浙江大学

本科实验报告

课程名称：	计算机逻辑设计基础
姓　　名：	秦嘉俊
学　　院：	竺可桢学院
系：	所在系
专　　业：	计算机科学与技术
学　　号：	3210106182
指导教师：	董亚波

2022 年 12 月 24 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: 寄存器和寄存器传输设计

学生姓名： 秦嘉俊 专业： 计算机科学与技术 学号： 3210106182

同组学生姓名： 张瑞 指导老师： 董亚波

实验地点： 玉湖七幢 629 实验日期： 2022 年 12 月 25 日

一、实验目的和要求

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/定时器的工作原理与设计方法

二、实验内容和原理

实验设备

- 装有 Xilinx ISE 14.7 的计算机 1 台
- SWORD 开发板 1 套

内容

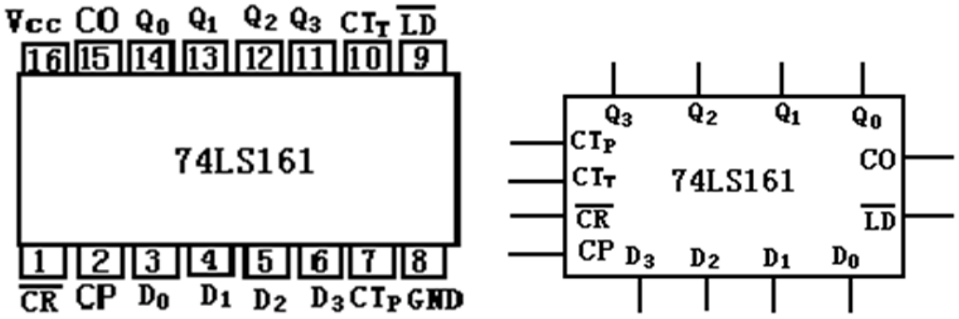
1. 任务 1：采用行为描述设计同步四位二进制计数器 74LS161
2. 任务 2：基于 74LS161 设计时钟应用

原理

2.1 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 的引脚分布如下图所示



(a) 74LS161 引脚分布　　(b) 引脚功能说明

其中，CR 为异步清零端，LD 为同步加载端，CT_P 和 CT_T 为同步使能端，CO 为进位输出端。当清零端 $CR = 0$，计数器输出 Q_3、Q_2、Q_1、Q_0 立即为全 0，这个时候为异步复位功能。当 $CR = 1$ 且 $LD = 0$ 时，在 CP 信号上升沿作用后, 74LS161 输出端 Q_3、Q_2、Q_1、Q_0 的状态分别与并行数据输入端 D_3、D_2、D_1、D_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。

