学校代码：11517

河南工程学院

HENAN INSTITUTE OF ENGINEERING

# 数字系统设计

题　　目　基于 VHDL 的电压表设计

学生姓名　张通　陈朋朋　陈赞

专业班级　电子科学与技术 1042 班

学　　号　207　238　248

系　（部）　电气信息工程学院

指导教师(职称)　瓮 嘉 民

完成时间　2013 年 6 月 28 日

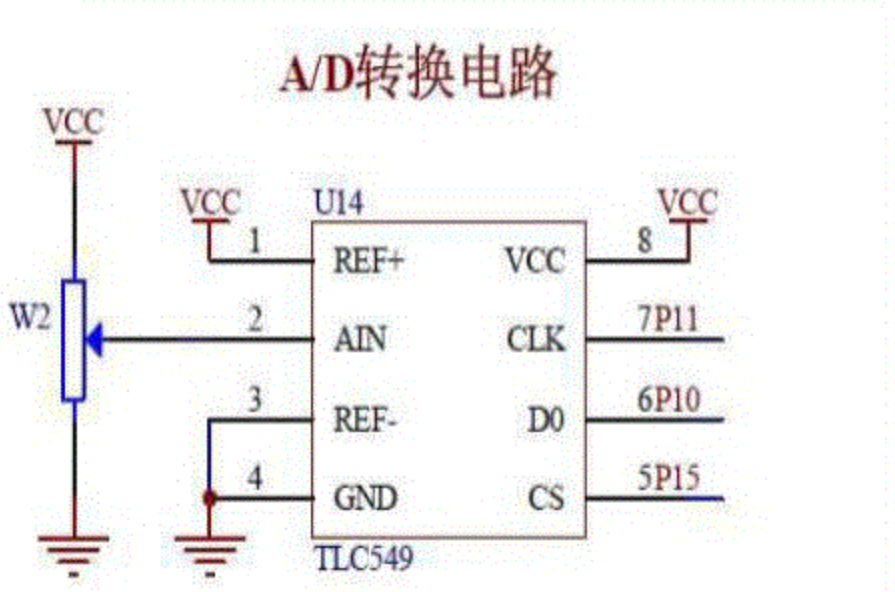
## 目　录

对于信号的采集和处理．多数是以单片机或CPU为控制核心．虽然编程简单．控制灵活，但缺点是速度慢、控制周期长。单片机的速度大地限制了A／D高速性能的利用。而FPGA的时钟频率可高达100 MHz以上。本设计以高集成度的芯片为核心．进行数据采集控制、数据时序控制等、具有开发周期短、灵活性强、

通用能力好、易于开发、扩展等优点．既降低了设计难度．又加快了产品的开发周期。

一；TLC549芯片介绍

1.1： TLC549是美国德州仪器公司生产的8位串行A/D转换器芯片，可与通用微处理器、控制器通过CLK、CS、DATA OUT三条口线进行串行接口。具有4MHz片内系统时钟和软、硬件控制电路，转换时间最长17μs， TLC549为40 000次/s。总失调误差最大为±0.5LSB，典型功耗值为6mW。采用差分参考电压高阻输入，抗干扰，可按比例量程校准转换范围，VREF-接地，VREF+—VREF-≥1V，可用于较小信号的采




样。

1.2： TLC549的内部框图和管脚名称上图所示。

它与控制模块通过SPI接口连接．即通过I／／O CLOCK、CS、DATA OUT 3个引脚串行与控制模块连接 TLC549有片内系统时钟．该时钟与I／O clock相互独立工作．无需特殊速度和相位匹配。

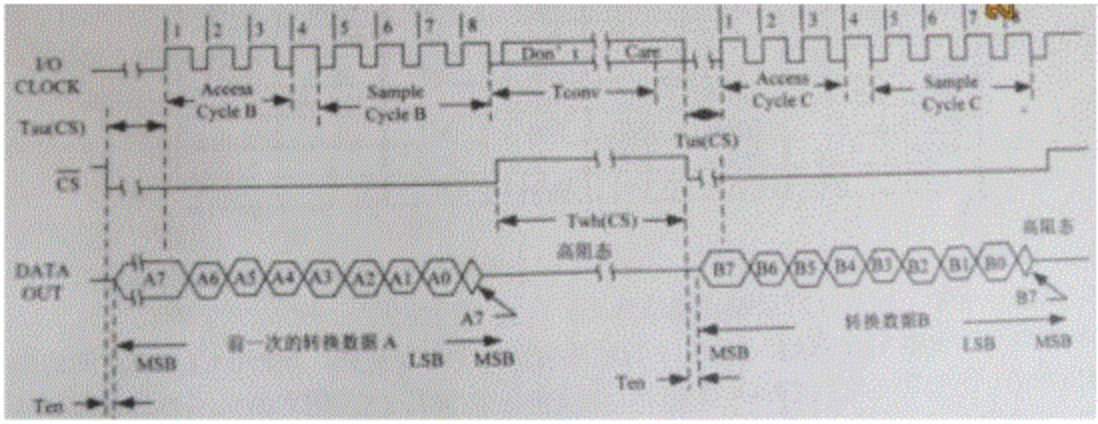
1.3： TLC549的极限参数如下：

- ●电源电压：6.5V；
- ●输入电压范围：0.3V～VCC＋0.3V；
- ●输出电压范围：0.3V～VCC＋0.3V；
- ●峰值输入电流(任一输入端)：±10mA；
- ●总峰值输入电流(所有输入端)：±30mA；
- ●工作温度： TLC549C：0℃～70℃
  TLC549I：－40℃～85℃
  TLC549M：－55℃～125℃

1.4： TLC549的工作时序

TLC549是一个八位的串行模数转换器，AD转换时间最大17微秒，I/O时钟可达1.1ＭＨＺ。如图４.３１所示为TLC549的访问时序，从图中可以看由当CS/拉低时，ADC前一次的转换数据（A）的最高位A7立即出现在数据线DATE ＯＵＴ上，之后的数据在时钟I/O ＣＬＯＣＫ的下降沿改变，可在I/O ＣＬＯＣ

