电路原理部分

测量误差

相对误差分类

某值相对误差：$\delta = \frac{|测量值-实际值|}{某值} \times 100\%$
引用相对误差：$\gamma = \frac{|测量值-实际值|}{仪表量程（上限-下限）} \times 100\%$
最大引用误差：$\gamma_m = \frac{\Delta_m}{x_m} \times 100\%$，$\gamma \leq a\%$，其中a为等级指数

直读式数字表的误差表示

直读式数字表的误差表示=被测量大小的相对量+固定量+显示误差

$$\Delta = \pm(a\%)x \pm 几个字$$

$$\Delta = \pm(a\%)x \pm (b\%)x_m$$

$$\Delta = \pm(a\%)x \pm (b\%)x_m \pm 几个字$$

其中，a - 误差相对项系数
x - 被测量的指示值
b - 误差固定项系数
x_m - 仪表满刻度值，即量程

数字表的显示位数：

根据最高位是否能完整显示0 – 9来定义。若可以完整显示，记完整位；否则，计半位。

0000~1999 → 三位半 → 2000字
0000~8999 → 三位半 → 9000字
0000~9999 → 四位 → 10000字

1个字的精度/分辨率：

(1/总字数)x_m

字误差：字数乘以分辨率

例3 一个准确度等级为0.02的四位半数字电压表，其2V量程档的准确度为$\Delta_x = \pm(0.02\%)x \pm 2$个字，请问，2V量程测量2V和0.2V时引起的误差各是多少？

该四位半数字电压表显示范围为00000~19999 → 2个字误差：$0.01\%x_m$

$$\Delta_x = \pm 0.02\%x \pm 0.01\%x_m \qquad \gamma_x = \frac{\Delta_x}{x} = \pm 0.02\% \pm 0.01\%\frac{x_m}{x}$$

	x_m	x	$0.02\%x$	$0.01\%x_m$	Δ_x	γ_x
测量2V	2V	2V	0.4mV	0.2mV	$\pm 0.6mV$	±0.03%
测量0.2V	2V	0.2V	0.04mV	0.2mV	$\pm 0.24mV$	±0.12%

要减小测量误差，须选择合适的量程使被测量大于该量程的**三分之二**。
⚠实际测量时，满量程测量很容易引起超量程，使得仪表工作不正常。

有效数字

从左边第一个非0位到最后一位所包含的数字，最后一位为可疑数字

仪器误差数据

数字万用表（3字半）

电阻测量：

量程	分辨率	准确度
600Ω	0.1Ω	±(0.8%读数+3字)
6KΩ	1Ω	
60KΩ	10Ω	
600KΩ	100Ω	
6MΩ	1KΩ	
60MΩ	10KΩ	±(2.0%读数+5字)

电容测量:

量程	分辨率	准确度
9.999nF	0.001nF	±(4.0%读数+20字)
99.99nF	0.01nF	±(3.0%读数+5字)
999.9nF	0.1nF	
9.999μF	0.001μF	
99.99μF	0.01μF	
999.9μF	0.1μF	
9.999mF	0.001mF	±(5.0%读数+10字)
99.99mF	0.01mF	

电压测量: (仪表输入阻抗10MΩ)

量程	分辨率	准确度
600mV	0.lmV	±(0.5%读数+2字)
6V	0.001V	
60V	0.01V	
600V	0.1V	
1000V	1V	±(0.8%读数+2字)

电流测量：（仪表输入阻抗6mA/600uA/60uA是99Ω，60mA/600mA/10A是0.99欧姆，交流电流档内阻0.99Ω）

量程	分辨率	准确度
60 μA	0.01 μA	±(0.8%读数＋10字)
600 μA	0.1 μA	
6 mA	1 μA	
60 mA	10 μA	±(1.0%读数＋10字)
600 mA	100 μA	
10 A	10 mA	±(2.0%读数＋5字)

仪器台上直流电压表

1. 量程/内阻：200mV/524.9k、2V/502.5k、20V/500.2k，200V/500k
2. 测量精度为0.5级

仪器台上直流电流表

1. 量程/内阻：2mA/51、20mA/5.1、200mA/0.51，2A/0.05
2. 测量精度为0.5级

数电实验

课程使用芯片介绍：

1. 74LS161二进制加法计数器

输入									触发器状态			
CP	$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	D	C	B	A	Q_3	Q_2	Q_1	Q_0
×	0	×	×	×	×	×	×	×	0	0	0	0
↑	1	0	×	×	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
↑	1	1	1	1	×	×	×	×	4位二进制加法计数			
×	1	1	0	×	×	×	×	×	保持功能			
×	1	1	×	0	×	×	×	×	保持功能			

CP：输入的信号；

CR：异步清零操作，低电平触发，优先级最高；

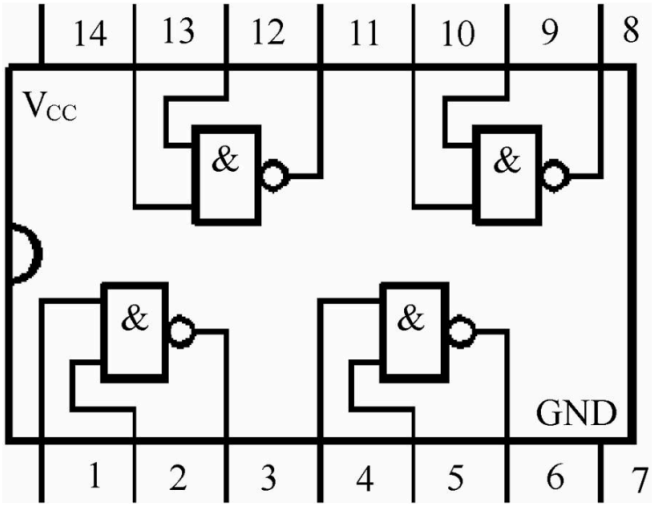
LD：置数模式（即通过输入数字）优先级第二高，低电平触发，写入3456引脚读取到的数，同步置数（即读取到CP有上升沿）；

