第25章　简易频率计的设计与验证

25.1 章节导读

频率测量在电子设计领域和测量领域经常被使用，因此频率测量方法的学习和掌握是非常有必要的。在本章节，我们将为读者讲解等精度测量法的原理和实现方法，使用FPGA 结合所学知识设计并实现一个简易频率计。

25.2 理论学习

频率测量在诸多领域都有广泛的应用，常用的频率测量方法有两种，分别是频率测量法和周期测量法。

频率测量法：在时间 t 内对被测时钟信号的时钟周期 N 进行计数，然后求出单位时间内的时钟周期数，即为被测时钟信号的时钟频率。

周期测量法：先测量出被测时钟信号的时钟周期 T，然后根据频率 f = 1 / T 求出被测时钟信号的频率。

但是上述两种方法都会产生±1 个被测时钟周期的误差，在实际应用中有一定的局限性；而且根据两种方式的测量原理，很容易发现频率测量法适合于测量高频时钟信号，而周期测量法适合于低频时钟信号的测量，但二者都不能兼顾高低频率同样精度的测量要求。

等精度测量法与前两种方式不同，其最大的特点是，测量的实际门控时间不是一个固定值，它与被测时钟信号相关，是被测时钟信号周期的整数倍。在实际门控信号下，同时对标准时钟和被测时钟信号的时钟周期进行计数，再通过公式计算得到被测信号的时钟频率。

由于实际门控信号是被测时钟周期的整数倍，就消除了被测信号产生的±1 时钟周期的误差，但是会产生对标准时钟信号±1 时钟周期的误差。等精度测量原理示意图如图25-1 所示。

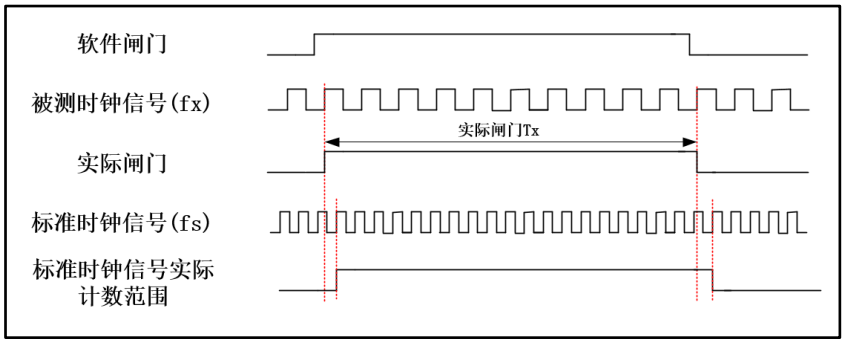

图 25-1　等精度测量原理示意图

结合等精度测量原理和原理示意图可得：被测时钟信号的时钟频率 fx 的相对误差与被测时钟信号无关；增大“软件闸门”的有效范围或者提高“标准时钟信号”的时钟频率 fs，可以减小误差，提高测量精度。

了解了等精度测量原理之后，我们来说明一下被测时钟信号的计算方法。

首先我们先分别对实际闸门下被测时钟信号和标准时钟信号的时钟周期进行计数。

实际闸门下被测时钟信号周期数为 X，设被测信号时钟周期为 T_{fx}，它的时钟频率 fx = $1/T_{fx}$，由此可得等式：X * T_{fx} = X / fx = T_x(实际闸门)。

实际闸门下标准时钟信号周期数为 Y，设被测信号时钟周期为 T_{fs}，它的时钟频率 fs = $1/T_{fs}$，由此可得等式：Y * T_{fs} = Y / fs = T_x(实际闸门)。

其次，将两等式结合得到只包含各自时钟周期计数和时钟频率的等式：X / fx = Y / fs = T_x(实际闸门)，等式变换，得到被测时钟信号时钟频率计算公式：fx = X * fs / Y。

最后，将已知量标准时钟信号时钟频率 fs 和测量量 X、Y 带入计算公式，得到被测时钟信号时钟频率 fx。

25.3 实战演练

在理论学习小结，我们对等精度测量的相关知识做了讲解，接下来，我们根据等精度测量原理设计一个简易频率计。

25.3.1 实验目标

利用所学知识，设计一个基于等精度测量原理的简易频率计，对输入的未知时钟信号做频率测量，并对测量结果进行在线抓取。

要求：标准时钟信号频率为 100MHz，实际闸门时间大于或等于 1s，目的是减小误差，提高测量精度。

25.3.2 硬件资源

如图 25-2 所示，使用板卡引出 I/O 口 L13 作为模拟测试时钟输出端口；使用板卡引出 I/O 口 M13 作为待测试时钟输入端口。

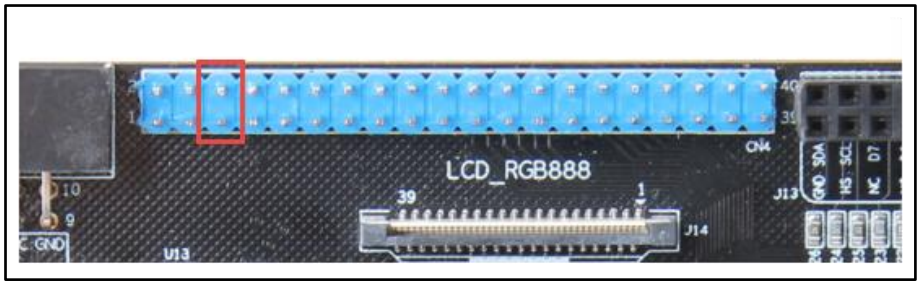

图 25-2 硬件资源

25.3.3 程序设计

了解了实验要求之后，根据等精度测量相关知识，我们开始实验工程的程序设计。接下来我们将会通过整体说明和分步介绍的方式，对整个实验工程做详细说明，带领读者一步步实现简易频率计的设计。