K的上升沿读取数据。读完8位数据后，ＡＤＣ开始转换这一次采样的信号（Ｂ），以便下一次读取。转换时片选信号ＣＳ／要置高电平。设计操作时序是要注意Ｔｓｕ（ＣＳ）．Ｔｃｏｎｖ及I/O ＣＬＯＣＫ的频率几个参数。Ｔｓｕ（ＣＳ）为CS/拉低到I/O ＣＬＯＣＫ第一个时钟到来时间，至少要１.４微秒；Ｔｃｏｎｖ为ＡＤＣ的转换时钟，不超过１７微秒，I/O ＣＬＯＣＫ不能超过１.１ＭＨＺ。




TLC549访问时序图

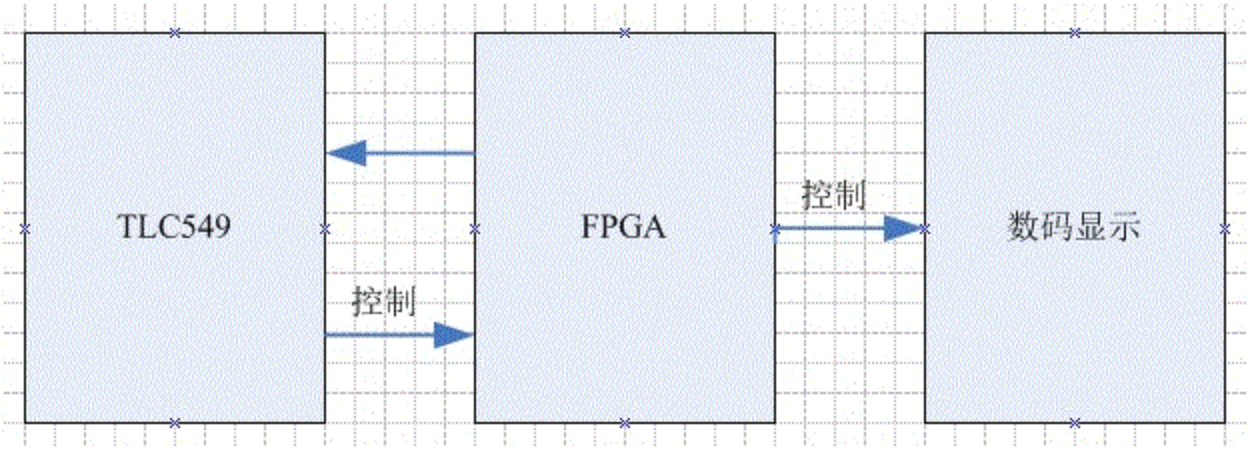
二　设计要求

设计一个数字电压表，利用8位A/D转换器，将连续的模拟电信号转换成离散的数字电信号，并加以显示，要求其量程为0-2.5V，
分辨率约为1mV，六数码管显示，其中四位显示转换后的数据，后面两位显示ADC转换译码后的数据，通过调整实验箱上的SW1旋钮来改变电压的大小。计算公式，V=(D/256)*Vref ，其中D为ＡＤＣ转换后读取的2位十六进制数，Vref是参考电压值，这里是2.5V。

三　整体设计

数字电压表的基本原理

数字电压表整体设计框图，如下图所示，数字电压表系统由A/D（TLC549）转换模块、FPGA控制模块、数码显示模块三部分构成。FPGA控制模块控制外部A/D转换器自动采样模拟信号，通过A/D芯片转换为数字信号，再由FPGA控制模块控制数码管动态扫描向外部数码管显示电路输出数据。