74LS161 的时序图如下所示：

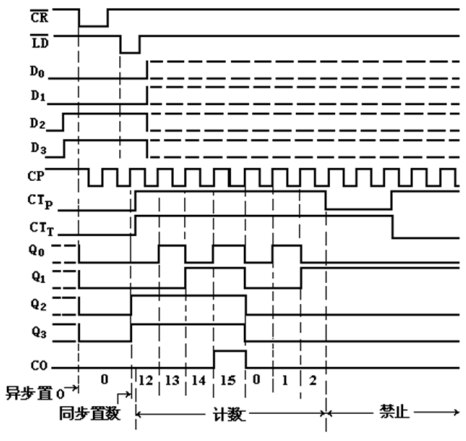


图 1: 74LS161 的时序图

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；该变与非门输出信号和输入信号，可以实现同步加载。

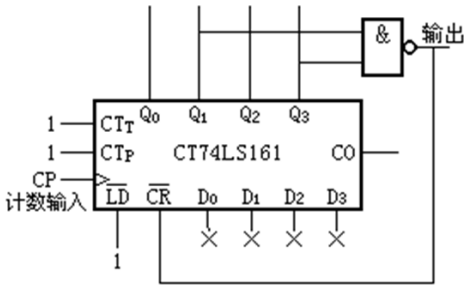


图 2: 利用 74LS161 实现十进制计数器

下面是是实现 16×16 进制计数器的逻辑图。

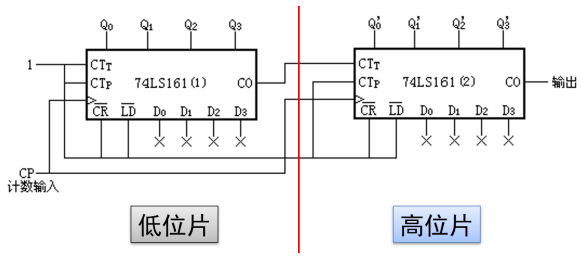


图 3: 利用 74LS161 实现 16×16 进制计数器

其中，在记到 1111 之前，$CO_1 = 0$，高位片保持不变。计到 1111 时，$CO_1 = 1$，高位片在下一个进位加一。

2.2 时钟应用设计

下图是在上面的基础上实现的 50 进制计数器。其中，十进制数 50 对应的二进制数为 0011 0010，则计数器应该从 0000 0000 计到 0011 0001。在图 2.6 中，一旦出现 0011 0010，则与非门会输出 0，CR 输入为 0，计数器清零，重新开始计数。

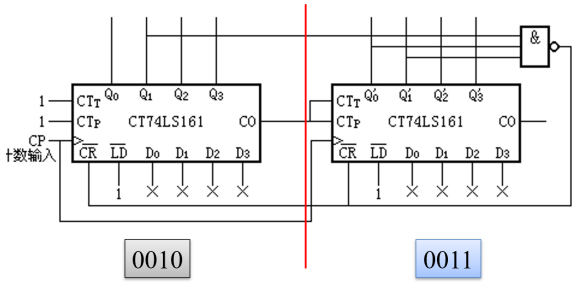


图 4: 利用 74LS161 实现的 50 进制计数器

我们可以使用类似的办法实现 60 进制和 24 进制计数器，那么就可以实现 24 小时内时间的显示。数字钟的初值通过初始化语句来实现，选择大实验版上的 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，最后两秒显示秒的十位和个位。

三、实验过程和数据记录

3.1 采用行为描述设计同步四位二进制计数器 74LS161

1. 新建工程, 工程名称用 My74LS161, Top Level Source Type 用 HDL
2. 用行为描述设计, 其中 CR 是异步清零，低电平有效; LD 是同步置位，低电平有效

   输入代码如下:

```
        module My74LS161(
            input CP,CR,
            input Ld,
            input CTP, CTT,
            input [3:0] D,
            output reg [3:0] Q,
            output CO
        );
            initial Q = 4'b0;
            always @(posedge CP,negedge CR) begin
                if(~CR) begin
                    Q <= 0;
                end
                else
                if(~Ld) begin
                    Q <= D;
                end
                else begin
                    if(CTT & CTP)
                    begin
                    Q <= Q + 1;
                    end
```

```
            end
        end

        assign CO = (&Q) & CTT;

    endmodule
```

3. 仿真

   仿真代码如下：

```
    module My74LS161_sim;

    // Inputs
    reg CP;
    reg CR;
    reg CTP;
    reg CTT;
    reg Ld;
    reg [3:0] D;

    // Outputs
    wire [3:0] Q;
    wire CO;

    // Instantiate the Unit Under Test (UUT)
    My74LS161 uut (
        .CP(CP),
        .CR(CR),
        .CTP(CTP),
        .CTT(CTT),
        .Ld(Ld),
        .D(D),
        .Q(Q),
        .CO(CO)
    );

    initial begin
        // Initialize Inputs
        CP = 0;
```

```
            CR = 0;
            CTP = 0;
            CTT = 0;
            Ld = 0;
            D = 0;

            // Wait 100 ns for global reset to finish
            #100;
          CR = 1;
            Ld = 1;
            D = 4'b1100;
            CTT = 0;
            CTP = 0;

            #30 CR = 0;
            #20 CR = 1;
            #10 Ld = 0;
            #30 CTT = 1;
            CTP = 1;
            #10 Ld = 1;

            #510;
            CR = 0;
            #20 CR = 1;
            #500;

            // Add stimulus here
        end
        always begin
            CP = 0;#20;
            CP = 1;#20;
        end

    endmodule
```