CTp和CTt：有低电平则进入保持

2. 74LS00与非门芯片:

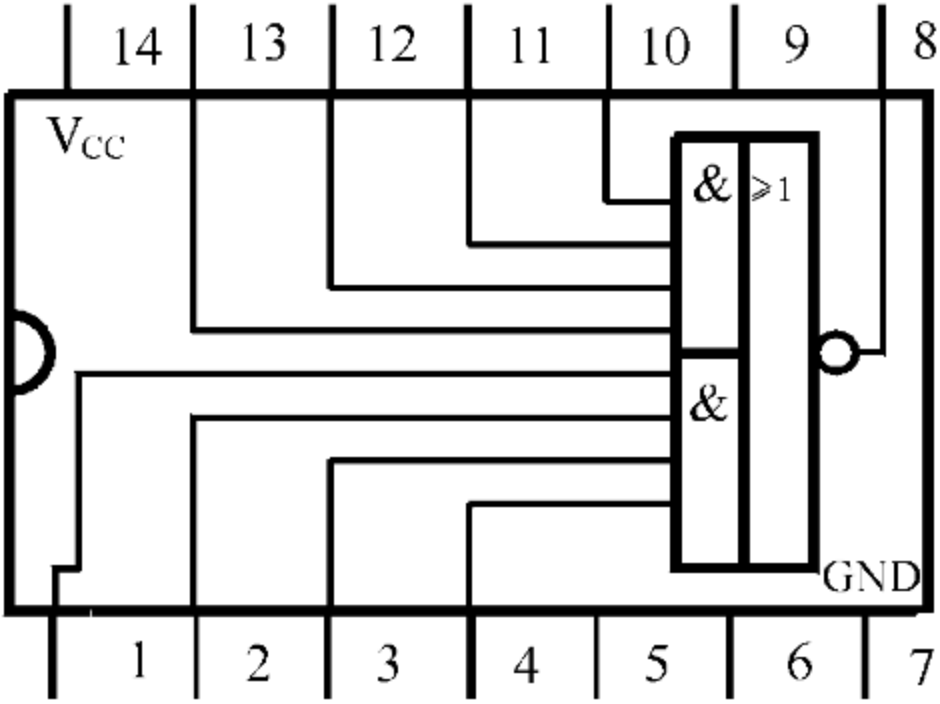
74LS00与非门

芯片引脚

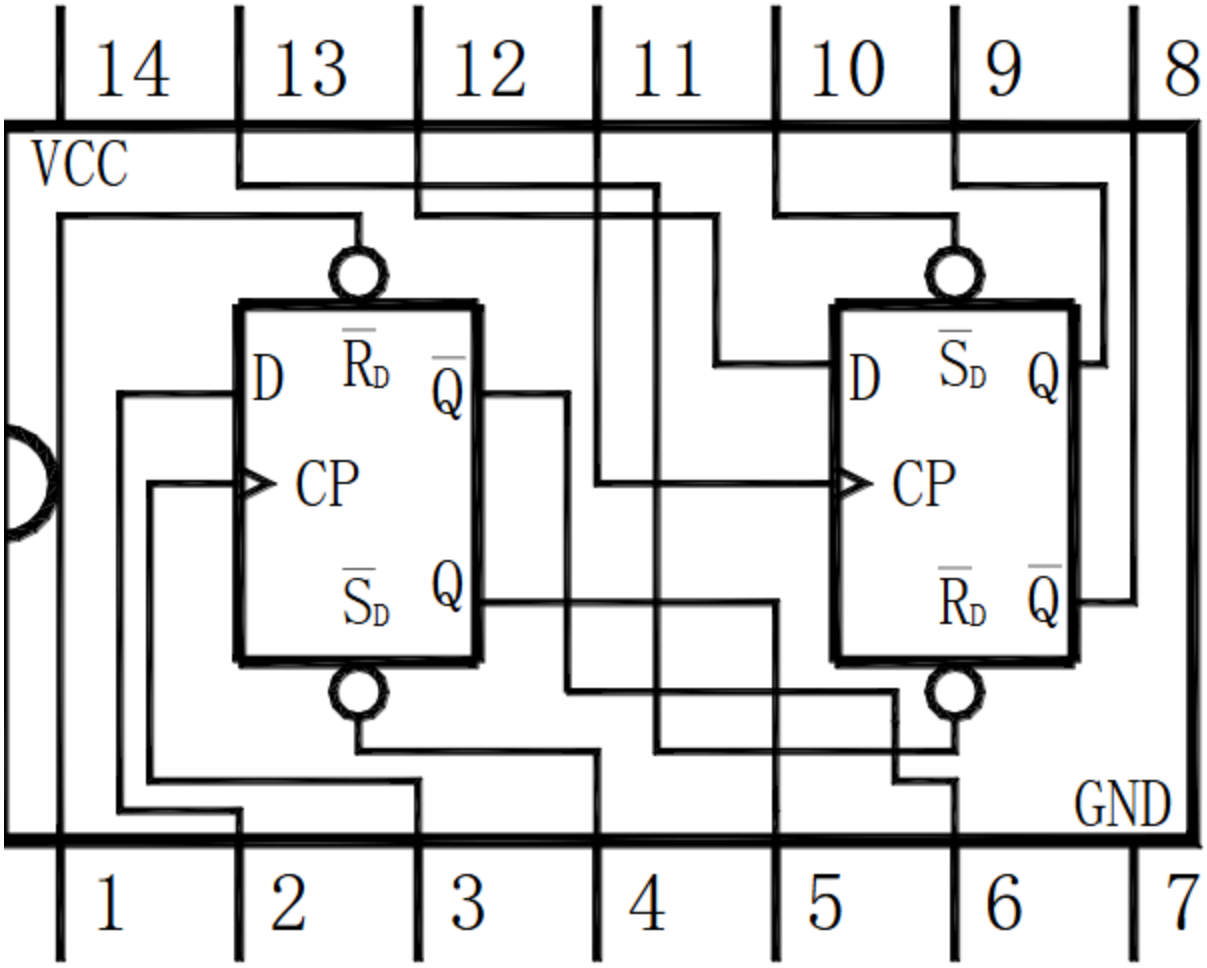


3. 74LS55与或非门芯片

$$L = \overline{A_1A_2A_3A_4 + B_1B_2B_3B_4}$$



4. 74LS74型双D触发器

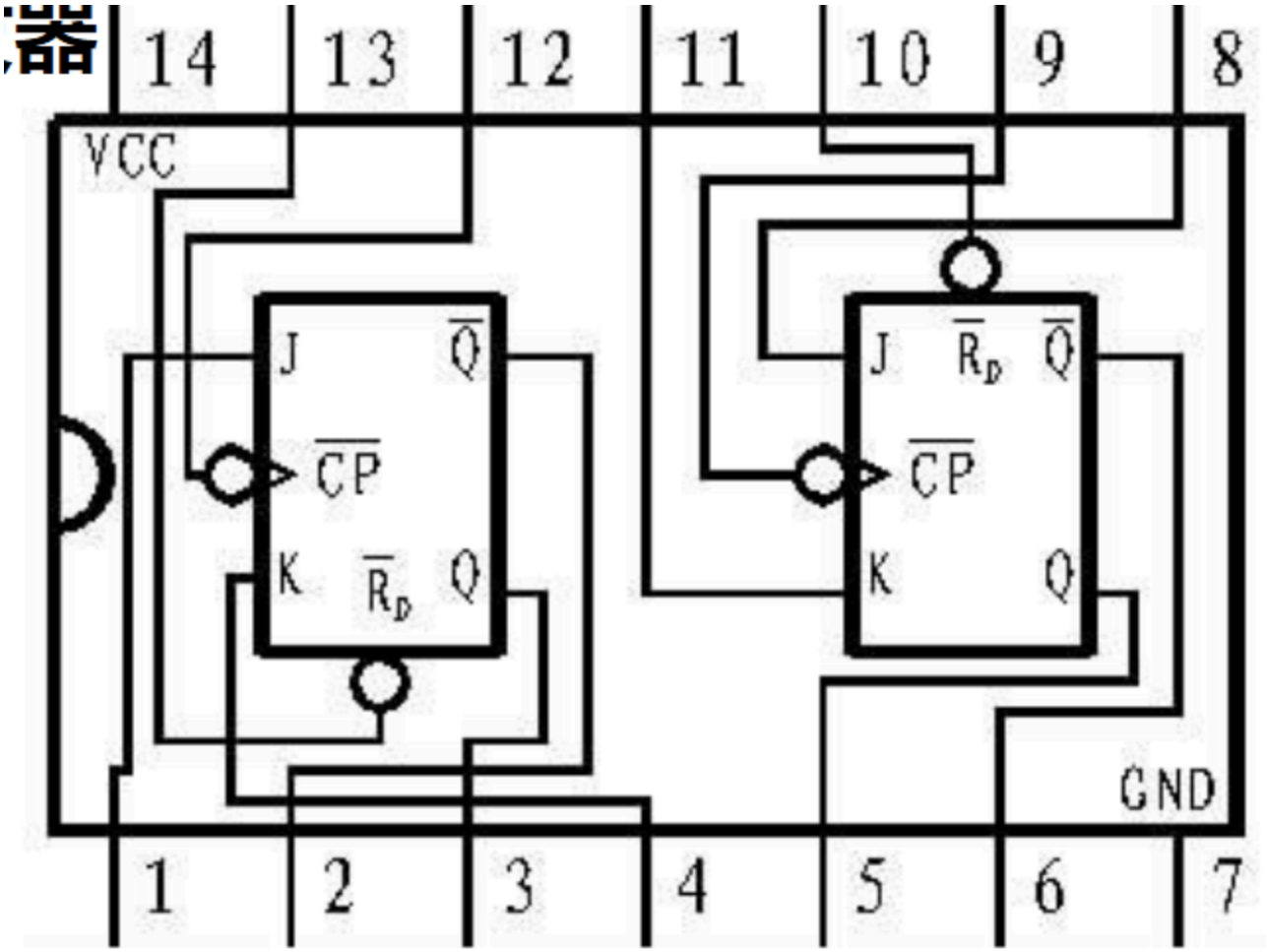


74LS74双D触发器

1. 异步置位、复位
2. 逻辑功能：$Q^{n+1} = D$
3. 上升沿触发

	$\overline{S}_D$	$\overline{R}_D$	CP	D	Q	$\overline{Q}$
置位	0	1	×	×	1	0
清零	1	0	×	×	0	1
（禁用，不确定）	0	0	×	×	1^*	1^*
（D触发器）	1	1	↑	1	1	0
	1	1	↑	0	0	1
（保持）	1	1	0	×	Q_0	$\overline{Q}_0$