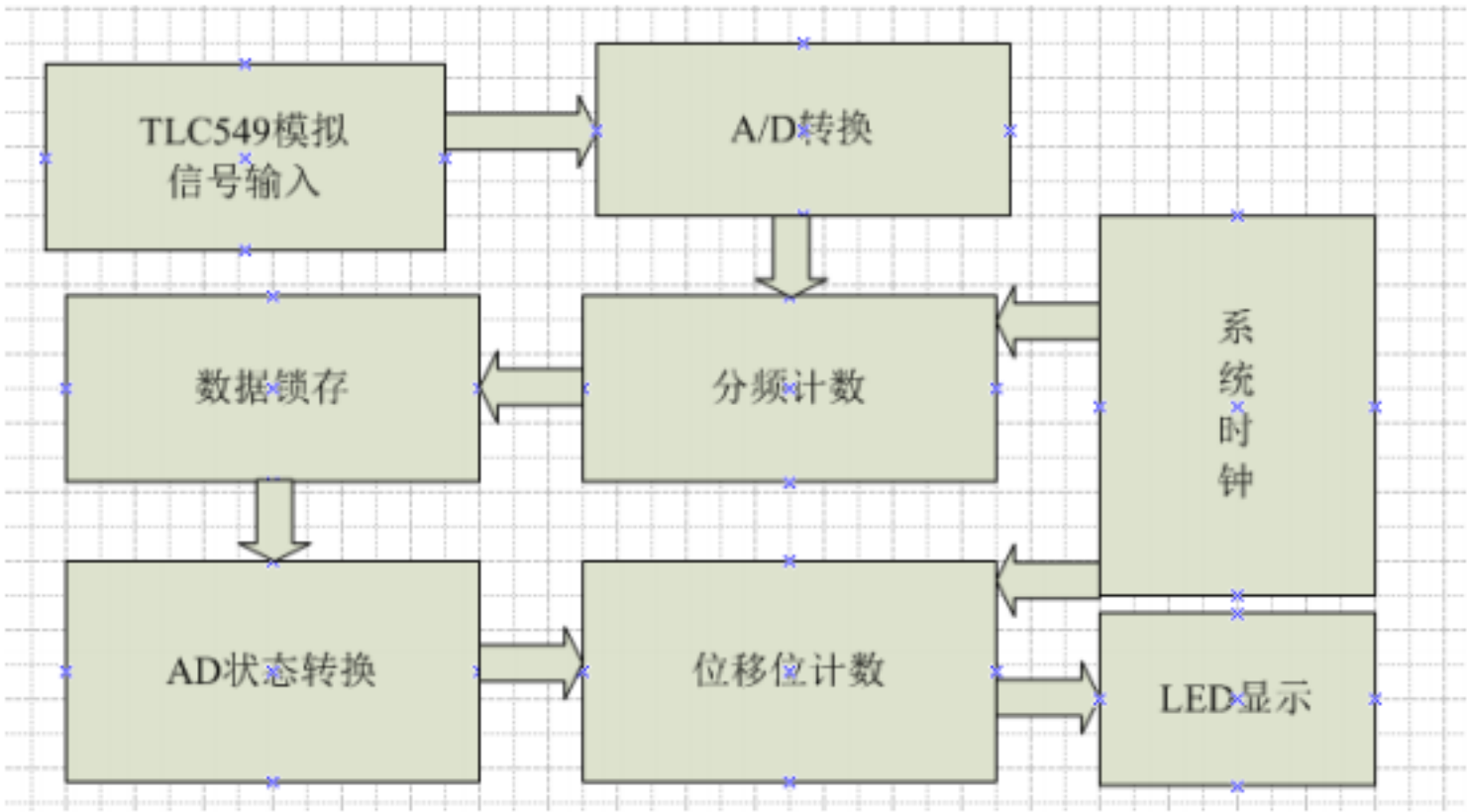

系统原理图

A/D转换器负责采集模拟电压，转换成8位数字信号送入FPGA转换控制模块，FPGA转换控制模块负责A/D转换的启动、地址锁存、输入通道选择、数据读取、

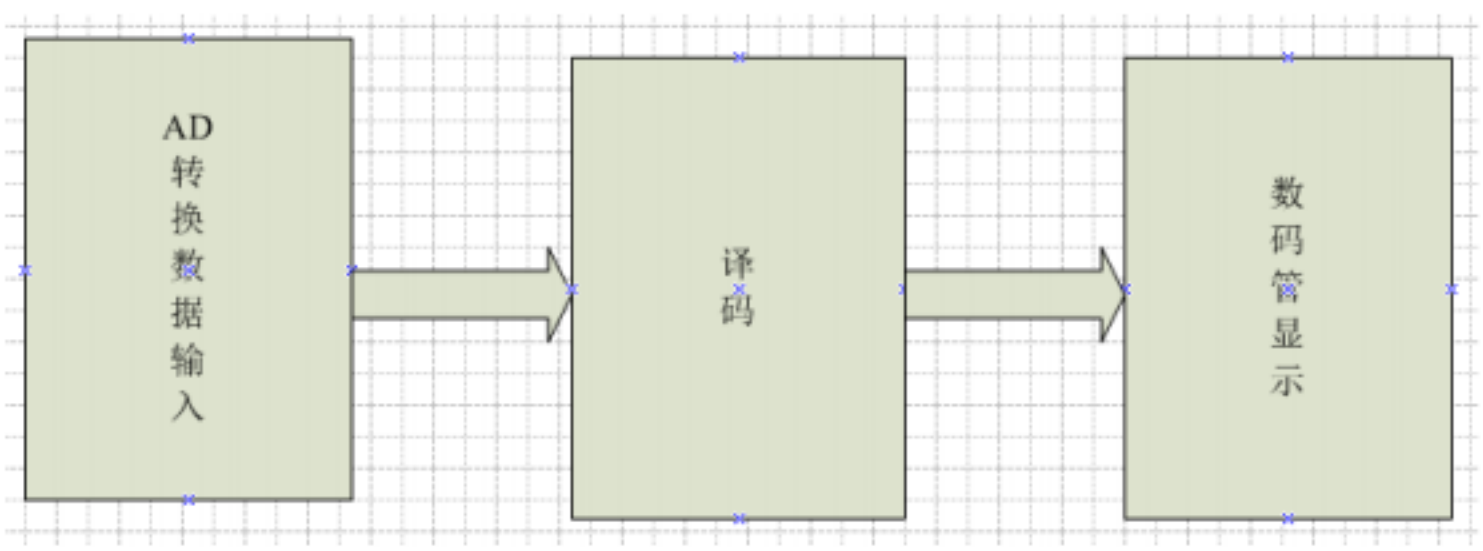
转换等工作，数码显示模块负责实现当前的电压值。

## 四　模块设计

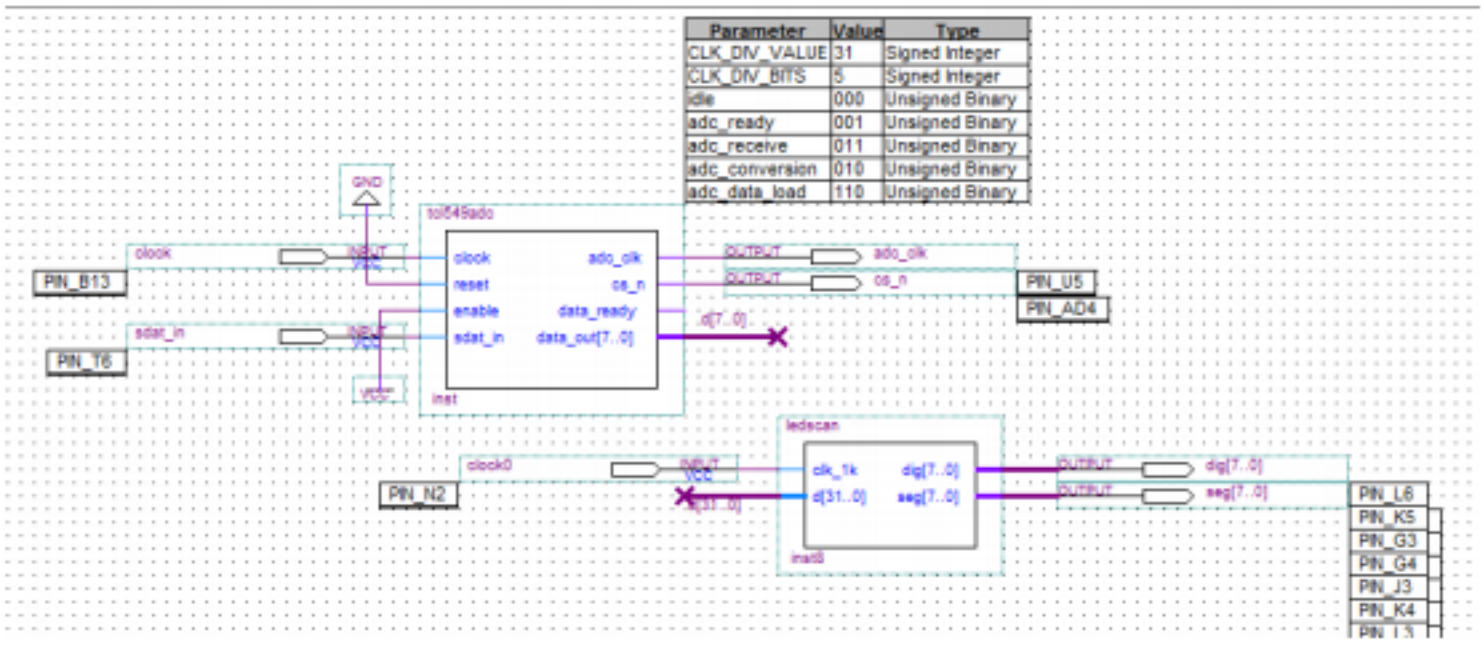
### 4.1　TLC549模块原理框图




### 4.2　数码管显示模块
如下图




### 4.3　原理图模块
如下图



## 五　小组成员感受

这次实训的两个星期，真的学到了很多的东西。我们组做的是《基于VHDL的电压表设计》，刚开始选题时只是觉得这个题目好理解，应该比较简单，可接下来开始准备的时候才发现当时 的想法太可笑了。

在实训的第一个星期，我们找了很多资料来编写修改源程序，刚开始想法很多，程序弄的也比较复杂，又花了好久时间去画方框图和波形的仿真图，可到第四周老师发实验箱时，才发现以前有好多的设计是错误的，因为实验箱上没有对应的部位，然后就悲剧了，只好再改程序。

仔细的看了老师给的资料《EDA修改稿》，然后再结合之前的资料，终于把最终的程序修改好了，然后进行了第一次试验，结果显示的是一对乱码，仔细检查后发现是数码管的显示部分出了问题，参考《EDA修改稿》上的动态数码管显示原理，然后修改自己的显示程序，再进行引脚的锁定，下载验证。乱码没有了，可又有其他问题了。

显示的电压数据跟实测的数据完全不吻合，后经过老师的指点和我们的共同努力，终于成功了。经过这次课程设计我真的学到了很多很多的知识，同时也感受到团队的强大，因为一个人着急的时候，思绪是最乱的，然后根本没法去思考问题，这时队友的作用就显现出来了，他可以提醒你和提出一些非常重要的看法，对解决问题有着至关重要的作用。

团队就是强大。回想这次课程设计的经历，发现老师给了我们很多很关键的援助，或是暗示，又或是直接指出问题所在，总是通过不同的方法帮组我们解决问题，给我们最及时的帮组，谢谢您！！

---------张通

