write( temporaryLine, STRING'("A = ""2"""));
      writeline( temporaryOutputFile, temporaryLine);
      sig_rst <= '0';
      sig_B <= '0';
      sig_R <= '0';
      -- Sync on CLK rising edge --
      WAIT UNTIL sig_clk'EVENT AND sig_clk = '1';
      write( temporaryLine, STRING'("% Cycle 3"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Log outputs to file --
      WriteLogOutputs( temporaryOutputFile, sig_O, sig_Odata);
      write( temporaryLine, STRING'(";"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Outputs checking --


-----------------------------------
-- CLK cycle number: 4 --
-----------------------------------
      -- Inputs initialization --
      sig_B <= '1';
      if not check_data( STRING'("second tirage")) then
         stop <= true;
         write( temporaryLine, STRING'("bad user type value checked, in file abro.esi, line 8"));
         writeline( temporaryOutputFile, temporaryLine);
         Assertion <= FAILURE;
      else
         sig_Bdata <= text_to_data( STRING'("second tirage"));
      end if;
      write( temporaryLine, STRING'("B = ""second tirage"""));
      writeline( temporaryOutputFile, temporaryLine);
      sig_rst <= '0';
      sig_A <= '0';
      sig_R <= '0';
      -- Sync on CLK rising edge --
      WAIT UNTIL sig_clk'EVENT AND sig_clk = '1';
      write( temporaryLine, STRING'("% Cycle 4"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Log outputs to file --
      WriteLogOutputs( temporaryOutputFile, sig_O, sig_Odata);
      write( temporaryLine, STRING'(";"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Outputs checking --
```

```


-----------------------------------
-- CLK cycle number: 5 --
-----------------------------------
      -- Inputs initialization --
      if AssertionFlag then
         if not(((sig_Adata = esterelString'("1" & (NUL,NUL,NUL,NUL,NUL,NUL,NUL,
NUL,NUL))) and sig_B = '1')) then
            Assertion <= NOTE;
            write(temporaryLine, STRING'("% %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%"));
            writeline( temporaryOutputFile, temporaryLine);
            write(temporaryLine, STRING'("% NOTE: Break point reached"
                      & LF
                      & "%   file 'abro.esi'"
                      & LF
                      & "%   line '8'"));
            writeline( temporaryOutputFile, temporaryLine);
            write(temporaryLine, STRING'("% %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%" & LF));
            writeline( temporaryOutputFile, temporaryLine);
            ASSERT FALSE
               REPORT LF
                      & "------------------------------"
                      & LF
                       & "-- NOTE: Break point reached in abro.esi, line 8"
                      & LF
                      & "------------------------------"
                      & LF
               SEVERITY NOTE;
         end if;
      end if;
      sig_R <= '1';
      write( temporaryLine, STRING'("R"));
      writeline( temporaryOutputFile, temporaryLine);
      sig_rst <= '0';
      sig_A <= '0';
      sig_B <= '0';
      -- Sync on CLK rising edge --
      WAIT UNTIL sig_clk'EVENT AND sig_clk = '1';
      write( temporaryLine, STRING'("% Cycle 5"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Log outputs to file --
      WriteLogOutputs( temporaryOutputFile, sig_O, sig_Odata);
      write( temporaryLine, STRING'(";"));
      writeline( temporaryOutputFile, temporaryLine);
      -- Outputs checking --


-----------------------------------
-- CLK cycle number: 6 --
-----------------------------------
      -- Inputs initialization --
      -- A (""3"");
      write( temporaryLine, STRING'("% "));
      write( temporaryLine, STRING'("A (""3"");"));
      writeline( temporaryOutputFile, temporaryLine);
      -- B (""troisieme tirage"");
```

```
      write( temporaryLine, STRING'("% "));
      write( temporaryLine, STRING'("B (""troisieme tirage"");"));
      writeline( temporaryOutputFile, temporaryLine);
      stop <= true;
   end process;

end ABRO_abro_MixedView;
```