5. 74LS107双JK触发器



1. 带清除
2. $Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$
3. 主从触发，下升沿触发

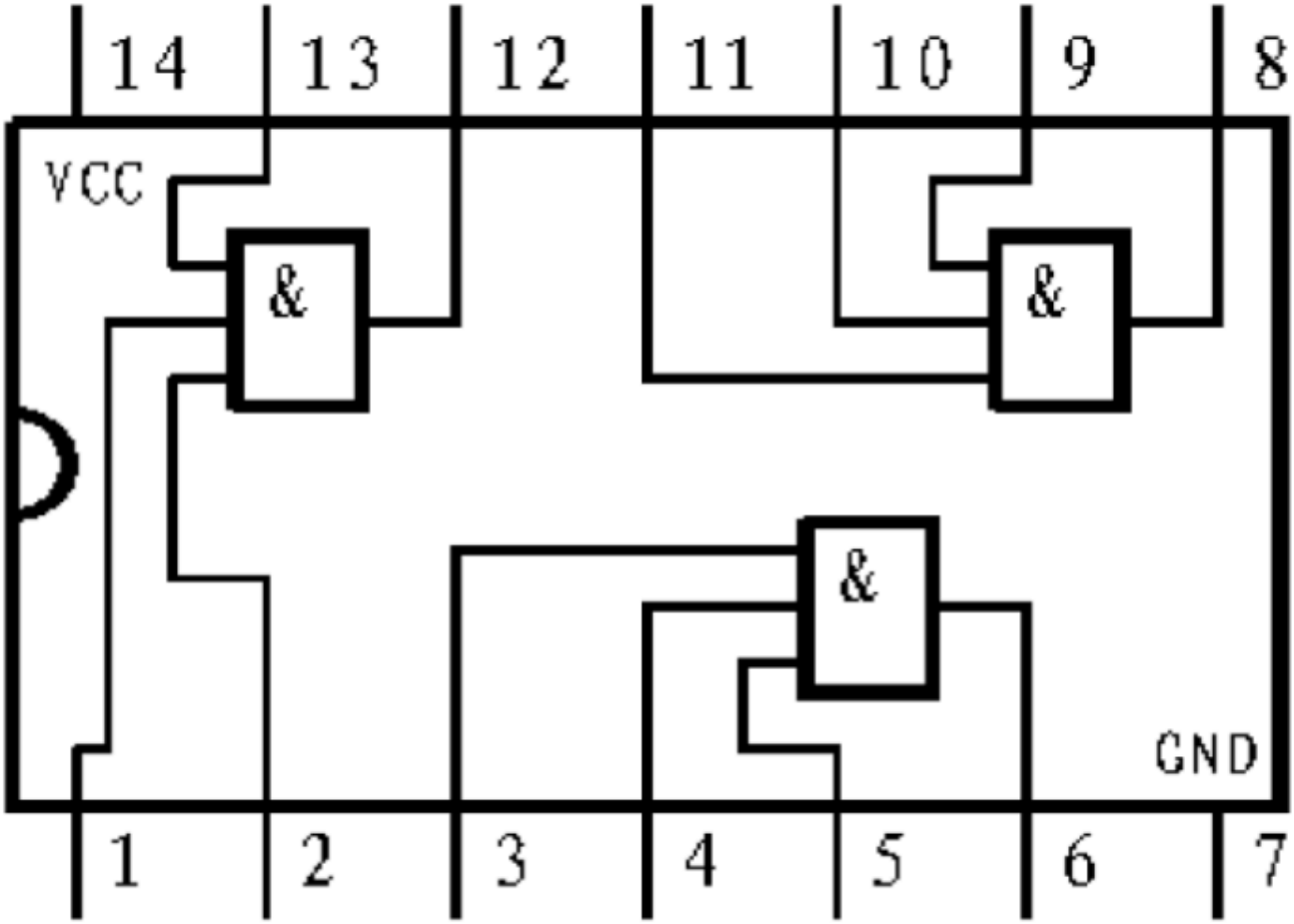
74LS107没有直接置位端，初态“1”可用J端置位

$\overline{R}_D$	$\overline{CP}$	J	K	Q	$\overline{Q}$
0	×	×	×	0	1
1	↓	0	0	Q_0	$\overline{Q}_0$
1	↓	1	0	1	0
1	↓	0	1	0	1
1	↓	1	1	翻转	
1	1	×	×	Q_0	$\overline{Q}_0$

!!!

