

实验五

试验名称：使用 ChipScope Pro 分析调试 FPGA 设计

0、ChipScope Pro 简介：

ChipScope Pro 是用于分析调试 Xilinx FPGA 设计的片内逻辑的工具，ChipScope Pro 的主要功能是通过 JTAG 口，在线实时地读出 FPGA 的内部信号。基本原理是利用 FPGA 中未使用的 BlockRAM，根据用户设定的触发条件将要观测信号实时地保存到这些 BlockRAM 中，然后通过 JTAG 口传送到 PC 机，显示出时序波形。

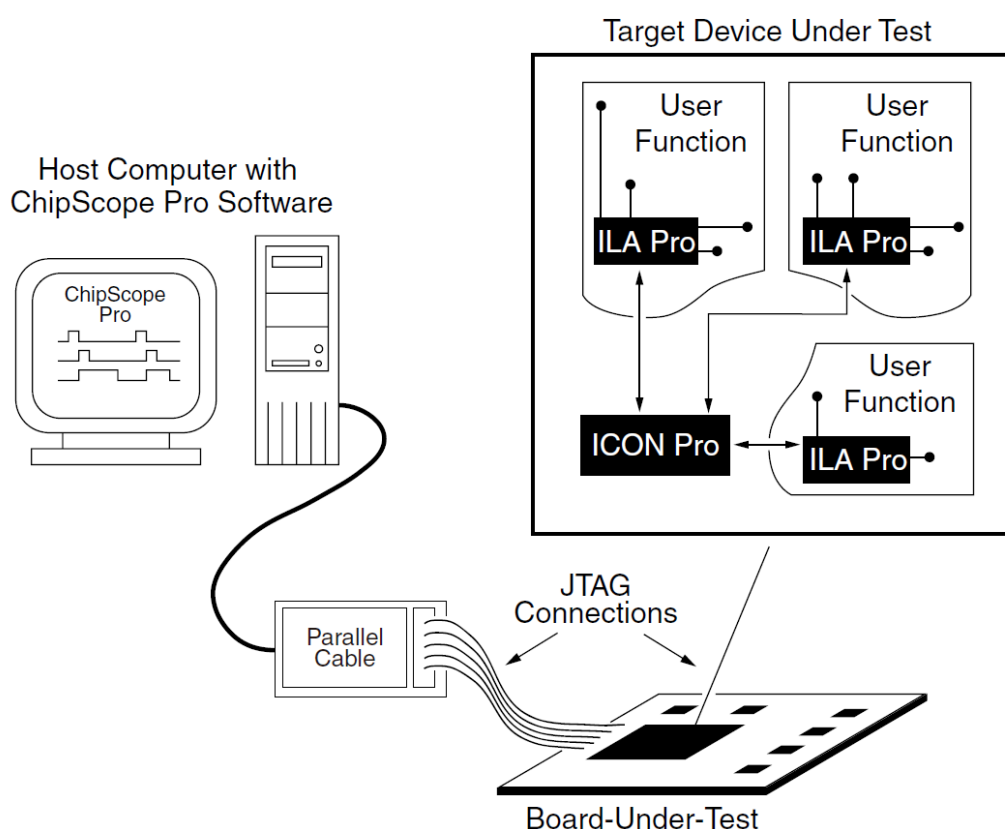


图 1、ChipScope Pro System Block Diagram

一般来说，ChipScope Pro 在工作时需要在用户设计中实例化两种核：

- (1) 集成控制器核 (ICON core, Integrated Controller core)，负责 ILA 核和边界扫描端口的通信，一个 ICON 核可以连接 1~15 个 ILA 核。
- (2) 集成逻辑分析仪核 (ILA core, Integrated Logic Analyzer core)，提供触发和跟踪捕获的功能；
- (3) VIO (Virtual Input/Output)，A module that can monitor and drive signals in your design in real-time. You can think of them as virtual push-buttons (for input) and LEDs (for output). These can be used for debugging purposes, or they can be incorporated into your design as a permanent I/O interface.

ChipScope Pro 工具箱包含 3 个工具：

- ChipScope Pro Core Generator (核生成器)
- ChipScope Pro Core Inserter (核插入器)
- ChipScope Pro Analyzer (分析器)

ChipScope Pro Core Generator 的作用是根据设定条件生成在线逻辑分析仪的 IP 核，包括 ICON 核、ILA 核、VIO 等核，设计人员在原 HDL 代码中实例化这些核，然后进行布局布线、下载配置文件，就可以利用 ChipScope Pro Analyzer 设定触发条件、观察信号波形。

有关 ChipScope Pro 的详细参考资料，请参阅《ChipScope Pro 10.1 Software and Cores User Guide, UG029 (v10.1) March 24, 2008》。

1、Step by Step —— 使用 ChipScope Pro 分析参考设计

1.1 参考设计功能描述:

(0) 对 Spartan - 3E FPGA Starter Kit Board 上的 50MHz 的时钟晶振 (C9) 进行分频, 产生 5MHz 的时钟信号 (CLK_5MHz)。

分频电路模块的 Verilog 源码如下:

```
// File: clock_div10.v
module clock_div10(output reg clk_div10, input clock, input rst_n );

    parameter FIVESTAGES = 3'b100;
    reg [2:0] cnt;

    always @( posedge clock or negedge rst_n )
    begin
        if ( !rst_n ) begin
            clk_div10 <= 1'b0;
            cnt <= 3'b0;
        end
        else begin
            if ( cnt < FIVESTAGES ) begin
                cnt <= cnt + 1'b1;
                clk_div10 <= clk_div10;
            end
            else begin
                cnt <= 3'b0;
                clk_div10 <= ~clk_div10;
            end
        end
    end
endmodule
```

(1) 设计一个 32 位计数器, 在分频获得的 5MHz 的时钟信号控制下进行计数。该计数器的输入/输出端口如下:

i)、输出端口:

MSB12 —— 32 位计数器最高 12 位, 其中:

MSB12[11:8] 连接虚拟输入/输出端口 (VIO)

MSB12[7:0] 连接 Spartan - 3E FPGA Starter Kit Board 上的 8 个 LED: LED7 ~LED0。

cnt4b —— 32 位计数器最低 4 位, 连接逻辑分析仪数据捕捉端口

ii) 输入端口:

clock —— 连接 50MHz 的时钟晶振 (C9);

rst_n —— 异步复位输入端口, 低电平 (1'b0) 有效; 连接开发板上 SW0, 当 SW0 = 0 (off) 时, 计数器复位。

dir —— 计数方向控制, 高电平 (1'b1) 有效, 连接开发板上 SW1:

a) 当计数器复位 SW0 = 0 (off) 时,

如果 SW1=0 (off) 时, 计数器初始值: count = 32'b0;

如果 SW1=1 (on) 时, 计数器初始值: count = 32'hffff_ffff;

b) 当计数器复位 SW1 = 0 (off) 时, 计数器递增计数;

当计数器复位 SW1=1 (on) 时, 计数器递减计数;

计数器的 Verilog 源码如下:

```
// counter.v
module counter( output [11:0] MSB12,
               output [ 3:0] cnt4b,
               input clock,
               input rst_n,
               input dir);

    reg [31:0] temp;
    always @( posedge clock, negedge rst_n )
    begin
        if ( !rst_n ) begin
            temp <= (dir) ? 32'hFFFF_FFFF : 32'b0;
        end
        else begin
            if (!dir) temp <= temp + 1'b1;
            else temp <= temp - 1'b1;
        end
    end
    assign MSB12 = temp[31:20],
           cnt4b = temp[ 3:0];
endmodule
```

(2) 顶层模块:

```
// File: counter_scope.v
`include "clock_div10.v"
`include "counter.v"
module counter_scope( output [7:0] LEDOut, input clock, rst_n, dir );
    wire clk_5MHz;
    clock_div10 m_clk_div(.clk_div10(clk_5MHz), .clock(clock), .rst_n(rst_n) );

    wire [11:0] MSB12;
    wire [ 3:0] cnt4b;
    counter m_cnt( .MSB12(MSB12), .cnt4b(cnt4b),
                  .clock(clk_5MHz), .rst_n(rst_n), .dir(dir) );
    assign LEDOut = MSB12[ 7:0];
    wire [ 3:0] VLED;
    assign VLED = MSB12[11:8];
endmodule
```

(4) 与顶层模块 counter_scope 的同名的用户约束文件:

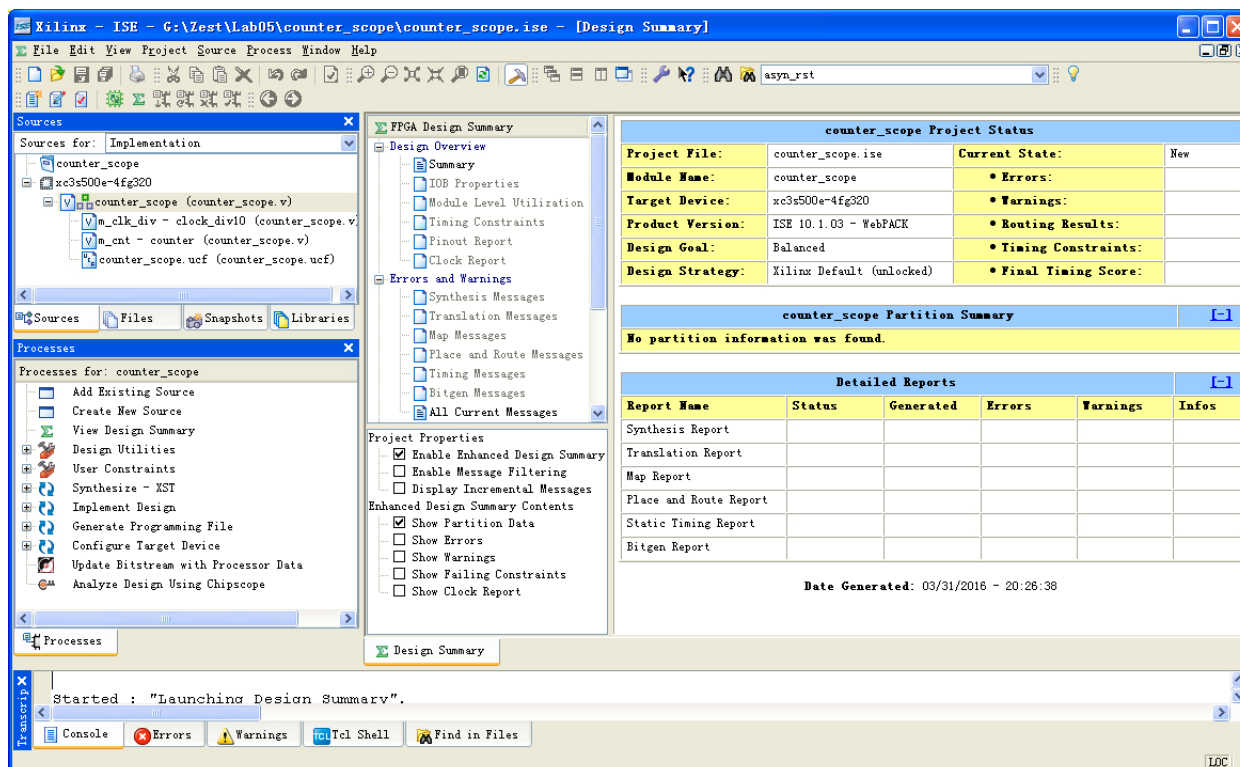
```
NET "clock" LOC = "C9" | IOSTANDARD = LVCMOS33 ;

NET "LEDOut<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<4>" LOC = "C11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;

NET "rst_n" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "dir" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
```

