# 2018《FPGA 应用实验》实验报告

实验编号：05　　实验时间：2018.04.10

实验名称：使用 ChipScope Pro 分析 6 位计数器

班级：F1503003　学号：515030910067　姓名：杨超琪

## 1、实验平台

采用 Xilinx 公司的 FPGA 集成开发环境 Xilinx ISE Design Suite 10.1 sp3，实验开发板为 Xilinx Spartan-3E FPGA Starter Kit。

## 2、实验设计要求：

（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 clk_div7 = 50MHz/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= 7142857.14Hz 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；连接开发版上 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，连接开发版上 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 = 6’b0；

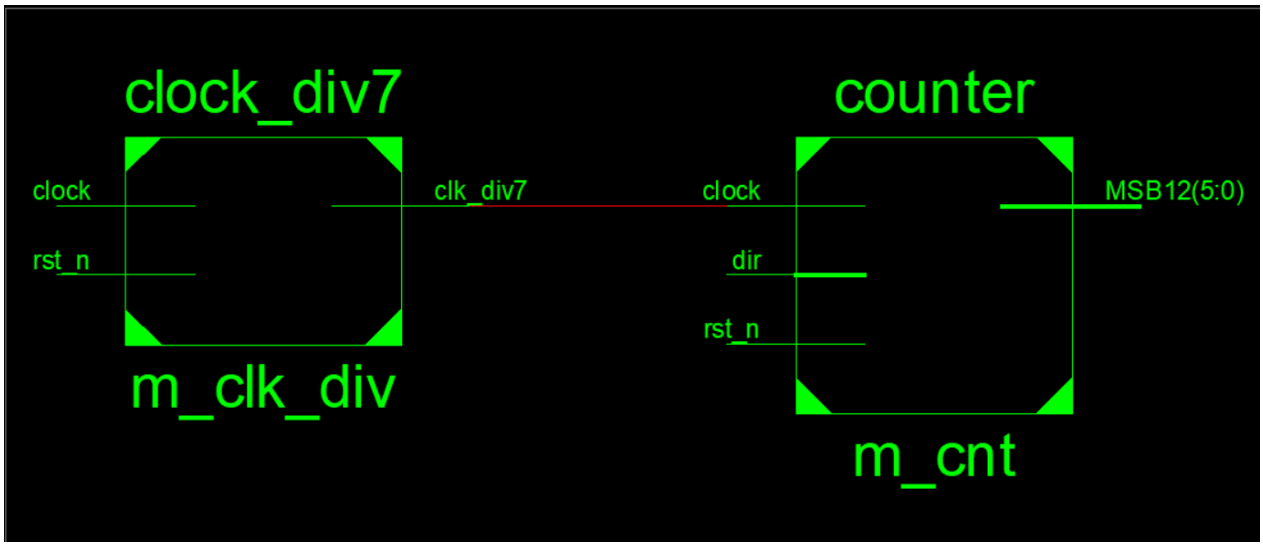
如果 SW1 =1（on）时，计数器初始值：count = 6b’h3f；

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 =10（on）时，计数器递减计数；

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

## 3、模块设计框图




## 4、实验原理：

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 ChipScope Pro 板的工作原理如下：

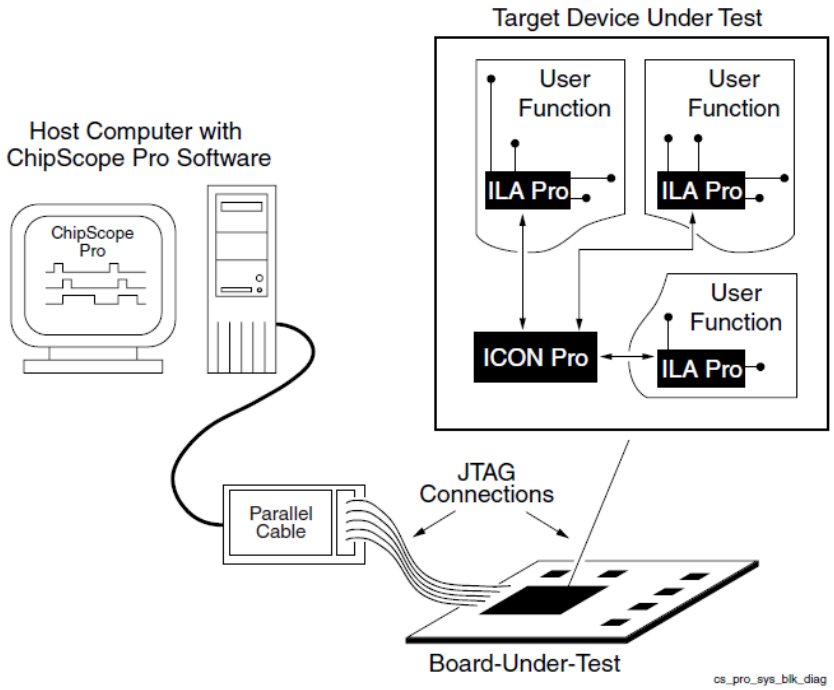



*Figure 1-1:* **ChipScope Pro System Block Diagram**

ChipScope Pro 的数据流图如下：

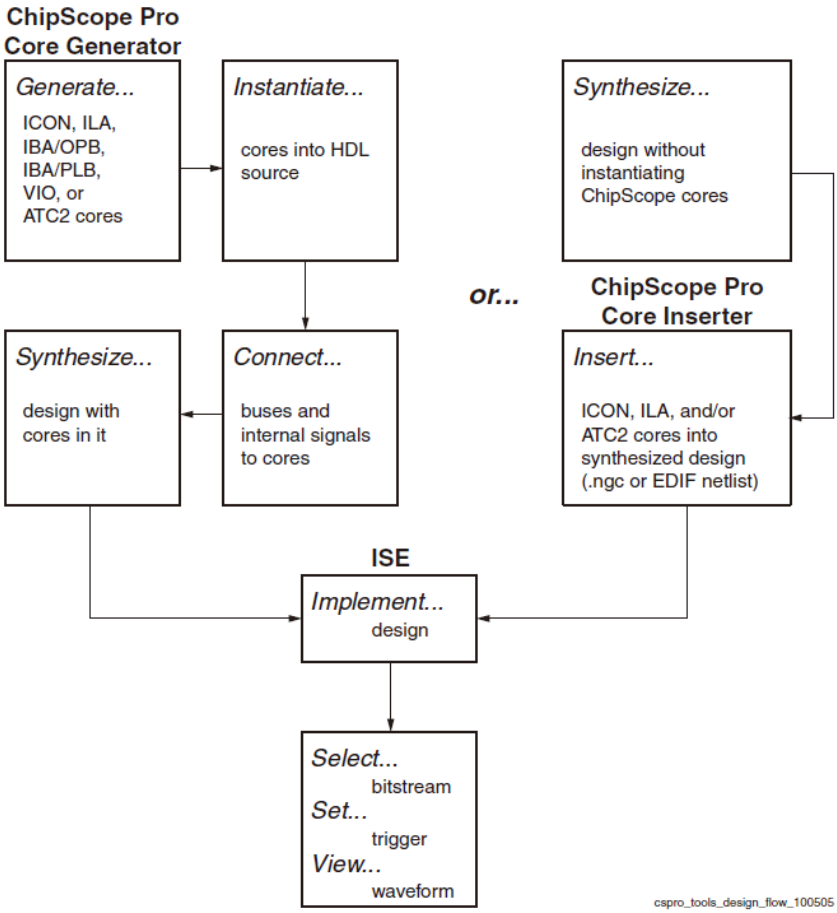



*Figure 1-2:* **ChipScope Pro Tools Design Flow**

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 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。

