实验 12

——寄存器和寄存器传输设计

姓名：　　学号：　　专业：

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：

实验时间：2019-11-20　　实验地点：紫金港东 4-509　　指导老师：洪奇军

一．实验目的

掌握寄存器传输电路的工作原理

掌握寄存器传输电路的设计方法

掌握 ALU 和寄存器传输电路的综合应用

二．实验设备与材料

2.1 实验设备

装有 Xilinx ISE 14.7 的计算机　　1 台

SWORD 开发板　　1 套

2.2 实验材料

无

三．实验任务

基于 ALU 的数据传输应用设计

四．实验原理

4.1 寄存器

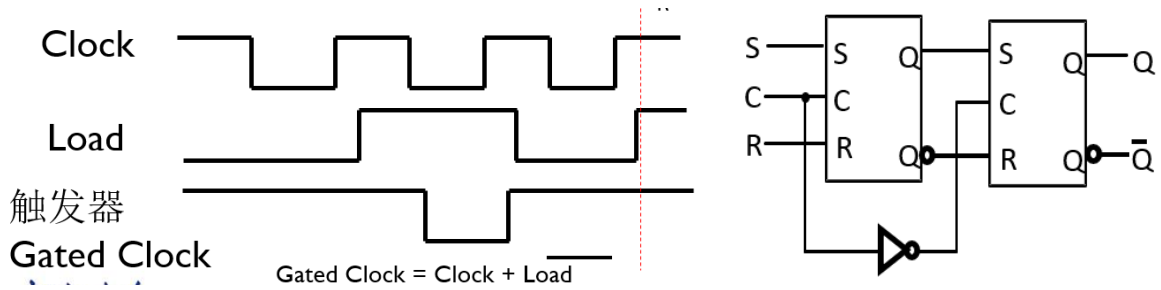
- 一组二进制存储单元

- 一个寄存器可以用于存储一系列二进制值，通常用于进行简单数据存储、移动和处理等操作
- 能存储信息并保存多个时钟周期，能用信号来控制“保存”或“加载”信息

采用门控时钟的寄存器:

如果 Load 信号为 1，允许时钟信号通过，如果为 0 则阻止时钟信号通过

例如:对于上升沿触发的边沿触发器，或负向脉冲触发的主从触发器:

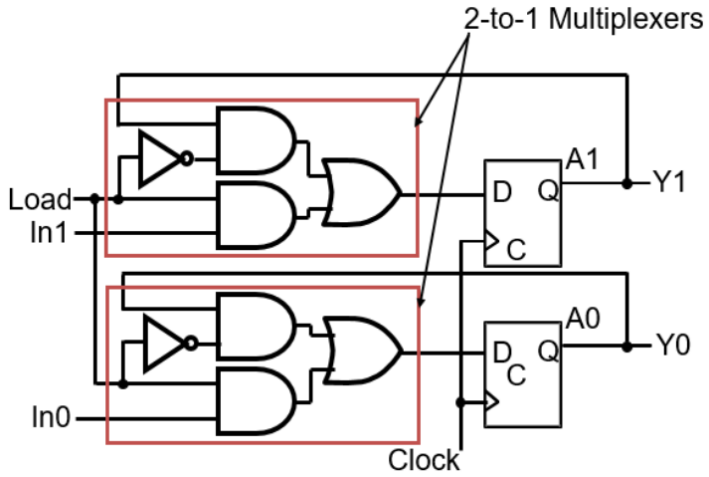


采用 Load 控制反馈的寄存器:

进行有选择地加载寄存器的更可靠方法是:

保证时钟的连续性

选择性地使用加载控制来改变寄存器的内容



Load=0 时， In1, In0 同 0 与为 0, 无法输入数据

(通 1) A1, A0 同~Load=1 与保持不变

(通 2) A1, A0 与 Load=0 或保持不变为 A1, A0

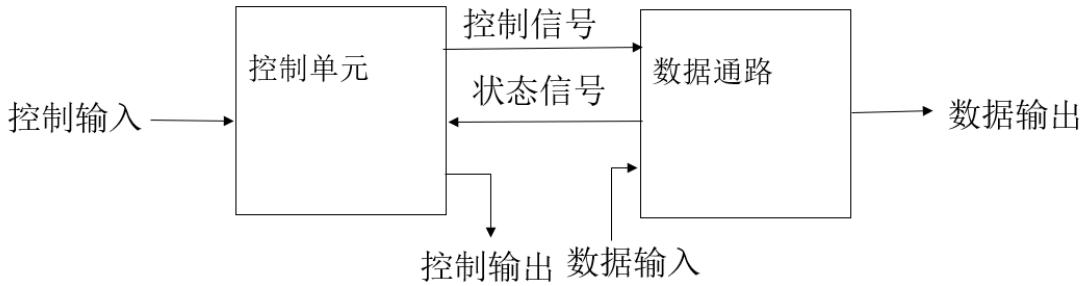
Load=1 时， In1, In0 同 load=1 与后为 In1, In0, 作为 D 触发器的输入