数码管显示数字原理: 用四个二进制的高低电平表示一个不大于10的数，四个位由高到低分为Q_D、Q_C、Q_B、Q_A。

6. 74LS11三输入与门芯片



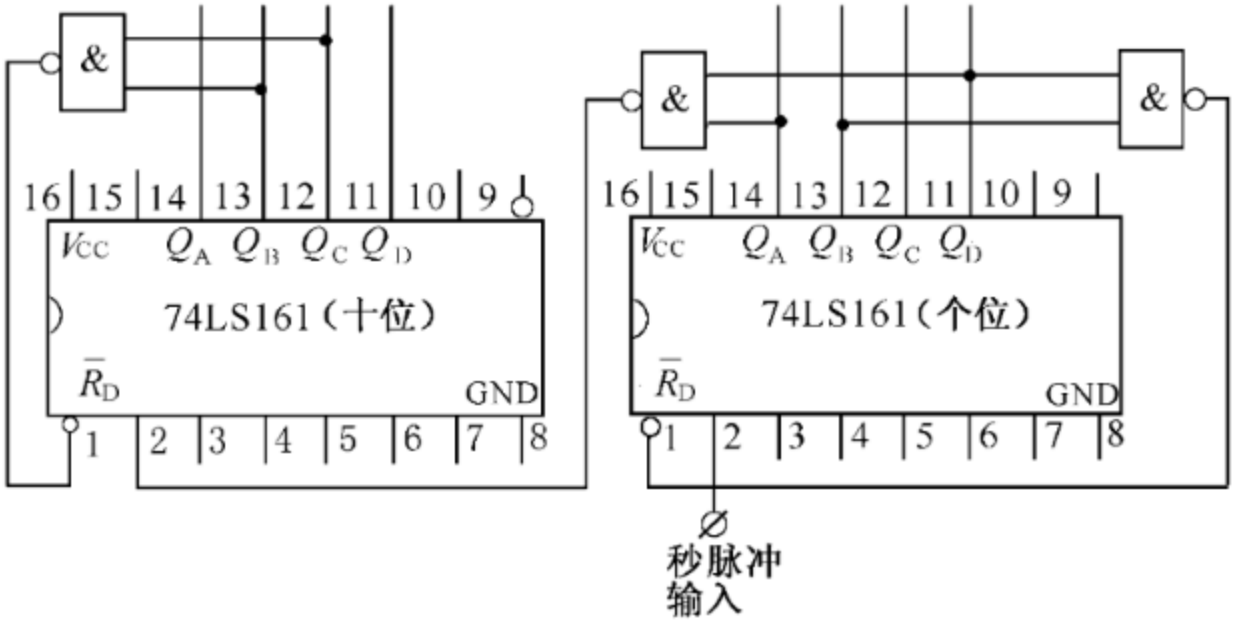
Quartus文件名后缀

后缀	全称
.bdf	Block Design File
.bsf	Block Symbol File
.qpf	Quartus Projct Flie
.vhd	VHDL Design File
.vwf	Waveform.File
.sof	编译后产生的、将下载到FPGA中的文件的

数字式计时器电路：

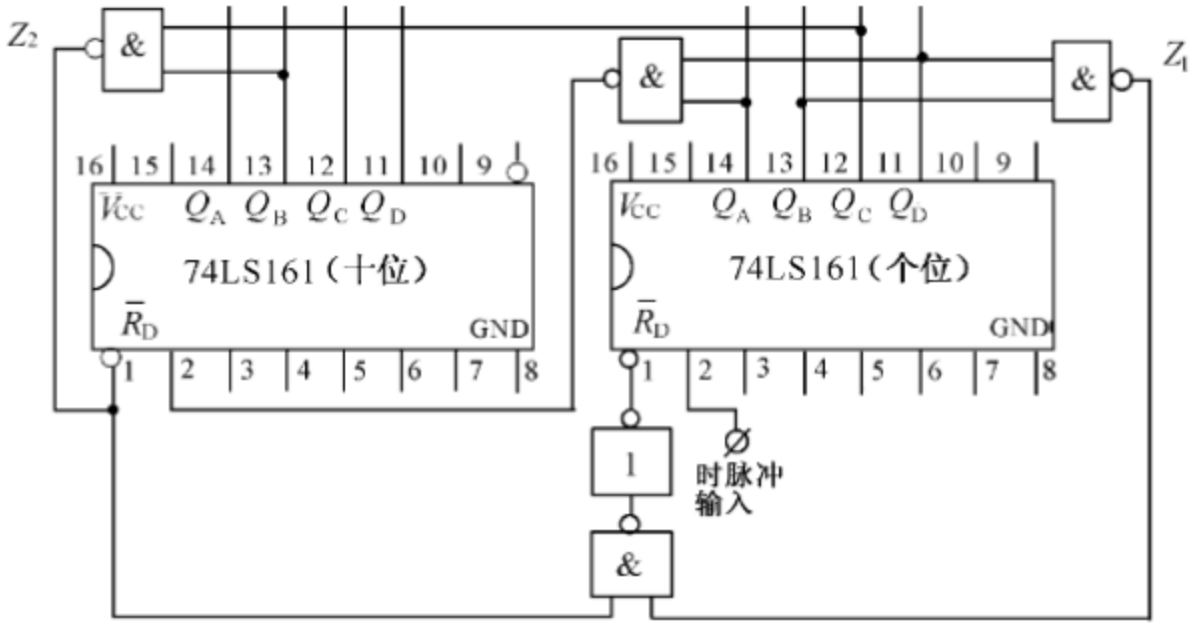
1. 60进制：

分为个位数码管和十位数码管，个位数码管10进制，十位数码管6进制。个位计时器当计数到10（二进制1010）时，CR置0，所以就需要一个与非门对Q_B和Q_D取与非。