转眼间，实训就要接近尾声了， 电压表这个课程设计，总的来说，我认为还是有点难度的，尤其是程序这一块，不怎么容易搞定，但是经过我们组成员的坚持不懈，我们终于还是搞定了，虽然做的不是很好，但是还是勉强能够有一点成绩的，在这个过程当中，老师也对我们做的东西提出了宝贵的一件，做了很全面的指导，在他的指导下，我们有了很大的进步。

这次实训对我们是个很大的挑战，时间仅仅有 2 周，实训一结束就是期末考试，所以我们只能一边完成实训，一边还要投入紧张的复习中去，这无疑给我们增加了沉重的负担。虽然面临着巨大的挑战，但是我们每个人都全身心投入，中间也出现了很多问题，在同学们的帮助下以及老师的指导下，最终我们完成了这次的实训，虽然累，但却很高兴。

我们的实践能力还很欠缺，尤其表现在理论与实际联系上。主要原因可能的是思考的少以至于眼界不是十分开阔，进一步导致在实践的过程中感想却不敢做。总的来说，我对这次的实践收获很大。

电子电工实习，是以学生自己动手掌握一定的操作技能，并亲手设计、制作、

组装与调试为特色。它将基本技能训练，几本工艺知识和创新有机结合，培养我们的实践能力和创新精神，作为信息时代的大学生，作为国家重点培养的高技能人才，仅仅会操作鼠标是不够的，基本的动手能力和思考能力是必备的。

通过两周的学习，我学到了很多知识，不仅培养了我的动手能力，而且培养了我的思维能力。这些知识不仅在课堂上有效，对以后电子工艺课的学习有很大的指导意义，在日常生活以及以后的工作中都有着很大的意义。

最后，我想说，在这个实训的过程当中，我感受最深的就是　我们组员团结的时刻，有我们都不会的时候，也有我们无从下手的时候，但是我们都一一过来了，更重要的是老师对我们的指导，所以我很感谢老师，有了您的帮助，我们会做的更好。

姓名：陈赞

短暂的十几天实训已经过去，对于我来说这段时间我学会了很多东西。不仅让我更深层次的对课本理论知识深入的了解，还提高了实际动手能力。本次实训是在实验平台上实现电压表的功能，需要熟练掌握 quartus 11 软件的操作和 vhdl 语言的基本设计思路和方法。实验过程中我们出现了这样或那样的问题，经过老师的批评和指出问题关键，我们又进行了一次次修改和尝试，最终实验取得成功。首先得谢谢翁老师的指导和同学的帮助，总得来说这次实训我收获很大。