最后得到如下的仿真波形图

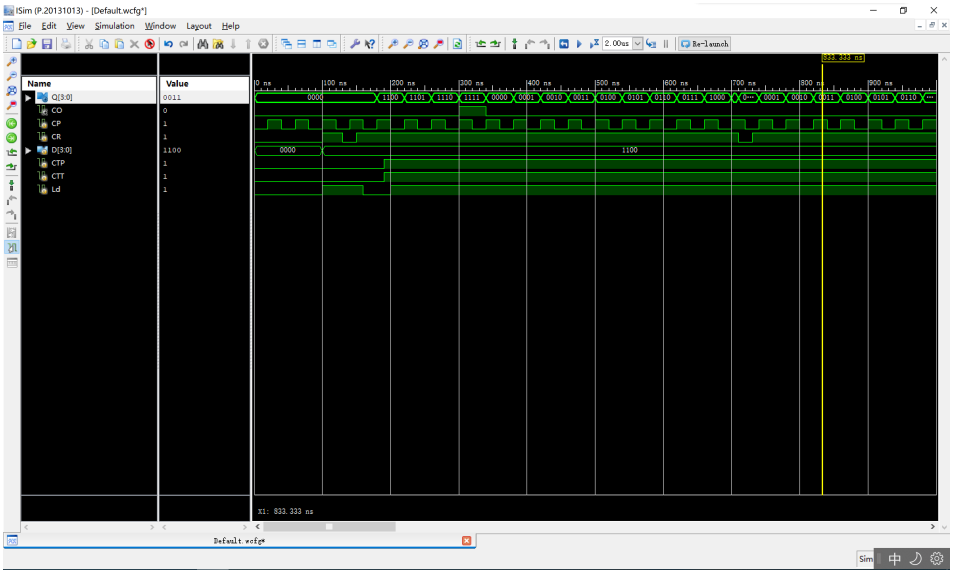


图 5: 仿真波形图

3.2 基于 74LS161 设计时钟应用

1. 新建工程, 工程名称用 MyClock, Top Level Source Type 用 HDL.

2. 用行为描述设计, 要求调用 My74LS161; 调用分频模块，用 100ms 作为秒的驱动时钟; 调用实验 13 的 8 位数码管显示模块显示时分秒; 用 SW[15] 初始化时钟为 23:58:30

   其中我们的代码如下：

```
        module top(
            input clk2,
            input CTP,
            input CTT,
            input [7:0]hour_in,
            input [7:0]min_in,
            input [7:0]sec_in,
            input sec_ld, min_ld, hour_ld,
            output clk,
            output [23:0]num,
            output SEG_CLK, SEG_DT,
            output [63:0] disp_num
            );

```

```
15          wire [7:0]hour;
16          wire [7:0]min;
17          wire [7:0]sec;
18
19          assign num[7:0] = sec;
20          assign num[15:8] = min;
21          assign num[23:16] = hour;
22
23          clk_100ms m7(.clk(clk2),.clk_100ms(clk));
                        //128
24
25
26          My74LS161 m0(clk,~(sec[3]&sec[1]),sec_ld,CTP,CTT,
               sec_in[3:0],sec[3:0]);//9
27          My74LS161 m1(clk,~(sec[6]&sec[5]),sec_ld,sec[3]&sec
               [0],sec[3]&sec[0],sec_in[7:4],sec[7:4]);//5
28          My74LS161 m2(clk,~(min[3]&min[1]),min_ld,sec[6]&sec
               [4]&sec[3]&sec[0],sec[6]&sec[4]&sec[3]&sec[0],
               min_in[3:0],min[3:0]);
29          My74LS161 m3(clk,~(min[6]&min[5]),min_ld,(sec[6]&
               sec[4]&sec[3]&sec[0]&min[3]&min[0]),(sec[6]&sec[
               4]&sec[3]&sec[0]&min[3]&min[0]),min_in[7:4],min[
               7:4]);
30          My74LS161 m4(clk,~(hour[2]&hour[5]|hour[3]&hour[1])
               ,hour_ld,min[6]&min[4]&sec[6]&sec[4]&sec[3]&sec[
               0]&min[3]&min[0],min[6]&min[4]&sec[6]&sec[4]&sec
               [3]&sec[0]&min[3]&min[0],hour_in[3:0],hour[3:0])
               ;
31          My74LS161 m5(clk,~(hour[2]&hour[5]),hour_ld,hour[3]
               &hour[0]&min[6]&min[4]&sec[6]&sec[4]&sec[3]&sec[
               0]&min[3]&min[0],hour[3]&hour[0]&min[6]&min[4]&
               sec[6]&sec[4]&sec[3]&sec[0]&min[3]&min[0],
               hour_in[7:4],hour[7:4]);
32
33
34          SEGP2S m6(.clk(clk2),.reg_num(num),.LE(8'b11000000)
               ,.start_sig(clk),.SEG_CLK(SEG_CLK),.SEG_DT(
               SEG_DT),.num(disp_num));
```

```
35          endmodule
```

