EE26 Lab 4 Tutorial

Zengxu Yang

March 9, 2020

1 Use Floating Point IP

1. In the Vivado application menu, click `Window` → `IP Catalog` to open the `IP Catalog` panel.

2. Enter `float` in the `Search` field to find the `Floating-point` IP, as shown in Figure 1.

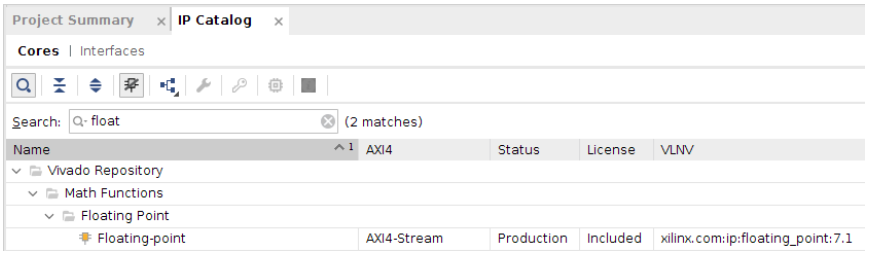

Figure 1: IP Catalog

3. Double click on the `Floating-point` IP to open `Customize IP` dialog. In `Operation Selection`, choose `Add/Subtract`; in `Add/Subtract and Multiply-Add Operator options` choose `Both`; in `Component`

`Name`, change the name to be `fp_adder_subtractor` as shown in Figure 2. Leave other configurations to their default values. Click `OK`, `Generate` to add the `fp_adder_subtractor` IP to your project design sources. By default, this IP uses 32 bit IEEE 754 single precision

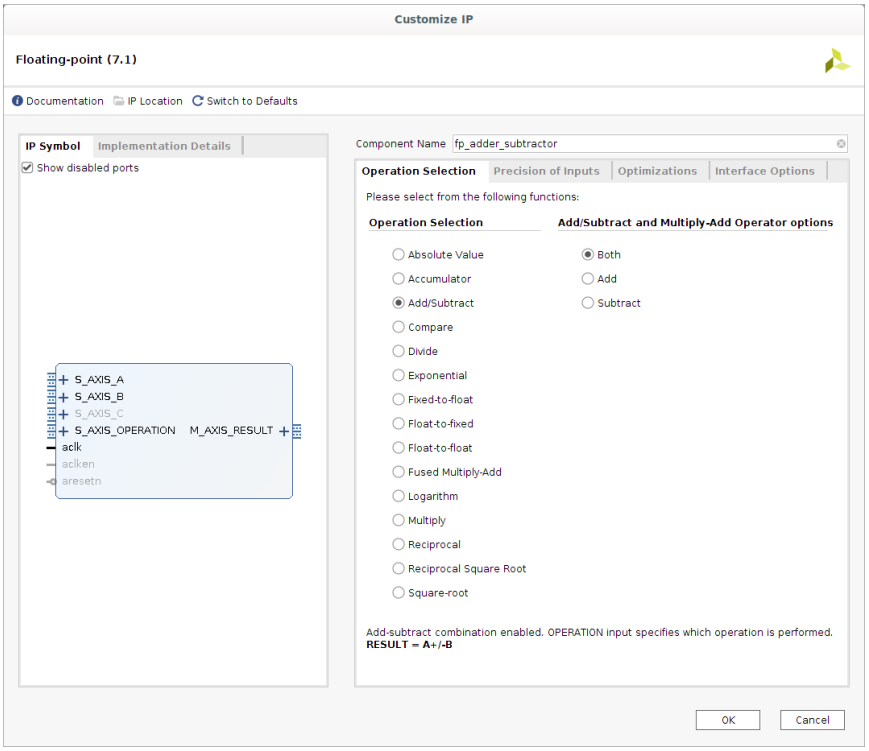

Figure 2: Customize IP

floating-point numbers and one clock cycle fully parallel circuits.

