EE26 Lab 4 Optimization

Zengxu Yang

March 31, 2020

As we use the Xilinx Floating-point IP in our FFT engine, we encounter the problem of insufficient resources.

By default, each butterfly structure uses 12 adders, subtractors, or multipliers. Each adder, subtractor, or multiplier uses 2 DSP slices. So each butterfly structure uses 24 DSP slices.

We use 12 butterfly structures in our 8-point FFT engine. This gives 288 DSP slices. A Nexys 4 board we use has only 240 DSP slices, resulting in insufficient resources, as shown in Figure 1:

Resource	Utilization	Available	Utilization %
LUT	30971	63400	48.85
LUTRAM	2016	19000	10.61
FF	58511	126800	46.14
DSP	288	240	120.00
IO	21	210	10.00

Figure 1: Original Synthesis Utilization

We can optimize our design and lower the utilization.

1. Go to `Sources` panel then `IP Sources`. Double click on `fp_adder_subtractor`. This bring up a `Re-customize IP` dialog.

2. Go to `Optimization` and change `DSP Slice Usage` from `Full Usage` to `Medium Usage`, click `OK` then click `Generate`. See Figure 2:

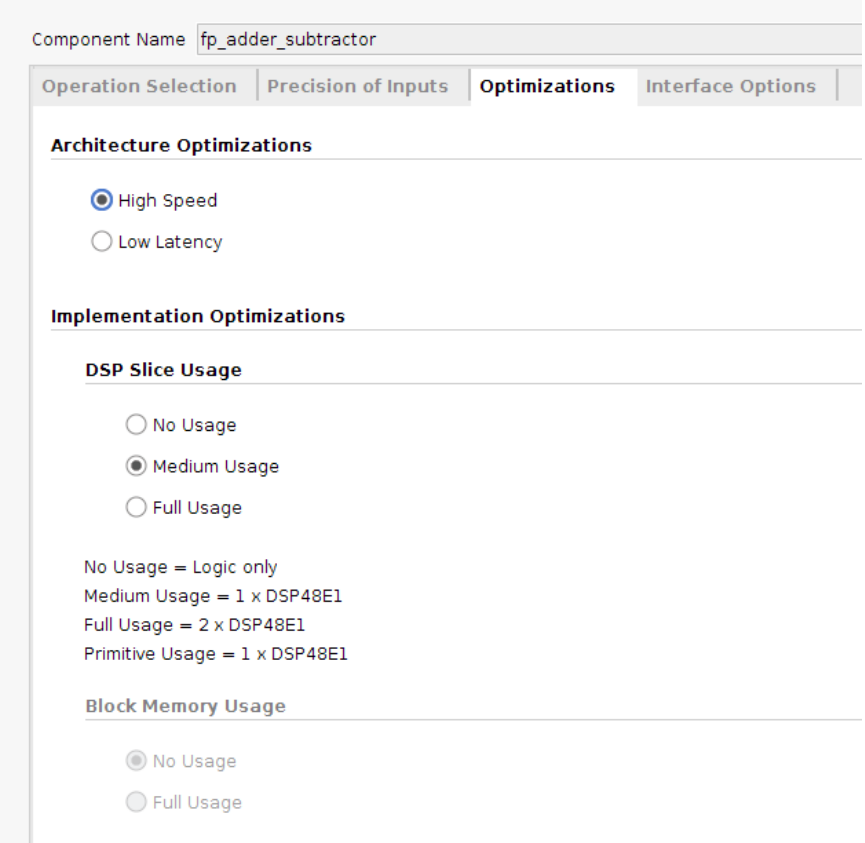

Figure 2: Optimization

3. Do the same to `fp_multiplier`.

4. Click `Run Synthesis` again. This reduces DSP slices to 144, as shown in Figure 3:

Resource	Utilization	Available	Utilization %
LUT	43307	63400	68.31
LUTRAM	2064	19000	10.86
FF	79391	126800	62.61
DSP	144	240	60.00
IO	21	210	10.00

Figure 3: Optimized Synthesis Utilization

5. Optionally, you can change the Floating-point IP to nonblocking without delays. This makes them pure combinational circuits and eliminates clocks.

6. Repeat step 1 and then go to `Interface Options`, in `Flow Control Options` change `Flow Control` to `NonBlocking`, then uncheck `Use Maximum Latency` and change `Latency` to 0. See Figure 4:

7. Do the same to `fp_multiplier`.

8. Note that the interface of the Floating-point IP has changed. The new inteface uses less signals. You need to edit your code for the new interface. You just need to remove some signals.

```
COMPONENT fp_adder_subtractor
  PORT (
    s_axis_a_tvalid : IN STD_LOGIC;
    s_axis_a_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    s_axis_b_tvalid : IN STD_LOGIC;
    s_axis_b_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
```

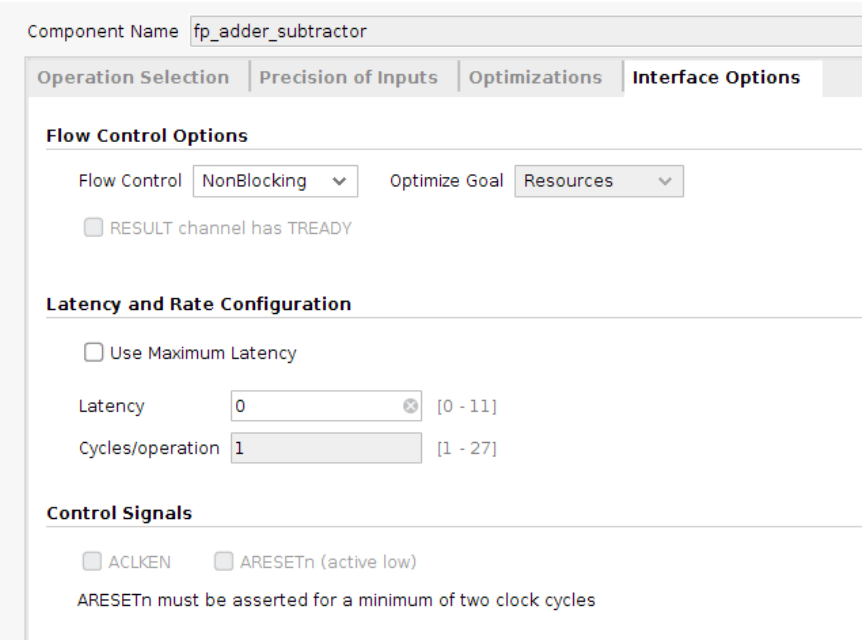

Figure 4: Optimization

```
    s_axis_operation_tvalid : IN STD_LOGIC;
    s_axis_operation_tdata : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    m_axis_result_tvalid : OUT STD_LOGIC;
    m_axis_result_tdata : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

COMPONENT fp_multiplier
  PORT (
    s_axis_a_tvalid : IN STD_LOGIC;
    s_axis_a_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    s_axis_b_tvalid : IN STD_LOGIC;
```

```
    s_axis_b_tdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    m_axis_result_tvalid : OUT STD_LOGIC;
    m_axis_result_tdata : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;
```

9. Click `Run Synthesis` again. This uses even less resources, as shown in Figure 5:

Resource	Utilization	Available	Utilization %
LUT	39275	63400	61.95
FF	47	126800	0.04
DSP	144	240	60.00
IO	21	210	10.00

Figure 5: Further Optimized Synthesis Utilization