```
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    testbench of cordic
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.04.02
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input:
--
--
--    Output:
--------------------------------------------------------------------------------

library STD;
use STD.textio.all;

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

use WORK.cordic_pkg.all;


entity cordic_tb is
end cordic_tb;




architecture beh of cordic_tb is

  component cordic
    port (
      clk, rst    : in  std_logic;
      load        : in  std_logic;
      ready       : out std_logic;
      xi, yi, zi  : in  std_logic_vector (31 downto 0);
      xo, yo, zo  : out std_logic_vector (31 downto 0) );
  end component;

  for cordic_0: cordic use entity work.cordic;



  constant nBit : integer := 32;

  signal clk, rst, load, ready : std_logic := '0';
  signal xi, yi, zi : std_logic_vector(31 downto 0) := X"0000_0000";
  signal xo, yo, zo : std_logic_vector(31 downto 0) := X"0000_0000";

 begin

   cordic_0 : cordic port map ( clk => clk, rst => rst,
                                load => load, ready => ready,
```

```
                          xi  => xi, yi  => yi, zi  => zi,
                          xo  => xo, yo  => yo, zo  => zo  );


clk <= not clk after half_period;

rst <= '0', '1' after 2* half_period;


process
begin

   wait until rst = '1';

-------------------------------------------------------------------------------
-- printf ("\nGrinding on [K, 0, 0]\n");
-- Circular (X0C, 0L, 0L);
-------------------------------------------------------------------------------
   for i in 0 to 4  loop
     wait until clk = '1';
   end loop;  -- i

   xi <= Conv2fixedPt(1.0/K, nBit);
   yi <= Conv2fixedPt(0.0, nBit);
   zi <= Conv2fixedPt(0.0, nBit);
   wait for 1 ns;
   load <= '1', '0' after clk_period;
   DispReg(xi, yi, zi, 0);

   while (ready /= '1') loop
     wait until (clk'event and clk='1');
   end loop;
   DispReg(xo, yo, zo, 1);

-------------------------------------------------------------------------------
-- printf ("\nGrinding on [K, 0, pi/6] -> [0.86602540, 0.50000000, 0]\n");
-- Circular (X0C, 0L, HalfPi / 3L);
-------------------------------------------------------------------------------
   for i in 0 to 4  loop
     wait until clk = '1';
   end loop;  -- i

   xi <= Conv2fixedPt(1.0/K, nBit);
   yi <= Conv2fixedPt(0.0, nBit);
   zi <= Conv2fixedPt(pi/6.0, nBit);
   wait for 1 ns;
   load <= '1', '0' after clk_period;
   load <= '1', '0' after clk_period;
   DispReg(xi, yi, zi, 0);

   while (ready /= '1') loop
     wait until (clk'event and clk='1');
   end loop;
   DispReg(xo, yo, zo, 1);

-------------------------------------------------------------------------------
-- printf ("\nGrinding on [K, 0, pi/4] -> [0.70710678, 0.70710678, 0]\n");
-- Circular (X0C, 0L, HalfPi / 2L);
-------------------------------------------------------------------------------
   for i in 0 to 4  loop
     wait until clk = '1';
   end loop;  -- i

   xi <= Conv2fixedPt(1.0/K, nBit);
   yi <= Conv2fixedPt(0.0, nBit);
   zi <= Conv2fixedPt(pi/4.0, nBit);
   wait for 1 ns;
   load <= '1', '0' after clk_period;
   load <= '1', '0' after clk_period;
   DispReg(xi, yi, zi, 0);
```

```
      while (ready /= '1') loop
        wait until (clk'event and clk='1');
      end loop;
      DispReg(xo, yo, zo, 1);

   ------------------------------------------------------------------------------
   -- printf ("\nGrinding on [K, 0, pi/3] -> [0.50000000, 0.86602540, 0]\n");
   -- Circular (X0C, 0L, 2L * (HalfPi / 3L));
   ------------------------------------------------------------------------------
      for i in 0 to 4  loop
        wait until clk = '1';
      end loop;  -- i

      xi <= Conv2fixedPt(1.0/K, nBit);
      yi <= Conv2fixedPt(0.0, nBit);
      zi <= Conv2fixedPt(pi/3.0, nBit);
      wait for 1 ns;
      load <= '1', '0' after clk_period;
      load <= '1', '0' after clk_period;
      DispReg(xi, yi, zi, 0);

      while (ready /= '1') loop
        wait until (clk'event and clk='1');
      end loop;
      DispReg(xo, yo, zo, 1);

      for i in 0 to 4  loop
        wait until clk = '1';
      end loop;  -- i
   end process;


   process
   begin
     wait for 2000* clk_period;
     assert false report "end of simulation" severity failure;
   end process;

  --   XXXXXXX XXXXXX XXXXXX XXXXXX XXXXXXX XXXXXX XXXXX

end beh;
```