```
::::::::::::::
cordic_pkg.vhdl
::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    utility package of cordic
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.04.03
--
--  Author:
--
--    Young W. Lim
--
--  Functions:
--  Conv2fixedPt (x : real; n : integer) return std_logic_vector;
--  Conv2real (s : std_logic_vector (31 downto 0) ) return real;
--
--
--------------------------------------------------------------------------------

library STD;
use STD.textio.all;

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


package cordic_pkg is

  function Conv2fixedPt (x : real; n : integer) return std_logic_vector;
  function Conv2real (s : std_logic_vector (31 downto 0) ) return real;

  procedure DispReg (x, y, z : in std_logic_vector (31 downto 0);
                     flag : in integer );
  procedure DispAng (angle : in std_logic_vector (31 downto 0)) ;

  constant clk_period : time := 20 ns;
  constant half_period : time := clk_period / 2.0;

  constant pi : real := 3.141592653589793;
  constant K : real := 1.646760258121;

end cordic_pkg;



package body cordic_pkg is

  --------------------------------------------------------------------------------
  function Conv2fixedPt (x : real; n : integer) return std_logic_vector is
  --------------------------------------------------------------------------------
    constant shft : std_logic_vector (n-1 downto 0) := X"2000_0000";
    variable s : std_logic_vector (n-1 downto 0) ;
    variable z : real := 0.0;
  --------------------------------------------------------------------------------
  begin
      -- shft = 2^29 = 536870912
      -- bit 31 : msb - sign bit
```

```
    -- bit 30,29 : integer part
    -- bit 28 ~ 0 : fractional part
    -- for the value of 0.5
    -- first 4 msb bits [0, 0, 0, 1] --> X"1000_0000"
    --
    -- To obtain binary number representation of x,
    -- where the implicit decimal point between bit 29 and bit 28,
    -- multiply "integer converted shft"
    --
    z := x * real(to_integer(unsigned(shft)));

    s := std_logic_vector(to_signed(integer(z), n));

    return s;

  end Conv2fixedPt;
  ------------------------------------------------------------------------------


  ------------------------------------------------------------------------------
  function Conv2real (s : std_logic_vector (31 downto 0) ) return real is
  ------------------------------------------------------------------------------
    constant shft : std_logic_vector (31 downto 0) := X"2000_0000";
    variable z : real := 0.0;
  ------------------------------------------------------------------------------
  begin
    z := real(to_integer(signed(s))) / real(to_integer(unsigned(shft)));
    return z;
  end Conv2real;
  ------------------------------------------------------------------------------

  ------------------------------------------------------------------------------
  procedure DispReg (x, y, z : in std_logic_vector (31 downto 0);
                     flag : in integer ) is
  ------------------------------------------------------------------------------
    variable l : line;
  begin
    if (flag = 0) then
      write(l, String'("---------------------------------------- "));
      writeline(output, l);
      write(l, String'("  xi = ")); write(l, real'(Conv2real(x)));
      write(l, String'("  yi = ")); write(l, real'(Conv2real(y)));
      write(l, String'("  zi = ")); write(l, real'(Conv2real(z)));
    elsif (flag = 1) then
      write(l, String'("  xo = ")); write(l, real'(Conv2real(x)));
      write(l, String'("  yo = ")); write(l, real'(Conv2real(y)));
      write(l, String'("  zo = ")); write(l, real'(Conv2real(z)));
    else
      write(l, String'("  xn = ")); write(l, real'(Conv2real(x)));
      write(l, String'("  yn = ")); write(l, real'(Conv2real(y)));
      write(l, String'("  zn = ")); write(l, real'(Conv2real(z)));
    end if;
    writeline(output, l);
  end DispReg;
  ------------------------------------------------------------------------------

  ------------------------------------------------------------------------------
  procedure DispAng (angle : in std_logic_vector (31 downto 0)) is
  ------------------------------------------------------------------------------
    variable l : line;
  begin
    write(l, String'("  angle = ")); write(l, real'(Conv2real(angle)));
    writeline(output, l);
    write(l, String'("..................................... "));
    writeline(output, l);
  end DispAng;


end cordic_pkg;
::::::::::::::
```

c6.rom.vhdl

```
