PLL Experiment in Vivado

Technical Email: alinx@aithtech.com Sales Email: rachel.zhou@alinx.com

1 Document Introduction

Many beginners have doubts when they only have one 200Mhz clock input on the board. What if you want to work at 100Mhz or 150Mhz? In fact, PLLs are integrated in many FPGA chips. Other manufacturers may not call PLLs, but they also have similar functional modules. The PLL can multiply the frequency and generate many other clocks. This experiment learns the use of PLL and the IP core usage of vivado by calling PLL IP core.

2 Experiment Environment

- Windows 7 SP1 64 bit
- vivado(vivado2017.4)
- ALINX Brand FPGA Development Board (AXKU040 FPGA Development Board)
- Oscilloscope

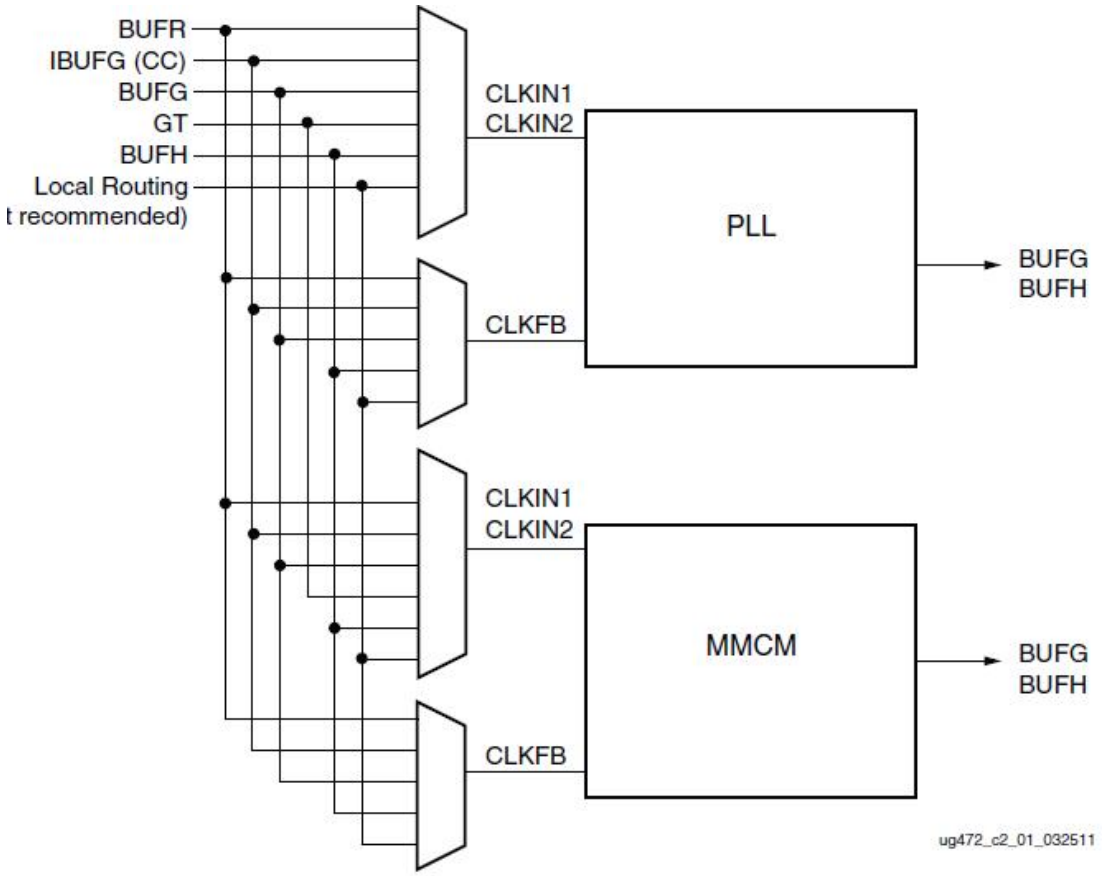
3 Experiment Principle

PLL (phase-locked loop) is an important resource in FPGA. Since a complex FPGA system often requires multiple different frequencies, phase clock signals. Therefore, the number of PLLs in an FPGA chip is an important indicator to measure the capabilities of the FPGA chip. In FPGA design, the high-speed FPGA design of the clock system is extremely important. A low-jitter, low-latency system clock increases the success rate of the FPGA design.

This experiment will use the PLL to output a square wave to the expansion port J16 on the AX7325 development board to demonstrate the method of using PLL in Vivado software.

The 7 family of FPGAs uses dedicated global, regional IO and clock resources to manage the various clocking needs in the design. Clock Management Tiles (CMT) provides clock frequency synthesis, deskew, and jitter filtering.

