奋斗的小孩之 altera 系列

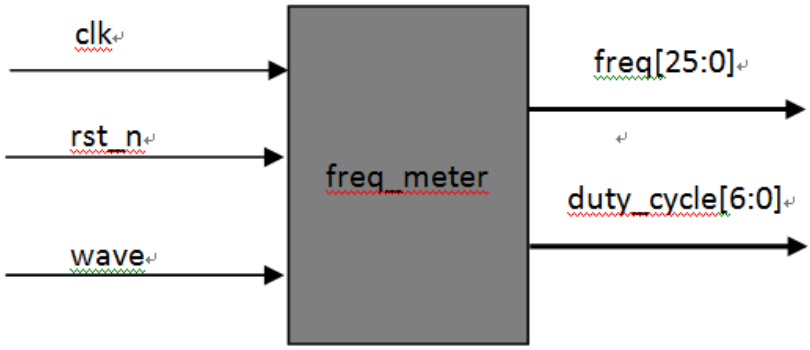
第二十篇 频率计

对于每一个的小实验，我们都可以把它看作是一个小项目，逐步的去分析，设计，调试，最后完成功能。下面我们就开始我们的“小项目”。

项目名称：频率计

具体要求：检测方波的频率和占空比。

通过分析上述的“项目名称”和“具体要求”，我们可以设计出如下的架构：



wave：方波输入

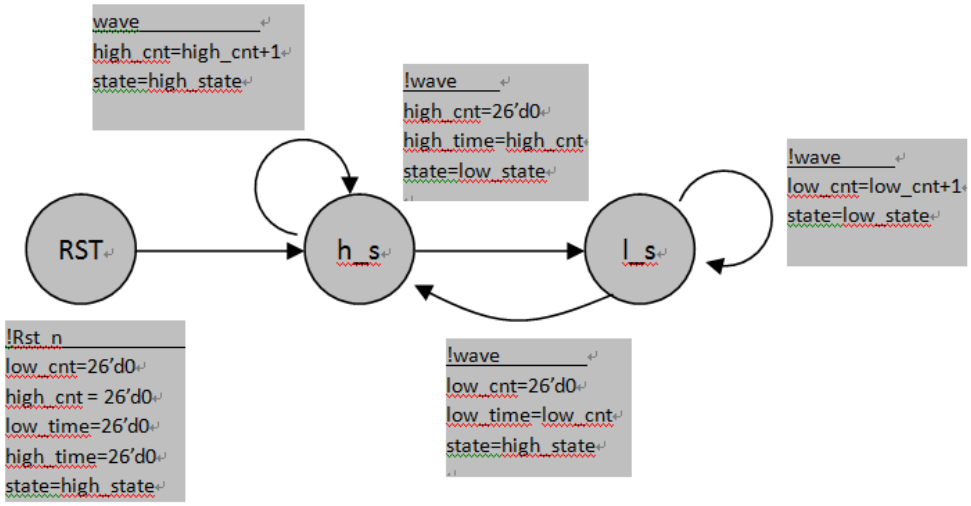
freq：测试出的方波频率

duty_cycle:测试出的方波占空比

wave 是经过处理后的数字信号，不是外界的模拟信号。

系统设计：

1. 工程的名称：freq_meter。
2. 测试出高电平的时间和低电平的时间，然后经过计算得出频率和占空比。
3. 状态转移图如下：



h_s：代表 high_state;

l_s：代表 low_state

low_cnt:低电平计数器

high_cnt:高电平计数器

low_time：低电平的周期数（周期数*周期就是时间）

high_time：高电平的周期数

上述计数器的位宽都是 26 位,因为笔者想要想要可以测试到 1Hz，故而需要计数到 50_000_000。

设计代码如下：

```
/*
模块名称：freq_meter
模块功能：检测方波的频率和占空比。
编写时间：2016-08-15
作者：至芯科技----奋斗的小孩
邮箱：zxopenhxs@126.com
*/
module freq_meter (clk, rst_n, wave, freq, duty_cycle);

    input clk;
    input rst_n;
    input wave;

    output [25:0] freq;
    output [6:0] duty_cycle;

    reg [25:0] low_cnt;
    reg [25:0] high_cnt;
    reg [25:0] low_time;
    reg [25:0] high_time;
```

```
reg state;

localparam high_state = 1'b0;
localparam low_state = 1'b1;

always @ (posedge clk or negedge rst_n)
   begin
      if (!rst_n)
         begin
            low_cnt <= 26'd0;
            high_cnt <= 26'd0;
            low_time <= 26'd0;
            high_time <= 26'd0;
            state <= high_state;
         end
      else
         begin
            case (state)
               high_state : begin
                     if (wave == 1'b1)
```

```
                begin
                    high_cnt <= high_cnt + 1'b1;
                    state <= high_state;
                end
            else
                begin
                    high_cnt <= 26'd0;
                    high_time <= high_cnt;
                    state <= low_state;
                end
        end

    low_state   : begin
        if (wave == 1'b0)
            begin
                low_cnt <= low_cnt + 1'b1;
                state <= low_state;
            end
        else
            begin
                low_cnt <= 26'd0;
```

```
                        low_time <= low_cnt;
                        state <= high_state;
                    end
                end

                default : state <= low_state;

            endcase
        end
    end

assign freq = 1_000_000_000/(low_time * 20 + high_time * 20);
assign duty_cycle = (high_time * 100)/(high_time + low_time);

endmodule
```

解析：

```
high_cnt <= 26'd0;
high_time <= high_cnt;
state <= low_state;
```

上面的描述是同时进行的，不要拿软件的想法去理解 HDL 语言。

读者一定要谨记。

频率=1s/T，T=高电平的时间+低电平的时间。时间=周期数*周期。占空比=（高电平的时间/周期）100%。我们的时间单位都是以 ns 来计算的，所以要把 1s 换成 1_000_000_000ns,驱动时钟是 50MHz 的，周期为 20ns。计算占空比的时候，我们把周期 20ns 全部省略了。所以计算公式如下：

```
freq = 1_000_000_000/(low_time * 20 + high_time * 20);
duty_cycle = (high_time * 100)/(high_time + low_time);
```

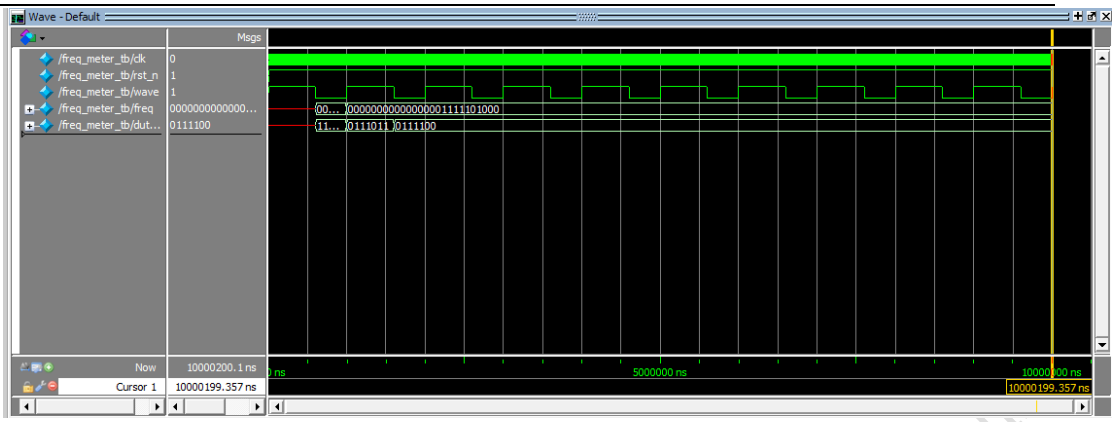
激励代码如下：

```
/*
模块名称：freq_meter_tb
模块功能：为 freq_meter 模块提供激励信号
编写时间：2016-08-15
作者：至芯科技----奋斗的小孩
邮箱：zxopenhxs@126.com
*/
`timescale 1ns/1ps

module freq_meter_tb;
```

```
reg clk;

reg rst_n;

reg wave;


wire [25:0] freq;

wire [6:0] duty_cycle;


initial begin

   clk = 1'b1;

   rst_n = 1'b0;

   # 200.1

   rst_n = 1'b1;


   # 1_000_000_0//仿真 10ms

   $stop;

end


always # 10 clk = ~clk;


initial begin

   wave = 1'b1;
```

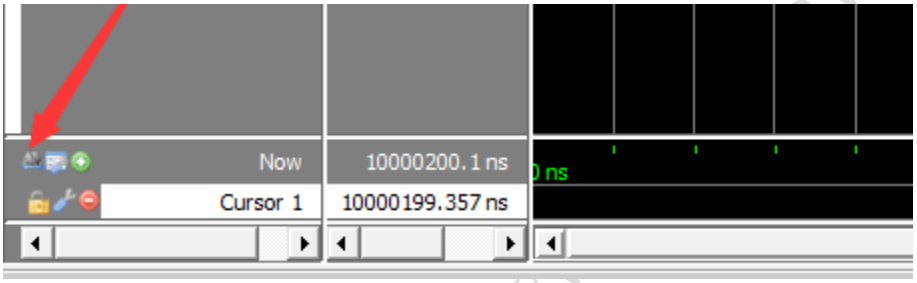
```
        forever begin//产生占空比为 60%，频率为 1KHz 的方波
            # 600_000
            wave = 1'b0;
            # 400_000
            wave = 1'b1;
        end
    end

    freq_meter freq_meter_dut(
            .clk(clk),
            .rst_n(rst_n),
            .wave(wave),
            .freq(freq),
            .duty_cycle(duty_cycle)
        );

endmodule
```

仿真波形如下：

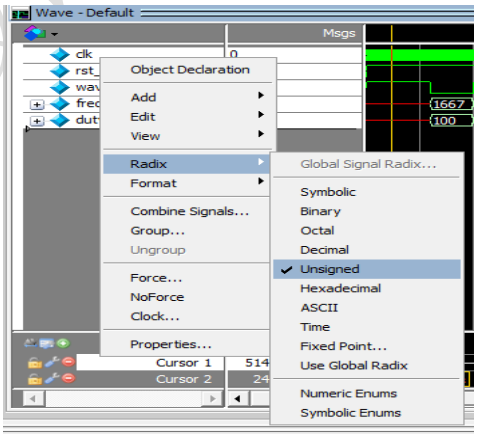


下面笔者介绍一些 modelsim 中的基本操作：

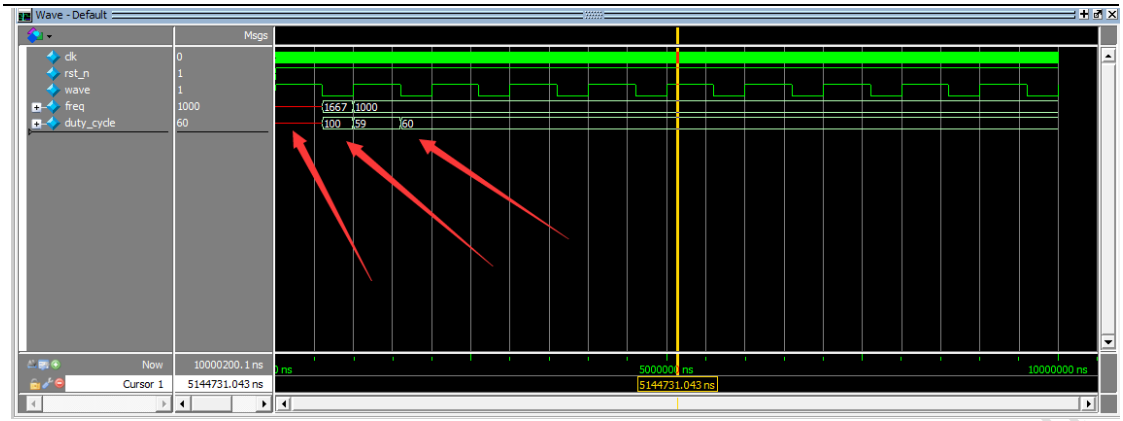
1. 简化信号名称：点击功能按钮（如下图）。



2. 更改显示的进制:选中你想更改进制的信号，然后右击，选择 Radix，选择你想选择的进制。



设置完成之后，仿真图如下：



因为前面还没有测试完一个周期，所以出现了不准确的数值。当测试完一个周期后，数值很准确，几乎没有误差。

友情提示：被测试的频率越大误差越大，希望读者根据自己的误差要求去计算出可以测试的频率的范围。

设计正确。如果还是有不明白的读者可以发邮件到我邮箱或者加群询问。

制作人：奋斗的小孩　　fpga 交流群：282124839