APRIL 2023, 4 BY ACKYE

Smart ZYNQ (SP&SL version) Engineering XIII Testing of PL-based PLL clock modules

This article describes how to use the clock module that comes with ZYNQ and output multiple clock signals.

This article is demonstrated on vivado2018.3, please research for other versions

(Note: The content of this section applies to the boards of Smart ZYNQ SP and SL Edition, if it is Smart ZYNQ Standard Edition, please refer to the corresponding board directory)

With FPGA design, sometimes we need different clocks to drive each module, for example, before the HDMI part we need to have both 40 and 200MHz clocks, or if we have a camera sensor in our design, then we usually need to give the sensor a 24-30MHz clock, or the system has SDRAM, at this time we need to give a clock of 100MHz or even higher to achieve fast data interaction, But usually we have only one crystal on the board, and if we want to generate different clocks, we need to use the PLL phase-locked loop function

The full English name of PLL is Phase Locked Loop, that is, phase-locked loop, which is a feedback control circuit. PLL performs system-level clock management and offset control of the clock network, with clock multiplication, frequency division, phase offset, and programmable duty cycle. The clock resource in the Xilinx 7 family of devices consists of a clock management unit CMT, each consisting of an MMCM and a PLL. For a simple design, the entire FPGA system uses a clock or divides the clock by writing code, but for a slightly more complex system, the system often needs to use multiple clocks and clock phase offsets, and the clock can not be multiplied by writing the clock output by writing

code, so learning the use of Xilinx MMCM/PLL IP cores is an important part of our learning FPGA. In this chapter, we will introduce you to the MMCM/PLL IP core with a simple example

PLL (phase-locked loop), that is, phase-locked loop. It can provide the system with low jitter, low latency, different frequencies, different phases of clock, greatly increasing the flexibility of system design and the success rate of FPGA design, it can be said that in the design of FPGA system, PLL is a very important resource, but also an important indicator to measure FPGA chip capabilities.

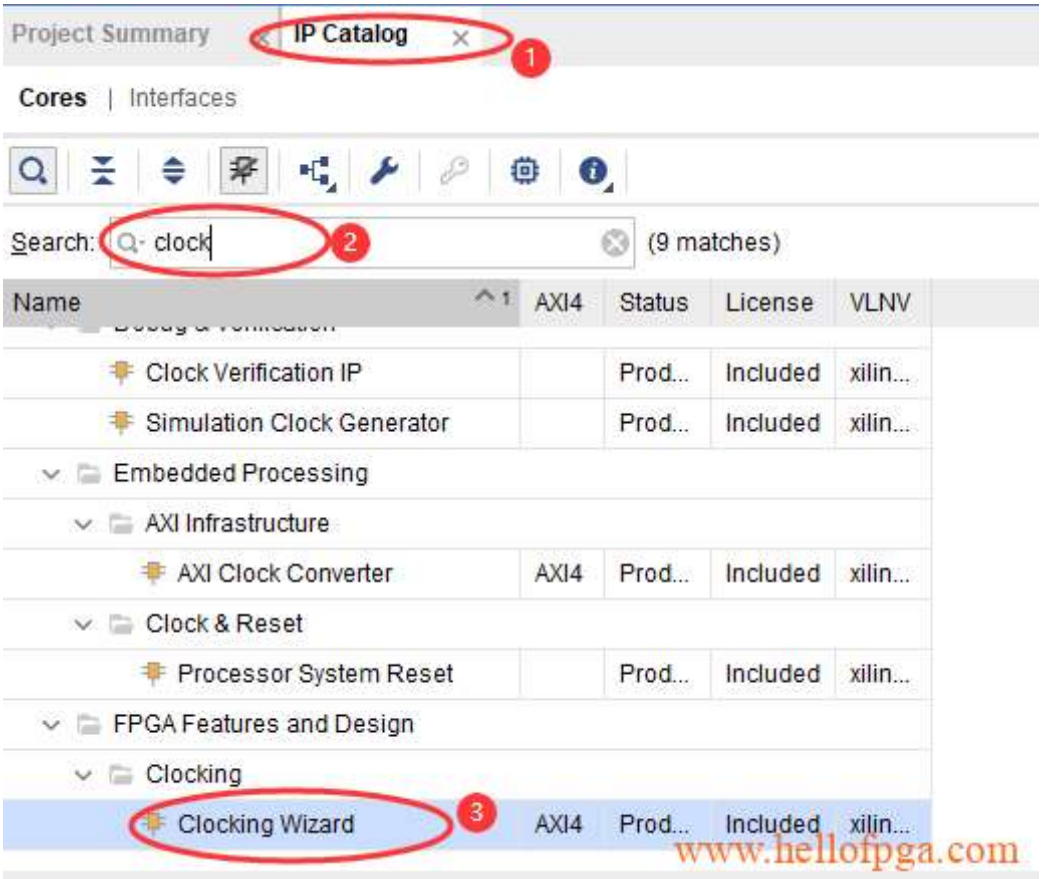
In this experiment, the PLL output clocks (100m and 25m) will drive two identical LED modules (two instantiations of the LED module in Project 100) by using clocks of different frequencies (90m and <>m) to observe the difference through the display frequency of the LED, and at the same time map the waveforms of the PLL output with the same frequency (<>mhz) but different phases (<> degrees) to the IO port, which is convenient for observation with an oscilloscope