1. 整体说明

根据实验要求，结合等精度测量相关知识，实验工程整体框架如图 25-3 所示；子功能模块的简要描述，具体见表格 25-1。

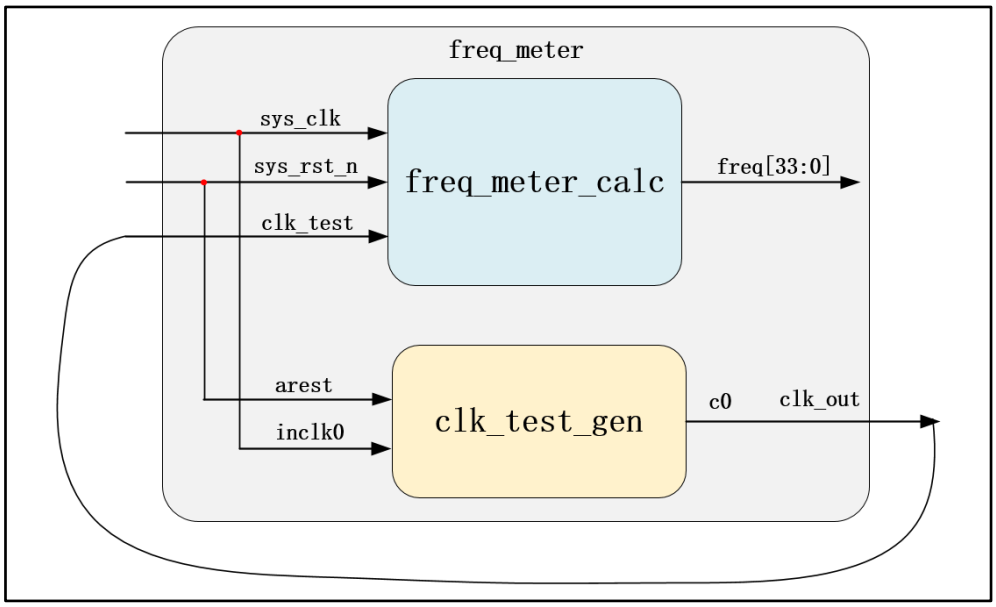

图 25-3 实验工程整体框图

表格 25-1 子功能模块简介

模块名称	功能描述
freq_meter	顶层模块
clk_test_gen	时钟生成模块，生成待检测时钟
freq_meter_calc	频率计算模块，检测并计算时钟频率

由图表可知，本实验工程包括 3 个子模块，其中频率计算模块(freq_meter_calc)是实验工程的核心模块，他将输入的待检测信号利用等精度测量法进行计算，得出被测时钟信号时钟频率并输出；被测时钟生成模块(clk_test_gen)负责产生某一频率的待检测时钟信号；最后的顶层模块(freq_meter)，内部将上述 2 个子功能模块实例化其中，连接各自对应信号，外部输入时钟、复位和待检测信号，输出段选、位选和待检测数据。

被测时钟生成模块(clk_test_gen)为调用 IP 核生成，关于 IP 核的调用前面章节有详细介绍，该模块内容不再赘述。

接下来，我们会分别对频率计算模块(freq_meter_calc)和顶层模块(freq_meter)的内容做一下介绍。

注：由频率计算模块输出的测量结果的单位为 Hz，为提高频率计测量范围，将结果除以 1000，即单位为 MHz；被测时钟生成模块(clk_test_gen)负责产生待检测时钟信号，如有条件的读者可用信号发生器代替该模块，直接输入待检测时钟信号。

2. 频率计算模块

模块框图

频率计算模块的作用是：使用等精度测量法对输入的待检测时钟信号进行频率测量，并将测量结果输出。频率计算模块框图，具体见图 25-4；模块输入输出端口描述，具体见表格 25-2。

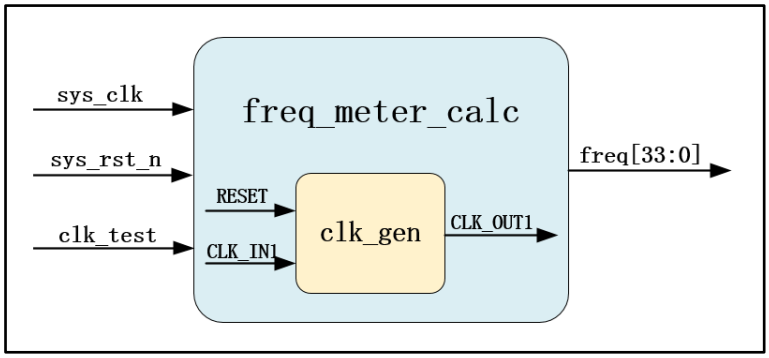

图 25-4 频率计算模块框图

模块内部实例化一个时钟生成 IP 核，负责将 50MHz 系统时钟信号(sys_clk)倍频生成 100MHz 标准时钟。

为什么要使用 100MHz 时钟信号作为标志信号呢？在前文的等精度测量原理中，我们知道，被测时钟信号的时钟频率 fx 的相对误差与被测时钟信号无关；增大“软件闸门”的有效范围或者提高“标准时钟信号”的时钟频率 fs，可以减小误差，提高测量精度。

使用 100 MHz 时钟作为标准时钟信号，实际闸门时间大于或等于 1s，就可使测量的最大相对误差小于或等于 10^{-8}，即精度达到 1 / 100 MHz，大大提高了测量精度。

表格 25-2 模块输入输出信号功能描述

信号	位宽	类型	功能描述
sys_clk	1Bit	Input	系统时钟，频率 50MHz
sys_rst_n	1Bit	Input	复位信号。低电平有效
clk_test	1Bit	Intput	待检测时钟信号输入
freq	34Bit	Output	频率测量结果输出

模块有 3 路输入信号：时钟(sys_clk)、复位(sys_rst_n)和待检测时钟信号(clk_test)；有输出信号 1 路，输出频率测量结果(freq)。

波形图绘制

在模块框图小节，我们已经对本模块做了简单介绍，接下来，我们将通过波形图的绘制，对各信号波形进行详细说明，教会读者等精度测量的实现方法。频率计算模块整体波形图如图 25-5 所示。