其中 SEGP2S 就是我们上次实验所设计的模块

```
1           module SEGP2S(
2               input wire clk,
3               input [31:0] reg_num,
4               input [7:0] LE,
5               input start_sig,
6               output wire SEG_CLK,
7               output wire SEG_CLR,
8               output wire SEG_EN,
9               output wire SEG_DT,
10              output wire [63:0] num
11              );
12
13              wire finish, start, SL;
14              wire [63:0] disp_num;
15              wire [64:0] Segment;
16
17              assign SEG_CLK = clk | finish;
18              assign SEG_CLR = 1'b1;
19             assign SEG_EN = 1'b1;
20              assign SEG_DT = Segment[64];
21
22              assign finish = Segment[0] & Segment[1] & Segment[2
                    ] & Segment[3] & Segment[4] & Segment[5] &
                    Segment[6] & Segment[7] & Segment[8] &
23              Segment[9] & Segment[10] & Segment[11] & Segment[12
                    ] & Segment[13] & Segment[14] & Segment[15] &
                    Segment[16] & Segment[17] &
24              Segment[18] & Segment[19] & Segment[20] & Segment[
                    21] & Segment[22] & Segment[23] & Segment[24] &
                    Segment[25] & Segment[26] &
25              Segment[27] & Segment[28] & Segment[29] & Segment[
                    30] & Segment[31] & Segment[32] & Segment[33] &
                    Segment[34] & Segment[35] &
26              Segment[36] & Segment[37] & Segment[38] & Segment[
                    39] & Segment[40] & Segment[41] & Segment[42] &
```

```
   Segment[43] & Segment[44] &
27          Segment[45] & Segment[46] & Segment[47] & Segment[
   48] & Segment[49] & Segment[50] & Segment[51] &
   Segment[52] & Segment[53] &
28          Segment[54] & Segment[55] & Segment[56] & Segment[
   57] & Segment[58] & Segment[59] & Segment[60] &
   Segment[61] & Segment[62] &
29          Segment[63];
30
31
32          MyMC14495 m0(.point(1'b0),.LE(LE[0]),.D0(reg_num[0]
   ),.D1(reg_num[1]),.D2(reg_num[2]),.D3(reg_num[3]
   ),.a(disp_num[0]),.b(disp_num[1]),.c(disp_num[2]
   ),.d(disp_num[3]),.e(disp_num[4]),.f(disp_num[5]
   ),.g(disp_num[6]),.p(disp_num[7]));
33          MyMC14495 m1(.point(1'b0),.LE(LE[1]),.D0(reg_num[4]
   ),.D1(reg_num[5]),.D2(reg_num[6]),.D3(reg_num[7]
   ),.a(disp_num[8]),.b(disp_num[9]),.c(disp_num[10
   ]),.d(disp_num[11]),.e(disp_num[12]),.f(disp_num
   [13]),.g(disp_num[14]),.p(disp_num[15]));
34          MyMC14495 m2(.point(1'b0),.LE(LE[2]),.D0(reg_num[8]
   ),.D1(reg_num[9]),.D2(reg_num[10]),.D3(reg_num[
   11]),.a(disp_num[16]),.b(disp_num[17]),.c(
   disp_num[18]),.d(disp_num[19]),.e(disp_num[20])
   ,.f(disp_num[21]),.g(disp_num[22]),.p(disp_num[
   23]));
35          MyMC14495 m3(.point(1'b0),.LE(LE[3]),.D0(reg_num[12
   ]),.D1(reg_num[13]),.D2(reg_num[14]),.D3(reg_num
