APRIL 2023, 4 BY ACKYE

Smart ZYNQ (SP&SL Edition) Engineering XXI PL-based VIO online debugging functional test

VIO stands for Virtual Input Output Unit, which can monitor and drive logic signals inside the FPGA in real time, and this article will demonstrate the specific debugging process.

Many times we need to enter specific values when adjusting the program to find the best parameters, in the past we can only transmit data to the system through serial port or buttons, but this has certain requirements for hardware (there must be a corresponding hardware interface), and in this way we need to write additional FPGA code corresponding to the serial port. Here xilinx gives us another idea, that is, VIO virtual input and output core, through VIO, we only need to pass the JTAG data line on VIVADO to complete the above requirements.

Unlike ILA, VIO's data is real-time (the current value can be displayed), while ILA can only capture waveforms for a period of time, and VIO can support parameter input.

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)

The following is the specific implementation process

Addition of VIO modules

This article uses a simple project of counters to demonstrate the use of VIO function modules.

Create a project: (This article only introduces the VIO part, if you are not familiar with the project creation process, you can see Project 1)

Smart ZYNQ (SP&SL version) Project 1 Light up an LED with ZYNQ's PL resources (full graphic)

Here is a very simple program for demonstration

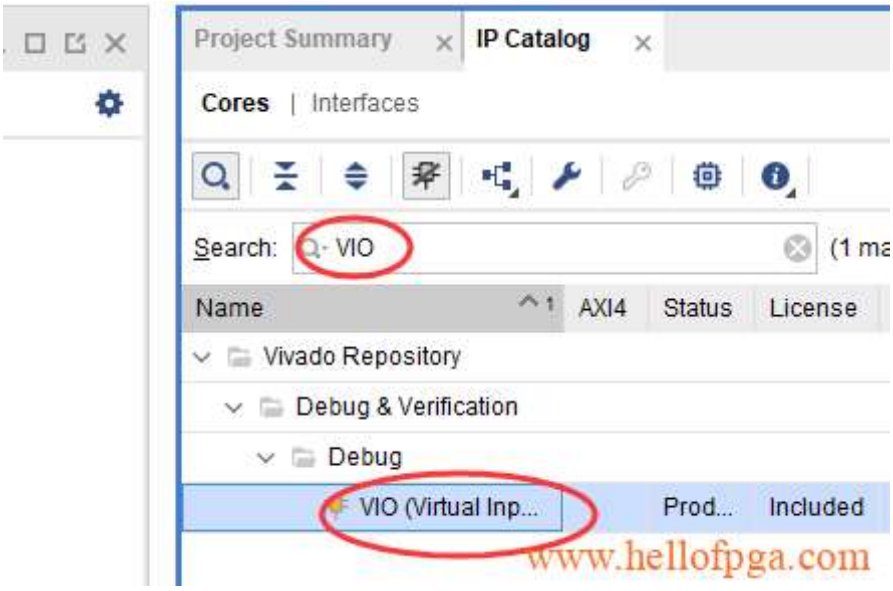
```
`timescale 1ns / 1ps
module VIO_TEST(
     input CLK,
     output LED0,
     output LED1
 );


 wire [1:0]LED_STATE;//时钟计数器

 assign LED0=LED_STATE[0];
 assign LED1=LED_STATE[1];

endmodule
```

Very simple, LED0, LED1 state corresponds to LED_STATE [0], LED_STATE [1] respectively

We next use VIO to enter the value of the LED_STATE. It is equivalent to using VIO to control the turn on and off of LED lights