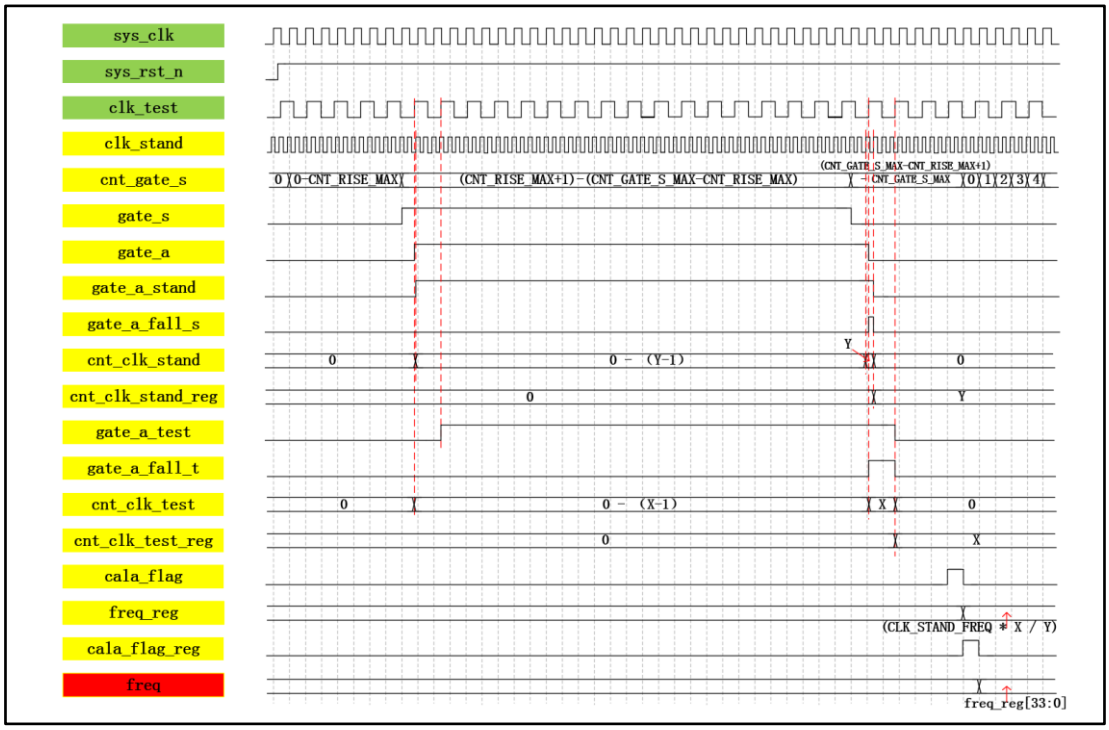

图 25-5 频率计算模块整体波形图

结合等精度测量法原理，我们对各信号波形的设计与实现进行详细说明。

第一部分：软件闸门 gate_s 及相关信号的设计与实现

由等精度测量原理可知，实现等精度测量必不可少的是实际闸门，而实际闸门是由软件闸门得来，所以我们先来生成一下软件闸门。我们计划一个完整周期的软件闸门为 1.5s，前 0.25s 保持低电平，中间 1s 保持高电平，最后 0.25s 保持低电平。低电平部分是为了将各计数器清 0，并计算待测时钟信号时钟频率；高电平部分就是软件闸门有效部分，高电平保持 1s 是为了提高测试精度，在前文已有提及。

软件闸门的生成我们需要声明计数器进行时间计数，计数时钟使用系统时钟 sys_clk。声明软件闸门计数器 cnt_gate_s，计数时钟为 50MHz 系统时钟，时钟周期为 20ns，计数器 cnt_gate_s 初值为 0，在(0 – CNT_GATE_S_MAX)范围内循环计数。

声明软件闸门 gate_s，只有计数器 cnt_gate_s 计数在((CNT_RISE_MAX+1)-(CNT_GATE_S_MAX-CNT_RISE_MAX))范围内保持有效高电平，高电平保持时间为 1s，其他时刻均为低电平。两信号波形图如下。

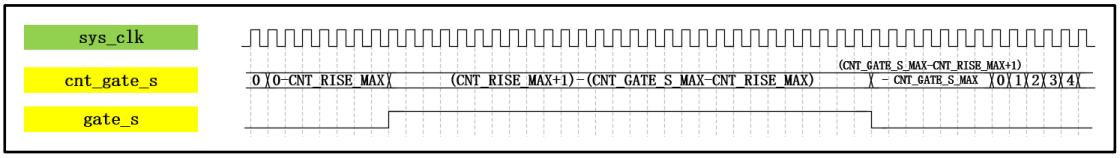

图 25-6 gate_s、cnt_gate_s 信号波形图

第二部分：实际闸门 gate_a 的设计与实现

生成软件闸门后，使用被测时钟对软件闸门进行同步生成实际闸门 gate_a，实际闸门波形图如下。

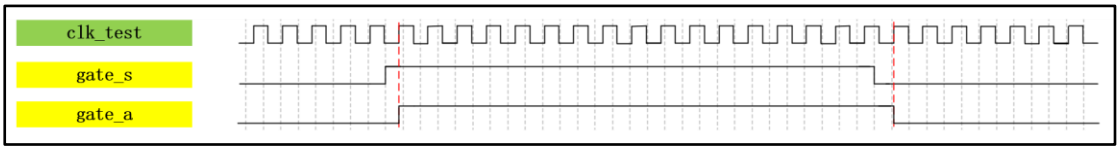

图 25-7 gate_a 信号波形图

第三部分：实际闸门下，标准信号和被测信号时钟计数相关信号的波形设计与实现

在实际闸门下，分别对标准信号和被测信号的时钟周期进行计数。声明计数器 cnt_clk_stand，在实际闸门下对标准时钟信号 clk_stand 进行时钟周期计数；声明计数器 cnt_clk_test，在实际闸门下对被测时钟信号 clk_test 进行时钟周期计数，两计数器波形如下。

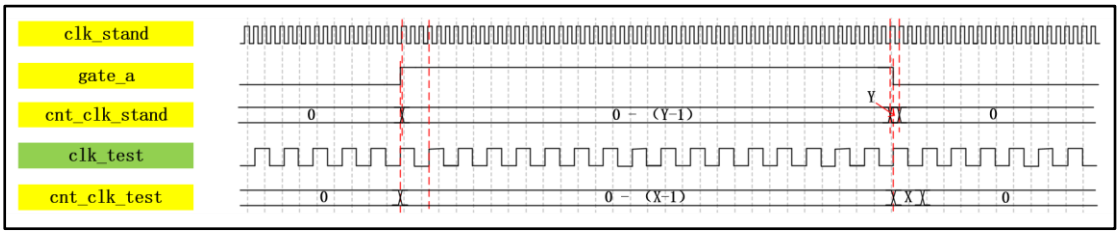

图 25-8 cnt_clk_stand、cnt_clk_test 信号波形图

计数器 cnt_clk_stand、cnt_clk_test 在实际闸门下计数完成后，需要进行数据清零，方便下次计数。但是被测时钟频率的计算需要计数器的数据，所以在计数器数据清零之前我们需要将计数器数据做一下寄存，对于数据寄存的时刻，我们选择实际闸门的下降沿。

