实验五　综合设计电路实验

张蔚桐 *2015011493* 自 *55*

1　仿真和预习

实验要求设计数字电压表，显示输入正弦波电压的峰值。因此考虑将电路分割为如下几个部分进行设计

1.1　峰值提取部分

这部分输入将输入的正弦波电压转换为尽可能稳定的输出电压，输出电压的值为输入正弦波的峰峰值。

具体的电路如图 1所示，这是两个对称峰值检测电路分别检测电路的上峰值和下谷值。

其中，上半侧电路检测峰值，U1A 为一个输入阻抗变换和 U1B，二极管同时构成精密二极管，串接电容 C6 接地，完成正峰值检测。C6 并联 R10 完成放电过程，方便动态检测电路。在 U1B 输出正向峰值。

下半部电路输出谷值，这里就不加赘述了。

输出之后进行 R4C2 的一个简单的一节低通滤波虑去前面 C6 放电出现的不稳定震荡，这方面 R5C3，R8C4 组成的简答低通滤波均完成此项内容，且效果稳定。

之后，U2A 运放完成减法操作，检出峰峰值，之后滤波后阻抗变换跟随输出，稳定这个模块内部的工作，输出稳定电压。

经过仿真可以得到输出的电压准确度，稳定性均较好。

相对于娶她设计，这个电路能够应对多种波形输入，并且尤其适合中高频信号的输入。对于极低频信号，可能因为峰值检测电容放电的影响出现明显的纹波影响电路稳定性。

1.2　电压转换部分

这部分输入一个稳定的正电压，输出一个频率稳定的方波，具体的电路如图 2所示

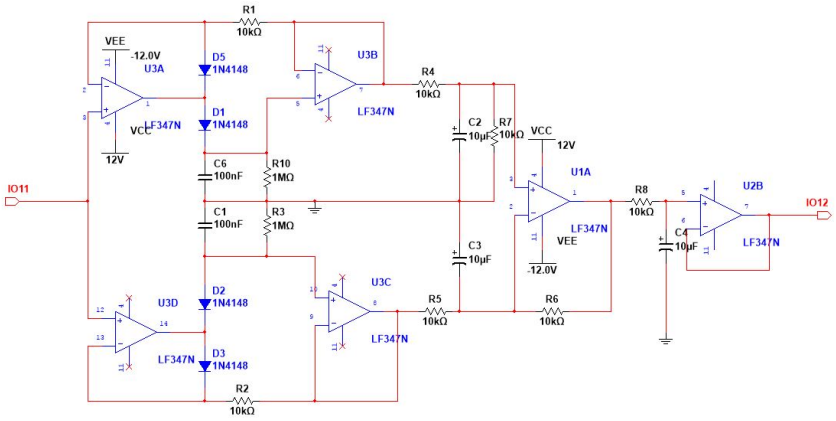

Fig. 1: 峰值提取电路

这个电路参考了模拟电子技术基础书上的压频转换电路的设计，具体原理略去。经过简单计算可以得到，电路在输入电压的控制下可以约为输出 $50U_I\mathrm{Hz/V}$ 频率的方波，经过稳压之后输出共后级电路处理。

