本科实验报告

课程名称:	计算机逻辑设计基础
姓　　名:	张晋恺
学　　院:	竺可桢学院
系:	所在系
专　　业:	计算机科学与技术
学　　号:	3230102400
指导教师:	董亚波

2024 年 6 月 4 日

浙江大学实验报告

课程名称: 计算机逻辑设计基础 实验类型: 综合

实验项目名称: 基本开关电路

学生姓名: 张晋恺 专业: 计算机科学与技术 学号: 3230102400

同组学生姓名: 杨吉祥 指导老师: 董亚波

实验地点: 东 4-511 实验日期: 2024 年 6 月 5 日

一、实验目的

1. 掌握同步四位二进制计数器 74LS161 的工作原理和设计方法
2. 掌握时钟/定时器的工作原理和设计方法

二、实验内容和原理

实验内容

- 任务一：采用行为描述设计同步四位二进制计数器 74LS161
- 任务二：基于 74LS161 设计时钟应用

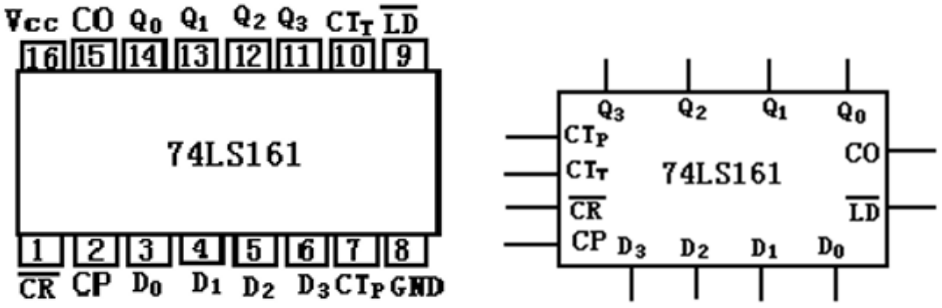
实验原理

同步四位二进制计数器 74LS161

74LS161 是常用的四位二进制可预置的同步加法计数器, 可灵活运用在各种数字电路，实现分频器等很多重要的功能

74LS161 的功能描述如下：

- 含有异步清零端 $\overline{CR}$
- 同步加载端 $\overline{LD}$
- 使能端 CT_P,CT_T
- 进位输出端 CO



(a) 74LS161 引脚图　　(b) 74LS161 引脚说明

图 1: 74LS161

当清零端 $\overline{CR}=1,(CR=0)$，计数器输出 $Q_3Q_2Q_1Q_0$ 立即为全 0，这个时候为异步复位功能。当 $\overline{CR}=0,(CR=1)$, 且 $\overline{LD}=1,(LD=0)$，在 CP 信号上升沿作用后，74LS161 输出端 $Q_3Q_2Q_1Q_0$ 的状态分别与并行数据输入端 $D_3D_2D_1D_0$ 的状态一样，为同步置数功能。而只有当 $CR=LD=CT_P=CT_T=1$、CP 脉冲上升沿作用后，计数器加 1。进位输出端 CO 的其逻辑关系是 $CO=Q_0Q_1Q_2Q_3CT_T$。合理应用计数器的清零功能和置数功能，一片 74LS161 可以组成 16 进制以下的任意进制分频器。

输 入			输 出
$\overline{CR}$ $\overline{LD}$ CT_P CT_T CP	$D_3D_2D_1D_0$	$Q_3Q_2Q_1Q_0$
0 × × × ×	××××	0 0 0 0
1 0 × × ↑	$d_3d_2d_1d_0$	$d_3d_2d_1d_0$
1 1 0 1 ×	××××	保 持
1 1 × 0 ×	××××	保 持
1 1 1 1 ↑	××××	计 数