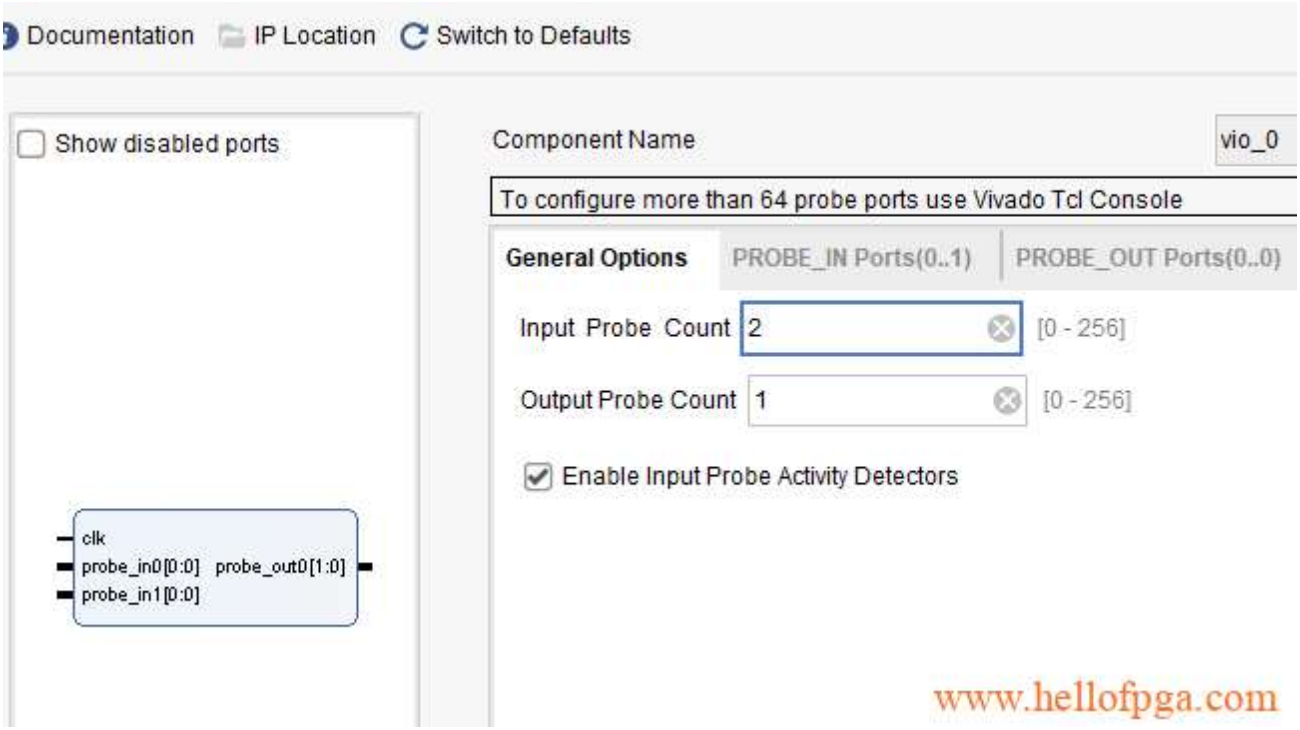
Call the VIO module

In the vivado software, open the IP Core Directory and search for VIO, as shown in the following figure



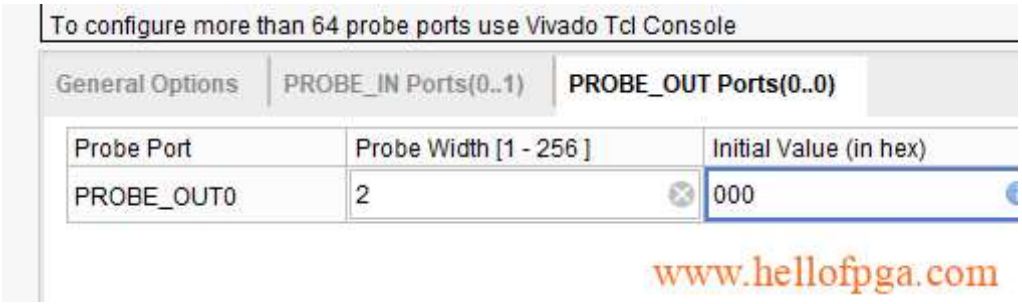
这里 input选择2个探针，来显示2个LED的状态

output selects 1 probe to enter the LED_STATE value into the system



Both inputs are set to 2 bit

Select 2 bits wide for the output data, and the default value is set to 0



After that, select OK to generate VIO resources

Add the instantiation code of the VIO module to the code we wrote earlier (instantiation is to generate an entity)

```
 vio_0 u_vio (
  .clk(CLK),                    // input wire clk
  .probe_in0(LED0),     // input wire [0 : 0] probe_in0
  .probe_in1(LED1),     // input wire [0 : 0] probe_in1

  .probe_out0(LED_STATE)  // output wire [2 : 0] probe_out0
);
```

The complete code is as follows:

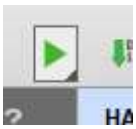
```
`timescale 1ns / 1ps
module VIO_TEST(
     input CLK,
     output LED0,
     output LED1
 );
```

```
 wire [1:0]LED_STATE;//时钟计数器

 assign LED0=LED_STATE[0];
 assign LED1=LED_STATE[1];


 vio_0 u_vio (
  .clk(CLK),                // input wire clk
  .probe_in0(LED0),    // input wire [0 : 0] probe_in0
  .probe_in1(LED1),    // input wire [0 : 0] probe_in1
  .probe_out0(LED_STATE)  // output wire [2 : 0] probe_out0
);

endmodule
```

Add constraint files

```
set_property IOSTANDARD LVCMOS33 [get_ports LED0]
set_property IOSTANDARD LVCMOS33 [get_ports LED1]
set_property IOSTANDARD LVCMOS33 [get_ports CLK]

set_property PACKAGE_PIN P20 [get_ports LED0]
set_property PACKAGE_PIN P21 [get_ports LED1]
set_property PACKAGE_PIN M19 [get_ports CLK]
```

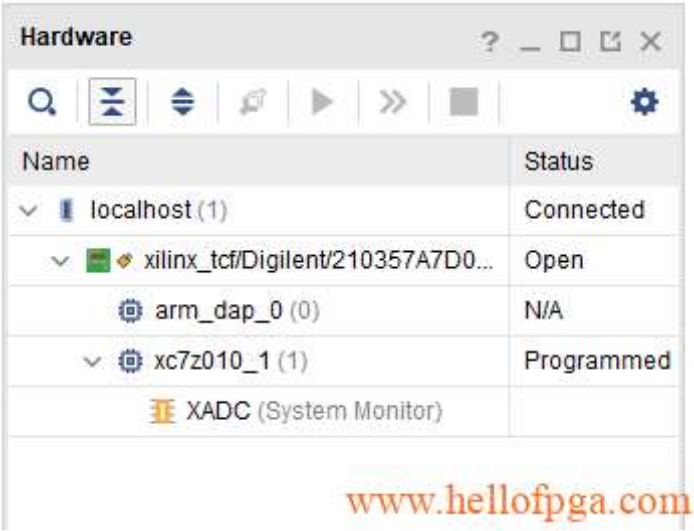
After that, click Run and Generate Bitstream to synthesize the cabling and generate the binary