在本实验我们使用了中 ChipScore Pro 中的三个 Core 工具，分别为：ICON、ILA、VIO：

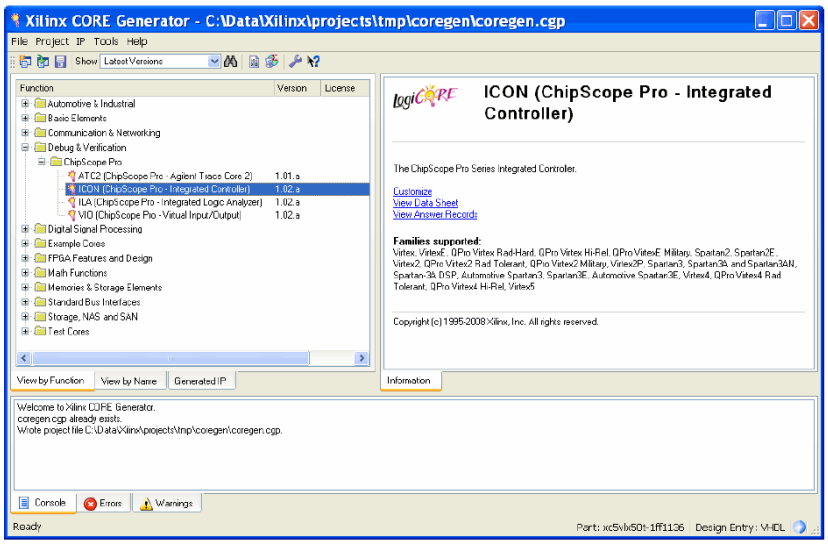



*Figure 2-2:* **Selecting the ICON Core**

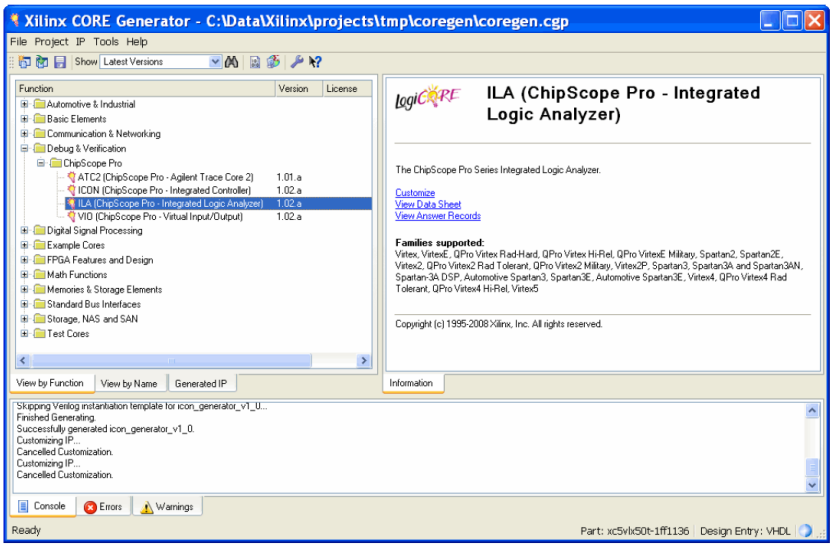



*Figure 2-5:* **Selecting the ILA Core**

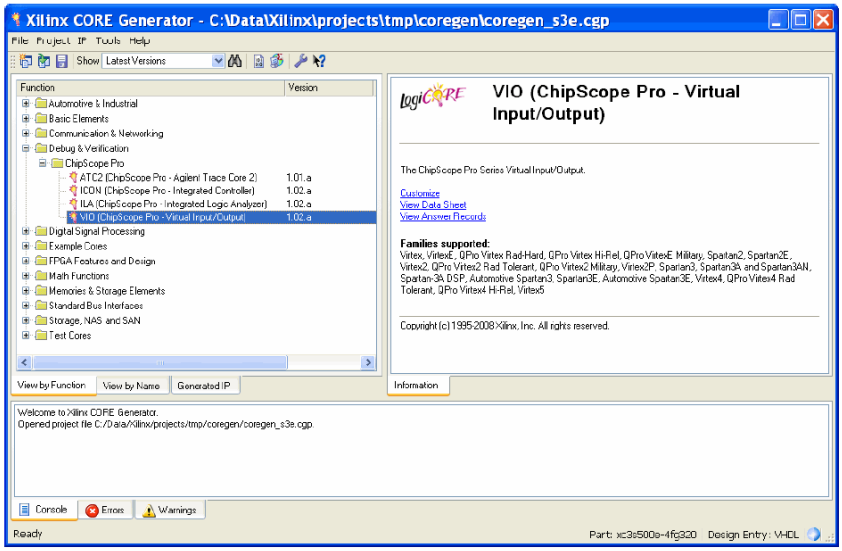



*Figure 2-10:* **Selecting the VIO Core**

## 5、Verilog 模块设计

（0）对 Spartan - 3E FPGA Starter Kit Board 上的50MHz 的时钟晶振（C9）进行分频，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 clk_div7 = 50MHz/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；

```
module clock_div7(output clk_div7, input clock, input rst_n );
  parameter N = 3'd7;
  parameter a = 3'd3;

  reg [2:0] cntposege;
  reg [2:0] cntnegege;

  always @( posedge clock or negedge rst_n ) // posedge of the clock
  begin
      if ( !rst_n )
          cntposege <= 3'b0;
      else begin
          if ( cntposege < N - 1'b1 )
              cntposege <= cntposege + 1'b1;
          else
              cntposege <= 3'b0;
      end
  end // get an clock from the posedge

  always @( negedge clock or negedge rst_n ) // negedge of the clock
  begin
      if ( !rst_n )
          cntnegege <= 3'b0;
      else begin
          if ( cntposege < N - 1'b1)
              cntnegege <= cntposege + 1'b1;
          else
              cntnegege <= 3'b0;
      end
  end // get an clock from the negedge
```

```
  reg clkpos;
  reg clkneg;

  always @(posedge clock or negedge rst_n) // div the posedge clock by a