programming

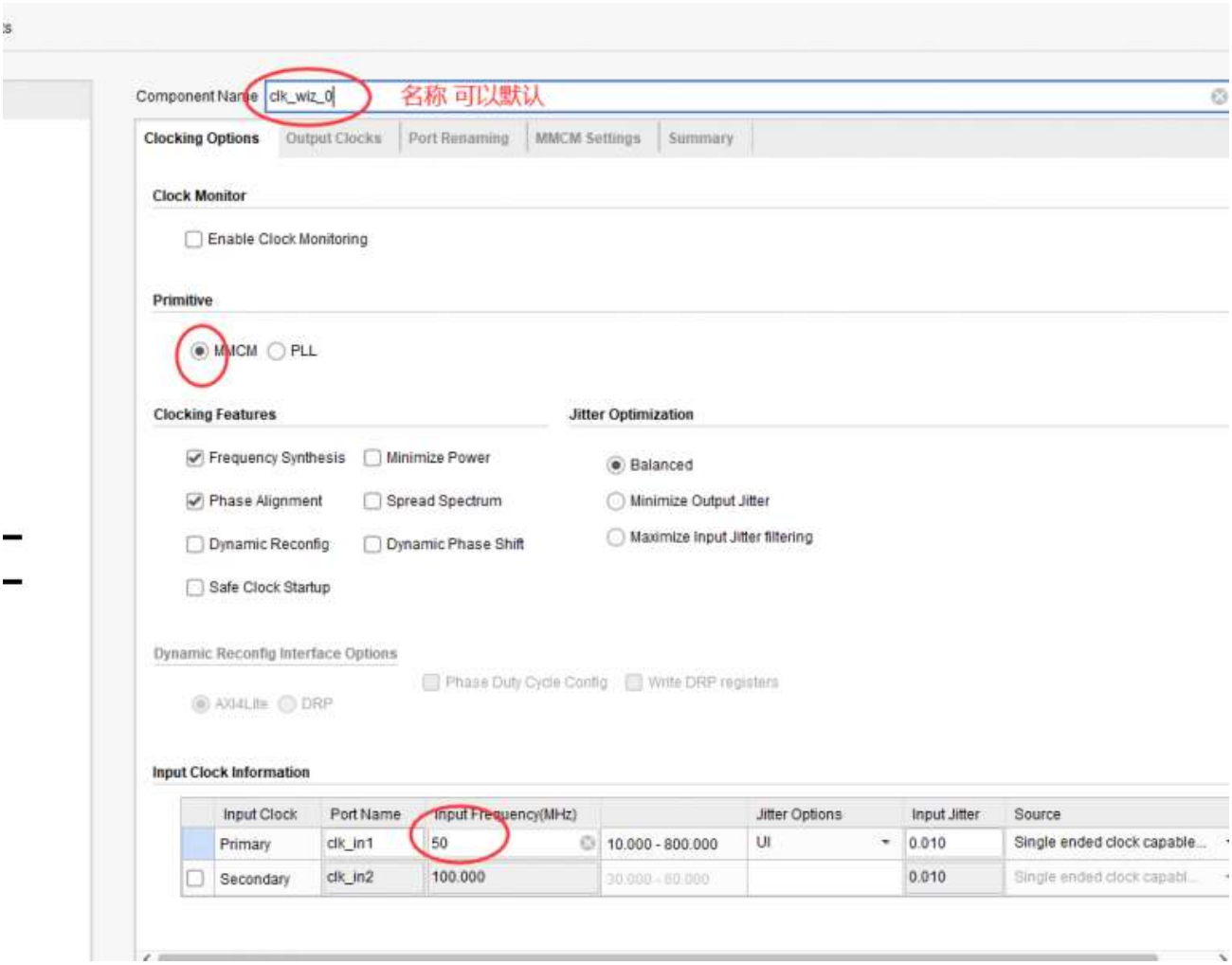
1) First create a project normally (you can refer to the previous example),