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    ROM Model (Data is embedded in this file as a constant)
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.11.12
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input: "angle_real.dat"
--              WD  := 32 -- data bus width
--              SH  :=  6 -- addr bus width
--              PWR := 64 -- address space (PWR = 2**SH)
--              addr(SH-1:0)
--              cs
--    Output:   data(WD-1:0)
-- ------------------------------------------------------------------------------

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 rom is
  generic (
    WD     : in natural := 32;  -- data bus width
    SH     : in natural :=  6;  -- addr bus width
    PWR    : in natural := 64); -- address space (PWR = 2**SH)

  port (
    addr   : in   std_logic_vector (SH-1 downto 0) := (others=>'0');
    cs     : in   std_logic := '0';
    data   : out  std_logic_vector (WD-1 downto 0) := (others=>'0') );

end rom;


architecture rtl of rom is
   type rarray is array (natural range <>) of real;

   constant angles : rarray :=
                     (  7.8539816339744830962E-01,
                        4.6364760900080611621E-01,
                        2.4497866312686415417E-01,
                        1.2435499454676143503E-01,
                        6.2418809995957348474E-02,
                        3.1239833430268276254E-02,
                        1.5623728620476830803E-02,
```

```
                        7.8123410601011112965E-03,
                        3.9062301319669718276E-03,
                        1.9531225164788186851E-03,
                        9.7656218955931943040E-04,
                        4.8828121119489827547E-04,
                        2.4414062014936176402E-04,
                        1.2207031189367020424E-04,
                        6.1035156174208775022E-05,
                        3.0517578115526096862E-05,
                        1.5258789061315762107E-05,
                        7.6293945311019702634E-06,
                        3.8146972656064962829E-06,
                        1.9073486328101870354E-06,
                        9.5367431640596087942E-07,
                        4.7683715820308885993E-07,
                        2.3841857910155798249E-07,
                        1.1920928955078068531E-07,
                        5.9604644775390554414E-08,
                        2.9802322387695303677E-08,
                        1.4901161193847655147E-08,
                        7.4505805969238279871E-09,
                        3.7252902984619140453E-09,
                        1.8626451492309570291E-09,
                        9.3132257461547851536E-10,
                        4.6566128730773925778E-10,
                        2.3283064365386962890E-10,
                        1.1641532182693481445E-10,
                        5.8207660913467407226E-11,
                        2.9103830456733703613E-11,
                        1.4551915228366851807E-11,
                        7.2759576141834259033E-12,
                        3.6379788070917129517E-12,
                        1.8189894035458564758E-12,
                        9.0949470177292823792E-13,
                        4.5474735088646411896E-13,
                        2.2737367544323205948E-13,
                        1.1368683772161602974E-13,
                        5.6843418860808014870E-14,
                        2.8421709430404007435E-14,
                        1.4210854715202003717E-14,
                        7.1054273576010018587E-15,
                        3.5527136788005009294E-15,
                        1.7763568394002504647E-15,
                        8.8817841970012523234E-16,
                        4.4408920985006261617E-16,
                        2.2204460492503130808E-16,
                        1.1102230246251565404E-16,
                        5.5511151231257827021E-17,
                        2.7755575615628913511E-17,
                        1.3877787807814456755E-17,
                        6.9388939039072283776E-18,
                        3.4694469519536141888E-18,
                        1.7347234759768070944E-18,
                        1.7347234759768070944E-18,
                        1.7347234759768070944E-18,
                        1.7347234759768070944E-18,
                        1.7347234759768070944E-18  );


begin

  ROM: process (addr, cs)
    type darray is array (0 to PWR-1) of std_logic_vector (WD-1 downto 0);
    variable romData : darray;
