```
-- File : add_16.vhd

library ieee;
use ieee.std_logic_1164.all;

library lib438;
use lib438.all;

entity add_16 is
  port (
    A : in  std_logic_vector (15 downto 0);
    B : in  std_logic_vector (15 downto 0);
    C : in  std_logic;
    O : out std_logic_vector (15 downto 0);
    G : out std_logic;
    P : out std_logic);
end add_16;

architecture arch of add_16 is

  component CLA is
    generic (
      DELAY : time := 6 ns
      );
    port (
      A_IN  : in  std_logic_vector (3 downto 0);
      B_IN  : in  std_logic_vector (3 downto 0);
      C_IN  : in  std_logic;
      P     : out std_logic;
      G     : out std_logic;
      F_OUT : out std_logic_vector (3 downto 0)
      );
  end component CLA;

  component LACG is
    generic (
      DELAY : time := 6 ns
      );
    port (
      C_IN : in  std_logic;

      P0_H : in  std_logic;
      G0_H : in  std_logic;

      CX   : out std_logic;

      P1_H : in  std_logic;
      G1_H : in  std_logic;

      CY   : out std_logic;

      P2_H : in  std_logic;
      G2_H : in  std_logic;

      CZ   : out std_logic;

      P3_H : in  std_logic;
      G3_H : in  std_logic;

      GOUT : out std_logic;
      POUT : out std_logic
      );
```



```
  end component LACG;

  signal G_3 : std_logic;
  signal G_2 : std_logic;
  signal G_1 : std_logic;
  signal G_0 : std_logic;

  signal P_3 : std_logic;
  signal P_2 : std_logic;
  signal P_1 : std_logic;
  signal P_0 : std_logic;

  signal C_3 : std_logic;
  signal C_2 : std_logic;
  signal C_1 : std_logic;
  signal C_0 : std_logic;

begin  -- arch

  GROUP_3 : CLA
    port map (
      A_IN  => A (15 downto 12),
      B_IN  => B (15 downto 12),
      C_IN  => C_3,
      P     => P_3,
      G     => G_3,
      F_OUT => O (15 downto 12));

  GROUP_2 : CLA
    port map (
      A_IN  => A (11 downto 8),
      B_IN  => B (11 downto 8),
      C_IN  => C_2,
      P     => P_2,
      G     => G_2,
      F_OUT => O (11 downto 8));

  GROUP_1 : CLA
    port map (
      A_IN  => A (7 downto 4),
      B_IN  => B (7 downto 4),
      C_IN  => C_1,
      P     => P_1,
      G     => G_1,
      F_OUT => O (7 downto 4));

  GROUP_0 : CLA
    port map (
      A_IN  => A (3 downto 0),
      B_IN  => B (3 downto 0),
      C_IN  => C,
      P     => P_0,
      G     => G_0,
      F_OUT => O (3 downto 0));

  LACG_0 : LACG
    port map (
      P3_H => P_3,
      G3_H => G_3,
      P2_H => P_2,
      G2_H => G_2,
      P1_H => P_1,
```

```
      G1_H => G_1,
      P0_H => P_0,
      G0_H => G_0,
      C_IN => C,
      CZ   => C_3,
      CY   => C_2,
      CX   => C_1,
      GOUT => G,
      POUT => P);

end arch;
```

```
-- File : add_64.vhd

library ieee;
use ieee.std_logic_1164.all;

library lib438;
use lib438.all;

entity add_64 is
  port (
    A : in  std_logic_vector (63 downto 0);
    B : in  std_logic_vector (63 downto 0);
    C : in  std_logic;
    G : out std_logic;
    P : out std_logic;
    O : out std_logic_vector (63 downto 0)
    );
end add_64;

architecture arch of add_64 is

  component add_16 is
    port (
      A : in  std_logic_vector (15 downto 0);
      B : in  std_logic_vector (15 downto 0);
      C : in  std_logic;
      O : out std_logic_vector (15 downto 0);
      G : out std_logic;
      P : out std_logic
      );
  end component add_16;

  component LACG is
    generic (
      DELAY : time := 6 ns
      );
    port (
      C_IN : in std_logic;

      P0_H : in std_logic;
      G0_H : in std_logic;

      CX : out std_logic;

      P1_H : in std_logic;
      G1_H : in std_logic;

      CY : out std_logic;

      P2_H : in std_logic;
      G2_H : in std_logic;

      CZ : out std_logic;

      P3_H : in std_logic;
      G3_H : in std_logic;

      GOUT : out std_logic;
      POUT : out std_logic
      );
  end component LACG;
```



```
  signal G_3 : std_logic;
  signal G_2 : std_logic;
  signal G_1 : std_logic;
  signal G_0 : std_logic;

  signal P_3 : std_logic;
  signal P_2 : std_logic;
  signal P_1 : std_logic;
  signal P_0 : std_logic;

  signal C_3 : std_logic;
  signal C_2 : std_logic;
  signal C_1 : std_logic;
  signal C_0 : std_logic;

begin  -- arch

  GROUP_3 : ADD_16
    port map (
      A => A (63 downto 48),
      B => B (63 downto 48),
      C => C_3,
      P => P_3,
      G => G_3,
      O => O (63 downto 48));

  GROUP_2 : ADD_16
    port map (
      A => A (47 downto 32),
      B => B (47 downto 32),
      C => C_2,
      P => P_2,
      G => G_2,
      O => O (47 downto 32));

  GROUP_1 : ADD_16
    port map (
      A => A (31 downto 16),
      B => B (31 downto 16),
      C => C_1,
      P => P_1,
      G => G_1,
      O => O (31 downto 16));

  GROUP_0 : ADD_16
    port map (
      A => A (15 downto 0),
      B => B (15 downto 0),
      C => C,
      P => P_0,
      G => G_0,
      O => O (15 downto 0));

  LACG_0 : LACG
    port map (
      P3_H => P_3,
      G3_H => G_3,
      P2_H => P_2,
      G2_H => G_2,
      P1_H => P_1,
      G1_H => G_1,
      P0_H => P_0,
```

```
      G0_H => G_0,
      C_IN => C,
      CZ   => C_3,
      CY   => C_2,
      CX   => C_1,
      GOUT => G,
      POUT => P);

end arch;
```

```
-- File : exp_add.vhd

library ieee;
