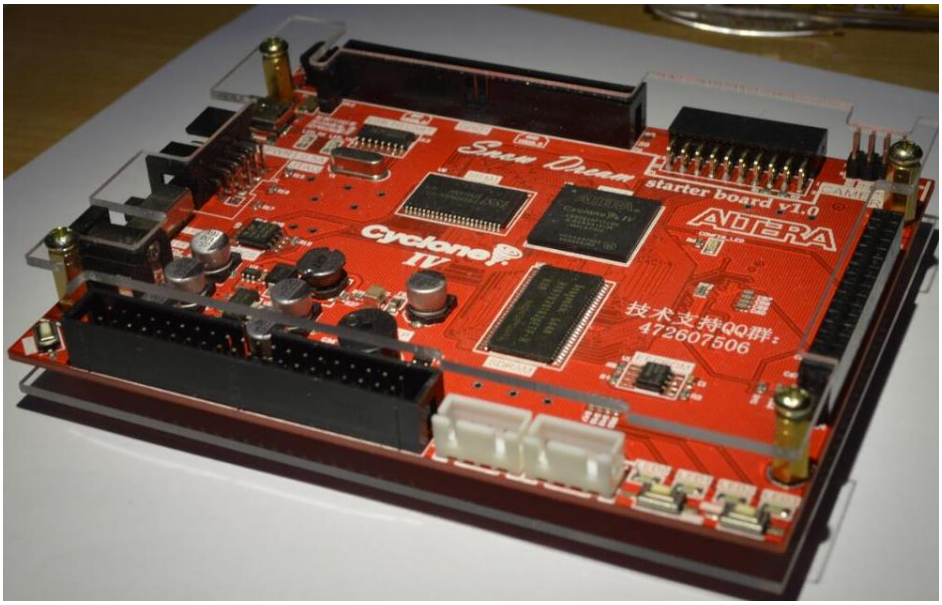
三、时序逻辑电路设计之计数器

实验目的：以计数器为例学会简单的时序逻辑电路设计

实验平台：芯航线 FPGA 核心板



实验原理：

时序逻辑电路是指电路任何时刻的稳态输出不仅取决于当前的输入，还与前一时刻输入形成的状态有关。这跟组合逻辑电路相反，组合逻辑的输出只会跟目前的输入成一种函数关系。换句话说，时序逻辑拥有储存元件（内存）来存储信息，而组合逻辑则没有。

计数器的核心元件是触发器，基本功能是对脉冲进行计数，其所能记忆脉冲最大的数目称为该计数器的模/值。计数器常用在分频、定时等处。计数器的种类很多，按照计数方式的不同可以分为二进制计数器、十进制计数器以及任意进制计数器，按照触发器的时钟脉冲信号来源可分为同步计数器与异步计数器。按照计数增减可分为加法计数器、减法计数器以及可逆计数器。

此处设计一个计数器，使其使能板载 LED 每 500ms，状态翻转一次。核心板晶振为 50MHz，也就是说时钟周期为 20ns，这样可以计算得出 500ms = 500_000_000ns/20 = 25_000_000；即需要计数器计数 25_000_000 次，也就是需要一个至少 25 位的计数器。且每当计数次数达到需要清零并重新计数。

Verilog HDL 之所以被称为硬件电路描述语言，就是因为我们不是在类似 C 一样进行普通的编程，而是在编写一个实际的硬件电路，例如 02 中设计的一个二选一选择器最后就是被综合称为一个真正的选择器。上面提到计数器即为加法器、比较器、寄存器以及选择器构成，如图 4-1 所示。