4. Repeat the last step, but change `Operation Selection` to `Multiply` and `Component Name` to `fp_multiplier`.

5. In `Sources` panel, click to expand the added `fp_adder_subtractor` IP. When asked, click `OK`.

6. In the expanded sources, double click on the `fp_adder_subtractor.vhd` file to open it, as shown in Figure 3.

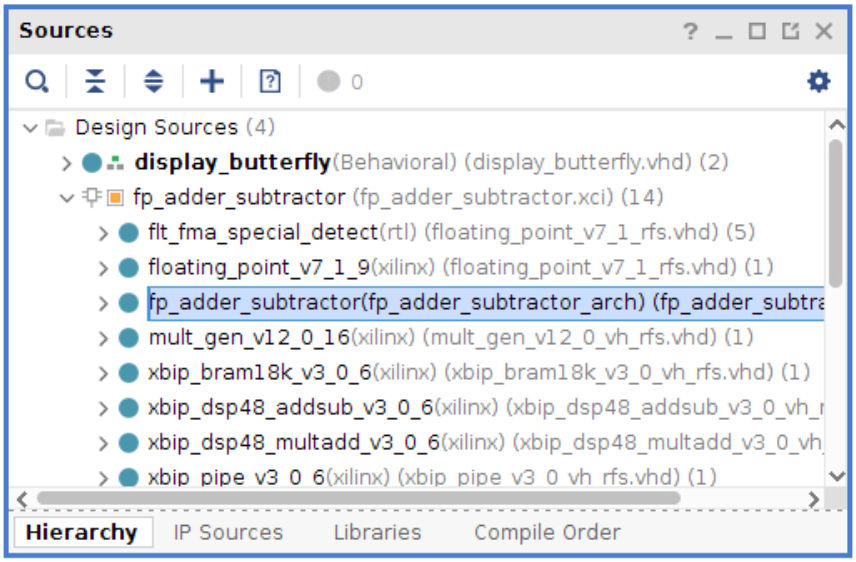

Figure 3: Floating-point IP

7. Do the same for `fp_multiplier` IP to open the `fp_multiplier.vhd` file.

In your butterfly design, you use these two Floating-point IP components to perform floating-point additions, subtractions, and multiplications. Check the entity of the `fp_adder_subtractor.vhd` file.

```
ENTITY fp_adder_subtractor IS
  PORT (
    aclk : IN STD_LOGIC;
    s_axis_a_tvalid : IN STD_LOGIC;
```

```
    s_axis_a_tready : OUT STD_LOGIC;
    s_axis_a_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    s_axis_b_tvalid : IN STD_LOGIC;
    s_axis_b_tready : OUT STD_LOGIC;
    s_axis_b_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    s_axis_operation_tvalid : IN STD_LOGIC;
    s_axis_operation_tready : OUT STD_LOGIC;
    s_axis_operation_tdata : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    m_axis_result_tvalid : OUT STD_LOGIC;
    m_axis_result_tready : IN STD_LOGIC;
    m_axis_result_tdata : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END fp_adder_subtractor;
```

We only need to care about `s_axis_a_tdata`, `s_axis_b_tdata`, `s_axis_operation_tdata`, and `m_axis_result_tdata`.

2 Interfaces

We need to edit our butterfly source code to use the floating-point IP. We change the `complex` record to be:

```
type complex is
    record
        r: std_logic_vector(31 downto 0);
```

```
        i: std_logic_vector(31 downto 0);
    end record;
```

in the package file.

Assume that `r` is a 32 bit vector, change the `to_float` to something like this to assign a floating-point value to it:

```
r <= std_logic_vector(to_float(0.5, 8, 23));
```

3 Array Type

You can create an array type in VHDL to simplify your design. For example:

```
type num_array is array (3 downto 0) of std_logic_vector(31 downto 0);
```

creates a 4 element arrary with each element of type `std_logic_vector`.

4 Generate Statements

You can use the `for ... generate` statements to simplify your design. A `generate` statement can map multiple components. This shortens your code if you have multiple components to use. For example:

```
addition_gen:
for i in 0 to 3 generate
    addition: fp_adder_subtractor port map (aclk => clk,
```

```
 s_axis_a_tvalid => '0',
 s_axis_a_tready => open,
 s_axis_a_tdata => add_in1(i),
  s_axis_b_tvalid => '0',
 s_axis_b_tready => open,
 s_axis_b_tdata => add_in2(i),
 s_axis_operation_tvalid => '0',
 s_axis_operation_tready => open,
  s_axis_operation_tdata => "00000000",
  m_axis_result_tvalid => open,
 m_axis_result_tready => '0',
   m_axis_result_tdata => add_out(i));
end generate addition_gen;
```

This block maps 4 adder/subtractor components with their inputs and outputs of array types.

5 Butterfly Structure Test

$$A = 1.5 + j0.5$$

$$B = 2.0 - j3.0$$

$$W = 0.5 + j0.5$$

$$A' = 4.0$$

$$B' = -1 + j$$

Using IEEE 754 representations, we get

```
A=(0x3fc00000, 0x3f000000)
B=(0x40000000, 0xc0400000)
W=(0x3f000000, 0x3f000000)
A'=(0x40800000, 0x00000000)
B'=(0xbf800000, 0x3f800000)
```

It has been proven that the floating-point IP is synthesizable. Please check whether you get the correct results by 7-segment displays. I uploaded a reference butterfly design file on Canvas.

6 8 Point FFT

Please check out Figure 4 for your basic FFT design. The weights W_8^k are calculated as:

$$W_8^k = \cos\left(\frac{2\pi k}{8}\right) - j\sin\left(\frac{2\pi k}{8}\right)$$

where $k = 0, 1, 2, 3$.

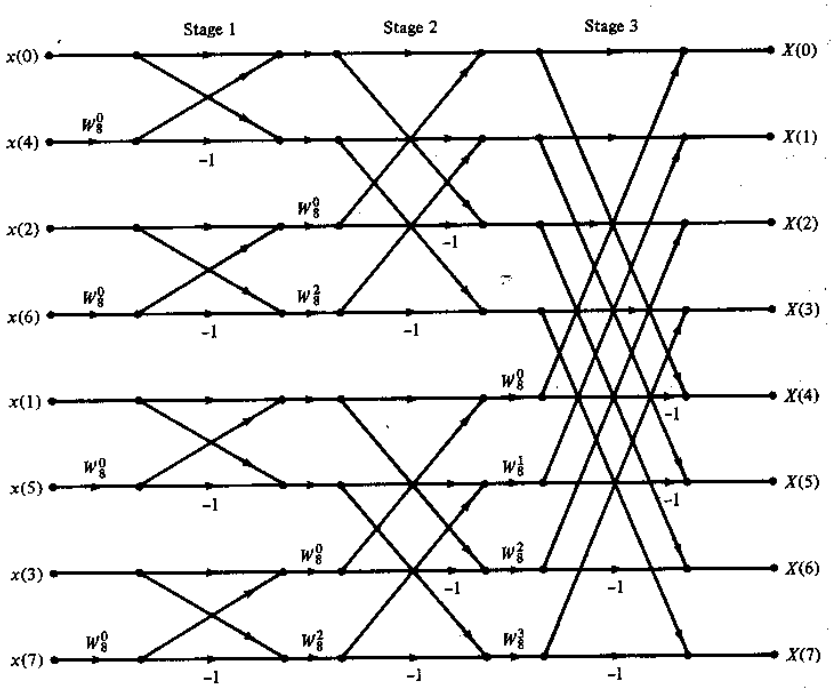

Figure 4: 8 Point FFT