浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	熊子宇
学　　院：	竺可桢学院
专　　业：	混合班
邮　　箱：	3200105278@zju.edu.cn
QQ　　号：	934215855
电　　话：	18255168425
指导教师：	洪奇军
报告日期：	2022 年 01 月 05 日

实验 11 寄存器和寄存器传输设计

课程名称：逻辑与计算机设计基础实验 实验类型：综合

实验项目名称：实验 11 寄存器和寄存器传输设计

学生姓名：熊子宇 学号：3200105278 同组学生姓名：薛婧

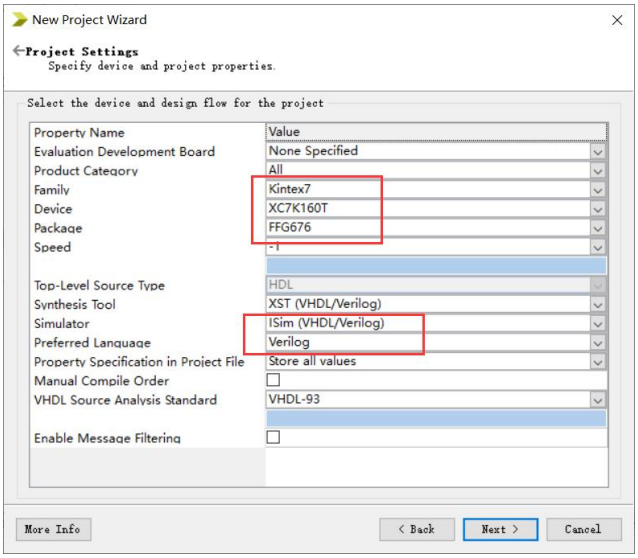
实验地点：紫金港东四 509 室 实验日期：2021 年 12 月 16 日

一、操作方法与实验步骤

1 基于 ALU 的数据传输应用设计

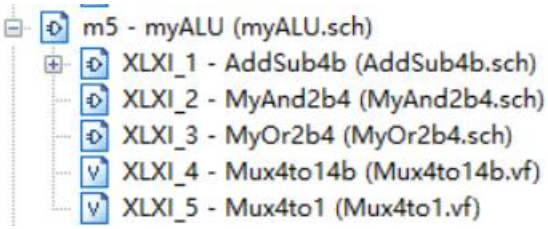
1.1 建立 MyALUTrans 工程

点击主菜单上的 New Project 进行如下设置,点击 Finish 完成 Project 创建



1.2 调用 ALU 模块

将实验 8 设计的 ALU 模块，包括 myALU, AddSub4b, MyAnd2b4, MyOr2b4, Mux4to14b 和 Mux4to1 拷贝到本工程中，如下图。



1.3 调用 pbdebounce 按键去抖动模块

时钟计数分频器 clkdiv.v 代码如下：

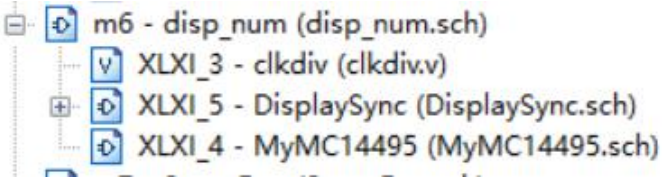
```
module clkdiv(input clk, input rst, output reg[31:0]clkdiv);

always @ (posedge clk or posedge rst) begin
    if (rst) clkdiv<=0;
    else clkdiv <= clkdiv + 1'b1;
end

endmodule
```

去抖动模块 pbdebounce.v 代码如下：

```
module pbdebounce(
    input wire clk_1ms,
    input wire button,
    output reg pbreg
    );

    reg [7:0] pbshift;

    always@(posedge clk_1ms) begin
        pbshift=pbshift<<1;
        pbshift[0]=button;
        if (pbshift==8'b0)
            pbreg=0;
        if (pbshift==8'hFF)
            pbreg=1;
    end
endmodule
```

1.4 调用 DispNum 模块

将以往实验实现的七段数码管显示模块 disp_num.sch 以及相关文件拷贝到本工程中。



1.5 设计按键数据输入 CreateNumber 模块

新建 verilog 文件 CreateNumber.v 并输入以下代码：

```
module CreateNumber (
    input wire [3:0] btn_out,
    input wire [15:0] SW,
    input wire [3:0] C1,
    output reg [15:0] num
    );
    wire [3:0] A1,B1,Result;
    wire [3:0] A2,B2,C2;

    initial num <= 16'b1010_1011_1100_1101;

    AddSub4b a1(.A(num[3:0]), .B(4'b0001), .Ctrl(SW[0]), .S(A1));
    AddSub4b a2(.A(num[7:4]), .B(4'b0001), .Ctrl(SW[1]), .S(B1));
```

```
    Mux4to14b m4(.I0(num[3:0]),.I1(num[7:4]),.I2(num[11:8]),
                 .I3(4'b0),.s(SW[8:7]),.o(Result[3:0]));

    assign A2 = (SW[15]==1'b0)?A1:Result;
    assign B2 = (SW[15]==1'b0)?B1:Result;
   assign C2 = (SW[15]==1'b0)?C1:Result;
    always@(posedge btn_out[0]) num[3:0] <= A2;
    always@(posedge btn_out[1]) num[7:4] <= B2;
    always@(posedge btn_out[2]) num[11:8] <= C2;

endmodule
```

1.6 新建源文件 top，并右键设为“Top Module”

新建 Verilog 文件 top.v 并输入以下代码。

```
module top(
    input wire clk,
   input wire [2:0]BTN,//按钮控制自增/自减（mode0），或者是选择寄存器（mode1）
   input wire [15:0]SW,
    //sw[15] 选择 mode, 0 为 mode0,1 为 mode1 ；
    //sw[0]控制 A 加/减； sw[1]控制 B 加/减；
    //sw[6:5]控制 ALU 运算：00-加，01-减，10-与，11-或
    //sw[8:7]对应 SelectBus（mode1）：00-选择 A，01-选择 B，10-选择 C
    output wire [3:0]AN,
    output wire [7:0]SEGMENT,
   output wire BTNX4
    );

    wire [15:0] num;//num[3:0] A, num[7:4] B, num[11:8] C, num[15:12]
result
    wire [2:0] btn_out;
    wire C0;
    wire [31:0] clk_div;
    wire [3:0] C;
   assign BTNX4 = 1'b0;
    pbdebounce m0(clk_div[17],BTN[0],btn_out[0]);//BTN[0]去抖动
   pbdebounce m1(clk_div[17],BTN[1],btn_out[1]);
    pbdebounce m2(clk_div[17],BTN[2],btn_out[2]);
    clkdiv m3(clk,0,clk_div);

    CreateNumber m4(btn_out, SW, C, num);
    myALU m5(.S(SW[6:5]), .A(num[3:0]), .B(num[7:4]), .C(C), .C0(C0));

    disp_num
m6(.clk(clk), .HEXS({num[3:0],num[7:4],C,num[11:8]}), .LES(4'b????), .
points(4'b1111),
                    .RST(1'b0), .AN(AN), .Segment(SEGMENT));
endmodule
```

1.7 建立用户时序约束并为模块的端口指定引脚分配

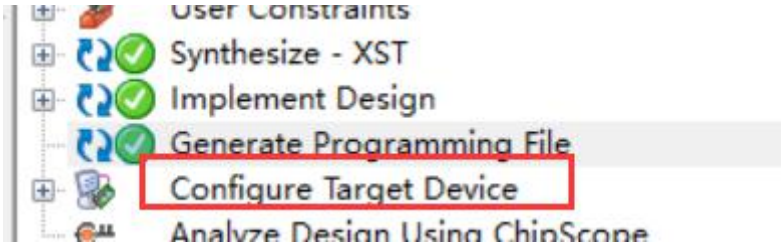
建立引脚分配文件并输入代码：

```
NET"SW[0]"LOC=AA10  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[1]"LOC=AB10  | IOSTANDARD=LVCMOS15 ;
NET"SW[2]"LOC=AA13  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[3]"LOC=AA12  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[4]"LOC=Y13  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[5]"LOC=Y12  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[6]"LOC=AD11  | IOSTANDARD=LVCMOS15 ;#D1
```