use ieee.std_logic_1164.all;

entity exp_add is
  port (
    A_EXP : in  std_logic_vector(7 downto 0);
    B_EXP : in  std_logic_vector(7 downto 0);
    ADJ   : in  std_logic_vector(4 downto 0);  -- adjustment needed from
                                               -- right shifting mantissa
    O_EXP : out std_logic_vector(7 downto 0)
    );
end exp_add;

architecture arch of exp_add is

  component sub_8 is
    port (
      A : in  std_logic_vector(7 downto 0);
      B : in  std_logic_vector(7 downto 0);
      D : out std_logic_vector(7 downto 0)
      );
  end component sub_8;

  component add_16 is
    port (
      A : in  std_logic_vector (15 downto 0);
      B : in  std_logic_vector (15 downto 0);
      C : in  std_logic;
      O : out std_logic_vector (15 downto 0);
      G : out std_logic;
      P : out std_logic);
  end component add_16;

  signal s_a_b_sum   : std_logic_vector(15 downto 0);
  signal s_a_b_sum_p : std_logic;
  signal s_a_b_sum_g : std_logic;

  signal s_a_b_adj_sum   : std_logic_vector(15 downto 0);
  signal s_a_b_adj_sum_p : std_logic;
  signal s_a_b_adj_sum_g : std_logic;

  signal s_output : std_logic_vector(7 downto 0);

begin  -- arch

  O_EXP <= s_output;

  add_0 : add_16
    port map (
      A(15 downto 8) => (others => '0'),
      A(7 downto 0)  => A_EXP,
      B(15 downto 8) => (others => '0'),
      B(7 downto 0)  => B_EXP,
      C              => '0',
      O              => s_a_b_sum,
      G              => s_a_b_sum_g,
      P              => s_a_b_sum_p
      );

  add_1 : add_16
```



```
    port map (
      A => s_a_b_sum,
      B(15 downto 5) => (others => '0'),
      B(4 downto 0)  => ADJ,
      C => '0',
      O => s_a_b_adj_sum,
      G => s_a_b_adj_sum_g,
      P => s_a_b_adj_sum_p
      );

  sub_0 : sub_8
    port map (
      A => s_a_b_adj_sum(7 downto 0),
      B => X"7F",                        -- substract bias (127)
      D => s_output
      );

end arch;
```

```
-- File : fp_mult.vhd

library ieee;
use ieee.std_logic_1164.all;

entity fp_mult is
  port (
    A_H : in  std_logic_vector (31 downto 0);
    B_H : in  std_logic_vector (31 downto 0);
    O_H : out std_logic_vector (31 downto 0)
    );
end fp_mult;

architecture arch of fp_mult is

  component mant_mult is
    port (
      A_EXP : in  std_logic_vector(7 downto 0);
      B_EXP : in  std_logic_vector(7 downto 0);
      A_H : in  std_logic_vector(22 downto 0);
      B_H : in  std_logic_vector(22 downto 0);
      C_H : out std_logic;
      O_H : out std_logic_vector(47 downto 0));
  end component mant_mult;

  component post_norm is
    port (
      N_MANT : in  std_logic_vector(47 downto 0);
      ADJ    : out std_logic_vector(4 downto 0);
      O_MANT : out std_logic_vector(22 downto 0)
      );
  end component post_norm;

  component exp_add is
    port (
      A_EXP : in  std_logic_vector(7 downto 0);
      B_EXP : in  std_logic_vector(7 downto 0);
      ADJ   : in  std_logic_vector(4 downto 0);
      O_EXP : out std_logic_vector(7 downto 0)
      );
  end component exp_add;

  signal s_mm_c : std_logic;
  signal s_mm_o : std_logic_vector(47 downto 0);

  signal s_ea_adj : std_logic_vector(4 downto 0);

begin  -- arch

  O_H(31) <= A_H(31) xor B_H(31);

  MM_0 : MANT_MULT
    port map (
      A_EXP => A_H(30 downto 23),
      B_EXP => B_H(30 downto 23),
      A_H => A_H(22 downto 0),
      B_H => B_H(22 downto 0),
      C_H => s_mm_c,
      O_H => s_mm_o
      );

  EA_0 : EXP_ADD
```



```
    port map (
      A_EXP => A_H(30 downto 23),
      B_EXP => B_H(30 downto 23),
      ADJ   => s_ea_adj,
      O_EXP => O_H(30 downto 23)
      );

  PN_0 : POST_NORM
    port map (
      N_MANT => s_mm_o,
      ADJ    => s_ea_adj,
      O_MANT => O_H(22 downto 0)
      );

end arch;
```

```
-- File : left_shifter

library ieee;
use ieee.std_logic_1164.all;

entity left_shifter is
  port (
    A : in  std_logic_vector(23 downto 0);
    S : in  std_logic_vector(4 downto 0);
    O : out std_logic_vector(47 downto 0)
    );
end left_shifter;

architecture arch of left_shifter is

  component x_2_to_1_mux is
    port (
      A : in  std_logic_vector(47 downto 0);
      B : in  std_logic_vector(47 downto 0);
      S : in  std_logic;
      O : out std_logic_vector(47 downto 0)
      );
  end component x_2_to_1_mux;

  signal s_8_to_16 : std_logic_vector(47 downto 0);
  signal s_4_to_8  : std_logic_vector(47 downto 0);
  signal s_2_to_4  : std_logic_vector(47 downto 0);
  signal s_1_to_2  : std_logic_vector(47 downto 0);

begin  -- arch

  -- 1-bit shift
  X21M_1 : x_2_to_1_mux
    port map (
      A(47 downto 24) => (others => '0'),
      A(23 downto 0)  => A,
      B(47 downto 25) => (others => '0'),
      B(24 downto 1)  => A (23 downto 0),
      B(0)            => '0',
      S               => S(0),
      O               => s_1_to_2
      );

  -- 2-bit shift
  X21M_2 : x_2_to_1_mux
    port map (
      A              => s_1_to_2,
      B(47 downto 2) => s_1_to_2(45 downto 0),
      B(1 downto 0)  => (others => '0'),
      S              => S(1),
      O              => s_2_to_4
      );

  -- 4-bit shift
  X21M_4 : x_2_to_1_mux
    port map (