    variable initRom : boolean := false;

  begin  -- process Reg
    if (initRom=false) then
      for i in 0 to PWR-1 loop
        romData(i) := Conv2fixedPt(angles(i), WD);
```

```
      end loop;  -- i
      initRom := true;
    end if;

    if cs = '1' then
      data <= romData(to_integer(unsigned(addr)));
    else
      data <= (others=>'1');
    end if;
  end process ROM;

end rtl;
::::::::::::::
c6.rom.rfile.vhdl
::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    ROM Model (Data is read from a file)
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.11.12
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input: "angle_real.dat"
--               WD  := 32 -- data bus width
--               SH  :=  6 -- addr bus width
--               PWR := 64 -- address space (PWR = 2**SH)
--               addr(SH-1:0)
--               cs
--    Output:   data(WD-1:0)
--
--------------------------------------------------------------------------------

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 rom is
  generic (
    WD     : in natural := 32;  -- data bus width
    SH     : in natural :=  6;  -- addr bus width
    PWR    : in natural := 64); -- address space (PWR = 2**SH)

  port (
    addr   : in   std_logic_vector (SH-1 downto 0) := (others=>'0');
    cs     : in   std_logic := '0';
    data   : out  std_logic_vector (WD-1 downto 0) := (others=>'0') );

end rom;
```

```
architecture rfile of rom is
  type rarray is array (0 to PWR-1) of real;

  procedure ReadData (variable angles : out rarray) is
    file DataFile  : text open read_mode is "angle_real.dat";
    variable lbuf  : line;
    variable i     : integer := 0;
    variable rdata : real;

  begin
    while not endfile(DataFile) loop
      readline(DataFile, lbuf);
      read(lbuf, rdata);
      angles(i) := rdata;
      i := i + 1;
    end loop;
   end procedure;

begin

  ROM: process (addr, cs)
    variable angles : rarray;
    type darray is array (0 to PWR-1) of std_logic_vector (WD-1 downto 0);
    variable romData : darray;
    variable initRom : boolean := false;
  begin
    if (initRom=false) then
      ReadData(angles);
      for i in 0 to PWR-1 loop
        romData(i) := Conv2fixedPt(angles(i), WD);
      end loop;  -- i
      initRom := true;
    end if;

    if cs = '1' then
      data <= romData(to_integer(unsigned(addr)));
    else
      data <= (others=>'1');
    end if;
  end process ROM;

end rfile;
:::::::::::::::
c6.rom_tb.vhdl
:::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    general testbench of c6.rom
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.11.14
--
--  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;
use WORK.all;


entity rom_tb is
end rom_tb;


architecture beh of rom_tb is

  component rom
    generic (
      WD     : in natural := 32;  -- data bus width
      SH     : in natural :=  6;  -- addr bus width
      PWR    : in natural := 64); -- address space (PWR = 2**SH)

    port (
      addr   : in   std_logic_vector (SH-1 downto 0) := (others=>'0');
      cs     : in   std_logic := '0';
      data   : out  std_logic_vector (WD-1 downto 0) := (others=>'0') );
  end component;

  -- for rom_0: rom use entity work.rom(rfile);

  constant WD : integer := 32;
  constant SH : integer := 6;
  constant PWR : integer := 64;

  signal clk, rst: std_logic := '0';
  signal addr    : std_logic_vector(SH-1 downto 0) := (others=>'0');
  signal data    : std_logic_vector(WD-1 downto 0) := (others=>'0');

 begin

   rom_0 : rom generic map (WD=>32, SH=>6, PWR=>64)
     port map (addr => addr, cs => '1', data => data);


   clk <= not clk after half_period;

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


   process
    file DataFile  : text open write_mode is "angle_real_out.dat";
    variable lbuf  : line;
    variable wdata : integer;
   begin

      wait until rst = '1';

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

       wait for 0 ns;
```

```
      for i in 0 to PWR-1  loop
        wait until (clk'event and clk='1');

        addr <= std_logic_vector(to_unsigned(i, SH));

        wait for 0 ns;
        wait until (clk'event and clk='0');
        wdata := to_integer(signed(data));
        write(lbuf, wdata, right, 10);
        writeline(DataFile, lbuf);


        wait for 0 ns;

      end loop;


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


   process
   begin
     wait for 100* clk_period;
     assert false report "end of simulation" severity failure;
   end process;

  --   XXXXXXX XXXXXX XXXXXX XXXXXX XXXXXXX XXXXXX XXXXX