Step 1: 选择一个硬盘分区，在根下在建立一个工程

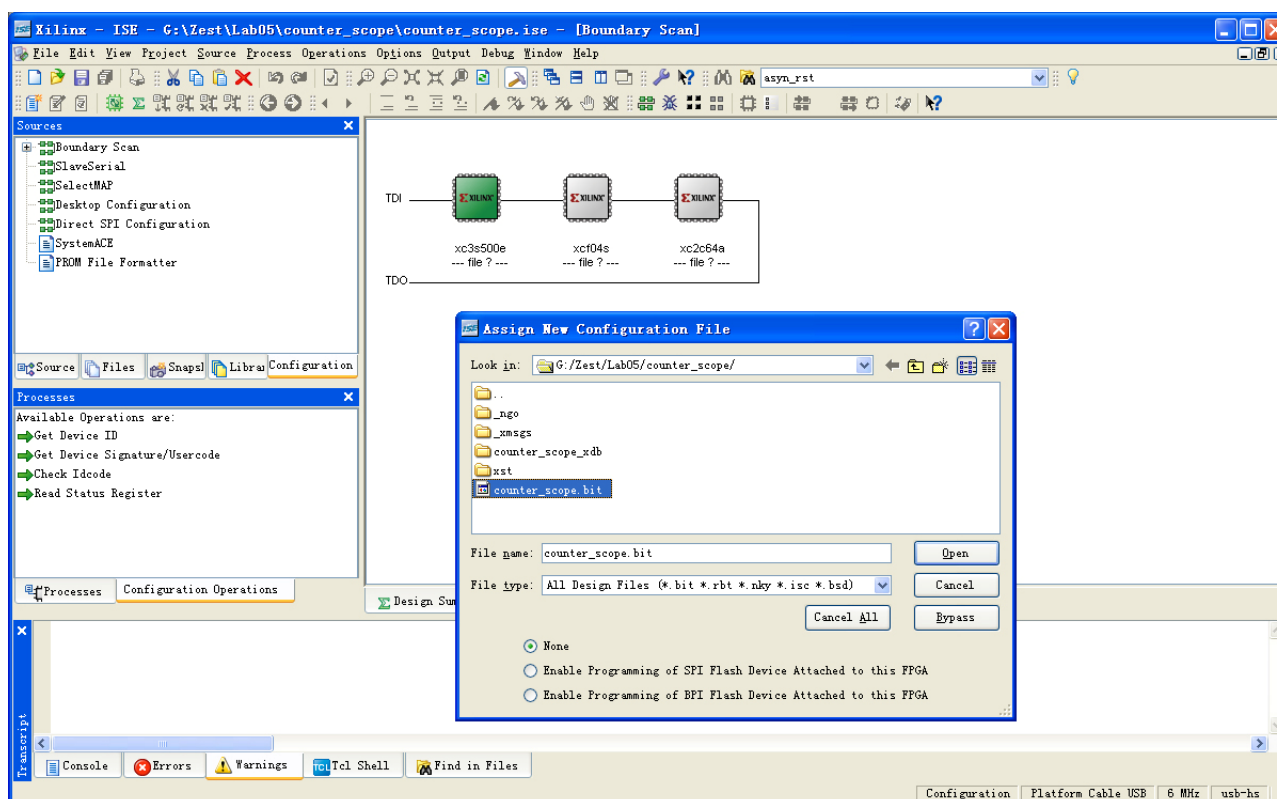
- (1) 如: `X>Lab05\counter_scope\counter_scope`
分别加入顶层模块 `counter_scope.v` 和用户约束文件 `counter_scope.ucf`



- ## （2）综合、实现、产生编程文件

在综合、实现、产生编程文件每一步中，都会产生多条警告信息，产生这些警告信息的原因是，在顶层模块 `counter_scope` 中定义的变量中的一些位没有使用，没有连接任何驱动或连接任何输入。

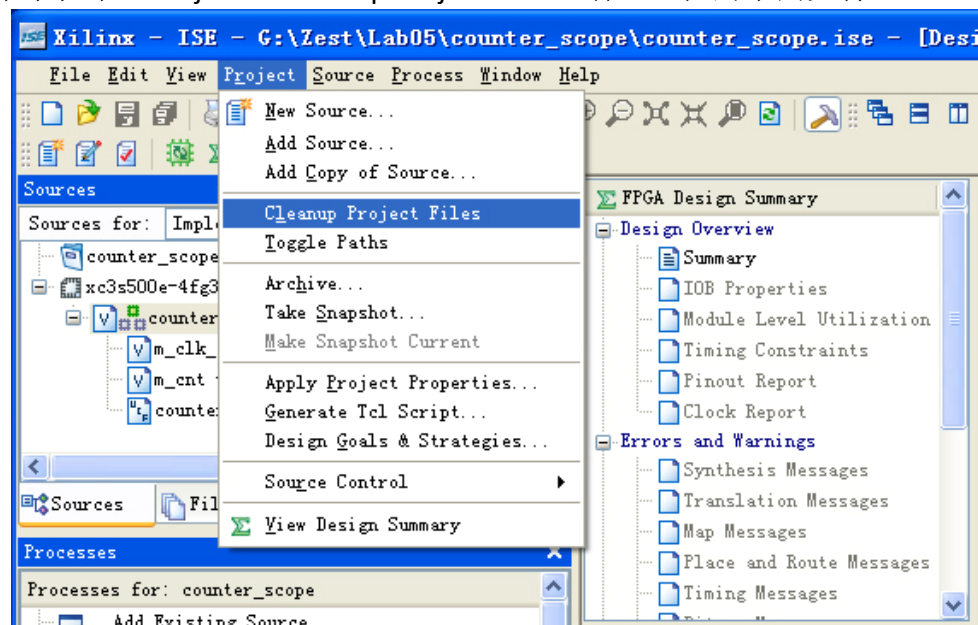
- (3) 打开 iMPACT 软件，选择编程文件 counter_scope.bit 进行下载编程：



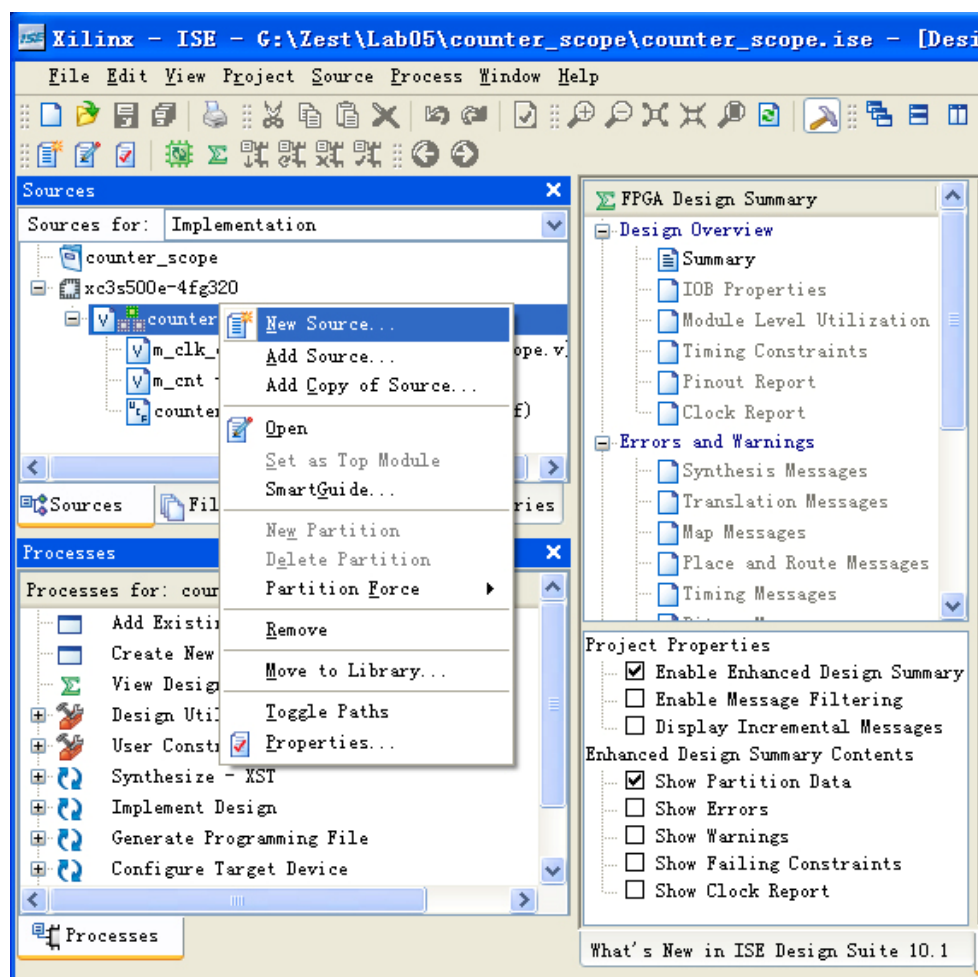
编程完成后，将 SW0 推到打开（on，1'b1）位置，可以观察到，8 个 LED 灯在闪烁计数。表示计数器工作正常。

Step 2: 在 counter_scope 工程中添加用于 ChipScope Pro 分析的模块

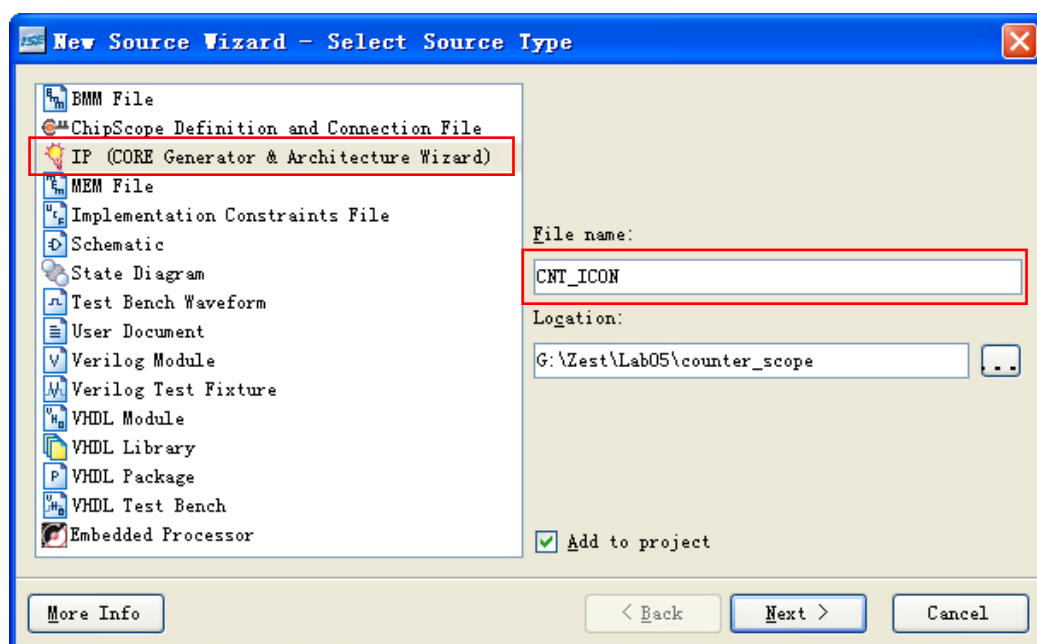
(1) 依次点击菜单栏中：Project → Cleanup Project Files，清理工程的中间文件



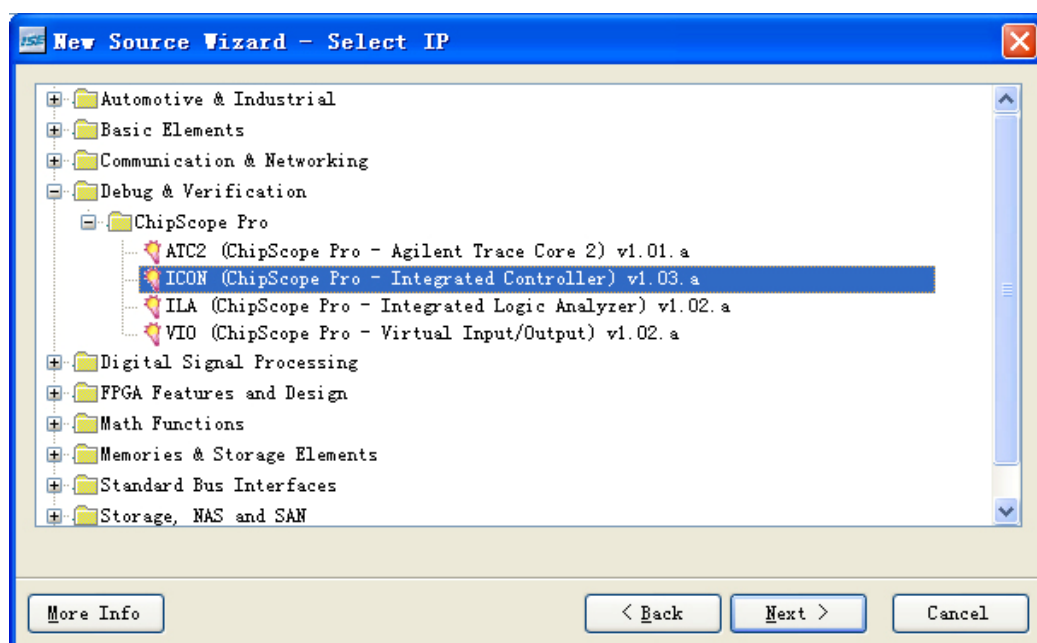
(2) 添加“ICON”模块，右键点击“Sources”窗口，然后选择添加“New Sources”，如下图所示：



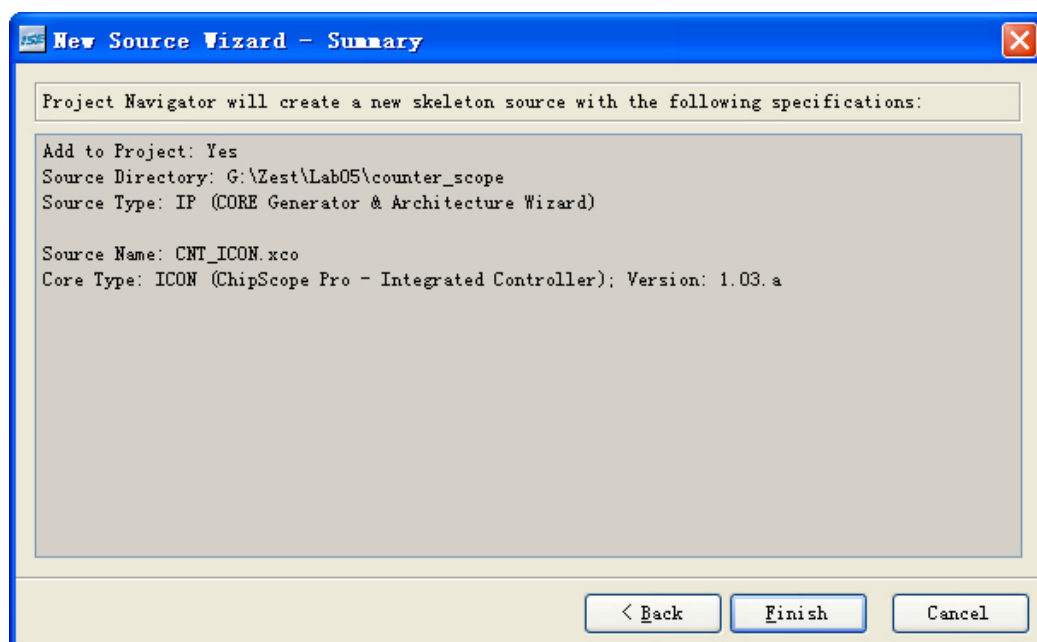
弹出“New Sources Wizard”对话框，选择“IP (CORE Generator & Architectue Wizard)”，文件名为：CNT_ICON，准备插入“集成控制器”模块；