  begin
     if ( !rst_n )
        clkpos <= 1'b0;
     else if (cntposege <= a-1'b1)
        clkpos <= 1'b0;
     else
        clkpos <= 1'b1;
  end

  always @(negedge clock or negedge rst_n) // div the negedge clock by a
  begin
     if ( !rst_n )
        clkneg <= 1'b0;
     else if (cntnegege <= a-1'b1)
        clkneg <= 1'b0;
     else
        clkneg <= 1'b1;
  end

  assign clk_div7 = clkpos & clkneg; //when merge the negedge and the po-
endmodule                          // sedge clock, we get the needed clock
```

（1）设计一个 6 位计数器，在分频获得的频率近似为 F= 7142857.14Hz 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；连接开发版上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，连接开发版上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 = 6’b0；

如果 SW1 =1（on）时，计数器初始值：count = 6b’h3f；

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 =10（on）时，计数器递减计数；

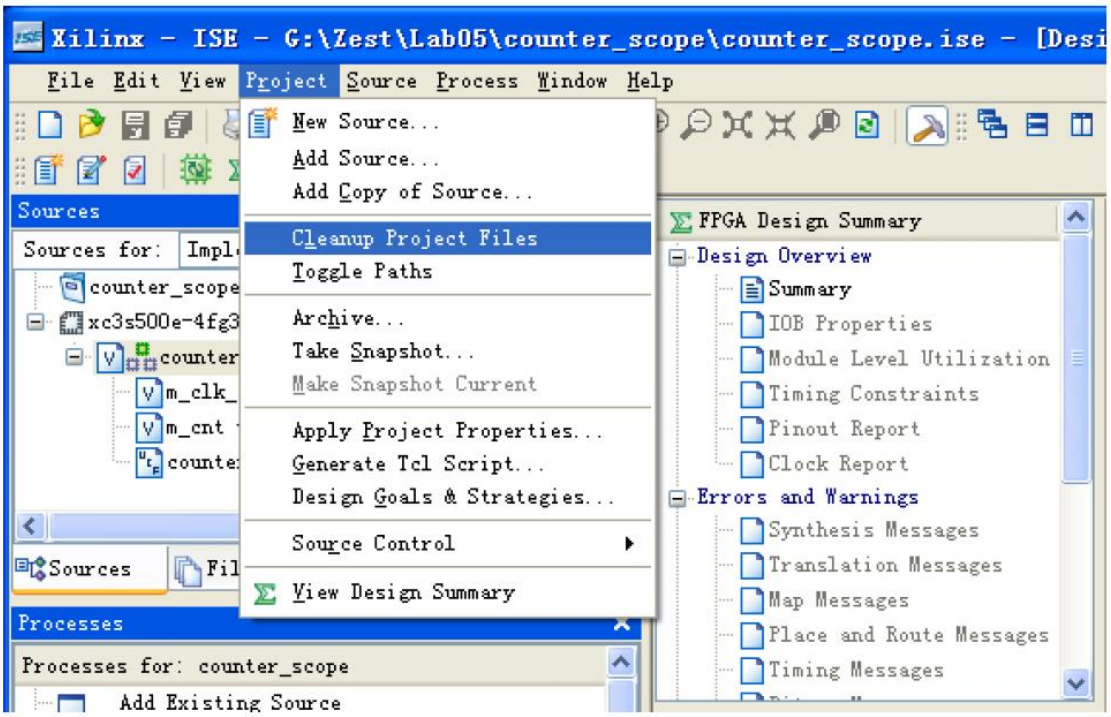
module counter( output [5:0] count, input clock, input rst_n, input dir);

```
module counter( output [5:0] MSB12, input clock, input rst_n, input dir);
  reg [5:0] temp;
  always @( posedge clock or negedge rst_n )
  begin
      if ( !rst_n ) begin
          temp <= (!dir) ? 6'h0 : 6'h3F; // if reset, set value for temp
      end
      else begin
          if (!dir) temp <= temp + 1'b1; // if !dir, then count up
          else temp <= temp - 1'b1;      // if dir, then count down
      end
  end
  assign MSB12 = temp;
endmodule
```