---陈朋朋

附录1：程序清单

```
module tlc549adc(clock,reset,enable,sdat_in,adc_clk,cs_n,
data_ready,data_out);
//I/O
input  clock;//系统时钟
input  reset;//复位，高电平有效
input  enable;//转换使能
input  sdat_in;//TLC549串行数据输入
output adc_clk;//TLC549 I/O时钟
output cs_n;//TLC549 片选控制
output data_ready;//指示有新的数据输出
output[7:0]   data_out;//AD转换数据输出
reg    adc_clk_r;
reg    cs_n_r;
reg    data_ready_r;
reg[7:0]   data_out_r;
reg    sdat_in_r;//数据输出锁存
//内部寄存器
reg[7:0]   q;//移位寄存器，用于接收或发送数据
```

```
reg[2:0]   adc_state;//状态机ADC
reg[2:0]   adc_next_state;
reg[5:0]   bit_count;//移位计数器
reg    bit_count_rst;//ADC时钟计数器全能控制
reg    div_clk;
reg[CLK_DIV_BITS-1:0]    clk_count;//时钟分频计数器
reg    buf1,buf2;
//内部信号
wire   ready_done;//cs_n拉低(大于1.4us)后的标志
wire   rec_done;//数据读取完毕的标志
wire   conv_done;//数据转换完毕的标志
//50mhz
parameter  CLK_DIV_VALUE=1024;
parameter  CLK_DIV_BITS=5;
//状态机M1状态参数表
parameter  idle=3'b000,
           adc_ready=3'b001,
           adc_receive=3'b011,
           adc_conversion=3'b010,
           adc_data_load=3'b110;
//I/O寄存器输出
assign adc_clk=adc_clk_r;
assign cs_n=cs_n_r;
assign data_out=data_out_r;
assign data_ready=data_ready_r;
//同步输入数据信号
always@(posedge clock)
begin
    sdat_in_r<=sdat_in;
end
//时钟分频计数器
always@(posedge clock)
begin
    if(reset==1'b1)
        clk_count<=5'd0;
    else
    begin
        if(clk_count<CLK_DIV_VALUE)
        begin
            clk_count<=clk_count+1'b1;
            div_clk<=1'b0;
        end
        else
        begin
```

```
        clk_count<=5'd0;
        div_clk<=1'b1;
        end
    end
end
//状态机ADC
always@(posedge clock)
begin
    if(reset==1'b1)
        adc_state<=idle;
    else
        adc_state<=adc_next_state;
end
//ADC状态机转换逻辑
always@(adc_state or ready_done or rec_done or conv_done or enable)
begin
    cs_n_r<=1'b0;
    bit_count_rst<=1'b0;
    data_ready_r<=1'b0;
    case(adc_state)
        idle://初始状态
            begin
                cs_n_r<=1'b1;
                bit_count_rst<=1'b1;//复位移位计数器
                if(enable==1'b1)
                    adc_next_state<=adc_ready;
                else
                        adc_next_state<=idle;
            end
        adc_ready://准备接收
            begin
                if(ready_done==1'b1)
                    adc_next_state<=adc_receive;
                else
                    adc_next_state<=adc_ready;
            end
        adc_receive://接收数据
            begin
                if(rec_done==1'b1)
                    adc_next_state<=adc_conversion;
                else
                    adc_next_state<=adc_receive;
            end
        adc_conversion://转换前的采样的数据
```

```
            begin
                cs_n_r<=1'b1;
                if(conv_done==1'b1)
                    adc_next_state<=adc_data_load;
                else
                    adc_next_state<=adc_conversion;
            end
        adc_data_load:
            begin
                data_ready_r <= 1'b1;          //数据输出标志
                adc_next_state <= idle;
            end
        default : adc_next_state <= idle;
    endcase
end
//*****************************************
//位移位计数器
always @(posedge clock)
begin
    if (reset == 1'b1)
        bit_count <= 6'd0;
    else if (bit_count_rst == 1'b1)
        bit_count <= 6'd0;
    else if (div_clk == 1'b1)
        bit_count <= bit_count + 1'b1;
end
assign ready_done = (bit_count == 6'd4);    //准备读取数据
assign rec_done = (bit_count == 6'd19);      //接收数据完毕
assign conv_done = (bit_count == 6'd63);    //ADC 转换完毕
//在接收位计数器4—20间8个adc clk
always @(bit_count)
begin
  if ((bit_count < 6'd20) && (bit_count >= 6'd4))
        adc_clk_r <= ~bit_count[0];
  else
        adc_clk_r <= 1'b0;
end
always @(posedge clock)
begin
  buf1 <= adc_clk_r;
  buf2 <= buf1;
end
//读取数据
always @(posedge clock)
```

```
begin
  if(buf1 && ~buf2)                //ADC 时钟上升沿
    q <= {q[6:0],sdat_in_r};
  else if(data_ready_r == 1'b1)        //输出读取的数据
    data_out_r <= q;
end
endmodule
```