点击“Next”，依次选择：“Debug & Verificaton” → “ChipScope Pro” → “ICON”；

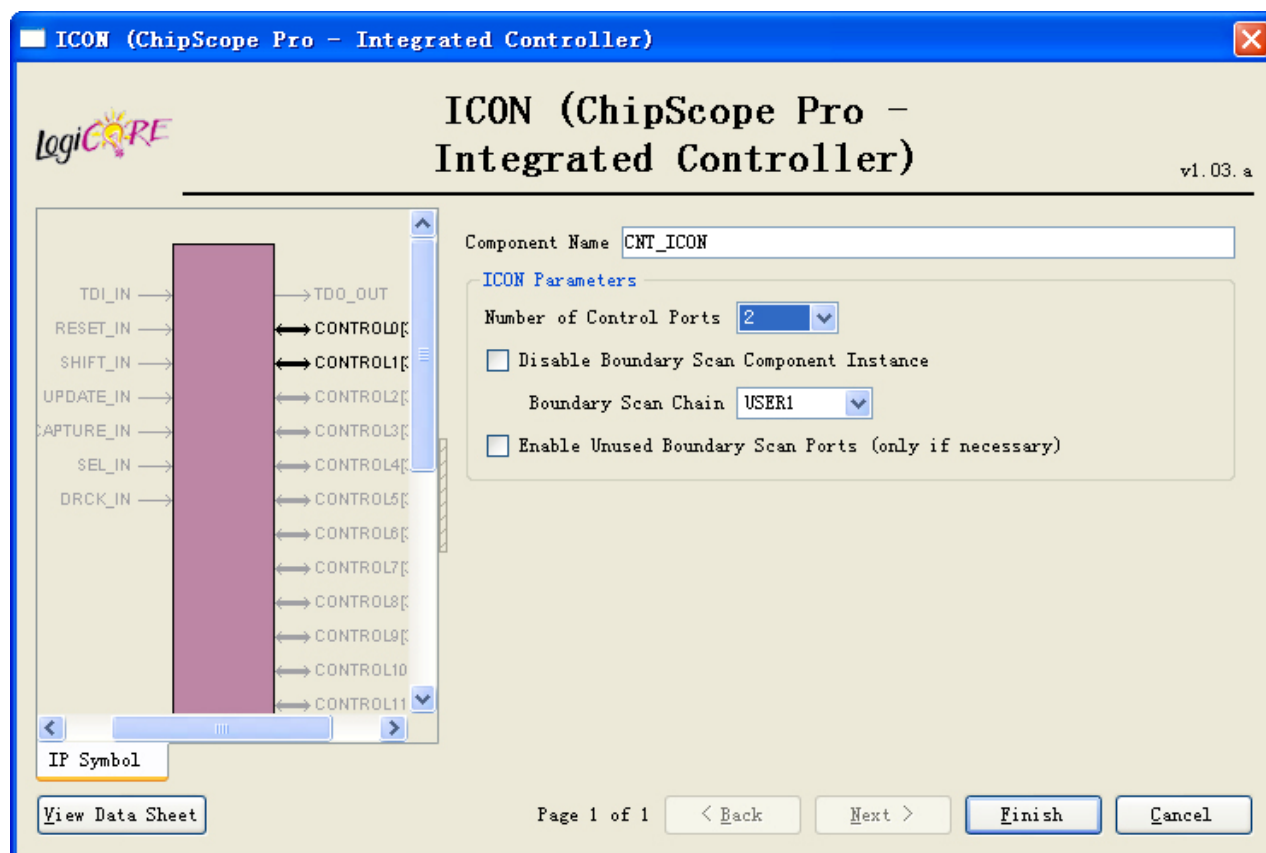


点击“Next”，

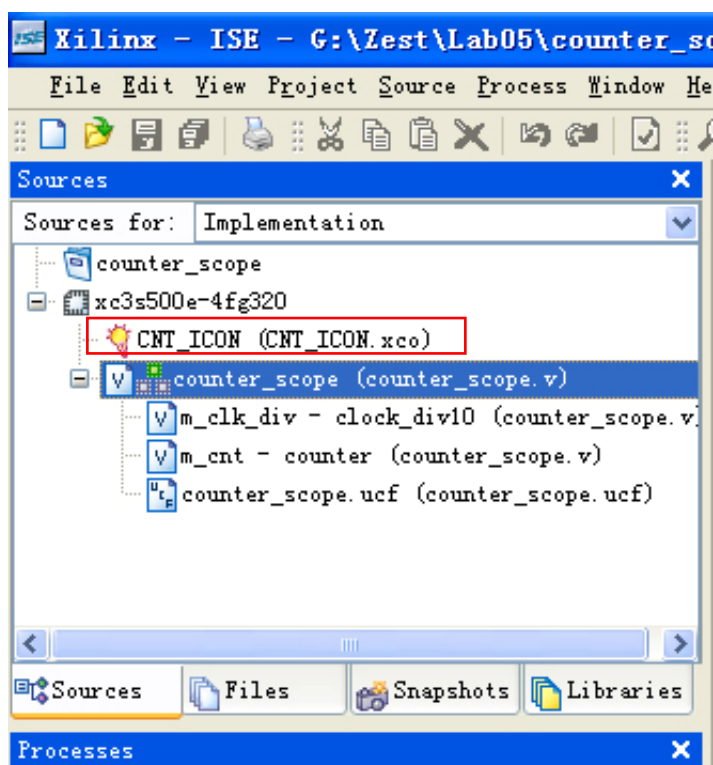


点击“Finish”，ISE 运行一会，出现“ICON (ChipScope Pro Integrated Controller)”对话框；

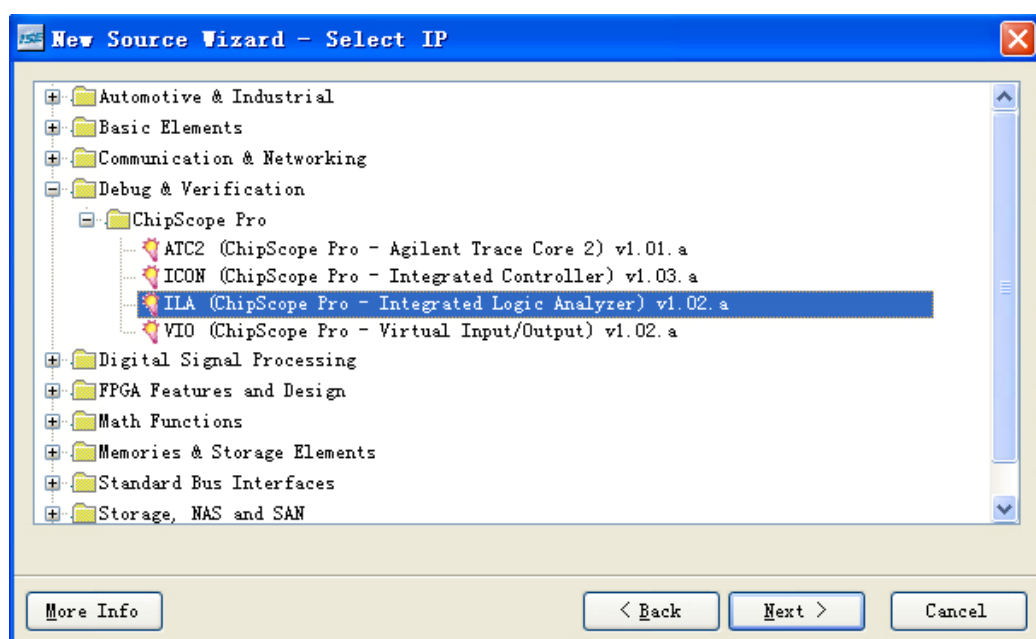
在“ICON (ChipScope Pro Integrated Controller)”对话框中：



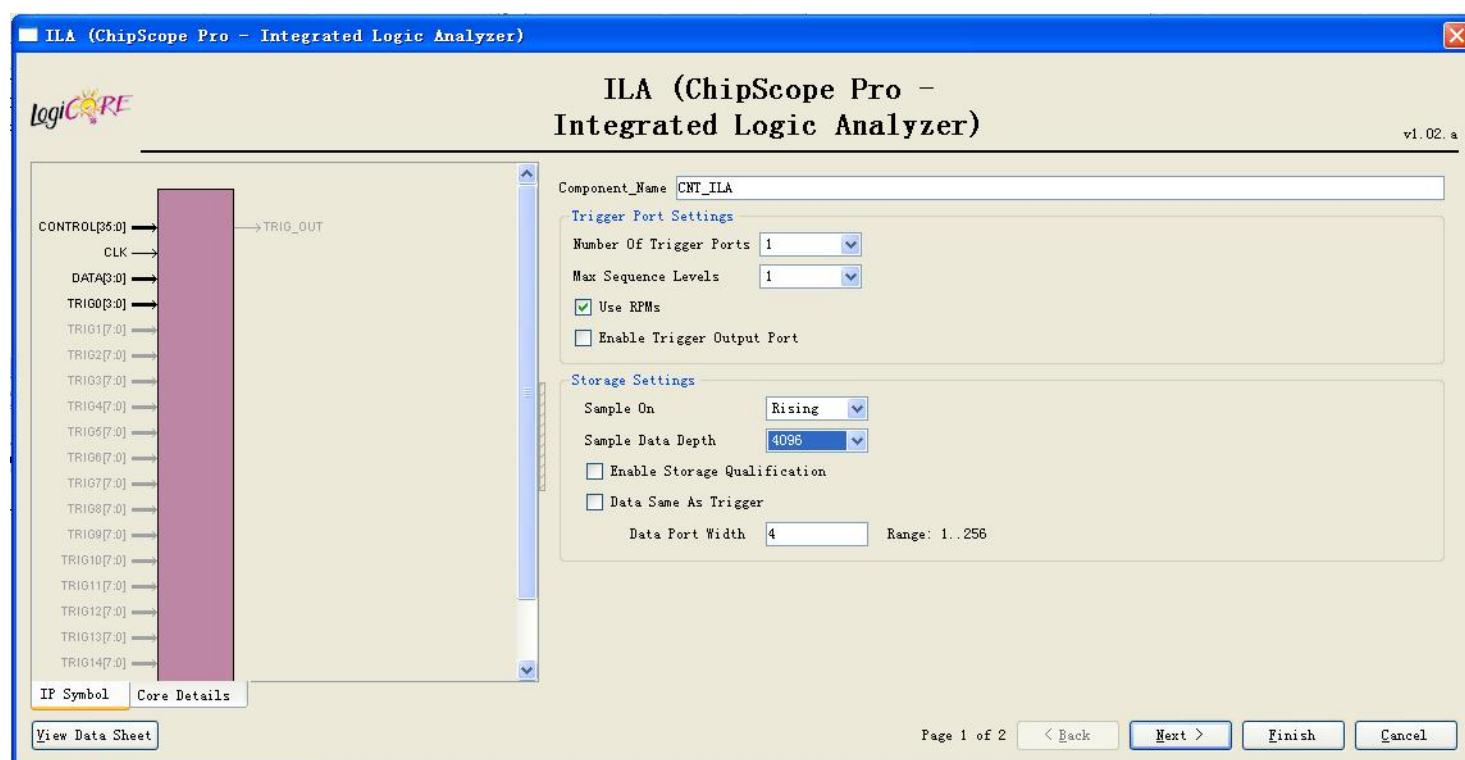
将“Number of Control Ports”改为：2，然后，点击“Finish”。运行结束后，在工程中添加了刚才设定的“CNT_ICON”模块。



(2) 添加“ILA”模块，文件名为：CNT_ILA，依次选择：“Debug & Verification” → “ChipScope Pro” → “ILA”；