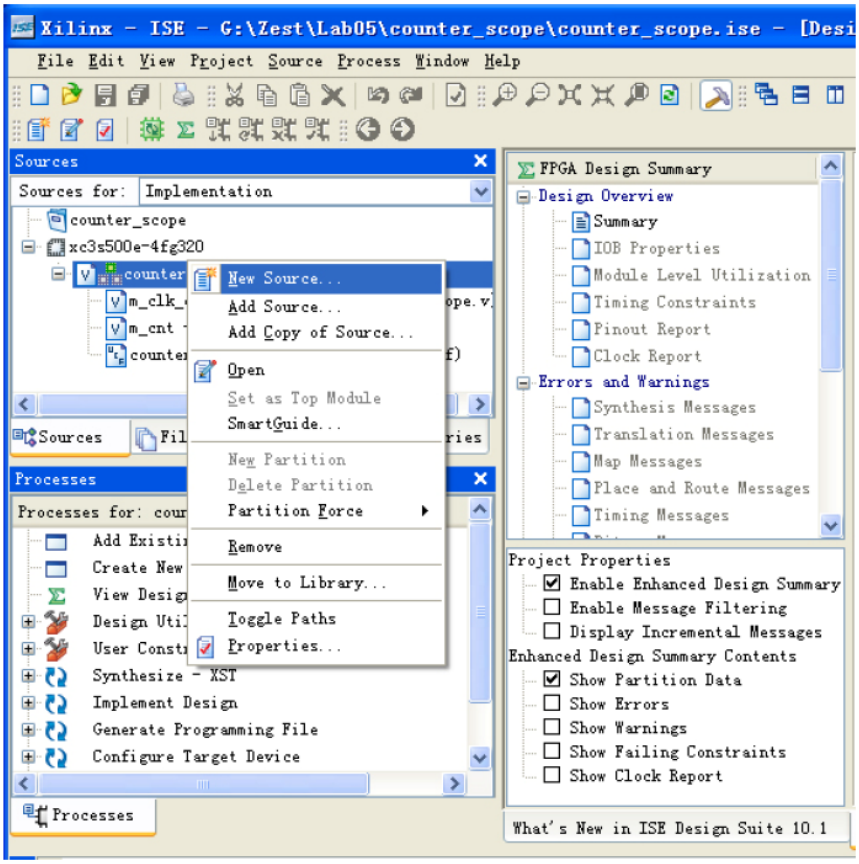
在上述步骤完成以后，可以将bit文件烧制到初级板上进行运行，待程序不再报错，运行正常以后以后，我们开始使用ChipScope Pro模块来进行我们的实验。

（2）、在counter_scope 工程中添加用于 ChipScope Pro 分析的模块

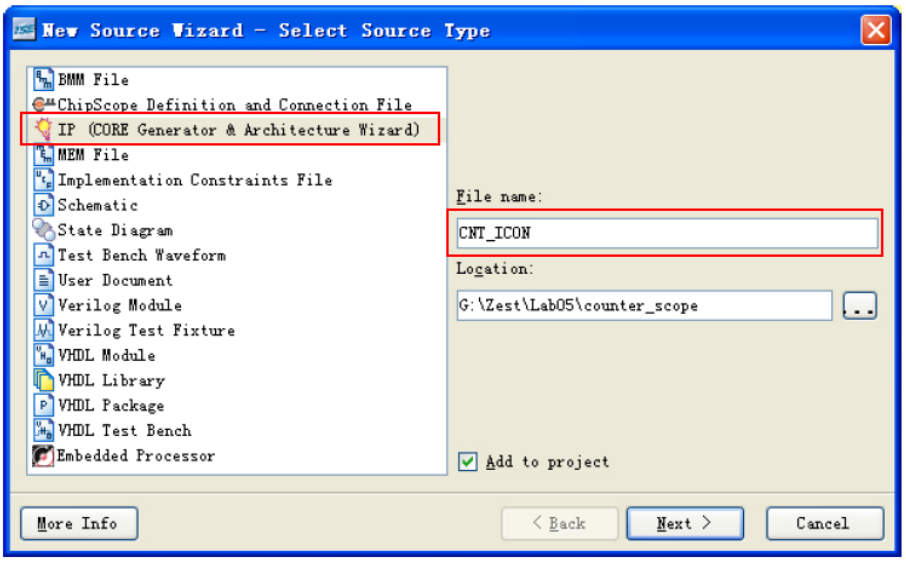
依次点击菜单栏中：Project →Cleanup Project Files，清理工程的中间文件