   [15]),.a(disp_num[24]),.b(disp_num[25]),.c(
   disp_num[26]),.d(disp_num[27]),.e(disp_num[28])
   ,.f(disp_num[29]),.g(disp_num[30]),.p(disp_num[
   31]));
36          MyMC14495 m4(.point(1'b0),.LE(LE[4]),.D0(reg_num[16
   ]),.D1(reg_num[17]),.D2(reg_num[18]),.D3(reg_num
   [19]),.a(disp_num[32]),.b(disp_num[33]),.c(
   disp_num[34]),.d(disp_num[35]),.e(disp_num[36])
   ,.f(disp_num[37]),.g(disp_num[38]),.p(disp_num[
   39]));
```

```
37          MyMC14495 m5(.point(1'b0),.LE(LE[5]),.D0(reg_num[20
               ]),.D1(reg_num[21]),.D2(reg_num[22]),.D3(reg_num
               [23]),.a(disp_num[40]),.b(disp_num[41]),.c(
               disp_num[42]),.d(disp_num[43]),.e(disp_num[44])
               ,.f(disp_num[45]),.g(disp_num[46]),.p(disp_num[
               47]));
38          MyMC14495 m6(.point(1'b0),.LE(LE[6]),.D0(reg_num[24
               ]),.D1(reg_num[25]),.D2(reg_num[26]),.D3(reg_num
               [27]),.a(disp_num[48]),.b(disp_num[49]),.c(
               disp_num[50]),.d(disp_num[51]),.e(disp_num[52])
               ,.f(disp_num[53]),.g(disp_num[54]),.p(disp_num[
               55]));
39          MyMC14495 m7(.point(1'b0),.LE(LE[7]),.D0(reg_num[28
               ]),.D1(reg_num[29]),.D2(reg_num[30]),.D3(reg_num
               [31]),.a(disp_num[56]),.b(disp_num[57]),.c(
               disp_num[58]),.d(disp_num[59]),.e(disp_num[60])
               ,.f(disp_num[61]),.g(disp_num[62]),.p(disp_num[
               63]));
40
41
42          SLReg9b m8(.clk(clk), .S_L(SL), .s_in(1'b1), .p_in
               ({disp_num[7:0], 1'b0}), .Q(Segment[8:0]));
43          SLReg8b m9(.clk(clk), .S_L(SL), .s_in(Segment[8]),
               .p_in(disp_num[15:8]), .Q(Segment[16:9]));
44          SLReg8b m10(.clk(clk), .S_L(SL), .s_in(Segment[16])
               , .p_in(disp_num[23:16]), .Q(Segment[24:17]));
45          SLReg8b m11(.clk(clk), .S_L(SL), .s_in(Segment[24])
               , .p_in(disp_num[31:24]), .Q(Segment[32:25]));
46          SLReg8b m12(.clk(clk), .S_L(SL), .s_in(Segment[32])
               , .p_in(disp_num[39:32]), .Q(Segment[40:33]));
47          SLReg8b m13(.clk(clk), .S_L(SL), .s_in(Segment[40])
               , .p_in(disp_num[47:40]), .Q(Segment[48:41]));
48          SLReg8b m14(.clk(clk), .S_L(SL), .s_in(Segment[48])
               , .p_in(disp_num[55:48]), .Q(Segment[56:49]));
49          SLReg8b m15(.clk(clk), .S_L(SL), .s_in(Segment[56])
               , .p_in(disp_num[63:56]), .Q(Segment[64:57]));
50
51          //assign SL = 1'b0;
```

```
52              SR_LATCH m24(.S(start & finish), .R(~finish),.Q(SL)
                    );
53
54              LED m26(.clk(SEG_CLK), .s_in(SEG_DT), .num(num));
55
56              Load_Gen m25(.clk(clk), .btn_in(start_sig), .
                    Load_out(start));
57
58          endmodule
```

