```
::::::::::::::
makefile
::::::::::::::
.SUFFIXES : .o .vhdl

.vhdl.o :
        ghdl -a  $<


#----------------------------------------------------------------------------
# Source Directories
#----------------------------------------------------------------------------
MUX    = "/home/young/MyWork/VLSI/a.Cblocks/Mux/rtl"
PKG    = "/home/young/MyWork/5.cordic_vhdl/e.package"

#----------------------------------------------------------------------------
# Target Directories
#----------------------------------------------------------------------------
CORDIC = "/home/young/MyWork/5.cordic_vhdl/b.rtl"


#----------------------------------------------------------------------------
import_files :
        \cp ${MUX}/c7.mux.vhdl      .
        \cp ${PKG}/cordic_pkg.vhdl  .


#----------------------------------------------------------------------------
# make CONF=mux run or make CONF=rtl run
#----------------------------------------------------------------------------
# CONF       = mux
# CONF       = rtl
#----------------------------------------------------------------------------
# CNF_bshift = c3.bshift_tb_conf.mux.o  \
# CNF_bshift = c3.bshift_tb_conf.rtl.o  \
#----------------------------------------------------------------------------
CONF       = mux

CNF_bshift = c3.bshift_tb_conf.${CONF}.o  \

OBJ        = cordic_pkg.o                 \
             c7.mux.o                     \
             c3.bshift.o                  \
             c3.bshift.mux.o              \
             c3.bshift_tb.o               \
             c3.bshift_tb_conf.${CONF}.o  \

EXE        = bshift_tb bshift_tb_conf
```

```
conf :  ${OBJ} clean_exe
        ghdl -e bshift_tb_conf


run : import_files  clean conf
        ghdl -r bshift_tb_conf --stop-time=1us --vcd=bshift.vcd
        # --disp-tree=inst --stop-time=1us --vcd=bshift.vcd
        # gtkwave bshift.vcd &


#---------------------------------------------------------------------------
export_files :
        \cp c3.bshift.vhdl          ${CORDIC}
        \cp c3.bshfit.${CONF}.vhdl  ${CORDIC}

#---------------------------------------------------------------------------
clean :
        \rm -f *.o *~ *# *.cf *.tar .print
        \rm -f *_tb
        \rm -f *_conf
        \rm -f *.vcd
        \rm -f ${EXE}

clean_exe :
        \rm -f ${EXE}

#---------------------------------------------------------------------------
SRC        = cordic_pkg.vhdl                 \
             c7.mux.vhdl                     \
             c3.bshift.vhdl                  \
             c3.bshift.mux.vhdl              \
             c3.bshift_tb.vhdl               \
             c3.bshift_tb_conf.rtl.vhdl      \
             c3.bshift_tb_conf.mux.vhdl      \


print :
        more makefile ${SRC}   > bshift.print

tar :
        mkdir src
        cp makefile ${SRC} src
        tar cvf bshift.tar src
        \rm -fr src
::::::::::::::
cordic_pkg.vhdl
::::::::::::::
----------------------------------------------------------------------------
--
--  Purpose:
```

```
--
--    utility package of cordic
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.03.22
--
--  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;
::::::::::::::
c7.mux.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    Mux
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
```

```vhdl
--
--    2012.04.03
--
--  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 mux is
  generic (
    WD      : in natural := 32);

  port (
    an    : in   std_logic_vector (WD-1 downto 0) := (others=>'0');
    bn    : in   std_logic_vector (WD-1 downto 0) := (others=>'0');
    s     : in   std_logic := '0';
    cn    : out  std_logic_vector (WD-1 downto 0) := (others=>'0') );
end mux;


architecture rtl of mux is
begin

  process (an, bn, s)
  begin  -- process
    if (s='1') then
      cn <= bn;
    else
      cn <= an;
    end if;
  end process;

end rtl;
```

```
::::::::::::::
c3.bshift.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    Barrel Shifter
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2013.10.23
--
--  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 bshift is
  generic (
    WD     : in natural := 32;
    SH     : in natural := 5 );

  port (
    di   : in  std_logic_vector (WD-1 downto 0) := (others=>'0');
    nbit : in  std_logic_vector (SH-1 downto 0) := (others=>'0');
    cs   : in  std_logic ;
    dq   : out std_logic_vector (WD-1 downto 0) := (others=>'0'));
```

```
end bshift;


::::::::::::::
c3.bshift.mux.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    Barrel Shifter Based on Mux
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2013.10.23
--
--  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;



architecture mux of bshift is

  component mux is
    generic (