图 2: 74LS161 功能表

时序波形如下：

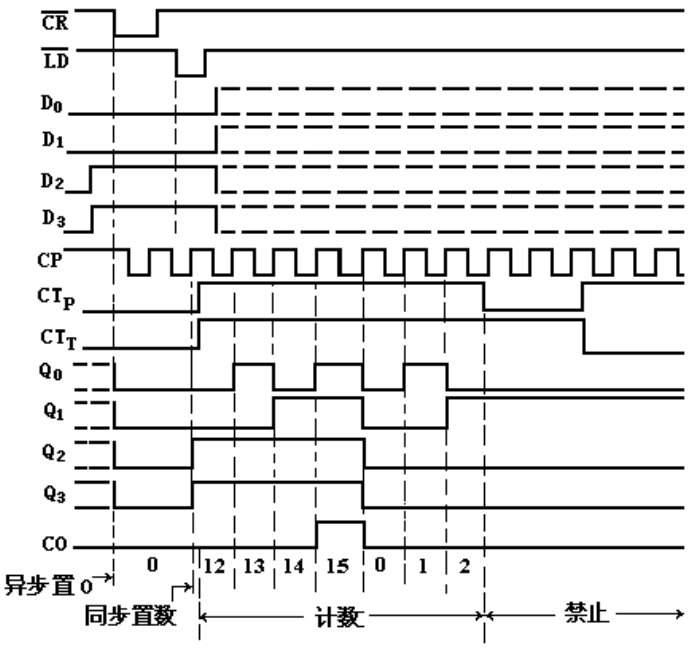


图 3: 74LS161 时序波形

例如，利用 74LS161 实现十进制计数器，只需用非门判断终止状态 1010，就可以实现十进制计数 (0000 到 1001)，改变与非门的输入信号，就可以实现其他进制计数；改变与非门输出信号和输入信号，可以实现同步加载功能。

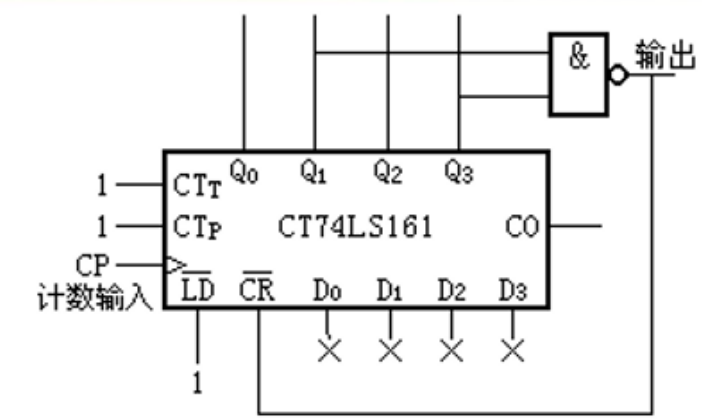


图 4: 十进制计数器

也可以使用串联设计 16x16 进制计数器

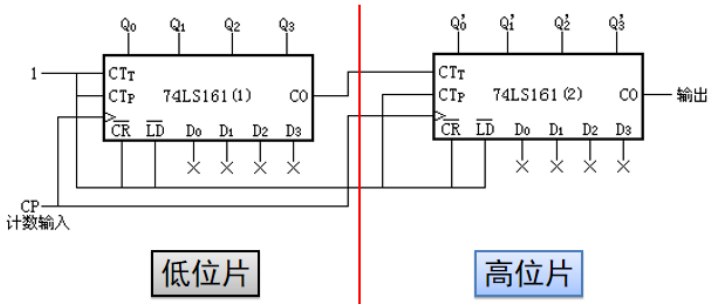


图 5: 16x16 进制计数器

在计到 1111 以前，CO_1 输出为 0，高位片保持原状态不变，当计数到 1111 时，CO_1 输出为 1，高位片在下一个时钟上升沿时加 1，这样就实现了 16x16 进制计数器。

