上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

《数字加法器与计时器的制作与调试报告》

学生姓名: 杨超琪

学生学号: 515030910067

专　　业: 信息安全

完成时间: 2017 / 4 / 13

学院(系): 电子信息与电气工程学院

目录

数字加法器与计时器

1.功能介绍

1.1 实验基本要求

用拨码开关输入两个四位二进制数，按下运算按钮后，以 led 的方式显示运算的结果。

1.2 后期扩展要求

（1）利用 Basys2 板上的微动开关提供清零功能。
（2）利用 Basys2 板上的后两位数码管显示加法结果的十六进制表示（直接防止溢出）
（3）利用 Basys2 板上四位数码管与八位 LED 管实现同步计时器。
（4）利用 Basys2 板上微动开关实现计时器与加法器功能间的切换。
（5）利用 Basys2 板上微动开关提供计时器的开始计时、暂停、清零功能。

2.设计方案与设计过程

2.1 设计方案

我们的目标是实现 2 个四位二进制数的加法与一个四位十六进制数的计时器，将要用到 Basys2 板的所有管脚。由于功能的复杂性与开关显示器的有限性，在这两个分立功能的实现上，许多构件将会被重用，给我们的程序逻辑设计带来很大的挑战，在这里，我们将实现显示器与开关的多功能性与重用性，将诸多功能集成于一块 Basys2 板上。

2.1 设计过程

2.1.1 管脚定义

```
1
2   NET "LED[0]" LOC = M5;          //LED灯管脚定义
3   NET "LED[1]" LOC = M11;
4   NET "LED[2]" LOC = P7;
5   NET "LED[3]" LOC = P6;
6   NET "LED[4]" LOC = N5;
7   NET "LED[5]" LOC = N4;
8   NET "LED[6]" LOC = P4;
9   NET "LED[7]" LOC = G1;
10
11  NET "addition_1[0]" LOC = G3;  //两个四位二进制加数管脚定义
12  NET "addition_1[1]" LOC = F3;
13  NET "addition_1[2]" LOC = E2;
14  NET "addition_1[3]" LOC = N3;
15  NET "addition_2[0]" LOC = P11;
16  NET "addition_2[1]" LOC = L3;
17  NET "addition_2[2]" LOC = K3;
18  NET "addition_2[3]" LOC = B4;
19
```

```
19
20  NET "a_to_g[0]" LOC = M12;      // 数码管管脚定义
21  NET "a_to_g[1]" LOC = L13;
22  NET "a_to_g[2]" LOC = P12;
23  NET "a_to_g[3]" LOC = N11;
24  NET "a_to_g[4]" LOC = N14;
25  NET "a_to_g[6]" LOC = L14;
26  NET "a_to_g[5]" LOC = H12;
27
28  NET "an[0]" LOC = K14 ;         //数码管使能引脚定义
29  NET "an[1]" LOC = M13 ;
30  NET "an[2]" LOC = J12 ;
31  NET "an[3]" LOC = F12 ;
32
33  NET "clk" LOC = B8 ;            //时钟与微动开关管脚定义
34  NET "add" LOC = G12;
35  NET "clr" LOC = C11;
36  NET "start" LOC = A7 ;
37  NET "shift" LOC = M4 ;
38
39  NET "start" CLOCK_DEDICATED_ROUTE = FALSE;              //微动开关与时钟的区分
40  NET "shift" CLOCK_DEDICATED_ROUTE = FALSE;
41  NET "add" CLOCK_DEDICATED_ROUTE = FALSE;
42  NET "clr" CLOCK_DEDICATED_ROUTE = FALSE;
43
```

管教定义的 39~42 这几行，大概是在说，当时钟与微动开关同时发送信号的时候，时钟会让位于微动开关，如果没有这个定义，将会出现 Unrouted 错误，我会在后面具体讨论。