end beh;
:::::::::::::::
c6.rom_tb_rfile.vhdl
:::::::::::::::
--------------------------------------------------------------------------------
--
--  Purpose:
--
--    configuration of cca adder testbench
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.11.12
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input:
--
--
--    Output:
--------------------------------------------------------------------------------

use WORK.all;


configuration rom_tb_rfile of rom_tb is
  for beh
    for rom_0: rom
```

```
      use entity work.rom(rfile) ;
    end for;
  end for;
end rom_tb_rfile;


::::::::::::::
makefile
::::::::::::::


anal : c1.adder.vhdl c1.adder.rca.vhdl c2.addsub.vhdl c3.bshift.vhdl \
        c4.dff.vhdl c5.counter.vhdl c6.rom.vhdl c7.mux.vhdl m1.disp.vhdl \
        cordic_pkg.vhdl cordic_rtl.vhdl cordic_tb.vhdl
        ghdl -a cordic_pkg.vhdl
        ghdl -a c1.adder.rca.vhdl
        ghdl -a c2.addsub.vhdl
        ghdl -a c3.bshift.vhdl
        ghdl -a c4.dff.vhdl
        ghdl -a c5.counter.vhdl
        ghdl -a c6.rom.vhdl
        ghdl -a c7.mux.vhdl
        ghdl -a m1.disp.vhdl
        ghdl -a cordic_rtl.vhdl
        ghdl -a cordic_tb.vhdl

elab : cordic_pkg.o \
        c1.adder.rca.o c2.addsub.o c3.bshift.o c4.dff.o \
        c5.counter.o c6.rom.o c7.mux.o m1.disp.o \
        cordic_rtl.o cordic_tb.o
        ghdl -e cordic_tb

run : cordic_pkg.o cordic_rtl.o cordic_tb.o
        ghdl -r cordic_tb --vcd=cordic.vcd

all : anal elab run

wave :
        gtkwave cordic.vcd &

cordic_files :
        more c1.adder.rca.vhdl  \
             c2.addsub.vhdl     \
             c3.bshift.vhdl     \
             c4.dff.vhdl        \
             c5.counter.vhdl    \
             c6.rom.vhdl        \
             c7.mux.vhdl        \
             m1.disp.vhdl       \
             cordic_pkg.vhdl    \
             cordic_rtl.vhdl    \
             cordic_tb.vhdl     > file/cordic.rtl.files
        tar cvf file/cordic.rtl.tar *.vhdl makefile



bshift : c3.bshift.mux.vhdl bshift_tb.vhdl cordic_pkg.vhdl
        ghdl -a cordic_pkg.vhdl
        ghdl -a c7.mux.vhdl
        ghdl -a c3.bshift.mux.vhdl
        ghdl -a bshift_tb.vhdl
        ghdl -e bshift_tb
        ghdl -r bshift_tb --vcd=bshift.vcd
        # gtkwave bshift.vcd &

#-----------------------------------------------------------------------------
SRC_adder = cordic_pkg.vhdl c1.adder.vhdl c1.adder.rca.vhdl \
            c1.adder.cca.vhdl c1.adder.cca.gprom.vhdl \
            c1.adder_tb.vhdl c1.adder_tb_cca.vhdl c1.adder_tb_rca.vhdl
EXE_adder = adder_tb adder_tb_cca adder_tb_rca
```

```
adder : ${SRC_adder}
        ghdl -a cordic_pkg.vhdl
        ghdl -a c1.adder.vhdl
        ghdl -a c1.adder.rca.vhdl
        ghdl -a c1.adder.cca.gprom.vhdl
        ghdl -a c1.adder.cca.vhdl
        ghdl -a c1.adder_tb.vhdl

        \rm -f adder_tb adder_tb_cca adder_tb_rca


adder_tb:
        ghdl -e adder_tb
        ghdl -r adder_tb --disp-tree=inst --vcd=adder.vcd --stop-time=1us

        gtkwave adder.vcd &


adder_tb_cca:
        ghdl -a c1.adder_tb_cca.vhdl
        ghdl -e adder_tb_cca
        ghdl -r adder_tb_cca --disp-tree=inst --vcd=adder.vcd --stop-time=1us

        gtkwave adder.vcd &

adder_tb_rca:
        ghdl -a c1.adder_tb_rca.vhdl
        ghdl -e adder_tb_rca
        ghdl -r adder_tb_rca --disp-tree=inst --vcd=adder.vcd --stop-time=1us

        gtkwave adder.vcd &

adder_files :
        more ${SRC_adder} makefile > file/adder.files
        tar cvf file/adder.rtl.tar ${SRC_adder} makefile

#-------------------------------------------------------------------------
SRC_rom = cordic_pkg.vhdl c6.rom.vhdl c6.rom.rfile.vhdl \
            c6.rom_tb.vhdl c6.rom_tb_rfile.vhdl
EXE_rom = rom_tb rom_tb_rfile
rom : ${SRC_rom}
        ghdl -a cordic_pkg.vhdl
        ghdl -a c6.rom.vhdl
        ghdl -a c6.rom.rfile.vhdl
        ghdl -a c6.rom_tb.vhdl

        \rm -f EXE_rom

rom_tb_rfile :
        ghdl -a c6.rom_tb_rfile.vhdl
        ghdl -e rom_tb_rfile
        ghdl -r rom_tb_rfile --disp-tree=inst --vcd=rom.vcd --stop-time=1us

        gtkwave rom.vcd &

rom_files :
        more ${SRC_rom} makefile > file/rom.files
        tar cvf file/rom.rtl.tar ${SRC_rom} makefile

#-------------------------------------------------------------------------
clean :
        \rm -f *.o *~ *# *.cf
        \rm -f *_tb
        \rm -f *_conf
        \rm -f *.vcd
        \rm -f ${EXE_adder} ${EXE_rom}


allfiles :
        more makefile *.vhdl > file/all.rtl.files
```

```
tar cvf file/all.rtl.tar *.vhdl makefile
```