2017《FPGA 应用实验》实验报告

实验编号：Lab05　　实验时间：2018.4.10

实验名称：使用 ChipScope Pro 分析调试 FPGA 设计

班级：F1503006　学号：515030910141　姓名：翟拙存

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；

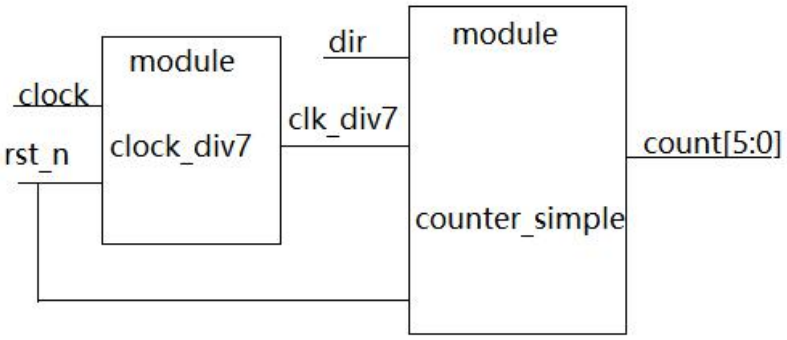
如果 SW1 =1（on）时，计数器初始值：count = 6b'h3f；

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：

1. ChipScope Pro 简介：

如图1所示，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。

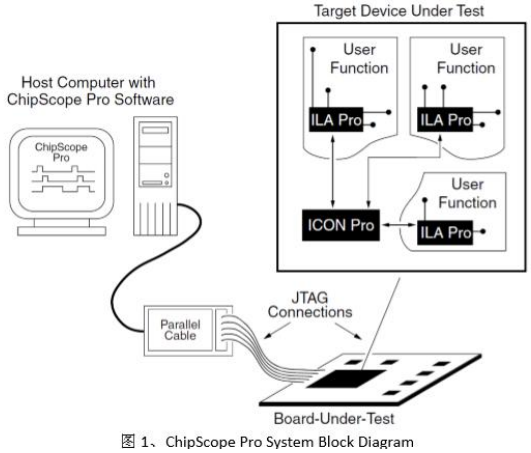


图 1、ChipScope Pro System Block Diagram

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.

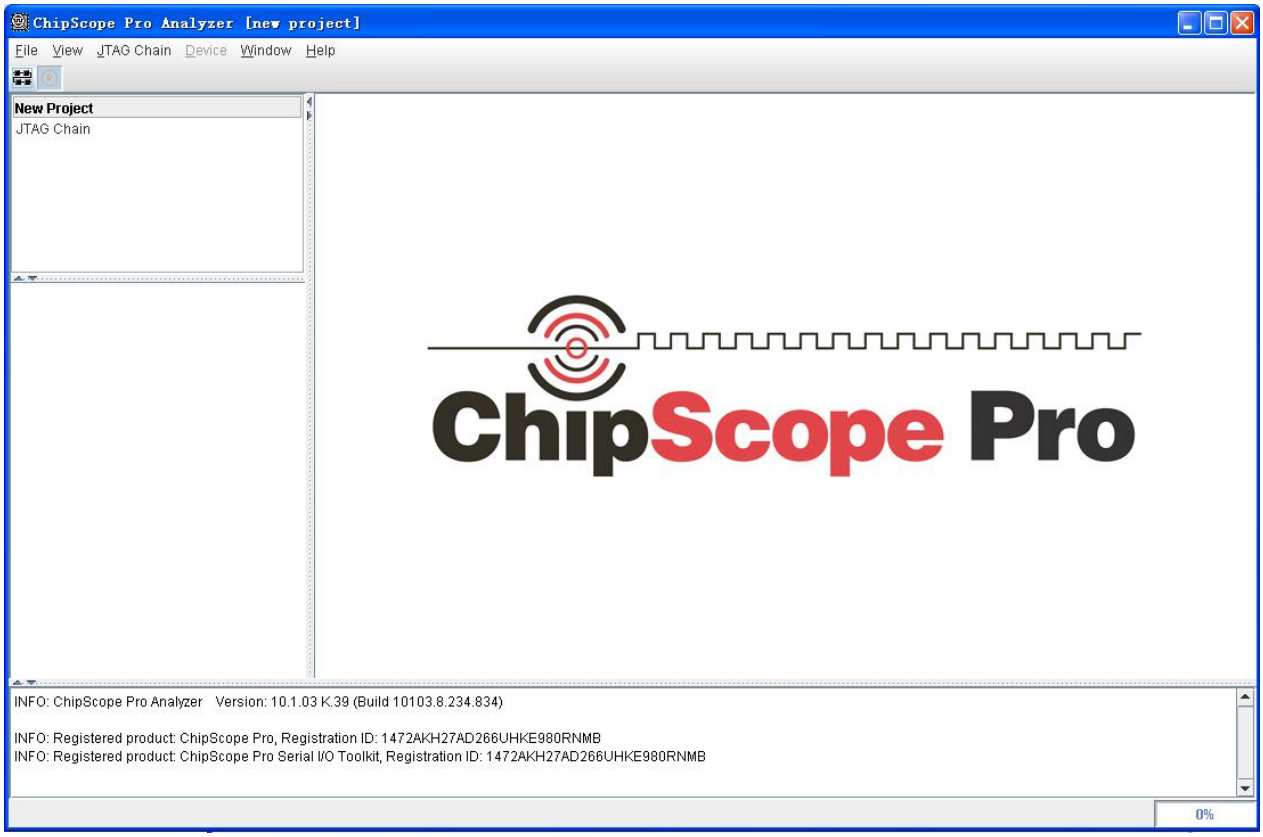
ChipScope Pro 工具箱包含 3 个工具：ChipScope Pro Core Generator（核生成器）、

