```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.numeric_std.ALL;
ENTITY filter IS
  PORT( clk                       :  IN    std_logic;
        clk_enable                :  IN    std_logic;
        reset                     :  IN    std_logic;
        filter_in                 :  IN    std_logic_vector(15 DOWNTO 0); -- sfix16_En15
        filter_out                :  OUT   std_logic_vector(15 DOWNTO 0)  -- sfix16_En15
        );

END filter;



-------------------------------------------------------------
--Module Architecture: filter
-------------------------------------------------------------
ARCHITECTURE rtl OF filter IS
  -- Local Functions
  -- Type Definitions
  TYPE delay_pipeline_type IS ARRAY (NATURAL range <>) OF signed(15 DOWNTO 0); -- sfix16_En15
  -- Constants
  CONSTANT coeff1                   : signed(15 DOWNTO 0) := to_signed(32766, 16); -- sfix16_En15
  CONSTANT coeff2                   : signed(15 DOWNTO 0) := to_signed(-31130, 16); -- sfix16_En15
```

```
  -- Signals
  SIGNAL delay_pipeline               : delay_pipeline_type(0 TO 1); -- sfix16_En15
  SIGNAL product2                     : signed(30 DOWNTO 0); -- sfix31_En30
  SIGNAL mul_temp                     : signed(31 DOWNTO 0); -- sfix32_En30
  SIGNAL product1_cast                : signed(31 DOWNTO 0); -- sfix32_En30
  SIGNAL product1                     : signed(30 DOWNTO 0); -- sfix31_En30
  SIGNAL mul_temp_1                   : signed(31 DOWNTO 0); -- sfix32_En30
  SIGNAL sum1                         : signed(31 DOWNTO 0); -- sfix32_En30
  SIGNAL add_temp                     : signed(32 DOWNTO 0); -- sfix33_En30
  SIGNAL output_typeconvert           : signed(15 DOWNTO 0); -- sfix16_En15
  SIGNAL output_register              : signed(15 DOWNTO 0); -- sfix16_En15


BEGIN


  -- Block Statements
  Delay_Pipeline_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      delay_pipeline(0 TO 1) <= (OTHERS => (OTHERS => '0'));
    ELSIF rising_edge(clk) THEN
      IF clk_enable = '1' THEN
        delay_pipeline(0) <= signed(filter_in);
        delay_pipeline(1) <= delay_pipeline(0);
```

```
    END IF;
  END IF;
END PROCESS Delay_Pipeline_process;



mul_temp <= delay_pipeline(1) * coeff2*16;
product2 <= mul_temp(30 DOWNTO 0);

product1_cast <= resize(product1, 32);

mul_temp_1 <= delay_pipeline(0) * coeff1*16;
product1 <= mul_temp_1(30 DOWNTO 0);

add_temp <= resize(product1_cast, 33) + resize(product2, 33);
sum1 <= add_temp(31 DOWNTO 0);

output_typeconvert <= resize(shift_right(sum1(30 DOWNTO 0) + ( "0" & (sum1(15) & NOT sum1(15) &
NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) &
NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) & NOT sum1(15) &
NOT sum1(15))), 15), 16);

Output_Register_process : PROCESS (clk, reset)
BEGIN
  IF reset = '1' THEN
    output_register <= (OTHERS => '0');
  ELSIF rising_edge(clk) THEN
    IF clk_enable = '1' THEN
```

```
      output_register <= output_typeconvert+32768;
    END IF;
  END IF;
END PROCESS Output_Register_process;

-- Assignment Statements
filter_out <= std_logic_vector(output_register);
END rtl;
```