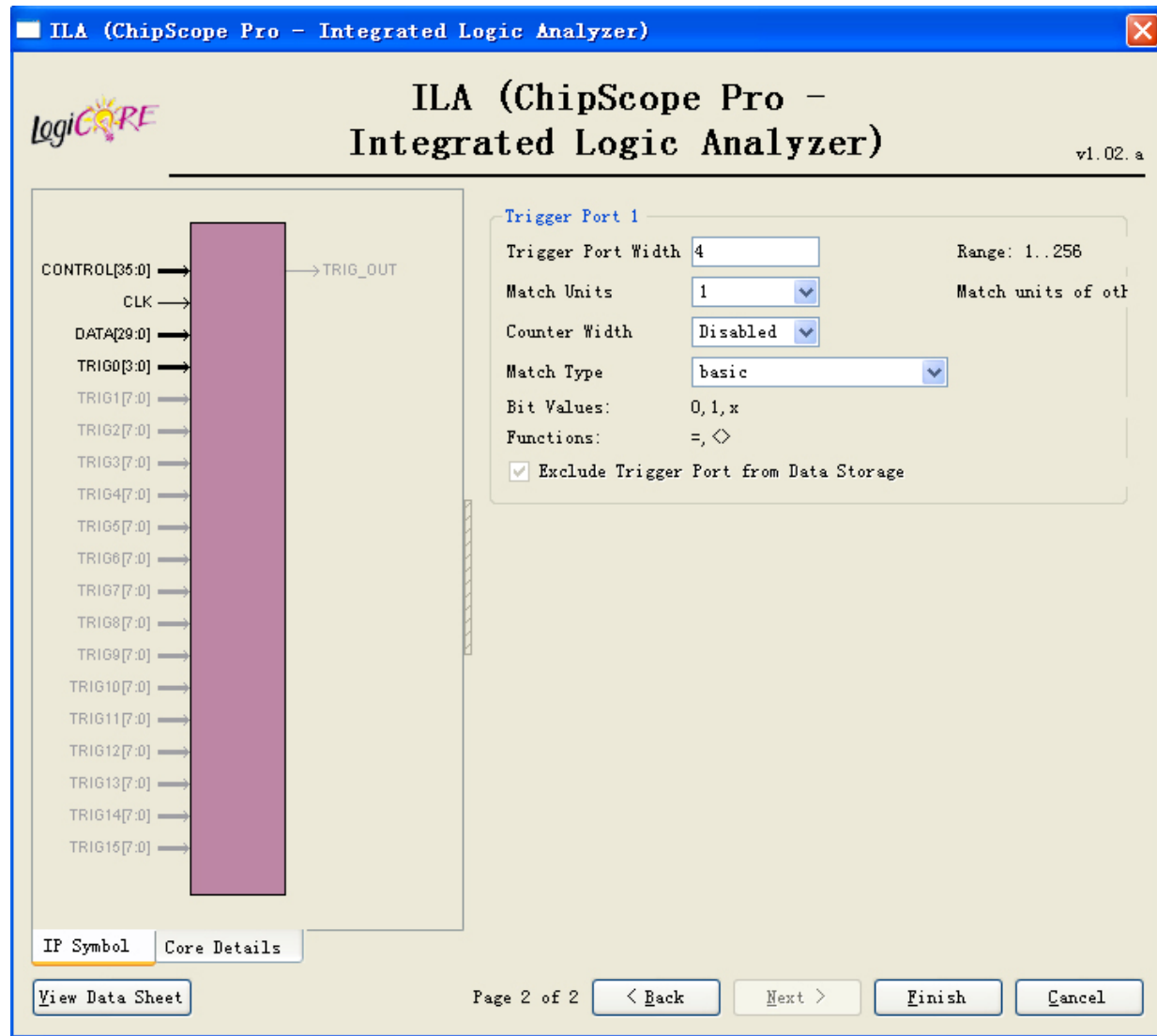


点击“Next”，将“Sample Data Depth”改为：4096，不选（uncheck）“Enable Storage Qualification”，“Data Same As Trigger”，将“Data Port Width”改为：4，如下图：

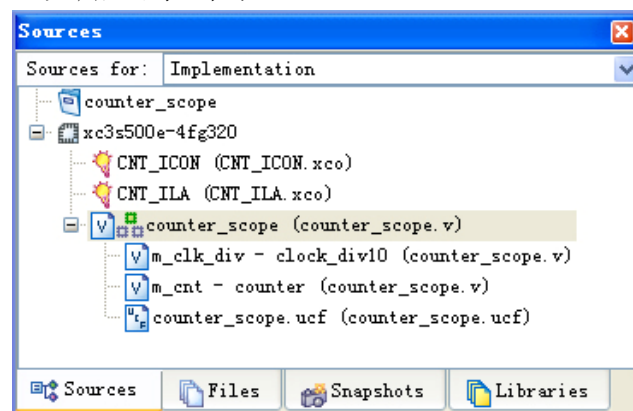


点击“Next”继续设置；

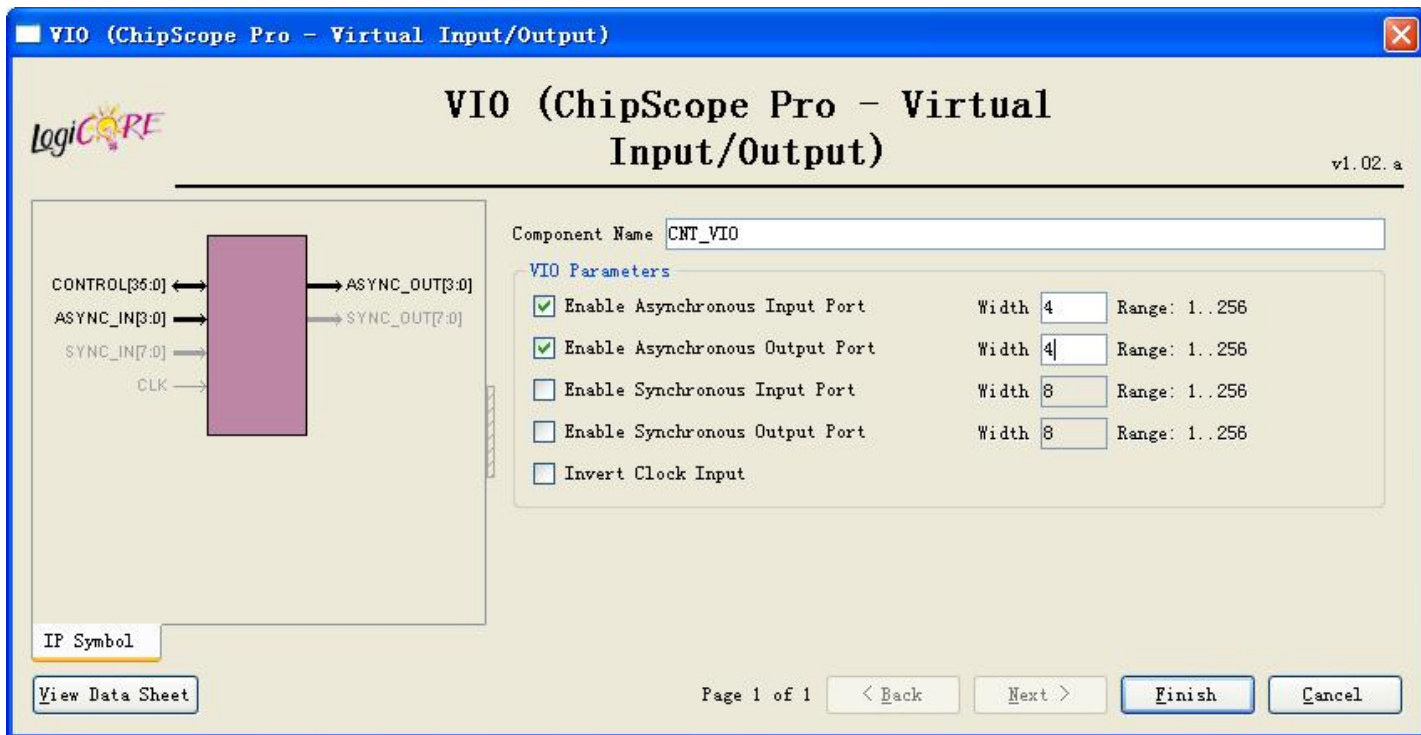
将“Trigger Port Width”设置为：4，然后点击“Finish”；



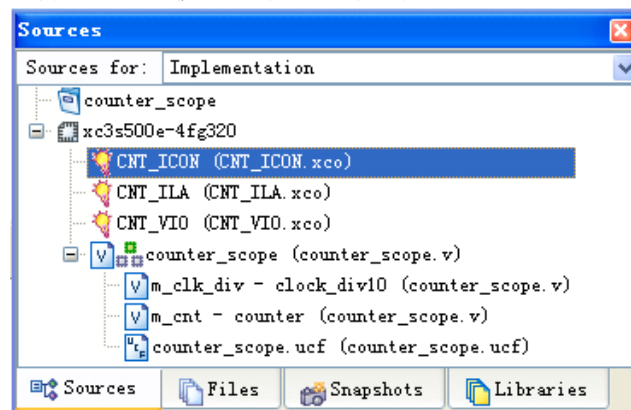
运行一段时间后，CNT_ILA 又加入到工程；



- (3) 添加虚拟输入/输出控制端口“VIO”模块，文件名为：CNT_VIO，依次选择：“Debug & Verificaton”
→ “ChipScope Pro” → “VIO”；
设置“Enable Asynchronous Input Port”为 4，
设置“Enable Asynchronous Output Port”为 4；



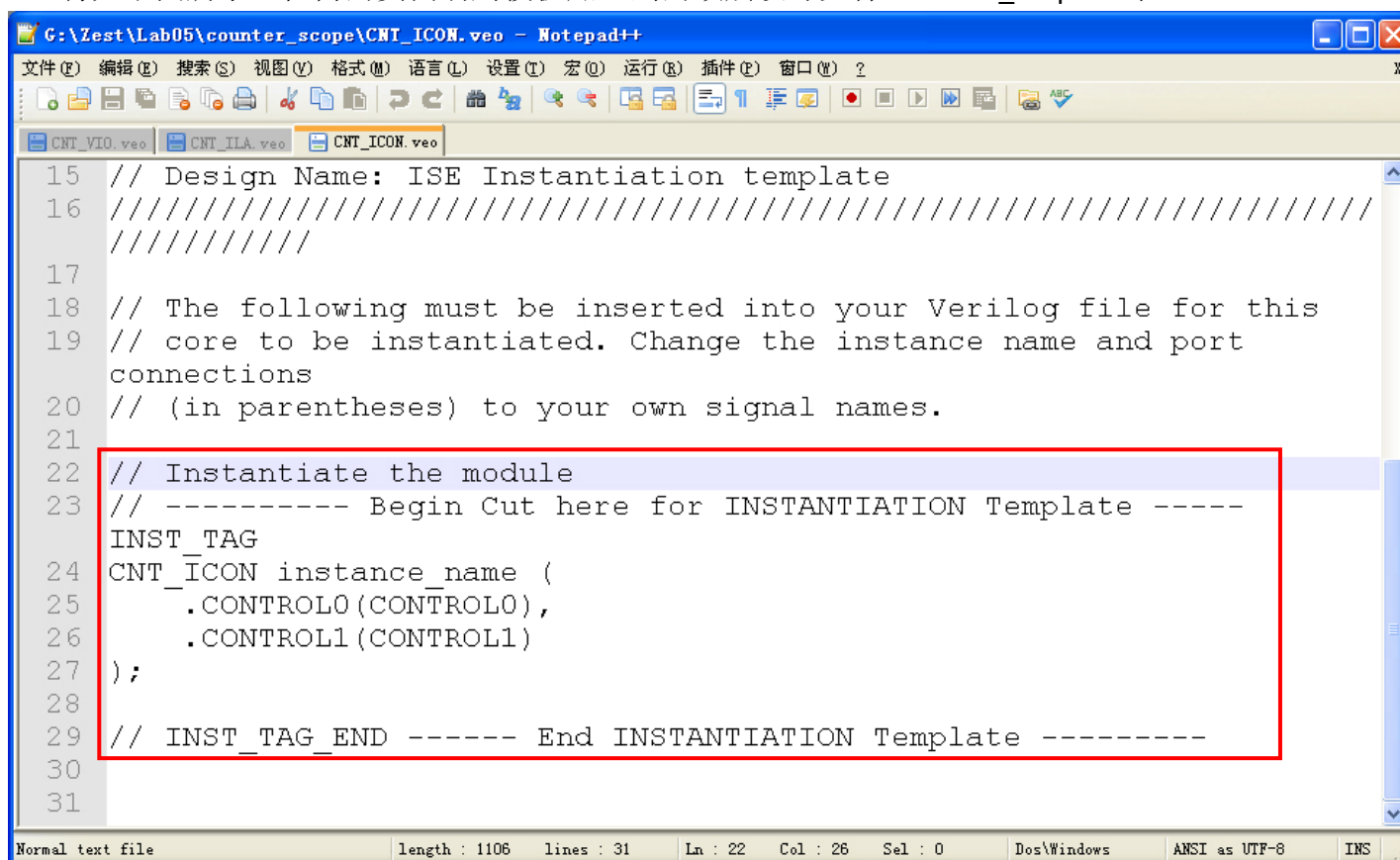
然后，点击“Finish”，运行后，VIO 模块也加入到工程。



接下来，依次将上面生成的三个模块的实例引用加入到工程的顶层模块：counter_scope 中，可以使用任何的文本编辑器，打开工程所在文件夹中的下面三个文件：

CNT_ICON.v eo
CNT_ILA.v eo
CNT_VIO.v eo

将如下图所示红框内的实例引用模板加入到的顶层设计文件 counter_scope.v 中：



```
15 // Design Name: ISE Instantiation template
16 ///////////////////////////////////////////////////////////////////
17
18 // The following must be inserted into your Verilog file for this
19 // core to be instantiated. Change the instance name and port
20 // (in parentheses) to your own signal names.
21
22 // Instantiate the module
23 // ----- Begin Cut here for INSTANTIATION Template -----
24 INST_TAG
25 CNT_ICON instance_name (
26     .CONTROL0(CONTROL0),
27     .CONTROL1(CONTROL1)
28 );
29 // INST_TAG_END ----- End INSTANTIATION Template -----
30
31
```