也可以把多个 74LS161 级联，实现更高位数的计数器。例如 50 进制计数器

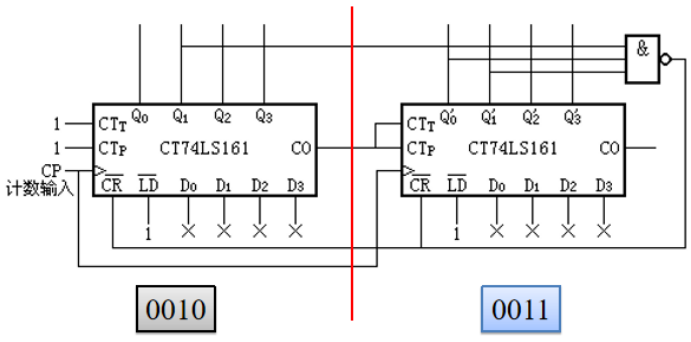


图 6: 50 进制计数器

十进制数 50 对应的二进制数为 0011 0010，可以实现从 0000 0000 到 0011 0001 的计数。

数字时钟

可以使用类似的方法设计数字时钟，时钟的设计原理如下：

- 时钟的计数范围为 0-23，分为 0-59，秒为 0-59
- 时钟的显示范围为 00-23:00-59:00-59
- 时钟的设计需要三个 74LS161 计数器，分别用于时、分、秒的计数
- 时钟的设计需要一个 74LS138 译码器，用于将计数器的输出转换为数码管的输入
- 时钟的设计需要一个 555 定时器，用于产生时钟信号

设计要求：数字钟的初值通过计数器同步置位的方式实现，默认加载 23:58:30。选择大实验板上的 6 个数码管显示，前两位显示小时的十位和个位，中间两位显示分钟的十位和个位，最后两位显示秒的十位和个位。

三、实验步骤和结果记录

任务一：采用行为描述设计同步四位二进制计数器 74LS161

建立工程

- 新建工程 My74LS161，添加源文件 My74LS161.v
- 用行为描述设计同步四位二进制计数器 74LS161，其中 CR 是异步清零，低电平有效 LD 是同步置位，低电平有效
- Verilog 代码如下：

```
module My74LS161(
    input CP,
    input CR,
    input LD,
    input CTP,CTT,
    input [3:0] D,
    output reg [3:0] Q,
    output CO
    );
    initial Q=4'b0000;
    always @(posedge CP,negedge CR) begin
        if(~CR) begin
            Q<=4'b0000;
        end
        else if(~LD) begin
            Q<=D;
        end
        else begin
            if(CTP & CTT) begin
                Q<=Q+1;
            end
        end
    end
    assign CO=(&Q)&CTT;
endmodule
```

仿真

仿真代码如下：

```
module My74LS161_tb(

);
reg clk,CR,LD,CTP,CTT;
reg [3:0] D;
wire [3:0] Q;
wire CO;
My74LS161 UUT(
    .CP(clk),
    .CR(CR),
    .LD(LD),
    .CTP(CTP),
    .CTT(CTT),
    .D(D),
    .Q(Q),
    .CO(CO)
    );
    initial begin
    CR = 0;
    D = 0;
    CTP = 0;
    CTT = 0;
    LD = 0;

    #100;
    CR = 1;
    LD = 0;
    D = 4'b1100;
    CTT = 0;
    CTP = 0;
    #20 CR = 1;
    #10 LD = 1;
    #30 CTT = 1;
    CTP = 1;
    #510;
    CR = 0;
    #500;
end
initial begin
    clk=0;
    forever #5 clk=~clk;
```

```
    end
endmodule
```

仿真波形如下：

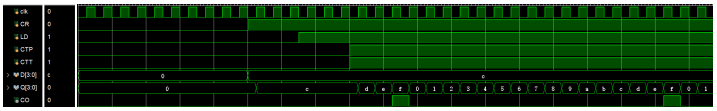


图 7: My74LS161 仿真波形

仿真代码解释：一开始，让 CR=0，给计数器置 0，然后 CR=1，LD=0，给计数器置数为 1100，然后 LD=1，CTT=1，CTP=1，开始计数，最后 CR=0，停止计数，对应仿真波形可知仿真结果正确。符合我们的预期。

任务二：基于 74LS161 设计时钟应用

建立工程

- 新建工程 MyClock，添加源文件 MyClock.v
- 用行为描述设计时钟应用，并进行仿真验证
- 调用 My74LS161，设计时、分、秒计数器
- 调用分频模块，每 128 个 clk 产生 1 个秒计数脉冲

```
module clk_100ms(clk, clk_100ms);
input wire clk;
output reg clk_100ms;
reg [31:0] cnt;
initial clk_100ms = 0;
always @ (posedge clk) begin
   if (cnt < 64) begin
       cnt <= cnt + 1'b1;
   end else begin
       cnt <= 0;
       clk_100ms <= ~clk_100ms;
   end
end
endmodule
```