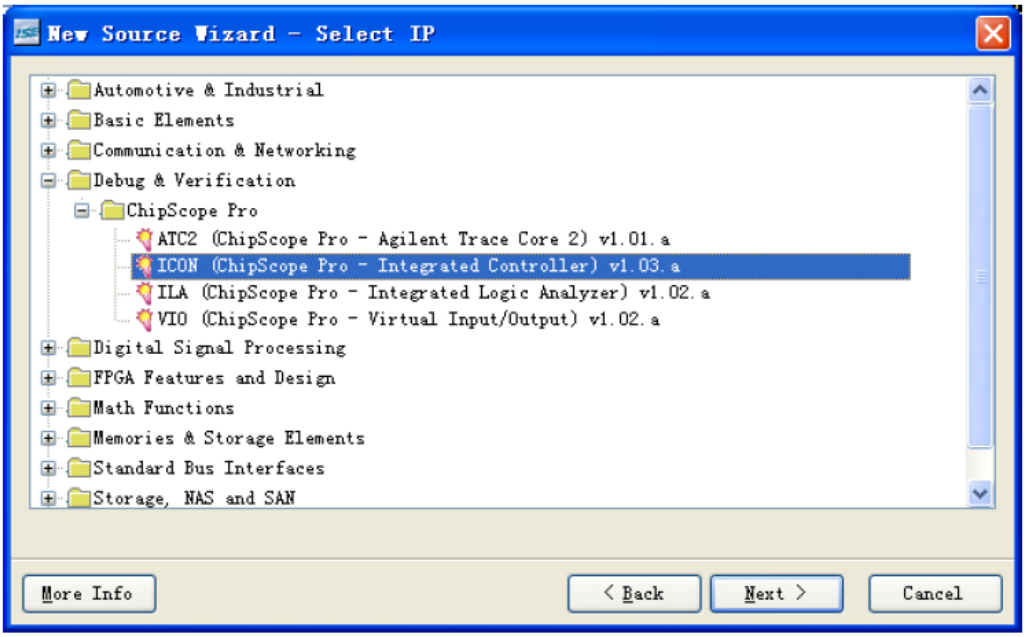
添加“ICON”模块，右键点击“Sources”窗口，然后选择添加“New Sources”，如下图所示：



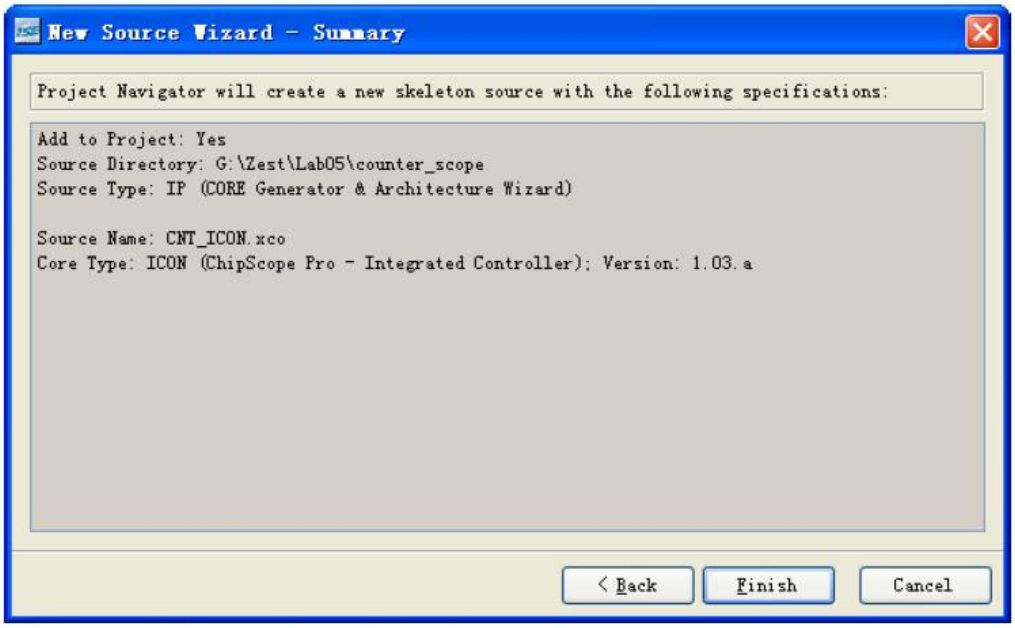

弹出“New Sources Wizard”对话框，选择“IP ( CORE Generator & Architectue Wizard)”，文件名为：CNT_ICON，准备插入“集成控制器”模块；



