```
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    testbench of cordic
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.03.13
--
--  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;


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;

  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";
  signal angle      : 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  );



   -- printf ("\nGrinding on [K, 0, 0]\n");
   -- Circular (X0C, 0L, 0L);
   --
   -- printf ("\nGrinding on [K, 0, pi/6] -> [0.86602540, 0.50000000, 0]\n");
   -- Circular (X0C, 0L, HalfPi / 3L);
   --
   -- printf ("\nGrinding on [K, 0, pi/4] -> [0.70710678, 0.70710678, 0]\n");
```

```
    -- Circular (X0C, 0L, HalfPi / 2L);
    --
    -- printf ("\nGrinding on [K, 0, pi/3] -> [0.50000000, 0.86602540, 0]\n");
    -- Circular (X0C, 0L, 2L * (HalfPi / 3L));

    xi <= Conv2fixedPt(0.0);
    yi <= Conv2fixedPt(0.0);
    zi <= Conv2fixedPt(0.0);
    load <= '1';

    wait until ready = '1';

    xi <= Conv2fixedPt(0.0);
    yi <= Conv2fixedPt(pi / 6.0);
    zi <= Conv2fixedPt(0.0);
    load <= '1';

    wait until ready = '1';

    xi <= Conv2fixedPt(0.0);
    yi <= Conv2fixedPt(pi / 4.0);
    zi <= Conv2fixedPt(0.0);
    load <= '1';

    wait until ready = '1';

    xi <= Conv2fixedPt(0.0);
    yi <= Conv2fixedPt(pi / 3.0);
    zi <= Conv2fixedPt(0.0);
    load <= '1';

    wait until ready = '1';



    XXXXXXX XXXXXX XXXXXX XXXXXX XXXXXXX XXXXXX XXXXX

end beh;
```