- 调用实验 13 的 8 位数码管显示模块，在 8 位数码管上从左到右显示两位 00、时、分、秒
- 将 SW[15] 拨到 1，初始化时钟为 23:58:30
- 控制 clk 周期宽度和仿真时长，仿真到时钟输出 00:00:02
- 在仿真中中将 8 位数码管显示模块的输入数据加入仿真波形进行观察
- Verilog 代码如下：

```
1
2  module MyClock(
3      input wire clk_in,
4      input wire [15:0]SW,//15控置移位使能，14控制数码管清零，
       ↪  13控制数码管使能，12控制分秒时的CR，11-4输入时间，
       ↪  3-1控制时钟是否并行输入,0控制开始计数
5      output wire SEGCLK,//数码管时钟信号
6       output wire SEGCLR,
7       output wire SEGDT,
8       output wire SEGEN
9  );
10    wire [63:0] disp_num;
11    wire [31:0] clk_num;
12    wire [7:0] hour,min,sec;
13    wire [7:0] time_in;
14    assign clk_num={8'b0000_0000,hour,min,sec};
15    assign time_in=SW[11:4];
16    wire clk;
17    clk_100ms clk_divide(.clk(clk_in),.clk_100ms(clk));//时钟分频模块
18
19    My74LS161
       ↪  sec1(.CP(clk),.CR((~(sec[3]&sec[1])|SW[12])),.LD(~SW[1]),.CT
       ↪  P(SW[0]),.CTT(SW[0]),.D(time_in[3:0]),.Q(sec[3:0]));//0-9
20    My74LS161 sec2(.CP(clk),
21    .CR((SW[12])?1'b1:(~(sec[6]&sec[5]&(~sec[4])))),
22    .LD(~SW[1]),.CTP(sec[3]&sec[0]),
23    .CTT(sec[3]&sec[0]),.D(time_in[7:4]),
24    .Q(sec[7:4]));//9 秒进位
25
26    My74LS161 min1(.CP(clk),.CR(~(min[3]&min[1])|SW[12]),.LD(~SW[2])
       ↪  ,.CTP(sec[6]&sec[4]&sec[3]&sec[0]),.CTT(sec[6]&sec[4]&sec[3]
       ↪  &sec[0]),.D(time_in[3:0]),.Q(min[3:0]));//59秒进位
27    My74LS161 min2(.CP(clk),.CR(~(min[6]&min[5])|SW[12]),.LD(~SW[2])
       ↪  ,.CTP(min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),.CTT(min[3]
       ↪  &min[0]&sec[6]&sec[4]&sec[3]&sec[0]),.D(time_in[7:4]),.Q(min
       ↪  [7:4]));//9分59秒进位
```

```
28
29  My74LS161 hour1(.CP(clk),.CR((~((hour[3]&hour[1])|(hour[2]&hour[5])))|SW[12]),//24时或者10时清空
30  .LD(~SW[3]),.CTP(min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),//59分59秒
31  .CTT(min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),
32  .D(time_in[3:0]),
33  .Q(hour[3:0]));//0-9
34  My74LS161 hour2(.CP(clk),.CR((~(hour[2]&hour[5]))|SW[12]),//24时清空
35  .LD(~SW[3]),
36  .CTP(hour[3]&hour[0]&min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),//9时59分59秒
37  .CTT(min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),
38  .D(time_in[7:4]),
39  .Q(hour[7:4]));//0-6
40
41
42  Disp_Decorder m0(.D(clk_num[3:0]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[7:0]));
43  Disp_Decorder m1(.D(clk_num[7:4]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[15:8]));
44  Disp_Decorder m2(.D(clk_num[11:8]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[23:16]));
45  Disp_Decorder m3(.D(clk_num[15:12]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[31:24]));
46  Disp_Decorder m4(.D(clk_num[19:16]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[39:32]));
47  Disp_Decorder m5(.D(clk_num[23:20]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[47:40]));
48  Disp_Decorder m6(.D(clk_num[27:24]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[55:48]));
49  Disp_Decorder m7(.D(clk_num[31:28]),.point(1'b0),.LE(1'b0),.SEGMENT(disp_num[63:56]));//数码管显示模块
50
51  wire [64:0] tmp;
52  wire finish,start,S_L;
53  wire start_shift;
54  Load_Gen k3(.clk(clk_in),.btn_in(clk),.Load_out(start_shift));//开始移位信号生成模块
55  assign start = SW[15] & start_shift;//开始移位信号
56  assign SEGCLK = clk_in | finish ;//移位结束后将时钟停止，避免继续移位
57  assign SEGCLR = SW[14];//清零信号，使所有数码管亮，低电平有效
58  assign SEGDT = tmp[64];
```