将上述三个文件中的“Instantiate the module”到“// INST_TAG_END”之间的拷贝到顶层模块中，完成 ICON、ILA、 VIO 实例引用添加的顶层模块文件即如下：

```

// File: counter_scope.v
`include "clock_div10.v"
`include "counter.v"
module counter_scope( output [7:0] LEDOut, input clock, rst_n, dir );
    // 用于连接虚拟复位控制和方向控制输入 din
    wire [1:0] vrst;
    wire [1:0] vdir;
    wire rst_L;
    wire dir_H;
    assign rst_L = rst_n | ( vrst[1] & vrst[0] );
    assign dir_H = dir | ( vdir[1] & vdir[0] );

    wire clk_5MHz;
    clock_div10 m_clk_div(.clk_div10(clk_5MHz), .clock(clock), .rst_n(rst_L) );

    wire [11:0] MSB12;
    wire [ 3:0] cnt4b;
    wire [ 3:0] VLED;
    counter m_cnt( .MSB12(MSB12), .cnt4b(cnt4b), .clock(clk_5MHz),
                  .rst_n(rst_L), .dir(dir_H) );
    assign LEDOut = MSB12[ 7:0];
    assign VLED = MSB12[11:8];

    // 定义两个连接控制端口的 36 位 wire 变量
    wire [35:0] CONTROL0; // 用于控制逻辑分析仪 CNT_ILA
    wire [35:0] CONTROL1; // 用于控制虚拟输入/输出口 CNT_VIO
    // Instantiate the module
    // ----- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ICON m_icon ( // 这里需要修改实例名
        .CONTROL0(CONTROL0),
        .CONTROL1(CONTROL1)
    );
    // INST_TAG_END ----- End INSTANTIATION Template -----

    // 定义连接触发采样端口的 4 位 wire 变量, 使用计数器输出最高 4 位触发
    wire [3:0] trig = VLED; // MSB12[11:8]
    // Instantiate the module
    // ----- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ILA m_ila ( // 这里需要修改实例名
        .CONTROL(CONTROL0),
        .CLK(clock), // 这里连接板上 50MHz 时钟
        .DATA(cnt4b), // 这里连接最低 4 位计数器输出
        .TRIG0(trig)
    );
    // INST_TAG_END ----- End INSTANTIATION Template -----

    // 定义虚拟复位和方向控制输入
    wire [3:0] din;
    // Instantiate the module
    // ----- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_VIO m_vio ( // 这里需要修改实例名
        .CONTROL(CONTROL1), // 这里使用 CONTROL1
        .ASYNC_IN(VLED), // 连接虚拟 LED, 即 VLED 连接 MSB12[11:8] 4 位

```

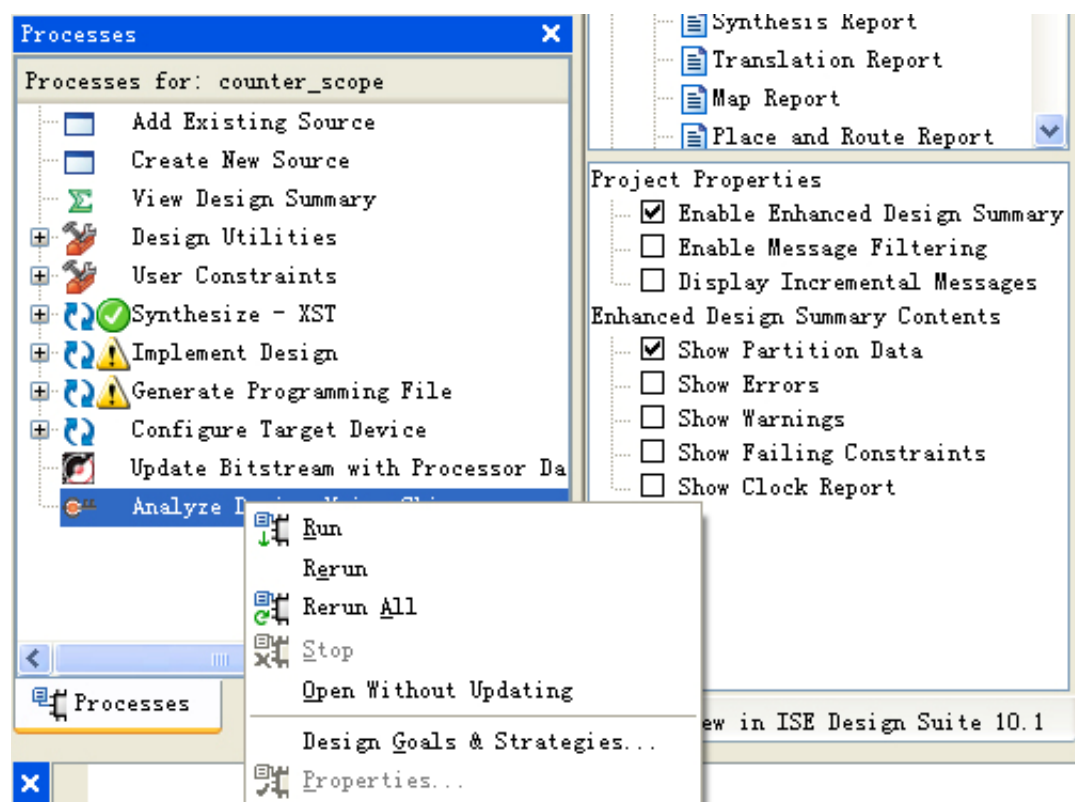
```

        .ASYNC_OUT(din)           // 连接虚拟复位控制和方向控制 { vrst, vdir }
    );
    // INST_TAG_END ----- End INSTANTIATION Template -----
    // 连接虚拟复位控制和方向控制 { vrst, vdir }
    assign vrst = din[3:2];
    assign vdir = din[1:0];
endmodule

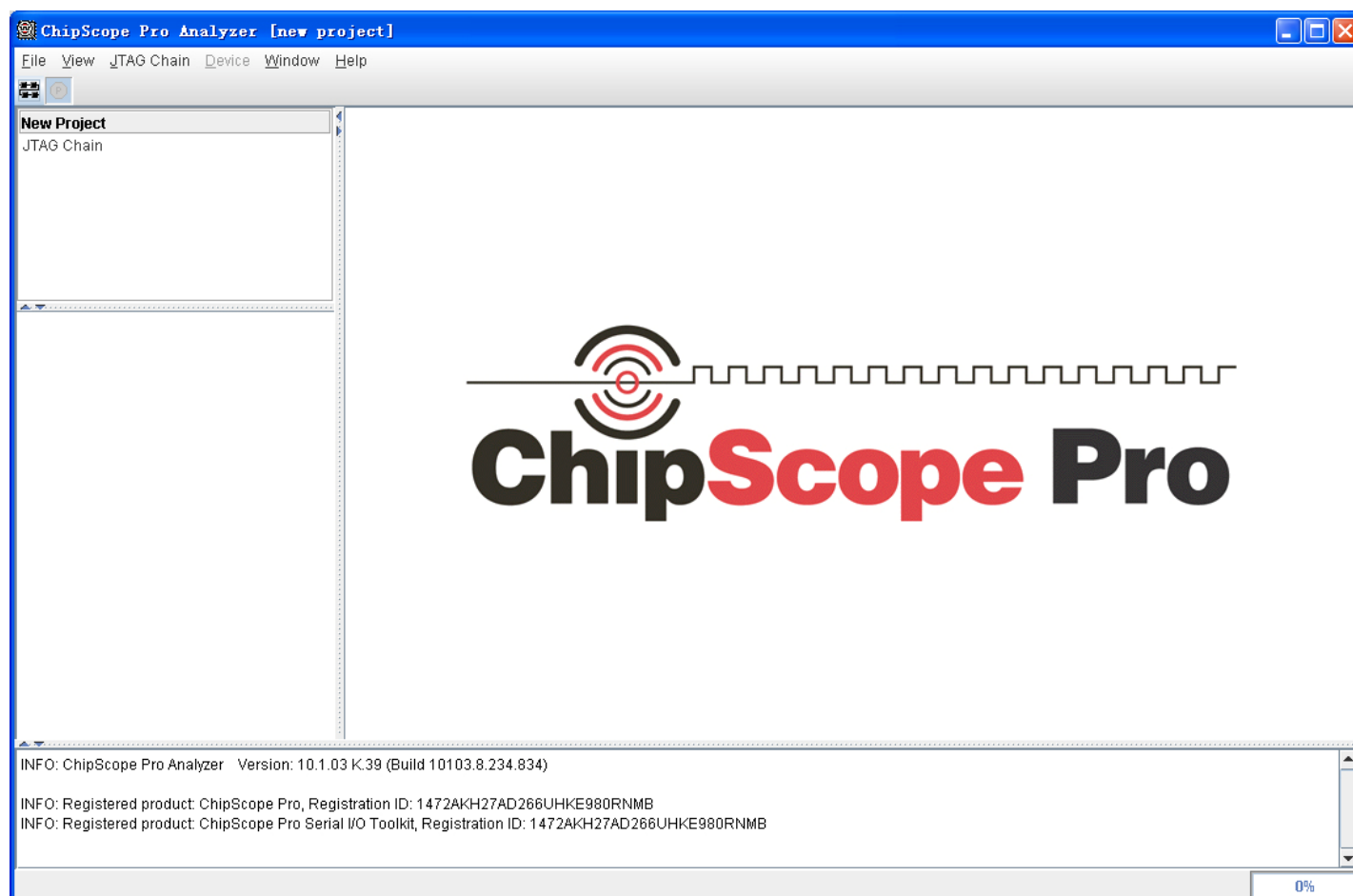
```

Step 3: 使用 ChipScope Pro 调试分析设计

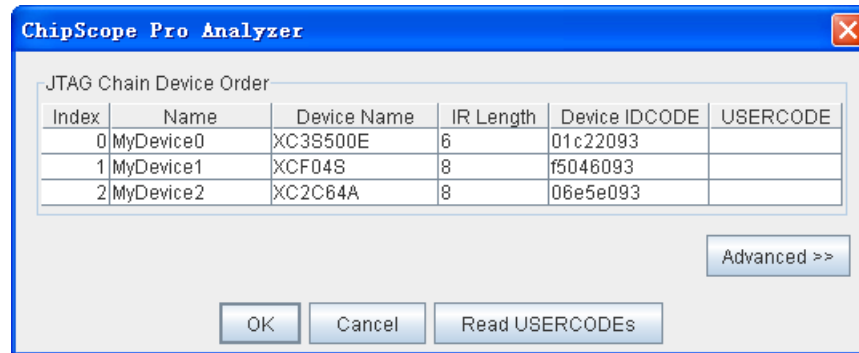
(1) 综合、实现、产生编程文件，然后，运行 ChipScope Analyzer，选择 Analyze Design Using ChipScope，右键点击 Run：