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

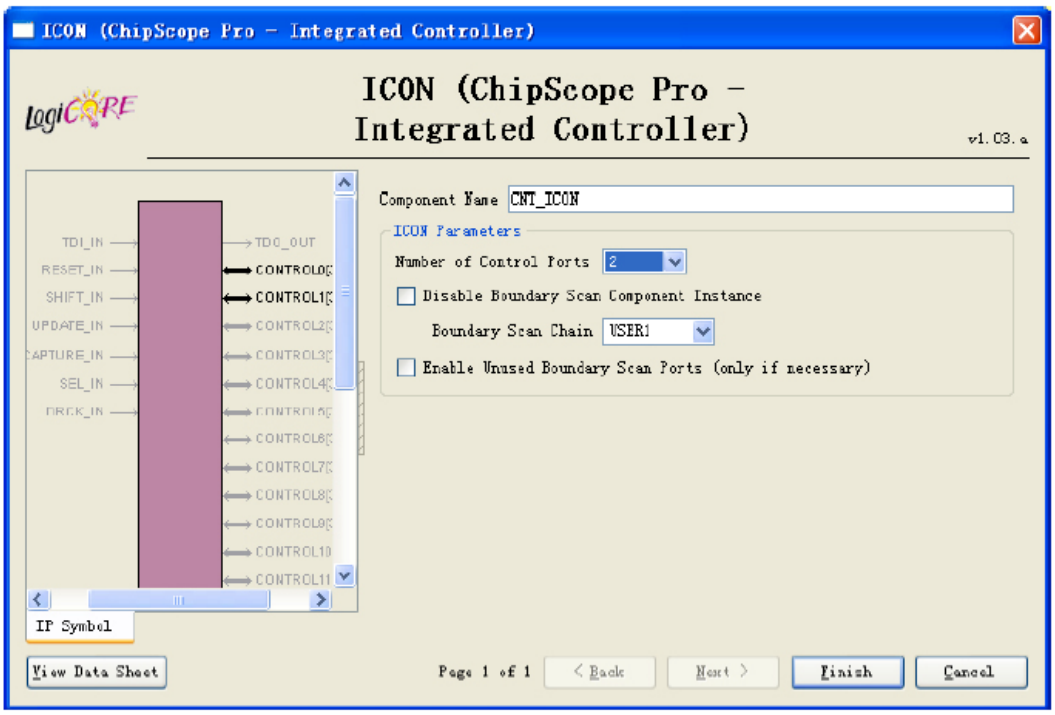



点击“Next”，

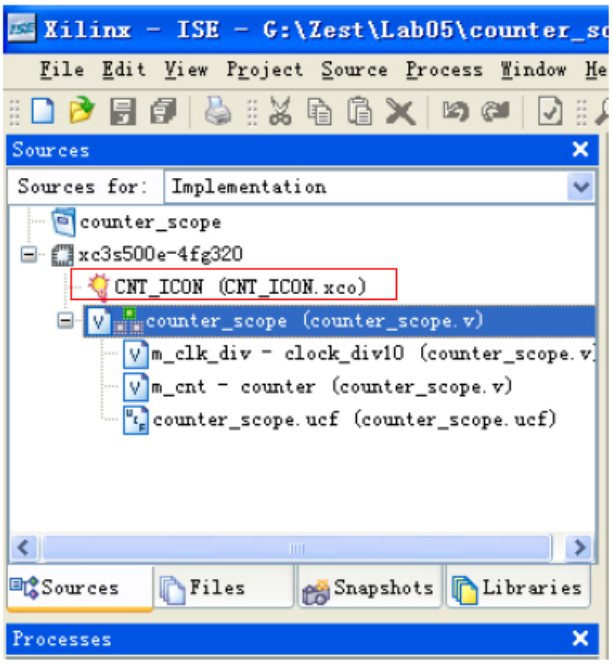



点击“Finish”，ISE 运行一会，出现“ICON (ChipScope Pro Integrated Controller)”对话框；

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。




用同样的方式，添加ILA与VIO文件到工程中。

```
21
22 // Instantiate the module
23 // ---------- Begin Cut here for INSTANTIATION Template -----
   INST_TAG
24 CNT_ICON instance_name (
25     .CONTROL0(CONTROL0),
26     .CONTROL1(CONTROL1)
27 );
28
29 // INST_TAG_END ------ End INSTANTIATION Template ---------
30
```