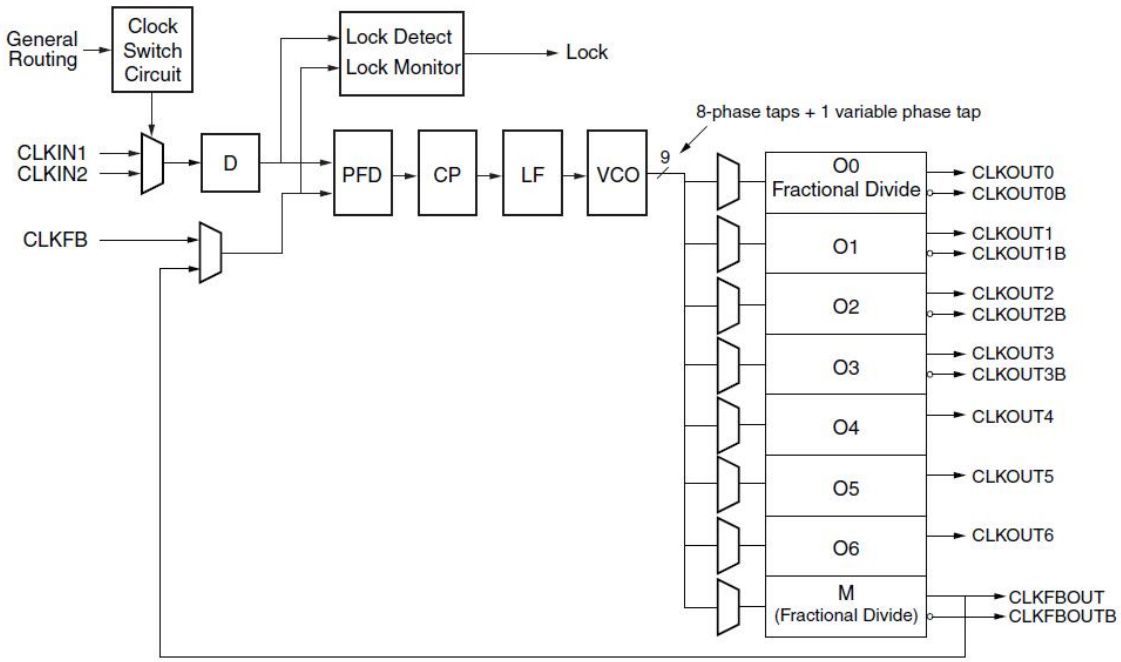
Each CMT includes an MMCM (mixed-mode clock manager) and a PLL. As shown in the figure below, the input of CMT can be BUFR, IBUFG, BUFG, GT, BUFH, local wiring (not recommended), and the output needs to be connected to BUFG or BUFH before use.



- Mixed Mode Clock Manager (MMCM)

The MMCM is used to generate different clock signals with a set phase and frequency relationship with a given input clock. MMCM provides extensive and powerful clock management capabilities

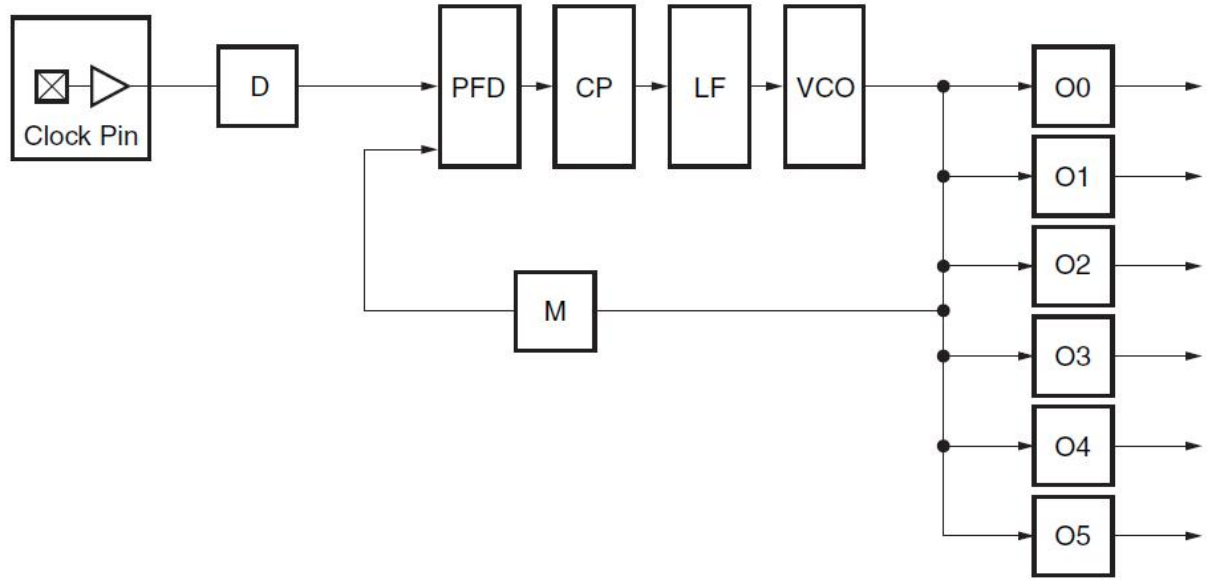
The functional block diagram inside MMCM is shown below:



➢ Digital phase locked loop (PLL)

Phase-locked loop (PLL) is mainly used for frequency synthesis. Multiple PLLs can be used to generate multiple clock signals from an input clock signal.