(通 1) A1, A0 同~Load=0 与后为 0, 无反馈数据

(通 2) In1, In2 与 0 或后作为 D 触发器的输入

4.2 寄存器传输

寄存器传输：寄存器中数据的传输和处理



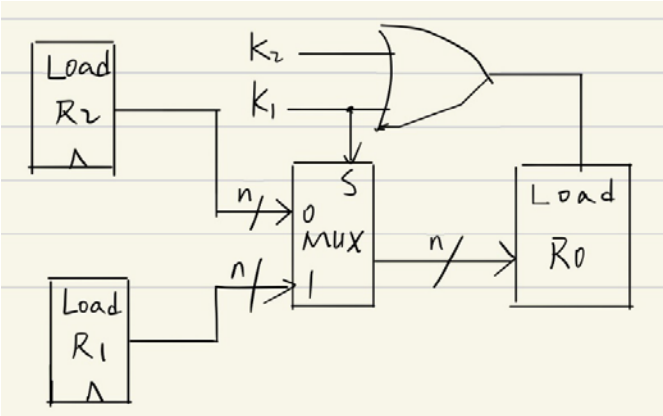
三个基本单元：寄存器组、操作、操作控制

基本操作:加载、计数、移位、加法、按位操作等

4.3 基于多路选择器总线的寄存器传输

连接寄存器的多路选择器可以产生灵活的数据传输结构

（注意：为清晰起见图中省略了时钟信号）



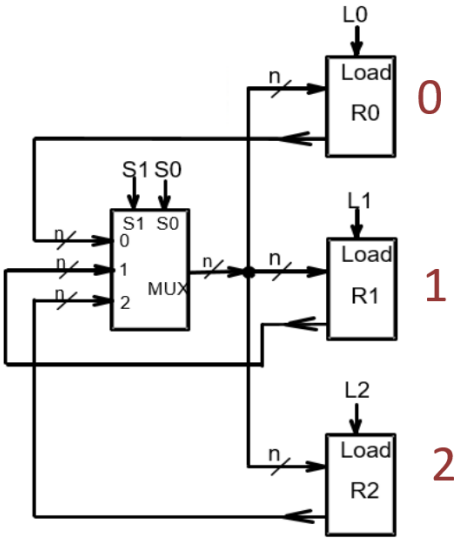
由一个多路选择器驱动的总线可以降低硬件开销

这个结构不能实现多个寄存器相互之间的并行传输操作

S1S0=00, 总线数据为 0,

S1S0=01, 总线数据为 1,

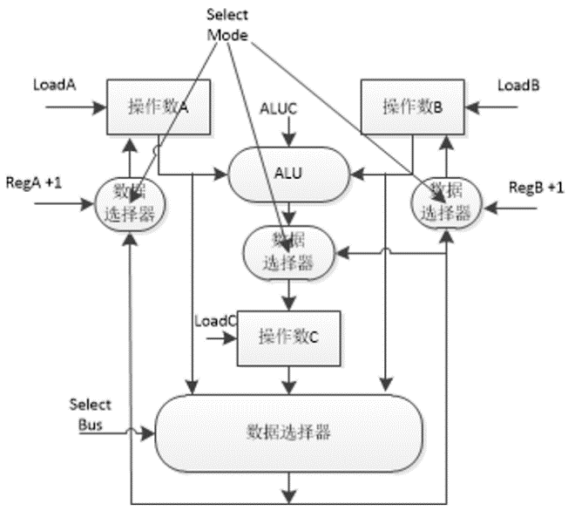
S1S0=10, 总线数据为 2,



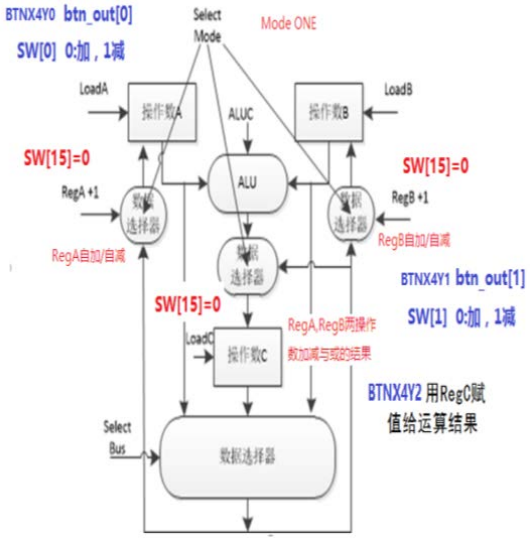
4.4 寄存器传输应用设计

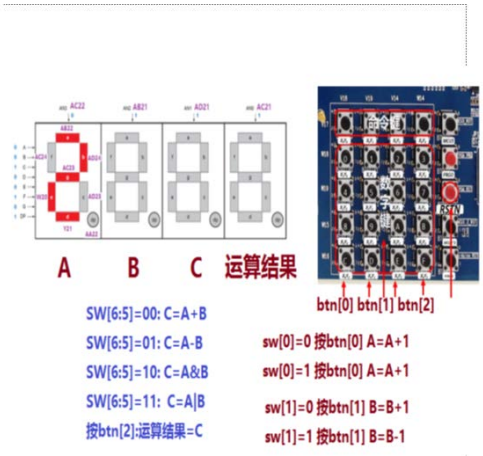
Mode1：ALU 运算输出控制

Mode2：数据传输控制



Mode1： ALU 运算输出控制





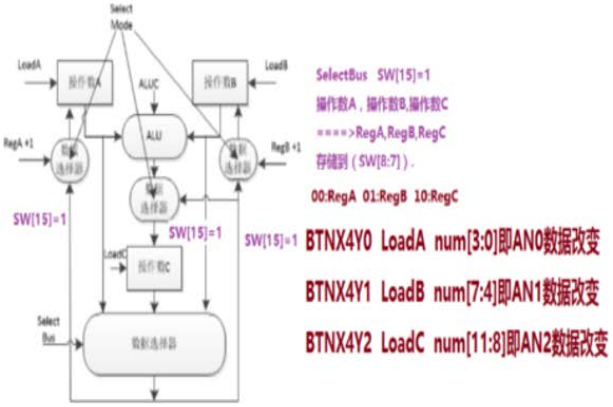
Mode2：数据传输控制

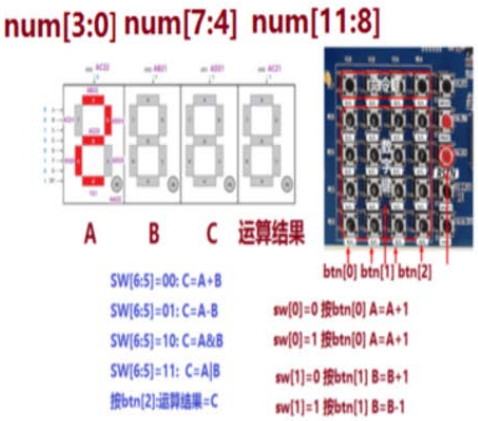
sw[8:7]对应 SelectBus 即总线上放的是什么数据：

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]





五 . 实验内容与步骤

任务：基于 ALU 的数据传输应用设计

新建工程，工程名称用 MyALUTrans，Top Level Source Type 用 HDL；

添加如下模块：

ALU 模块

4 位 4 选 1 模块

防抖动模块

显示模块

新建源文件，类型是 Verilog，文件名称用 Top；

右键设为“Set as Top Module”；

实现基于 ALU 的数据传输应用设计；

```
module Top(
    input wire clk,
    input wire[3:0] btn,
    input wire[15:0] SW,
    output wire[3:0] AN,
    output wire[7:0] segments,
    output wire BTN_X
    );
    reg[15:0] num;
    wire[3:0] A1,B1,C,A2,B2,C2,Result;

    wire[2:0] btn_out;
    wire Co;
    wire[31:0] clk_div;

    assign BTN_X=0;
```

```
    clkdiv m0(clk,1'b0,clk_div);
    pbdebounce m1(.clk_1ms(clk_div[17]),.button(btn[0]),.pbreg(btn_out[0]));
    pbdebounce m2(.clk_1ms(clk_div[17]),.button(btn[1]),.pbreg(btn_out[1]));
    pbdebounce m3(.clk_1ms(clk_div[17]),.button(btn[2]),.pbreg(btn_out[2]));


    AddSub4b m4(.A(num[3:0]),.B(4'b0001),.Ctrl(SW[0]),.S(A1));
    AddSub4b m5(.A(num[7:4]),.B(4'b0001),.Ctrl(SW[1]),.S(B1));

    Mux4to14b m6(
        .I0(num[3:0]),
        .I1(num[7:4]),
        .I2(num[11:8]),
        .I3(4'b0000),
        .S(SW[8:7]),
        .O(Result));

    assign A2 = (SW[15]==1'b0)?A1:Result;
    assign B2 = (SW[15]==1'b0)?B1:Result;
    assign C2 = (SW[15]==1'b0)?C: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;
    ALU m7(.A(num[3:0]),.B(num[7:4]),.S(SW[6:5]),.C(C),.Co(Co));

    disp_num m8(.clk(clk),.HEXS({num[11:0],C[3:0]}),
        .LES(4'b0000),.points(4'b1111),.RST(1'b0),
        .AN(AN),.Segment(segments));

endmodule
```

- UCF 引脚定义

输入：

sw[15]=0（Mode0）时：

按键控制输入：

sw[2] 控制 Reg A，

sw[3] 控制 Reg B，

sw[4] 对 RegC 赋值

按键加/减 1 控制：

sw[0] 对应 btn[0]，

sw[1] 对应 btn[1]

ALU 运算控制：

sw[6:5]：00-加，01-减，10-与，11-或

(RegC 是 Reg A, Reg B 加减与或的结果)

sw[15]=1（Mode1）时：数据传输控制

sw[8:7]对应 SelectBus：00-选择 A，01-选择 B，10-选择 C

sw[2] LoadA(num[3:0])，sw[3] LoadB(num[7:4])，sw[4] LoadC(ALU 结果)

(对 SW[8:7]选择出来后的结果加载到哪个寄存器中)

输出：

AN[0]：Reg B

AN[1]：Reg A

AN[2]：ALU 结果

AN[3]：Reg C

sw[15]=0 Mode0

按键控制输入：

BTNX4Y0 作为按键信号控制 RegA(自加或自减)