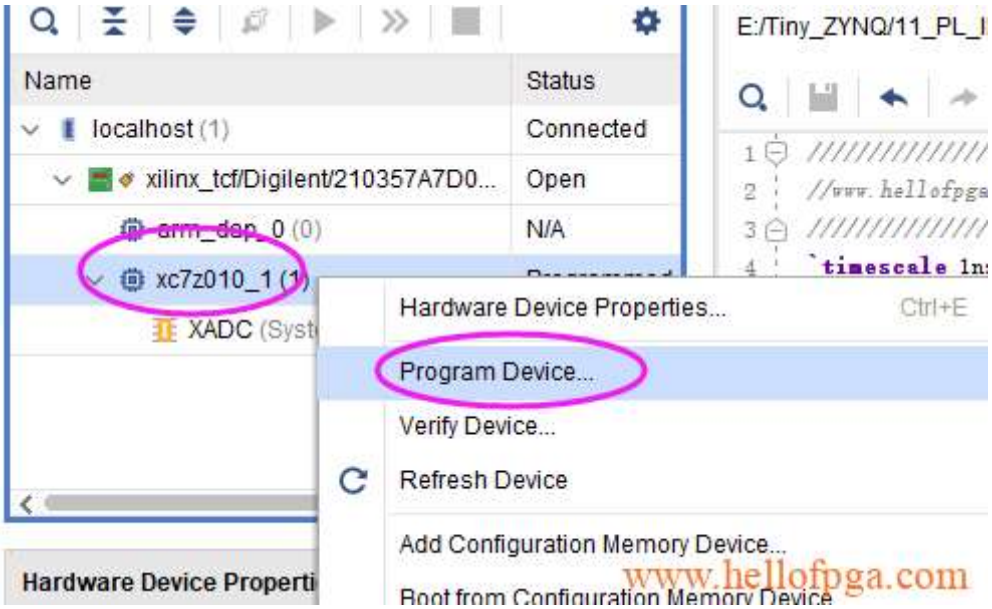
2. Download the code

2.1 Turn on JTAG and the motherboard, and power the motherboard, click open target to connect the target board

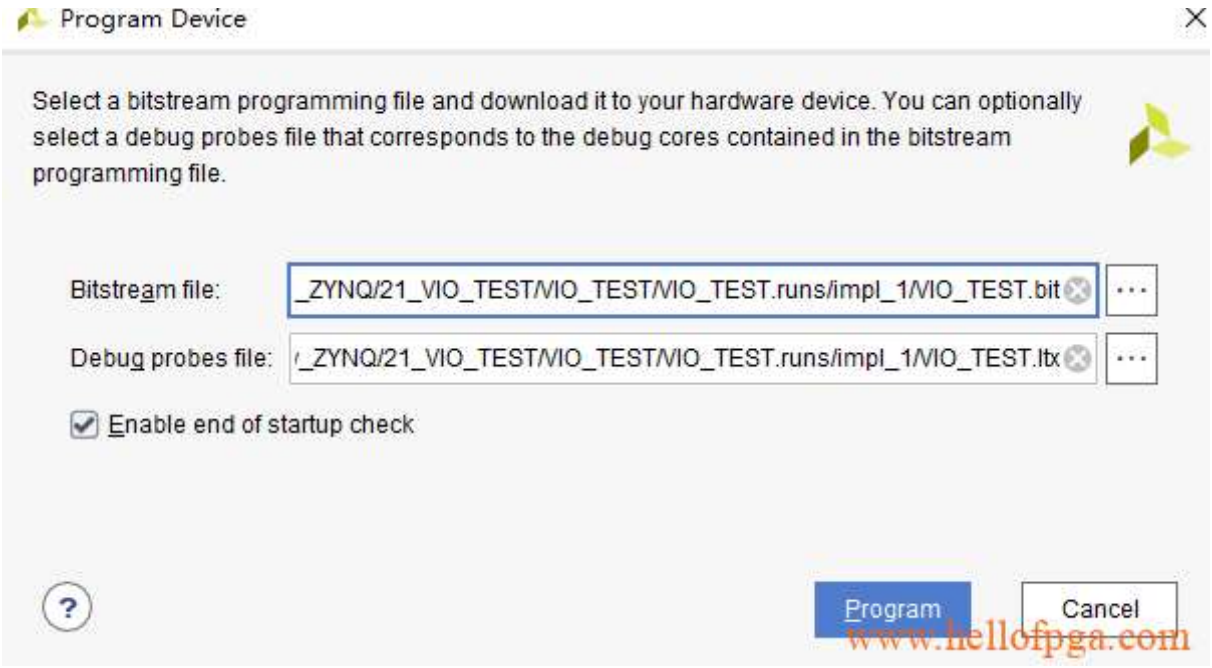
2.2 The following figure will appear after successful connection



2.3 Right-click to select the chip model, and click program device to download the FPGA code (these operations are consistent with normal FPGA engineering)