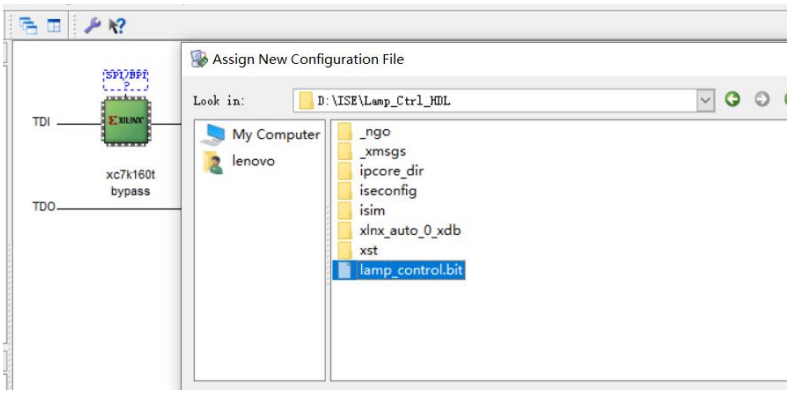
```
NET"SW[7]"LOC=AD10  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[8]"LOC=AE10  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[9]"LOC=AE12  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[10]"LOC=AF12  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[11]"LOC=AE8  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[12]"LOC=AF8  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[13]"LOC=AE13  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[14]"LOC=AF13  | IOSTANDARD=LVCMOS15 ;#D1
NET"SW[15]"LOC=AF10  | IOSTANDARD=LVCMOS15 ;#D1

NET"clk"LOC=AC18  | IOSTANDARD=LVCMOS18 ;
NET "btn[0]" LOC = W14  | IOSTANDARD = LVCMOS18 ;
NET "btn[0]" clock_dedicated_route = false;
NET "btn[1]" LOC = V14  | IOSTANDARD = LVCMOS18 ;
NET "btn[1]" clock_dedicated_route = false;
NET "btn[2]" LOC = V19  | IOSTANDARD = LVCMOS18 ;#SW[14]
NET "btn[2]" clock_dedicated_route = false;
NET "BTNX4" LOC = W16  | IOSTANDARD = LVCMOS18 ;#SW[15]

NET"SEGMENT[0]"LOC=AB22 | IOSTANDARD=LVCMOS33;#a
NET"SEGMENT[1]"LOC=AD24 | IOSTANDARD=LVCMOS33;#b
NET"SEGMENT[2]"LOC=AD23 | IOSTANDARD=LVCMOS33;#c
NET"SEGMENT[3]"LOC=Y21  | IOSTANDARD=LVCMOS33;#d
NET"SEGMENT[4]"LOC=W20  | IOSTANDARD=LVCMOS33;#e
NET"SEGMENT[5]"LOC=AC24 | IOSTANDARD=LVCMOS33;#f
NET"SEGMENT[6]"LOC=AC23 | IOSTANDARD=LVCMOS33;#g
NET"SEGMENT[7]"LOC=AA22 | IOSTANDARD=LVCMOS33;#point

NET"AN[0]"LOC=AD21 | IOSTANDARD=LVCMOS33;
NET"AN[1]"LOC=AC21 | IOSTANDARD=LVCMOS33;
NET"AN[2]"LOC=AB21 | IOSTANDARD=LVCMOS33;
NET"AN[3]"LOC=AC22 | IOSTANDARD=LVCMOS33;
```

1.8 下载到 SWORD 板

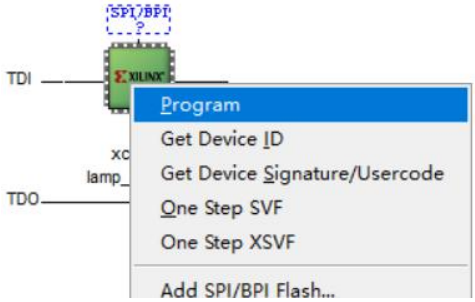
（1）通过 Synthesize - XST, Implement Design, Generate Programming File 后，点击 Configure Target Device



（2）载入 bit 文件（此处不是本工程的 bit 文件，仅为示意）



（3）点击 Program 下载至 SWORD 板



1.9 根据真值表，操作实验板，验证功能

二、实验结果与分析

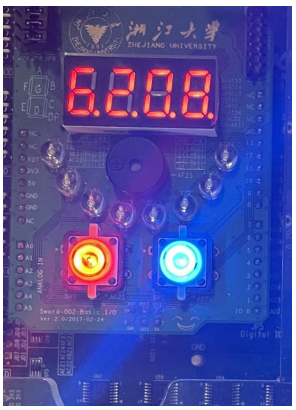
1 Mode 0　ALU 运算输出控制

sw[15] =0 为 Mode0，即 ALU 运算输出控制。

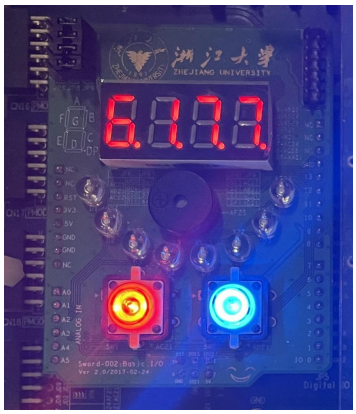
sw[0]控制 A 加/减； sw[1]控制 B 加/减；按键 btn[0]和 btn[1]可以对 A 和 B 进行自增或自减。

sw[6:5]控制 ALU 运算：00-加，01-减，10-与，11-或。

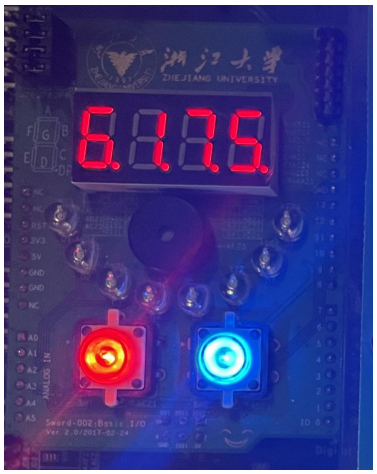
按下 btn[2]可以将运算结果赋给寄存器 C。



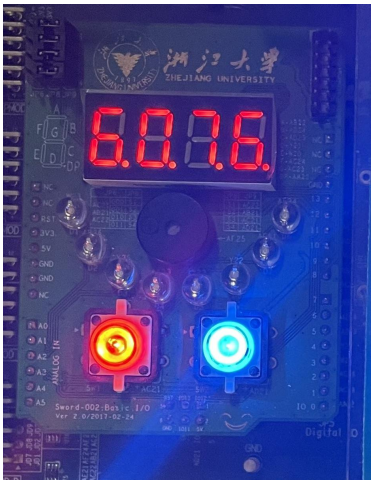
sw[15]=0, sw[6:5]=00
加法模式，6+2=8



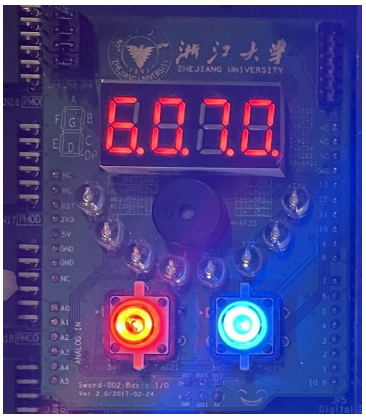
sw[15]=0,sw[6:5]=00,加法模式。
sw[1]=1,按下 btn[1]使 B 自减。
按下 btn[2]，使运算结果 7 赋值给 C



sw[15]=0, sw[6:5]=01
减法模式，6-1=5



sw[15]=0, sw[6:5]=11
或模式，0110 |0000 = 0110



sw[15]=0, sw[6:5]=10
与模式，0110 & 0000 = 0000

2 Mode1 数据传输控制

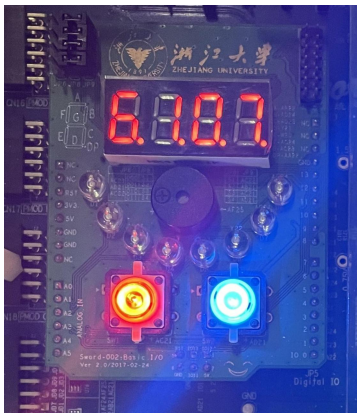
sw[15] =1 为 Mode1，即数据传输控制。

sw[8:7]对应向总线上存放的数据来源：

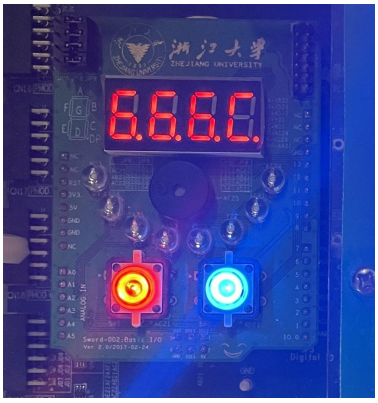
00-总线上的数据选择 A，按 BTX4Y0 存储到 num[3:0]