在此，我们用到了 Basys2 板上的所有器件。

2.1.2 程序参数定义

```
3
4   input wire [3:0] addition_1 ,       //第一个四位二进制加数
5   input wire [3:0] addition_2 ,       //第二个四位二进制加数
6   input wire add ,                    //加法微动开关
7   input wire clr ,                    //清除微动开关
8   input wire clk ,                    //时钟引脚
9   output reg [7:0] LED,               //LED管引脚定义
10  output reg [6:0]a_to_g ,            //数码管引脚定义
11  output reg[3:0] an,                 //数码管使能端定义
12  input wire start ,                  //计时器暂停/开始微动开关
13  input wire shift ) ;                //功能切换微动开关
14
15  reg cout ;                          //第二个和位
16  reg[3:0] num ;                      //第一个和位
17  reg [36:0] clk_cnt ;                //第一个时钟
18  reg [36:0] clk_cnt2 ;               //第二个时钟
19  reg rw ;                            //表征计时器start/pause的参量
20  reg shift_bt ;                      //表征功能切换的参量
21  reg [3:0]num2 ;                     //存储计时器数码管的每一个十六进制位
22  reg [3:0]stop ;                     //显示计时器暂停时的STOP
23  reg [7:0] LED_show ;                //LED管的八位二进制存储
24
```

2.1.3 功能切换顶层逻辑

$$initial:\begin{cases} rw=0 \\ shift\_bt=0 \end{cases}$$

$$shift\_bt\begin{cases} 0:\text{加法功能} \\ 1:\text{计数器功能}\rightarrow \text{rw}\begin{cases} 0:\text{开始计数} \\ 1:\text{暂停计数} \end{cases} \end{cases}$$

$$rw=0 \xleftrightarrow[\substack{start/pause\text{微动开关}\\ \text{在}shift\_bt=1\text{时才能运作}}]{} rw=1$$

$$shift\_bt=0 \xleftrightarrow[shift\text{微动开关}]{} shift\_bt=1$$

2.1.4 数码管使能引脚设计

```
always @(*)
case(shift_bt)
0:
    begin
        an[3]=~clk_cnt[15] ;
        an[2]=clk_cnt[15] ;
        an[0]=1 ;
        an[1]=1 ;
    end
1:
    case(rw)
    0:
        case(clk_cnt[16])
        1:

            case(clk_cnt[15])
            1:
                begin
                    an[0]=0 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            0:
                begin
                    an[0]=1 ;
                    an[1]=0 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            endcase

        0:
            case(clk_cnt[15])
            1:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=0 ;
                    an[3]=1 ;
                end
```

```
            0:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=0 ;
                end
            endcase
        endcase
    1:
        case(clk_cnt2[16])
        1:
            case(clk_cnt2[15])
            1:
                begin
                    an[0]=0 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            0:
                begin
                    an[0]=1 ;
                    an[1]=0 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            endcase
        0:
            case(clk_cnt2[15])
            1:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=0 ;
                    an[3]=1 ;
                end
            0:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=0 ;
                end
            endcase
        endcase
    endcase
endcase
```

在这里首先对 shift_bt 进行 case 分类：

0（加法类）：那就关闭前两个使能引脚，利用时钟，在数码管上分别错位显示加法结果的高十六进制位和低十六进制位。

1（计时器类）：那就继续对 rw 进行 case 分类：

0（开始计时类）：利用时钟的两位二进制，在四个数码管上分别错位显示计时结果的从高到低 4 个十六进制位。

1（暂停计时类）：此时时钟暂停，若继续使用第一个时钟，四个数码管有且仅有一个会显示结果，于是利用第二个独立时钟的两位二进制 ，在四个数码管上分别显示：STOP。

2.1.5 加法器微动开关设计

```
135
136  always@(posedge add or posedge clr or  posedge clk )
137  if(add)
138          begin
139                  {cout,LED_show[3:0]}= addition_1 +addition_2 ;
140                  {cout,num}= addition_1 +addition_2 ;
141                  LED_show[4]=cout ;
142          end
143  else
144  if(clr)
145          begin
146                  clk_cnt=0 ;
147                  LED_show=8'b00000000 ;
148                  cout=0 ;
149                  num=4'b0000 ;
150          end
151  else
152          case(rw)
153          0:
154              begin
155                  clk_cnt = clk_cnt+1 ;
156                  if(clk_cnt[36:33]>15)
157                      clk_cnt=0 ;
158
159                  clk_cnt2 = clk_cnt2+1 ;
160                  if(clk_cnt2[36:33]>15)
161                      clk_cnt2=0 ;
162              end
163          1:
164          begin
165              clk_cnt2 = clk_cnt2+1 ;
166              if(clk_cnt2[36:33]>15)
167                  clk_cnt2=0 ;
168          end
169          endcase
170
```