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。

2. 测试原理：

利用 ChipScope Pro 可读取出输出的可视波形，解决因当时钟频率过快时，无法观察到可视现象的问题。而可观察到内部逻辑的时序信号，将会大大帮助设计者验证自己设计的正确性以及可靠性。



5、Verilog 模块设计

1. 顶层模块 counter.v：

```
`timescale 1ns / 1ps
module counter(output [5:0]count, input clock , input rst_n , input dir);
    wire clk_div7;

    clock_div7 c(.clk_div7(clk_div7) , .clock(clock) , .rst_n(rst_n));
    counter_simple cs(.count(count) , .clock(clk_div7) , .rst_n(rst_n) , .dir(dir) );

    wire [35:0]CONTROL;
```

```
    // ---------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ILA instance_name (
    .CONTROL(CONTROL),
    .CLK(clock),
    .DATA(count),
    .TRIG0(count[5:4])
    );
    // INST_TAG_END ------ End INSTANTIATION Template ---------

    // Instantiate the module
    // ---------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ICON instance_name (
    .CONTROL0(CONTROL)
    );
    // INST_TAG_END ------ End INSTANTIATION Template ---------
endmodule
```

2. 简单计时器模块 counter_simple.v：

```
module counter_simple(output reg [5:0]count, input clock , input rst_n , input dir);
    always@(posedge clock or negedge rst_n)
    begin
        if (!rst_n)
            count <= (dir)? 6'b11_1111 : 6'b0;
        else
            begin
                if (!dir)
                    count <= count + 1;
                else
                    count <= count - 1;
            end
    end
endmodule
```

3. 七分频模块 clock_div7.v：

```
module clock_div7( output reg clk_div7 , input clock , input rst_n);
    reg[5:0]temp;
    always@(posedge clock or negedge rst_n)
    begin
        if(!rst_n)
        begin
```

```
            clk_div7<=1;
            temp<=6'b0;
        end
        else
        begin
            temp<={temp[4:0],clk_div7};
            clk_div7<=temp[5];
        end
    end
endmodule
```

4. count.ucf:

```
NET "count<0>" LOC = "E11";
NET "count<1>" LOC = "F11";
NET "count<2>" LOC = "C11";
NET "count<3>" LOC = "D11";
NET "count<4>" LOC = "E9";
NET "count<5>" LOC = "F9";

NET "clock" LOC = "C9";
NET "rst_n" LOC = "L13";
NET "dir" LOC = "L14";
```

注:顶层模块中的以下部分代码是编译之后从文件CNT_ICON.veo和CNT_ILA.veo中拷贝而来:

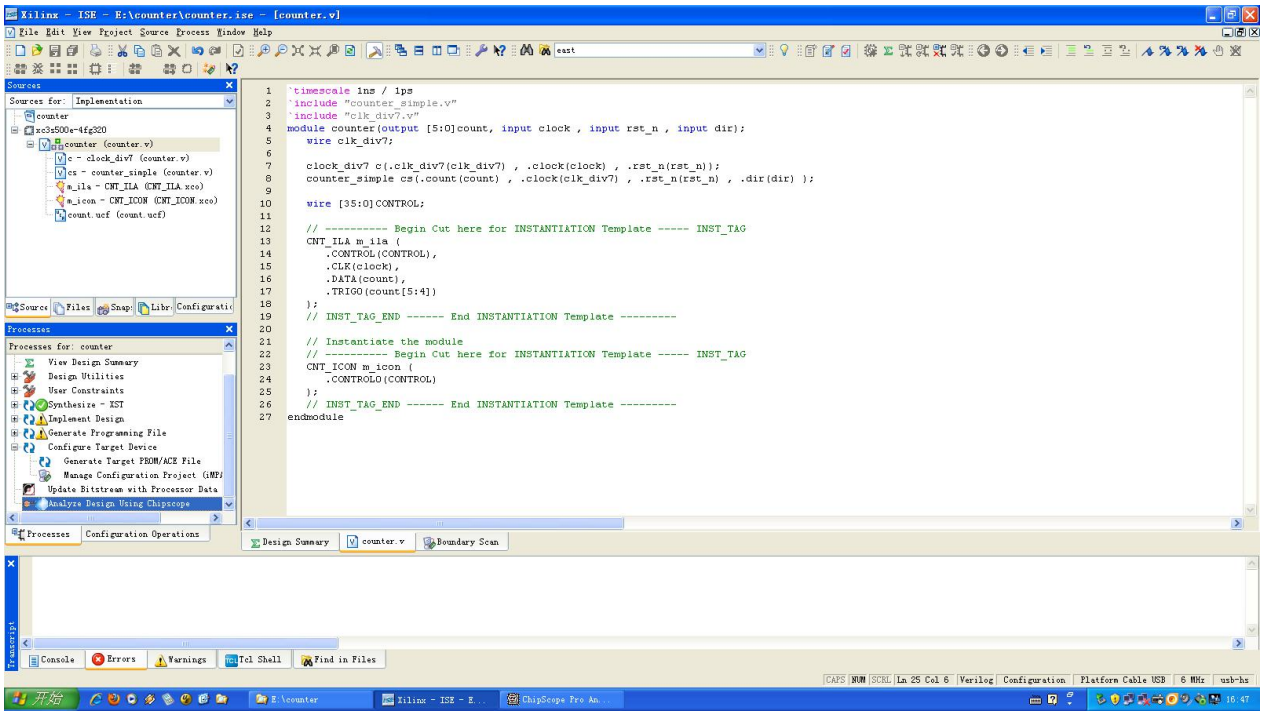
```
wire [35:0]CONTROL;

    // ---------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ILA instance_name (
    .CONTROL(CONTROL),
    .CLK(clock),
    .DATA(count),
    .TRIG0(count[5:4])
    );
    // INST_TAG_END ------ End INSTANTIATION Template ---------

    // Instantiate the module
    // ---------- Begin Cut here for INSTANTIATION Template ----- INST_TAG
    CNT_ICON instance_name (
    .CONTROL0(CONTROL)
```

```
);
// INST_TAG_END ------ End INSTANTIATION Template ---------
```

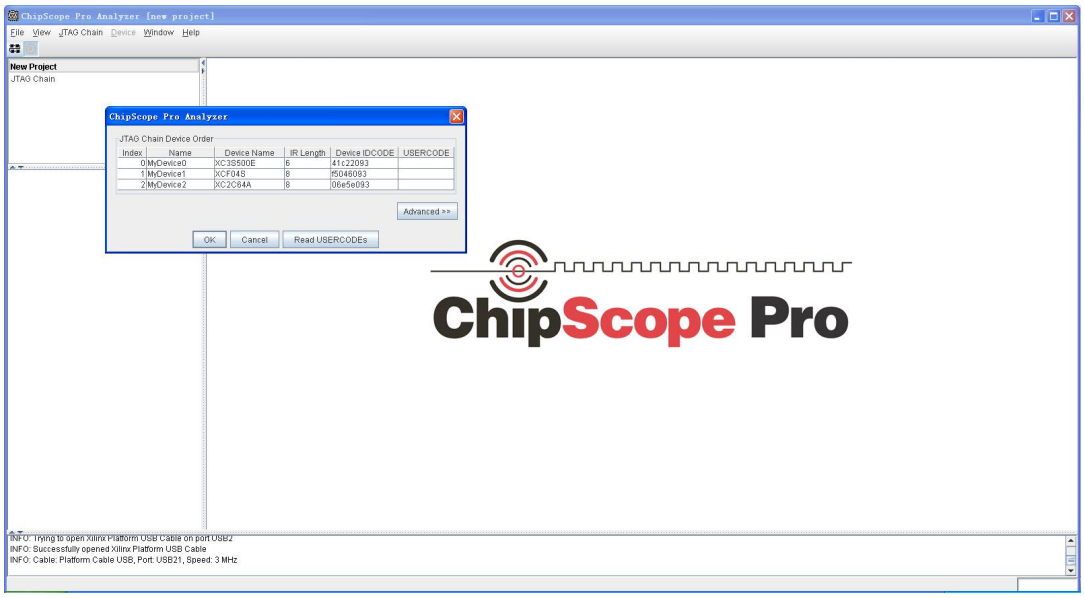
将以上代码添加到初始代码后（如下图）再进行综合、实现、产生编程文件，并使用 ChipScope Pro 进行分析。