声明寄存器 cnt_clk_stand_reg；在标准时钟信号 clk_stand 同步下对实际闸门打一拍得到 gate_a_s；使用实际闸门 gate_a 和 gate_a_s 得到标准时钟下的实际闸门下降沿标志信号 gate_a_fall_stand。当 gate_afall_stand 信号为高电平时，将计数器 cnt_clk_stand 数值赋值给寄存器 cnt_clk_stand_reg。

对于计数器 cnt_clk_test 的数值寄存，我们使用相同的方法，声明寄存器 cnt_clk_test_reg；在被检测时钟信号 clk_test 同步下对实际闸门打一拍得到 gate_a_t；使用实际闸门 gate_a 和 gate_a_t 得到被检测时钟下的实际闸门下降沿标志信号 gate_a_fall_test。当 gate_a_fall_test 信号为高电平时，将计数器 cnt_clk_test 数值赋值给 cnt_clk_test_reg。

上述各信号的信号波形如图 25-9 所示。

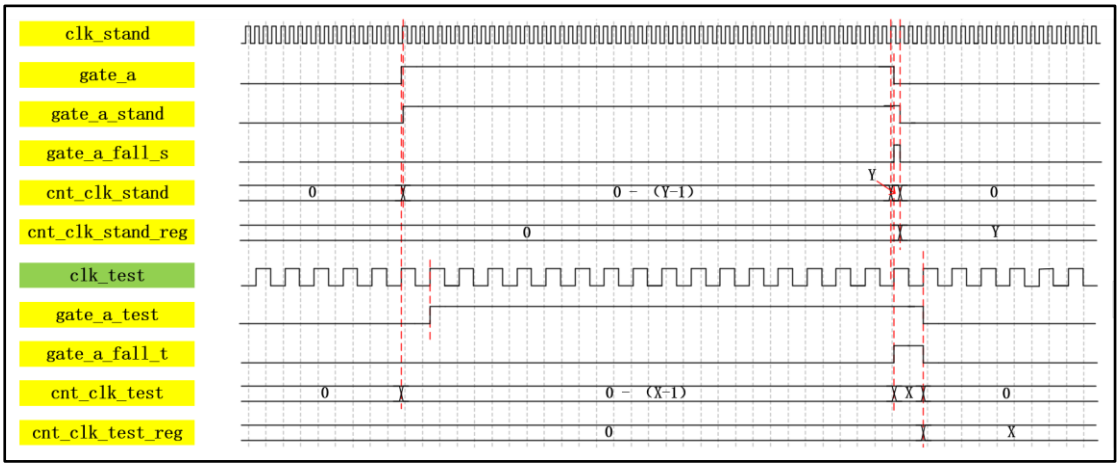

图 25-9 标准信号和被测信号时钟计数相关信号波形图

第四部分：频率计算结果 freq 等相关信号波形的设计与实现

实际闸门下的标准时钟和被测时钟的周期个数已经完成计数，且对结果进行了寄存，标准时钟信号的时钟频率为已知量，得到这些参数，结合公式可以进行频率的求解。同时，新的问题出现，在哪一时刻进行数据求解。

我们可以利用最初声明的软件闸门计数器 cnt_gate_s，声明计算标志信号 calc_flag，在计数器 cnt_gate_s 计数到最大值，将 calc_flag 拉高一个时钟周期的高电平作为计算标志，计算被检测时钟信号时钟频率 freq_reg(注意变量位宽是否满足要求)；然后在系统时钟下将计算标志信号 calc_flag 打一拍，得到时钟频率输出标志信号 calc_flag_reg，当时钟频率输出标志信号 calc_flag_reg 为高电平时，将时钟频率计算结果 freq_reg 赋值给输出信号 freq。各信号波形图如下。

