```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity pilot_detector is
    Port (
        -- Clock and reset
        clk          : in  std_logic;                     -- Clock signal
        resetn       : in  std_logic;                     -- Active-low reset

        -- AXI4-Stream Input (from ADC)
        s_axis_tdata  : in  std_logic_vector(31 downto 0); -- Input data (14 bits)
        s_axis_tvalid : in  std_logic;                     -- Input valid signal
        s_axis_tready : out std_logic;                     -- Input ready signal

        -- AXI4-Stream Output (to FIFO)
        m_axis_tdata  : out std_logic_vector(31 downto 0); -- Output data (16 bits,
padded)
        m_axis_tvalid : out std_logic;                     -- Output valid signal
        m_axis_tready : in  std_logic;                     -- Output ready signal
        threshold     : in std_logic_vector(31 downto 0);
        -- Status Outputs
        pilot_found    : out std_logic;                 -- High when pilot is found
        correlation_out : out signed(31 downto 0)
    );
end pilot_detector;

architecture Behavioral of pilot_detector is

    -- Parameters
    constant PILOT_LENGTH : integer := 7;
    constant DATA_BUF_SIZE : integer := 5 ;
    -- Type definitions
    type pilot_sequence_type is array (0 to PILOT_LENGTH-1) of signed(13 downto 0);
    type shift_reg_type     is array (0 to PILOT_LENGTH-1) of signed(13 downto 0);
     type data_reg_type is array (0 to DATA_BUF_SIZE-1) of signed(31 downto 0);
    type product_array_type is array (0 to 6) of signed(31 downto 0);
    type sum_stage1_type    is array (0 to 3) of signed(31 downto 0);
    type sum_stage2_type    is array (0 to 1) of signed(31 downto 0);

    -- Constants
    constant PILOT_SEQUENCE : pilot_sequence_type := (
        to_signed(8191, 14), to_signed(8191, 14), to_signed(8191, 14),
        to_signed(-8192, 14), to_signed(-8192, 14), to_signed(8191, 14),
        to_signed(-8192, 14)
    );

    -- Signals
    signal shift_reg      : shift_reg_type     := (others => (others => '0'));
    signal products       : product_array_type := (others => (others => '0'));
    signal sum_stage1     : sum_stage1_type    := (others => (others => '0'));
    signal sum_stage2     : sum_stage2_type    := (others => (others => '0'));
    signal sum_stage3     : signed(31 downto 0) := (others => '0');
    signal correlation    : signed(31 downto 0) := (others => '0');
    signal found          : std_logic := '0';
    signal forward_data   : std_logic := '0';
    signal threshold_int  : integer;
    signal data_buffer : data_reg_type := (others => (others => '0'));
    signal real_data : std_logic_vector(13 downto 0);
```

```
begin

    threshold_int <= to_integer(signed(threshold));  -- convert threshold to
integer
    real_data <= s_axis_tdata(31 downto 18);

    -- Shift Register
    process(clk)
    begin
        if rising_edge(clk) then
            if resetn = '0' then
                shift_reg <= (others => (others => '0'));
            elsif s_axis_tvalid = '1' and found = '0' then
                shift_reg <= shift_reg(1 to PILOT_LENGTH-1) & signed(real_data);
            end if;
        end if;
    end process;

  process(clk)
        begin
            if rising_edge(clk) then
                if resetn = '0' then
                    data_buffer <= (others => (others => '0'));
                elsif s_axis_tvalid = '1'  then
                    -- Shift new sample into the register (only if pilot not yet
found)
                    data_buffer <= data_buffer(1 to DATA_BUF_SIZE-1) &
signed(s_axis_tdata);
                end if;
            end if;
    end process;
    ------------------------------------------------------------------------
    -- Pipelined Correlation
    ------------------------------------------------------------------------

    -- Stage 0: Multiply input samples with pilot sequence
    process(clk)
    begin
        if rising_edge(clk) then
            if resetn = '0' then
                products <= (others => (others => '0'));
            else
                for i in 0 to 6 loop
                    products(i) <= resize(shift_reg(i) * PILOT_SEQUENCE(i), 32);
                end loop;
            end if;
        end if;
    end process;

    -- Stage 1: First level of addition (3 pairs + 1 leftover)
    process(clk)
    begin
        if rising_edge(clk) then
            if resetn = '0' then
                sum_stage1 <= (others => (others => '0'));
            else
                sum_stage1(0) <= products(0) + products(1);
                sum_stage1(1) <= products(2) + products(3);
```

```
            sum_stage1(2) <= products(4) + products(5);
            sum_stage1(3) <= products(6);  -- carry forward
        end if;
    end if;
end process;

-- Stage 2: Second level of addition (2 pairs)
process(clk)
begin
    if rising_edge(clk) then
        if resetn = '0' then
            sum_stage2 <= (others => (others => '0'));
        else
            sum_stage2(0) <= sum_stage1(0) + sum_stage1(1);
            sum_stage2(1) <= sum_stage1(2) + sum_stage1(3);
        end if;
    end if;
end process;

-- Stage 3: Third level of addition (final sum)
process(clk)
begin
    if rising_edge(clk) then
        if resetn = '0' then
            sum_stage3 <= (others => '0');
        else
            sum_stage3 <= sum_stage2(0) + sum_stage2(1);
        end if;
    end if;
end process;

-- Stage 4: Register the result
process(clk)
begin
    if rising_edge(clk) then
        if resetn = '0' then
            correlation <= (others => '0');
        elsif found = '0' then
            correlation <= sum_stage3;
        end if;
    end if;
end process;

------------------------------------------------------------------------
-- Detection Logic
------------------------------------------------------------------------

process(clk)
begin
    if rising_edge(clk) then
        if resetn = '0' then
            found <= '0';
            forward_data <= '0';
        elsif found = '0' then
            if correlation > to_signed(threshold_int, 32) then
                found <= '1';
                forward_data <= '1';
            end if;
        end if;
```

```
        end if;
    end process;

    -------------------------------------------------------------------------
    -- Output Forwarding
    -------------------------------------------------------------------------

    process(clk)
    begin
        if rising_edge(clk) then
            if resetn = '0' then
                m_axis_tdata <= (others => '0');
                m_axis_tvalid <= '0';
            elsif s_axis_tvalid = '1' and forward_data = '1' then
             m_axis_tdata <=
std_logic_vector(resize(signed(data_buffer(DATA_BUF_SIZE-4)), 32));
                m_axis_tvalid <= '1';
            else
                m_axis_tvalid <= '0';
            end if;
        end if;
    end process;

    -------------------------------------------------------------------------
    -- Assign Outputs
    -------------------------------------------------------------------------

    pilot_found     <= found;
    correlation_out <= correlation;
    s_axis_tready   <= '1'; -- Always ready

end Behavioral;
```