2) Add PLL IP cores. Check "IP Catalog" in the Vivado software

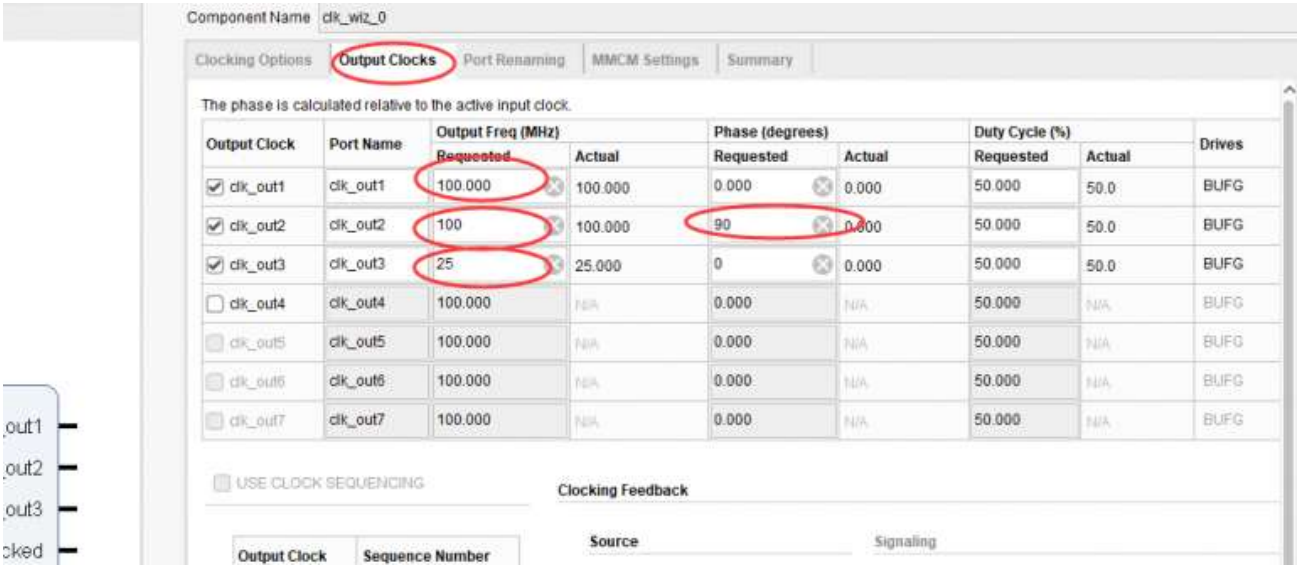
Search for clock in the pop-up window and double-click Clock Wizard in the search results. Enter the Clock Module Setup Wizard