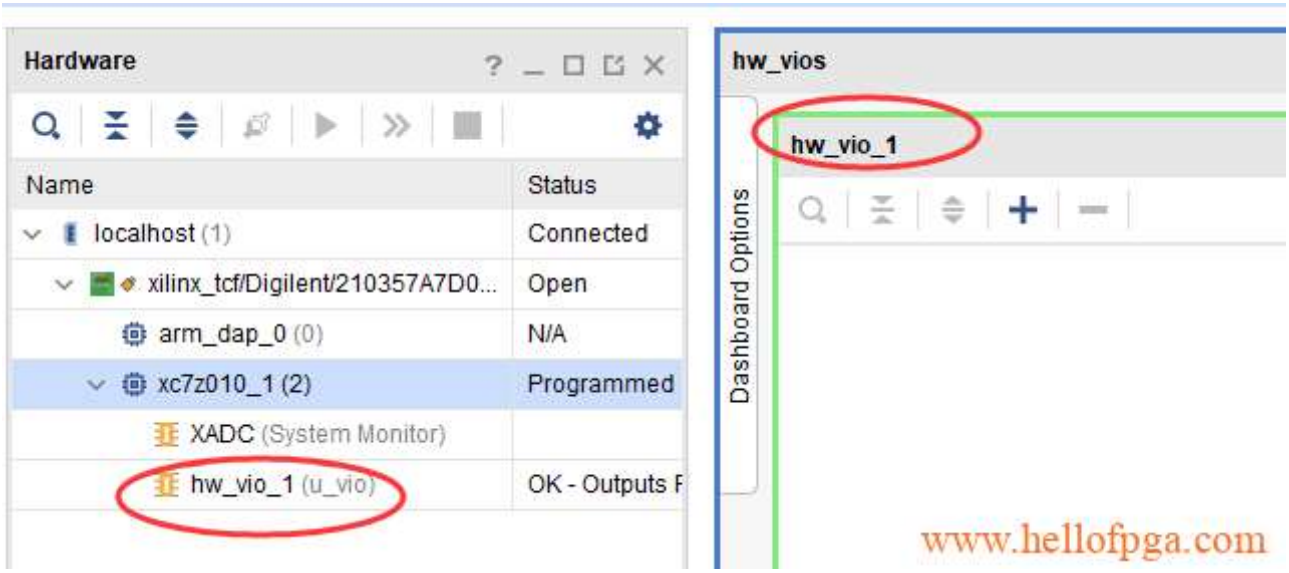


Click Program to download

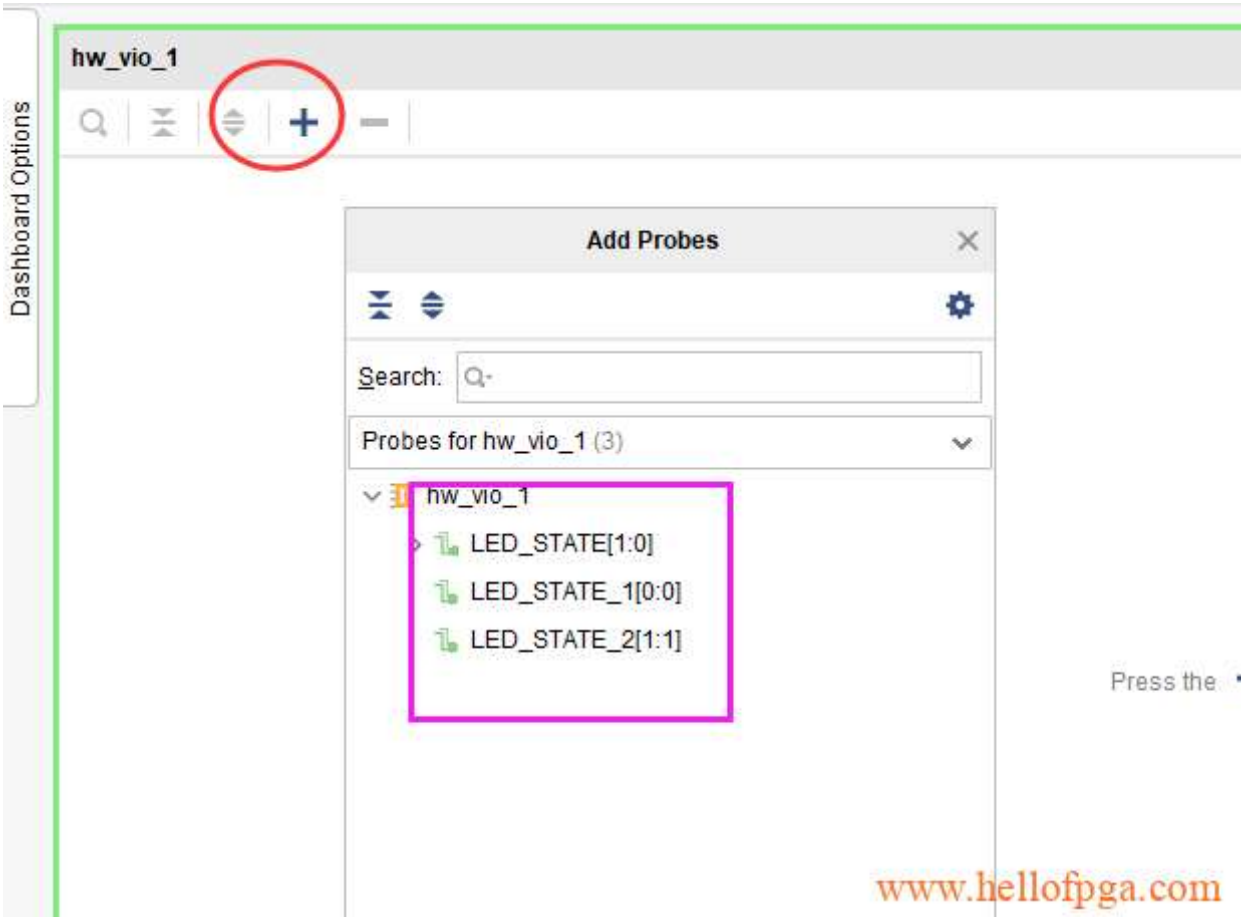


VIO debugging

After performing the previous download operation, the system will automatically pop up a VIO debugging window if it runs fine, as shown in the following figure