      A              => s_2_to_4,
      B(47 downto 4) => s_2_to_4(43 downto 0),
      B(3 downto 0)  => (others => '0'),
      S              => S(2),
      O              => s_4_to_8
      );
```



```
  -- 8-bit shift
  X21M_8 : x_2_to_1_mux
    port map (
      A              => s_4_to_8,
      B(47 downto 8) => s_4_to_8(39 downto 0),
      B(7 downto 0)  => (others => '0'),
      S              => S(3),
      O              => s_8_to_16
      );

  -- 16-bit shift
  X21M_16 : x_2_to_1_mux
    port map (
      A               => s_8_to_16,
      B(47 downto 16) => s_8_to_16(31 downto 0),
      B(15 downto 0)  => (others => '0'),
      S               => S(4),
      O               => O
      );

end arch;
```

```
-- File : mant_mult.vhd

library ieee;
use ieee.std_logic_1164.all;

library lib438;
use lib438.all;

entity mant_mult is
  port (
    A_EXP : in  std_logic_vector(7 downto 0);
    B_EXP : in  std_logic_vector(7 downto 0);
    A_H : in  std_logic_vector(22 downto 0);
    B_H : in  std_logic_vector(22 downto 0);
    C_H : out std_logic;
    O_H : out std_logic_vector(47 downto 0));
end mant_mult;

architecture arch of mant_mult is

  component left_shifter is
    port (
      A : in  std_logic_vector(23 downto 0);
      S : in  std_logic_vector(4 downto 0);
      O : out std_logic_vector(47 downto 0)
      );
  end component left_shifter;

  component adder64 is
    port (
      A : in  std_logic_vector(63 downto 0);
      B : in  std_logic_vector(63 downto 0);
      C : in  std_logic;
      G : out std_logic;
      P : out std_logic;
      O : out std_logic_vector(63 downto 0)
      );
  end component adder64;

  component RRU3_2 is
    generic (DLY_TIME : time := 2 ns);
    port (
      A_VAL  : in  std_logic_vector(47 downto 0);
      B_VAL  : in  std_logic_vector(47 downto 0);
      C_VAL  : in  std_logic_vector(47 downto 0);
      F0_VAL : out std_logic_vector(47 downto 0);
      F1_VAL : out std_logic_vector(47 downto 0)
      );
  end component RRU3_2;

  component RRU7_3 is
    port (
      A_VAL  : in  std_logic_vector(47 downto 0);
      B_VAL  : in  std_logic_vector(47 downto 0);
      C_VAL  : in  std_logic_vector(47 downto 0);
      D_VAL  : in  std_logic_vector(47 downto 0);
      E_VAL  : in  std_logic_vector(47 downto 0);
      F_VAL  : in  std_logic_vector(47 downto 0);
      G_VAL  : in  std_logic_vector(47 downto 0);
      F0_VAL : out std_logic_vector(47 downto 0);
      F1_VAL : out std_logic_vector(47 downto 0);
      F2_VAL : out std_logic_vector(47 downto 0)
```



```
      );
  end component RRU7_3;

  signal pp_00 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_01 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_02 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_03 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_04 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_05 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_06 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_07 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_08 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_09 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_10 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_11 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_12 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_13 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_14 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_15 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_16 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_17 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_18 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_19 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_20 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_21 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_22 : std_logic_vector(47 downto 0) := (others => '0');
  signal pp_23 : std_logic_vector(47 downto 0) := (others => '0');

  signal s_shift_01 : std_logic_vector(4 downto 0);
  signal s_shift_02 : std_logic_vector(4 downto 0);
  signal s_shift_03 : std_logic_vector(4 downto 0);
  signal s_shift_04 : std_logic_vector(4 downto 0);
  signal s_shift_05 : std_logic_vector(4 downto 0);
  signal s_shift_06 : std_logic_vector(4 downto 0);
  signal s_shift_07 : std_logic_vector(4 downto 0);
  signal s_shift_08 : std_logic_vector(4 downto 0);
  signal s_shift_09 : std_logic_vector(4 downto 0);
  signal s_shift_10 : std_logic_vector(4 downto 0);
  signal s_shift_11 : std_logic_vector(4 downto 0);
  signal s_shift_12 : std_logic_vector(4 downto 0);
  signal s_shift_13 : std_logic_vector(4 downto 0);
  signal s_shift_14 : std_logic_vector(4 downto 0);
  signal s_shift_15 : std_logic_vector(4 downto 0);
  signal s_shift_16 : std_logic_vector(4 downto 0);
  signal s_shift_17 : std_logic_vector(4 downto 0);
  signal s_shift_18 : std_logic_vector(4 downto 0);
  signal s_shift_19 : std_logic_vector(4 downto 0);
  signal s_shift_20 : std_logic_vector(4 downto 0);
  signal s_shift_21 : std_logic_vector(4 downto 0);
  signal s_shift_22 : std_logic_vector(4 downto 0);
  signal s_shift_23 : std_logic_vector(4 downto 0);