The functional block diagram inside the PLL is shown below:



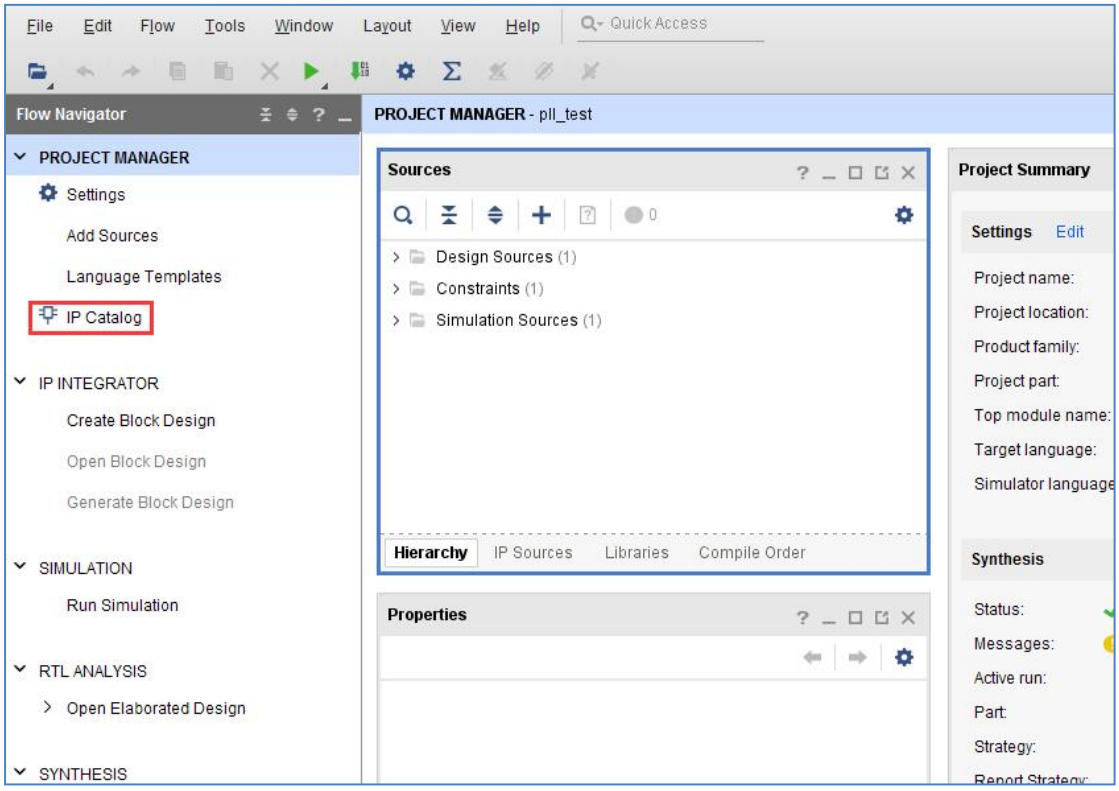
For more information on clock resources, I suggest you check out the "7 Series FPGAs Clocking Resources User Guide" from Xilinx.

4 Create a Project

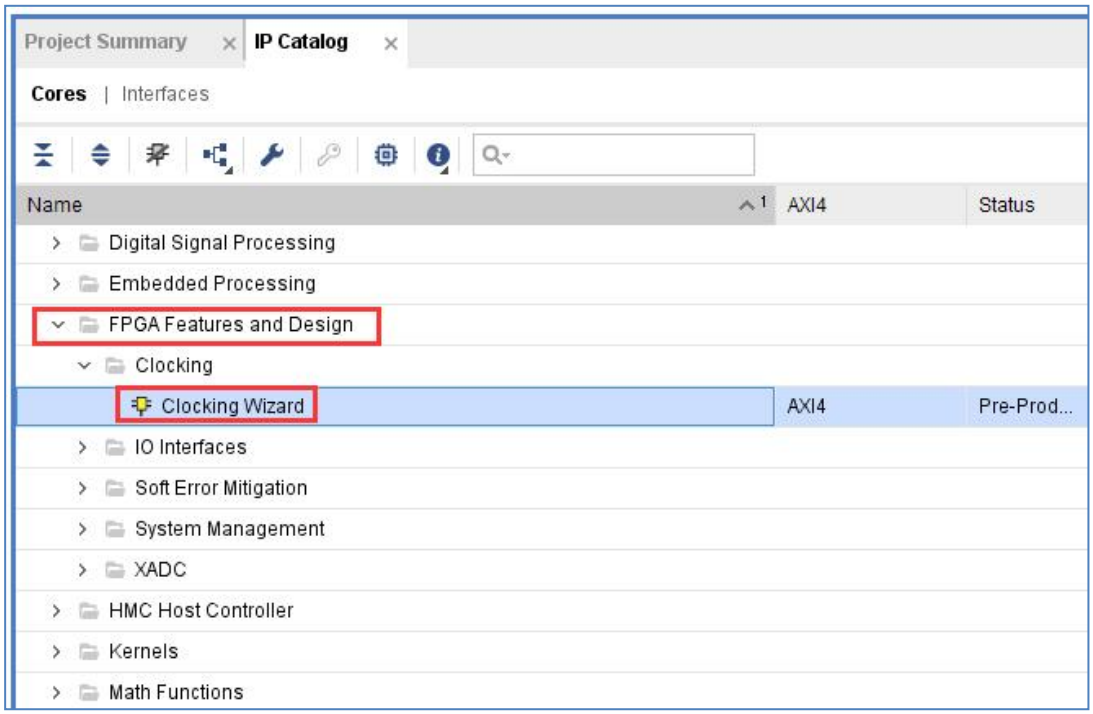
In this experiment, we will demonstrate that if you call the PLL IP core provided by Xilinx to generate clocks of different frequencies, and output one of the clocks to the external IO of the FPGA, that is, PIN76 of AXKU040 FPGA development board J16B

The following are the detailed steps of the program design.

