```vhdl
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.ALL;
use std.textio.all;

Entity EN_Shift is
    Generic ( N : natural := 64 );
    Port (
        A : in std_logic_vector( N-1 downto 0 );
        ShiftCount : in std_logic_vector( 5 downto 0 );
        Y_LL , Y_RL, Y_RA : out std_logic_vector (N-1 downto 0)
        );
end EN_Shift;

architecture IEEE_fn of EN_Shift is
    signal shift_val : integer range 0 to N-1;

    begin
        -- Convert ShiftCount to integer
        shift_val <= to_integer(unsigned(ShiftCount));

        -- Left logical shift
        Y_LL <= std_logic_vector(shift_left(unsigned(A), shift_val));

        -- Right logical shift
        Y_RL <= std_logic_vector(shift_right(unsigned(A), shift_val));

        -- Right arithmetic shift (Treat A as signed so the MSB is the sign bit and
        is extended)
        Y_RA <= std_logic_vector(shift_right(signed(A), shift_val));


end IEEE_fn;

architecture barrel of EN_Shift is
    -- Left logical levels
    signal ll0, ll1, ll2, ll3, ll4, ll5, ll6 : std_logic_vector(N-1 downto 0);
    -- Right logical levels
    signal rl0, rl1, rl2, rl3, rl4, rl5, rl6 : std_logic_vector(N-1 downto 0);
    -- Right arithmetic levels
    signal ra0, ra1, ra2, ra3, ra4, ra5, ra6 : std_logic_vector(N-1 downto 0);

    begin
    -- level 0: input
    ll0 <= A;
    rl0 <= A;
    ra0 <= A;

    -- level 1: shift by 1 if ShiftCount(0) = '1'
    -- Left logical
    ll1 <= ll0(N-2 downto 0) & '0'
           when ShiftCount(0) = '1' else
           ll0;

    -- Right logical
    rl1 <= '0' & rl0(N-1 downto 1)
           when ShiftCount(0) = '1' else
           rl0;

    -- Right arithmetic (sign bit extended)
    ra1 <= ra0(N-1) & ra0(N-1 downto 1)
           when ShiftCount(0) = '1' else
           ra0;

    -- level 2: shift by 2 if ShiftCount(1) = '1'
    -- Left logical
    ll2 <= ll1(N-3 downto 0) & (1 downto 0 => '0')
           when ShiftCount(1) = '1' else
           ll1;

    -- Right logical
    rl2 <= (1 downto 0 => '0') & rl1(N-1 downto 2)
           when ShiftCount(1) = '1' else
```

```
            rl1;

    -- Right arithmetic
    ra2 <= (1 downto 0 => ra1(N-1)) & ra1(N-1 downto 2)
           when ShiftCount(1) = '1' else
           ra1;

    -- level 3: shift by 4 if ShiftCount(2) = '1'
    -- Left logical
    ll3 <= ll2(N-5 downto 0) & (3 downto 0 => '0')
           when ShiftCount(2) = '1' else
           ll2;

    -- Right logical
    rl3 <= (3 downto 0 => '0') & rl2(N-1 downto 4)
           when ShiftCount(2) = '1' else
           rl2;

    -- Right arithmetic
    ra3 <= (3 downto 0 => ra2(N-1)) & ra2(N-1 downto 4)
           when ShiftCount(2) = '1' else
           ra2;

    -- level 4: shift by 8 if ShiftCount(3) = '1'
    -- Left logical
    ll4 <= ll3(N-9 downto 0) & (7 downto 0 => '0')
           when ShiftCount(3) = '1' else
           ll3;

    -- Right logical
    rl4 <= (7 downto 0 => '0') & rl3(N-1 downto 8)
           when ShiftCount(3) = '1' else
           rl3;

    -- Right arithmetic
    ra4 <= (7 downto 0 => ra3(N-1)) & ra3(N-1 downto 8)
           when ShiftCount(3) = '1' else
           ra3;

    -- level 5: shift by 16 if ShiftCount(4) = '1'
    -- Left logical
   ll5 <= ll4(N-17 downto 0) & (15 downto 0 => '0')
          when ShiftCount(4) = '1' else
          ll4;

    -- Right logical
    rl5 <= (15 downto 0 => '0') & rl4(N-1 downto 16)
          when ShiftCount(4) = '1' else
          rl4;

    -- Right arithmetic
    ra5 <= (15 downto 0 => ra4(N-1)) & ra4(N-1 downto 16)
           when ShiftCount(4) = '1' else
           ra4;

    -- level 6: shift by 32 if ShiftCount(5) = '1'
    -- Left logical
    ll6 <= ll5(N-33 downto 0) & (31 downto 0 => '0')
           when ShiftCount(5) = '1' else
           ll5;

    -- Right logical
    rl6 <= (31 downto 0 => '0') & rl5(N-1 downto 32)
            when ShiftCount(5) = '1' else
            rl5;

    -- Right arithmetic
    ra6 <= (31 downto 0 => ra5(N-1)) & ra5(N-1 downto 32)
           when ShiftCount(5) = '1' else
           ra5;

    -- Final outputs
    Y_LL <= ll6;
```

```
146         Y_RL <= rl6;
147         Y_RA <= ra6;
148 
149 
150     end barrel;
```