The system pop-up window is shown in the figure below, here We choose the default MMCM, the input frequency of the input clock is changed to 50 on the board (corresponding to the 50M clock chip on the PL side), and other options remain default



Next we set the output clock, PLL in the system can be doubled or downfrequency, the input clock is 50m, then we output 3 clocks for testing CLK1 is 100MHz (frequency increase), CLK2 is set to 100M but the phase offset is 90 degrees (can be compared with CLK1), CLK3 is 25M frequency reduction,



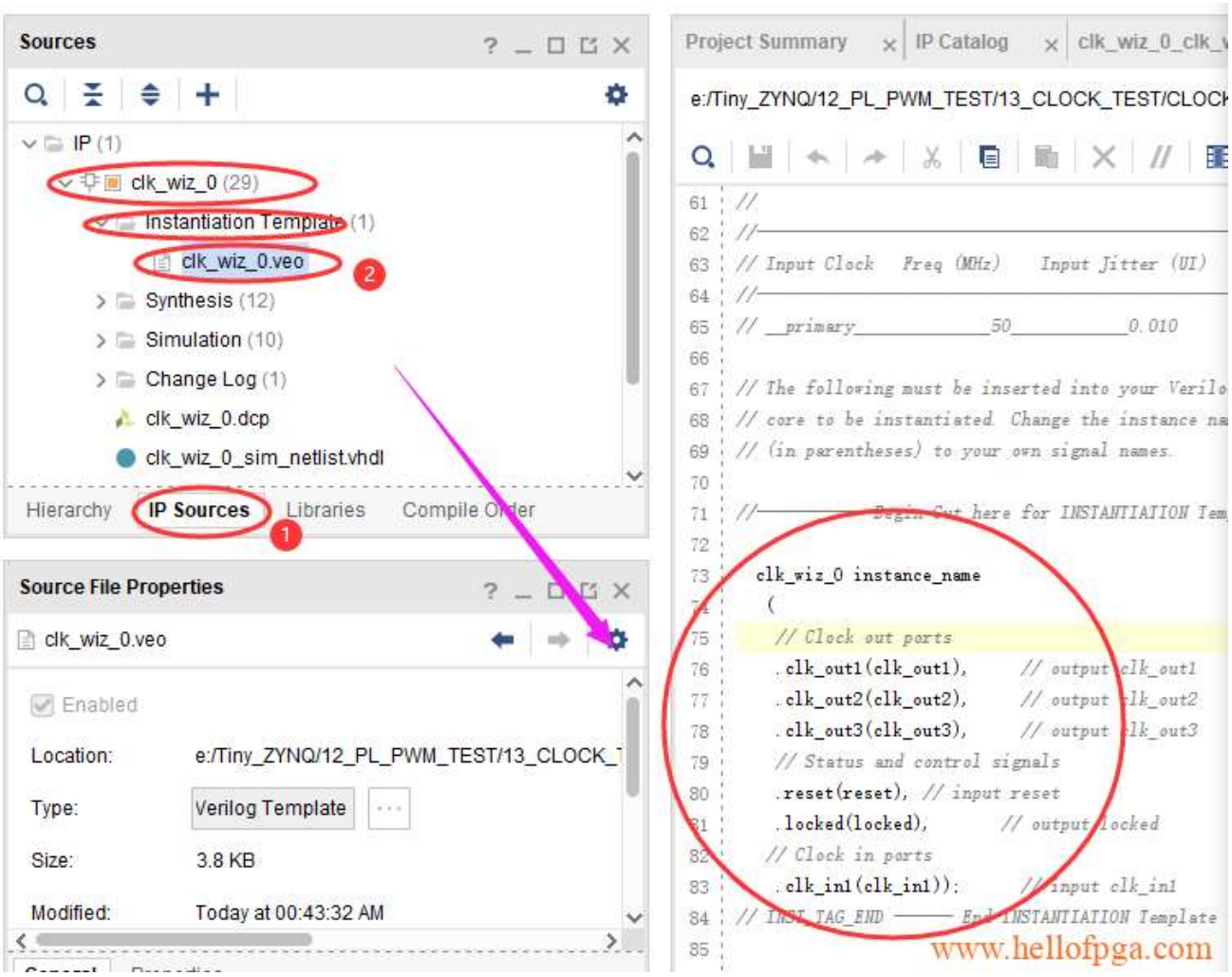
Some of the other options do not need to be modified to keep the default (in theory, not any frequency can be generated, **when the frequency system can not be generated**