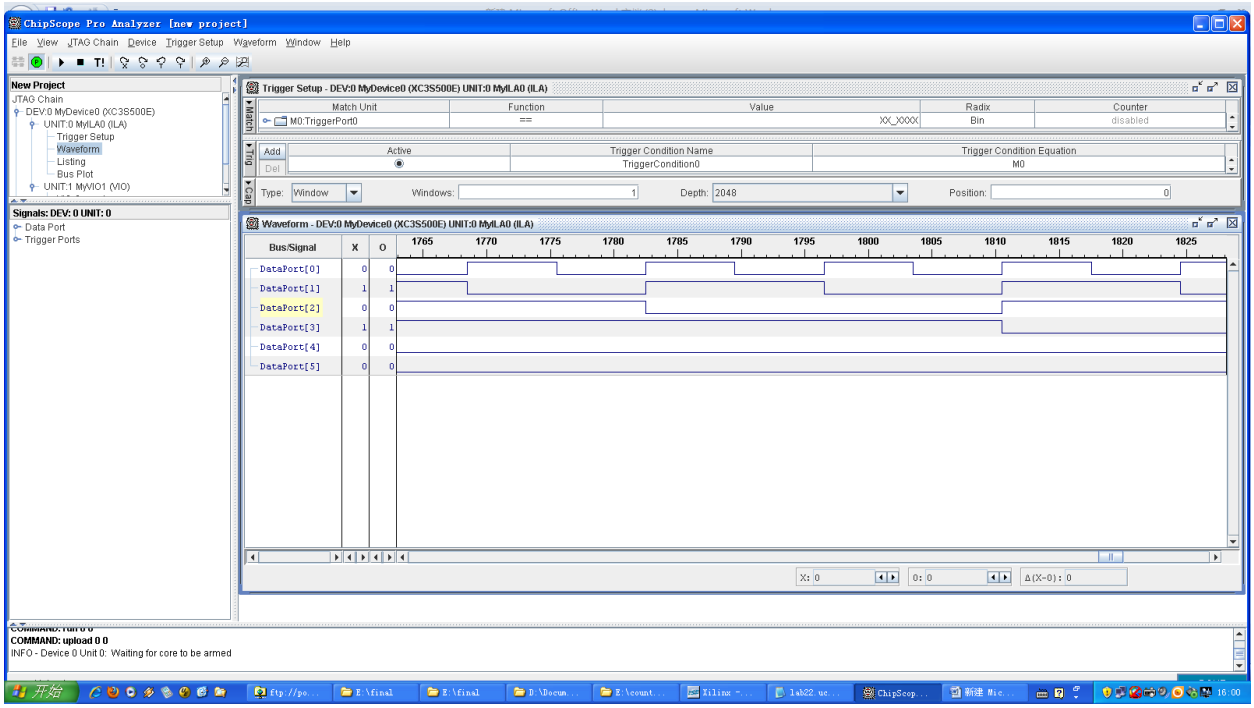
将以上的ICON、ILA、VIO代码段贴入到我们的运行模块中。

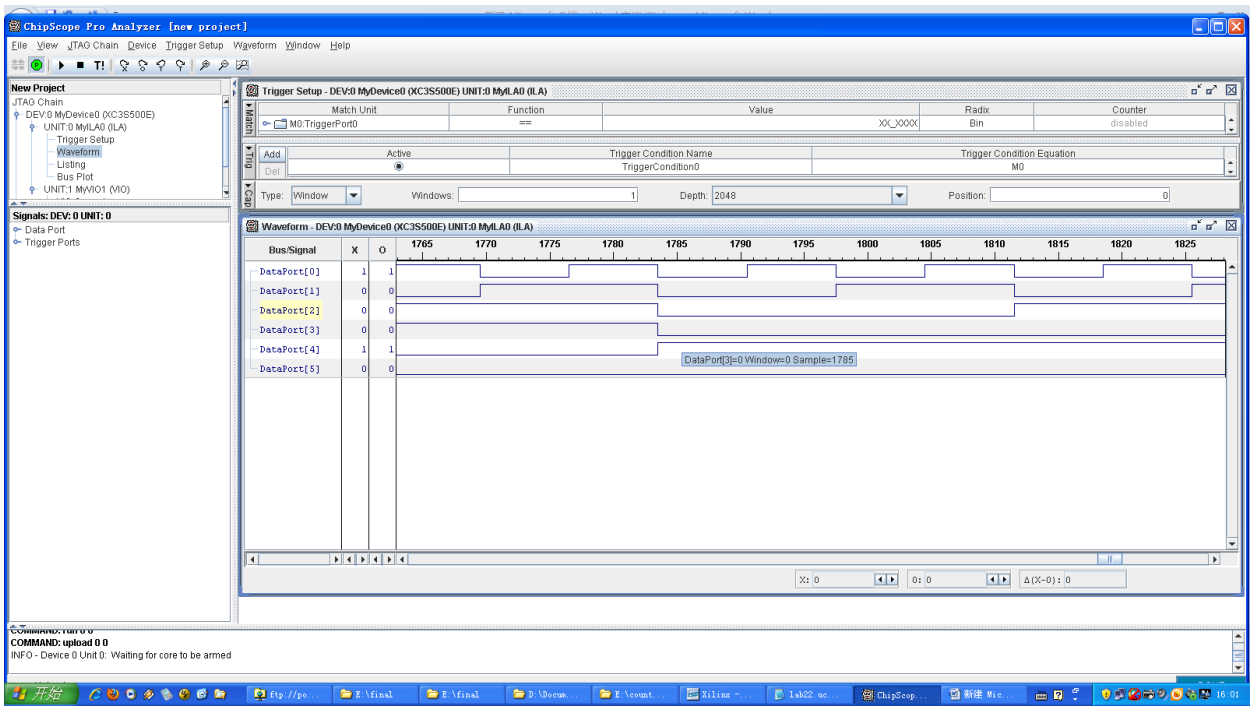
```
module counter_scope( output [5:0] LEDOut, input clock, rst_n, dir );
  wire clk_div7;
  wire [1:0] vrst;
  wire [1:0] vdir;
  wire rst_L;
  wire dir_H;

  assign rst_L = rst_n | (vrst[1] & vrst[0]); // create a new rst
  assign dir_H = dir | (vdir[1] & vdir[0]);   // create a new dir

  wire [5:0] MSB12;
  clock_div7 m_clk_div(.clk_div7(clk_div7), .clock(clock), .rst_n(rst_L) );
  counter
m_cnt( .MSB12(MSB12), .clock(clk_div7),.rst_n(rst_L), .dir(dir_H) );
  assign LEDOut = MSB12;

  wire [35:0] CONTROL0;
  wire [35:0] CONTROL1;
  wire [3:0] din;
  wire [1:0] trig = MSB12[5:4];

  CNT_ICON instance_name1 ( // implement an ICON module
       .CONTROL0(CONTROL0),
       .CONTROL1(CONTROL1)
  );

  CNT_ILA instance_name2 ( // implement an ILA module
       .CONTROL(CONTROL0),
       .CLK(clock),
       .DATA(MSB12),
       .TRIG0(trig)
  );

  CNT_VIO instance_name3 ( // implement a VIO module
       .CONTROL(CONTROL1),
       .ASYNC_IN(trig),
       .ASYNC_OUT(din)
  );
```

```
  assign vrst = din[3:2]; // set the parameter for vrst
  assign vdir = din[1:0]; // set the parameter for vdir
endmodule
```