01-总线数据上的选择 B，按 BTX4Y1 存储到 num[7:4]

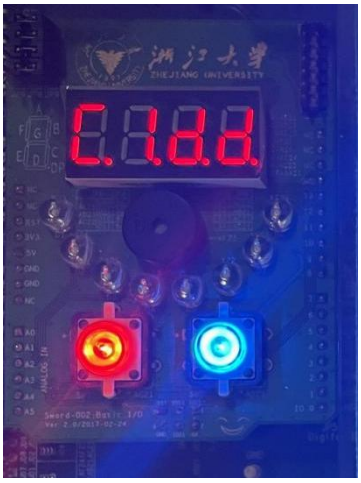
10-总线数据上的选择 C。按 BTX4Y2 存储到 num[11:8]



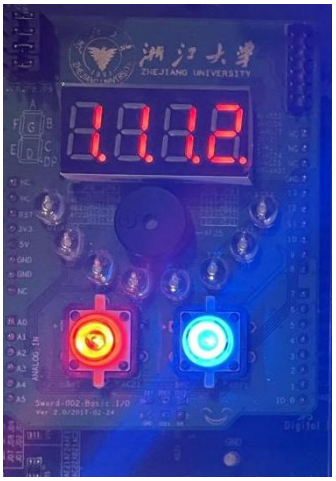
初始状态



sw[8:7]=00，总线选择 A，
分别按下 btn[0], btn[1], btn[2]
存储到第一、第二、第三个七段数码管处

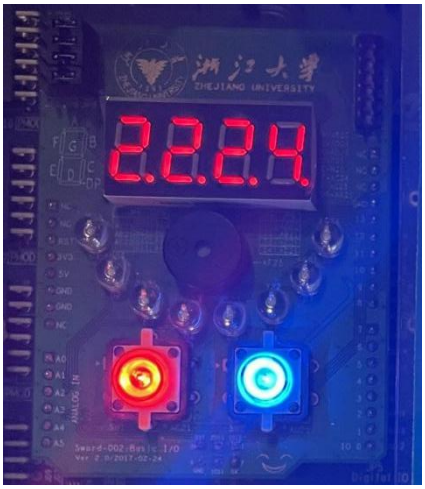


初始状态



sw[8:7]=01，总线选择 B，
分别按下 btn[0], btn[1], btn[2]
存储到第一、第二、第三个七段数码管处

初始状态



sw[8:7]=10，总线选择 C，
分别按下 btn[0], btn[1], btn[2]
存储到第一、第二、第三个七段数码管处

三、讨论、心得

本次实验是第一个用 Verilog 语言设计 top 结构的实验，具有较大的挑战性，对我造成了不小的困难。

实验内容比较简单，除去 top.v 以外，显示模块、ALU 模块和去抖动模块在以往实验中都已经有完善的模块代码，直接引入即可。最重要的是理清两种 mode 以及 sw 控制之间的逻辑关系，方可撰写赋值语句。特别核心的模块是 CreateNumber.v，根据 sw[15]，将 ALU 计算的结果或总线上的数据赋值给显示模块要用的 num[15:0]。

特别困难的部分在于，由于没有系统学习过 Verilog 语言，目前我只能看懂代码，但是自己撰写代码的时候会有很多语法错误，而且我对 wire 和 reg 的特性并不熟悉，不知道何时才能使用 always@语句以及<=语句。在写代码过程中，我需要模仿以往实验的顶层模块代码，以及查阅 Verilog 语言的指导手册，才能够消除很多错误。

通过本次实验，我进一步熟悉了 Verilog 语言的语法特性，能够独立撰写简单的 Verilog 代码。此外，对寄存器的传输结构理论我也有了更深的认识。

实验 12 计数器、定时器设计与应用

课程名称：逻辑与计算机设计基础实验 实验类型：综合

实验项目名称：实验 12 计数器、定时器设计与应用

学生姓名：熊子宇 学号：3200105278 同组学生姓名：薛婧

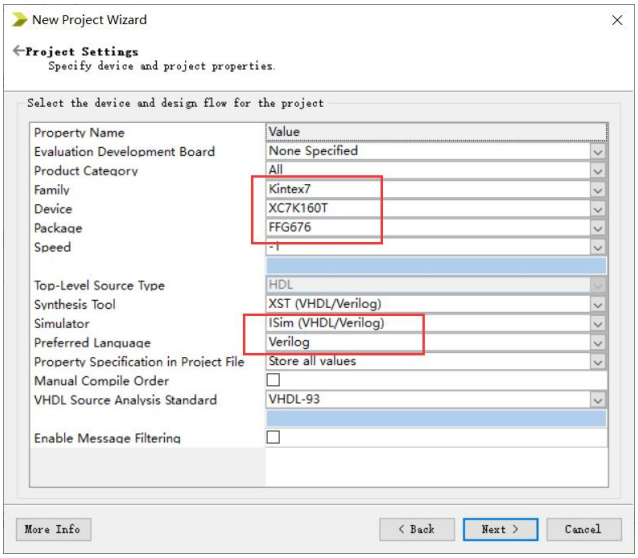
实验地点：紫金港东四 509 室 实验日期：2021 年 12 月 16 日

一、操作方法与实验步骤

1 采用行为描述设计同步四位二进制计数器 74LS161

1.1 建立 My74LS161 工程

点击主菜单上的 New Project 进行如下设置,点击 Finish 完成 Project 创建



1.2 行为描述方式设计同步四位二进制计数器 74LS161

新建 Verilog 文件 my74LS161.v 并输入以下代码。

```
module my74LS161(
        input wire CP,
        input wire CR,
        input wire [3:0] D,
        input wire CTP,
        input wire CTT,
        input wire Ld,
        output reg [3:0] Q,
        output reg CO
    );
        initial CO <= 0;
        always @ (posedge CP or negedge CR) begin
```

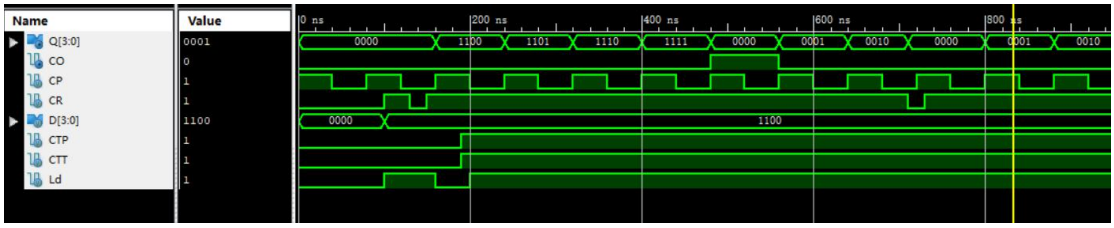
```
            if (CR==0) Q[3:0] <= 4'b0000; //不能用 negedge CR 作为条件
            else if (Ld==0) Q[3:0] <= D[3:0];
                else begin
                    if (CTP == 1 && CTT == 1) begin
                        if (Q[3:0] == 4'b1111)
                            begin
                            CO<=1;
                            Q[3:0] <=4'b0;
                            end
                        else
                            begin
                            CO<=0;
                            Q[3:0] <= Q[3:0]+1'b1;
                            end
                    end
                end
        end
endmodule
```