directly, VIVADO will match the closest output frequency value according to the frequency you input and several sets of frequencies of output)

After that, click OK and select Generate in the pop-up window to generate the clock module

Next, instantiate this clock module in our code

Here's a tip, in the source directory's IP—>clk_wiz_0—> Instantitation Template—>clk_wiz_0.veo file we can find the reference code for the module instantiation, which can be called directly (renamed).



Next, the clk_100m, and clk_100m_90deg (phase offset by 90 degrees) are mapped to the output of the module for easy observation with an oscilloscope, and two identical LED modules are instantiated at the same time, driving the two LED modules with 100m and 25m clocks, respectively.

```
`timescale 1ns / 1ps
module CLOCK_TEST(
    input clk,  //50M in
    output clk_out1,  //100M out
    output clk_out2,  //25M out
    output led1,
    output led2
);

  wire clk_100m;
  wire clk_100m_90deg;   //25M out 90deg
  wire clk_25m;
  clk_wiz_0 u_clock(
    .clk_out1(clk_100m),
```

```
    .clk_out2(clk_100m_90deg),
    .clk_out3(clk_25m),
    .reset(0),
    .locked(),
    .clk_in1(clk)
);

 assign clk_out1=clk_100m;
 assign clk_out2=clk_100m_90deg;

LED_MODULE U1(
    .clk(clk_100m),
    .led(led1)
);

LED_MODULE U2(
    .clk(clk_25m),
    .led(led2)
);
endmodule
```

Because the RESET signal in the clock module is active high, the above program directly assigns 0 to the reset signal, that is, .reset(0) (of course, you can also connect an external reset signal, which is only demonstrated here so it is not connected)

The LED module directly calls the LED module written in Project 50 (under normal circumstances, if a standard <>m waveform is input, the LED light changes its state once a second)

```
module LED_MODULE(
    input clk,
    output led
);
 parameter T1MS = 26'd50_000_000 ; //50M晶振时钟
```

```
reg [25:0]time_count;//时钟计数器
reg led_r;
always@(posedge clk)
    if(time_count>=T1MS)begin
        time_count<=26'd0;
        led_r<=~led_r;
    end
    else time_count<=time_count+1'b1;
assign led=led_r;
endmodule
```

Add a constraint file (because the 50m input is connected to normal IO, an additional CLOCK_DEDICATED_ROUTE constraint statement) should be added)

```
set_property IOSTANDARD LVCMOS33 [get_ports clk_out1]
set_property IOSTANDARD LVCMOS33 [get_ports clk_out2]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports led1]
set_property IOSTANDARD LVCMOS33 [get_ports led2]

set_property PACKAGE_PIN M19 [get_ports clk]
set_property PACKAGE_PIN H19 [get_ports clk_out1]
set_property PACKAGE_PIN E18 [get_ports clk_out2]
set_property PACKAGE_PIN P20 [get_ports led1]
set_property PACKAGE_PIN P21 [get_ports led2]
```