add（加法开关）:cout 与 LED_show[3:0]分别保存加法进位与后四个二进制位，形成一高一低两个十六进制位，num 与 LED_show[3:0]值一致，然后把 cout 值赋给 LED_show[4]。cout 与 num 将作为一高一低两个十六进制位在数码管上显示，LED_show[4:0]将作为五位二进制数在 LED 管上显示。实现 LED 与数码管的同步显示。

clr（清零开关）:当按下 clr 时，时钟 1 与所有数字信息都将清零，这个开关还可以复用在计时器的计时清零上。

clk（时钟）:这里定义了两个始终，时钟 1 用于计时，会被 clr 清零，也会被计时暂停；时钟 2 用于显示计时暂停后的 STOP 提示，不会被 clr 清零，一直在增加，这里用了 case 逻辑实现了两个始终的功能区别。

2.1.6 计时器微动开关设计

```
170
171   always @(posedge start or posedge shift)
172   if(shift)
173       begin
174           if(shift_bt==0)
175           begin
176               rw=1 ;
177               shift_bt=1 ;
178           end
179           else
180           begin
181               shift_bt=0 ;
182               rw=0;
183           end
184       end
185   else if(shift_bt==1)
186   begin
187       if(rw==1)
188           rw=0 ;
189       else
190           rw=1 ;
191   end
```

start(计时暂停/开始开关）：这个开关可以切换计时器的计时与终止，同时这里考虑到了只有在当前是计时功能时，这个开关才能运作。

shift（功能切换开关）：可以切换加法功能与计时功能，同时考虑到了进入加法模式时，时钟 1 切换回运行模式，进入计时模式时，时钟起初保持暂停。

2.1.7 LED 灯显示设计

```
192
193   always @(*)
194   case(shift_bt)
195   0:
196       LED=LED_show ;
197   1:
198       LED=clk_cnt[28:21] ;
199   endcase
200
```

这里 LED 灯对不同功能的显示由功能切换键 shift 控制，分别显示加法器的和结果，与计时器的计时结果。

2.1.8 数码管四位十六进制设计

```
115   always @(*)
116   case(clk_cnt[16])
117       1:
118
119           case(clk_cnt[15])
120           1:
121               num2=clk_cnt[36:33] ;
122           0:
123               num2=clk_cnt[32:29] ;
124           endcase
125
126       0:
127           case(clk_cnt[15])
128           1:
129               num2=clk_cnt[28:25] ;
130           0:
131               num2=clk_cnt[24:21] ;
132           endcase
133   endcase
```

按照时钟信号，赋予数码管数字不同时间不同的十六进制值。

2.1.9 数码管显示设计

```
always @(*)
case(shift_bt)
0:
    case(clk_cnt[15])
    1:
        case(num)
        0:a_to_g=7'b0000001;
        1:a_to_g=7'b1001111;
        2:a_to_g=7'b0010010;
        3:a_to_g=7'b0000110;
        4:a_to_g=7'b1001100;
        5:a_to_g=7'b0100100;
        6:a_to_g=7'b0100000;
        7:a_to_g=7'b0001111;
        8:a_to_g=7'b0000000;
        9:a_to_g=7'b0000100;
        'hA: a_to_g=7'b0001000;
        'hB: a_to_g=7'b1100000;
        'hC: a_to_g=7'b0110001;
        'hD: a_to_g=7'b1000010;
        'hE: a_to_g=7'b0110000;
        'hF: a_to_g=7'b0111000;
        default: a_to_g=7'b0000001;
        endcase
    0:
        case(cout)
        0:a_to_g=7'b0000001;
        1:a_to_g=7'b1001111;
        2:a_to_g=7'b0010010;
        3:a_to_g=7'b0000110;
        4:a_to_g=7'b1001100;
        5:a_to_g=7'b0100100;
        6:a_to_g=7'b0100000;
        7:a_to_g=7'b0001111;
        8:a_to_g=7'b0000000;
        9:a_to_g=7'b0000100;
        'hA: a_to_g=7'b0001000;
        'hB: a_to_g=7'b1100000;
        'hC: a_to_g=7'b0110001;
        'hD: a_to_g=7'b1000010;
        'hE: a_to_g=7'b0110000;
        'hF: a_to_g=7'b0111000;
        endcase
    endcase
```

```
    endcase
1:
    case(rw)
    0:
        case(num2)
            0:a_to_g=7'b0000001;
            1:a_to_g=7'b1001111;
            2:a_to_g=7'b0010010;
            3:a_to_g=7'b0000110;
            4:a_to_g=7'b1001100;
            5:a_to_g=7'b0100100;
            6:a_to_g=7'b0100000;
            7:a_to_g=7'b0001111;
            8:a_to_g=7'b0000000;
            9:a_to_g=7'b0000100;
            'hA: a_to_g=7'b0001000;
            'hB: a_to_g=7'b1100000;
            'hC: a_to_g=7'b0110001;
            'hD: a_to_g=7'b1000010;
            'hE: a_to_g=7'b0110000;
            'hF: a_to_g=7'b0111000;
        endcase
    1:
    case(clk_cnt2[16])
        1:
            case(clk_cnt2[15])
            1:
                a_to_g=7'b0100100;
            0:
                a_to_g=7'b1110000;
            endcase
        0:
            case(clk_cnt2[15])
            1:
                a_to_g=7'b0000001;
            0:
                a_to_g=7'b0011000;
            endcase
    endcase
    endcase
endcase
```

这里数码管的显示比较复杂，是一种复杂的功能复用。既要与（2.1.4）处的使能引脚精确配合使用，又要配合（2.1.8）十六进制位值，并符合（2.1.3）顶层逻辑的设计框架。

在这里首先对 shift_bt 进行 case 分类：

0（加法类）：根据时钟错位分别显示和的十六进制进位与低位。

1（计时器类）：那就继续对 rw 进行 case 分类：

0（开始计时类）：利用时钟的两位二进制，在四个数码管上分别错位显示计时结果的从高到低 4 个十六进制位。

1（暂停计时类）：此时时钟暂停，若继续使用第一个时钟，四个数码管有且仅有一个会显示结果，于是利用第二个独立时钟的两位二进制 ，在四个数码管上分别显示：STOP。

3.实验中遇到的问题及解决方案

所遇问题	解决方案
在实验中遇到的程序上最大的问题就是对参数的定义：reg 与 wire 的混淆勿用，导致程序一直一直在报错。	查找网上相关资料理解概念： wire 对应于连续赋值，如 assign reg 对应于过程赋值，如 always
硬件方面所遇最大的问题是 ISE 版本不兼容问题：开发环境 ISE14.2（nt64），在遇到微动开关时，synthesis-translate-map 都可以一一通过，但在 place&route 中会出现以下错误：Par:100 - Design is not completely routed. There are 180 signals that are not completely routed in this design. See the "final_top.unroutes" file for a list of all unrouted signals.	这个问题困扰了我很久很久，在向助教求助时，助教给予的解释是：版本不兼容。之后我又在实验室的电脑上运行了该程序，出现了不同的错误（这个问题我还是没想明白，因为看讲义的时候，那些微动开关是可以正常定义的）。 最后，在浏览了诸多论坛以后，我发现了解决的方案，就是在每一个微动开关管脚定义的时候，加上一句： NET "xxx" CLOCK_DEDICATED_ROUTE = FALSE; 意思就是当时钟与微动开关信号冲突时，牺牲时钟，让微动开关先处理事件。
由于功能集成，许多参量的值都会被重用，因此在模式切换的时候，一些未被考虑到的参数可能会导致一些难以预测的结果。 比如：在计时器计时暂停的时候，如果直接切换成加法器模式，此时时钟还处在暂停状态，两位十六进制和只会随机显示一位。	在把两个功能集成的时候，我设计了一个顶层逻辑结构，考虑了每一种情况中所有变量的取值，以及在模式切换时变量的随带更改。 用上了诸多的条件判断语句来保证每一种情况都能在规则内运行，防止误操作带来的不可逆中断。

4.实验感想与心得体会

只进行了两次 FPGA 课程的学习，并且在课上还有代码与详细操作步骤可供参考。这次独立完成一个 FPGA 的大作业制作，在最初有了一个大致的设想后，觉得这些功能应该是很容易就可以实现吧，但当开始着手时，发现非常困难，连实现一个很简单的功能都需要 debug 很久很久，第一个晚上一无所获，感觉完全实现自己的设想大概是无从谈起了。

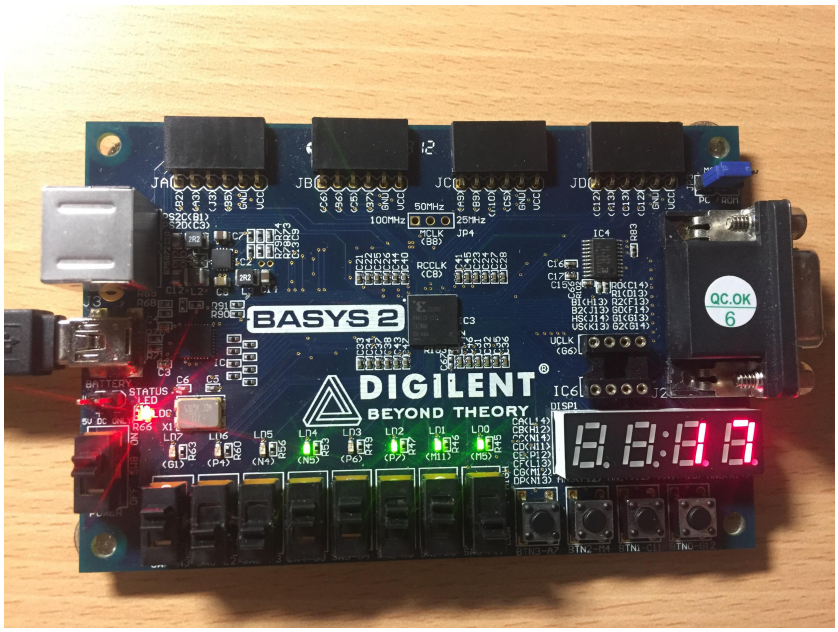
之后在图书馆里借来了两本《verilog 编程基础》，又花了一晚上把一些基础语法与表达式

大致熟悉了一遍，心理才感觉比较安定，第三个晚上便开始了大作业的实现。在这个过程中，询问过助教，查阅过网络论坛、博客，也问过班里的同学，也与这个版本的 ISE 作了许多斗争（最后发现每次闪退的原因是：不支持中文输入，一打中文注释就闪退）。当然，很庆幸的是，紧赶慢赶，经过无数次的调试，最终还是实现了最初的设想。

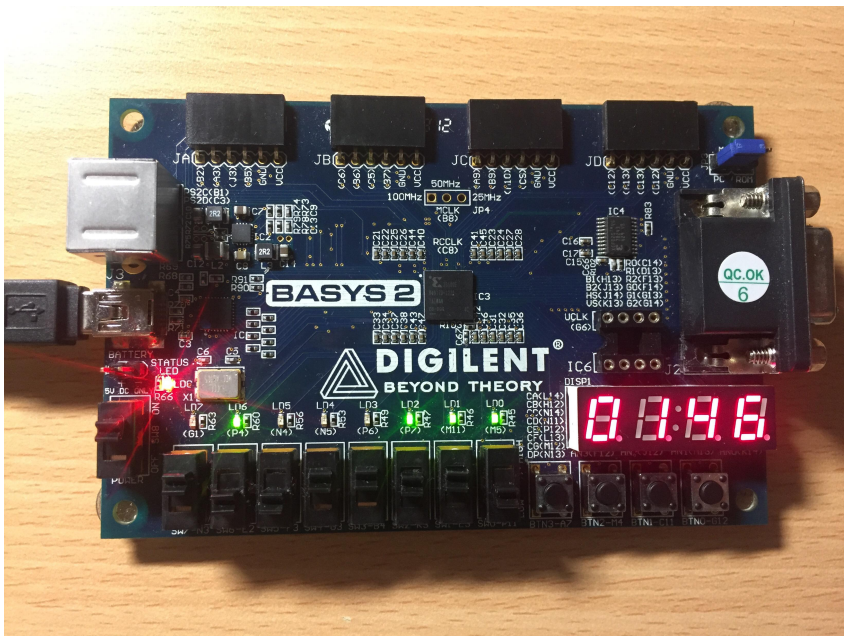
感谢在这个过程中帮助过我的老师，助教和同学们！

5. 作品展示

加法器：



计时器：



6.源代码

Verilog 文件：

```
module final_top(


input wire [3:0] addition_1 ,      //第一个四位二进制加数
input wire [3:0] addition_2 ,      //第二个四位二进制加数
input wire add ,                   //加法微动开关
input wire clr ,                   //清除微动开关
input wire clk ,                   //时钟引脚
output reg [7:0] LED,              //LED 管引脚定义
output reg [6:0]a_to_g ,           //数码管引脚定义
output reg[3:0] an,                //数码管使能端定义
input wire start ,                 //计时器暂停/开始微动开关
input wire shift ) ;               //功能切换微动开关

reg cout ;                         //第二个和位
reg [3:0] num ;                    //第一个和位
reg [36:0] clk_cnt ;               //第一个时钟
reg [36:0] clk_cnt2 ;              //第二个时钟
reg rw ;                           //表征计时器 start/pause 的参量
reg shift_bt ;                     //表征功能切换的参量
reg [3:0]num2 ;                    //存储计时器数码管的每一个十六进制位
reg [3:0]stop ;                    //显示计时器暂停时的 STOP
reg [7:0] LED_show ;               //LED 管的八位二进制存储

always @(*)
case(shift_bt)
0:
    begin
        an[3]=~clk_cnt[15] ;
        an[2]=clk_cnt[15] ;
        an[0]=1 ;
        an[1]=1 ;
    end
1:
    case(rw)
    0:
        case(clk_cnt[16])
        1:

            case(clk_cnt[15])
            1:
                begin
                    an[0]=0 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            0:
```

```
                begin
                    an[0]=1 ;
                    an[1]=0 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            endcase

        0:
            case(clk_cnt[15])
            1:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=0 ;
                    an[3]=1 ;
                end
            0:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=0 ;
                end
            endcase
        endcase
    1:
        case(clk_cnt2[16])
        1:
            case(clk_cnt2[15])
            1:
                begin
                    an[0]=0 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            0:
                begin
                    an[0]=1 ;
                    an[1]=0 ;
                    an[2]=1 ;
                    an[3]=1 ;
                end
            endcase
        0:
            case(clk_cnt2[15])
            1:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=0 ;
                    an[3]=1 ;
                end
```

```
            0:
                begin
                    an[0]=1 ;
                    an[1]=1 ;
                    an[2]=1 ;
                    an[3]=0 ;
                end
            endcase
        endcase
    endcase
endcase

always @(*)
case(clk_cnt[16])
    1:

        case(clk_cnt[15])
        1:
            num2=clk_cnt[36:33] ;
        0:
            num2=clk_cnt[32:29] ;
        endcase

    0:
        case(clk_cnt[15])
        1:
            num2=clk_cnt[28:25] ;
        0:
            num2=clk_cnt[24:21] ;
        endcase
endcase


always@(posedge add or posedge clr or  posedge clk )
if(add)
        begin
                {cout,LED_show[3:0]}= addition_1 +addition_2 ;
                {cout,num}= addition_1 +addition_2 ;
                LED_show[4]=cout ;
        end
else
if(clr)
        begin
                clk_cnt=0 ;
                LED_show=8'b00000000 ;
                cout=0 ;
                num=4'b0000 ;
        end
else
        case(rw)
        0:
            begin
                clk_cnt = clk_cnt+1 ;
                if(clk_cnt[36:33]>15)
```

```
                    clk_cnt=0 ;

                clk_cnt2 = clk_cnt2+1 ;
                if(clk_cnt2[36:33]>15)
                    clk_cnt2=0 ;
            end
        1:
        begin
            clk_cnt2 = clk_cnt2+1 ;
            if(clk_cnt2[36:33]>15)
                clk_cnt2=0 ;
        end
        endcase

always @(posedge start or posedge shift)
if(shift)
    begin
        if(shift_bt==0)
        begin
            rw=1 ;
            shift_bt=1 ;
        end
        else
        begin
            shift_bt=0 ;
            rw=0;
        end
    end
else if(shift_bt==1)
begin
    if(rw==1)
        rw=0 ;
    else
        rw=1 ;
end

always @(*)
case(shift_bt)
0:
    LED=LED_show ;
1:
    LED=clk_cnt[28:21] ;
endcase

always @(*)
case(shift_bt)
0:
    case(clk_cnt[15])
    1:
        case(num)
        0:a_to_g=7'b0000001;
        1:a_to_g=7'b1001111;
        2:a_to_g=7'b0010010;
        3:a_to_g=7'b0000110;
```

```
        4:a_to_g=7'b1001100;
        5:a_to_g=7'b0100100;
        6:a_to_g=7'b0100000;
        7:a_to_g=7'b0001111;
        8:a_to_g=7'b0000000;
        9:a_to_g=7'b0000100;
        'hA: a_to_g=7'b0001000;
        'hB: a_to_g=7'b1100000;
        'hC: a_to_g=7'b0110001;
        'hD: a_to_g=7'b1000010;
        'hE: a_to_g=7'b0110000;
        'hF: a_to_g=7'b0111000;
        default: a_to_g=7'b0000001;
        endcase
    0:
        case(cout)
        0:a_to_g=7'b0000001;
        1:a_to_g=7'b1001111;
        2:a_to_g=7'b0010010;
        3:a_to_g=7'b0000110;
        4:a_to_g=7'b1001100;
        5:a_to_g=7'b0100100;
        6:a_to_g=7'b0100000;
        7:a_to_g=7'b0001111;
        8:a_to_g=7'b0000000;
        9:a_to_g=7'b0000100;
        'hA: a_to_g=7'b0001000;
        'hB: a_to_g=7'b1100000;
        'hC: a_to_g=7'b0110001;
        'hD: a_to_g=7'b1000010;
        'hE: a_to_g=7'b0110000;
        'hF: a_to_g=7'b0111000;
        endcase
    endcase
1:
    case(rw)
    0:
        case(num2)
            0:a_to_g=7'b0000001;
            1:a_to_g=7'b1001111;
            2:a_to_g=7'b0010010;
            3:a_to_g=7'b0000110;
            4:a_to_g=7'b1001100;
            5:a_to_g=7'b0100100;
            6:a_to_g=7'b0100000;
            7:a_to_g=7'b0001111;
            8:a_to_g=7'b0000000;
            9:a_to_g=7'b0000100;
            'hA: a_to_g=7'b0001000;
            'hB: a_to_g=7'b1100000;
            'hC: a_to_g=7'b0110001;
            'hD: a_to_g=7'b1000010;
            'hE: a_to_g=7'b0110000;
            'hF: a_to_g=7'b0111000;
```

```
        endcase
    1:
    case(clk_cnt2[16])
        1:
            case(clk_cnt2[15])
            1:
                a_to_g=7'b0100100;
            0:
                a_to_g=7'b1110000;
            endcase
        0:
            case(clk_cnt2[15])
            1:
                a_to_g=7'b0000001;
            0:
                a_to_g=7'b0011000;
            endcase
    endcase
    endcase
endcase

endmodule
```

ucf 管脚定义文件：

```
NET "LED[0]" LOC = M5;
NET "LED[1]" LOC = M11;
NET "LED[2]" LOC = P7;
NET "LED[3]" LOC = P6;
NET "LED[4]" LOC = N5 ;
NET "LED[5]" LOC = N4;
NET "LED[6]" LOC = P4;
NET "LED[7]" LOC = G1;

NET "addition_1[0]" LOC = G3;
NET "addition_1[1]" LOC = F3;
NET "addition_1[2]" LOC = E2;
NET "addition_1[3]" LOC = N3;
NET "addition_2[0]" LOC = P11;
NET "addition_2[1]" LOC = L3;
NET "addition_2[3]" LOC = B4;
NET "addition_2[2]" LOC = K3;

NET "a_to_g[0]" LOC = M12;
NET "a_to_g[1]" LOC = L13;
NET "a_to_g[2]" LOC = P12;
NET "a_to_g[3]" LOC = N11;
NET "a_to_g[4]" LOC = N14;
NET "a_to_g[6]" LOC = L14;
NET "a_to_g[5]" LOC = H12;

NET "an[0]" LOC = K14 ;
NET "an[1]" LOC = M13 ;
NET "an[2]" LOC = J12 ;
```

```
NET "an[3]" LOC = F12 ;

NET "clk" LOC = B8 ;
NET "add" LOC = G12;
NET "clr" LOC = C11;
NET "start" LOC = A7 ;
NET "shift" LOC = M4 ;

NET "start" CLOCK_DEDICATED_ROUTE = FALSE;
NET "shift" CLOCK_DEDICATED_ROUTE = FALSE;
NET "add" CLOCK_DEDICATED_ROUTE = FALSE;
NET "clr" CLOCK_DEDICATED_ROUTE = FALSE;
```