1.3 仿真激励输入

建立基准测试波形文件并输入如下代码

```
initial begin
        CP = 1;
        CR = 0;
        D = 0;
        CTP = 0;
        CTT = 0;
        Ld = 0;
        #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;
    end
    always begin
        #40 CP = 0;
        #40 CP = 1;
        end
```

查看波形



2 基于 74LS161 设计时钟应用

2.1 建立 MyClock 工程

点击主菜单上的 New Project 进行设置,点击 Finish 完成 Project 创建

2.2 调用 74LS161 模块

将实验 8 设计的 ALU 模块，包括 myALU, AddSub4b, MyAnd2b4, MyOr2b4, Mux4to14b 和 Mux4to1 拷贝到本工程中，如下图。使用 Add Copy of Source 将 my74LS161.v 拷贝至本工程中。如图所示：

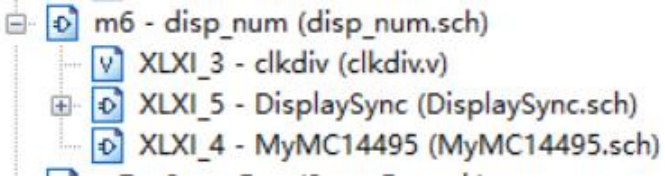
2.3 调用分频模块，用 100ms 作为分的驱动时钟

调用实验 10 使用的分频模块 clk_100ms.v，代码如下：

```
module counter_100ms(clk, clk_100ms);
    input wire clk;
    output reg clk_100ms;
    reg [31:0] cnt;
    always @ (posedge clk) begin
        if (cnt < 500_000_0) begin
             cnt <= cnt + 1;
        end else begin
             cnt <= 0;
             clk_100ms <= ~clk_100ms;
        end
    end
endmodule
```

2.4 调用 DispNum 模块

将以往实验实现的七段数码管显示模块 disp_num.sch 以及相关文件拷贝到本工程中。



2.5 实现 Sword 板 7 段数码管显示 Hex827Seg

将 SSeg7_Dev_IO.v 和.ngc 文件拷贝到本工程中。SSeg7_Dev_IO.v 代码如下：

```
module  SSeg7_Dev(input clk,        //  时钟
```

```
                    input rst,         //复位
                    input Start,       //串行扫描启动
                    input SW0,         //文本(16 进制)/图型(点阵)切换
                    input flash,       //七段码闪烁频率
                    input[31:0]Hexs,   //32 位待显示输入数据
                    input[7:0]point,   //七段码小数点：8 个
                    input[7:0]LES,         //七段码使能：=1 时闪烁
                    output seg_clk,    //串行移位时钟
                    output seg_sout,   //七段显示数据(串行输出)
                    output SEG_PEN,    //七段码显示刷新使能
                    output seg_clrn    //七段码显示汪零
                    );

endmodule
```

2.6 新建源文件 top，并右键设为“Top Module”

新建 Verilog 文件 top.v 并输入以下代码。

```
module top(input wire clk,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire [7:0] LED,
    output LEDEN,
    output seg_clk,
    output seg_sout,
    output SEG_PEN,
    output seg_clrn
);
    wire [15:0] cnt;
    wire [31:0] div;
    counter_100ms m1(clk,clk_100ms);

    my74LS161 a0  (.CR(1'b1),
.Ld(~(cnt[3] & cnt[0])),
.CTT(1'b1), .CTP(1'b1), .CP(clk_100ms), .D(4'b0), .Q(cnt[3:0]));

    my74LS161 a1(.CR(1'b1), .Ld(~(cnt[6] & cnt[4] & cnt[3] & cnt[0])),
.CTT(cnt[3]&cnt[0]), .CTP(1'b1), .CP(clk_100ms), .D(4'b0), .Q(cnt[7:4])
);

    my74LS161 a2  (.CR(1'b1),
.Ld(~( (cnt[11] & cnt[8]) || (cnt[13] & cnt[9] & cnt[8] & cnt[6] & cnt[4]
& cnt[3] & cnt[0]) )),
.CTT(cnt[6] & cnt[4] & cnt[3] & cnt[0]),
.CTP(1'b1), .CP(clk_100ms), .D(4'b0), .Q(cnt[11:8]));

    my74LS161 a3(.CR(1'b1), .Ld(~(cnt[13] & cnt[9] & cnt[8] & cnt[6] &
cnt[4] & cnt[3] & cnt[0])), .CTT(cnt[11] & cnt[8]),
.CTP(1'b1), .CP(clk_100ms), .D(4'b0), .Q(cnt[15:12]));

    disp_num
m2(.clk(clk), .HEXS({cnt[15:12],cnt[11:8],cnt[3:0],cnt[7:4]}), .LES(4'
b0000), .points(4'b0000),
                    .RST(1'b0), .AN(AN), .Segment(SEGMENT));

    clkdiv m3(.clk(clk), .rst(1'b0), .clkdiv(div[31:0]));

    SSeg7_Dev m4(.clk(clk), .rst(1'b0), .Start(div[20]), .SW0(1'b1),
```

```
    .flash(1'b1), .Hexs({12'h000,cnt[15:12],cnt[11:8],4'h0,cnt[7:4],cn
t[3:0]}), .point(8'b0010_0100),
    .LES(8'b00_1_00_1_00), .seg_clk(seg_clk), .seg_sout(seg_sout),
    .SEG_PEN(SEG_PEN), .seg_clrn(seg_clrn));
endmodule
```

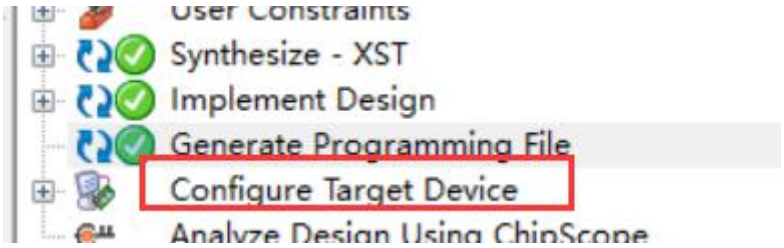
2.7 建立用户时序约束并为模块的端口指定引脚分配

建立引脚分配文件并输入代码：

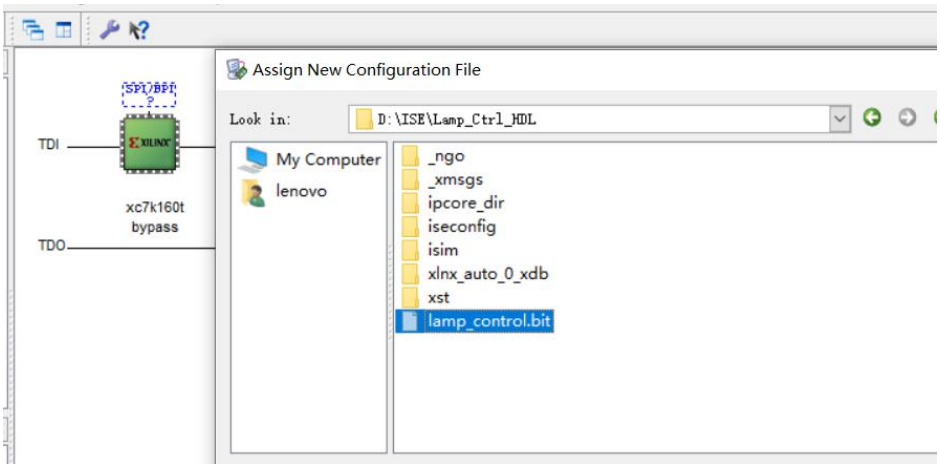
```
NET "clk"  LOC = AC18 | IOSTANDARD = LVCMOS18;
NET "clk"  TNM_NET = TM_CLK;
TIMESPEC TS_CLK_100M = PERIOD "TM_CLK" 10ns HIGH 50%;

NET "seg_clk"    LOC = M24 | IOSTANDARD = LVCMOS33;
NET "seg_clrn"   LOC = M20 | IOSTANDARD = LVCMOS33;
NET "seg_sout"   LOC = L24 | IOSTANDARD = LVCMOS33;
NET "SEG_PEN"    LOC = R18 | IOSTANDARD = LVCMOS33;

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;
NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
```