3. 仿真
   仿真实验要求

   - 分频模块改为 128 个 clk 周期驱动秒计数
   - 将时钟初始值改为 23:59:58
   - 控制 clk 周期宽度和仿真时长，仿真到时钟输出 00:00:02
   - 在 Top 模块中将 6 个 74LS161 模块的输出合并为 num[23:0] 输出，在仿真时用 16 进制显示
   - 同时输出 SEGCLK 和 SEGDT，观察是否有正确输出

   输入仿真代码如下

```
1           module top_sim;
2
3           // ins
4           reg clk2;
5           reg CTP;
6           reg CTT;
7           reg [7:0] hour_in;
8           reg [7:0] min_in;
9           reg [7:0] sec_in;
10          reg sec_ld;
11          reg min_ld;
12          reg hour_ld;
13
14          // Outputs
15          wire [23:0] num;
16          wire SEG_CLK;
```

```
17          wire SEG_DT;
18          wire [63:0] disp_num;
19          wire clk;
20          // Instantiate the Unit Under Test (UUT)
21          top uut (
22              .clk2(clk2),
23              .CTP(CTP),
24              .CTT(CTT),
25              .hour_in(hour_in),
26              .min_in(min_in),
27              .sec_in(sec_in),
28              .sec_ld(sec_ld),
29              .min_ld(min_ld),
30              .hour_ld(hour_ld),
31              .num(num),
32              .SEG_CLK(SEG_CLK),
33              .SEG_DT(SEG_DT),
34              .disp_num(disp_num),
35              .clk(clk)
36          );
37
38          initial begin
39              // Initialize ins
40              clk2 = 0;
41              CTP = 0;
42              CTT = 0;
43              hour_in = 0;
44              min_in = 0;
45              sec_in = 0;
46              sec_ld = 0;
47              min_ld = 0;
48              hour_ld = 0;
49
50              // Wait 100 ns for global reset to finish
51              #100;
52            #100;
53              CTT=0;
54              CTP=0;
```

```
55              sec_ld = 1;
56                min_ld = 1;
57                hour_ld = 1;
58                // Add stimulus here
59                hour_in[7:4]=2;
60                hour_in[3:0]=3;
61                min_in[7:4]=5;
62                min_in[3:0]=9;
63                sec_in[7:4]=5;
64                sec_in[3:0]=8;
65                CTT=0;
66                CTP=0;
67                #3000;
68                sec_ld = 0;
69                min_ld = 0;
70                hour_ld = 0;
71                #3000;
72                sec_ld = 1;
73                min_ld = 1;
74                hour_ld = 1;
75                CTT=1;
76                CTP=1;
77
78            end
79
80            always begin
81                clk2=0;#10;
82                clk2=1;#10;
83            end
84                // Add stimulus here
85        endmodule
```

得到如下的波形图：

(a) 首先我们初始化外部输入为 23 59 58

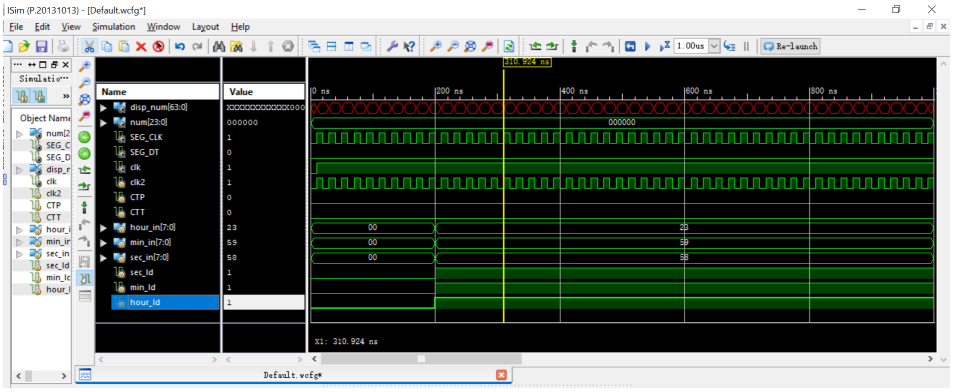


图 6: 仿真波形图 1

(b) 最开始 `ld` 信号均为 `0` 随后将 `ld=1`. 等待下一个时钟上升沿时 (`clk`) 会读入并行输入（也就是黄线这里）读入后可以看到 `disp_num` 就收到数据并开始移位，在下一个上升沿到来之前会完成移位