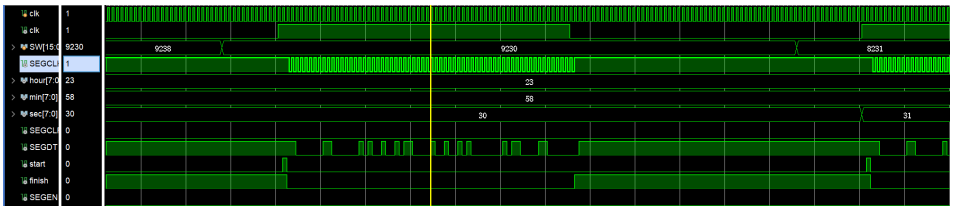
```
59      assign SEGEN = SW[13];//使能信号
60
61      AND_64bit m8(.in(tmp[63:0]),.out(finish));//移位结束信号
62
63      SR_LATCH m9(.S(start & finish
        ↪  ),.R(~finish),.Q(S_L));//移位信号锁存器
64
65      //构造 65 位左移移位寄存器
66      FD Dfiflop(.C(clk_in),.D((!S_L)?1'b1:1'b0),.Q(tmp[0]));//锁存器
        ↪  模块
67      shfit_reg8b m18(.clk(clk_in),.S_L(S_L),.s_in(tmp[0]),.p_in(disp
        ↪  _num[7:0]),.Q(tmp[8:1]));
68      shfit_reg8b m19(.clk(clk_in),.S_L(S_L),.s_in(tmp[8]),.p_in(disp
        ↪  _num[15:8]),.Q(tmp[16:9]));
69      shfit_reg8b m20(.clk(clk_in),.S_L(S_L),.s_in(tmp[16]),.p_in(dis
        ↪  p_num[23:16]),.Q(tmp[24:17]));
70      shfit_reg8b m21(.clk(clk_in),.S_L(S_L),.s_in(tmp[24]),.p_in(dis
        ↪  p_num[31:24]),.Q(tmp[32:25]));
71      shfit_reg8b m22(.clk(clk_in),.S_L(S_L),.s_in(tmp[32]),.p_in(dis
        ↪  p_num[39:32]),.Q(tmp[40:33]));
72      shfit_reg8b m23(.clk(clk_in),.S_L(S_L),.s_in(tmp[40]),.p_in(dis
        ↪  p_num[47:40]),.Q(tmp[48:41]));
73      shfit_reg8b m24(.clk(clk_in),.S_L(S_L),.s_in(tmp[48]),.p_in(dis
        ↪  p_num[55:48]),.Q(tmp[56:49]));
74      shfit_reg8b m25(.clk(clk_in),.S_L(S_L),.s_in(tmp[56]),.p_in(dis
        ↪  p_num[63:56]),.Q(tmp[64:57]));//移位寄存器模块
75      //
76  endmodule
```