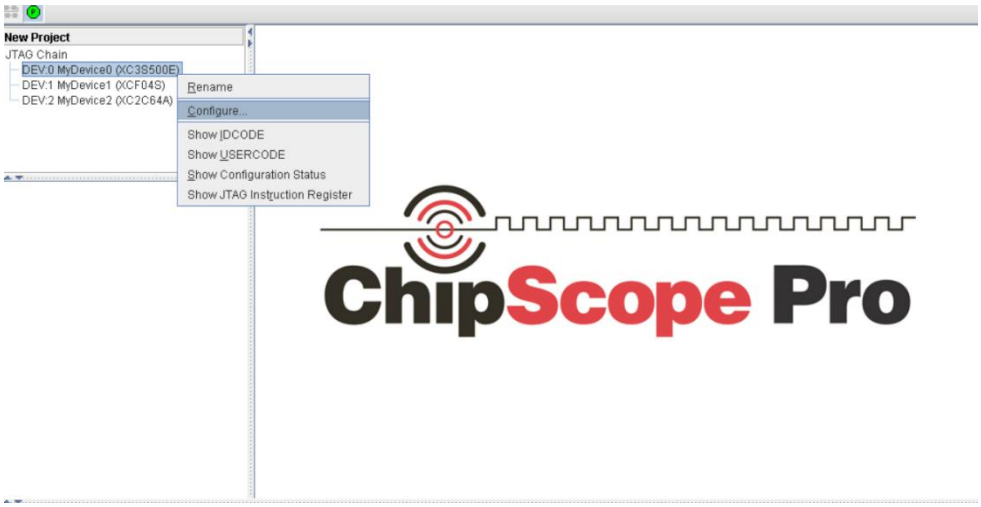
6、试验仿真结果和分析

1. ChipScope Pro Analyzer 启动后，界面如下图所示。

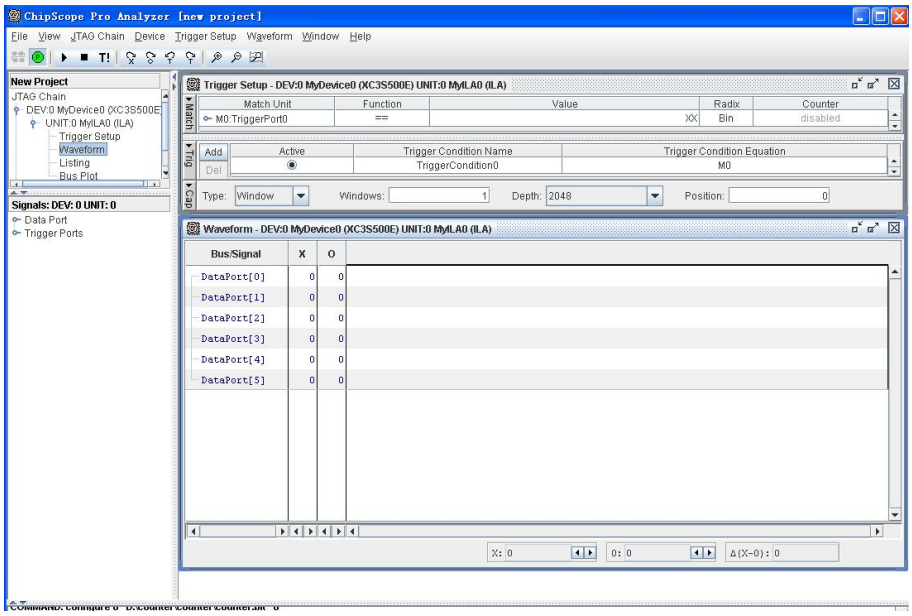
2. 配置目标芯片，在常用工具栏上点击图标：初始化边界扫描链，成功完成扫描后，将会列出 JTAG 链上的器件。选择我们使用的开发板 FPGA 芯片型号 XC3S500E。