  signal s_stage0_output00 : std_logic_vector(47 downto 0);
  signal s_stage0_output01 : std_logic_vector(47 downto 0);
  signal s_stage0_output02 : std_logic_vector(47 downto 0);
  signal s_stage0_output03 : std_logic_vector(47 downto 0);
  signal s_stage0_output04 : std_logic_vector(47 downto 0);
  signal s_stage0_output05 : std_logic_vector(47 downto 0);
  signal s_stage0_output06 : std_logic_vector(47 downto 0);
  signal s_stage0_output07 : std_logic_vector(47 downto 0);
  signal s_stage0_output08 : std_logic_vector(47 downto 0);
  signal s_stage0_output09 : std_logic_vector(47 downto 0);
```

```
  signal s_stage0_output10 : std_logic_vector(47 downto 0);

  signal s_stage1_output00 : std_logic_vector(47 downto 0);
  signal s_stage1_output01 : std_logic_vector(47 downto 0);
  signal s_stage1_output02 : std_logic_vector(47 downto 0);
  signal s_stage1_output03 : std_logic_vector(47 downto 0);
  signal s_stage1_output04 : std_logic_vector(47 downto 0);
  signal s_stage1_output05 : std_logic_vector(47 downto 0);

  signal s_stage2_output00 : std_logic_vector(47 downto 0);
  signal s_stage2_output01 : std_logic_vector(47 downto 0);
  signal s_stage2_output02 : std_logic_vector(47 downto 0);

  signal s_stage3_output00 : std_logic_vector(47 downto 0);
  signal s_stage3_output01 : std_logic_vector(47 downto 0);

  signal s_pp_sum_p : std_logic;
  signal s_pp_sum_g : std_logic;
  signal s_pp_sum_o : std_logic_vector(63 downto 0);

  signal s_in_01 : std_logic_vector(23 downto 0);
  signal s_in_02 : std_logic_vector(23 downto 0);
  signal s_in_03 : std_logic_vector(23 downto 0);
  signal s_in_04 : std_logic_vector(23 downto 0);
  signal s_in_05 : std_logic_vector(23 downto 0);
  signal s_in_06 : std_logic_vector(23 downto 0);
  signal s_in_07 : std_logic_vector(23 downto 0);
  signal s_in_08 : std_logic_vector(23 downto 0);
  signal s_in_09 : std_logic_vector(23 downto 0);
  signal s_in_10 : std_logic_vector(23 downto 0);
  signal s_in_11 : std_logic_vector(23 downto 0);
  signal s_in_12 : std_logic_vector(23 downto 0);
  signal s_in_13 : std_logic_vector(23 downto 0);
  signal s_in_14 : std_logic_vector(23 downto 0);
  signal s_in_15 : std_logic_vector(23 downto 0);
  signal s_in_16 : std_logic_vector(23 downto 0);
  signal s_in_17 : std_logic_vector(23 downto 0);
  signal s_in_18 : std_logic_vector(23 downto 0);
  signal s_in_19 : std_logic_vector(23 downto 0);
  signal s_in_20 : std_logic_vector(23 downto 0);
  signal s_in_21 : std_logic_vector(23 downto 0);
  signal s_in_22 : std_logic_vector(23 downto 0);
  signal s_in_23 : std_logic_vector(23 downto 0);

  constant zeros : std_logic_vector(24 downto 0) := (others => '0');

  signal s_fa : std_logic_vector(23 downto 0);
  signal s_fb : std_logic_vector(23 downto 0);

  function or_reduce( V: std_logic_vector )
    return std_ulogic is variable result: std_ulogic;
  begin
    for i in V'range loop
      if i = V'left then
        result := V(i);
      else
        result := result OR V(i);
      end if;
      exit when result = '1';
    end loop;
    return result;
  end or_reduce;
```



```
begin  -- arch

  C_H <= '1' when s_pp_sum_o(48) = '1' else '0';

  O_H <= zeros & A_H             when B_H = zeros(22 downto 0) else
         zeros & B_H             when A_H = zeros(22 downto 0) else
         s_pp_sum_o(47 downto 0) ;-- when s_pp_sum_o(46) = '0'  else
         --(others                                          => '1');

  s_fa <= or_reduce(A_EXP) & A_H;
  s_fb <= or_reduce(B_EXP) & B_H;

  pp_00(47 downto 24) <= (others                                  => '0');
  pp_00(23 downto 0)  <= s_fa when B_H(0) = '1' else (others => '0');

  s_shift_01 <= "00001" when B_H(1) = '1'  else "00000";
  s_shift_02 <= "00010" when B_H(2) = '1'  else "00000";
  s_shift_03 <= "00011" when B_H(3) = '1'  else "00000";
  s_shift_04 <= "00100" when B_H(4) = '1'  else "00000";
  s_shift_05 <= "00101" when B_H(5) = '1'  else "00000";
  s_shift_06 <= "00110" when B_H(6) = '1'  else "00000";
  s_shift_07 <= "00111" when B_H(7) = '1'  else "00000";
  s_shift_08 <= "01000" when B_H(8) = '1'  else "00000";
  s_shift_09 <= "01001" when B_H(9) = '1'  else "00000";
  s_shift_10 <= "01010" when B_H(10) = '1' else "00000";
  s_shift_11 <= "01011" when B_H(11) = '1' else "00000";
  s_shift_12 <= "01100" when B_H(12) = '1' else "00000";
  s_shift_13 <= "01101" when B_H(13) = '1' else "00000";
  s_shift_14 <= "01110" when B_H(14) = '1' else "00000";
  s_shift_15 <= "01111" when B_H(15) = '1' else "00000";
  s_shift_16 <= "10000" when B_H(16) = '1' else "00000";
  s_shift_17 <= "10001" when B_H(17) = '1' else "00000";
  s_shift_18 <= "10010" when B_H(18) = '1' else "00000";
  s_shift_19 <= "10011" when B_H(19) = '1' else "00000";
  s_shift_20 <= "10100" when B_H(20) = '1' else "00000";
  s_shift_21 <= "10101" when B_H(21) = '1' else "00000";
  s_shift_22 <= "10110" when B_H(22) = '1' else "00000";
  s_shift_23 <= "10111" when s_fb(23) = '1' else "00000";

  s_in_01 <= s_fa when B_H(1) = '1'  else (others => '0');
  s_in_02 <= s_fa when B_H(2) = '1'  else (others => '0');
  s_in_03 <= s_fa when B_H(3) = '1'  else (others => '0');
  s_in_04 <= s_fa when B_H(4) = '1'  else (others => '0');