After that, it is compiled, synthesized and downloaded

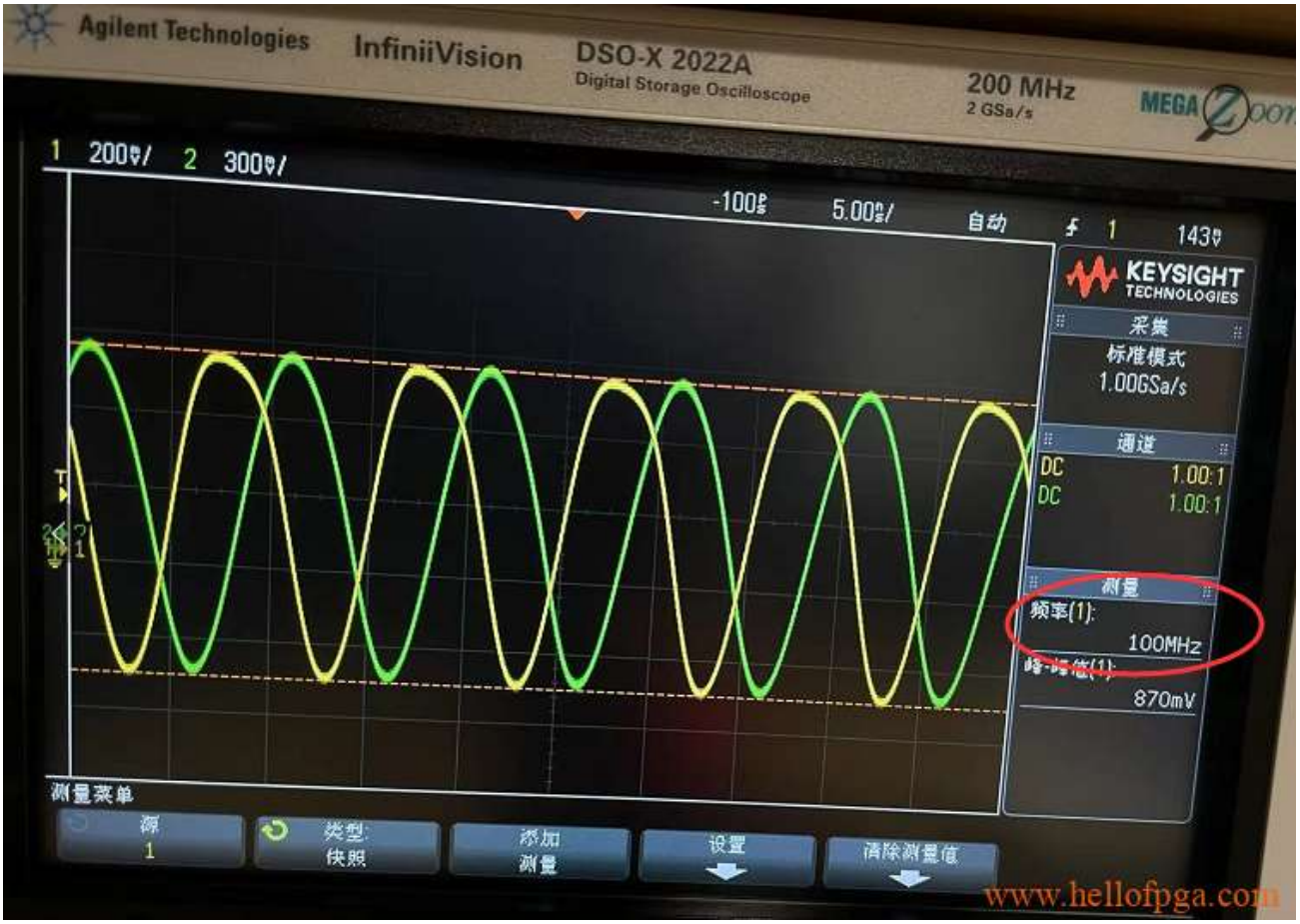
Watch the LED light flicker

After success, we can observe through the LED that LED1 flashes every 1 second (that is, 500ms changes the state), and LED1 flashes every 4 seconds (that is, 2 seconds to change the state), through theoretical analysis LED1's drive clock 100m (equivalent to 100_000_000), so the LED module's every 50_000_000 change state is just half a second that is 500ms), the same way the drive clock of LED2 is 25mhz (equivalent to 25_ 000_000), so the LED module's state change every 50_000_000 is exactly two cycles, that is, 2s clocks).