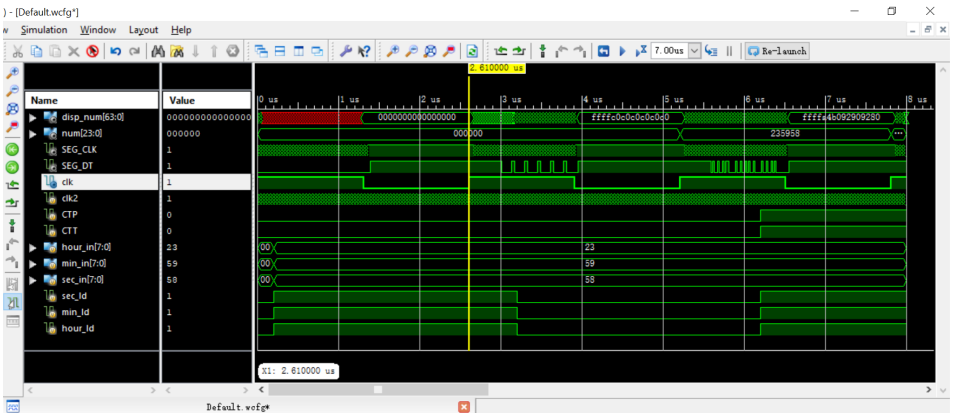


图 7: 仿真波形图 2

(c) 可以看到每次 `clk` 上升沿时 `num` 显示的数字都会自增一，达到 `235959` 时会 reset

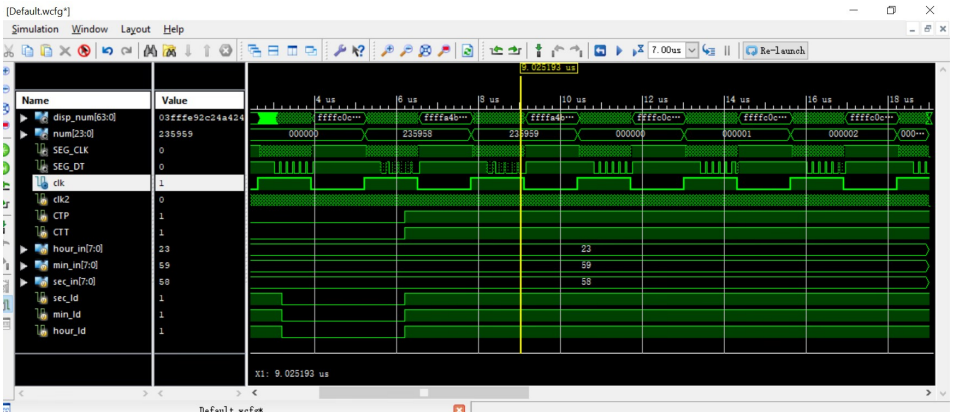


图 8: 仿真波形图 3

四、实验结果分析

相关结果都已经在前文写出。实验结果基本符合要求：仿真激励波形与真值表都相对应；仿真结果和 Verilog 代码在前文已经给出。

4.1 分析 Verilog 代码

在设计 My74LS161 中我们使用了 `assign CO = (&Q) & CTT;` 这样的语句。这里 `&` 是一个归约操作符, `&Q` 表示对 `Q` 所有位求与，这样使我们的代码简洁许多！（可惜以前不知道）

4.2 分析仿真结果

第二次实验的仿真中，我们可以观察一个周期内的 `SEG_DT` 的波形变化

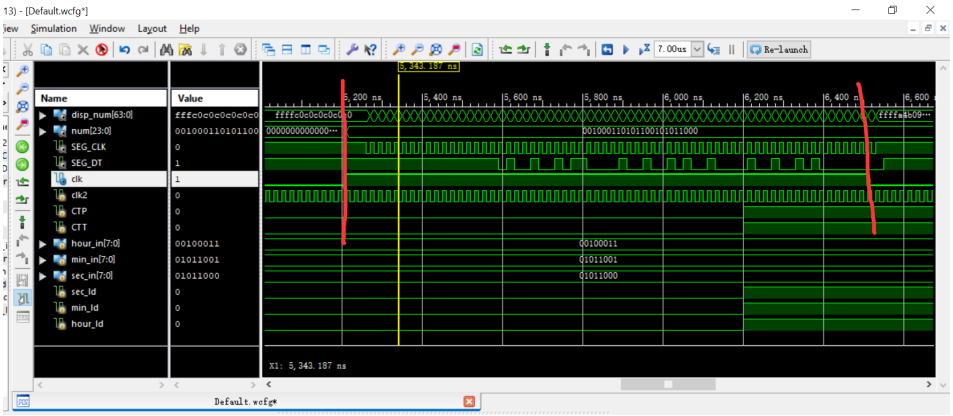


图 9: 仿真波形图

这是传输一次显示需要的 64 段码的波形 (SEG_DT) 这一连串信号会提供给我们的显示模块LED，如果有实验板就会体现为数码管显示数字。

五、讨论与心得

74LS161 逻辑功能较简单，代码也叫简洁，是对状态机的实践应用，加深了我对状态机的理论理解。在理解了 74LS161 的内部逻辑以后，调用模块对其进行改造实现任意进制的计数器也都十分方便，只要在到达进制上限以后进位归零即可。

再接再厉，继续努力！(划掉)

都结束了，这一刻还有点不舍，感谢老师感谢助教感谢室友感谢同学，也感谢自己，已经结束嘞！

难得相遇，终要别离；

江湖路远，后会有期！