BTNX4Y1 作为按键信号控制 RegB

BTNX4Y2 作为按键信号对 RegC 赋值

```
net "clk" loc=AC18 | iostandard=LVCMOS18;
net"btn[0]"loc=V18 | iostandard=LVCMOS18;
net"btn[1]"loc=V19 | iostandard=LVCMOS18;
net"btn[2]"loc=V14 | iostandard=LVCMOS18;
net"btn[3]"loc=W14 | iostandard=LVCMOS18;
net"BTN_X"loc=W16  | iostandard=LVCMOS18;

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"SW[5]"loc =Y12  | iostandard=LVCMOS15;
net"SW[6]"loc =AD11 | iostandard=LVCMOS15;
net"SW[7]"loc =AD10 | iostandard=LVCMOS15;
net"SW[8]"loc =AE10 | iostandard=LVCMOS15;
net"SW[9]"loc =AE12 | iostandard=LVCMOS15;
net"SW[10]"loc=AF12 | iostandard=LVCMOS15;
net"SW[11]"loc=AE8  | iostandard=LVCMOS15;
net"SW[12]"loc=AF8  | iostandard=LVCMOS15;
net"SW[13]"loc=AE13 | iostandard=LVCMOS15;
net"SW[14]"loc=AF13 | iostandard=LVCMOS15;
net"SW[15]"loc=AF10 | iostandard=LVCMOS15;

net"AN[0]"loc=AC21 | iostandard=LVCMOS33;
net"AN[1]"loc=AD21 | iostandard=LVCMOS33;
net"AN[2]"loc=AB21 | iostandard=LVCMOS33;
net"AN[3]"loc=AC22 | iostandard=LVCMOS33;

net"segments[0]"loc=AB22 | IOSTANDARD=LVCMOS33;
net"segments[1]"loc=AD24 | IOSTANDARD=LVCMOS33;
net"segments[2]"loc=AD23 | IOSTANDARD=LVCMOS33;
net"segments[3]"loc=Y21  | IOSTANDARD=LVCMOS33;
net"segments[4]"loc=W20  | IOSTANDARD=LVCMOS33;
net"segments[5]"loc=AC24 | IOSTANDARD=LVCMOS33;
net"segments[6]"loc=AC23 | IOSTANDARD=LVCMOS33;
net"segments[7]"loc=AA22 | IOSTANDARD=LVCMOS33;
```

- 实验结果

可以看到，

SW[15]=0 时，按下对应按钮，RegA 和 RegB 的值相应自增/自减，或者 RegC 被赋值

SW[15]=1 时，按下对应按钮，可以选择对 RegC 赋值的源寄存器

SW[6:5]可以控制 ALU 的运算方式

六．实验心得与体会

这次实验主要的内容是实现寄存器的设计以及寄存器的数据传输，包括对寄存器进行赋值，以及将一个寄存器的值赋到另一个寄存器，在本实验中，寄存器的特性得到了充分的体现，即可以暂时存储数据。寄存器是时序电路中一个非常重要的部分，在实现寄存器传输，用到了非阻塞型赋值语句，并了解了阻塞型赋值语句还有非阻塞型赋值语句的区别以及它们各自在什么时候适用。感觉在这次实验中学到了许多东西，还需继续加强学习，强化对时序电路的认识。

实验 13

——计数器、定时器设计与应用

姓名：　　学号：　　专业：

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：

实验时间：2019-11-20　　实验地点：紫金港东 4-509　　指导老师：洪奇军

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．实验设备与材料

2.1 实验设备

装有 Xilinx ISE 14.7 的计算机1 台

SWORD 开发板　　1 套

2.2 实验材料

无

三．实验任务

任务 1：采用行为描述设计同步四位二进制计数器 74LS161

任务 2：基于 74LS161 设计时钟应用

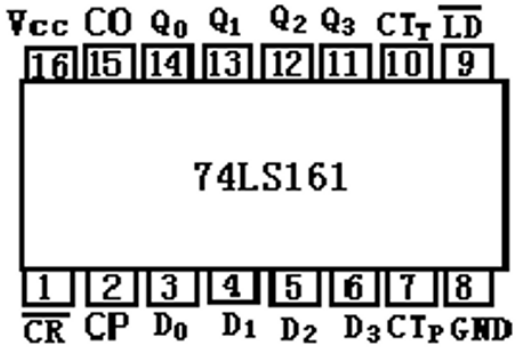
四 . 实验原理

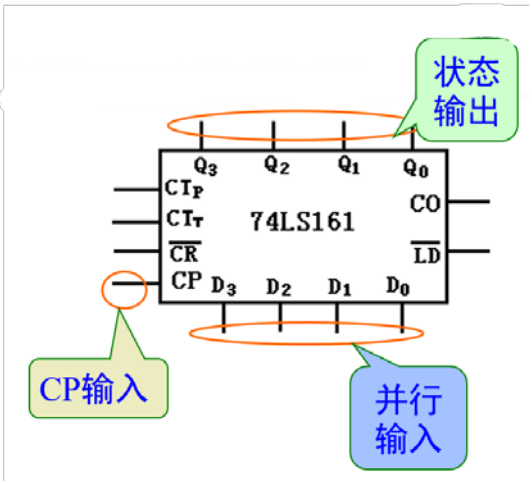
4.1 同步四位二进制计数器 74LS161

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 功能描述：





清零端$\overline{CR}$

置数端 $\overline{LD}$

使能端CT_P, CT_T

进位输出端CO

- 74LS161 功能表：

异步清0功能最优先

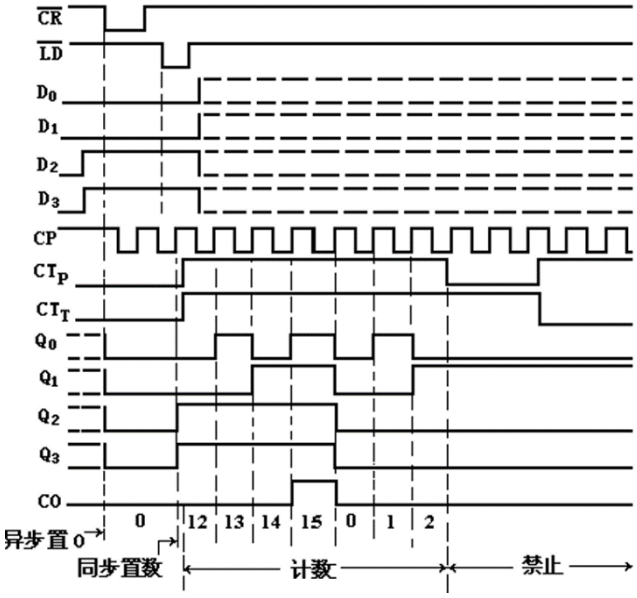
输入						输出			
$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	CP	$D_3D_2D_1D_0$	Q_3	Q_2	Q_1	Q_0
0	×	×	×	×	××××	0	0	0	0
1	0	×	×	↑	$d_3d_2d_1d_0$	d_3	d_2	d_1	d_0
1	1	0	1	×	××××	保持			
1	1	×	0	×	××××	保持			
1	1	1	1	↑	××××	计数			

同步并行置数

CP上升沿有效

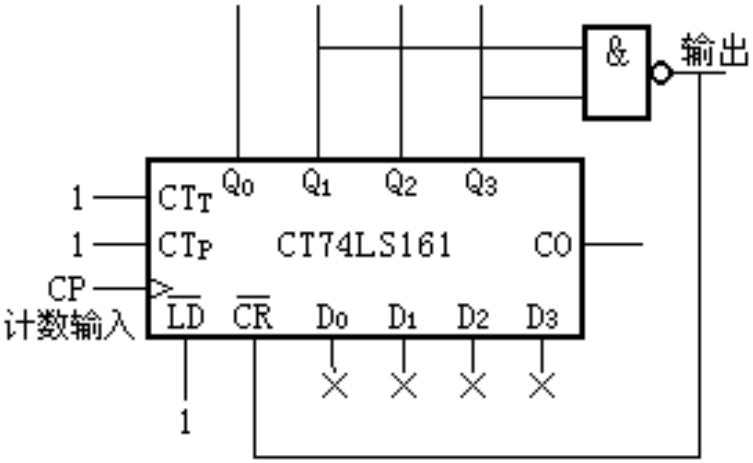
$CO = Q_3Q_2Q_1Q_0CT_T$

- 74LS161 时序图：



4.2 实现十进制计数器

```
0 0 0 0      0
1 0 0 0      1
0 1 0 0      2
1 1 0 0      3
0 0 1 0      4
1 0 1 0      5
0 1 1 0      6
1 1 1 0      7
0 0 0 1      8
1 0 0 1      9
0 1 0 1    10->0
```