  s_in_05 <= s_fa when B_H(5) = '1'  else (others => '0');
  s_in_06 <= s_fa when B_H(6) = '1'  else (others => '0');
  s_in_07 <= s_fa when B_H(7) = '1'  else (others => '0');
  s_in_08 <= s_fa when B_H(8) = '1'  else (others => '0');
  s_in_09 <= s_fa when B_H(9) = '1'  else (others => '0');
  s_in_10 <= s_fa when B_H(10) = '1' else (others => '0');
  s_in_11 <= s_fa when B_H(11) = '1' else (others => '0');
  s_in_12 <= s_fa when B_H(12) = '1' else (others => '0');
  s_in_13 <= s_fa when B_H(13) = '1' else (others => '0');
  s_in_14 <= s_fa when B_H(14) = '1' else (others => '0');
  s_in_15 <= s_fa when B_H(15) = '1' else (others => '0');
  s_in_16 <= s_fa when B_H(16) = '1' else (others => '0');
  s_in_17 <= s_fa when B_H(17) = '1' else (others => '0');
  s_in_18 <= s_fa when B_H(18) = '1' else (others => '0');
  s_in_19 <= s_fa when B_H(19) = '1' else (others => '0');
  s_in_20 <= s_fa when B_H(20) = '1' else (others => '0');
  s_in_21 <= s_fa when B_H(21) = '1' else (others => '0');
  s_in_22 <= s_fa when B_H(22) = '1' else (others => '0');
```

```
  s_in_23 <= s_fa when s_fb(23) = '1' else (others => '0');

  SHIFT_01 : LEFT_SHIFTER
    port map (
      A => s_in_01,
      S => s_shift_01,
      O => pp_01
      );

  SHIFT_02 : LEFT_SHIFTER
    port map (
      A => s_in_02,
      S => s_shift_02,
      O => pp_02
      );

  SHIFT_03 : LEFT_SHIFTER
    port map (
      A => s_in_03,
      S => s_shift_03,
      O => pp_03
      );

  SHIFT_04 : LEFT_SHIFTER
    port map (
      A => s_in_04,
      S => s_shift_04,
      O => pp_04
      );

  SHIFT_05 : LEFT_SHIFTER
    port map (
      A => s_in_05,
      S => s_shift_05,
      O => pp_05
      );

  SHIFT_06 : LEFT_SHIFTER
    port map (
      A => s_in_06,
      S => s_shift_06,
      O => pp_06
      );

  SHIFT_07 : LEFT_SHIFTER
    port map (
      A => s_in_07,
      S => s_shift_07,
      O => pp_07
      );

  SHIFT_08 : LEFT_SHIFTER
    port map (
      A => s_in_08,
      S => s_shift_08,
      O => pp_08
      );

  SHIFT_09 : LEFT_SHIFTER
    port map (
      A => s_in_09,
      S => s_shift_09,
```



```
      O => pp_09
      );

  SHIFT_10 : LEFT_SHIFTER
    port map (
      A => s_in_10,
      S => s_shift_10,
      O => pp_10
      );

  SHIFT_11 : LEFT_SHIFTER
    port map (
      A => s_in_11,
      S => s_shift_11,
      O => pp_11
      );

  SHIFT_12 : LEFT_SHIFTER
    port map (
      A => s_in_12,
      S => s_shift_12,
      O => pp_12
      );

  SHIFT_13 : LEFT_SHIFTER
    port map (
      A => s_in_13,
      S => s_shift_13,
      O => pp_13
      );

  SHIFT_14 : LEFT_SHIFTER
    port map (
      A => s_in_14,
      S => s_shift_14,
      O => pp_14
      );

  SHIFT_15 : LEFT_SHIFTER
    port map (
      A => s_in_15,
      S => s_shift_15,
      O => pp_15
      );

  SHIFT_16 : LEFT_SHIFTER
    port map (
      A => s_in_16,
      S => s_shift_16,
      O => pp_16
      );

  SHIFT_17 : LEFT_SHIFTER
    port map (
      A => s_in_17,
      S => s_shift_17,
      O => pp_17
      );

  SHIFT_18 : LEFT_SHIFTER
    port map (
      A => s_in_18,
```

```
    S => s_shift_18,
    O => pp_18
    );

SHIFT_19 : LEFT_SHIFTER
  port map (
    A => s_in_19,
    S => s_shift_19,
    O => pp_19
    );

SHIFT_20 : LEFT_SHIFTER
  port map (
    A => s_in_20,
    S => s_shift_20,
    O => pp_20
    );

SHIFT_21 : LEFT_SHIFTER
  port map (
    A => s_in_21,
    S => s_shift_21,
    O => pp_21
    );

SHIFT_22 : LEFT_SHIFTER
  port map (
    A => s_in_22,
    S => s_shift_22,
    O => pp_22
    );

SHIFT_23 : LEFT_SHIFTER
  port map (
    A => s_in_23,
    S => s_shift_23,
    O => pp_23
    );

STAGE0_RRU7TO3_0 : RRU7_3
  port map (
    A_VAL  => pp_00,
    B_VAL  => pp_01,
    C_VAL  => pp_02,
    D_VAL  => pp_03,
    E_VAL  => pp_04,
    F_VAL  => pp_05,
    G_VAL  => pp_06,
    F0_VAL => s_stage0_output00,
    F1_VAL => s_stage0_output01,
    F2_VAL => s_stage0_output02
    );

STAGE0_RRU7TO3_1 : RRU7_3
  port map (
    A_VAL  => pp_07,
    B_VAL  => pp_08,
    C_VAL  => pp_09,
    D_VAL  => pp_10,
    E_VAL  => pp_11,
    F_VAL  => pp_12,
    G_VAL  => pp_13,
```



```
    F0_VAL => s_stage0_output03,
    F1_VAL => s_stage0_output04,
    F2_VAL => s_stage0_output05
    );

STAGE0_RRU7TO3_2 : RRU7_3
  port map (
    A_VAL  => pp_14,
    B_VAL  => pp_15,
    C_VAL  => pp_16,
    D_VAL  => pp_17,
    E_VAL  => pp_18,
    F_VAL  => pp_19,
    G_VAL  => pp_20,
    F0_VAL => s_stage0_output06,
    F1_VAL => s_stage0_output07,
    F2_VAL => s_stage0_output08
    );

STAGE0_RRU3TO2_0 : RRU3_2
  port map (
    A_VAL  => pp_21,
    B_VAL  => pp_22,
    C_VAL  => pp_23,
    F0_VAL => s_stage0_output09,
    F1_VAL => s_stage0_output10
    );

STAGE1_RRU7TO3_0 : RRU7_3
  port map (
    A_VAL  => s_stage0_output00,
    B_VAL  => s_stage0_output01,
    C_VAL  => s_stage0_output02,
    D_VAL  => s_stage0_output03,
    E_VAL  => s_stage0_output04,
    F_VAL  => s_stage0_output05,
    G_VAL  => s_stage0_output06,
    F0_VAL => s_stage1_output00,
    F1_VAL => s_stage1_output01,
    F2_VAL => s_stage1_output02
    );

STAGE1_RRU7TO3_1 : RRU7_3
  port map (
    A_VAL  => s_stage0_output07,
    B_VAL  => s_stage0_output08,
    C_VAL  => s_stage0_output09,
    D_VAL  => s_stage0_output10,
    E_VAL  => (others => '0'),
    F_VAL  => (others => '0'),
    G_VAL  => (others => '0'),
    F0_VAL => s_stage1_output03,
    F1_VAL => s_stage1_output04,
    F2_VAL => s_stage1_output05
    );

STAGE2_RRU7TO3_0 : RRU7_3
  port map (
    A_VAL  => s_stage1_output00,
    B_VAL  => s_stage1_output01,
    C_VAL  => s_stage1_output02,
    D_VAL  => s_stage1_output03,
```

```
      E_VAL  => s_stage1_output04,
      F_VAL  => s_stage1_output05,
      G_VAL  => (others => '0'),
      F0_VAL => s_stage2_output00,
      F1_VAL => s_stage2_output01,
      F2_VAL => s_stage2_output02
      );

  STAGE3_RRU3TO2_0 : RRU3_2
    port map (
      A_VAL  => s_stage2_output00,
      B_VAL  => s_stage2_output01,
      C_VAL  => s_stage2_output02,
      F0_VAL => s_stage3_output00,
      F1_VAL => s_stage3_output01
      );

  ADD_PP_ARRAY : ADDER64
    port map (
      A(63 downto 48) => (others => '0'),
      A(47 downto 0)  => s_stage3_output00,
      B(63 downto 48) => (others => '0'),
      B(47 downto 0)  => s_stage3_output01,
      C               => '0',
      O               => s_pp_sum_o,
      G               => s_pp_sum_g,
      P               => s_pp_sum_p
      );

end arch;
```

```
-- File : post_norm.vhd

library ieee;
use ieee.std_logic_1164.all;

entity post_norm is
  port (
    N_MANT : in  std_logic_vector(47 downto 0);
    ADJ    : out std_logic_vector(4 downto 0);
    O_MANT : out std_logic_vector(22 downto 0)
    );
end post_norm;

architecture arch of post_norm is

  signal s_a_shift : std_logic_vector(23 downto 0);
  signal s_s_shift : std_logic_vector(4 downto 0);

begin  -- arch

  ADJ <= B"0_0001" when N_MANT(47) = '1' else
         B"0_0000";

  O_MANT <= s_a_shift(22 downto 0);

  s_a_shift <= N_MANT(47 downto 24) when N_MANT(47) = '1' else
               N_MANT(46 downto 23) when N_MANT(46) = '1' else
               N_MANT(45 downto 22) when N_MANT(45) = '1' else
               N_MANT(44 downto 21) when N_MANT(44) = '1' else
               N_MANT(43 downto 20) when N_MANT(43) = '1' else
               N_MANT(42 downto 19) when N_MANT(42) = '1' else
               N_MANT(41 downto 18) when N_MANT(41) = '1' else
               N_MANT(40 downto 17) when N_MANT(40) = '1' else
               N_MANT(39 downto 16) when N_MANT(39) = '1' else
               N_MANT(38 downto 15) when N_MANT(38) = '1' else
               N_MANT(37 downto 14) when N_MANT(37) = '1' else
               N_MANT(36 downto 13) when N_MANT(36) = '1' else
               N_MANT(35 downto 12) when N_MANT(35) = '1' else
               N_MANT(34 downto 11) when N_MANT(34) = '1' else
               N_MANT(33 downto 10) when N_MANT(33) = '1' else
               N_MANT(32 downto  9) when N_MANT(32) = '1' else
               N_MANT(31 downto  8) when N_MANT(31) = '1' else
               N_MANT(30 downto  7) when N_MANT(30) = '1' else
               N_MANT(29 downto  6) when N_MANT(29) = '1' else
               N_MANT(28 downto  5) when N_MANT(28) = '1' else
               N_MANT(27 downto  4) when N_MANT(27) = '1' else
               N_MANT(26 downto  3) when N_MANT(26) = '1' else
               N_MANT(25 downto  2) when N_MANT(25) = '1' else
               N_MANT(24 downto  1) when N_MANT(24) = '1' else
               N_MANT(23 downto  0);
end arch;
```

```
-- File : right_shifter.vhd

library ieee;
use ieee.std_logic_1164.all;

entity right_shifter is
  port (
    A : in  std_logic_vector(22 downto 0);
    S : in  std_logic_vector(4 downto 0);
    O : out std_logic_vector(22 downto 0)
    );
end right_shifter;

architecture arch of right_shifter is

  component x_2_to_1_mux_46 is
    port (
      A : in  std_logic_vector(47 downto 0);
      B : in  std_logic_vector(47 downto 0);
      S : in  std_logic;
      O : out std_logic_vector(47 downto 0)
      );
  end component x_2_to_1_mux_46;

  signal s_8_to_16 : std_logic_vector(47 downto 0);
  signal s_4_to_8  : std_logic_vector(47 downto 0);
  signal s_2_to_4  : std_logic_vector(47 downto 0);
  signal s_1_to_2  : std_logic_vector(47 downto 0);

  signal s_dont_care : std_logic_vector(24 downto 0);

begin  -- arch

  -- 1-bit shift
  X21M_1 : x_2_to_1_mux_46
    port map (
      A(47 downto 23) => (others => '0'),
      A(22 downto 0)  => A,
      B(47 downto 22) => (others => '0'),
      B(21 downto 0)  => A(22 downto 1),
      S               => S(0),
      O               => s_1_to_2
      );

  -- 2-bit shift
  X21M_2 : x_2_to_1_mux_46
    port map (
      A               => s_1_to_2,
      B(47 downto 21) => (others => '0'),
      B(20 downto 0)  => s_1_to_2(22 downto 2),
      S               => S(1),
      O               => s_2_to_4
      );

  -- 4-bit shift
  X21M_4 : x_2_to_1_mux_46
    port map (
      A               => s_2_to_4,
      B(47 downto 19) => (others => '0'),
      B(18 downto 0)  => s_2_to_4(22 downto 4),
      S               => S(2),
      O               => s_4_to_8
```



```
      );

  -- 8-bit shift
  X21M_8 : x_2_to_1_mux_46
    port map (
      A               => s_4_to_8,
      B(47 downto 15) => (others => '0'),
      B(14 downto 0)  => s_4_to_8(22 downto 8),
      S               => S(3),
      O               => s_8_to_16
      );

  -- 16-bit shift
  X21M_16 : x_2_to_1_mux_46
    port map (
      A               => s_8_to_16,
      B(47 downto 7)  => (others => '0'),
      B(6 downto 0)   => s_8_to_16(22 downto 16),
      S               => S(4),
      O(47 downto 23) => s_dont_care,
      O(22 downto 0)  => O
      );

end arch;
```

```
-- File : sub_8.vhd

library ieee;
use ieee.std_logic_1164.all;

entity sub_8 is
  port (
    A : in  std_logic_vector(7 downto 0);
    B : in  std_logic_vector(7 downto 0);
    D : out std_logic_vector(7 downto 0)
    );
end sub_8;

architecture arch of sub_8 is

  signal B_cmpl : std_logic_vector(7 downto 0);
  signal P      : std_logic_vector(7 downto 0);
  signal G      : std_logic_vector(7 downto 0);
  signal C      : std_logic_vector(7 downto 0);

begin  -- arch

  B_cmpl <= not B;
  P      <= A xor B_cmpl;
  G      <= A and B_cmpl;
  C(0)   <= '1';

  D(7) <= P(7) xor C(7);
  D(6) <= P(6) xor C(6);
  D(5) <= P(5) xor C(5);
  D(4) <= P(4) xor C(4);
  D(3) <= P(3) xor C(3);
  D(2) <= P(2) xor C(2);
  D(1) <= P(1) xor C(1);
  D(0) <= P(0) xor C(0);

  C(1) <= G(0) or
          (P(0));
  C(2) <= G(1) or
          (P(1) and G(0)) or
          (P(1) and P(0));
  C(3) <= G(2) or
          (P(2) and G(1)) or
          (P(2) and P(1) and G(0)) or
          (P(2) and P(1) and P(0));
  C(4) <= G(3) or
          (P(3) and G(2)) or
          (P(3) and P(2) and G(1)) or
          (P(3) and P(2) and P(1) and G(0)) or
          (P(3) and P(2) and P(1) and P(0));
  C(5) <= G(4) or
          (P(4) and G(3)) or
          (P(4) and P(3) and G(2)) or
          (P(4) and P(3) and P(2) and G(1)) or
          (P(4) and P(3) and P(2) and P(1) and G(0)) or
          (P(4) and P(3) and P(2) and P(1) and P(0));
  C(6) <= G(5) or
          (P(5) and G(4)) or
          (P(5) and P(4) and G(3)) or
          (P(5) and P(4) and P(3) and G(2)) or
          (P(5) and P(4) and P(3) and P(2) and G(1)) or
          (P(5) and P(4) and P(3) and P(2) and P(1) and G(0)) or
          (P(5) and P(4) and P(3) and P(2) and P(1) and P(0));
  C(7) <= G(6) or
          (P(6) and G(5)) or
          (P(6) and P(5) and G(4)) or
          (P(6) and P(5) and P(4) and G(3)) or
          (P(6) and P(5) and P(4) and P(3) and G(2)) or
          (P(6) and P(5) and P(4) and P(3) and P(2) and G(1)) or
          (P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and G(0)) or
          (P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and P(0));

end arch;
```

```
-- File : tb_add_16.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity TB_ADDER8 is
end entity TB_ADDER8;

architecture TB_ADDER8 of TB_ADDER8 is

  component add_16 is
    port (
      A : in  std_logic_vector (15 downto 0);
      B : in  std_logic_vector (15 downto 0);
      C : in  std_logic;
      O : out std_logic_vector (15 downto 0);
      G : out std_logic;
      P : out std_logic);
  end component add_16;

  signal X_CIN_H : std_logic;
  signal X_A_VAL : std_logic_vector (15 downto 0);
  signal X_B_VAL : std_logic_vector (15 downto 0);
  signal X_F_OUT : std_logic_vector (15 downto 0);
  signal X_G_H   : std_logic;
  signal X_P_H   : std_logic;

begin

  UUT : ADD_16
    port map (
      C => X_CIN_H,
      A => X_A_VAL,
      B => X_B_VAL,
      O => X_F_OUT,
      G => X_G_H,
      P => X_P_H
      );

  STIMULUS_PROC :
  process
  begin
    X_A_VAL <= X"0000"; X_B_VAL <= X"0000"; X_CIN_H <= '0';
    wait for 100 ns;
    X_A_VAL <= X"0000"; X_B_VAL <= X"0000"; X_CIN_H <= '1';
    wait for 100 ns;
    X_A_VAL <= X"00FF"; X_B_VAL <= X"00FF"; X_CIN_H <= '0';
    wait for 100 ns;
    X_A_VAL <= X"00FF"; X_B_VAL <= X"00FF"; X_CIN_H <= '1';
    wait for 100 ns;
    X_A_VAL <= X"0055"; X_B_VAL <= X"00AA"; X_CIN_H <= '0';
    wait for 100 ns;
    X_A_VAL <= X"0055"; X_B_VAL <= X"00AA"; X_CIN_H <= '1';
    wait for 100 ns;
    X_A_VAL <= X"0355"; X_B_VAL <= X"01AA"; X_CIN_H <= '1';
    wait for 100 ns;
    X_A_VAL <= X"FFFF"; X_B_VAL <= X"FFFF"; X_CIN_H <= '0';
    wait for 100 ns;
  end process;

end architecture TB_ADDER8;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use STD.TEXTIO.all;
use IEEE.STD_LOGIC_TEXTIO.all;

entity TB_FP_MULT is
end entity TB_FP_MULT;

use WORK.all;

architecture TB_FP_MULT of TB_FP_MULT is

component FP_MULT is
  port (
    A_H : in  STD_LOGIC_VECTOR ( 31 downto 0 );
    B_H : in  STD_LOGIC_VECTOR ( 31 downto 0 );
    O_H : out STD_LOGIC_VECTOR ( 31 downto 0 )
  );
end component FP_MULT;

signal A_INPUT    : STD_LOGIC_VECTOR ( 31 downto 0 );
signal B_INPUT    : STD_LOGIC_VECTOR ( 31 downto 0 );
signal F_DETAILED : STD_LOGIC_VECTOR ( 31 downto 0 );

alias A_EXP : STD_LOGIC_VECTOR ( 7 downto 0 ) is A_INPUT ( 30 downto 23 );
alias B_EXP : STD_LOGIC_VECTOR ( 7 downto 0 ) is B_INPUT ( 30 downto 23 );

signal A_EXP_INT : INTEGER;
signal B_EXP_INT : INTEGER;
signal R_EXP_INT : INTEGER;

signal CLOCK : STD_LOGIC;
signal CYCLE : INTEGER := 0;

begin
UUT_DETAILED: FP_MULT
  port map (
    A_H => A_INPUT,
    B_H => B_INPUT,
    O_H => F_DETAILED
  );

CYCLE_PROC:
  process
  begin
    CLOCK <= '1';
