

```
.....  
c3.bshift.mux.vhdl  
.....
```

```
-----  
--  
-- Purpose:  
--  
--   Barrel Shifter Based on Mux  
--  
-- 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;
```

```
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 := 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);
    end loop;

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

end process;

```

```
end mux;
::::::::::::
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);
            nbit    : in std_logic_vector (SH-1 downto 0);
            dq      : out std_logic_vector (WD-1 downto 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
```

```

bshift_0 : bshift generic map (WD=>32, SH=>5)
  port map (di => di, nbit => cnt, 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;
```