仿真

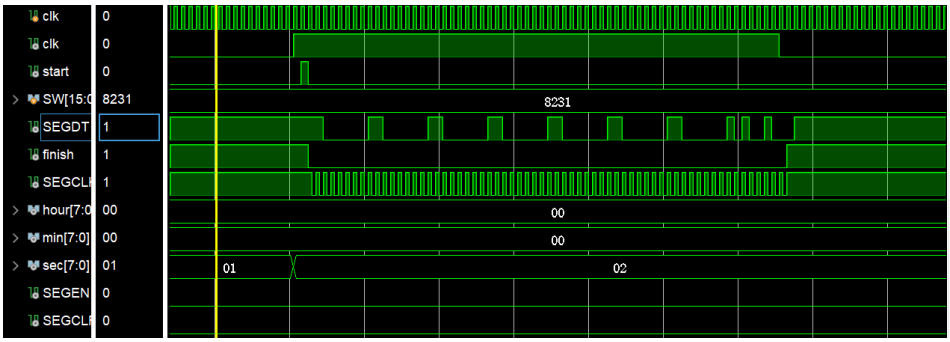
仿真代码如下：

```
1   module MyClock_tb(
2
3   );
4   reg clk;
5   reg [15:0]SW;
6   wire SEGCLK;
7   wire SEGCLR;
8   wire SEGDT;
9   wire SEGEN;
```

```
    MyClock MyClock_inst(
        .clk_in(clk),
        .SW(SW),
        .SEGCLK(SEGCLK),
        .SEGCLR(SEGCLR),
        .SEGDT(SEGDT),
        .SEGEN(SEGEN)
    );

    initial begin
        clk=0;
        forever begin
            #5 clk=~clk;
        end
    end
    initial begin
        SW=16'b0;
        SW[15]=1;//允许移位
        SW[12]=1;//设置 CR 为 1
        #1280; SW[1]=1;SW[11:4]=8'b00110000;//设置时间为 30 秒
        #1280; SW[1]=0;
        #1280; SW[2]=1;SW[11:4]=8'b01011000;//设置时间为 58 分
        #1280; SW[2]=0;
        #1280; SW[3]=1;SW[11:4]=8'b00100011;//设置时间为 23 时
        #1280; SW[3]=0;
        #1280; SW[12]=0;SW[0]=0;SW[1]=0;SW[2]=0;SW[0]=1;//CR正常,
        ↪  LD端不使能,正常计数
    end
endmodule
```



(a) 初始化



(b) 末状态

图 8: MyClock 仿真波形

仿真解释：为了让时钟实现自动更新，我们需要把输入的时钟进行分频作为计数器的时钟输入，分频次数为每 128 个 clk 周期产生一个时钟脉冲，这是因为我们需要移位 64 位数据，也就是 64 个上升沿 128 个 clk，然后我们需要一个开始信号，设为 SW[0]，当开始信号为 1 时，时钟开始计数，当时钟计数到设定的时间后，时钟停止计数，然后我们需要一个清零信号，当清零信号为 0 时，时钟清零，然后我们需要一个使能信号，当使能信号为 1 时，数码管显示时钟，当使能信号为 0 时，数码管不显示时钟。仿真波形中，我们可以看到时钟从 23:58:30 开始计数，到 00:00:02 停止计数，符合我们的预期。也可以看到在下一个计数时钟上升沿，数码管的数据已经更新，这样就实现了自动更新的时钟，不需要手动来按开关了。仿照 lab13 对于仿真波形 SEGDT 对应到数码管上的输出，也是正确无误的。因此，我们的仿真验证通过

在实验板上下载验证

Top 模块代码如下：

```
module Clock_Top(
    input wire clk,
    input wire [15:0]SW,//15控置移位使能，14控制数码管清零，13控制数码管使能，
    ↪  12控制分秒时的CR,3-1控制时钟是否并行输入,0控制开始计数
    output wire SEGCLK,//数码管时钟信号
    output wire SEGCLR,
```

```
    output wire SEGDT,
    output wire SEGEN
    );

    MyClock myClock(.clk_in(clk),.SW(SW),.SEGCLK(SEGCLK),.SEGCLR(SEGCLR)
     ↪  ,.SEGDT(SEGDT),.SEGEN(SEGEN));
endmodule
```