程序2：**********

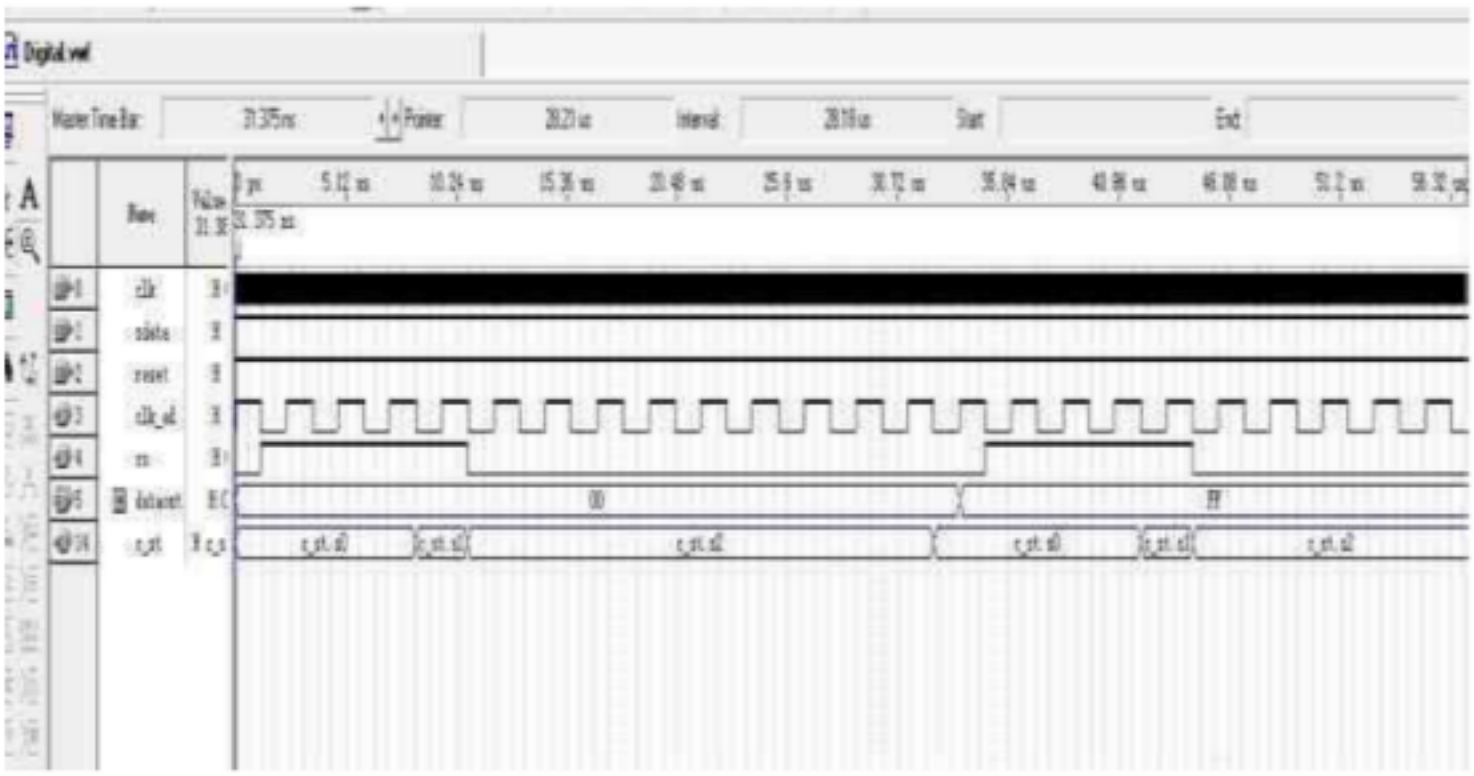
```
    module ledscan(clk_1k,d,dig,seg);                //模块名scan_led
input clk_1k;                        //输入时钟
input [31:0] d;                      //输入要显示的数据
output [7:0] dig;                    //数码管选择输出引脚
output [7:0] seg;                    //数码管段输出引脚
wire[15:0] dd;
reg [7:0] seg_r;                     //定义数码管输出寄存器
wire[15:0] num;
reg [7:0] dig_r;                     //定义数码管选择输出寄存器
reg [3:0] disp_dat;                  //定义显示数据寄存器
reg [2:0] count;                     //定义计数寄存器
assign dig=dig_r;                    //输出数码管选择
assign seg=seg_r;                    //输出数码管译码结果
assign dd=d*2500/256;
assign num[15:12]=dd/1000;
assign num[11:8]=dd%1000/100;
assign num[7:4]=dd%100/10;
assign num[3:0]=dd%10;
always @(posedge clk_1k)             //定义上升沿触发进程
begin
    count <=count+1'b1;
    if(count==3'd5)
    count<=3'd0;
end
always @(posedge clk_1k)
begin
 case(count)                                                    //选
择扫描显示数据
    3'd0:disp_dat=num[15:12];          //第一个数码管
     3'd1:disp_dat=num[11:8];          //第二个数码管
     3'd2:disp_dat=num[7:4];           //第三个数码管
     3'd3:disp_dat=num[3:0];           //第四个数码管
    /*3'd4:disp_dat=d[15:12];         //第五个数码管
     3'd3:disp_dat=d[11:8];           //第六个数码管 */
     3'd4:disp_dat=d[7:4];            //第七个数码管
     3'd5:disp_dat=d[3:0];            //第八个数码管
```

```
  endcase
  case(count)                              //选择数码管显示位
        3'd0:dig_r=8'b01111111;            //选择第一个数码管显示
         3'd1:dig_r=8'b10111111;          //选择第二个数码管显示
         3'd2:dig_r=8'b11011111;          //选择第三个数码管显示
         3'd3:dig_r=8'b11101111;          //选择第四个数码管显示
      /*  3'd4:dig_r=8'b11110111;          //选择第五个数码管显示
         3'd3:dig_r=8'b11111011;        //选择第六个数码管显示 */
         3'd4:dig_r=8'b11111101;          //选择第七个数码管显示
         3'd5:dig_r=8'b11111110;          //选择第八个数码管显示
     endcase
end
always @(disp_dat)
begin
    case(disp_dat)                                 //七段译码
        4'h0:seg_r=8'hc0;                           //显示0
        4'h1:seg_r=8'hf9;                          //显示1
        4'h2:seg_r=8'ha4;                          //显示2
        4'h3:seg_r=8'hb0;                          //显示3
        4'h4:seg_r=8'h99;                           //显示4
        4'h5:seg_r=8'h92;                           //显示5
        4'h6:seg_r=8'h82;                           //显示6
        4'h7:seg_r=8'hf8;                           //显示7
        4'h8:seg_r=8'h80;                           //显示8
        4'h9:seg_r=8'h90;                           //显示9
        4'ha:seg_r=8'h88;                           //显示a
        4'hb:seg_r=8'h83;                           //显示b
        4'hc:seg_r=8'hc6;                           //显示c
        4'hd:seg_r=8'ha1;                           //显示d
        4'he:seg_r=8'h86;                           //显示e
        4'hf:seg_r=8'h8e;                           //显示f
    endcase
end
endmodule
```

附录2：引脚锁定图

| Node Name | Direction | Location | I/O Bank | VREF Group |
|---|---|---|---|---|
| adc_clk | Output | PIN_U5 | 1 | B1_N1 |
| clock | Input | PIN_B13 | 4 | B4_N1 |
| clock0 | Input | PIN_N2 | 2 | B2_N1 |
| cs_n | Output | PIN_AD4 | 8 | B8_N1 |
| dig[7] | Output | PIN_M4 | 2 | B2_N1 |
| dig[6] | Output | PIN_L3 | 2 | B2_N1 |
| dig[5] | Output | PIN_K4 | 2 | B2_N1 |
| dig[4] | Output | PIN_J3 | 2 | B2_N1 |
| dig[3] | Output | PIN_G4 | 2 | B2_N0 |
| dig[2] | Output | PIN_G3 | 2 | B2_N0 |
| dig[1] | Output | PIN_K5 | 2 | B2_N0 |
| dig[0] | Output | PIN_L6 | 2 | B2_N1 |
| sdat_in | Input | PIN_T6 | 1 | B1_N0 |
| seg[7] | Output | PIN_L9 | 2 | B2_N1 |
| seg[6] | Output | PIN_L10 | 2 | B2_N1 |
| seg[5] | Output | PIN_N9 | 2 | B2_N1 |
| seg[4] | Output | PIN_U10 | 1 | B1_N0 |
| seg[3] | Output | PIN_J6 | 2 | B2_N0 |
| seg[2] | Output | PIN_K6 | 2 | B2_N0 |
| seg[1] | Output | PIN_M3 | 2 | B2_N1 |
| seg[0] | Output | PIN_J8 | 2 | B2_N0 |
| <<new node>> | | | | |