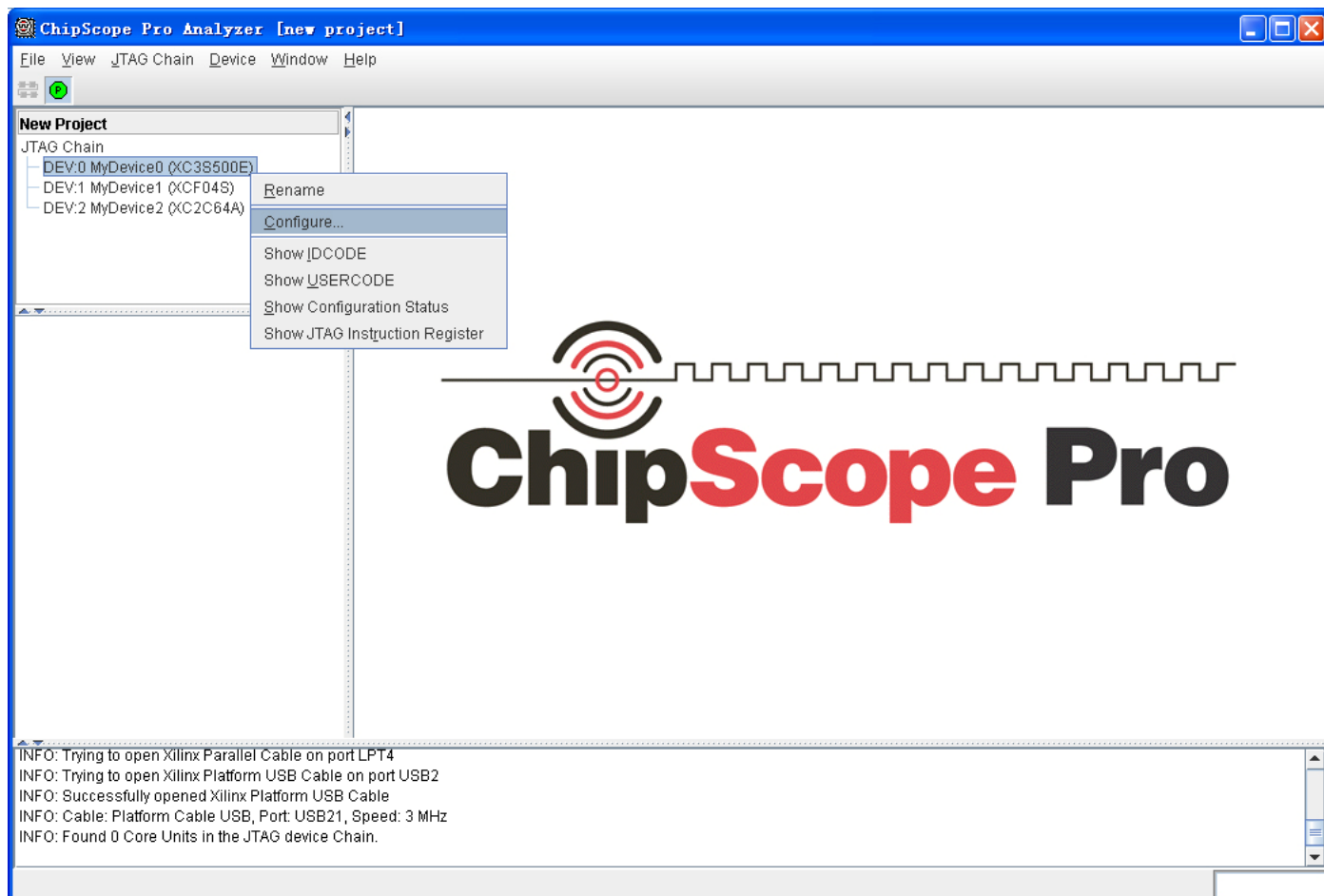
ChipScope Pro Analyzer 启动后，界面如下图所示。



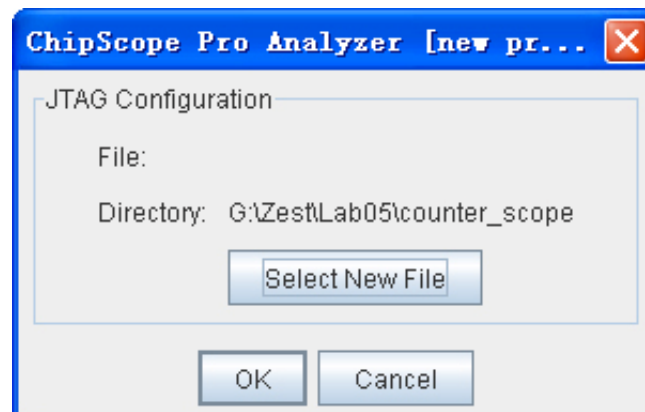
(2) 配置目标芯片，在常用工具栏上点击图标：，初始化边界扫描链，成功完成扫描后，将会列出 JTAG 链上的器件。选择我们使用的开发板 FPGA 芯片型号 XC3S500E。



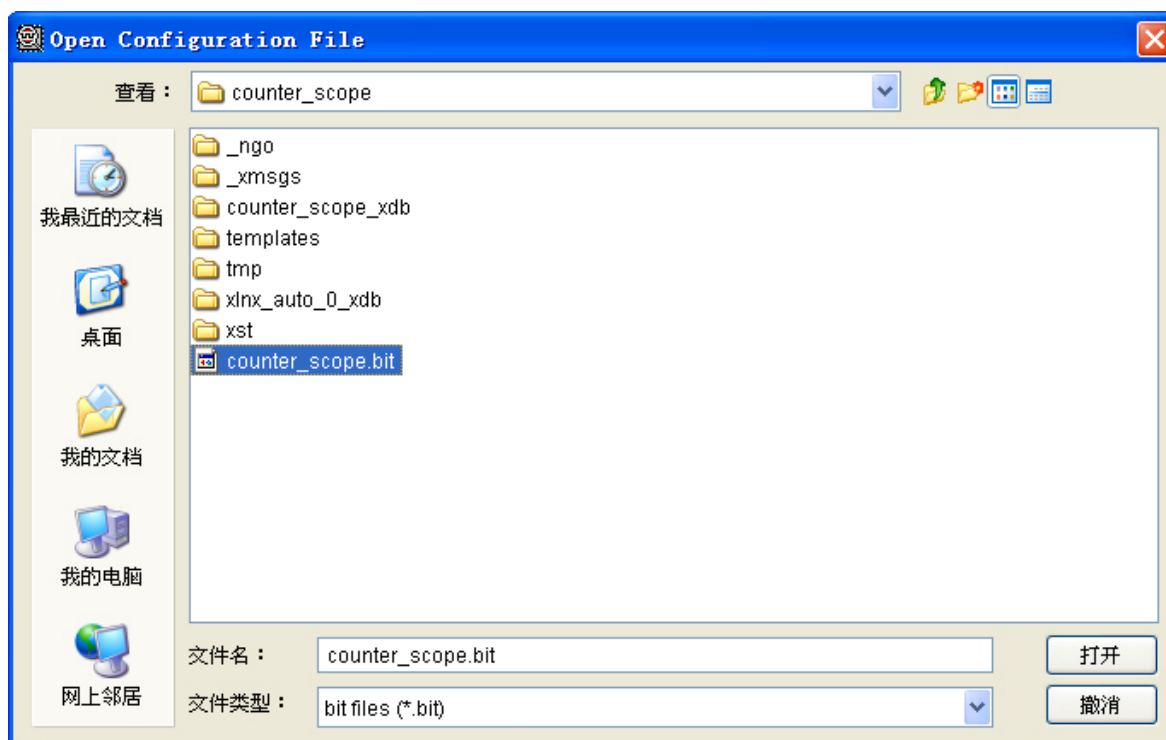
点击“DEV:0 MyDevice0 (XC3S500E) → Configure”进行配置。



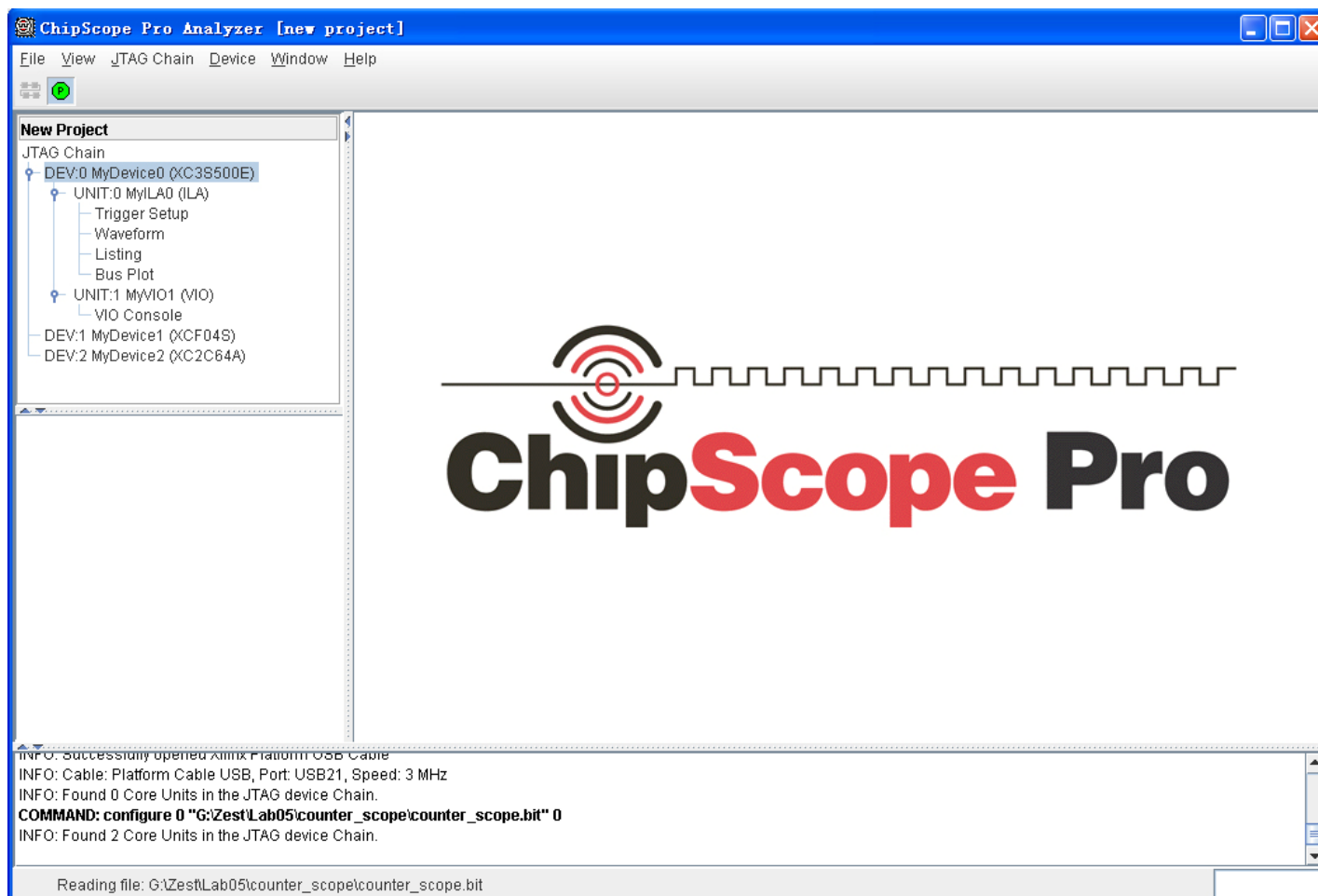
点击选择“Select New File”



在弹出的配置对话框中，选择需要下载的.bit 文件。需要注意的是：ChipScope 利用 JTAG 链来观察芯片内部逻辑，因此在生成配置文件时只能利用.bit 格式的配置文件。



使用编程文件 “counter_scope.bit” 对 Spartan-3E XC3S500E 配置后，就可以对刚刚完成的设计进行在线调试与分析了。



(3) 设置触发条件

把 ChipScope 设计和工程下载到 FPGA 中以后，还需要设定触发条件才能在 Analyzer 中捕获到有效波形。Analyzer 的触发设置由 Match（匹配）、Trig（触发）以及 Capture（捕获）三部分。其中 Match 用于设置匹配函数，Trig 用于把一个或多个触发条件组合起来构成最终的触发条件，Capture 用于设定窗口的数目和触发位置。