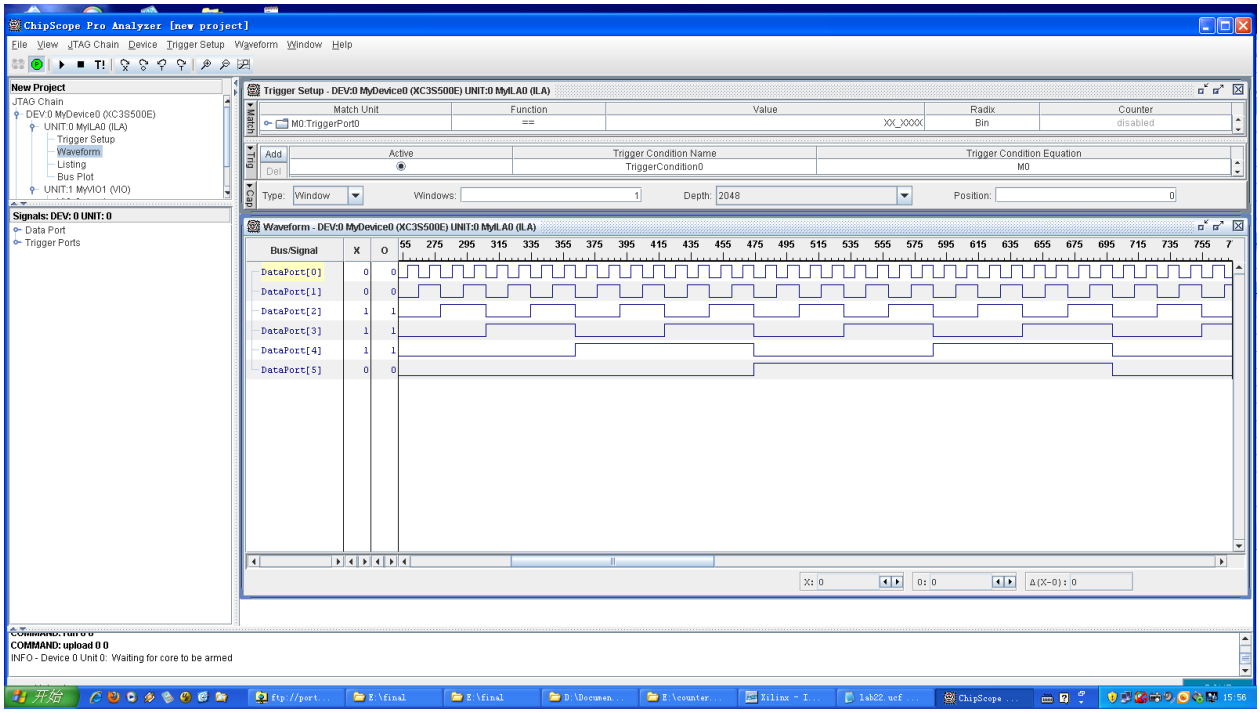
## 6、试验仿真结果和分析

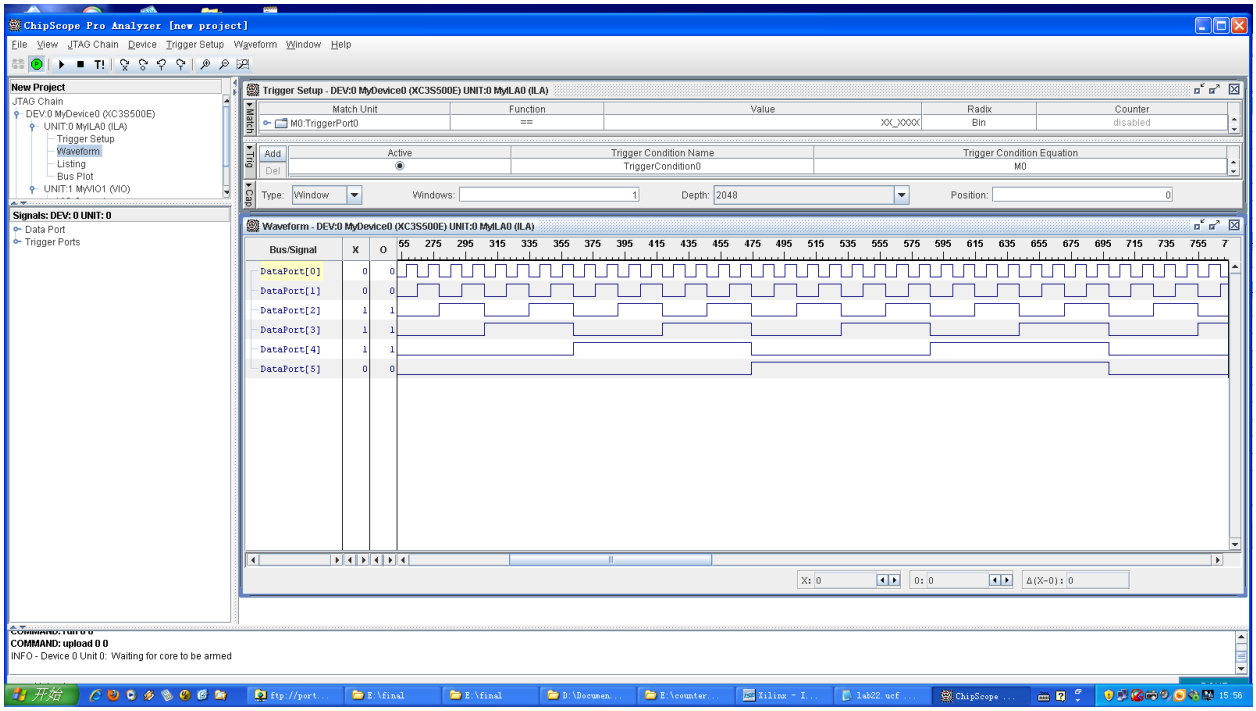
从下面两张图中可以看出，通过正向与反向计数，我们的计数器频率始终是50MHz/7，很好地达到了实验要求，并且占空比为50%。



接下来两张图展示的是我们正向计数与反向计数的效果：



之后，我们又对 reset 结果进行了测试，发现也很好地达成了要求：