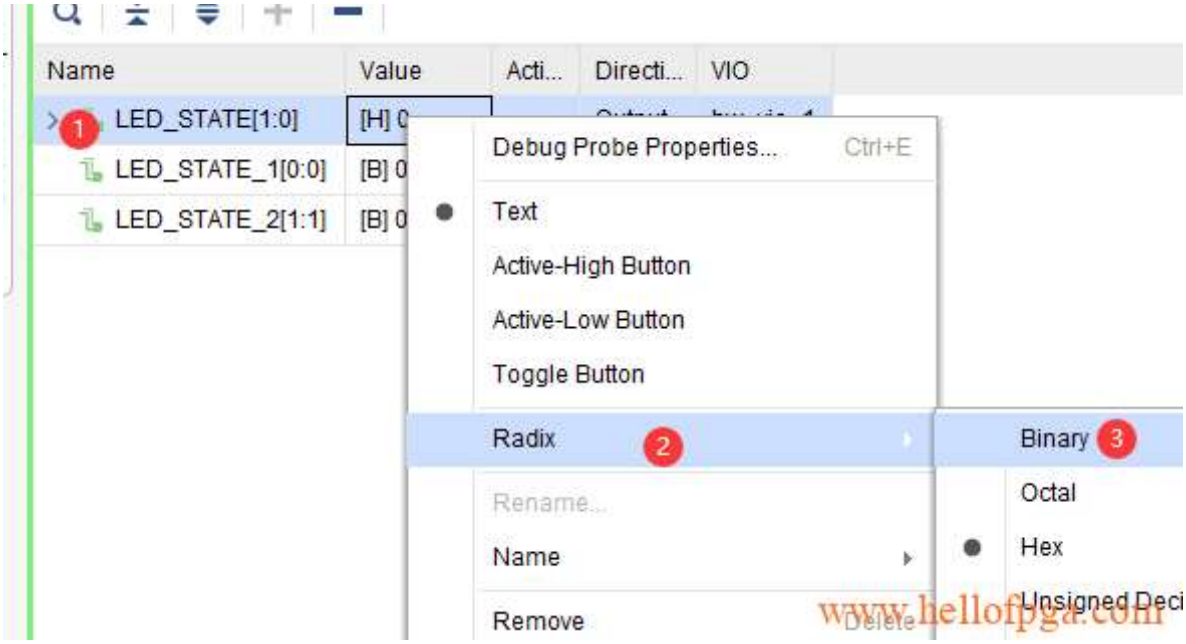


The default debugging window is empty and there are no parameters to adjust, click the plus sign here, then check all the signals that appear, and click OK