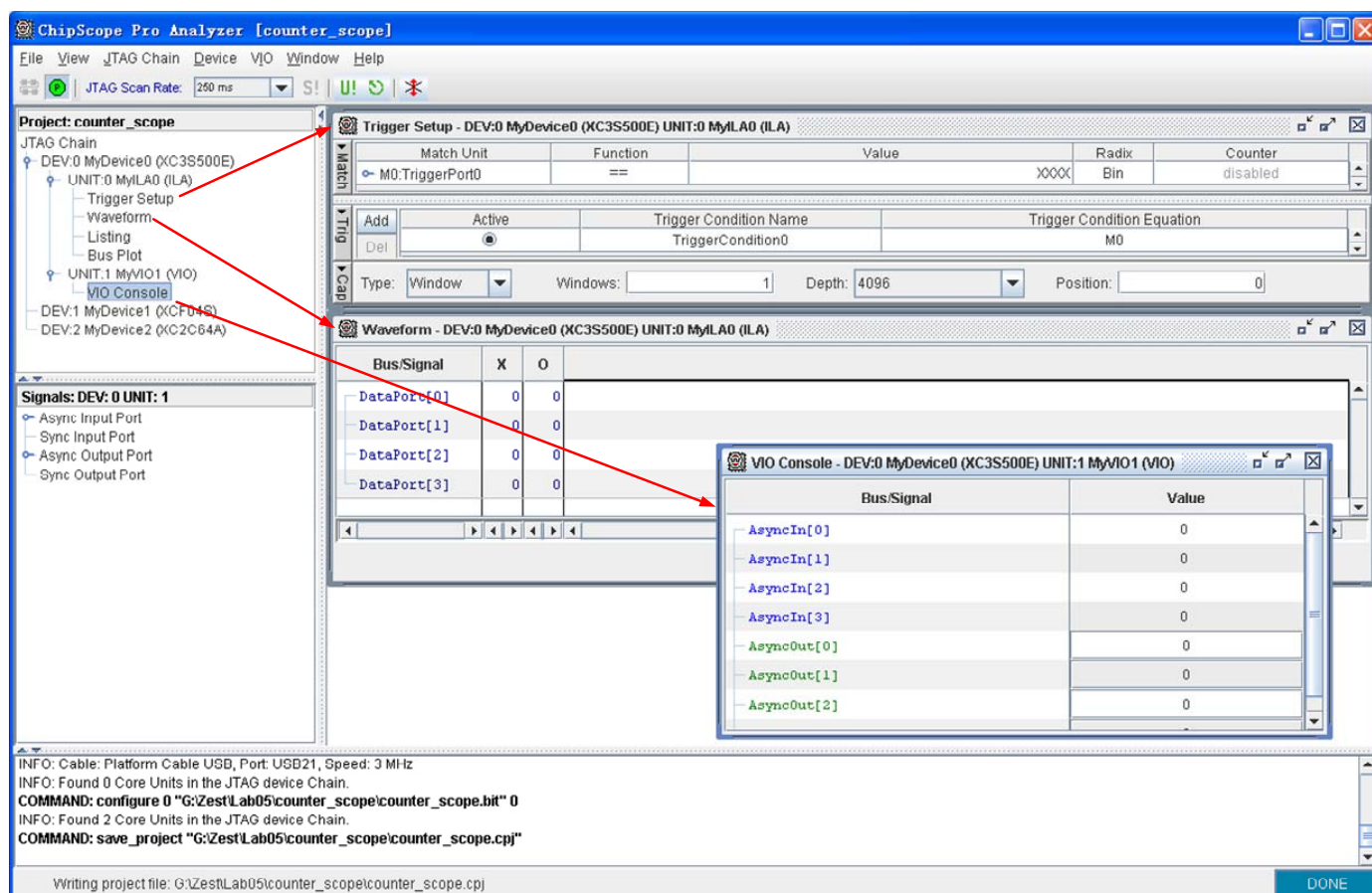
双击 Trigger Setup，典型的配置界面如下图所示。

(4) 观察信号波形

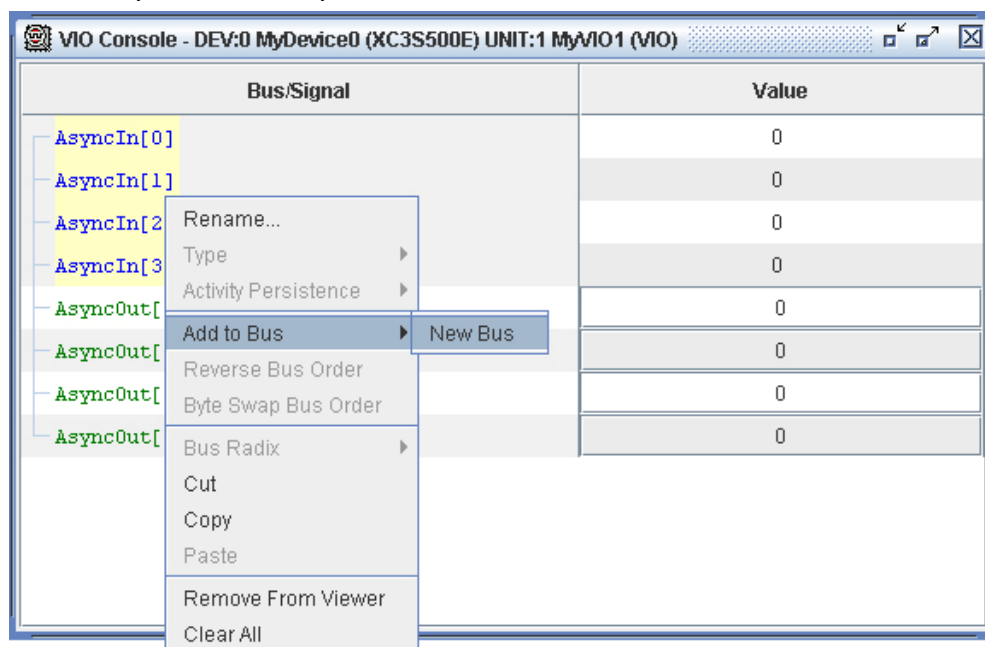
观察信号波形需要打开 Waveform 窗口，双击 Waveform 命令，显示界面如下图所示。

(5) 使用虚拟控制台

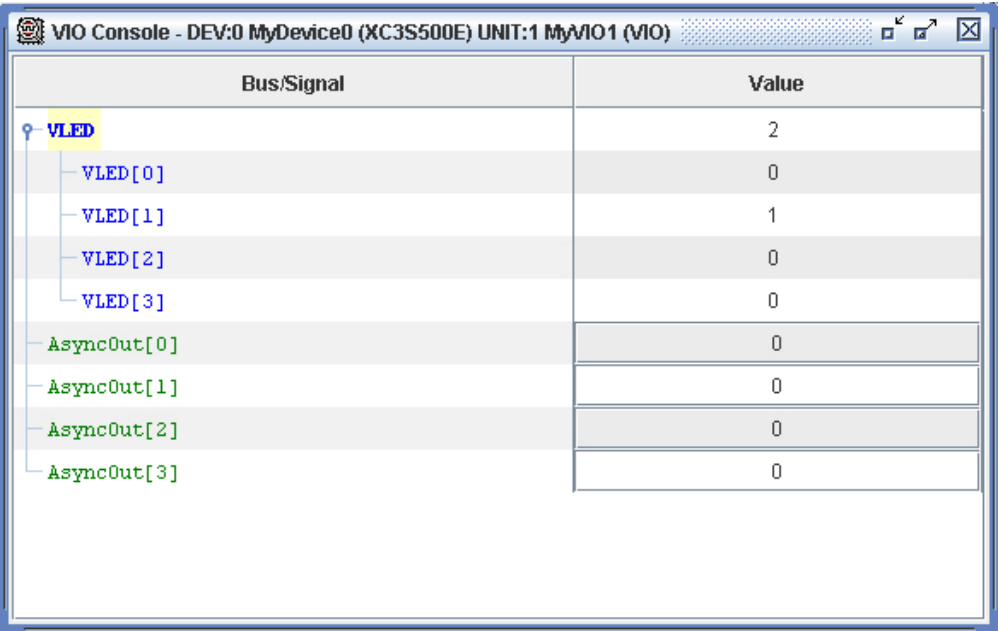
双击 VIO Console 命令，弹出虚拟控制台窗口如下图所示。



在 VIO 窗口选择：AsyncIn[0] ~ AsyncIn[4]，点击右键，Add to Bus → New Bus，



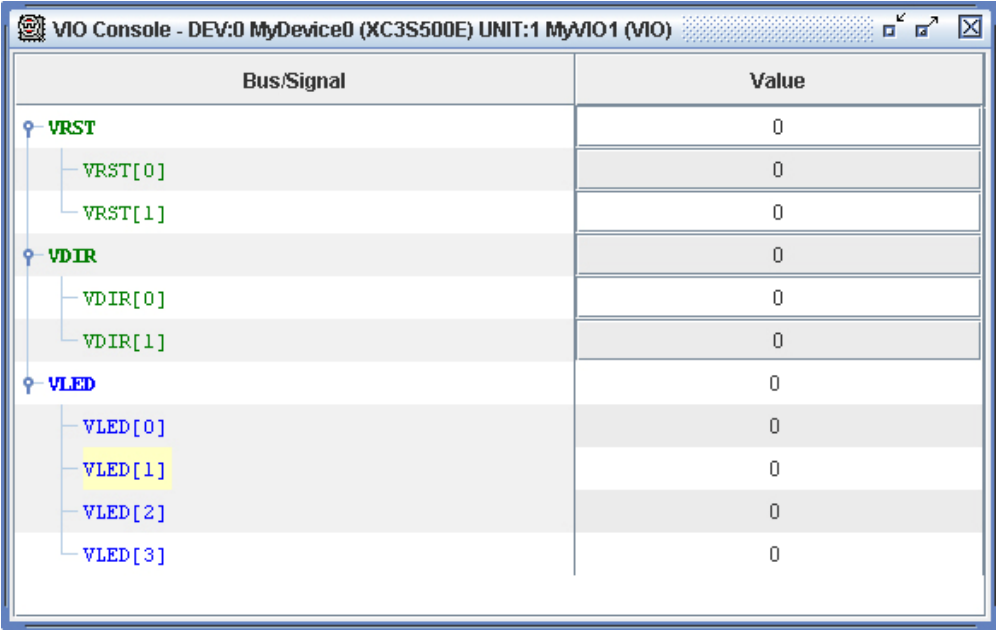
新总线的取名“VLED”，然后，将其中的每一位的“AsyncIn”重命名为 VLED，然后，将开发板上的 SW0 设为 on（1'b1），启动计数，运行一段时间，VIO Console 上的 VLED 也如同开发板上的 LED 一样开始变化，如下图所示：



The screenshot shows the VIO Console window for MyDevice0 (XC3S500E) UNIT:1 MyVIO1 (VIO). It displays a table of bus signals and their values. The 'VLED' bus is expanded, showing four bits: VLED[0] (0), VLED[1] (1), VLED[2] (0), and VLED[3] (0). Below it, four 'AsyncOut' signals are listed, all with a value of 0.

Bus/Signal	Value
VLED	2
VLED[0]	0
VLED[1]	1
VLED[2]	0
VLED[3]	0
AsyncOut[0]	0
AsyncOut[1]	0
AsyncOut[2]	0
AsyncOut[3]	0

计数器复位，然后继续添加新总线，将 AsyncOut[2] 、 AsyncOut[3]添加到总线 VRST，并将改名为 VRST[0]和 VRST[1]，将 AsyncOut[0]、 AsyncOut[1]添加到总线 VDIR，并改名为 VDIR[0]、VDIR[1]。如下图所示：



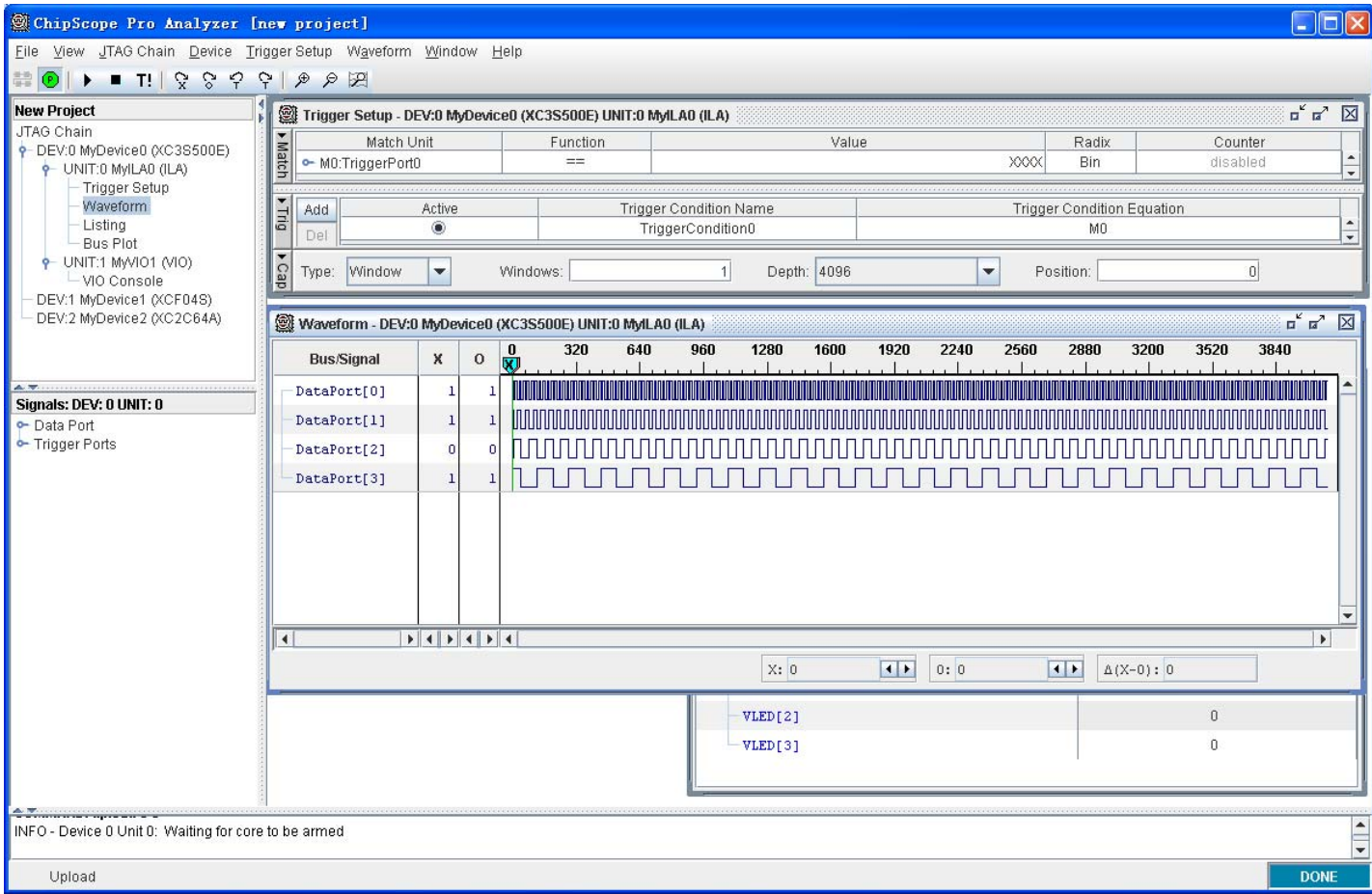
The screenshot shows the VIO Console window with three buses: VRST, VDIR, and VLED. VRST has two bits, VRST[0] and VRST[1], both with a value of 0. VDIR has two bits, VDIR[0] and VDIR[1], both with a value of 0. VLED has four bits, VLED[0] (0), VLED[1] (0), VLED[2] (0), and VLED[3] (0). The VLED[1] row is highlighted in yellow.