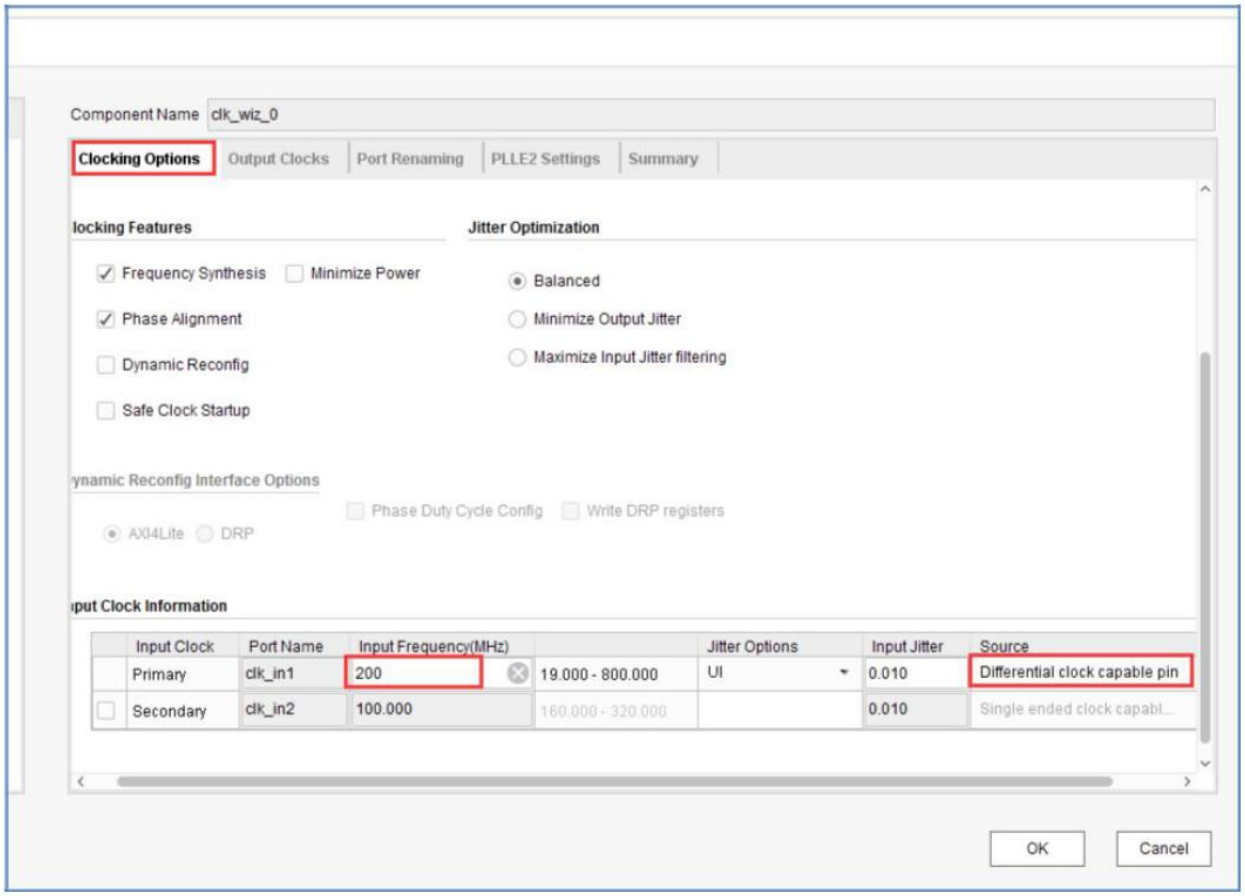
1. Create a new “pll_test” project and click “IP Catalog” under the “ Project Manager “ interface.



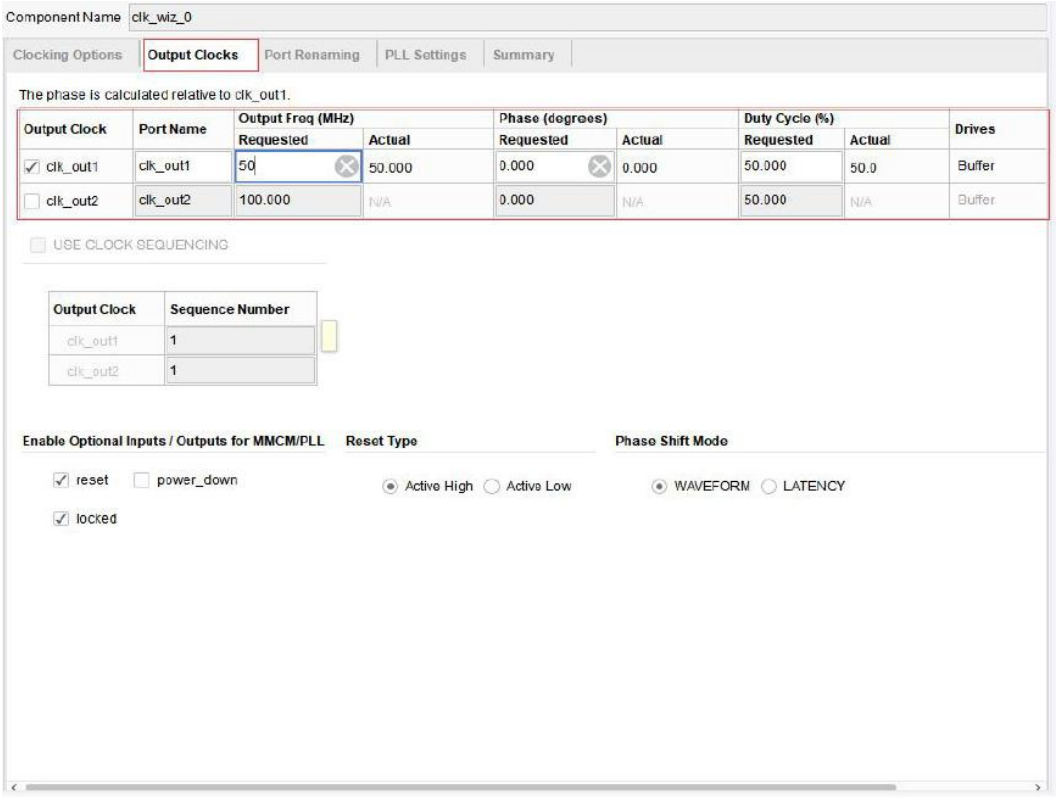
2. Then select the “Clocking Wizard” under “FPGA Features and Design\Clocking” in the “IP Catalog” interface, and double-click to open the configuration interface.



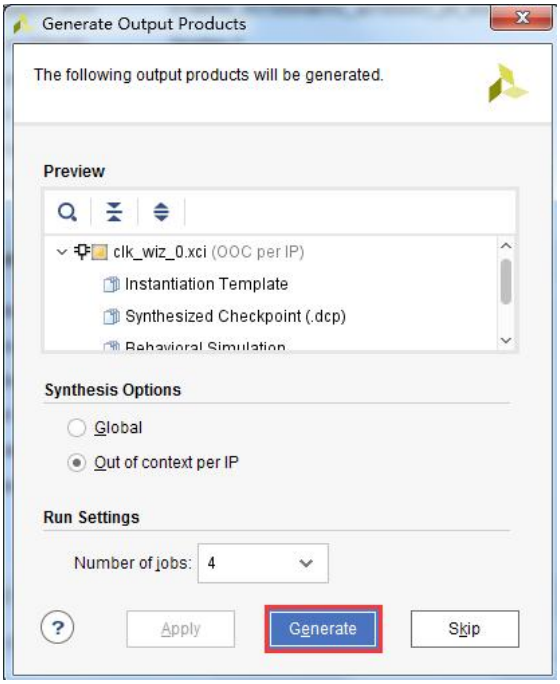
3. By default, the name of this “Clocking Wizard “is “ clk_wiz_0” , we will not make changes here. In the first interface, “Clocking Options”, we select the “PLL” resource and the input clock frequency is “200Mhz ”. Source as “Differential Clock Capable Pin”



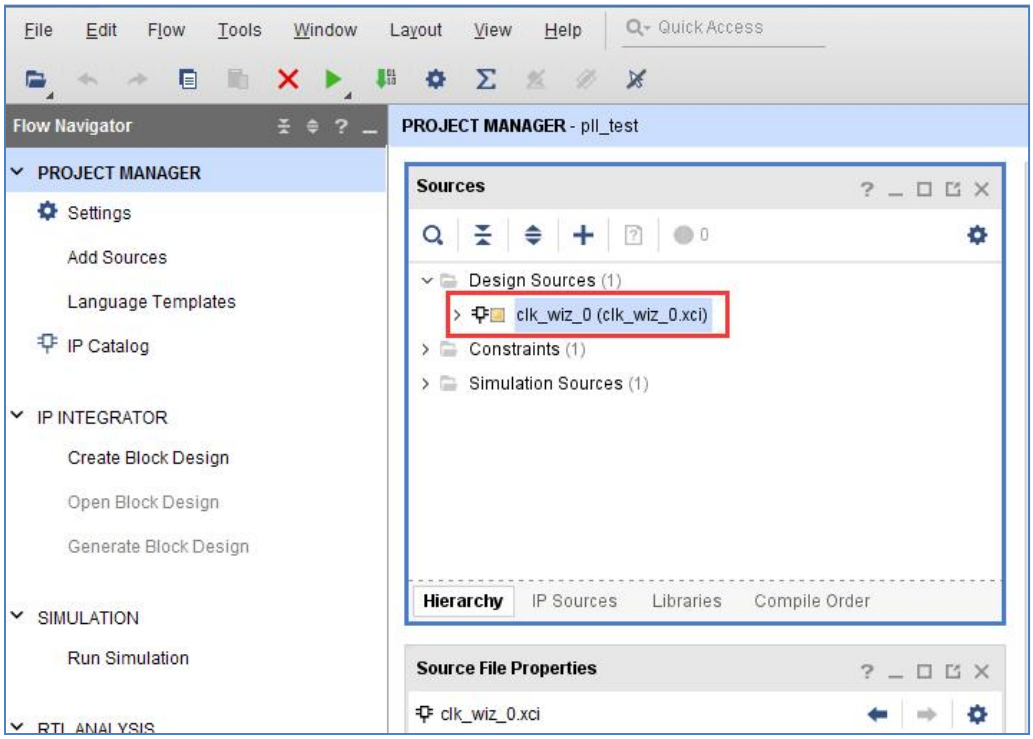
4. In the “Output Clocks” interface, select the the “clk_out1” as output, the frequencies are 50Mhz. Here you can also set the phase of the clock output, we do not set, keep the default phase, click OK to complete,