2.8 下载到 SWORD 板

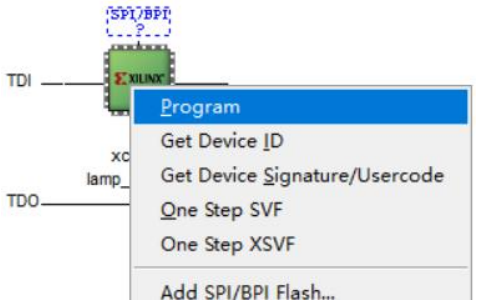
（1）通过 Synthesize - XST, Implement Design, Generate Programming File 后，点击 Configure Target Device



（2）载入 bit 文件（此处不是本工程的 bit 文件，仅为示意）



（3）点击 Program 下载至 SWORD 板

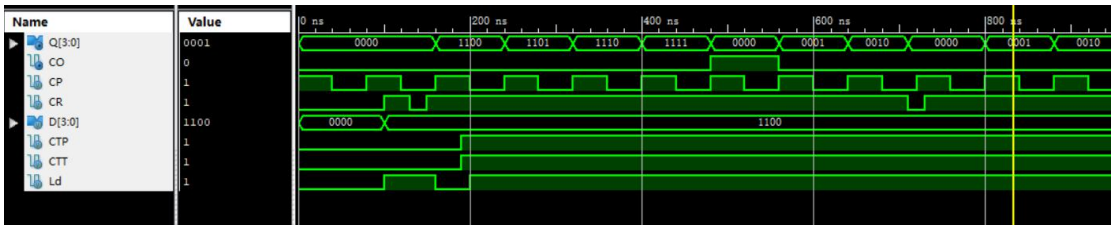


2.9 根据真值表，操作实验板，验证功能

二、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

仿真激励波形分析



分析：

基本 SR 锁存器的逻辑功能如下表所示。

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		

如图所示，CP 为时钟。

第一个时钟周期，~CR=0，Q[3:0]被初始化为 0000。

第二个时钟周期，~CR 在非时钟正边沿处变为 1，由于 74LS161 计数器是正边沿触发，所以计数器无法进行并行载入，Q[3:0]仍然为 0000。

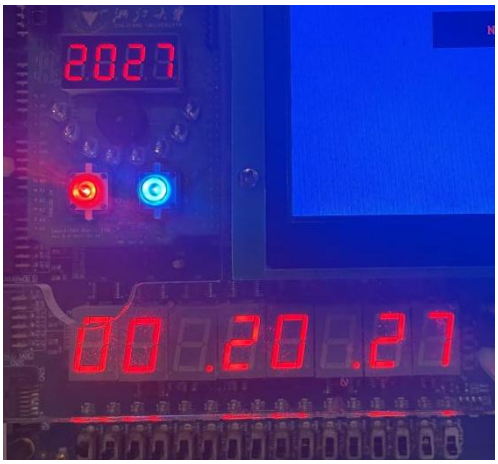
第三个时钟周期正边沿处，~CR=1,~LD=0，计数器为并行载入状态，此时 D[3:0]=1100 被赋给 Q[3:0]，从而 Q[3:0]=1100。

第四个时钟周期及以后，~CR=1, ~LD=1, CTP=1, CTT=1。每逢时钟正边沿，计数器处于计数状态。Q[3:0]不断自增。当 Q[3:0]=1111 再自增时产生进位，则 Q[3:0]=0000,而在该时钟周期内 CO=1。

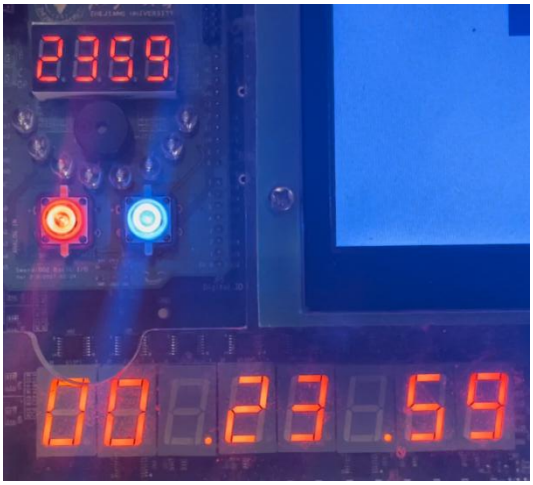
当~CR=0 时，计数器被异步地重置为 0000。当~CR 恢复为 1 后，在下一个时钟周期正边沿时 Q[3:0]从 0000 开始继续计数。

2 基于 74LS161 设计时钟应用

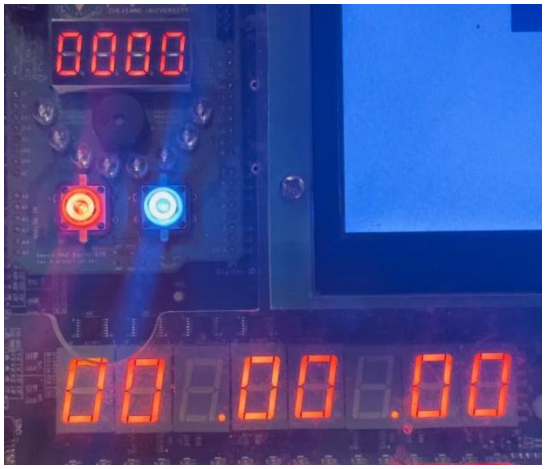
计时显示 02:43



计时显示 20:27



计时显示 23:59



显示 23:59 下一秒后显示 00:00

三、讨论、心得

本次实验是计数器的设计与应用。计数器是寄存器的重要应用之一，在理论课上被详细分析过。而在实验课程中，通过亲自设计时钟，我更好地掌握了寄存器、计数器的相关原理。尤其是设计 24 进制、60 进制等非常规进制的过程，更能让我体会计数器同步 Reset 的过程。

本次实验的要求是用 Verilog 代码描述设计计数器的基本单元，再用该模块继续设计时钟。在实验 11 的基础上，我在代码撰写的熟练程度上有了明显的提升。显示模块在以往实验的基础上修改即可。真正的难点是在非常规进制进位的控制上。我第一次写好的程序下载验证后，发现在 23:00 就会发生进位变成 00:00，检验后发现是并行载入的条件出错。

