```vhdl
-------------------------------------------------------------------------------
--
--  Purpose:
--
--    Barrel Shifter Based on Mux
--
--  Discussion:
--
--
--  Licensing:
--
--    This code is distributed under the GNU LGPL license.
--
--  Modified:
--
--    2012.07.19
--
--  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');
    dq    : out std_logic_vector (WD-1 downto 0) := (others=>'0'));

end bshift;


architecture mux of bshift is

  component mux is
    generic (
      WD     : in natural := 32);

    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;

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


  ILOOP: for i in WD-1 downto 0 generate
    ZERO: if (i > (1 << j))  generate
      U0: mux port map (an => di(i),
                           bn => '0', i
                           s  => nbit(SH-1),
                           cn => mout(i, SH-1));
    end generate ZERO;

    REST: if (i <= (1 << j)) generate
      U1 : mux port map (an => di(i),
                            bn => di(i+(1 << (SH-1))),
                            s  => nbit(SH-1),
                            cn => mout(i, SH-1));
    end generate REST;
  end generate ILOOP;


  JLOOP: for j in SH-2 downto 0 generate
    ILOOP: for i in WD-1 downto 0 generate
      ZERO: if (i > (1 << j))  generate
        U0: mux port map (an => mout(i, j+1);
                             bn => '0',
                             s  => nbit(j),
                             cn => mout(i, j));
      end generate ZERO;

      REST: if (i <= (1 << j)) generate
        U1: mux port map (an => mout(i, j+1),
                             bn => mout(i+(1 << j), j+1),
                             s  => nbit(j),
                             cn => mout(i, j));
      end generate REST;
    end generate ILOOP;
  end generate JLOOP;

  OLOOP: for i in WD-1 downto 0 generate
    dq(i) <= mout(i, 0);
  end generate OLOOP;


end rtl;
```