5. In the pop-up dialog box, click the “Generate” button to generate the design file for the “PLL IP”



6. At this time, the “IP” of “clk_wiz_0.xci” will be automatically added to our “pll_test” project. Users can double-click it to modify the “IP” configuration.



7. Let's write a top-level design file to instantiate the “PLL IP”. Write the “pll_test.v” code as follows.

```
module pll_test(
input       sys_clk_p,      // Differentia system clock 200Mhz input on board
input       sys_clk_n,
input       rst_n,
output      clk_out           //pll clock output J14_Pin1
            );
wire        locked;//时钟信号输出锁定
wire        pll_clk_o;//锁相环输出时钟
//////////////////////PLL IP call//////////////////////////////
clk_wiz_0 instance_name
   (
    // Clock out ports
    .clk_out1(pll_clk_o),     // output clk_out1
    // Status and control signals
    .reset(~rst_n), // input reset
    .locked(locked),       // output locked
   // Clock in ports
    .clk_in1_p(sys_clk_p),    // input clk_in1_p
    .clk_in1_n(sys_clk_n));    // input clk_in1_n
///////////////调用ODDR使时钟信号通过普通IO输出//////////////////
```

Call ORRR to output the clock signal through ordinary IO

```
ODDRE1 #(
      .IS_C_INVERTED(1'b0),  // Optional inversion for C
      .IS_D1_INVERTED(1'b0), // Unsupported, do not use
      .IS_D2_INVERTED(1'b0), // Unsupported, do not use
      .SRVAL(1'b0)           // Initializes the ODDRE1 Flip-Flops to the specified value (1'b0, 1'b1)
)
ODDRE1_inst (
      .Q(clk_out),   // 1-bit output: Data output to IOB
      .C(pll_clk_o),   // 1-bit input: High-speed clock input
      .D1(1'b1), // 1-bit input: Parallel data input 1
      .D2(1'b0), // 1-bit input: Parallel data input 2
      .SR(1'b0)  // 1-bit input: Active High Async Reset
   );
endmodule
```

In the program, first instantiate clk_wiz_0 IP, input the differential 200Mhz clock signal to clk_in1_p and clk_in1_n of clk_wiz_0, and connect the output of clk_out1 to the pll_clk_o network.

Note: The purpose of instantiation is to call the instantiated module to complete the code function in the upper level module. The format of the instantiated signal in Verilog is as follows: the module name must be the same as the module name to be instantiated, such as clk_wiz_0 in the program. Including the module signal names must also be consistent, such as clk_in1, clk_out1, clk_out2..... The connection signal is the signal transmitted between the TOP program and the module, and the connection signals between the module and the module cannot conflict with each other, otherwise a compile error will occur.

Module Name Extension Name

(

. Module Signal 1 (Connection Signal 1),

. Module Signal 2 (Connection Signal 2),

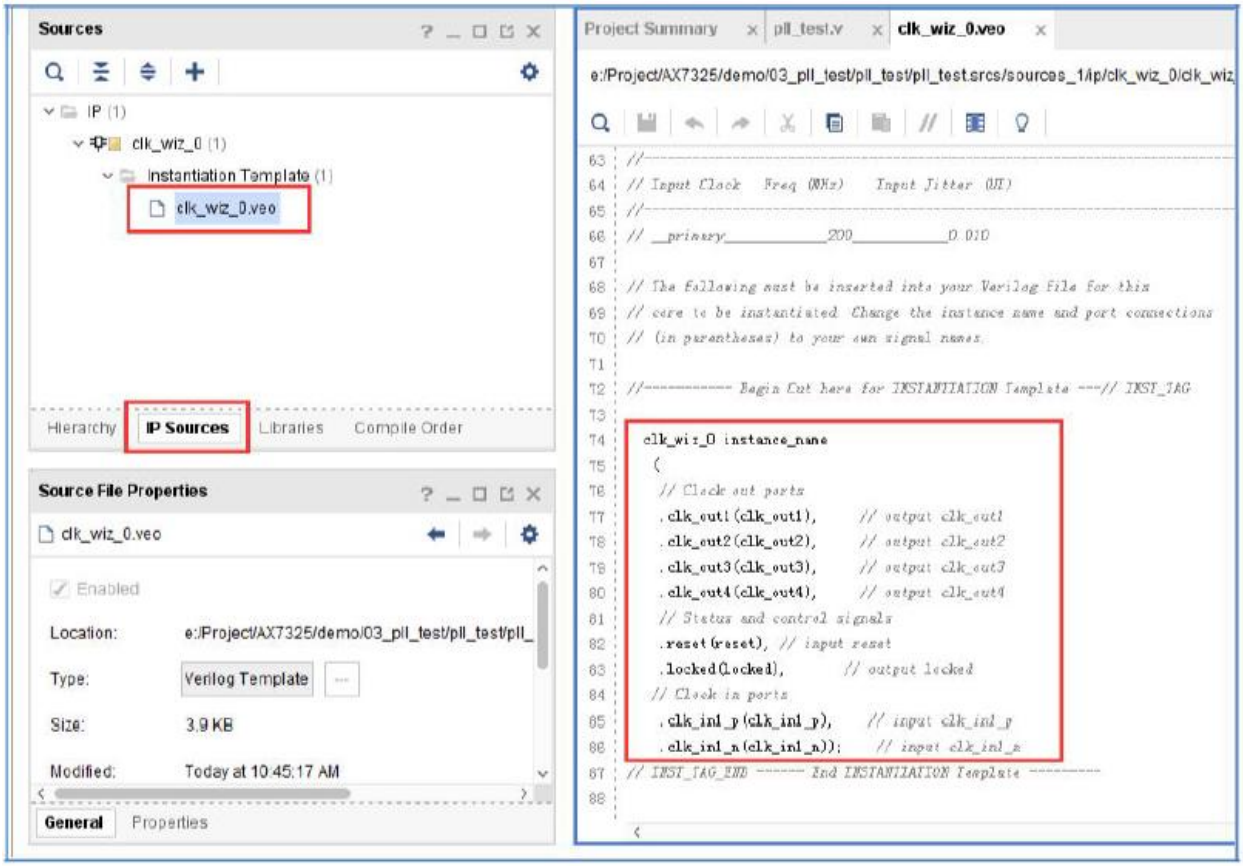
. Module Signal 3 (Connection Signal 3),

…… ……

. Module Signal N (Connection Signal N),

);

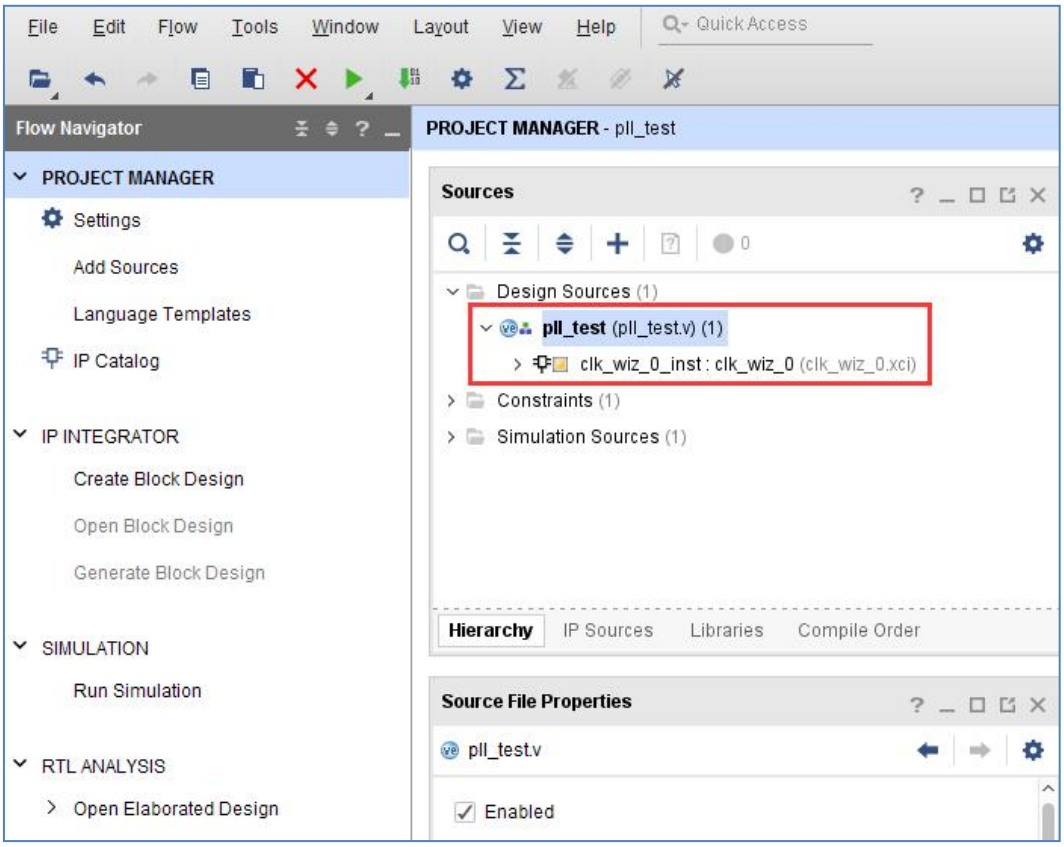
For how to instantiate IP, we can open its instantialion Template file, copy the following part of the code, and then modify the instance_name, port input and output signal names. (The program only sets 1 output clk_out1 here)



In addition, we add an “ODDR” primitive to our program, so that the clock signal of the “BUFG” output of “clk_wiz_0” can be output to the normal IO of the FPGA. If you connect directly to the “OBUF” from “BUFG”, an error will occur in the process of the compiler map.