2. 24进制：

个位数码管部分基本不改动，但十位的CR需要在计时达到24（即二进制0100 0010）时置0。



全加器

$$
\begin{aligned}
S_i &= \bar{A}\bar{B}C_{i-1} + \bar{A}B\overline{C_{i-1}} + A\bar{B}\overline{C_{i-1}} + ABC_{i-1} \\
&= (\bar{A}B + A\bar{B})\overline{C_{i-1}} + (\bar{A}\bar{B} + AB)C_{i-1} \\
&= (A \oplus B)\overline{C_{i-1}} + \overline{(A \oplus B)}C_{i-1} \\
&= A \oplus B \oplus C_{i-1} \\
&= S_i' \oplus C_{i-1} \\
&= S_i'\overline{C_{i-1}} + \overline{S_i'}C_{i-1}
\end{aligned}
\qquad S_i' = A \oplus B
$$

$$
\begin{aligned}
C_i &= (A+B)C_{i-1} + (A+B)\overline{S'_i} \\
&= (A+B)\cdot(C_{i-1} + \overline{S'_i}) \\
&= \overline{\overline{A+B} + \overline{C_{i-1}\,\overline{S'_i}}} \\
&= \overline{\bar{A}\bar{B} + \overline{C_{i-1}}S'_i}
\end{aligned}
$$

历年题

注意：记得检查自启动！！！

1. 首先需要做出110->101->100->011->010->110五个三位二进制循环，输入1HZ信号，将三位输出接入数码管 再做出三位奇偶校验器并接入前面循环的输出，接到发光二极管上，要求在4和2时亮，6, 5, 3时灭 最后将时钟脉冲接入示波器CH1，奇偶校验器结果接入示波器CH2，并将时钟换成1024HZ
   验收要求是
   (1) 1HZ时观察6，5，4，3，2循环是否正确，奇偶校验器是否正常工作
   (2) 1024HZ时，示波器上半部分显示CH1波形，下班部分显示CH2波形，要求显示两个周期左右，

保证波形稳定

做法：对减法计数器取反可得加法计数器，用161反馈置数法做出1——5循环再取反，奇偶校验器不用直接列异或表达式，可以只考虑出现的情况画卡诺图列真值表

2. 时序电路设计，要求给定输入X与时钟信号CP，输出为Q0Q1Q2，X=0时输出4→5→6→4……的序列，X=1时输出0→1→2→0……的序列

   （1）X接逻辑开关，CP接1Hz脉冲，用数码管显示结果。

   （2）CP接1024Hz脉冲，用示波器显示稳定的图像，要求CH1接Q1，CH2接Q0。

   做法：用161做出3进制计数器，再将X取反接到数码管。这个思路很妙，不知道是怎么想到的。
3. 电路实现010-011-100-101-110（搭完此电路，可以先用LED灯给老师验收） 以上述作为输出，进一步搭建奇偶判断电路或者能否被3整除的电路，用示波器观测输入与输出波形

   做法：常规的反馈计数法的加法计数器
4. 利用JK触发器和与非门芯片，实现信号发生器，其能重复输出10011101信号，并完成连线

   做法：先搭000-111的八进制计数器，再画卡诺图列出输出信号的逻辑表达式进行译码。

VHDL语言

基本结构

```
LIBRARY ieee;                                    --库调用说明
USE ieee.std_logic_1164.all;                     --程序包调用说明
USE ieee.std_logic_unsigned.all;
ENTITY decode IS                                 --实体开始
    PORT
    (
          A : IN STD_LOGIC_VECTOR(4 DOWNTO 1); --端口说明
          Z : OUT STD_LOGIC
    );                                           --”)”前无”;”
END decode;                                      --实体结束
ARCHITECTURE decode_architecture OF decode IS    --结构体开始
BEGIN
    Z<='1' WHEN (A>=2 AND A<=10)                 --功能描述
    ELSE '0';
END decode_architecture;                         --结构体结束
```

①库(LIBRARY)：用以存储预先完成的程序包和数据集合体

②程序包(PACKAGE)：声明在设计或实体中将用到的常数、数据类型、元件及子程序等

③实体(ENTITY)：定义本设计的输入/输出端口

④结构体(ARCHITECTURE)：定义实体的实现，即电路的具体描述

- LIBRARY:LIBRARY 语句声明库，常用ieee；
- PACKAGE：USE 语句声明程序包，以下为ieee标准库所包含的程序包：

程序包	预定义内容
std_logic_1164	定义了std_logic，std_logic_vector等数据类型
numeric_std	定义了一组基于std_logic_1164中定义的类型的算术运算，如“+”、“-”，SHL、SHR等
std_logic_arith	定义了有符号与无符号类型，及基于这些类型的算术运算
std_logic_signed	定义了基于std_logic与std_logic_vector类型的有符号的算术运算
std_logic_unsigned	定义了基于std_logic与std_logic_vector类型的无符号的算术运算

- ENTITY： 使用 `ENTITY decode IS` 定义，定义结束用 `END decode` 实体,用于确定设计的输入/输出；
- ARCHITECTURE：语法如下图：用于规定实体的内部结构或实体行为，相当于main函数；

```
ARCHITECTURE decode_architecture OF decode IS  --结构体开始
BEGIN
    Z<='1' WHEN (A>=2 AND A<=10)          --功能描述
    ELSE '0';
END decode_architecture;                  --结构体结束
```