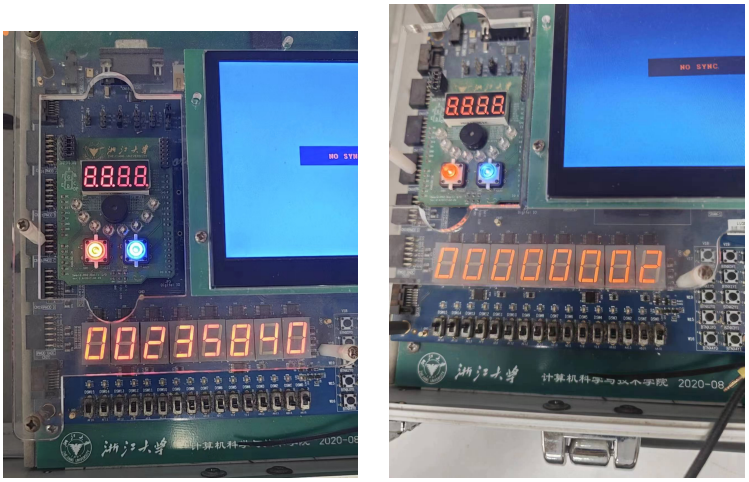
为了赋值简单，修改代码如下

```
 assign hour_in=8'b0010_0011;
 assign min_in=8'b0101_1000;
 assign sec_in=8'b0011_0000;
wire clk;
clk_100ms clk_divide(.clk(clk_in),.clk_100ms(clk));//时钟分频模块

My74LS161 sec1(.CP(clk),.CR((~(sec[3]&sec[1])|SW[12])),.LD(~SW[1]),.CTP(
 ↪  SW[0]),.CTT(SW[0]),.D(sec_in[3:0]),.Q(sec[3:0]));//0-9


My74LS161 sec2(.CP(clk),
.CR((SW[12])?1'b1:(~(sec[6]&sec[5]&(~sec[4])))),
.LD(~SW[1]),.CTP(sec[3]&sec[0]),
.CTT(sec[3]&sec[0]),.D(sec_in[7:4]),
.Q(sec[7:4]));//9 秒进位

My74LS161 min1(.CP(clk),.CR(~(min[3]&min[1])|SW[12]),.LD(~SW[2]),.CTP(se
 ↪  c[6]&sec[4]&sec[3]&sec[0]),.CTT(sec[6]&sec[4]&sec[3]&sec[0]),.D(min_
 ↪  in[3:0]),.Q(min[3:0]));//59秒进位
My74LS161 min2(.CP(clk),.CR(~(min[6]&min[5])|SW[12]),.LD(~SW[2]),.CTP(mi
 ↪  n[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),.CTT(min[3]&min[0]&sec[6]&s
 ↪  ec[4]&sec[3]&sec[0]),.D(min_in[7:4]),.Q(min[7:4]));//9分59秒进位

My74LS161 hour1(.CP(clk),.CR((~((hour[3]&hour[1])|(hour[2]&hour[5])))|SW
 ↪  [12]),//24时或者10时清空
.LD(~SW[3]),.CTP(min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]
 ↪  ),//59分59秒
.CTT(min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),
.D(hour_in[3:0]),
.Q(hour[3:0]));//0-9
My74LS161 hour2(.CP(clk),.CR((~(hour[2]&hour[5]))|SW[12]),//24 时清空
.LD(~SW[3]),
.CTP(hour[3]&hour[0]&min[6]&min[4]&min[3]&min[0]&sec[6]&sec[4]&sec[3]&se
 ↪  c[0]),//9时59分59秒
```

```
.CTT(min[3]&min[0]&sec[6]&sec[4]&sec[3]&sec[0]),
.D(hour_in[7:4]),
.Q(hour[7:4]));//0-6
```

在实验板上运行的具体操作如下

1. SW[13] 拨到 1，使能数码管
2. SW[15] 拨到 1，允许移位
3. SW[14] 拨到 1，数码管初始化，不用清零
4. SW[0] 保持为 0，先不开始计数
5. SW[12] 保持为 1，依次调节 SW[3:1]，设置时间为 23:58:30
6. SW[12] 拨到 0，SW[0] 拨到 1，开始计数，拍照记录
7. 观察自动计数到 00:00:02



(a) 计数中　　(b) 末状态

图 9: 下载运行

四、实验结果分析

相关实验结果以及对于仿真结果的分析，上板的具体操作步骤，以及具体现象，已在上一章节中展示，此处不再赘述。

五、讨论与心得

本次实验是我们计算机逻辑设计与基础的最后一次实验，因此我格外认真，在周日晚上就把实验相关代码完成了，周一的时候，自己去实验室尝试上板了一下，也十分成功。但是过程中还是有一些小的磕磕绊绊，比如在考虑怎么样让时钟实现自动更新的过程中，花了我不小的功夫，后面才想到可以用秒时钟比移位时钟慢的这一特点，再运用 Load_Gen 模块，来实现自动更新脉冲。

有了前一次实验的 SEG_DRV 模块的设计经验，本次实验本质上来说只需要掌握模 M 计数器的设计方法，就能比较顺利的完成，再加上任务量不大，可以说这次实验给了我 14 次实验以来完美的收尾。接下来，就需要认真写好最后一个大作业。非常期待在这门课结束后，我能在计算机组成这门课的实验里面找到更多的乐趣!