Bus/Signal	Value
VRST	0
VRST[0]	0
VRST[1]	0
VDIR	0
VDIR[0]	0
VDIR[1]	0
VLED	0
VLED[0]	0
VLED[1]	0
VLED[2]	0
VLED[3]	0

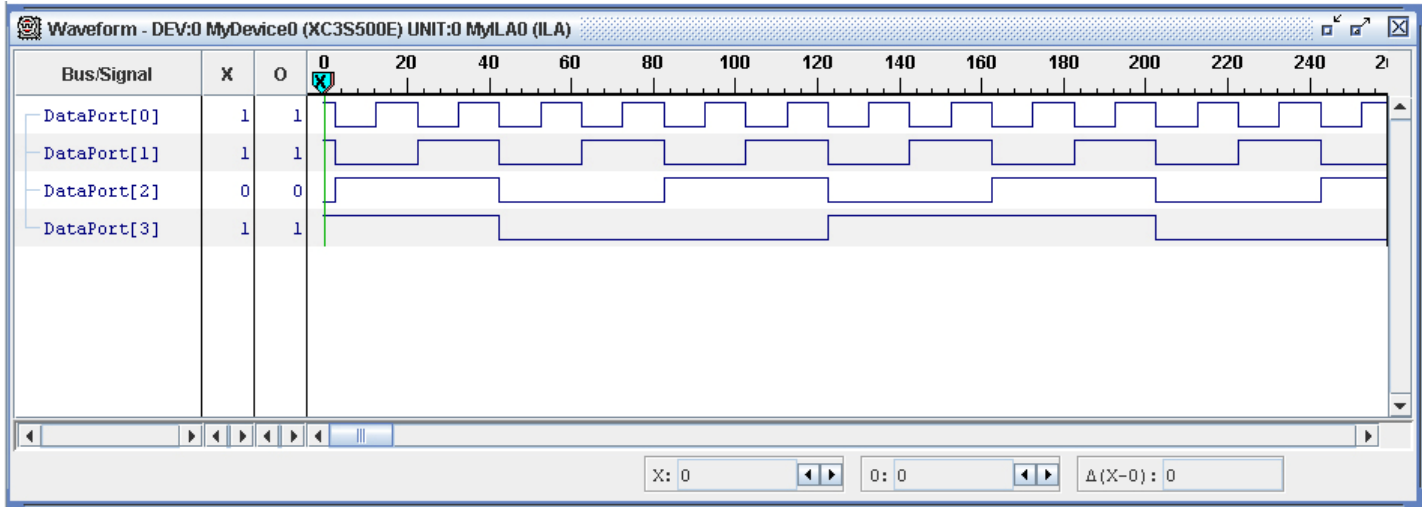
直接在 VIO 窗口中点击 VRST[0]和 VRST[1]，将其值修改为 1，此时，计数器启动开始计数。将 VRST[0]和 VRST[1]的值修改为 0 后，计数器停止。

到此为止，虚拟复位输入，虚拟 LED 显示输出，均设置完成。

(6) 不使用触发条件采集数据，先将计数器复位，注意，板上的复位键 SW0 和虚拟复位键对，均要在复位位置，然后将“Waveform”置前，点击菜单栏上的 **T!** 按钮，开始采集数据。




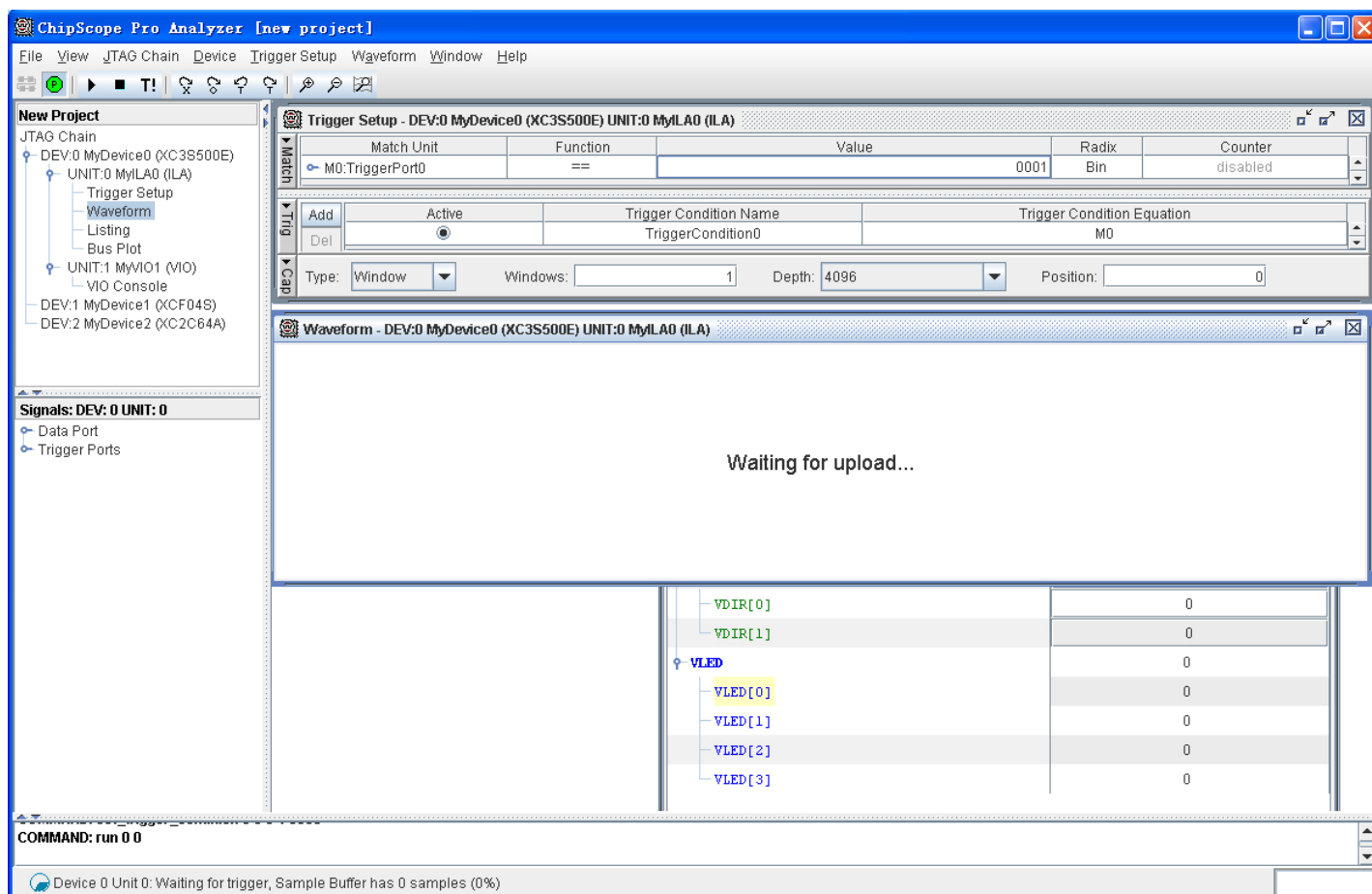
放大波形，如下图，观察，分频后时钟周期是否为 5MHz？为什么？



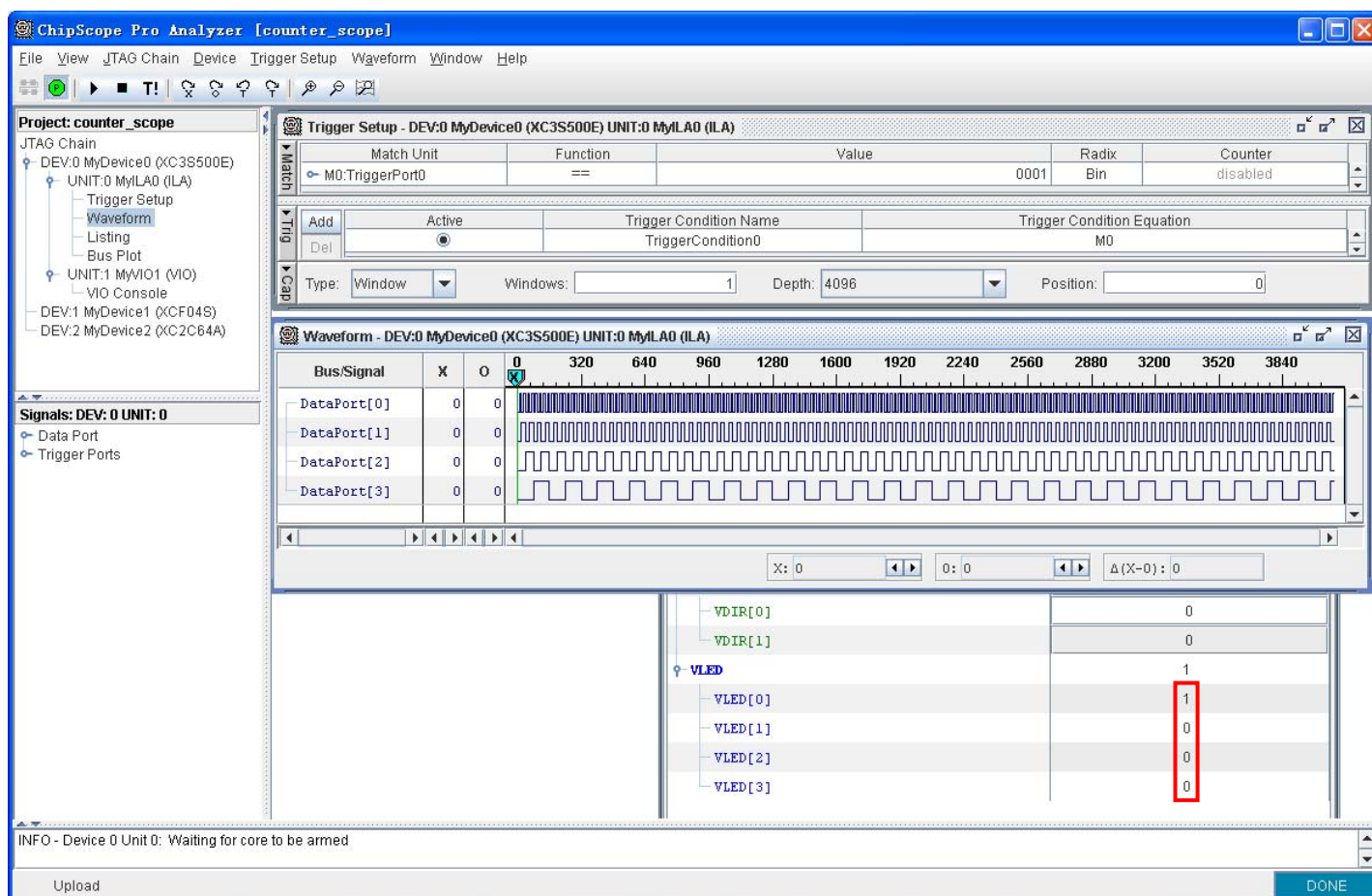
(7) 设定触发条件采集数据

重新复位计数器, 在 Trigger Setup 栏 Match 区域的 M0:Trigger Port0 行的 Value 列输入触发条件: 0001。

点击: “” 采集数据, 然后启动计数器。逻辑分析仪等待触发条件, 即 $MSB_{12}[11:8]=4'B0001$, 条件满足时, 开始采样。



当 $MSB_{12}[11:8]=4'B0001$ 时:



至此, 使用 ChipScope Pro 分析参考设计全部流程完成。

实验五

试验名称：使用 ChipScope Pro 分析 6 位计数器

1、功能描述：

(0) 设计一个 7 分频电路模块，将开发板的时钟信号分频，获得的 $\text{clk_div7} = 50\text{MHz}/7$ 的时钟信号；

```
module clock_div7(output clk_div7, input clock, input rst_n );
```

这里， clk_div7 是分频时钟信号输出；

clock 是系统时钟输出；

rst_n 是异步复位；

(1) 设计一个 6 位计数器，在分频获得的频率近似为 $F = 7142857.14\text{Hz}$ 的时钟信号控制下，进行计数。该计数器的输入/输出端口如下：

i)、输出端口：

count —— 6 位计数器，连接 Spartan - 3E FPGA Starter Kit Board 上的 8 个 LED：LED7 ~ LED2。

ii) 输入端口：

clock —— 连接 50MHz 的时钟晶振 (C9)；

rst_n —— 异步复位输入端口，低电平 (1'b0) 有效；连接开发板上 SW0，当 $\text{SW0} = 0$ (off) 时，计数器复位。

dir —— 计数方向控制，高电平 (1'b1) 有效，连接开发板上 SW1；

a) 当计数器复位 $\text{SW0} = 0$ (off) 时，

如果 $\text{SW1} = 0$ (off) 时，计数器初始值： $\text{count} = 6'b0$ ；

如果 $\text{SW1} = 1$ (on) 时，计数器初始值： $\text{count} = 6'b'h3f$ ；

b) 当计数器复位 $\text{SW1} = 0$ (off) 时，计数器递增计数；

当计数器复位 $\text{SW1} = 1$ (on) 时，计数器递减计数；

```
module counter( output [5:0] count, input clock, input rst_n, input dir);
```

2、使用 ChipScope Pro 分析设计：

在设计添加 ICON 和 ILA。这里，ILA 的“Sample Data Depth”改为：2048，“Data Port Width”为：6，与 count 相连，“Trigger Port Width”设置为：2，与 $\text{count}[5:4]$ 相连。

3、设计中需要解决的问题：

(1) 设计 7 分频电路模块，**提示：使用移位寄存器**

(2) 插入 ICON 和 ILA 模块，注意端口位宽和正确连接；

(3) 使用 ChipScope Pro 观察分析采样信号，以验证 7 分频电路正确工作。