通过本次实验的学习，我进一步巩固了用行为描述设计电路的方法，对时序逻辑电路和寄存器有了更深入的认识。

实验 13 移位寄存器设计与应用

课程名称：逻辑与计算机设计基础实验 实验类型：综合

实验项目名称：实验 13 移位寄存器设计与应用

学生姓名：熊子宇 学号：3200105278 同组学生姓名：薛婧

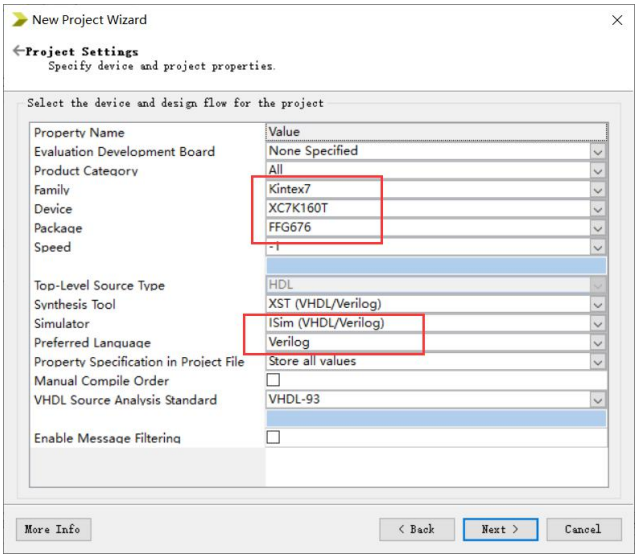
实验地点：紫金港东四 509 室 实验日期：2021 年 12 月 23 日

一、操作方法与实验步骤

1 设计 8 位带并行输入的右移移位寄存器

1.1 建立 ShiftReg8b 工程

点击主菜单上的 New Project 进行如下设置,点击 Finish 完成 Project 创建



1.2 结构化描述方式设计 8 位带并行输入的右移移位寄存器

新建 Verilog 文件 `shift_reg.v` 并输入以下代码。

```
module shift_reg(
   input wire clk, S_L, s_in,
   input wire [7:0] p_in,
   output wire [7:0] Q);

    wire [7:0]D_in, Sout, Soutbar;
    FD fd0 (.C(clk), .D(D_in[0]), .Q(Q[0]));
    FD fd1 (.C(clk), .D(D_in[1]), .Q(Q[1]));
    FD fd2 (.C(clk), .D(D_in[2]), .Q(Q[2]));
    FD fd3 (.C(clk), .D(D_in[3]), .Q(Q[3]));
    FD fd4 (.C(clk), .D(D_in[4]), .Q(Q[4]));
```

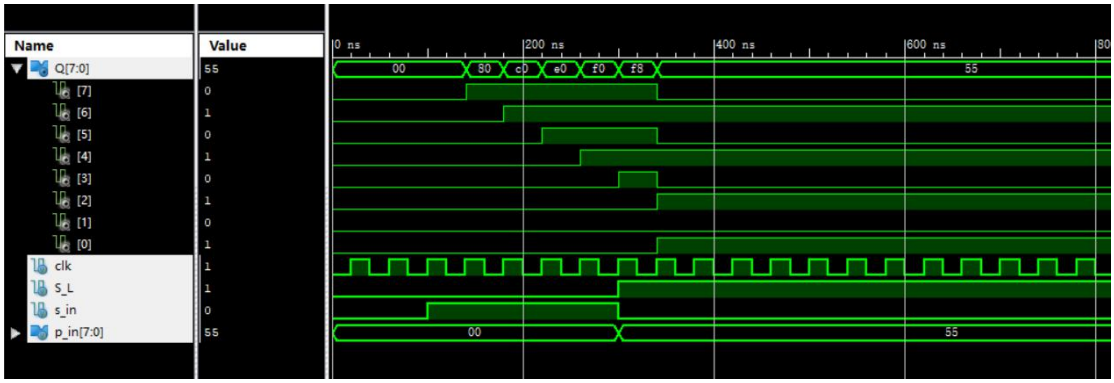
```
  FD fd5 (.C(clk), .D(D_in[5]), .Q(Q[5]));
  FD fd6 (.C(clk), .D(D_in[6]), .Q(Q[6]));
  FD fd7 (.C(clk), .D(D_in[7]), .Q(Q[7]));

   AND2 a07 (.I0(s_in), .I1(~S_L), .O(Soutbar[7]));
   AND2 a06 (.I0(Q[7]), .I1(~S_L), .O(Soutbar[6]));
   AND2 a05 (.I0(Q[6]), .I1(~S_L), .O(Soutbar[5]));
   AND2 a04 (.I0(Q[5]), .I1(~S_L), .O(Soutbar[4]));
   AND2 a03 (.I0(Q[4]), .I1(~S_L), .O(Soutbar[3]));
   AND2 a02 (.I0(Q[3]), .I1(~S_L), .O(Soutbar[2]));
   AND2 a01 (.I0(Q[2]), .I1(~S_L), .O(Soutbar[1]));
   AND2 a00 (.I0(Q[1]), .I1(~S_L), .O(Soutbar[0]));

   AND2 a10 (.I0(p_in[0]), .I1(S_L), .O(Sout[0]));
   AND2 a11 (.I0(p_in[1]), .I1(S_L), .O(Sout[1]));
   AND2 a12 (.I0(p_in[2]), .I1(S_L), .O(Sout[2]));
   AND2 a13 (.I0(p_in[3]), .I1(S_L), .O(Sout[3]));
   AND2 a14 (.I0(p_in[4]), .I1(S_L), .O(Sout[4]));
   AND2 a15 (.I0(p_in[5]), .I1(S_L), .O(Sout[5]));
   AND2 a16 (.I0(p_in[6]), .I1(S_L), .O(Sout[6]));
   AND2 a17 (.I0(p_in[7]), .I1(S_L), .O(Sout[7]));

   OR2 r0 (.I0(Soutbar[0]), .I1(Sout[0]), .O(D_in[0]));
   OR2 r1 (.I0(Soutbar[1]), .I1(Sout[1]), .O(D_in[1]));
   OR2 r2 (.I0(Soutbar[2]), .I1(Sout[2]), .O(D_in[2]));
   OR2 r3 (.I0(Soutbar[3]), .I1(Sout[3]), .O(D_in[3]));
   OR2 r4 (.I0(Soutbar[4]), .I1(Sout[4]), .O(D_in[4]));
   OR2 r5 (.I0(Soutbar[5]), .I1(Sout[5]), .O(D_in[5]));
   OR2 r6 (.I0(Soutbar[6]), .I1(Sout[6]), .O(D_in[6]));
   OR2 r7 (.I0(Soutbar[7]), .I1(Sout[7]), .O(D_in[7]));

endmodule
```

1.3 仿真激励输入

建立基准测试波形文件并输入如下代码

```
initial begin
       // Initialize Inputs
       clk = 0;
       S_L = 0;
       s_in = 0;
       p_in[7:0] = 8'b0000_0000;

       #100;

       S_L = 0;
       s_in = 1;
       p_in =0;

     #200;
       S_L = 1;
       s_in = 0;
       p_in[7:0] = 8'b0101_0101;
       #500;
       end

       always begin
           clk = 0; #20;
           clk = 1; #20;
       end
```

查看波形



2 设计跑马灯应用

2.1 建立 MyMarquee 工程

点击主菜单上的 New Project 进行设置,点击 Finish 完成 Project 创建

2.2 调用 ShiftReg8b

将任务 1 中设计的 shift_reg.v 拷贝到本工程中。

2.3 调用 pbdebounce 按键去抖动模块

时钟计数分频器 clkdiv.v 代码如下：

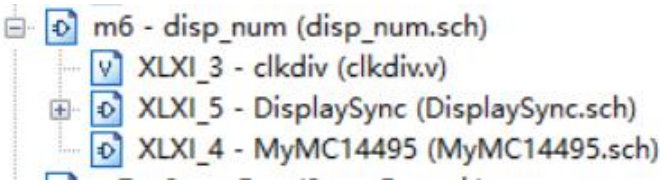
```
module clkdiv(input clk, input rst, output reg[31:0]clkdiv);

always @ (posedge clk or posedge rst) begin
    if (rst) clkdiv<=0;
    else clkdiv <= clkdiv + 1'b1;
end

endmodule
```

去抖动模块 pbdebounce.v 代码如下：

```
module pbdebounce(
    input wire clk_1ms,
    input wire button,
    output reg pbreg
    );

    reg [7:0] pbshift;

    always@(posedge clk_1ms) begin
        pbshift=pbshift<<1;
        pbshift[0]=button;
        if (pbshift==8'b0)
            pbreg=0;
        if (pbshift==8'hFF)
            pbreg=1;
    end
endmodule
```

