浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	竺可桢学院
专　　业：	混合班
指导教师：	董亚波
报告日期：	2025 年 5 月 22 日

浙江大学实验报告

课程名称：数字逻辑设计 实验类型：综合

实验项目名称：计数器、定时器设计与应用

学生姓名：__ 学号：___同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2025 年 5 月 21 日

一、 实验目的

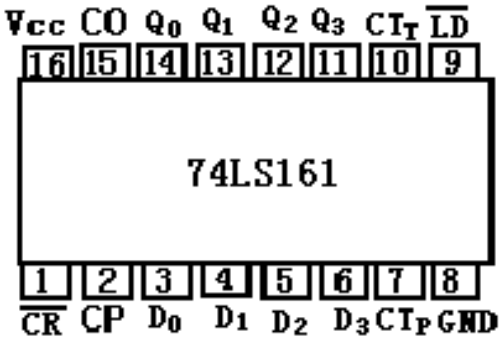
- 掌握同步四位二进制计数器 74LS161 的工作原理和设计方法
- 掌握时钟/定时器的工作原理与设计方法

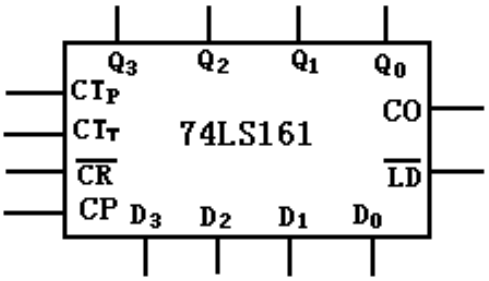
二、操作方法与实验步骤

1. 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
- 其功能描述如下：





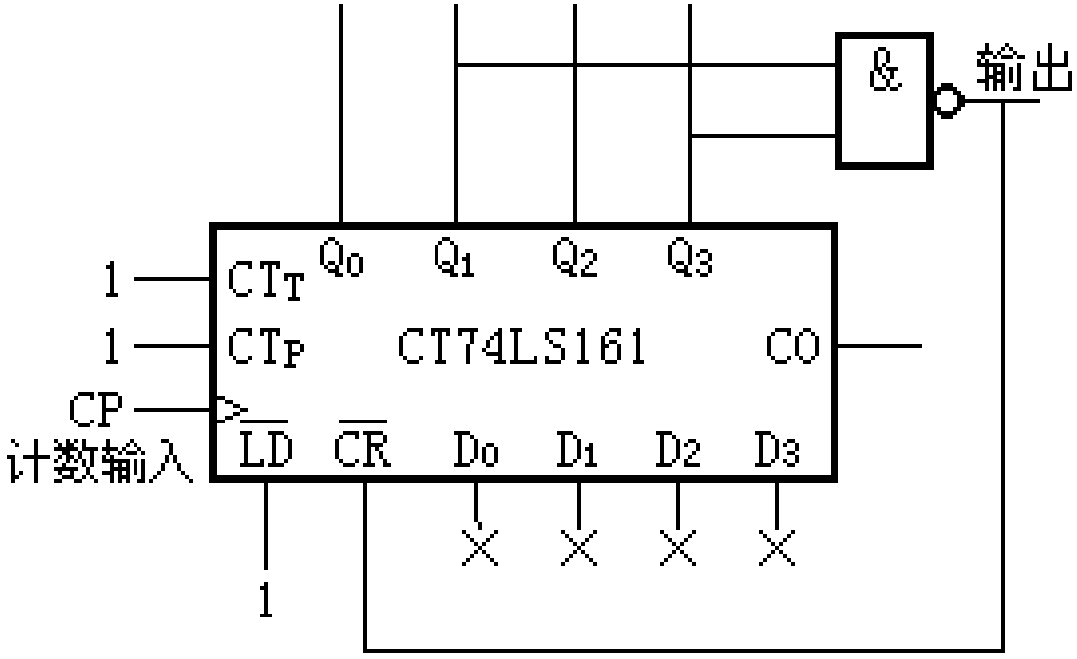
其中，CP 为时钟信号输入，D_0~D_3 为并行输入，Q_0~Q_3 为状态输出，CT_T 与 CT_P 为使能端，CR 为异步清零端，LD 为同步加载端，CO 为进位输出端。