To avoid this error, another way is to add a constraint to the constraint file, let the compiler ignore the requirements of the timing constraints, directly through the ordinary logical resources. However, the delay and jitter of the Clock output will be worse.

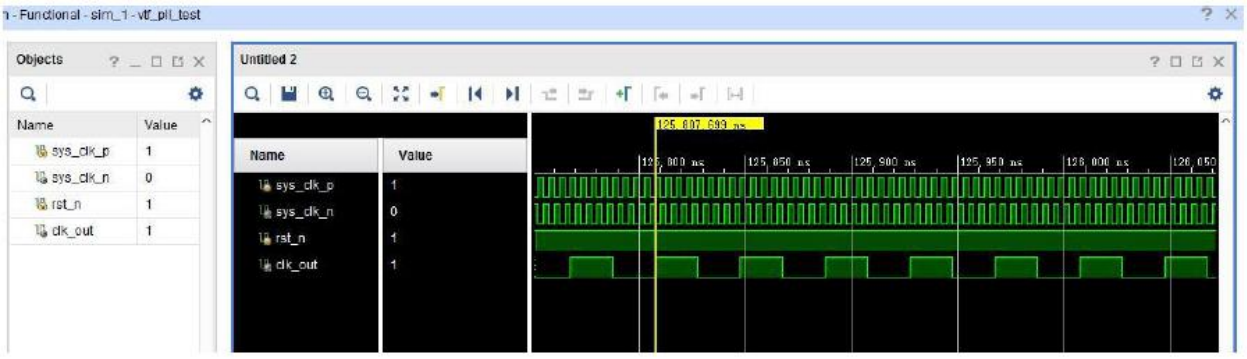
8. After saving the project, “pll_test” automatically becomes the top file, and “clk_wiz_0” becomes a submodule of the “Pll_test” file



9. Add the “xdc” pin constraint file “pll.xdc” to the project.

5 Simulation

Add a “vtf_pll_test” simulation file. After running, clk_out of the PLL has a clock signal output, and its output frequency is 25Mhz.

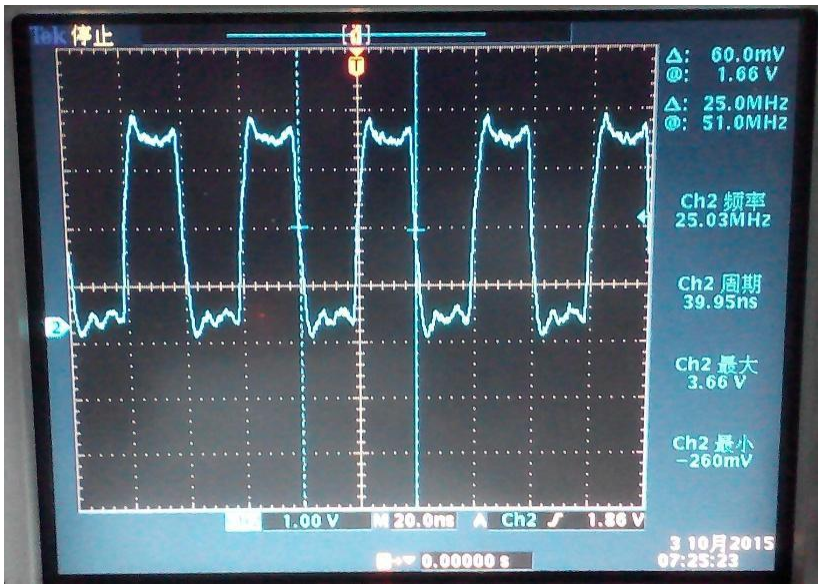


6 Measuring PLL output waveform

Compile the project and generate the “pll_test.bit” file, then download the bit file to the FPGA. Then we can use the oscilloscope to measure the output clock waveform.

Connect the ground wire of the oscilloscope probe to the ground on the development board (PIN76 of AXKU040 FPGA development board J16B), and connect the signal terminal to PIN77 of AXKU040 development board J16B. (Need to pay attention when measuring, avoiding the oscilloscope head touching other pins and causing short circuit between power supply and ground.)

At this time, we can see the 25Mhz clock waveform in the oscilloscope. The amplitude of the waveform is 3.3V, the duty ratio is 1:1, and the waveform display is as shown below:



If you want to output waveforms of other frequencies, you can modify the output of the clock to “clk_out2” or “clk_out3” or “clk_out4” of “clk_wiz_0”. You can also modify the “clk_out4” of “clk_wiz_0” to the frequency you want. Also note here, because the output of the clock is obtained by the PLL's multiplication and division factor of the input clock signal, so not all clock frequencies can be used. It should also be noted here, because the output of the clock is obtained by the PLL's multiplication and division factor of the input clock signal, so not all clock frequencies can be accurately generated by the PLL. However, the PLL will also automatically calculate the clock frequency for which the actual output is close.

It should also be noted that the bandwidth and sampling rate of some users' oscilloscopes are too low, which will cause the high-frequency part to attenuate too much when measuring high-frequency clock signals, and the amplitude of the measured waveform will become lower.