2.4 调用 DispNum 模块

将以往实验实现的七段数码管显示模块 disp_num.sch 以及相关文件拷贝到本工程中。



2.5 设计按键数据输入 CreateNumber 模块

新建 verilog 文件 CreateNumber.v 并输入以下代码：

```
module CreateNumber (
    input wire [1:0] btn_out,
    input wire [1:0] SW,
    output reg [3:0] regA,
    output reg [3:0] regB
    );
    wire [3:0] A,B;

    initial begin
        regA <= 4'b0000;
        regB <= 4'b0000;
        end

    AddSub4b a1(.A(regA[3:0]), .B(4'b0001), .Ctrl(SW[0]), .S(A));
    AddSub4b a2(.A(regB[3:0]), .B(4'b0001), .Ctrl(SW[1]), .S(B));

    always@(posedge btn_out[0]) regA[3:0] <= A;
    always@(posedge btn_out[1]) regB[3:0] <= B;

endmodule
```

2.6 根据原理设计 100ms 时钟 clk_100ms.v

新建 Verilog 文件 clk_100ms.v 并输入以下代码。

```
module counter_100ms(clk, clk_100ms);
    input wire clk;
    output reg clk_100ms;
    reg [31:0] cnt;
    always @ (posedge clk) begin
        if (cnt < 500_000_00) begin
            cnt <= cnt + 1;
        end else begin
            cnt <= 0;
            clk_100ms <= ~clk_100ms;
        end
    end
endmodule
```

2.7 调用 LED 显示模块

将 LEDP2S_IO.v 和 LEDP2S.ngc 文件拷贝到本工程中。LEDP2S_IO.v 代码如下：

```
module    LEDP2S(input wire clk,              //parallel to serial
                 input wire rst,
                 input wire Start,
                 input wire[DATA_BITS-1:0] PData,
                 output wire sclk,
```

```
                    output wire sclrn,
                    output wire sout,
                    output reg  EN
                    );

parameter
    DATA_BITS = 16,                             // data length
    DATA_COUNT_BITS = 4,                        // data shift bits
    DIR = 0;                                  // Shift direction =0 左移

endmodule
```

2.8 新建源文件 top，并右键设为“Top Module”

新建 Verilog 文件 top.v 并输入以下代码。

```
module top(input wire clk,
    input wire [1:0] BTN,//实现 A,B 两个操作数每按一下加 1
    input wire [4:0] sw,
    //sw[2] = 0, 串行/循环右移；sw[4]=0 为串行右移（输入值为 sw[3]），sw[4]=1
为循环右移
    //sw[2] = 1，并行输入。输入的值为{regA, regB}
    //sw[0]控制 regA 自增，sw[1]控制 regB 自增

   output wire BTNX4,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire [7:0] LED,
    output ledclk,
    output ledsout,
    output ledclrn,
    output LEDEN

);
    wire [3:0] regA, regB;
    wire [1:0] btn_out;
    wire shift_in;
    wire [31:0] clk_div;
    assign BTNX4 = 1'b0;
    counter_100ms m1(clk,clk_100ms);

    clkdiv m2(clk,0,clk_div);

    pbdebounce b0(clk_div[17],BTN[0],btn_out[0]);//BTN[0]去抖动
   pbdebounce b1(clk_div[17],BTN[1],btn_out[1]);

    assign shift_in = sw[4] ? LED[0]: sw[3];

    shift_reg r0 (.clk(clk_100ms), .S_L(sw[2]), .s_in(shift_in),
    .p_in({regA[3:0],regB[3:0]}), .Q(LED[7:0]));

    CreateNumber m4(btn_out[1:0], sw[1:0], regA, regB);

    disp_num
d0(.clk(clk), .HEXS({regA[3:0],regB[3:0],LED[7:0]}), .LES(4'b0000), .p
oints(4'b0000),
                    .RST(1'b0), .AN(AN), .Segment(SEGMENT));


    LEDP2S #(.DATA_BITS(16), .DATA_COUNT_BITS(4), .DIR(0))
```

```
    U7(.clk(clk), .rst(1'b0), .Start(clk_div[20]), .PData({8'hFF, LED}),
    .sclk(ledclk), .sclrn(ledclrn), .sout(ledsout), .EN(LEDEN));

endmodule
```

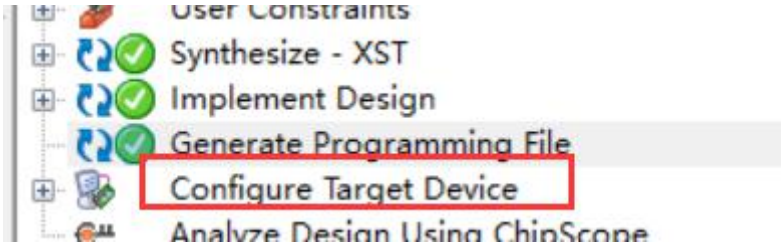
2.9 建立用户时序约束并为模块的端口指定引脚分配

建立引脚分配文件并输入代码：

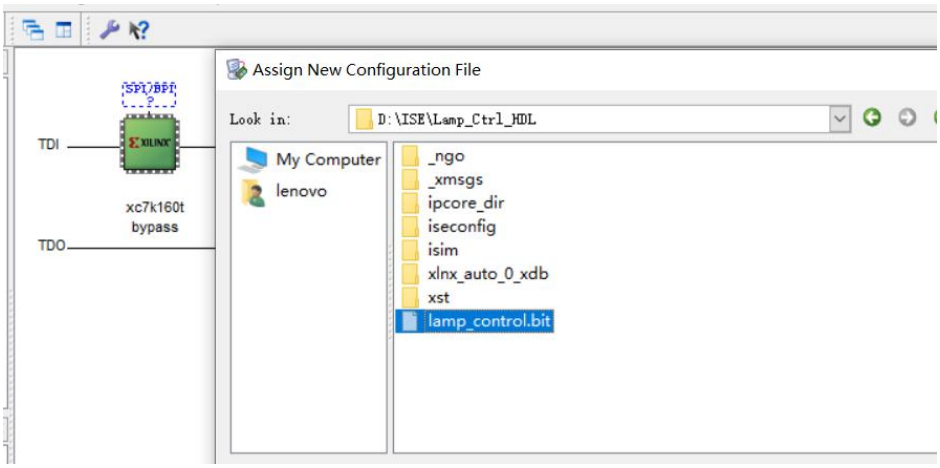
```
NET "ledclk"    LOC = N26 | IOSTANDARD = LVCMOS33;
NET "ledclrn"   LOC = N24 | IOSTANDARD = LVCMOS33;
NET "ledsout"   LOC = M26 | IOSTANDARD = LVCMOS33;
NET "LEDEN"    LOC = P18 | IOSTANDARD = LVCMOS33;

NET "clk"  LOC = AC18 | IOSTANDARD = LVCMOS18;
NET "clk"  TNM_NET = TM_CLK;
TIMESPEC TS_CLK_100M = PERIOD "TM_CLK" 10ns HIGH 50%;

NET "BTN[0]" LOC = V18 | IOSTANDARD = LVCMOS18;
NET "BTN[0]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "BTN[1]" LOC = V19 | IOSTANDARD = LVCMOS18;
NET "BTN[1]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "BTNX4" LOC = W16  | IOSTANDARD = LVCMOS18 ;#SW[15]

NET "sw[0]"LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "sw[1]"LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "sw[2]"LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "sw[3]"LOC = AA12 | IOSTANDARD = LVCMOS15;
NET "sw[4]"LOC =Y13  | IOSTANDARD = LVCMOS15;

NET"LED[0]"LOC=W23 | IOSTANDARD=LVCMOS33;
NET"LED[1]"LOC=AB26 | IOSTANDARD=LVCMOS33;
NET"LED[2]"LOC=Y25 | IOSTANDARD=LVCMOS33;
NET"LED[3]"LOC=AA23 | IOSTANDARD=LVCMOS33;
NET"LED[4]"LOC=Y23 | IOSTANDARD=LVCMOS33;
NET"LED[5]"LOC=Y22 | IOSTANDARD=LVCMOS33;
NET"LED[6]"LOC=AE21 | IOSTANDARD=LVCMOS33;
NET"LED[7]"LOC=AF24 | IOSTANDARD=LVCMOS33;

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;
NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
```