- 其功能表如下：

输入									输出			
$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	CP	D_3	D_2	D_1	D_0	Q_3	Q_2	Q_1	Q_0
0	×	×	×	×	×	×	×	×	0	0	0	0
1	0	×	×	↑	d_3	d_2	d_1	d_0	d_3	d_2	d_1	d_0
1	1	0	1	×	×	×	×	×	保持			
1	1	×	0	×	×	×	×	×	保持			
1	1	1	1	↑	×	×	×	×	计数			

实现十进制计数器

- 利用与非门判断终止状态 1010
- 实现十进制计数 （0000 到 1001）
- 改变与非门的输入信号 ，可以实现其它进制计数。
- 改变与非门输出信号的功能和输入信号，可以实现同步加载
- 其原理图如图所示：



数字时钟

- 设计一个数字钟，使用 74LS161 模块，设计 60 进制和 24 进制计数器，实现 24 小时内时间的实时显示。
- 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。

2. 采用行为描述设计同步四位二进制计数器 74LS161

实验步骤如下：

- 在 vivado 中新建工程 My74LS161
- 新建源文件，用行为描述设计 My74LS161.v 文件，注意 CR 和 LD 均为低电平有效
- 设计 My74LS161.v 文件的仿真波形代码，进行行为仿真

3. 基于 74LS161 设计时钟应用

实验步骤如下：

- 在 vivado 中新建工程 MyClock
- 新建源文件 MyClock.v 文件
- 要求调用 My74LS161，设计时、分、秒计数器
- 同时调用实验 12 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
- 自行设计激励代码，对 MyClock.v 进行仿真，要求从 23:58:30 仿真到 00:00:02
- 在开发板上对其功能进行进一步验证

三、实验结果与分析

1. 采用行为描述设计同步四位二进制计数器 74LS161

按要求创建工程，需要新建文件 My74LS161.v，代码如下：

```
`timescale 1ns / 1ps
module My74LS161(
    input wire clk,
    input wire CRn,
    input wire LDn,
    input wire [3:0] D,
    input wire CTT,
    input wire CTP,
    output wire [3:0] Q,
    output wire CO
);
    reg [3:0] out = 4'b0;
    always @(posedge clk or negedge CRn) begin
        if(!CRn) begin
            out <= 0;
        end
        else begin
            if(!LDn) begin
                out <= D;
            end
```

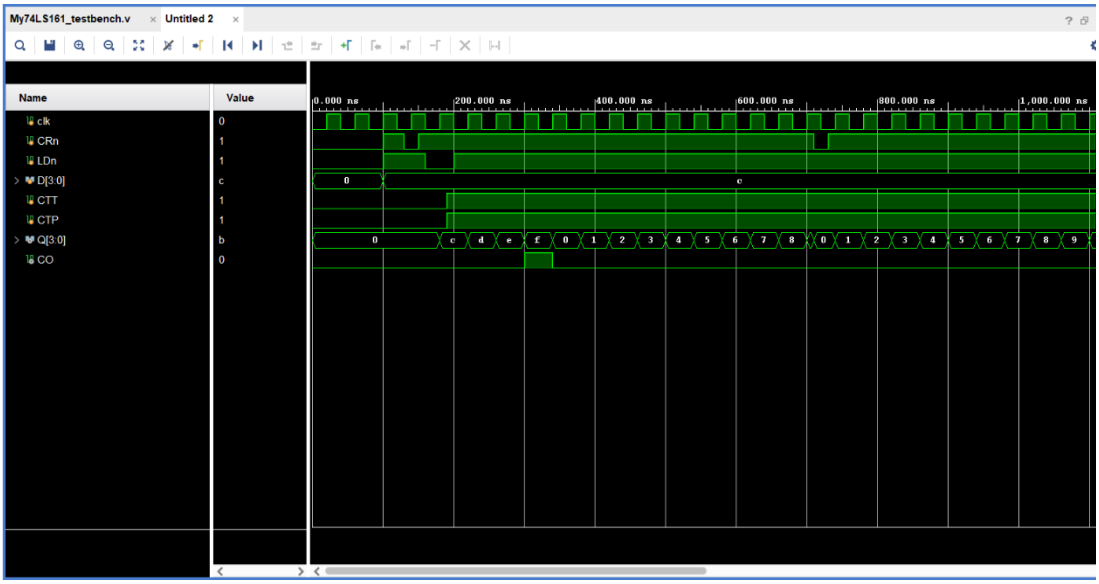
```
            else begin
                if(CTP && CTT) begin
                    out <= Q + 1;
                end
                else begin
                    out <= Q;
                end
            end
        end
    end
    assign Q = out;
    assign CO = (Q == 4'hF);
endmodule
```

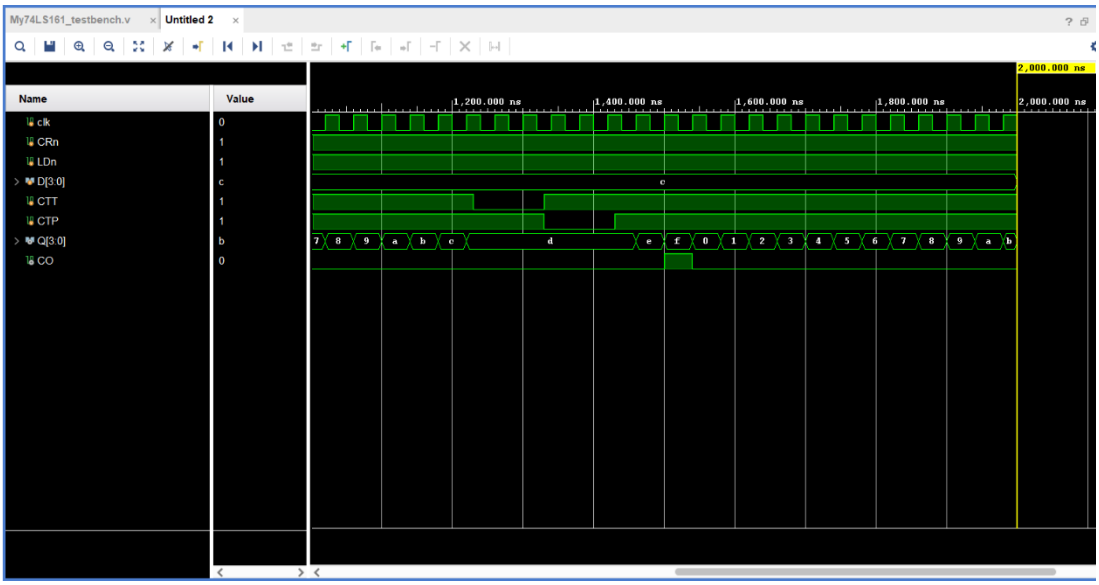
对 My74LS161.v 进行仿真，仿真代码如下：

```
`timescale 1ns / 1ps
module My74LS161_testbench();
    reg clk;
    reg CRn;
    reg LDn;
    reg [3:0] D;
    reg CTT;
    reg CTP;
    wire [3:0] Q;
    wire CO;
    My74LS161
uut(.clk(clk), .CRn(CRn), .LDn(LDn), .D(D), .CTT(CTT), .CTP(CTP), .Q(Q)
, .CO(CO));
    initial begin
        CRn = 0;
        D = 0;
        CTP = 0;
        CTT = 0;
        LDn = 0;

        #100;
        CRn = 1;
        LDn = 1;
        D = 4'b1100;
        CTT = 0;
        CTP = 0;
        #30 CRn = 0;
        #20 CRn = 1;
        #10 LDn = 0;
        #30 CTT = 1;
```

```
        CTP = 1;
        #10 LDn = 1;

        #510;
        CRn = 0;
        #20 CRn = 1;
        #500;
        CTT=0;
        #100;
        CTT=1;
        CTP=0;
        #100;
        CTP=1;
    end
    always begin
       clk=0;#20;
       clk=1;#20;
    end
endmodule
```

仿真波形如图所示（可能需要放大以看清数字）：

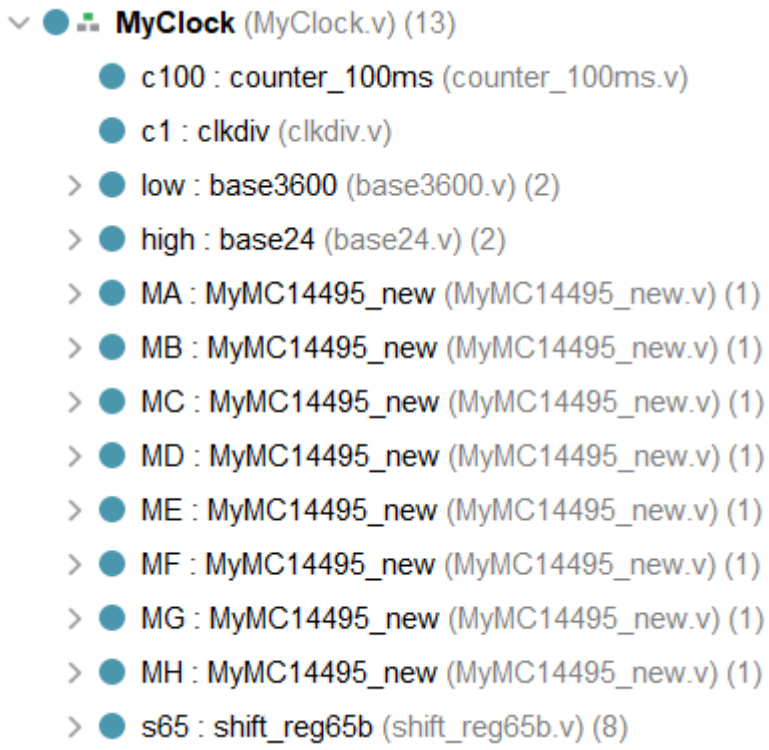




该仿真波形完成了同步加载、保持与异步清零，能正确实现 74LS161 的功能。

2. 基于 74LS161 设计时钟应用

按要求创建工程，需要添加之前的文件 clkdiv.v, MyMC14495.v, MyMC14495_new.v, shift_reg8b.v, shift_reg9b.v, shift_reg65b.v, clk_100ms.v（不再重复展示），并新建 base10.v, base24.v, base60.v, base3600.v（均用于封装计数器，因为后三者都需要依赖于十进制计数器）， MyClock.v。
文件结构如图所示：



base10.v 代码如下：

```
`timescale 1ns / 1ps
module base10(
```

```
    input wire clk,
    input wire CRn,
    input wire LDn,
    input wire [3:0] D,
    input wire CTT,
    output wire [3:0] Q,
    output wire CO
);
    wire Crn, crn, temp;
    assign Crn = CRn & ~(Q[3] & Q[1]);
    My74LS161
m0(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D), .CTT(CTT), .CTP(1'b1), .Q(Q)
,. CO(temp));
    assign CO = (Q == 4'b1001);
endmodule
```

base24.v 代码如下：

```
`timescale 1ns / 1ps
module base24(
    input wire clk,
    input wire CRn,
    input wire LDn,
    input wire [7:0] D,
    input wire CTT,
    output wire [7:0] Q,
    output wire CO
);
    wire Crn,tempa,tempb;
    assign Crn = CRn & ~(Q[5] & Q[2]);
    base10
low(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[3:0]), .CTT(CTT), .Q(Q[3:0]),
.CO(tempa));
    base10
high(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[7:4]), .CTT(CTT&tempa), .Q(Q
[7:4]), .CO(tempb));
    assign CO = (Q == 8'h23);
endmodule
```

base60.v 代码如下：

```
`timescale 1ns / 1ps
module base60(
    input wire clk,
    input wire CRn,
    input wire LDn,
    input wire [7:0] D,
    input wire CTT,
```

```
    output wire [7:0] Q,
    output wire CO
);
    wire Crn,tempa,tempb;
    assign Crn = CRn & ~(Q[6] & Q[5]);
    base10
low(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[3:0]), .CTT(CTT), .Q(Q[3:0]),
.CO(tempa));
    base10
high(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[7:4]), .CTT(CTT&tempa), .Q(Q
[7:4]), .CO(tempb));
    assign CO = (Q == 8'h59);
endmodule
```

base3600.v 代码如下：

```
`timescale 1ns / 1ps
module base3600(
    input wire clk,
    input wire CRn,
    input wire LDn,
    input wire [15:0] D,
    input wire CTT,
    output wire [15:0] Q,
    output wire CO
);
    wire Crn,tempa,tempb;
    assign Crn = CRn & ~(Q[14] & Q[13] & Q[6] & Q[5]);
    base60
low(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[7:0]), .CTT(CTT), .Q(Q[7:0]),
.CO(tempa));
    base60
high(.clk(clk), .CRn(Crn), .LDn(LDn), .D(D[15:8]), .CTT(CTT&tempa), .Q(
Q[15:8]), .CO(tempb));
    assign CO = (Q == 16'h5959);
endmodule
```

MyClock.v 代码如下：

```
`timescale 1ns / 1ps
module MyClock(
    input wire clk,
    input wire [15:0] SW,
    output wire SEG_CLK, SEG_DT, SEG_CLR, SEG_EN
);
    wire [31:0] num;
    wire [63:0] seg;
    wire [31:0] clk_div;
```

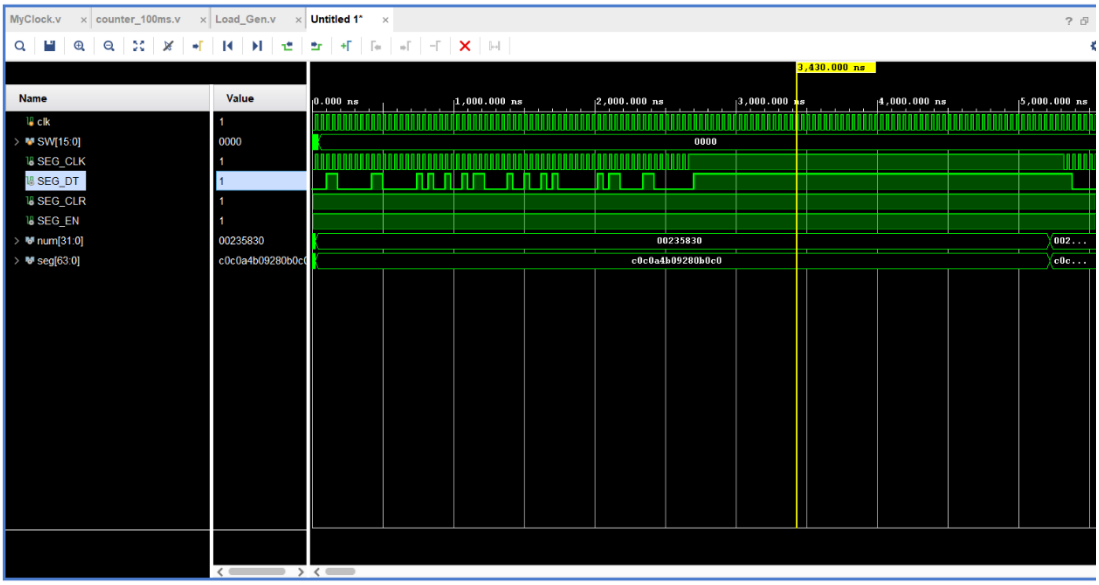
```
    wire [64:0] Q;
    wire finish;
    wire Load_S;
    wire [3:0] A, B, C, D, E, F, G, H;
    wire clk_100ms, CTh, CTd;
    counter_100ms c100(.clk(clk), .clk_100ms(clk_100ms));
    assign num={A,B,C,D,E,F,G,H};
    assign A = 0;
    assign B = 0;
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    base3600
low(.clk(clk_100ms), .CRn(1'b1), .LDn(~SW[15]), .D(16'h5830), .CTT(1'b1
), .Q({E,F,G,H}), .CO(CTh));
    base24
high(.clk(clk_100ms), .CRn(1'b1), .LDn(~SW[15]), .D(8'h23), .CTT(CTh),
.Q({C,D}), .CO(CTd));
//    Load_Gen
mS(.clk(clk), .clk_1ms(clk_div[6]), .btn_in(clk_100ms),.Load_out(Load_S
));
    MyMC14495_new MA(.D(A), .point(1'b0), .LE(1'b0), .s(seg[63:56]));
    MyMC14495_new MB(.D(B), .point(1'b0), .LE(1'b0), .s(seg[55:48]));
    MyMC14495_new MC(.D(C), .point(1'b0), .LE(1'b0), .s(seg[47:40]));
    MyMC14495_new MD(.D(D), .point(1'b0), .LE(1'b0), .s(seg[39:32]));
    MyMC14495_new ME(.D(E), .point(1'b0), .LE(1'b0), .s(seg[31:24]));
    MyMC14495_new MF(.D(F), .point(1'b0), .LE(1'b0), .s(seg[23:16]));
    MyMC14495_new MG(.D(G), .point(1'b0), .LE(1'b0), .s(seg[15:8]));
    MyMC14495_new MH(.D(H), .point(1'b0), .LE(1'b0), .s(seg[7:0]));
    shift_reg65b
s65(.S_L(clk_div[11]), .clk(clk), .s_in(1), .p_in({seg,1'b0}), .Q(Q));
    assign SEG_EN = 1'b1;
    assign SEG_CLR = 1'b1;
    assign SEG_DT = Q[64];
    assign finish = (~Q[63:0]) == 0;
    assign SEG_CLK = clk | finish;
endmodule
```