    wait for 100 ns;
    CLOCK <= '0';
    wait for 100 ns;
    CYCLE <= CYCLE + 1;
  end process;

  A_EXP_INT <= CONV_INTEGER ( A_EXP ) - 127 ;
  B_EXP_INT <= CONV_INTEGER ( B_EXP ) - 127 ;
  R_EXP_INT <= A_EXP_INT + B_EXP_INT;

DATA_PROC:
  process ( CLOCK ) is
  type DATA_STOR is array ( 0 to 31 ) of STD_LOGIC_VECTOR ( 31 downto 0 );
  constant A_VALS : DATA_STOR := (
    X"3D800000", X"3F800000", X"BF800000", X"3F8F0000",  --  0,  1,  2,  3
```



```
    X"BF800000", X"3F800000", X"3F000000", X"3F890000",  --  4,  5,  6,  7
    X"BF000000", X"42C80000", X"42C80000", X"C2C80000",  --  8,  9, 10, 11
    X"41D80000", X"C1D80000", X"4620F800", X"4620F800",  -- 12, 13, 14, 15
    X"49791900", X"49791900", X"42CA0000", X"C1D80000",  -- 16, 17, 18, 19
    X"C2CE0000", X"42CE0000", X"497E07A0", X"C97E07A0",  -- 20, 21, 22, 23
    X"1F0AC723", X"269117C6", X"269FF7C6", X"5F0AC723",  -- 24, 25, 26, 27
    X"733A4000", X"E85B79A2", X"41700000", X"4640E400"   -- 28, 29, 30, 31
  );
  constant B_VALS : DATA_STOR := (
    X"3D000000", X"3F800000", X"3F800000", X"BF800000",
    X"BF800000", X"3F000000", X"3F870000", X"BF00A000",
    X"3F800000", X"3F800000", X"BF800000", X"C0400000",
    X"43110000", X"43110000", X"3E800000", X"BE800000",
    X"42CC0000", X"C2CC0000", X"49791900", X"49FCB8F0",
    X"448FC000", X"C481C000", X"497E07B0", X"497E07B0",
    X"1E9FF7C6", X"5F0AC723", X"269FF7C6", X"5F0AC723",
    X"685B79A2", X"733A4000", X"43988000", X"45D42800"
  );
  variable FIRST : BOOLEAN := TRUE;

  variable INDEX : INTEGER := 0;

  begin
    if RISING_EDGE ( CLOCK ) or (FIRST = TRUE ) then
      INDEX := CYCLE mod 32;
      A_INPUT <= A_VALS(INDEX);
      B_INPUT <= B_VALS(INDEX);
      FIRST := FALSE;
    end if;
  end process;


RECORDING_PROC:
  process ( CLOCK ) is
  file OUT_FILE: TEXT open WRITE_MODE is "outputvals.txt";
  variable BUF : LINE;                          -- set BUF up to send line
  constant STR : STRING ( 1 to 3 ) := " ";
  variable TM  : TIME;
  begin
    if CLOCK'EVENT and CLOCK = '1' and CYCLE > 1 then -- if rising edge
      WRITE  ( BUF, CYCLE );
      WRITE  ( BUF, STR );
      HWRITE ( BUF, A_INPUT );
      WRITE  ( BUF, STR);
      HWRITE ( BUF, B_INPUT );
      WRITE  ( BUF, STR);
      WRITE  ( BUF, STR);
      HWRITE ( BUF, F_DETAILED );
      WRITE  ( BUF, STR);
      WRITE  ( BUF, NOW);
      WRITE  ( BUF, STR);
      TM := F_DETAILED'LAST_EVENT;
      WRITE  ( BUF, TM );
      WRITELINE (OUT_FILE, BUF);            -- and send to file
    end if;
  end process;

end TB_FP_MULT;
```

```
-- File : tb_right_shifter.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity tb_right_shifter is
end entity tb_right_shifter;

architecture tb_right_shifter of tb_right_shifter is

  component right_shifter is
    port (
      A : in  std_logic_vector(22 downto 0);
      S : in  std_logic_vector(4 downto 0);
      O : out std_logic_vector(22 downto 0)
      );
  end component right_shifter;

  signal X_A_VAL : std_logic_vector(22 downto 0);
  signal X_S_VAL : std_logic_vector(4 downto 0);
  signal X_O_OUT : std_logic_vector(22 downto 0);

begin
  UUT : right_shifter
    port map (
      A => X_A_VAL,
      S => X_S_VAL,
      O => X_O_OUT);

-- 01_0001_1100_1011_0100_1101
-- 11_1110_1111_1111_1110_0100
-- 10_0101_0101_0110_0001_1111
-- 10_1100_0101_1101_0001_0100

  STIMULUS_PROC :
  process
  begin
    X_A_VAL <= B"000_0101_0100_0010_1000_0001";
    X_S_VAL <= B"0_0001";
    wait for 100 ns;

    X_A_VAL <= B"000_0000_0000_0000_0001_0111";
    X_S_VAL <= B"0_0010";
    wait for 100 ns;

    X_A_VAL <= B"000_1000_1000_1000_0001_1010";
    X_S_VAL <= B"0_0100";
    wait for 100 ns;

    X_A_VAL <= B"010_0001_1100_1001_1111_0101";
    X_S_VAL <= B"0_1000";
    wait for 100 ns;

    X_A_VAL <= B"101_1101_1010_1011_1010_0000";
    X_S_VAL <= B"1_0000";
    wait for 100 ns;

    X_A_VAL <= B"010_1100_0001_0101_0101_0001";
    X_S_VAL <= B"0_0011";
    wait for 100 ns;

    X_A_VAL <= B"010_0111_0010_1001_1111_0000";
```



```
    X_S_VAL <= B"0_0101";
    wait for 100 ns;

    X_A_VAL <= B"001_0000_0001_0110_0100_1111";
    X_S_VAL <= B"0_0110";
    wait for 100 ns;
  end process;

end architecture tb_right_shifter;
```

```
-- File : x_2_to_1_mux_46.vhd

library ieee;
use ieee.std_logic_1164.all;

entity x_2_to_1_mux_46 is

  port (
    A : in  std_logic_vector(47 downto 0);
    B : in  std_logic_vector(47 downto 0);
    S : in  std_logic;
    O : out std_logic_vector(47 downto 0)
    );

end x_2_to_1_mux_46;

architecture arch of x_2_to_1_mux_46 is

begin  -- arch

  with S select
    O <=
    A               when '0',
    B               when '1',
    (others => 'X') when others;

end arch;
```

```
-- File : x_2_to_1_mux.vhd

library ieee;
use ieee.std_logic_1164.all;

entity x_2_to_1_mux is
  port (
    A : in  std_logic_vector(47 downto 0);
    B : in  std_logic_vector(47 downto 0);
    S : in  std_logic;
    O : out std_logic_vector(47 downto 0)
    );
end x_2_to_1_mux;

architecture arch of x_2_to_1_mux is

begin  -- arch

  with S select
    O <=
    A               when '0',
    B               when '1',
    (others => 'X') when others;

end arch;
```