附录3：波形仿真图

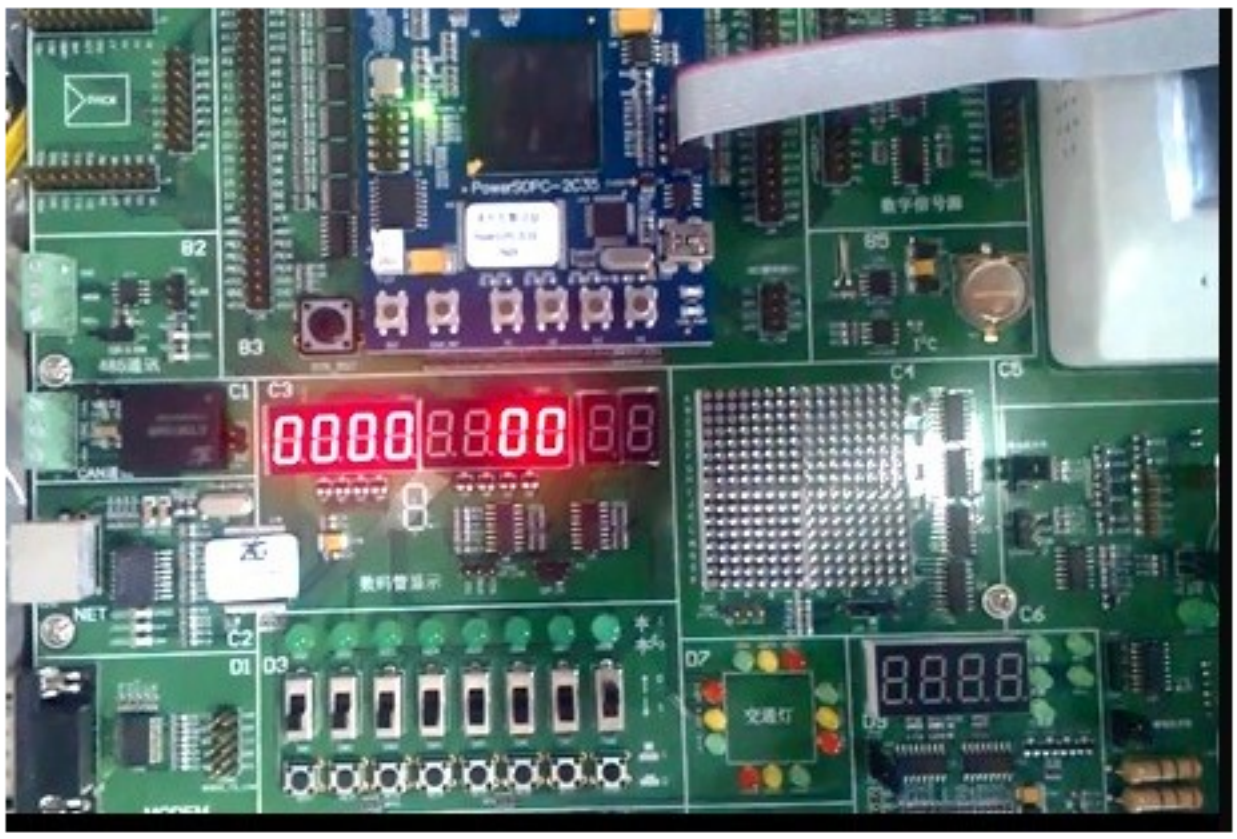


备注：因两个模块之间是用原理图连接的，故仿真结果与下载验证结果有区别。

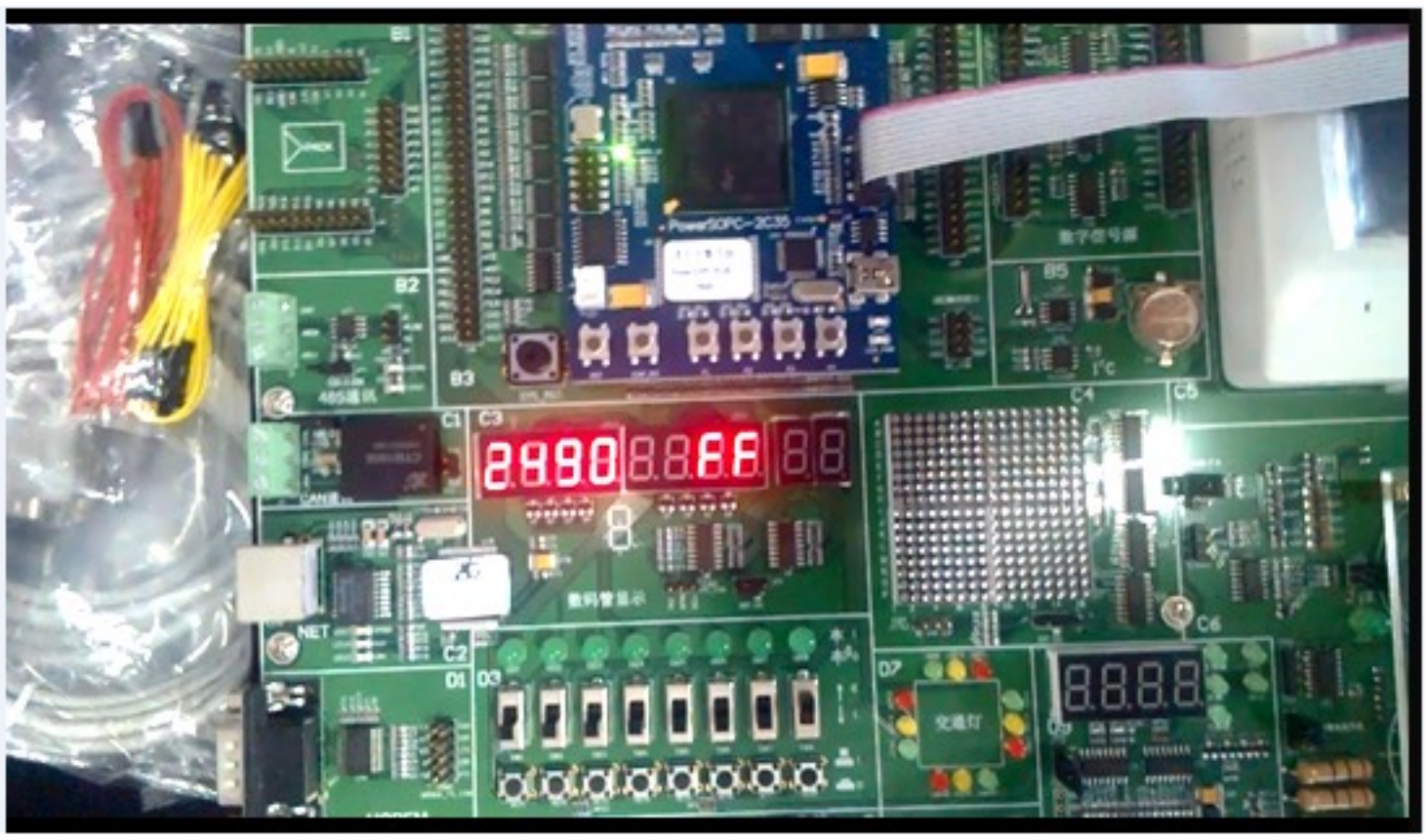
附录4：下载验证照片

1，　　输出电压表示　左边四位表示转换好的电压值 单位是mV，右边两位是未译码由TLC549直接输出的16进制数。

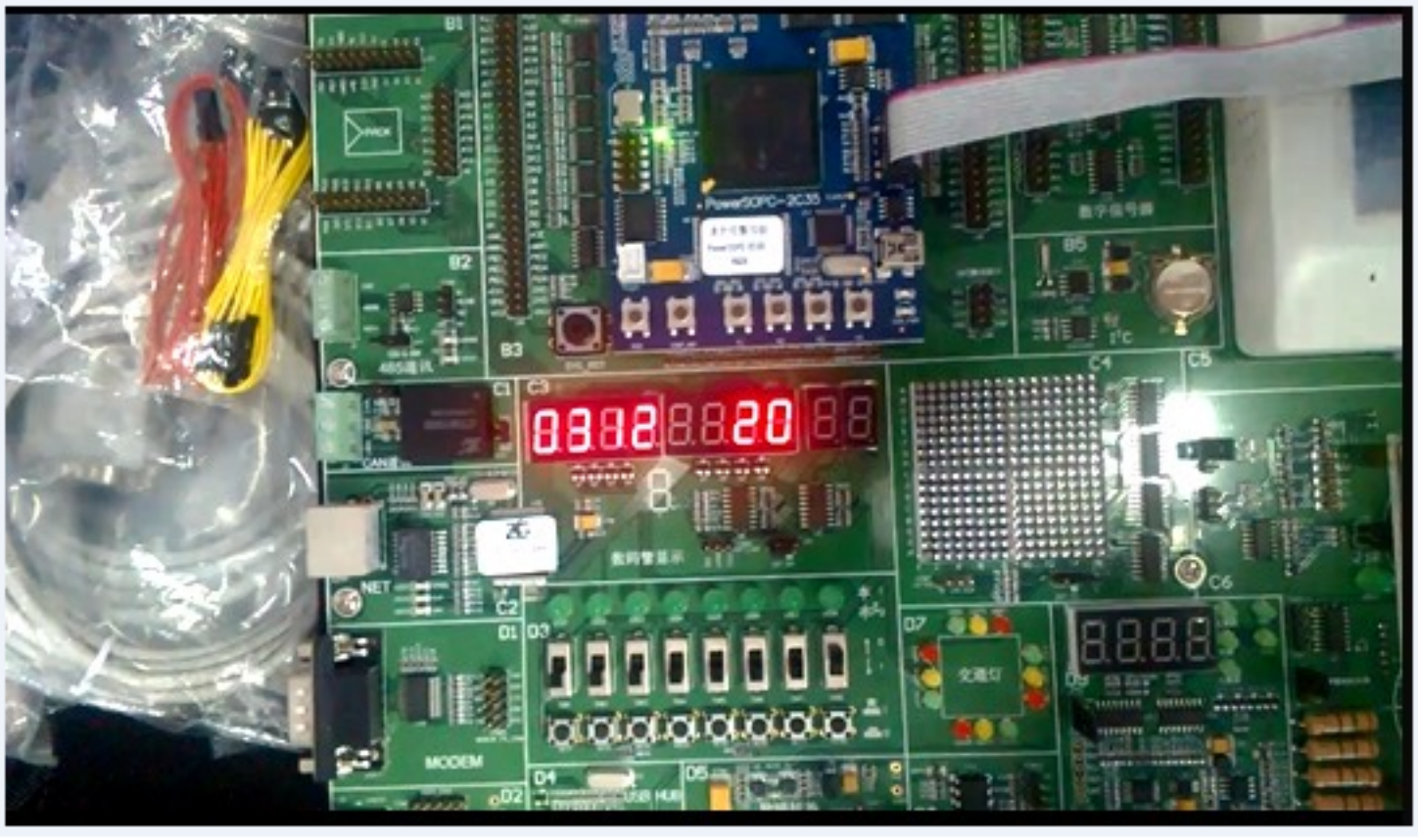
这里显示的是0000mV　00




2　　最大电压值　2490mV　FF



3　显示电压值 0312mV　十六进制数为 20




附录5：参考资料

1，EDA技术实用教程-Verilog HDL版 （第四版） 潘松 黄继业 潘明 编著
科学出版社

2，EDA修改稿。