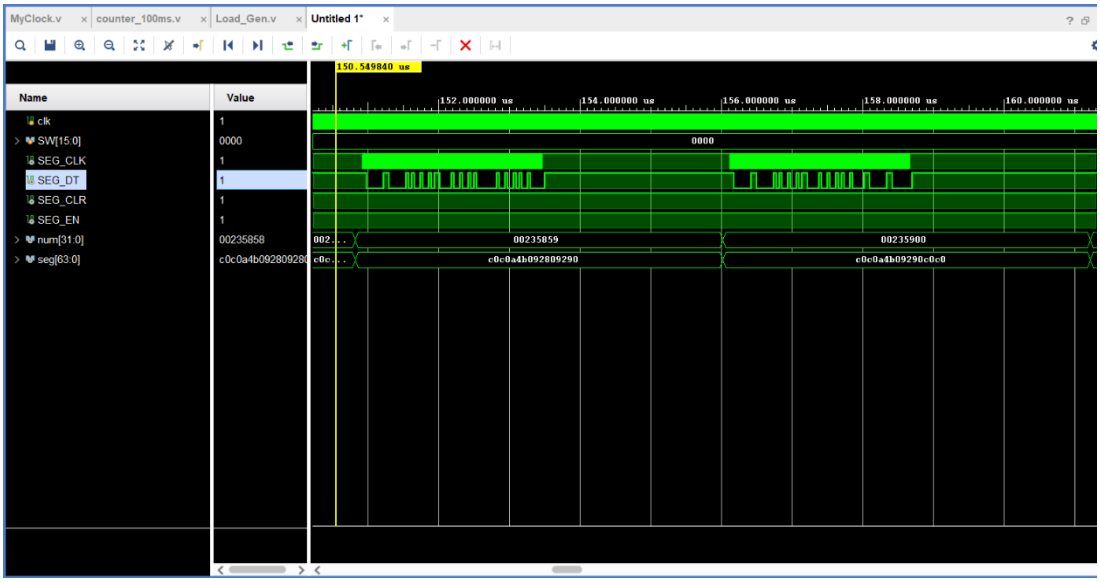
对 MyClock.v 进行仿真，仿真代码如下（需要改变 MyClock.v 中高延迟的部分）：

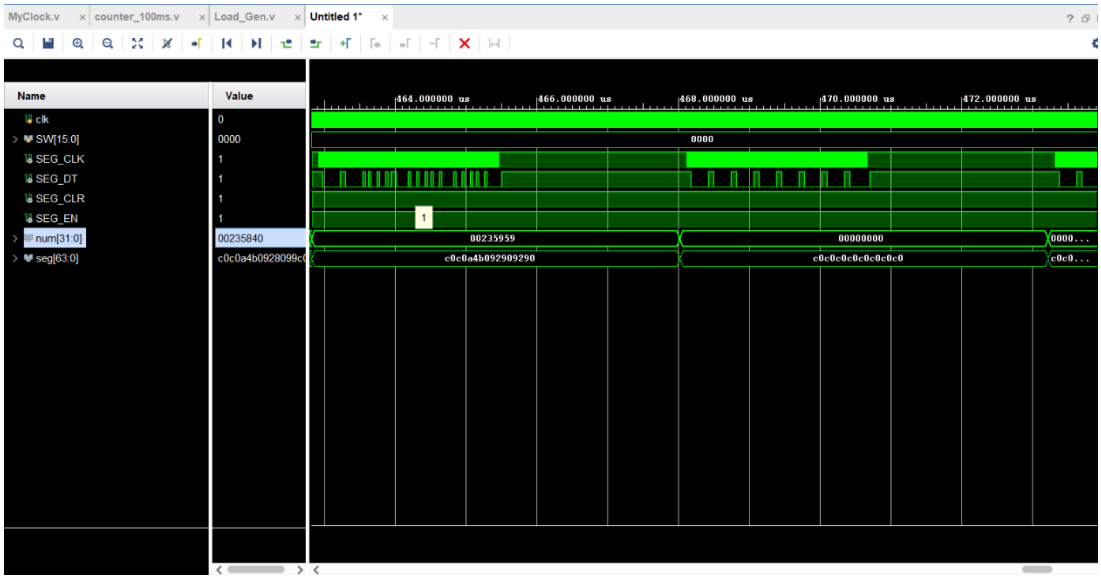
```
`timescale 1ns / 1ps
module MyClock_testbench();
    reg clk;
    reg [15:0] SW;
    wire SEG_CLK, SEG_DT, SEG_CLR, SEG_EN;
    MyClock
uut(.clk(clk), .SW(SW), .SEG_CLK(SEG_CLK), .SEG_DT(SEG_DT), .SEG_CLR(SE
G_CLR), .SEG_EN(SEG_EN));
```