      WD     : in natural := 1);
    port (
      an    : in   std_logic_vector (WD-1 downto 0);
      bn    : in   std_logic_vector (WD-1 downto 0);
      s     : in   std_logic;
      cn    : out  std_logic_vector (WD-1 downto 0) );
  end component;


  type array2d is array (WD-1 downto 0, SH-1 downto 0) of std_logic;
  signal mout: array2d  := ((others=> (others=> '0')));
  signal m_in: array2d  := ((others=> (others=> '0')));

begin

  -- j = SH-1 : top level mux row
  ILOOP: for i in WD-1 downto 0 generate
    -- Sign bit extension
    ZERO: if ((WD-1-i) < 2**(SH-1)) generate
      U0: mux generic map (WD => 1)
              port map (an(0) => di(i),
                        bn(0) => di(WD-1), -- '0', -- di(WD-1),
                        s  => nbit(SH-1),
                        cn(0) => mout(i, SH-1));
    end generate ZERO;


    -- stride 2**(SH-1)
    REST: if ((WD-1-i) >= 2**(SH-1)) generate
      U1 : mux generic map (WD => 1)
               port map (an(0) => di(i),
                         bn(0) => di(i+2**(SH-1)),
                         s  => nbit(SH-1),
                         cn(0) => mout(i, SH-1));
    end generate REST;


  end generate ILOOP;


  -- rest of mux rows
  JLOOP: for j in SH-2 downto 0 generate
    ILOOP: for i in WD-1 downto 0 generate
      -- Sign bit extension
      ZERO: if ((WD-1-i) < 2**j)  generate
        U0: mux generic map (WD => 1)
                port map (an(0) => m_in(i, j+1),
                          bn(0) => m_in(WD-1, j+1), -- '0', -- m_in(WD-1, j+1),
                          s  => nbit(j),
                          cn(0) => mout(i, j));
      end generate ZERO;
      -- Stride 2**j
      REST: if ((WD-1-i) >= 2**j) generate
        U1: mux port map (an(0) => m_in(i, j+1),
                          bn(0) => m_in(i+2**j, j+1),
                          s  => nbit(j),
                          cn(0) => mout(i, j));
      end generate REST;


    end generate ILOOP;
  end generate JLOOP;



  process (mout)
    variable tmp: array2d  := ((others=> (others=> '0')));
  begin

    tmp := mout;

    m_in <= tmp;

    for i in WD-1 downto 0 loop
        -- dq(i) <= tmp(i, 0);
      dq(i) <= tmp(i, 0) and cs;
    end loop;

    -- wait on mout; --, di, nbit;

  end process;



end mux;
::::::::::::::
c3.bshift_tb.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    testbench of bshfit
--
--  Discussion:
--
--
--  Licensing:
--
```

```
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.07.27
--
--  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 bshift_tb is
end bshift_tb;




architecture beh of bshift_tb is

  component bshift
    generic (
      WD     : in natural := 32;
      SH     : in natural := 5 );

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

  end component;
  -- for bshift_0: bshift use entity work.bshift(mux);



  constant nBit : integer := 32;

  signal clk, rst: std_logic := '0';
  signal di      : std_logic_vector(31 downto 0) := X"7FFF_FFFF";
  signal dq      : std_logic_vector(31 downto 0) := X"0000_0000";
  signal cnt     : std_logic_vector( 4 downto 0) := "00000";

 begin

   DUT: bshift generic map (WD=>32, SH=>5)
     port map (di  => di, nbit => cnt, cs => '1', dq => dq);


   clk <= not clk after half_period;

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


   process
   begin

      wait until rst = '1';

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

      -- di <= X"7FFF_FFFF";
      di <= X"8000_0000";
      wait for 0 ns;

      for i in 0 to 31  loop
        wait until (clk'event and clk='1');

        cnt <= std_logic_vector(to_unsigned(i, 5));

        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;
::::::::::::::
c3.bshift_tb_conf.rtl.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    configuration of rca bshift testbench
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2013.10.23
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input:
--
--
--    Output:
-------------------------------------------------------------------------------

use WORK.all;


configuration bshift_tb_conf of bshift_tb is
  for beh
    for DUT: bshift
    end for;
  end for;
end bshift_tb_conf;


::::::::::::::
c3.bshift_tb_conf.mux.vhdl
::::::::::::::
------------------------------------------------------------------------------
--
--  Purpose:
--
--    configuration of rca bshift testbench
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2013.10.23
--
--  Author:
--
--    Young W. Lim
--
--  Parameters:
--
--    Input:
--
--
--    Output:
-------------------------------------------------------------------------------

use WORK.all;


configuration bshift_tb_conf of bshift_tb is
  for beh
    for DUT: bshift
      use entity work.bshift(mux) ;
    end for;
  end for;
end bshift_tb_conf;
```