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

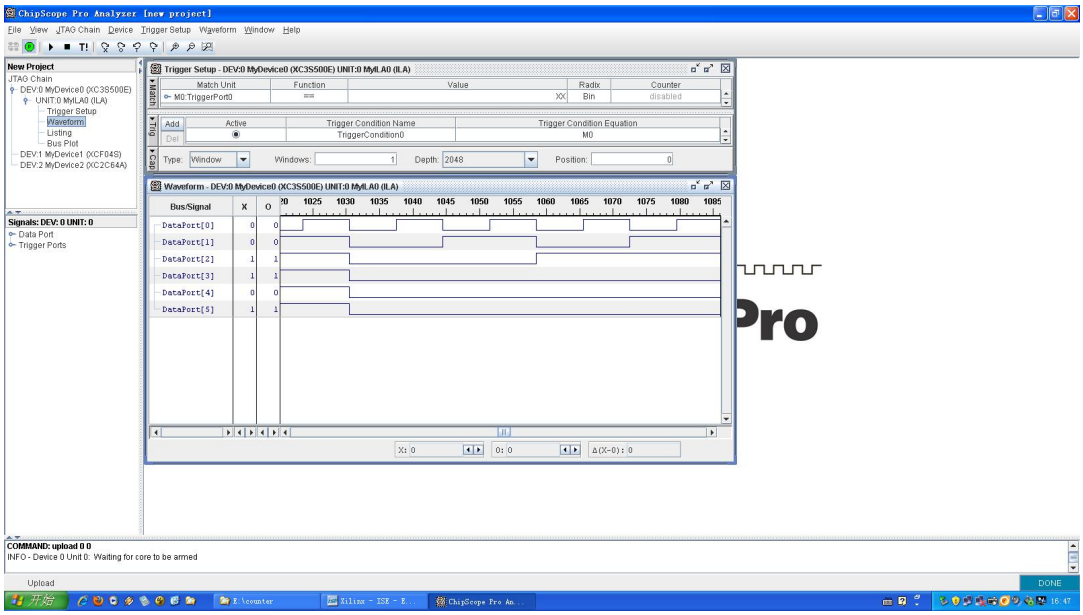


4.观察信号波形需要打开 Waveform 窗口，双击 Waveform 命令，显示界面如下图所示。

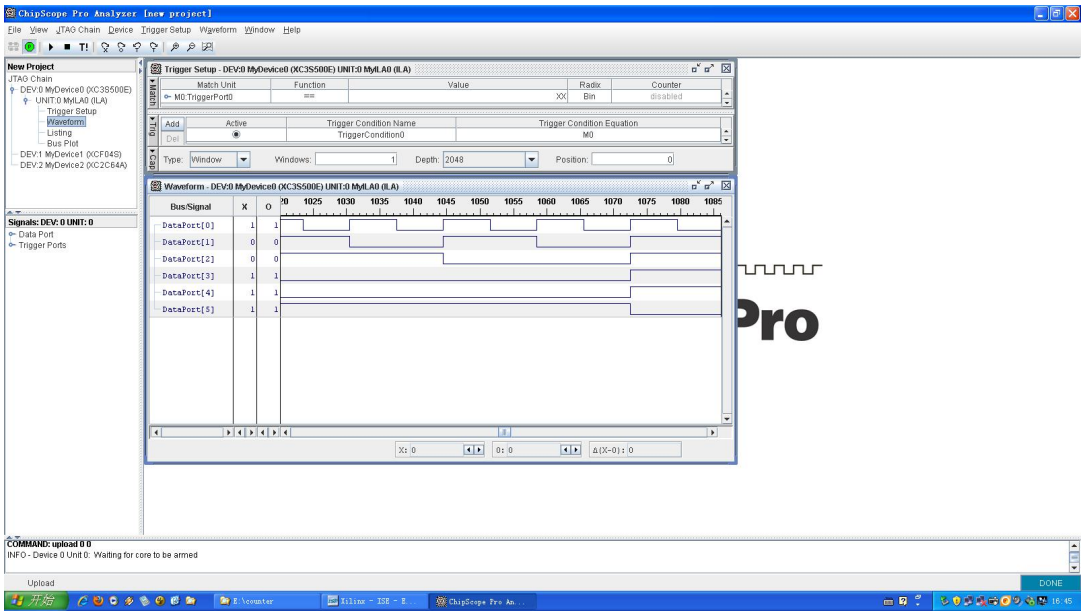


5. 最终结果如下图所示：

当 dir=0 时：



当 dir=1 时：



从图中可以看出，7 分频以及计数器等模块全部正确。