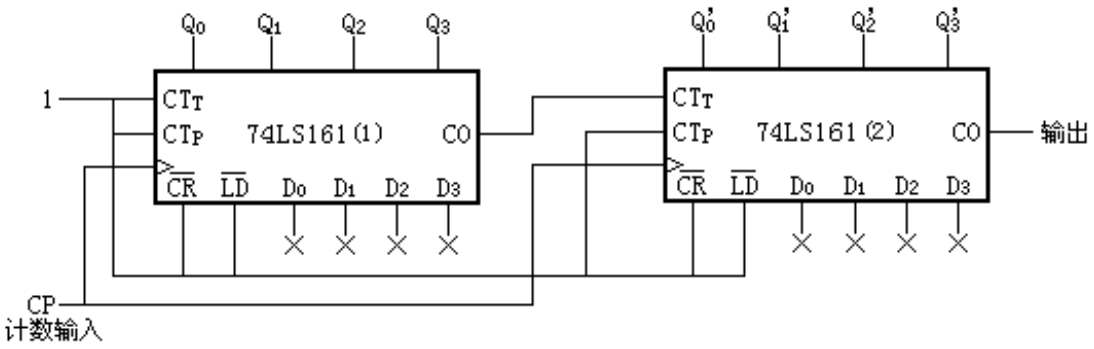


利用与非门拾取状态 1010

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

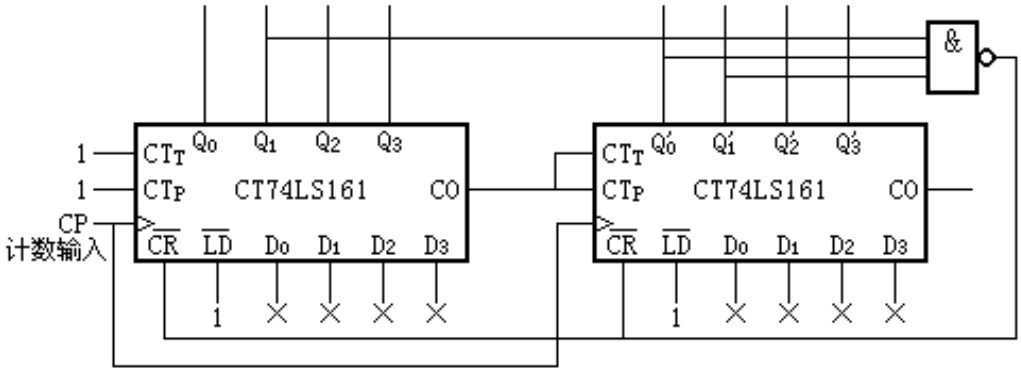
4.3 实现 16×16 进制计数器



在计到 1111 以前，CO1 = 0，高位片保持原状态不变

在计到 1111 时，CO1 = 1，高位片在下一个 CP 加 1

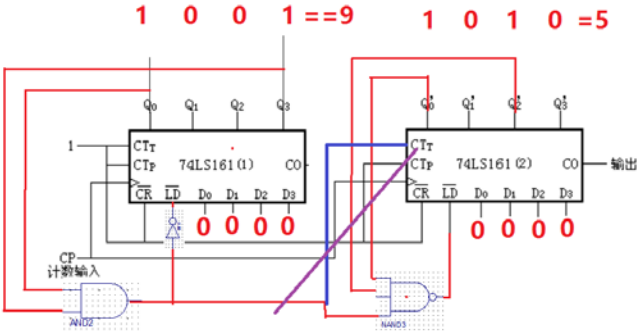
4.4 实现 50 进制计数器（16 进制）



十进制 50 对应的二进制数为 0011 0010

实现从 0000 0000 到 0011 0001 的 50 进制计数器

4.5 分钟 60 进制（十进制显示）



4.6 数字时钟

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

数字钟的初值通过初始化语句来实现；

用数码管前两位显示小时的十位和个位，后两位显示分钟的十位和个位；

五 . 实验内容与步骤

任务 1：采用行为描述设计同步四位二进制计数器 74LS161

任务 2：基于 74LS161 设计时钟应用

说明：按 74LS161 功能表来写代码

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		

5.1 设计同步四位二进制计数器 74LS161

新建工程，工程名称用 My74LS161，Top Level Source Type 用 HDL

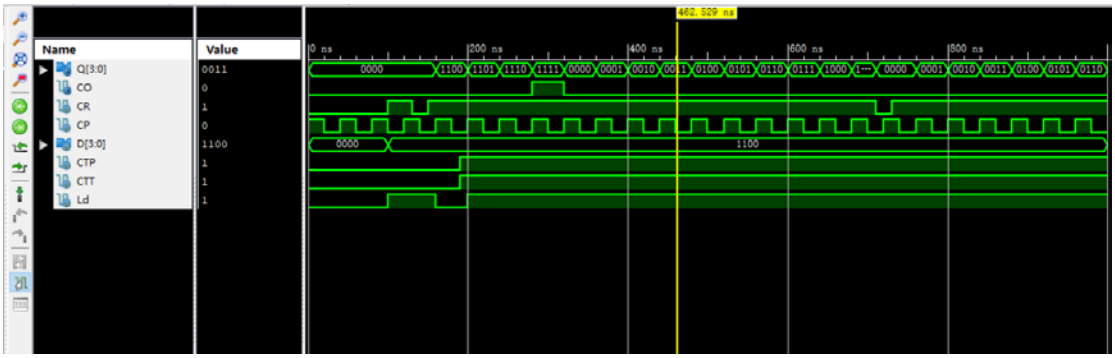
用行为描述设计，CR 是异步清零，LD 是同步置位

```
module my74LS161(
    input wire CR,
    input wire CP,
    input wire[3:0] D,
    input wire CTP,
    input wire CTT,
    input wire Ld,
    output reg[3:0] Q,
    output wire CO
    );
    wire[3:0] a,b,c,d;

    assign CO = Q[3]&&Q[2]&&Q[1]&&Q[0]&&CTT;

    assign a = D;
    assign b = Q + 4'b0001;

    always @(posedge CP or negedge CR) begin
        if(CR==0) Q<=4'b0000;
        else begin
            if(Ld==0) Q<=a;
            else if(CTT && CTP) Q<=b;
        end
    end
endmodule
```

激励代码：

```
initial begin
    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
    CP=1;#20;
    CP=0;#20;
end
```