2.10 下载到 SWORD 板

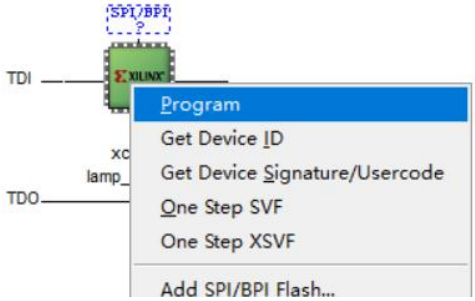
（1）通过 Synthesize - XST, Implement Design, Generate Programming File 后，点击 Configure Target Device



（2）载入 bit 文件（此处不是本工程的 bit 文件，仅为示意）



（3）点击 Program 下载至 SWORD 板

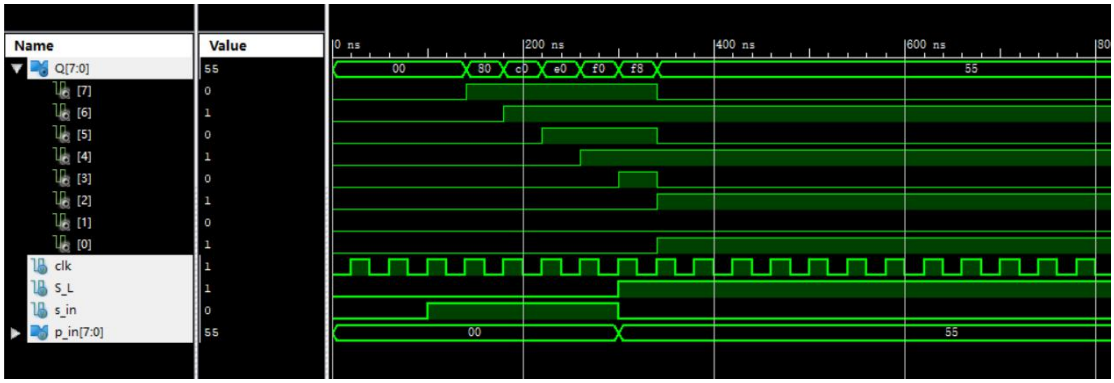


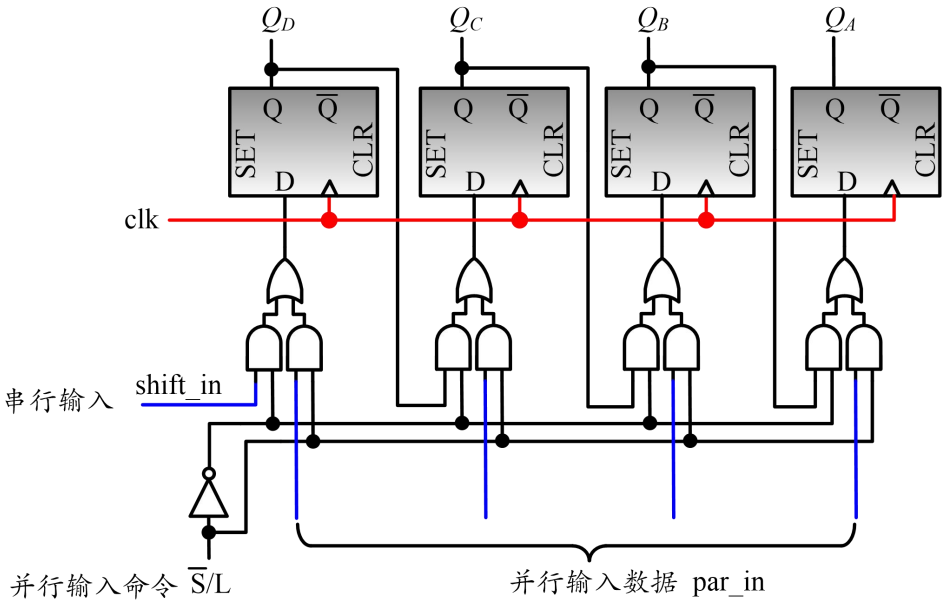
2.11 根据真值表，操作实验板，验证功能

二、实验结果与分析

1 设计 8 位带并行输入的右移移位寄存器

仿真激励波形分析





分析：如上图为四位带并行输入的右移移位寄存器，八位的原理类似。首先将 Q[7:0]初始化为 0。

最初并行输入命令 S_L=0，为串行输入，输入内容 s_in=0，所以 Q[7:0]似乎保持不变，一直为 0，实际上在每个时钟周期中 0 在不断移位。串行输入内容 s_in=1 后，移位过程显式地表现出来。1 首先进入 Q[7]，在下个时钟周期移入 Q[6]，随后每个时钟周期不断右移，直到移入 Q[3]。

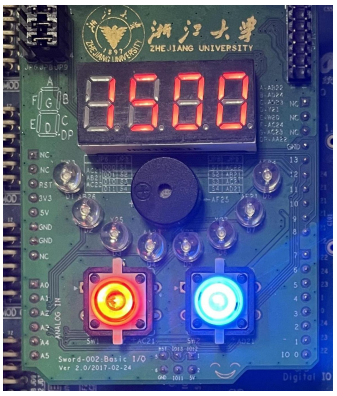
S_L=1，为并行载入信号，载入内容 p_in[7:0]=h55。在时钟正边沿时，寄存器发生并行载入，Q[7:0]=h55 并在以后的时钟周期内保持不变。

2 设计跑马灯应用

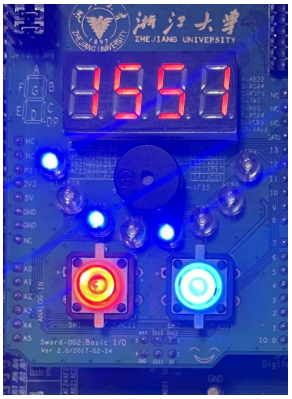
sw[2] = 0, 串行/循环右移；sw[4]=0 为串行右移（输入值为 sw[3]），sw[4]=1 为循环右移

sw[2] = 1，并行输入。输入的值为{regA, regB}

sw[0]控制 regA 自增，sw[1]控制 regB 自增。

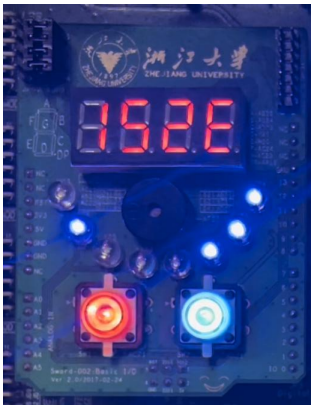


初始状态



sw[2] = 1，并行输入

sw[4]=0，串行右移，输入 sw[3]=0



sw[4]=0，串行右移，输入 sw[3]=1

sw[4]=1，循环右移

三、讨论、心得

本实验是数字逻辑设计实验课程的最后一个实验，综合运用了理论课程的时序逻辑电路、移位寄存器和实验课程的显示模块、按键模块、LED 模块等众多模块，是一个综合性的设计实验，比较考验了我在这门课程上的总体的学习掌握程度。

一学期的课程结束后，我掌握比较好的方面有：组合逻辑电路的设计，包括七段数码管的显示、多路选择器的运用、算术运算函数的组合逻辑电路、层次设计等，寄存器及其应用，简单的时序逻辑电路的设计。比较扎实的理论课知识为我的实验打下了较好的基础。另外，在实验的前半段，主要采用原理图方式设计，对硬件语言的初学者来说十分友好。由原理图逐渐过渡到门级原语、结构化表述以及行为描述，安排比较合理，由浅入深。

但是我的缺点也比较突出。第一，Verilog 代码仍然只会读不太会写。对于具体的语法细节比较模糊，自己撰写容易出现语法错误。在本门课程结束之后，我需要系统地学习 Verilog 语法，为后续进阶的硬件课程夯实基础。第二，引脚约束只会调用老师已经给好的引脚，对于引脚约束的原理和本质并不十分清楚，因此经常出错。

总的来说，在数逻理论和实验的学习过程中，我比较好地完成了所安排的任务，让我感受到了计算机硬件课程的乐趣。

四、个人生活照片