语法

标识符

由英文字母大小写、数字和“_”组成

> **Warning**
>
> ***大小写不区分，关键字不能做标识符;一定以英文字母开头;下划线不能连用也不能放结尾***

数据对象

分类：端口（PORT）、信号（SIGNAL）、变量（VARIALBE）、常量（CONSTANT）

- PORT：语法如下：

```
    PORT
    (
        A : IN STD_LOGIC_VECTOR(4 DOWNTO 1); --端口说明
        Z : OUT STD_LOGIC
    );                                       --”)”前无”;”
```

端口模式：说明信号的流向

- **IN** 定义输入端口，信号可以读取，但不能给输入端口赋值
- **OUT** 定义输出端口，信号不能在内部反馈使用
- **INOUT** 定义双向端口，信号可以输入实体，也可以从实体输出
- **BUFFER** 定义缓冲端口，它是一个输出端口，但信号同时也可以在内部反馈使用

- SIGNAL：

数据对象：端口、信号、变量和常量等，都是数据对象

信号：全局量

1. 主要用于实体、结构体或程序包之间的信息交流，相当于电路中连接元件的导线，是实际的信号
2. 定义格式：SIGNAL 信号名: 数据类型 [:=初始值];

   例： SIGNAL count: std_logic_vector(3 DOWNTO 0) :="0000";
3. 信号赋值语句语法格式：信号名<=表达式;

   例： count <=a+b;

- VARIABLE：

变量：一个名称

1. 不是实际的信号，仅是为书写方便而引入的一个名称
2. 定义格式

   VARIABLE 变量名: 数据类型 [:=初始值];

   例： VARIABLE a: integer :=0;
3. 变量赋值语句语法格式

   变量名:=表达式;

   例： a:=b+c;

- CONSTANT：

常量：全局量

1. 在设计描述中保持某一规定类型的特定值**不变的量**
2. 定义格式

   CONSTANT 常量名: 数据类型:=表达式;

   例： CONSTANT mod: integer:=6;

数据类型

标准定义的数据类型：

1. **布尔量(Boolean)**：取值为**FALSE**和**TRUE**定义格式
2. **位(Bit)**：取值只能是**0**和**l**
3. **位数组(Bit_vector)**：基于位（Bit）的数组
4. **字符(Character)**：用单引号括起来的ASCII字符，如‘A’、‘-’、‘1’或‘0’
5. **字符串(String)**：字符构成的一维数组，用双引号括起来，如“ERROR”或B“11010011”
6. **整数(Integer)**：范围从-(231-1)到+(231-1)

IEEE预定义的数据类型：

1. **std_logic**：标准逻辑位，取值‘0’（强0）、‘1’（强1）、‘Z’（高阻态）、‘X’（强未知的）、‘W’（弱未知的）、‘L’（弱0）、‘H’（弱1）、‘-’（忽略）、‘U’（未初始化的）
2. **std_logic_vector**：标准逻辑位数组，是std_logic类型数据的数组

   例： SIGNAL count: std_logic_vector(3 DOWNTO 0) :=“0000”;

运算符

算术运算符

操作符	功能	操作符	功能
+	加	REM	取余
-	减	+	正号
*	乘	-	负号
/	除	**	乘方
MOD	取模	ABS	取绝对值

关系运算符

操作符	功能	操作符	功能
=	等于	>	大于
/=	不等于	<=	小于等于
<	小于	>=	大于等于

逻辑运算符

操作符	功能	操作符	功能
NOT	逻辑非	NOR	逻辑或非
AND	逻辑与	XOR	逻辑异或
NAND	逻辑与非	XNOR	逻辑异或非
OR	逻辑或		

连接运算符

连接运算符 ‘&’ ：将多个对象或数组连接成一个更大的数组

例： o1<= ‘0’& a(7 DOWNTO 1);
o1<= a(7 DOWNTO 1) &‘0’;

--将a右移一位赋给o1
--将a左移一位赋给o1

注释：以“--”开头，到本行结束为止

注释语法

-- 开头，到本行结束为止

属性

关于实体、结构体、类型及信号的一些特征

数值类属性

用于返回数组、块等数据的有关值，表示方法如’ left(左边界)，’ right(右边界)，’ low(下边界)，’ high(上边界)，’ length(数组长度)等；

例： dd: IN std_logic_vector(8 DOWNTO 0);

dd’left=8; dd’right=0; dd’low=0; dd’high=8; dd’length=9;

函数类属性

用来返回有关信号行为功能的信息；

例： 已知时钟信号clk，则

clk'event AND clk='1'表示时钟的上升沿；

clk'event AND clk='0'表示时钟的下降沿。

顺序语句

分类： IF_THEN_ELSE 、 CASE_WHEN 、 FOR_LOOP 、 NULL

IF_THEN_ELSE：

```
IF 条件 THEN
    顺序处理语句1;
ELSE
    顺序处理语句2;
END IF;
```

→ 二选择IF语句

单IF语句、多选择IF语句、嵌套结构IF语句等

⚠注意

1. 每个IF语句都有一条END IF语句与其对应；
2. 在多选择IF语句和嵌套结构IF语句中，每个END IF语句只与其前面最近的IF语句搭配。

CASE_WHEN：

```
CASE 控制表达式 IS
    WHEN 条件表达式1
    => 顺序处理语句1;
    ……
    WHEN 条件表达式n
    => 顺序处理语句n;
END CASE;
```

例：

```
CASE din IS
    WHEN “00” =>
    dout<="00";
    WHEN "01" =>
    dout<="01";
    WHEN "10" =>
    dout<="01";
    WHEN "11" =>
    dout<="10";
END CASE;
```

CASE_WHEN语句根据满足的条件直接选择多项顺序语句中的一项执行

条件表达式的4种表示形式：

① 单个值　　WHEN 值 => 顺序处理语句；

② 多个值相或　WHEN 值|值|……|值 => 顺序处理语句；

③ 一个连续的整数范围

WHEN 值 TO 值 => 顺序处理语句；

④ 其他取值

WHEN OTHERS => 顺序处理语句；

FOR_LOOP：

```
[标号:] FOR 循环变量 IN 循环范围 LOOP
        顺序处理语句；
    END LOOP [标号] ；
```

FOR_LOOP语句与高级语言的循环语句一样，使程序按照一定的规则重复执行

例：

```
FOR i IN 0 TO 31 LOOP
    temp:=temp XOR din(i);
END LOOP;
```

NULL：

NULL是个空操作语句，执行NULL语句使程序走到下一个语句

例：

```
CASE sel IS
        WHEN 0 | 7 => q<=NOT d;
        WHEN OTHERS => NULL;
END CASE;
```

并行语句

进程语句是个复合语句，由一段程序构成，内部的所有语句都是顺序执行的

```
[标号:] PROCESS（敏感信号表）
[声明区];    --此处声明变量、数据类型等用于进程中的局部声明
        BEGIN                          --进程开始
            顺序语句；
            ……
            顺序语句；
        END PROCESS [标号];            --进程结束
```

各个进程是并行执行的

敏感信号

Conception：进程中一些被关注的信号，只有当其中的某个信号变化时，该进程才被执行；

```
PROCESS(clk,rst)                                    --进程开始
    BEGIN
    IF(rst='0') THEN q<="0000";                     --异步清零
    ELSIF(clk'event AND clk='1') THEN q<=q+1;
                                        --clk上升沿计数加1
    END IF;
END PROCESS;                                        --进程结束
```

异步清零4位二进制计数器

并行信号赋值

- 信号赋值语句在进程内部使用时，它以顺序语句的形式出现；
- 当信号赋值语句在进程之外使用时，它又是并行语句的形式出现。
- 并行信号赋值语句有3种形式

① 简单信号赋值语句：`q<=c+d;`

② 条件信号赋值语句：

```
y<=a0 WHEN s="00"ELSE
a1 WHEN s="01" ELSE
a2 WHEN s="10" ELSE
a3;
```

③ 选择信号赋值语句：

```
WITH s SELECT
    --注意：前3行句末是“，”
    y<= a0 WHEN "00",
    al WHEN "01",
    a2 WHEN "10",
    a3 WHEN OTHERS;
    --最后一句结尾为“；”
```

实例

设计一个异步清零、同步置数、带计数使能和进位输出的60进制BCD码计数器。

输入									输出			
CP	$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	D	C	B	A	Q_3	Q_2	Q_1	Q_0
×	0	×	×	×	×	×	×	×	0	0	0	0
↑	1	0	×	×	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
↑	1	1	1	1	×	×	×	×	加法计数			
×	1	1	0	×	×	×	×	×	保持功能			
×	1	1	×	0	×	×	×	×	保持功能			

设计数器的：

- 时钟端为**clk**，
- 异步清零端为**nclr**，
- 同步置数端为**ld**，
- 置入数据端为**din**，
- 计数使能端为**en**，
- 计数器输出端为**qh**和**ql**，
- 进位输出为**co**

代码实例：

```
LIBRARY ieee;                          --库调用说明
USE ieee.std_logic_1164.ALL;           --程序包调用说明
USE ieee.std_logic_unsigned.ALL;

ENTITY cnt60 IS                        --实体开始
PORT
(clk,nclr,ld,en:IN std_logic;          --端口说明
 din: IN std_logic_vector(7 DOWNTO 0);
 qh: BUFFER std_logic_vector(3 DOWNTO 0);
 ql: BUFFER std_logic_vector(3 DOWNTO 0);
 co: OUT std_logic);
END cnt60;                             --实体结束
```

```
ARCHITECTURE behave OF cnt60 IS --结构体开始
BEGIN
 CO<=' 1'  WHEN (qh=" 0101"  AND ql=" 1001"  AND en=' 1' )
 ELSE '0' ;                          --进位输出
 PROCESS(clk, nclr)                  --进程开始
 BEGIN
 --进程主体代码--
 END PROCESS;                        --进程结束
END behave;                          --结构体结束
```

```
IF(nclr=' 0' ) THEN
  --异步清零
  qh<=" 0000" ;
  ql<=" 0000" ;
ELSIF(clk' event AND clk=' 1' ) THEN
  --clk上升沿
  IF(ld=' 1' ) THEN
    --同步置数
    qh<=din(7 DOWNTO 4);
    ql<=din(3 DOWNTO 0);
```

```
ELSIF(en=’ 1’ ) THEN  --计数
  IF(q1=” 1001” ) THEN
    --低位逢十进一
    q1<=” 0000” ;
    IF(qh=” 0101” ) THEN
    --高位逢六进一
     qh<=“0000” ;
    ELSE qh<=qh+1;
    END IF;
  ELSE q1<=q1+1;
  END IF;
END IF;
END IF;
```