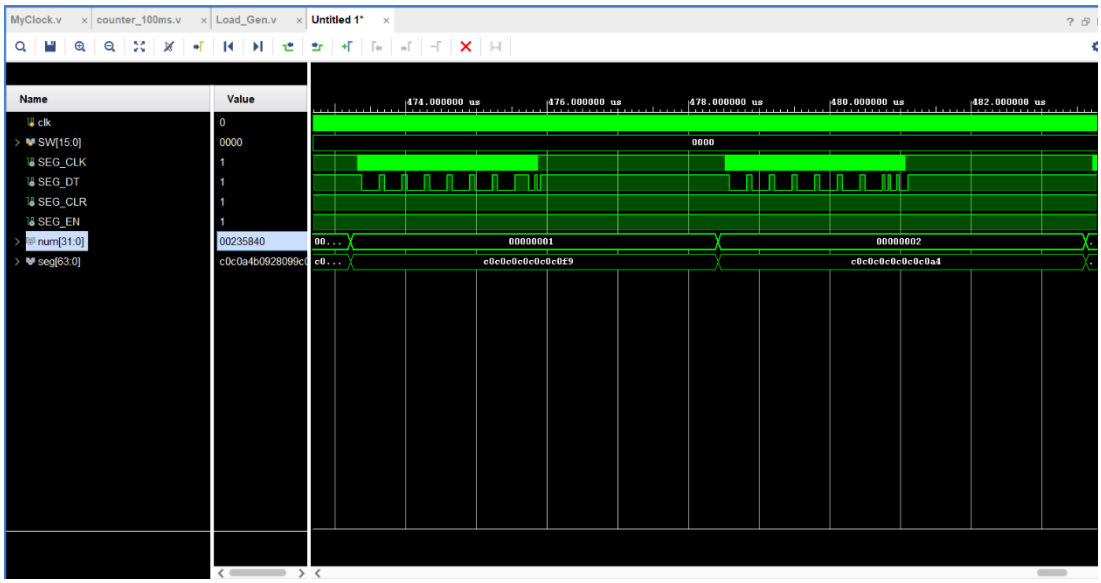
```
    initial begin
        SW[14:0]=0;
        SW[15]=1;
        #40;
        SW[15]=0;
    end
    always begin
        clk=0;#20;
        clk=1;#20;
    end
endmodule
```

仿真波形如图所示（只截取开头、23:59:00 所在段以及 00:00:02 所在段，也可能需要将图片放大）：









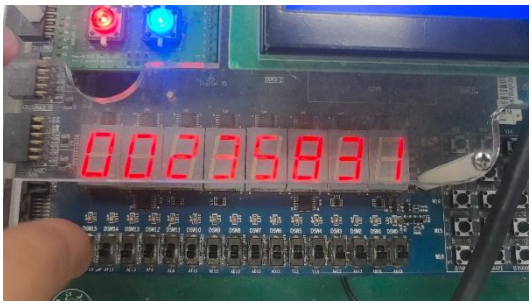
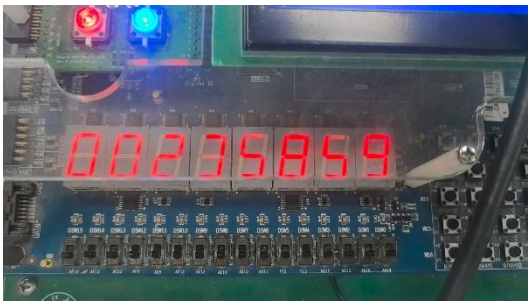
可以看到，这一段波形中，num 可以正常完成计时功能，即分、秒为 60 进一，时为 24 进一，而 seg 则表示了数码管的应该的显示情况，这在 SEG_DT 上也能体现，然而仿真和上板的区别还是相当大的，故需要上板验证（关于其区别我会在讨论、心得中指出）。