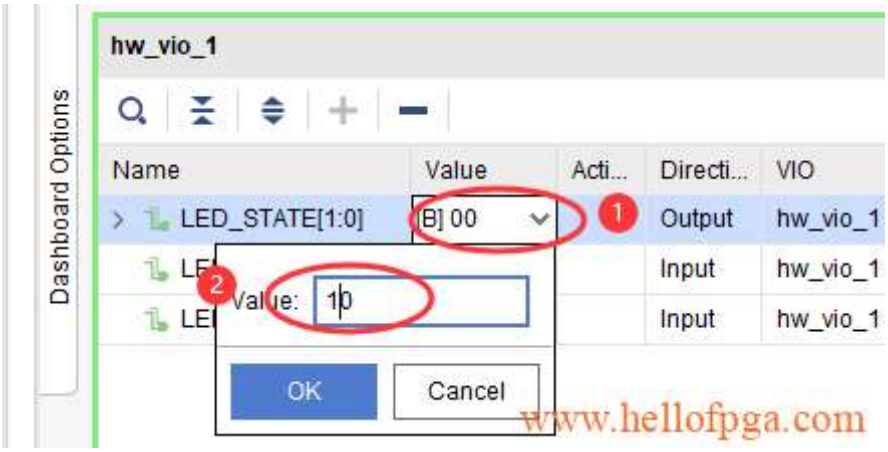


At this point we have the debug window of the signal we want

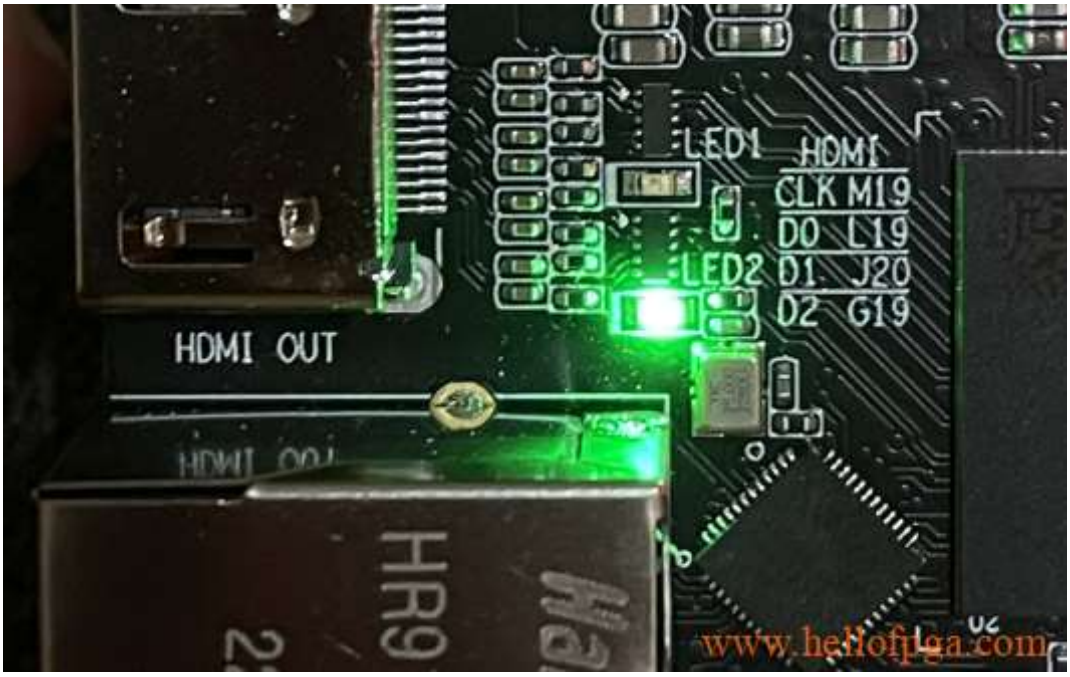
Because LED_STATE we need to adjust bitwise, we adjust the hexadecimal of the LED_STATE to binary here