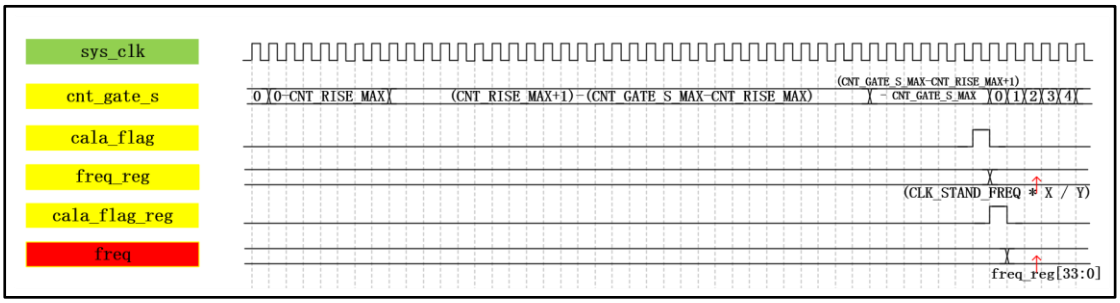

图 25-10 calc_flag、freq 信号波形图

到了这里，频率计算模块涉及的各信号波形均已设计并实现，经过整合后就得到频率计算模块整体波形图。本模块波形图的设计仅供参考，读者也可按照自己思路设计波形图。

代码编写

波形图各信号讲解完毕，参照绘制的波形图编写模块参考代码。频率计数模块参考代码，具体见代码清单 25-1。

代码清单 25-1 频率计算模块参考代码(freq_meter_calc.v)

```
1 module  freq_meter_calc
2 (
```

```
    input   wire            sys_clk     ,   //系统时钟,频率 50MHz
    input   wire            sys_rst_n   ,   //复位信号,低电平有效
    input   wire            clk_test    ,   //待检测时钟

    output  reg     [33:0]  freq            //待检测时钟频率

);
//*********************************************************************//
//****************** Parameter And Internal Signal ********************//
//*********************************************************************//
//parameter define
parameter   CNT_GATE_S_MAX  =   28'd74_999_999  ,   //软件闸门计数器计数最大值
            CNT_RISE_MAX    =   28'd12_500_000  ;   //软件闸门拉高计数值
parameter   CLK_STAND_FREQ  =   28'd100_000_000 ;   //标准时钟时钟频率
//wire  define
wire            clk_stand           ;   //标准时钟,频率 100MHz
wire            gate_a_fall_s       ;   //实际闸门下降沿(标准时钟下)
wire            gate_a_fall_t       ;   //实际闸门下降沿(待检测时钟下)

//reg   define
reg     [27:0]  cnt_gate_s          ;   //软件闸门计数器
reg             gate_s              ;   //软件闸门
reg             gate_a              ;   //实际闸门
reg             gate_a_stand        ;   //实际闸门打一拍(标准时钟下)
reg             gate_a_test         ;   //实际闸门打一拍(待检测时钟下)
reg     [47:0]  cnt_clk_stand       ;   //标准时钟周期计数器
reg     [47:0]  cnt_clk_stand_reg   ;   //实际闸门下标志时钟周期数
reg     [47:0]  cnt_clk_test        ;   //待检测时钟周期计数器
reg     [47:0]  cnt_clk_test_reg    ;   //实际闸门下待检测时钟周期数
reg             calc_flag           ;   //待检测时钟时钟频率计算标志信号
reg     [63:0]  freq_reg            ;   //待检测时钟频率寄存
reg             calc_flag_reg       ;   //待检测时钟频率输出标志信号

//*********************************************************************//
//***************************** Main Code *****************************//
//*********************************************************************//
//cnt_gate_s:软件闸门计数器
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt_gate_s  <=  28'd0;
    else    if(cnt_gate_s == CNT_GATE_S_MAX)
        cnt_gate_s  <=  28'd0;
    else
        cnt_gate_s  <=  cnt_gate_s + 1'b1;

//gate_s:软件闸门
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        gate_s  <=  1'b0;
    else    if((cnt_gate_s>= CNT_RISE_MAX)
                && (cnt_gate_s <= (CNT_GATE_S_MAX - CNT_RISE_MAX)))
        gate_s  <=  1'b1;
    else
        gate_s  <=  1'b0;

//gate_a:实际闸门
always@(posedge clk_test or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        gate_a  <=  1'b0;
    else
        gate_a  <=  gate_s;

```

```
65 //cnt_clk_stand:标准时钟周期计数器,计数实际闸门下标准时钟周期数
66 always@(posedge clk_stand or negedge sys_rst_n)
67     if(sys_rst_n == 1'b0)
68         cnt_clk_stand   <=  48'd0;
69     else    if(gate_a == 1'b0)
70         cnt_clk_stand   <=  48'd0;
71     else    if(gate_a == 1'b1)
72         cnt_clk_stand   <=  cnt_clk_stand + 1'b1;
73
74 //cnt_clk_test:待检测时钟周期计数器,计数实际闸门下待检测时钟周期数
75 always@(posedge clk_test or negedge sys_rst_n)
76     if(sys_rst_n == 1'b0)
77         cnt_clk_test    <=  48'd0;
78     else    if(gate_a == 1'b0)
79         cnt_clk_test    <=  48'd0;
80     else    if(gate_a == 1'b1)
81         cnt_clk_test    <=  cnt_clk_test + 1'b1;
82
83 //gate_a_stand:实际闸门打一拍(标准时钟下)
84 always@(posedge clk_stand or negedge sys_rst_n)
85     if(sys_rst_n == 1'b0)
86         gate_a_stand    <=  1'b0;
87     else
88         gate_a_stand    <=  gate_a;
89
90 //gate_a_fall_s:实际闸门下降沿(标准时钟下)
91 assign  gate_a_fall_s = ((gate_a_stand == 1'b1) && (gate_a == 1'b0))
92                         ? 1'b1 : 1'b0;
93
94 //cnt_clk_stand_reg:实际闸门下标志时钟周期数
95 always@(posedge clk_stand or negedge sys_rst_n)
96     if(sys_rst_n == 1'b0)
97         cnt_clk_stand_reg   <=  32'd0;
98     else    if(gate_a_fall_s == 1'b1)
99         cnt_clk_stand_reg   <=  cnt_clk_stand;
100
101 //gate_a_test:实际闸门打一拍(待检测时钟下)
102 always@(posedge clk_test or negedge sys_rst_n)
103     if(sys_rst_n == 1'b0)
104         gate_a_test <=  1'b0;
105     else
106         gate_a_test <=  gate_a;
107
108 //gate_a_fall_t:实际闸门下降沿(待检测时钟下)
109 assign  gate_a_fall_t = ((gate_a_test == 1'b1) && (gate_a == 1'b0))
110                         ? 1'b1 : 1'b0;
111
112 //cnt_clk_test_reg:实际闸门下待检测时钟周期数
113 always@(posedge clk_test or negedge sys_rst_n)
114     if(sys_rst_n == 1'b0)
115         cnt_clk_test_reg   <=  32'd0;
116     else    if(gate_a_fall_t == 1'b1)
117         cnt_clk_test_reg   <=  cnt_clk_test;
118
119 //calc_flag:待检测时钟时钟频率计算标志信号
120 always@(posedge sys_clk or negedge sys_rst_n)
121     if(sys_rst_n == 1'b0)
122         calc_flag   <=  1'b0;
123     else    if(cnt_gate_s == (CNT_GATE_S_MAX - 1'b1))
124         calc_flag   <=  1'b1;
125     else
126         calc_flag   <=  1'b0;
127
128 //freq_reg:待检测时钟信号时钟频率寄存
129 always@(posedge sys_clk or negedge sys_rst_n)
```

```
130     if(sys_rst_n == 1'b0)
131         freq_reg    <=  64'd0;
132     else    if(calc_flag == 1'b1)
133         freq_reg    <=  (CLK_STAND_FREQ * cnt_clk_test_reg / cnt_clk_stand_reg);
134
135 //calc_flag_reg:待检测时钟频率输出标志信号
136 always@(posedge sys_clk or negedge sys_rst_n)
137     if(sys_rst_n == 1'b0)
138         calc_flag_reg   <=  1'b0;
139     else
140         calc_flag_reg   <=  calc_flag;
141
142 //freq:待检测时钟信号时钟频率
143 always@(posedge sys_clk or negedge sys_rst_n)
144     if(sys_rst_n == 1'b0)
145         freq    <=  34'd0;
146     else    if(calc_flag_reg == 1'b1)
147         freq    <=  freq_reg[33:0];
148
149 //*****************************************************************//
150 //************************** Instantiation ************************//
151 //*****************************************************************//
152 //---------- clk_gen_inst ----------
153 clk_gen clk_gen_inst
154 (
155     .resrt    (~sys_rst_n ),
156     .clk_in1  (sys_clk    ),
157
158     .clk_out1 (clk_stand  )
159 );
160
161 endmodule
```

参考代码编写完成，对于代码的仿真验证，等到顶层模块介绍完毕后，对整体工程进行仿真，不再对本模块进行单独仿真。

3. 顶层模块

模块框图

顶层模块较为简单，内部实例化各子功能模块，连接各自对应信号，顶层模块框图，具体见图 25-11；外部有 3 路输入、1 路输出，共 4 路信号。输入为时钟、复位和待检测时钟信号；生成的待检测时钟信号，具体见表格 25-3。

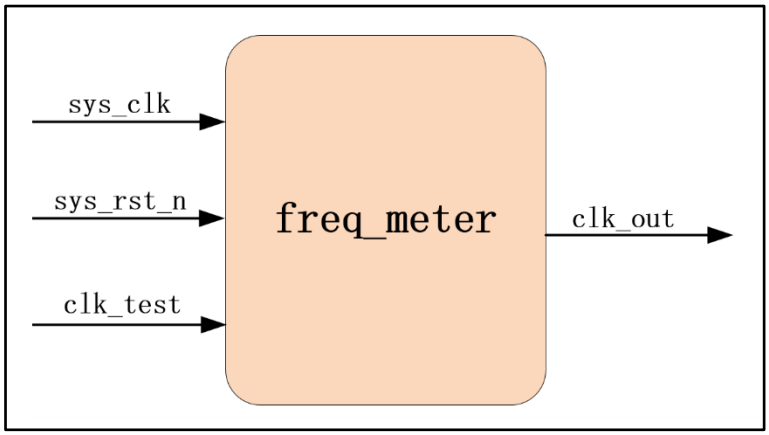

图 25-11 顶层模块框图

表格 25-3 模块输入输出信号功能描述

信号	位宽	类型	功能描述
sys_clk	1Bit	Input	系统时钟，频率 50MHz
sys_rst_n	1Bit	Input	复位信号。低电平有效
clk_test	1Bit	Intput	待检测时钟信号输入
clk_out	1Bit	Output	输出待检测时钟信号

代码编写

顶层模块较为简单，无需波形图的绘制，直接编写顶层模块参考代码。顶层模块参考代码，具体见代码清单 25-2。

代码清单 25-2 顶层模块参考代码(freq_meter.v)

```
1  module  freq_meter
2  (
3      input   wire            sys_clk     ,   //系统时钟,频率 50MHz
4      input   wire            sys_rst_n   ,   //复位信号,低电平有效
5      input   wire            clk_test    ,   //待检测时钟
6
7      output  wire            clk_out         //生成的待检测时钟
8  );
9
10 //wire  define
11 wire    [33:0]  freq    ;   //计算得到的待检测信号时钟频率
12
13 //*****************************************************************//
14 //************************** Instantiation ************************//
15 //*****************************************************************//
16 //---------- clk_gen_test_inst ----------
17 clk_test_gen    clk_gen_test_inst
18 (
19     .reset     (~sys_rst_n ),  //复位端口,高电平有效
20     .clk_in1   (sys_clk    ),  //输入系统时钟
21
22     .clk_out1  (clk_out    )   //输出生成的待检测时钟信号
23 );
24
25 //------------- freq_meter_calc_inst --------------
26 freq_meter_calc freq_meter_calc_inst
27 (
28     .sys_clk    (sys_clk    ),   //系统时钟,频率 50MHz
29     .sys_rst_n  (sys_rst_n  ),   //复位信号,低电平有效
30     .clk_test   (clk_test   ),   //待检测时钟
31
32     .freq       (freq       )    //待检测时钟频率
33 );
34
35 endmodule
```

4. 仿真验证

仿真代码编写

顶层模块参考代码编写完成，实验工程通过编译，整个实验工程也已介绍完毕，开始对实验工程进行整体仿真。仿真参考代码如代码清单 25-3 所示。

代码清单 25-3 顶层模块仿真参考代码(tb_freq_meter.v)

```
1  `timescale  1ns/1ns
2  module tb_freq_meter();
3
4  //*********************************************************************//
5  //****************** Parameter And Internal Signal ********************//
6  //*********************************************************************//
7  //wire  define
8  wire    clk_out;
9
10 //reg   define
11 reg             sys_clk     ;
12 reg             sys_rst_n   ;
13 reg             clk_test    ;
14
15 //*********************************************************************//
16 //***************************** Main Code *****************************//
17 //*********************************************************************//
18 //时钟、复位、待检测时钟的生成
19 initial
20     begin
21         sys_clk     =   1'b1;
22         sys_rst_n   <=  1'b0;
23         #200
24         sys_rst_n   <=  1'b1;
25         #500
26         clk_test    =   1'b1;
27     end
28
29 always  #10     sys_clk =   ~sys_clk    ;   //50MHz 系统时钟
30 always  #100    clk_test=   ~clk_test   ;   //5MHz 待检测时钟
31
32 //重定义软件闸门计数时间,缩短仿真时间
33 defparam freq_meter_inst.freq_meter_calc_inst.CNT_GATE_S_MAX    = 240   ;
34 defparam freq_meter_inst.freq_meter_calc_inst.CNT_RISE_MAX      = 40    ;
35
36 //*********************************************************************//
37 //************************** Instantiation ****************************//
38 //*********************************************************************//
39 //------------- freq_meter_inst -------------
40 freq_meter  freq_meter_inst
41 (
42     .sys_clk     (sys_clk   ),   //系统时钟,频率 50MHz
43     .sys_rst_n   (sys_rst_n ),   //复位信号,低电平有效
44     .clk_test    (clk_test  ),   //待检测时钟
45
46     .clk_out     (clk_out   )    //生成的待检测时钟
47
48 );
49
50 endmodule
```

仿真波形分析

对顶层模块仿真参考代码进行仿真，仿真波形中，我们只查看频率计算模块的各信号仿真波形。频率计算模块整体仿真波形，具体见图 25-12；频率计算模块局部仿真波形，具体见图 25-13~图 25-19。

由整体和局部仿真波形可以看出，模块仿真波形和绘制波形图，各信号波形变化一致，模块通过仿真验证。

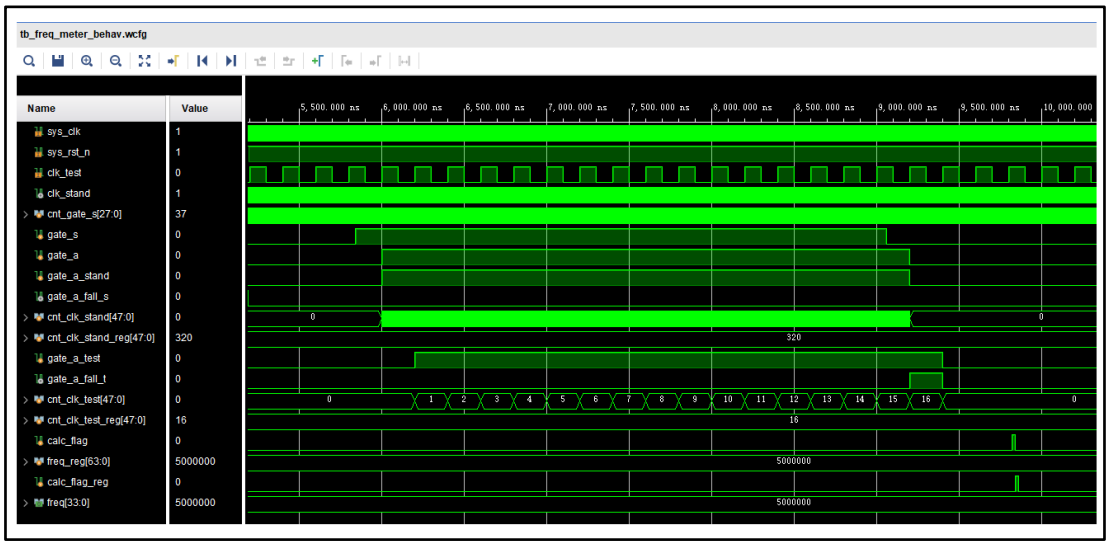

图 25-12 频率计算模块整体仿真波形图

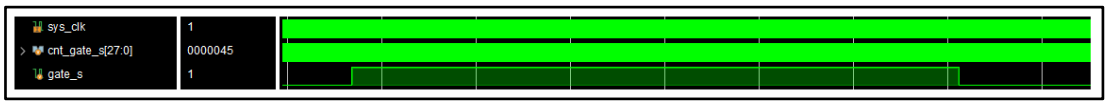

图 25-13 频率计算模块局部仿真波形图（一）

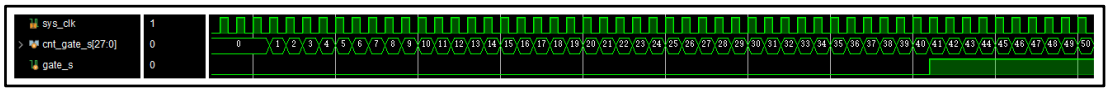

图 25-14 频率计算模块局部仿真波形图（二）

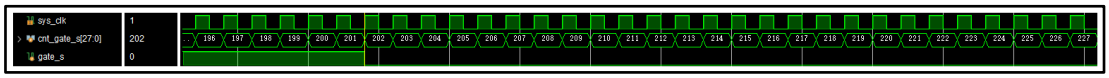

图 25-15 频率计算模块局部仿真波形图（三）

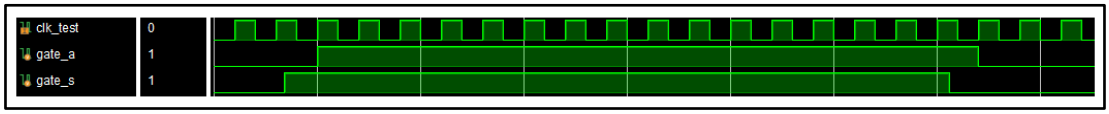

图 25-16 频率计算模块局部仿真波形图（四）

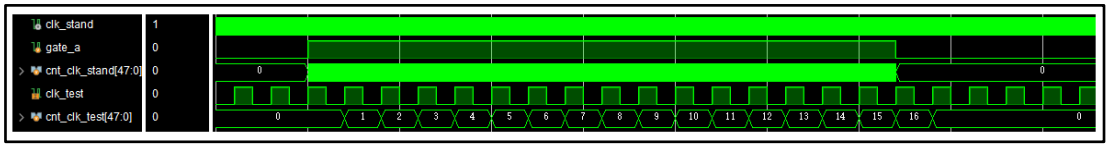

图 25-17 频率计算模块局部仿真波形图（五）

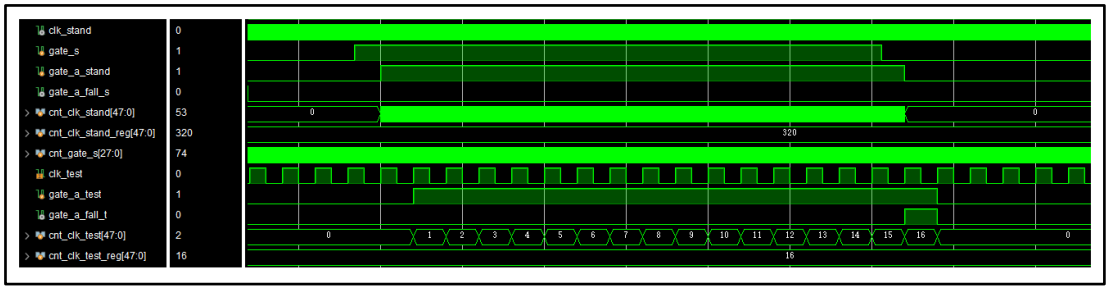

图 25-18 频率计算模块局部仿真波形图（六）

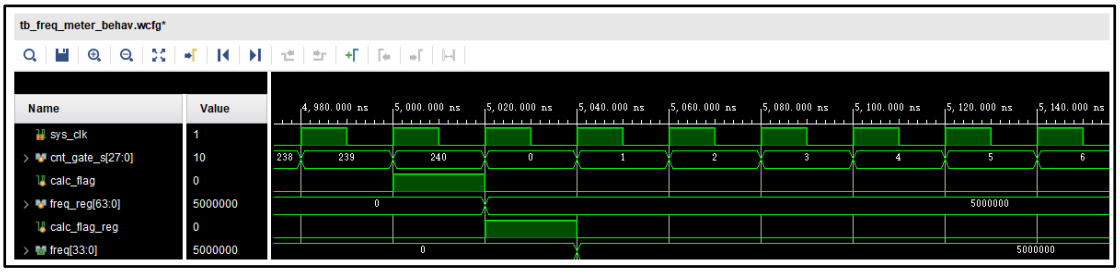

图 25-19 频率计算模块局部仿真波形图（七）

25.3.4 上板验证

1. 引脚约束

仿真验证通过后，准备上板验证，上板验证之前先要进行引脚约束。工程中各输入输出信号与开发板引脚对应关系如表格 25-4 所示。

表格 25-4 引脚分配表

信号名	信号类型	对应引脚	备注
sys_clk	Input	W19	时钟
sys_rst_n	Input	N15	复位
clk_test	Input	M13	待测试时钟输入端口
clk_out	Output	L13	模拟测试时钟输出端口

引脚约束参考代码如代码清单 25-4 所示：

代码清单 25-4 引脚约束参考代码（freq_meter.ucf）

```
#时钟复位
set_property PACKAGE_PIN Y19 [get_ports sys_rst_n]
set_property PACKAGE_PIN W19 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports sys_clk]
set_property IOSTANDARD LVCMOS33 [get_ports sys_rst_n]

#拓展 IO 口
set_property PACKAGE_PIN K22 [get_ports clk_test]
set_property PACKAGE_PIN K21 [get_ports clk_out]
set_property IOSTANDARD LVCMOS33 [get_ports clk_test]
set_property IOSTANDARD LVCMOS33 [get_ports clk_out]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_test_IBUF]
```

2. 结果验证

如图 25-20 所示，开发板连接 12V 直流电源和 Xilinx 下载器 JTAG 端口；使用短路帽或导线连接 L13、M13 I/O 口。线路正确连接后，打开开关为板卡上电。

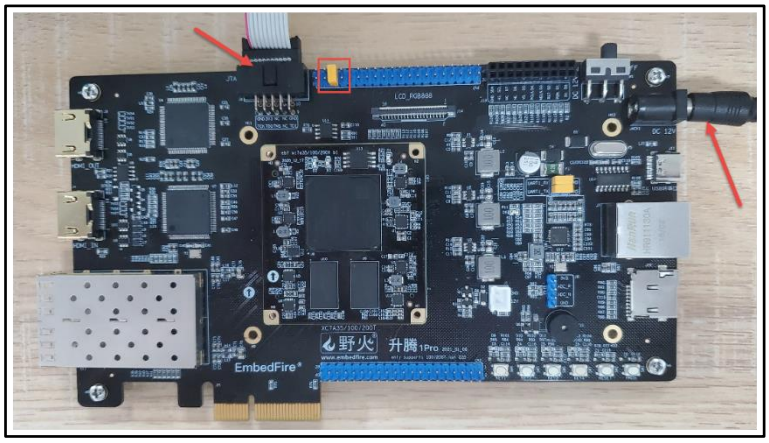

图 25-20 程序下载连线图

如图 25-21 所示，点击“Program”为开发板下载程序。

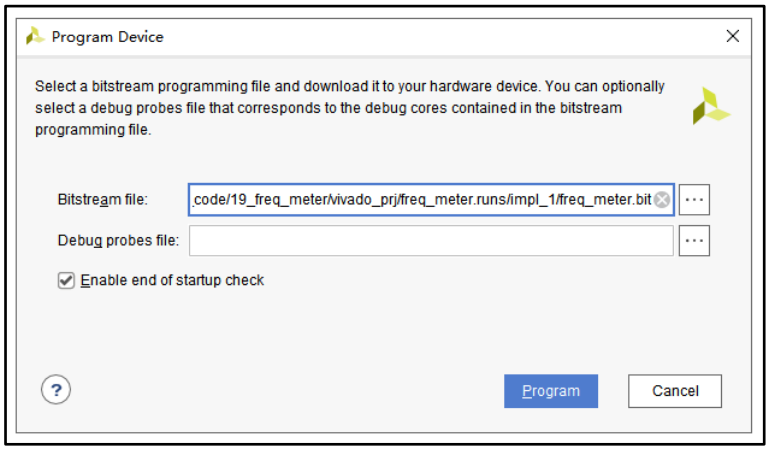

图 25-21 程序下载窗口

由图 25-22 可知，我们实验工程中 PLL 核模拟输出的待测时钟信号的时钟频率为 105.769MHz。

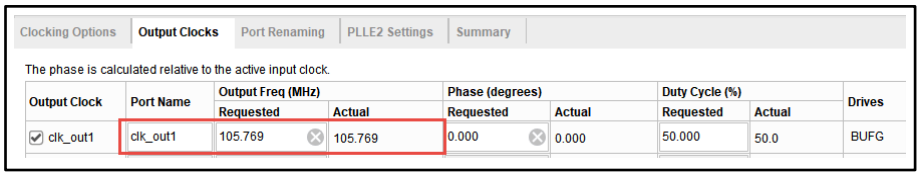

图 25-22 待测时钟信号时钟频率

如图 25-23 所示，我们使用 ChipScope 嵌入式逻辑分析仪在线抓取的频率值，其余我们需要检测的 PLL 核模拟输出额的待测时钟是一致的。

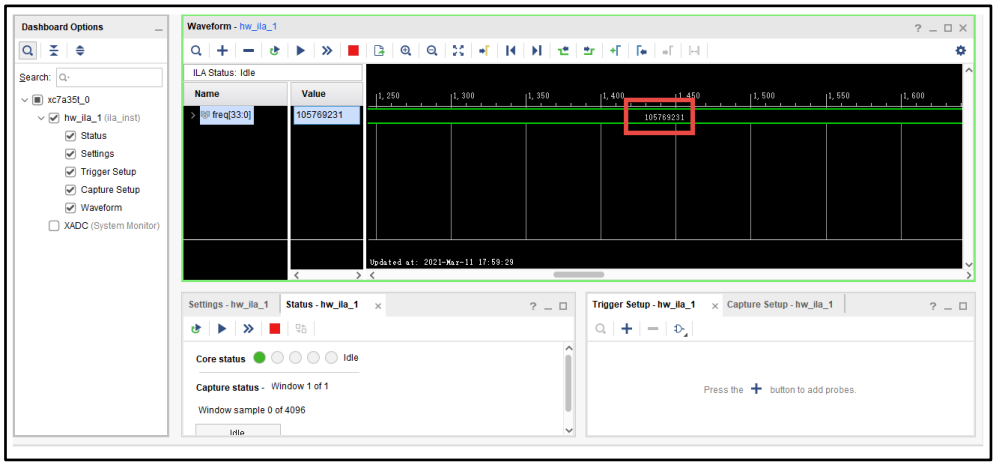

图 25-23 在线抓取频率值

25.4 章末总结

本章节带领读者学习了等精度测量法，并使用此方法设计并实现了简易频率计，对于等精度测量法的内容，希望读者要切实掌握。

25.5 拓展训练

1. 更改代码，调节基准时钟频率或软件闸门的保持时间对同一信号进行频率测量，观察测量结果，思考基准时钟频率大小和软件闸门保持时间对测量结果的影响；

2.使用其他两种频率测量法，设计简易频率计，与本实验工程比较测量结果。