电路涉及的电阻较多，因此受到电阻型号和相对误差的影响，电路的具体的参数（如选择的电阻电容）选择在电路搭建的过程中也需要进一步的选取，这里的仿真就先略去。

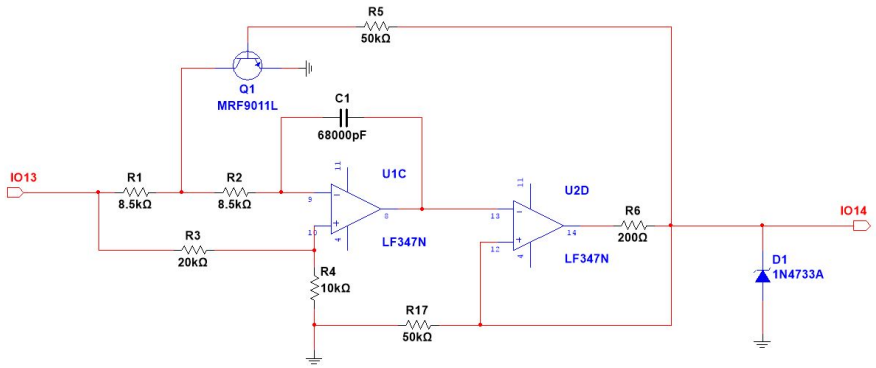

Fig. 2: 压频转换电路

1.3 FPGA 部分

这部分要求将模拟电路输出的方波信号计数，得到最后要求显示的频率，为数字电路部分，电路模块图如图 3所示。其中 FCore 模块是核心模块，负责在 FPGA 上的 50MHz 时钟下输出输入波形 in 的频率并在数码管中扫描显示。选通端由 74138 确定，数码管由 7448 驱动。

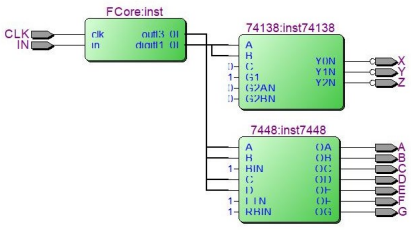

Fig. 3: 频率计数电路布置

FCore 模块的具体实现由 Verilog 代码给出。模块首先将输入时钟分频为 1Hz，并在 1Hz 时钟上升沿设置输出记号。在输入 in 的每个上升沿，模块进行十进制计数。如果此时发现设置了输出记号，则清除此记号之后将当前值输出显示清零。

FCore 也同时满足了一些诸如扫描数码管的功能，具体见下面的代码

```
01 module FCore (clk,in,out,digit);
02 input clk;
03 input in;
04 output [1:0] digit;
05 output [3:0] out;
06 reg sec;
07 reg a = 0;
08 reg b = 0;
09 reg [1:0] digit = 2'b00;
10 reg [3:0] out;
11 reg [3:0] num [2:0];
12 reg [3:0] _num [2:0];
13 reg [11:0] div = 12'b1;
14 reg [27:0] counter = 28'b0;
15 always @ (posedge clk)
16 begin
17 //TODO: div the 50MHz clock to 1Hz
18  if(counter == 28'd25000000)
19  begin
20   sec <= ~sec;
21   counter = 28'b1;
```

```
22  end
23  else
24   counter <= counter + 1'b1;
25
26  //TODO: div a sweeping signal
27  if(div == 28'b0)
28  begin
29   div <= div + 1'b1;
30   if(digit == 2'b10)
31    digit <= 2'b00;
32   else
33    digit <= digit + 1'b1;
34  end
35  else
36   div <= div + 1'b1;
37
38  if(div == 28'd10)
39   out <= num[digit];
40 end
41
42 always @ (posedge sec)
43 begin
44 //TODO: clear the output
45 a <= ~b;
46 end
47
48 always @ (posedge in)
49 begin
50 //TODO: count
51  if(a == b)
52  begin
53   if(_num[0] < 4'd9)
54    _num[0] <= _num[0] + 4'b0001;
55   else
56   begin
57    if(_num[1] < 4'd9)
```

```
58      begin
59        _num[1] <= _num[1] + 4'b0001;
60        _num[0] <= 4'b0000;
61      end
62      else
63      begin
64        _num[0] <= 4'b0000;
65        _num[1] <= 4'b0000;
66        _num[2] <= _num[2] + 4'b0001;
67      end
68    end
69  end
70  else
71  begin
72    b <= a;
73    num[0] <= _num[0];
74    num[1] <= _num[1];
75    num[2] <= _num[2];
76    _num[0] <= 4'b0001;
77    _num[1] <= 4'b0000;
78    _num[2] <= 4'b0000;
79  end
80 end
81 endmodule
```