准备工作做好了接下来我们来实际调试，首先没有任何动作情况下两个LED灯应该默认是熄灭的，之后我们点选LED_STATE信号的 VALUE,将信号改成10



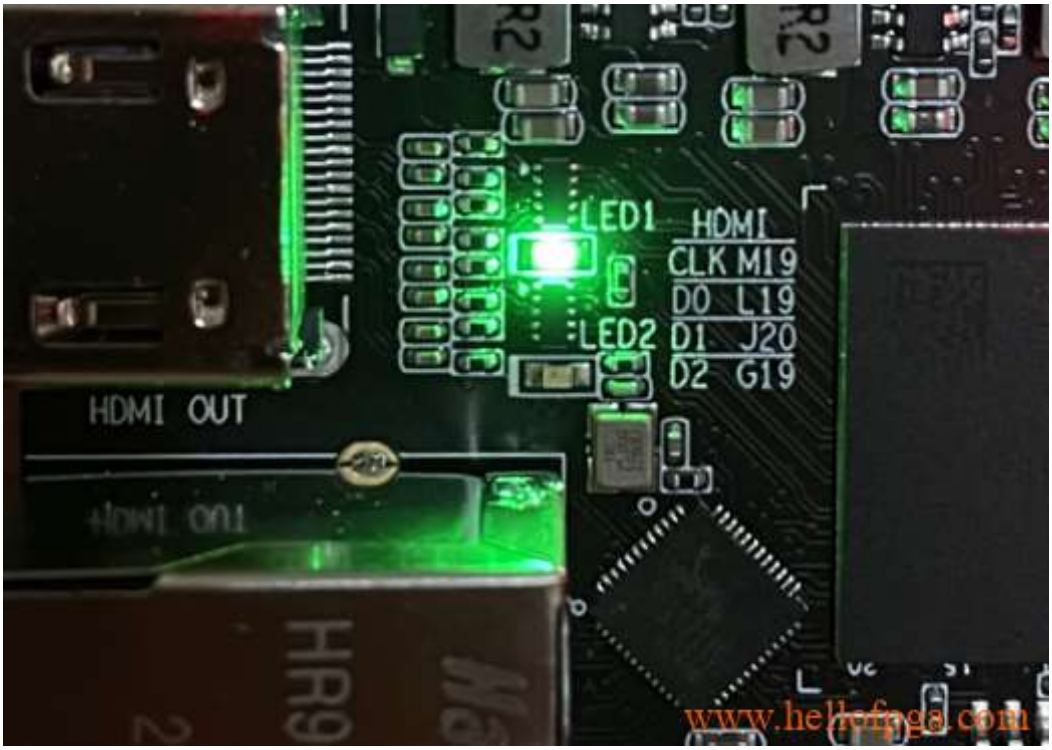
修改之后 点选OK后，我们会发现其中一个LED(LED2)灯被点亮



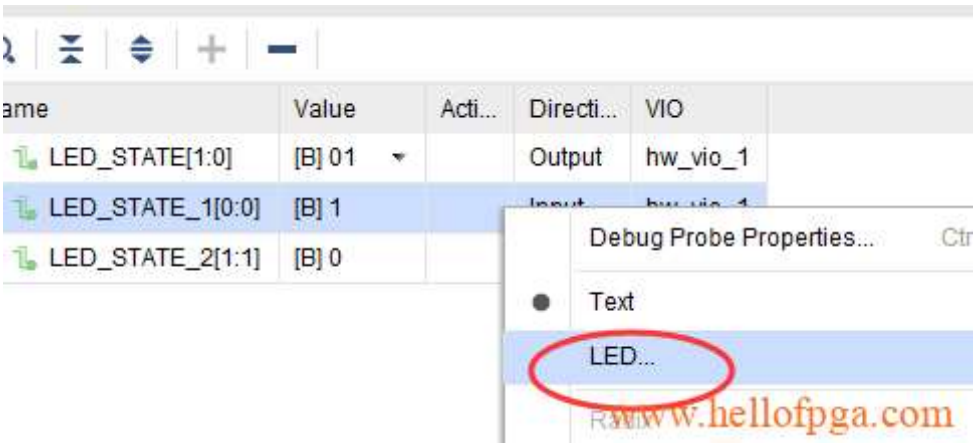
此时观察我们的VIO窗口可以看到 下面两个input窗口其中一个变成1了，并且右边有个向上箭头（代表这个信号有变化，并且是由低变高）

同样我们把LED_STATE由10改成01（可以看到下面input窗口中也同样读到信号变成1和0了，并且一个带向上箭头，一个带向下箭头）

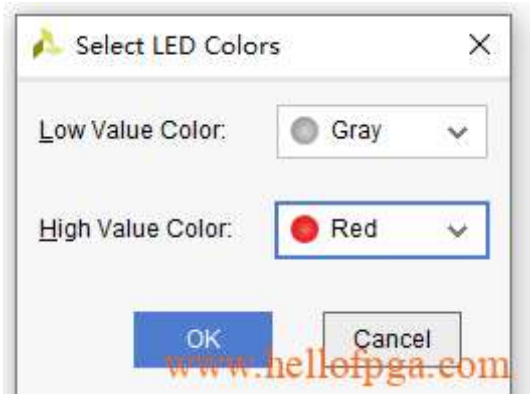
同样的LED灯和之前反向了(LED1被点亮)



VIO还有其他玩法，我们还可以将我们之前观察的信号换算成LED指示灯



Set the low level to gray and the high level to red



The same is done with another input signal, and the following debugging results are obtained (red and gray, corresponding to the LED lights on the board, making debugging more intuitive)

Of course, there are many ways to play (including the output can be set to the form of buttons) here is only a simple demonstration, so you can try it yourself

Through the above experiments, we can know that VIO can write and read out the specified signals in the actual running program in real time, which greatly shortens the debugging time when we do engineering.

The following is the complete project (for learning reference only):

21_VIO_TEST_XC7Z020 **Download**

SMART ZYNQ SP & SL