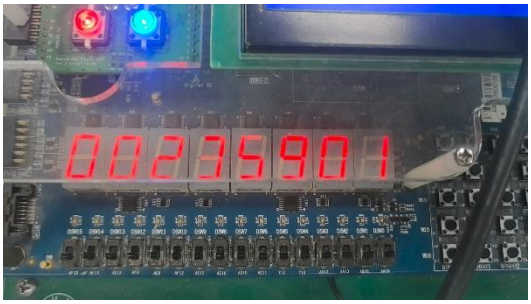
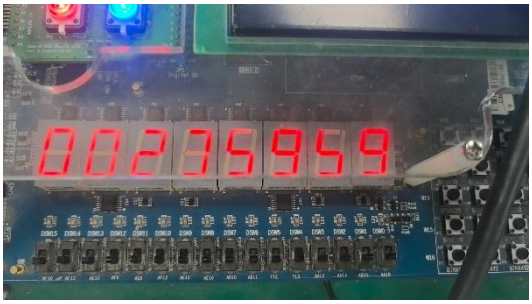
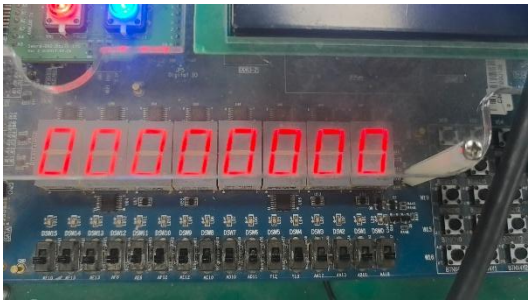
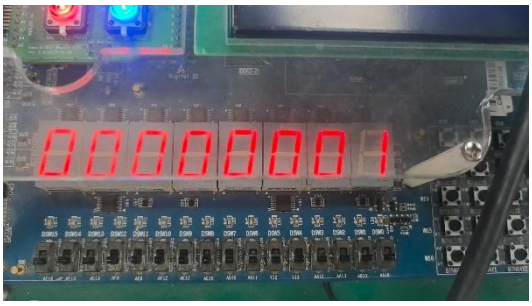
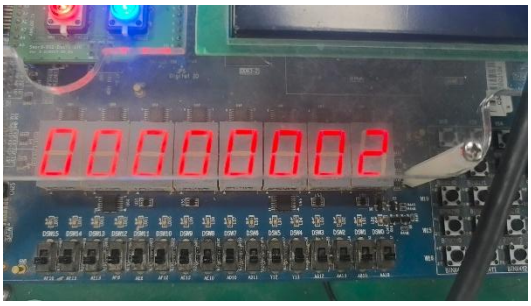
将其下载到板上进行验证，约束文件如下：

```
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]
```

```
# Switch
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]

# 7-Segment Serial
set_property PACKAGE_PIN M24 [get_ports SEG_CLK]
set_property PACKAGE_PIN M20 [get_ports SEG_CLR]
set_property PACKAGE_PIN L24 [get_ports SEG_DT]
set_property PACKAGE_PIN R18 [get_ports SEG_EN]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_CLK]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_CLR]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_DT]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_EN]
```

上板结果如图所示（主要体现同步加载以及临界值状态）：

可以看到，随着时间的推移，数码管上的数字确实能够起到计时的作用，即每过一段时间秒数增加一，且分、秒为 60 进制，时为 24 进制。

四、讨论、心得

这一次实验中，我学会了计数器的设计及其应用——数字时钟。在本次实验中，我实际上受到了实验板上很大的困扰，即仿真结果无误，但一旦上板，就会出现各式各样的问题——2,3,5 会有一段数码管不亮；2,5 显示成了 8；小数点在疯狂闪烁；数字会在错误的地方进位，等等。这些都无法很好地找出根治的方法，且随着数码管刷新频率的变化，出现的问题也会变化，我因为这一个问题花费了

差不多三个小时，最后终于找到一个比较满意的结果（刷新频率为 2048 个时钟周期）。即使这样，当仔细观察的时候，仍然会发现 3,5 的某一位数码管比较淡，而当刷新频率改为其他 2 的整数次幂的时候，要么数字显示不全，要么数字变成 8 了，也很难再找到一个效果更好的了。我也是通过这一次实验，非常深刻的理解到硬件相较于软件的不同之处——其出现的错误是可以没有任何逻辑的。我猜测其问题发生的原因与数码管的延迟、经过某一位的使能信号数量有关系，也许当前两个数字显示的是 8 的时候，情况会好一些，但是由于时间原因，我也没办法再进行验证了。

再讲讲我对这个学期数逻实验课的感受。通过本学期实验课的学习，我从零开始学会了 verilog 的基本语法，同时也能够使用 verilog 语言实现简单的功能。而对于课上的内容，实际上，我希望能提供一个封装过的模板，然后在比较关键的地方进行挖空，这样在保证能学到核心部分的同时，尽量减少次要部分由写法差异带来的困扰。而在试验过程中，老师和助教都很有耐心，即使实验做到了七八点钟，都还会在教室进行指导，本人还是挺满意的。