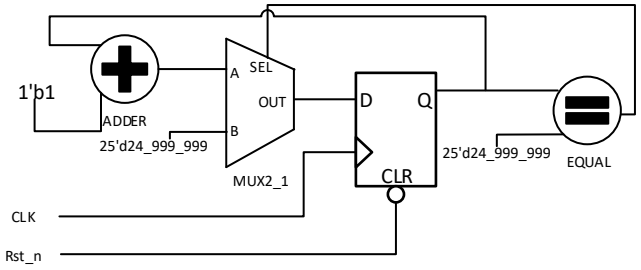


图 4-1 计数器逻辑电路图

实验内容：

按照 02 章所讲，建立工程子文件夹后，新建一个以名为 counter 的工程保存在 prj 下，并在本工程目录的 rtl 文件夹下新建 verilog file 文件在此文件下输入以下内容并以 counter.v 保存。这里之所以在计数值计数到 25'd24_999_999 而不是 25'd25_000_000 是因为计数器是从 0 开始计数而不是 1。这里每当计数器计数到预设的值后就让 led 取反一次。

```
module counter(Clk50M,Rst_n,led);

    input Clk50M;   //系统时钟，50M
    input Rst_n;    //全局复位，低电平复位

    output reg led; //led 输出

    reg [24:0]cnt;  //定义计数器寄存器

//计数器计数进程
    always@(posedge Clk50M or negedge Rst_n)
    if(Rst_n == 1'b0)
        cnt <= 25'd0;
    else if(cnt == 25'd24_999_999)
        cnt <= 25'd0;
    else
        cnt <= cnt + 1'b1;

//led 输出控制进程
    always@(posedge Clk50M or negedge Rst_n)
    if(Rst_n == 1'b0)
        led <= 1'b1;
    else if(cnt == 25'd24_999_999)
        led <= ~led;
    else
        led <= led;

endmodule
```

进行分析和综合直至没有错误以及警告。

为了测试仿真编写测试激励文件，新建 counter_tb.v 文件并输入以下内容再次进行分析和综合直至没有错误以及警告，保存到 testbench 文件夹下。这里生成了一个周期为 20ns 的时钟 clk，并且例化了需要测试的 counter.v。

```
`timescale 1ns/1ns
`define clock_period 20
module counter_tb;
    reg clk;
    reg rst_n;

    wire led;

    counter counter0(
        .Clk50M(clk),
        .Rst_n(rst_n),
        .led(led)
    );

    initial clk = 1;
    always #(`clock_period/2) clk = ~clk;

    initial begin
        rst_n = 1'b0;
        #(`clock_period *200);
        rst_n = 1'b1;
        #2000000000;
        $stop;
    end

endmodule
```

设置好仿真脚本后进行功能仿真，可以看到如图 3-2 所示的波形文件，可以看出高低电平变化的时间均是 0.5s 也就是 500ms，得出符合既定的设计要求，至此功能仿真结束。

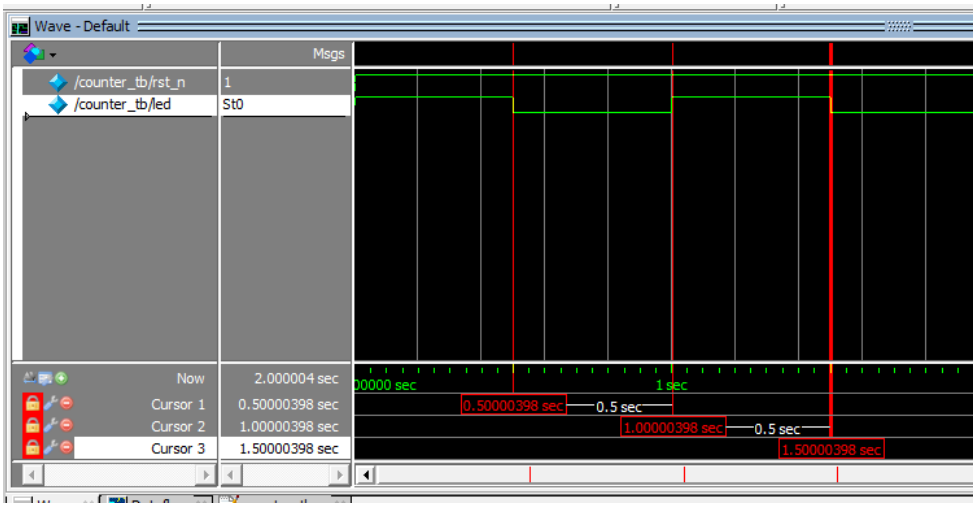


图 3-2 功能仿真波形文件

在进行上述的功能仿真时可以发现需要仿真时间较长，这是由于将计数器的计数值太大，因此可以将 counter.v 的 cnt 计数值修改为 24_999 来减少仿真时间，这时会发现仿真时间大幅度缩短，且图 3-3 中高低电平变化时间变为 500_000ns，相比 500ms 缩短了 1000 倍，也可以说明功能仿真正确。

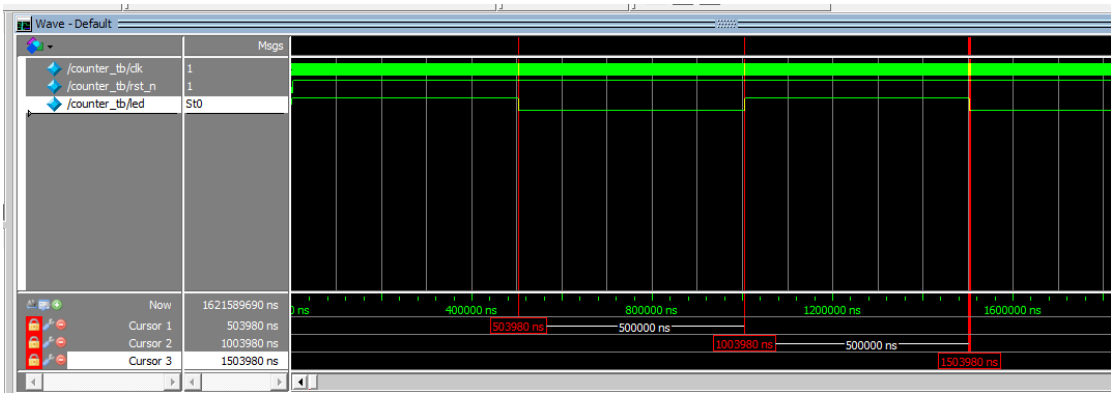


图 3-3 缩小计数值后的功能仿真波形

进行全编译后进行门级仿真，可以看到如图 3-4 所示波形图，在这可以看出由于门电路的延迟高低电平变化时间并不严格等于 0.5s。

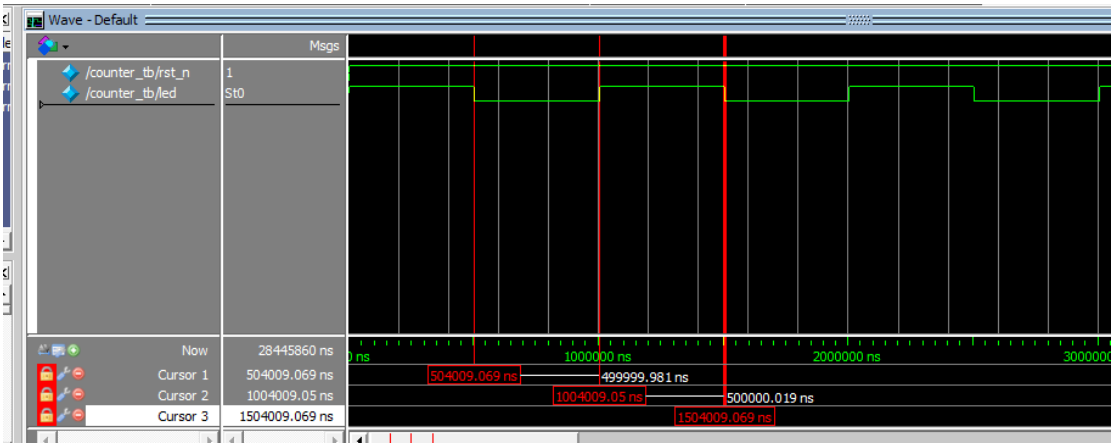


图 3-4 时序仿真波形

现进行分配引脚，此处介绍另一种分配引脚的方式，采用 tcl 文件。首先在 File—New 中选中 Tcl Script File，新建一个 tcl 文件。并输入以下内容后以 PIN.tcl 名称保存到 prj 文件

夹下。此处由于不同批次可能会引脚分配略有不同，请根据对应的引脚表来编写。

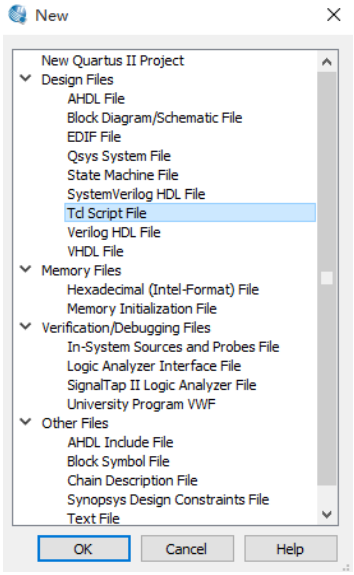


图 3-5 新建 tcl 文件

```
set_location_assignment PIN_E1 -to Clk50M
set_location_assignment PIN_M1 -to Rst_n

set_location_assignment PIN_P11 -to led
```

然后单击 Tools--Tcl Script，弹出图 3-7 对话框后选中编写好的 PIN.tcl 文件，点击 Open Files 编写的内容就会出现在下面的框图中，此时再点击 Run 会弹出图 3-8 对话框，提示已经运行完毕。我们这时可以打开 Pin planer 查看分配好的引脚。

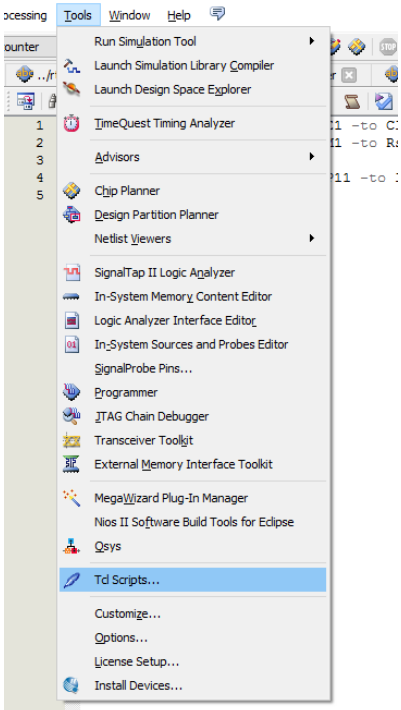


图 3-7 设置 Tcl 脚本

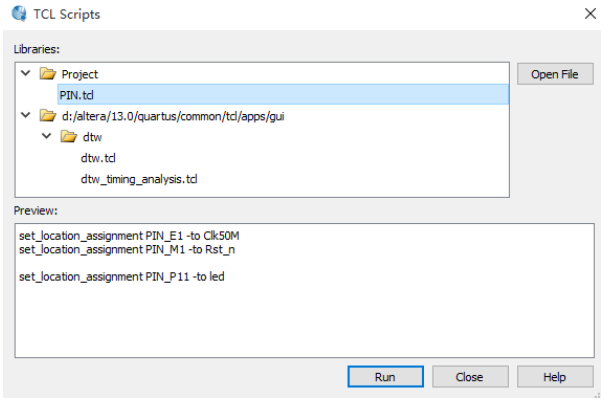


图 3-8 运行 Tcl 脚本

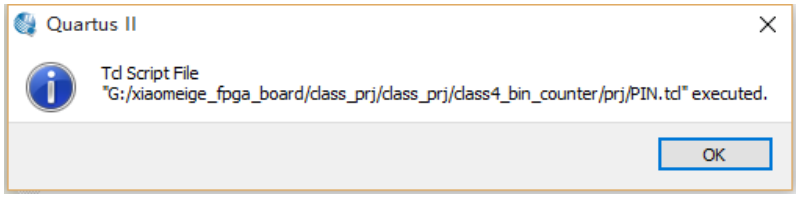


图 3-9 脚本运行成功

全编译后可以在 RTL viewer 中可以看到图 3-10 所示的硬件逻辑电路，也存在前面讲到的加法器、比较器、寄存器以及选择器构成的计数器。下载到开发板中可以看到图 3-11 现象，LED0 以 500ms 的时间进行闪烁，如果有示波器也可以测量这时候的引脚波形进行观察。

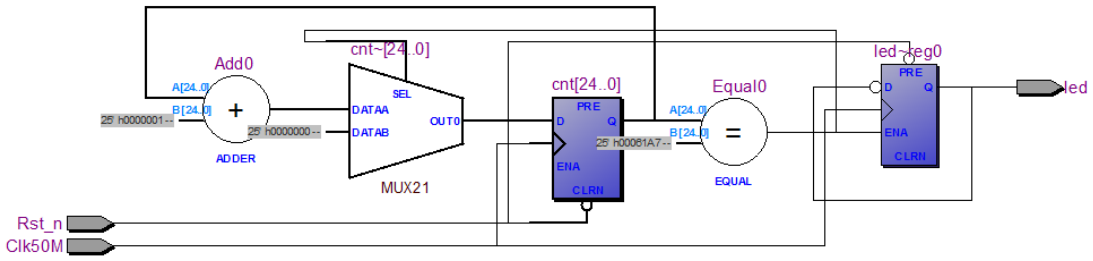


图 3-10 RTL viewer

图 3-11 实验现象

至此，就完成了一个基本的时序逻辑，计数器的设计。请以此为基础自行设计使得每个灯以不同的频率闪烁，并进行仿真以及板级验证。

如有更多问题，欢迎加入芯航线 FPGA 技术支持群交流学习：472607506

小梅哥
芯航线电子工作室