仿真波形：



5.2 基于 74LS161 设计时钟应用

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

用结构化描述设计，

调用 My74LS161

调用分频模块，用 100ms 作为分的驱动时钟

调用显示模块

（设计一个数字钟，使用 60 进制和 24 进制计数器，实现 24 小时内时间的实时显示）

下载验证

- Top 设计：

```
module Top(clk, AN, SEGMENT);
    input wire clk;
    output wire[3:0] AN;
    output wire[7:0] SEGMENT;

    wire[15:0] num;
    clk_100ms m4(.clk(clk), .clk_100ms(clk_100ms));
    my74LS161 m0(.CR(1'b1), .Ld(~(num[3]&num[0])), .CTT(1'b1),
        .CTP(1'b1), .CP(clk_100ms), .D(4'b0000), .Q(num[3:0]));
    my74LS161 m1(.CR(1'b1), .Ld(~(num[6]&num[4]&num[3]&num[0])),
        .CTT(num[3] & num[0]), .CTP(1'b1), .CP(clk_100ms), .D(4'b0000), .Q(num[7:4]));
    assign hCarry1 = (~num[13])&(num[11]&num[8])&num[6]&num[4]&num[3]&num[0];
    assign hCarry2 = num[13]&(~num[12])&(num[9]&num[8])&(num[6]&num[4])&(num[3]&num[0]);
    assign hCarry = hCarry1|hCarry2;
    my74LS161 m2(.CR(1'b1), .Ld(~hCarry), .CTT(num[6]&num[4]&num[3]&num[0]),
        .CTP(1'b1), .CP(clk_100ms), .D(4'b0000), .Q(num[11:8]));
    assign t23_59 = num[13]&(~num[12])&(num[9]&num[8])&(num[6]&num[4])&(num[3]&num[0]);
    assign tX9_59 = (num[11]&num[8])&(num[6]&num[4])&(num[3]&num[0]);
    my74LS161 m3(.CR(1'b1), .Ld(~(t23_59)), .CTT(tX9_59), .CTP(1'b1), .CP(clk_100ms), .D(4'b0000), .Q(num[15:12]));
    disp_num m5(.clk(clk), .HEXS(num), .LES(4'b0000), .points(4'b1111), .RST(1'b0), .AN(AN), .Segment(SEGMENT));
endmodule
```

- ucf 引脚约束：

```
net "clk" loc=AC18 | iostandard=LVCMOS18;

net"AN[0]"loc=AD21 | iostandard=LVCMOS33;
net"AN[1]"loc=AB21 | iostandard=LVCMOS33;
net"AN[2]"loc=AC21 | iostandard=LVCMOS33;
net"AN[3]"loc=AC22 | 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;
```

- 实验结果

可以看到，四位 7 段数码管上所显示 24 小时制时间值随时间推移而增加，到 23: 59 时回到 00: 00 重新开始

六．实验心得与体会

本次实验的内容较为简单，一部分为设计 4 位二进制同步计数器，一部分为设计时钟应用，主要用到的原理为计数器还有寄存器相关知识，比较难的地方在于十进制数 24 进位条件的判断。目前为止对于 Verilog 语言的使用已经比较熟悉，对行为描述和结构化描述都有了一定的了解，深刻的感受到 Verilog 语言在设计较复杂电路时的简洁还有优越性。课后还需继续加强学习，希望可以圆满的结束所有的实验。