Observe two 100Mhz waveforms through an oscilloscope

Connect the oscilloscope pens to the H19 and E18 as shown in the following figure:

Two 100MHz waveforms can be seen through the oscilloscope, and there is a certain phase deviation between the two

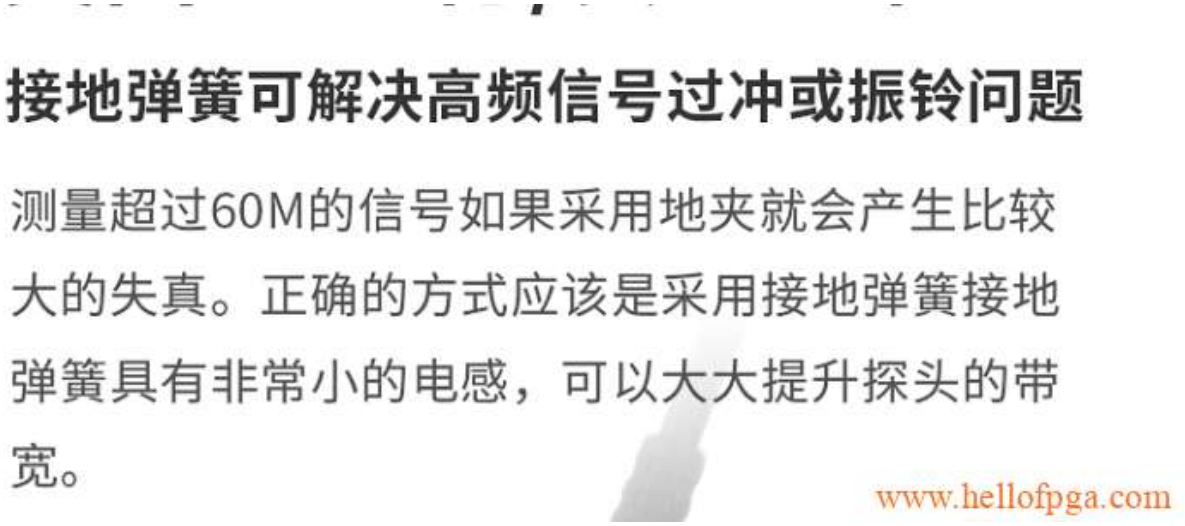


These are some simple application tests for PLL modules

Note: Since the FPGA output impedance is about 50 ohms, most oscilloscopes input impedance is 1M, and because the GPIO is directly connected to the oscilloscope (without any load), the impedance mismatch between the two causes some oscilloscopes to appear obvious ringing signals when measuring high-speed square wave signals (it is a normal phenomenon that does not require too much attention). When the measured waveform has a large overshoot affecting observation, you can try to connect a resistor in series between the oscilloscope and the signal to eliminate the ringing signal caused by impedance mismatch (50-200 ohms, the larger the better). The overshoot of the waveform and the size of the load, as well as the measurement method, are also related to the input impedance of the oscilloscope.

The ringing signal is similar to the figure below (normal), and this test only needs to observe the waveform frequency

Postscript (I also found on the Internet that the oscilloscope's grounding clip is replaced with a spring probe that can solve the problem of ringing in the waveform caught by the oscilloscope, and I will buy a spring probe to test it later), and then update this content when the spring probe comes back



The complete project is as follows:

13_CLOCK_TEST_XC7Z020 **Download**

SMART ZYNQ SP & SL