实验 14

——移位寄存器设计与应用

姓名：　　学号：　　专业：

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：

实验时间：2019-11-20　　实验地点：紫金港东 4-509　　指导老师：洪奇军

一．实验目的

掌握支持并行输入的移位寄存器的工作原理

掌握支持并行输入的移位寄存器的设计方法

二．实验设备与材料

2.1 实验设备

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

2.2 实验材料

无

三．实验任务

任务 1：设计 8 位带并行输入的右移移位寄存器

任务 2：设计主板 LED 灯驱动模块

任务 3：设计主板七段数码管驱动模块

四 . 实验原理

4.1 移位寄存器

每来一个时钟脉冲，寄存器中的数据按顺序向左或向右移动一位

必须采用主从触发器或边沿触发器，不能采用锁存器

数据移动方式：

左移、右移、循环移位

数据输入输出方式：

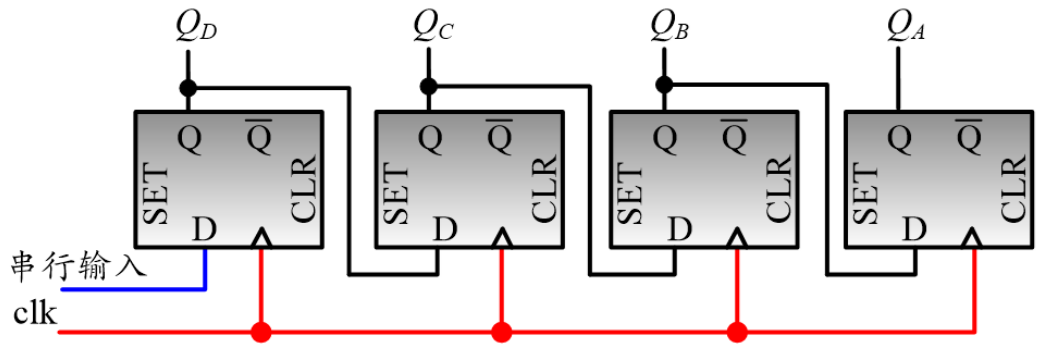
串行输入，串行输出

串行输入，并行输出

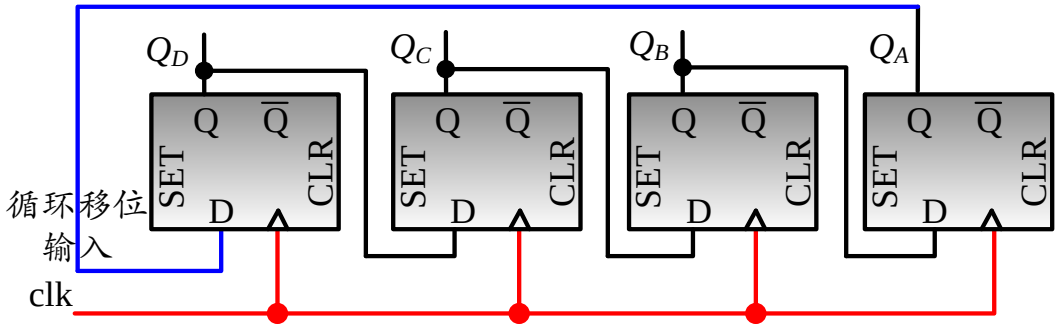
并行输入，串行输出

串行输入右移移位寄存器

使用 D 触发器构成串行输入的右移移位寄存器：

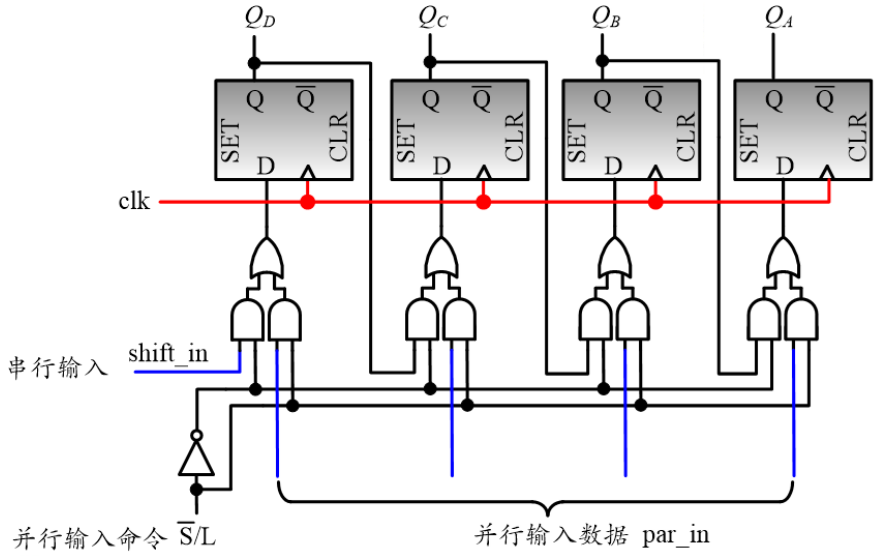


循环右移移位寄存器：

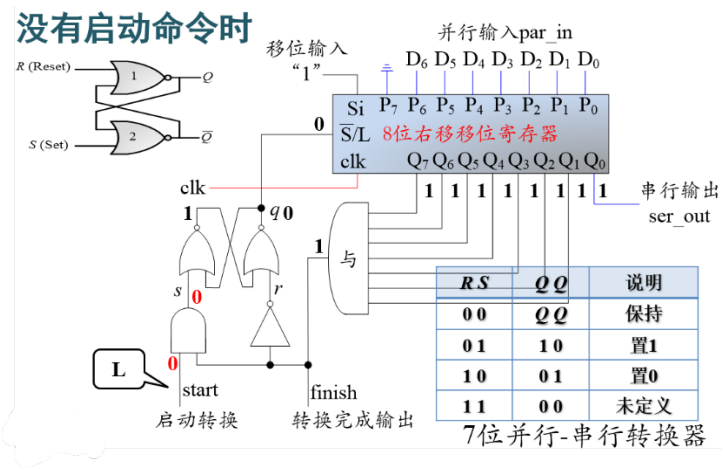


4.2 带并行输入的移位寄存器

数据输入方式：串行输入、并行输入

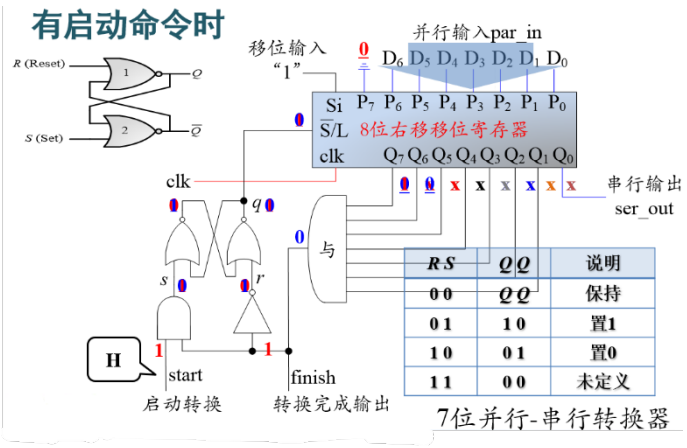


4.3 并行 - 串行转换器



经过若干时钟周期

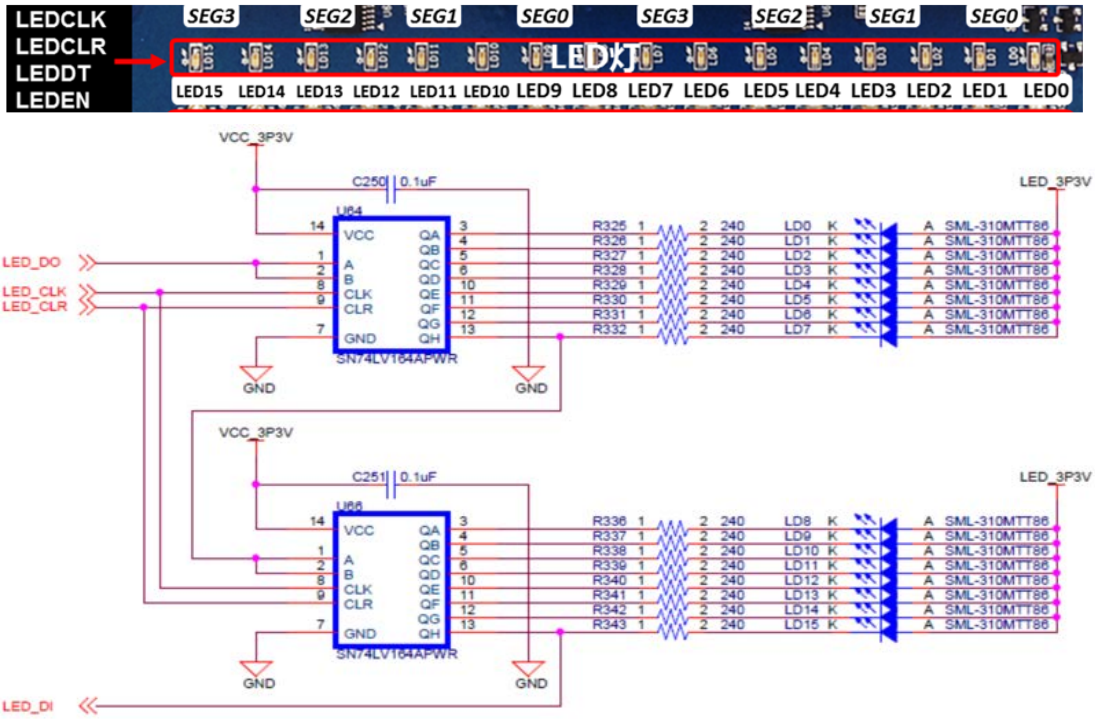
电路达到稳定状态



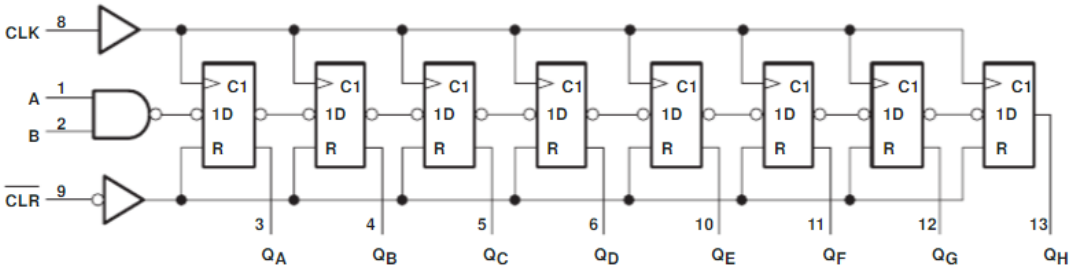
4.4 接口说明：主板 LED 灯

采用 2 个 74LV164A 构成 16 位串行输入并行输出移位寄存器

并行输出控制 16 个 LED 灯



- 74LV164A

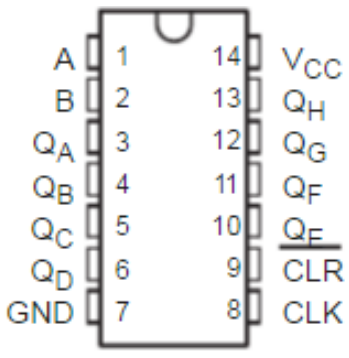


FUNCTION TABLE

INPUTS				OUTPUTS		
$\overline{CLR}$	CLK	A	B	Q_A	$Q_B \ldots Q_H$	
L	X	X	X	L	L	L
H	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	↑	H	H	H	Q_{An}	Q_{Gn}
H	↑	L	X	L	Q_{An}	Q_{Gn}
H	↑	X	L	L	Q_{An}	Q_{Gn}

Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.

Q_{An}, Q_{Gn} = the level of Q_A or Q_G before the most recent ↑ transition of the clock: indicates a 1-bit shift.



74HC164、74HCT164 是高速硅门 CMOS 器件，与低功耗肖特基型 TTL(LSTTL)器件的引脚兼容：74HC164、74HCT164 是 8 位边沿触发式移位寄存器，串行输入数据，然后并行输出。数据通过两个输入端(DSA 或 DSB)之一串行输入；任一输入端可以用作高电平使能端，控制另一输入端的数据输入。两个输入端或者连接在一起，或者把不用的输入端接高电平，一定不要悬空。

时钟(CP)次山低变高时，数据右移一位，输入到 Q0，Q0 是两个数据输入端（DSA 和 DSB)的逻辑与，它将在上升时钟边沿之前保持一个建立时间的长度.

主复位(MR)输入端上的一个低电平将使其它所有输入端都无效，同时非同步地清除寄存器，强制所有的输出为低电平。

#主板 LED 输出引脚约束

NET "LEDCLK" LOC = N26 | IOSTANDARD = LVCMOS33;

NET "LEDCLR" LOC = N24 | IOSTANDARD = LVCMOS33;

NET "LEDDT" LOC = M26 | IOSTANDARD = LVCMOS33;

NET "LEDEN" LOC = P18 | IOSTANDARD = LVCMOS33;

LEDCLK：74LV164A 的时钟

LEDCLR：清零

LEDDT：数据串行输入

LEDEN：控制 LED 电源，1 为使能

- LED 并行显示模块 M6：SPIO

15 位 LED 指示灯控制(IP Core)

逻辑实验的输出 LED 显示模块，相当于通用输入输出接口：GPIO

15 位用于 LED 指示控制，其余用于扩展，器件编号改为 U7

本课程用于调试显示和 CPU 的简单外设

基本功能：

输入 32 位二进制数据：P_Data

clk：时钟，

EN：输出使能，

Start：串行扫描启动，

rst：复位

串行输出：

led_clk=时钟，

led_sout=串行输出数据，

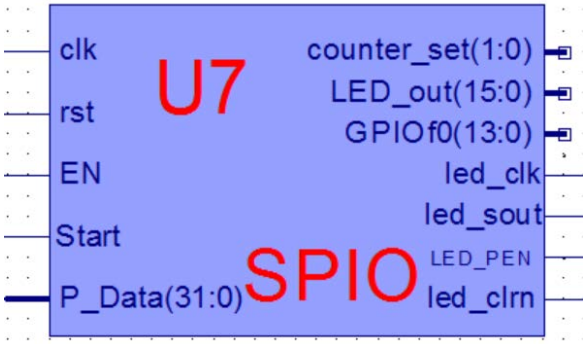
LED_PEN=使能，

led_clrn=清零

并行输出：LED_out、counter_set、GPIOf0

核模块符号文档：SPIO.sym

本实验提供 U7 的 IP 核



- LED 并行显示模块 IP 核端口信号

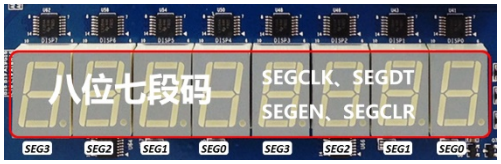
PIO/LED-GPIO IP 核端口信号

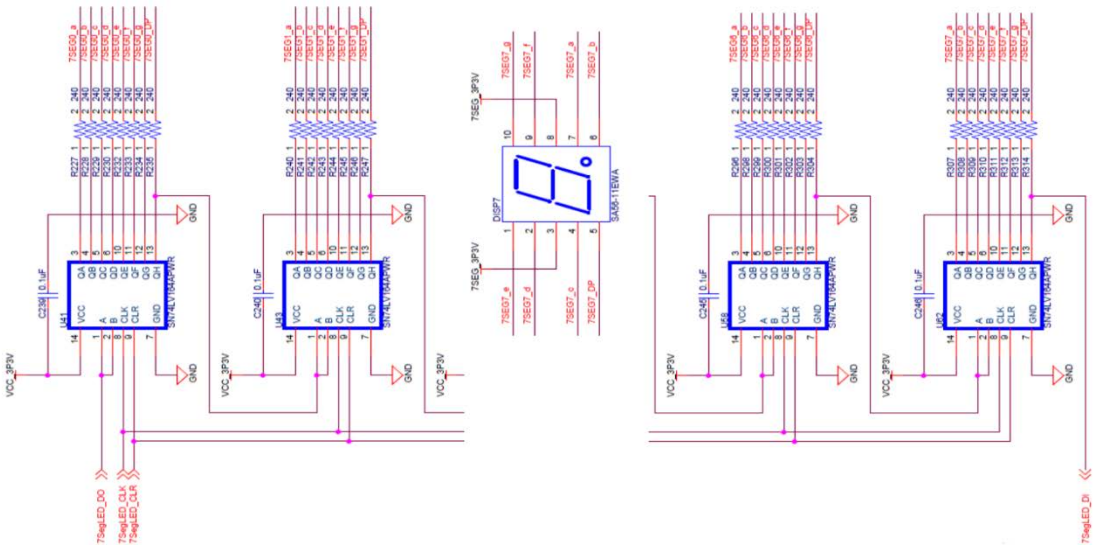
可作为 IP 核调用空文档：端口文档

```
module SPIO(
    input clk,
    input rst,
    input Start,
    input EN,
    input [31:0] P_Data,
    output reg[1:0] counter_set,
    output [15:0] LED_out,
    output wire led_clk,
    output wire led_sout,
    output wire led_clrn,
    output wire LED_PEN,
    output reg[13:0] GPIOf0
    );
endmodule
```

4.5 接口说明：主板七段数码管

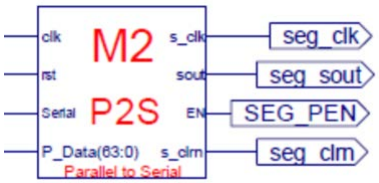
静态译码 74LS164 移位输出





共阴

通过 P2S 模块输出：P_Data[63:0]=SEGMENT[63:0]



#七段码移位输出引脚约束

NET "SEGCLK" LOC = M24 | IOSTANDARD = LVCMOS33;

NET "SEGCLR" LOC = M20 | IOSTANDARD = LVCMOS33;

NET "SEGDT" LOC = L24 | IOSTANDARD = LVCMOS33;

NET "SEGEN" LOC = R18 | IOSTANDARD = LVCMOS33;

SEGCLK：74LV164A 的时钟

SEGCLR：清零

SEGDT：数据串行输入

SEGEN：控制数码管电源，1 为使能

- 七段码显示器 IP 核 M3：SSeg7_Dev

8 位七段数码管显示器(IP Core)，逻辑实验的输出显示模块

本课程用于调试显示和 CPU 的简单外设，器件编号改为 U6

基本功能

输入 32 位二进制数据：Hexs

SW[0]=1，显示 8 位 16 进制数，SW[0]=0，显示七段码 LED 点阵

SW[0]=1 时，SW[1]=1 高 16 位，SW[1]=0 低 16 位，

flash 七码闪烁频率，由通用分频器 U8(Div[25])提供，

Start 串行扫描启动，point：七段小数点，LES：七段码使能，闪烁指示

串行输出：

seg_clk：时钟，

seg_out：串行七段显示数据，

SEG_PEN：使能，

seg_clrn：清零

核模块符号文档：SSeg7_Dev.sym

由实验二优化扩展，本实验提供 U6 的 IP 核

- 七段码显示器 IP 核端口信号

可作为 IP 核调用空文档：端口文档

```
module SPIO(
    input clk,
    input rst,
    input Start,
    input SW0,
    input flash,
    input [31:0]Hexs,
    input [7:0]points,
    input [7:0]LES,
    output seg_clk,
    output seg_sout,
    output SEG_PEN,
    output seg_clrn
    );
endmodule
```

五．实验内容与步骤

任务 1：设计 8 位带并行输入的右移移位寄存器

任务 2：设计跑马灯应用

5.1 设计 8 位带并行输入的右移移位寄存器

新建工程，工程名称用 ShfitReg8b，Top Level Source Type 用 HDL

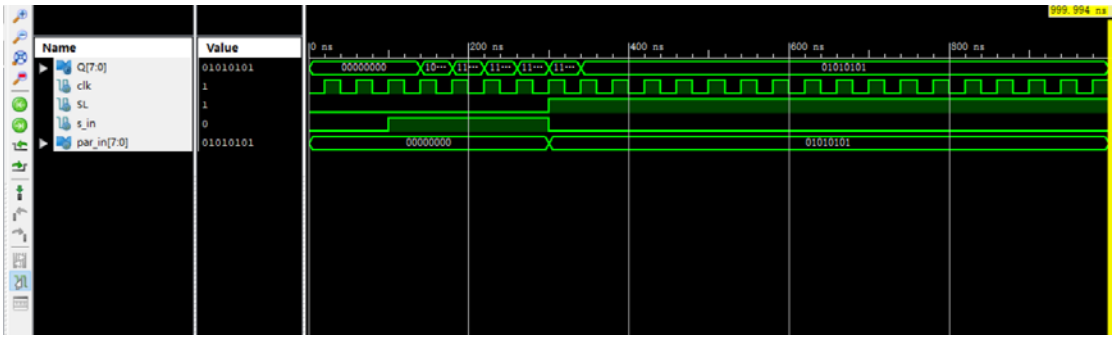
用结构化描述设计

```
module ShiftReg8b(
    input wire clk, SL, s_in,
    input wire[7:0] par_in,
    output wire[7:0] Q
    );
    wire[7:0] D;
    wire[7:0] OR0;
    wire[7:0] OR1;

    FD fd0(.C(clk), .D(D[0]), .Q(Q[0]));
    FD fd1(.C(clk), .D(D[1]), .Q(Q[1]));
    FD fd2(.C(clk), .D(D[2]), .Q(Q[2]));
    FD fd3(.C(clk), .D(D[3]), .Q(Q[3]));
    FD fd4(.C(clk), .D(D[4]), .Q(Q[4]));
    FD fd5(.C(clk), .D(D[5]), .Q(Q[5]));
    FD fd6(.C(clk), .D(D[6]), .Q(Q[6]));
    FD fd7(.C(clk), .D(D[7]), .Q(Q[7]));

    OR2 or0(.I0(OR0[0]), .I1(OR1[0]), .O(D[0]));
    OR2 or1(.I0(OR0[1]), .I1(OR1[1]), .O(D[1]));
    OR2 or2(.I0(OR0[2]), .I1(OR1[2]), .O(D[2]));
    OR2 or3(.I0(OR0[3]), .I1(OR1[3]), .O(D[3]));
    OR2 or4(.I0(OR0[4]), .I1(OR1[4]), .O(D[4]));
    OR2 or5(.I0(OR0[5]), .I1(OR1[5]), .O(D[5]));
    OR2 or6(.I0(OR0[6]), .I1(OR1[6]), .O(D[6]));
    OR2 or7(.I0(OR0[7]), .I1(OR1[7]), .O(D[7]));

    AND2 and00(.I0(SL), .I1(par_in[0]), .O(OR0[0]));
    AND2 and01(.I0(SL), .I1(par_in[1]), .O(OR0[1]));
    AND2 and02(.I0(SL), .I1(par_in[2]), .O(OR0[2]));
    AND2 and03(.I0(SL), .I1(par_in[3]), .O(OR0[3]));
    AND2 and04(.I0(SL), .I1(par_in[4]), .O(OR0[4]));
    AND2 and05(.I0(SL), .I1(par_in[5]), .O(OR0[5]));
    AND2 and06(.I0(SL), .I1(par_in[6]), .O(OR0[6]));
    AND2 and07(.I0(SL), .I1(par_in[7]), .O(OR0[7]));

    INV inv0(.I(SL), .O(nSL));

    AND2 and10(.I0(nSL), .I1(Q[1]), .O(OR1[0]));
    AND2 and11(.I0(nSL), .I1(Q[2]), .O(OR1[1]));
    AND2 and12(.I0(nSL), .I1(Q[3]), .O(OR1[2]));
    AND2 and13(.I0(nSL), .I1(Q[4]), .O(OR1[3]));
    AND2 and14(.I0(nSL), .I1(Q[5]), .O(OR1[4]));
    AND2 and15(.I0(nSL), .I1(Q[6]), .O(OR1[5]));
    AND2 and16(.I0(nSL), .I1(Q[7]), .O(OR1[6]));
    AND2 and17(.I0(nSL), .I1(s_in), .O(OR1[7]));
endmodule
```

激励代码:

```
initial begin
    clk = 0;
    SL = 0;
    s_in = 0;
    par_in = 0;
    #100;
    SL = 0;
    s_in = 1;
    par_in =0;
    #200;
    SL = 1;
    s_in = 0;
    par_in = 8'b0101_0101;
    #500;
end

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

仿真波形：



5.2 设计跑马灯应用

新建工程，工程名称用 MyMarquee，Top Level Source Type 用 HDL

用结构化描述设计，

调用 ShfitReg8b，

调用分频模块，用 1s 作为移位寄存器驱动时钟，

调用显示模块，

调用 CreateNumber 模块，

```
module top(
    input wire clk,
    input wire[4:0] SW,
    output wire[7:0] SEGMENTS,
    output wire[3:0] AN,
    output wire[7:0] LED,
    output wire BTN_X);
    assign BTN_X=0;
    wire[1:0] btn;
    wire[15:0] num;
    wire[31:0] clk_div;
    clkdiv m0(.clk(clk), .rst(1'b0), .clkdiv(clk_div));
    clk_1s m1(.clk(clk), .clk_1s(clk_1s));
    pbdebounce m2(.clk_1ms(clk_div[17]),.button(SW[0]),.pbreg(btn[0]));
    pbdebounce m3(.clk_1ms(clk_div[17]),.button(SW[1]),.pbreg(btn[1]));
    CreateNumber m5(.btn({2'b00,btn[1:0]}), .sw(4'b0000), .num(num));
    ShiftReg8b m6(
        .clk(clk_1s),
        .rst(1'b0),
        .SL(SW[2]),
        .s_in(SW[3]),
        .ifCirc(SW[4]),
        .par_in(num[7:0]),
        .Q(LED)
    );
    disp_num m7(
        .clk(clk),
        .HEXS(num),
        .LES(4'b0000),
        .points(4'b1111),
        .RST(1'b0),
        .AN(AN),
        .Segment(SEGMENTS)
    );
endmodule
```

- 下载验证

用 sw[0]和 sw[1]作为 regA 和 regB 的按键自增控制输入

sw[2]=1，并行输入，将{RegA,RegB}赋给移位寄存器

sw[2]=0，串行/循环右移移位

sw[4]作为移位寄存器的模式选择：

sw[4]=0，串行右移，串行输入值为 sw[3]

sw[4]=1，循环右移

8 位的移位寄存器的值用 LED 灯表示

- ucf 引脚约束

```
net "clk" loc=AC18 | iostandard=LVCMOS18;
net"AN[0]"loc=AD21 | iostandard=LVCMOS33;
net"AN[1]"loc=AB21 | iostandard=LVCMOS33;
net"AN[2]"loc=AC21 | iostandard=LVCMOS33;
net"AN[3]"loc=AC22 | iostandard=LVCMOS33;
net"SEGMENTS[0]"loc=AB22 | IOSTANDARD=LVCMOS33;
net"SEGMENTS[1]"loc=AD24 | IOSTANDARD=LVCMOS33;
net"SEGMENTS[2]"loc=AD23 | IOSTANDARD=LVCMOS33;
net"SEGMENTS[3]"loc=Y21  | IOSTANDARD=LVCMOS33;
net"SEGMENTS[4]"loc=W20  | IOSTANDARD=LVCMOS33;
net"SEGMENTS[5]"loc=AC24 | IOSTANDARD=LVCMOS33;
net"SEGMENTS[6]"loc=AC23 | IOSTANDARD=LVCMOS33;
net"SEGMENTS[7]"loc=AA22 | IOSTANDARD=LVCMOS33;

net"BTN_X"loc=W16  | iostandard=LVCMOS18;
net"SW[0]"loc=V18  | iostandard=LVCMOS18;
net"SW[1]"loc=V19  | iostandard=LVCMOS18;

net"SW[2]"loc=AE13 | IOSTANDARD=LVCMOS15;
net"SW[3]"loc=AF8  | IOSTANDARD=LVCMOS15;
net"SW[4]"loc=AE8  | IOSTANDARD=LVCMOS15;

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

- 实验结果

可以看到 LED 灯在闪烁中显示出七段数码管上的数字，符合预期要求。

六．实验心得与体会

本次实验的主要内容为移位寄存器的设计与应用，移位寄存器还有跑马灯都比较简单，而主板 LED 灯和 8 位七段数码管的接口设计比较难，需要用到并行串行转换，并且得理解串行输入的思想，经过这次实验，我对移位寄存器的相关知识还有应用都有了一定的了解，这次实验是最后一次实验，经过一个学期的实验，认识到自己还有很多不足，需要继续加强学习，希望可以在期末考试中取得一个满意的成绩。

个人生活照: