

南开大学

# 综合课程设计 实验指导

基于 FPGA 的 Quartus 设计与 VHDL 编程实验

<b>I 实验基础 .....</b>	<b>2</b>
<b>第一章 QUARTUS 使用简介 .....</b>	<b>2</b>
1.1 概述 .....	2
1.2 QUARTUS 运行界面简介 .....	2
1.3 QUARTUS 项目构建 .....	3
<b>第二章 常用集成电路简介 .....</b>	<b>9</b>
2.1 概述 .....	9
2.2 选择器 .....	9
2.3 编/译码器 .....	9
2.4 计数器 .....	9
2.5 寄/锁存器 .....	10
<b>第三章 VHDL 编程简介 .....</b>	<b>11</b>
3.1 概述 .....	11
3.2 基本语法简介 .....	11
3.3 VHDL 编程示例 .....	11
<b>II 项目实例 .....</b>	<b>14</b>
<b>第四章 实验准备及注意事项 .....</b>	<b>14</b>
<b>第五章 示例实验 .....</b>	<b>15</b>
5.1 地址译码及无条件输出实验 .....	15
5.2 计数器实验 .....	17
5.3 时分秒显示实验 .....	19
<b>III 实验安排 .....</b>	<b>21</b>
<b>第六章 基础实验题 .....</b>	<b>21</b>
6.1 第一类实验题 .....	21
6.2 第二类实验题 .....	22
6.3 第三类实验题 .....	24
<b>第七章 自主设计实验提示 .....</b>	<b>26</b>
7.1 电子表 .....	26
7.2 电子琴 .....	26
7.3 红绿灯 .....	26
<b>附录 .....</b>	<b>27</b>
附录一 VHDL 语法速查 .....	27
附录二 常用电路逻辑速查 .....	30

# I 实验基础

## 第一章 Quartus 使用简介

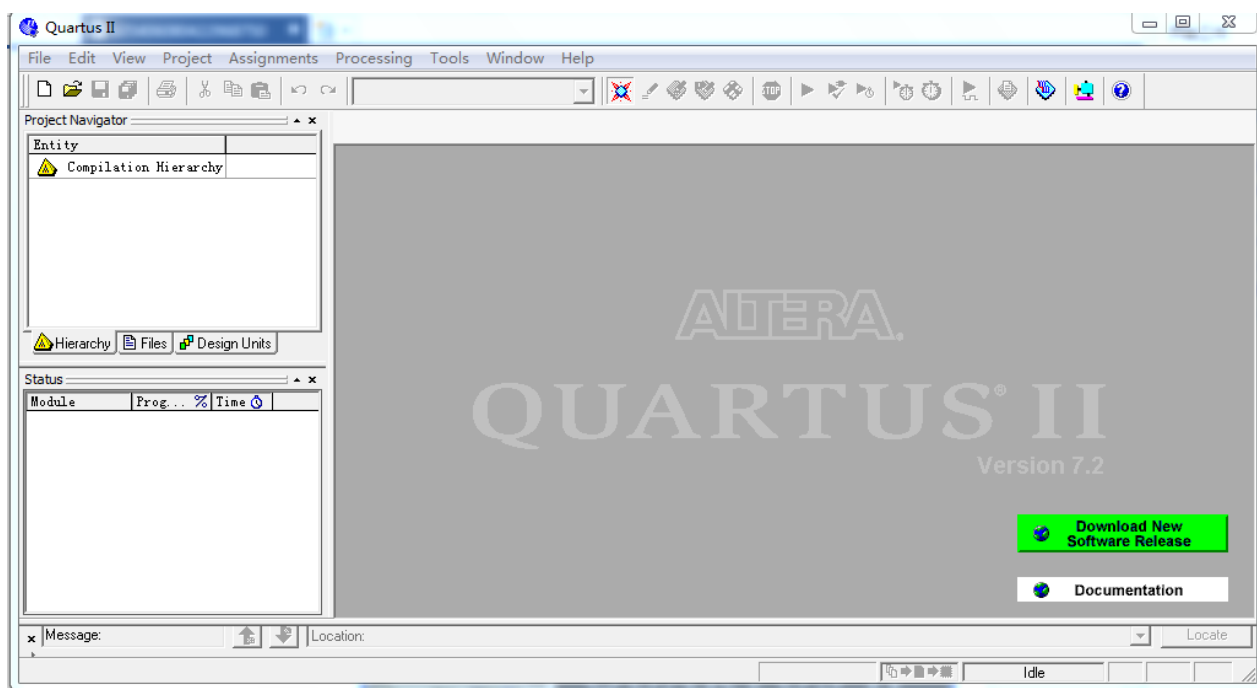
### 1.1 概述

本章主要介绍整个课程都要用到的Quartus软件的基本用法。具体包括: Quartus 运行界面简介、项目构建的步骤、原理图文件的基本编辑方法、模块化设计以及仿真的基本步骤。至于VHDL硬件描述语言将在后面章节介绍。注意, 也许由于Quartus的版本不同可能造成一些差异, 但本质相同无影响, 书中截图为 Quartus II 7.2。

### 1.2 Quartus 运行界面简介

Quartus主界面如图所示:

主要包括标题栏、菜单栏、工具栏、项目导航(Project Navigator)区、状态(Status)显示区、信息(Messages)显示区、编辑窗口区等。项目导航区含有三个标签(或称选项卡): 体系结构(Hierarchy)、文件(Files)和设计构件(Design Units), 可从三个角度来展示项目结构。



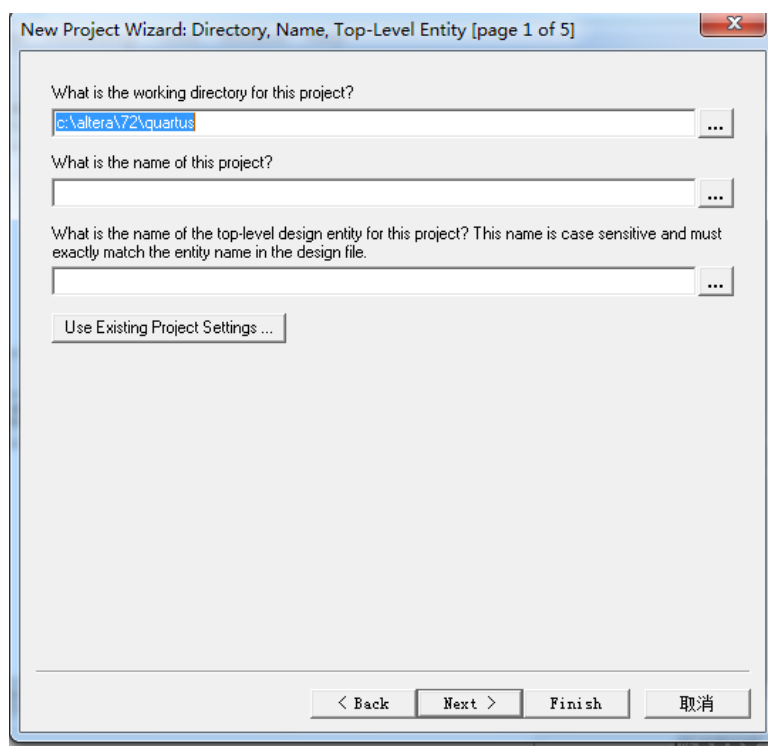
### 1.3 Quartus 项目构建

#### 1.3.1 创建项目

打开 File 菜单, 选择命令 “New Project Wizard”。首先出现 Instruction 对话框, 按 “Next” 按钮, 引出下一个对话框, 如图所示。

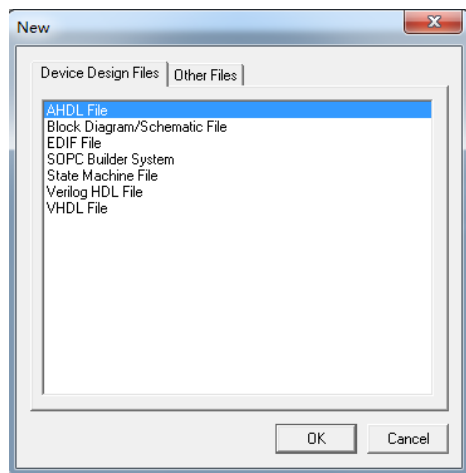
在文本框中分别输入 (或用浏览配合) 项目文件夹名、项目名和顶层入口名 (顶层入口名在后面可调整)。然后直接按 “Finish” 按钮。

需要注意一点, 工程的名字要适当, 如果要用 VHDL 语言写, 则这个名字还要用到。



#### 1.3.2 新建文件 (或打开已有文件)、编辑并存入项目

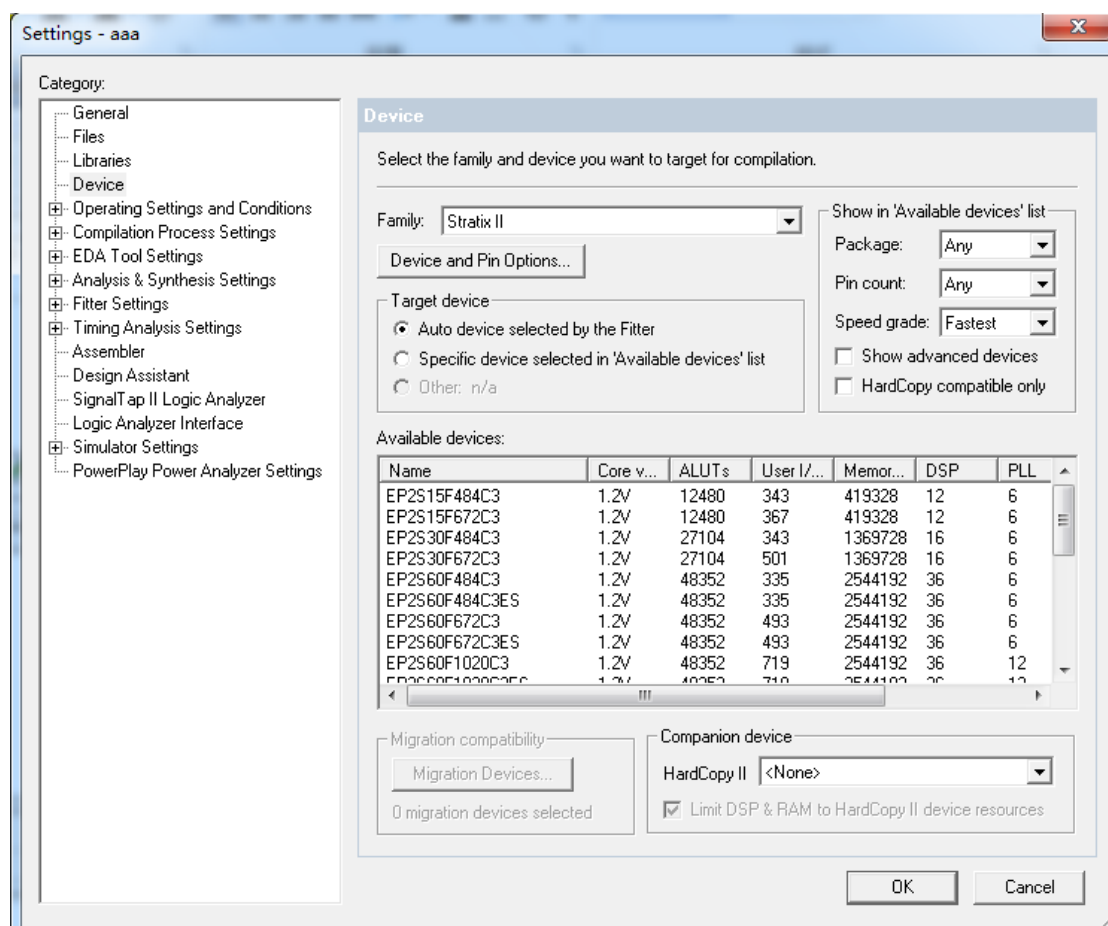
若是新建文件, 打开 File 菜单, 选择 “New” 命令。在出现的 New 对话框 (见图 1.3) 中选择所要设计文件的类型: 如果用语言描述, 则选择 “VHDL File” 或其他语言文件; 如果采用图形编辑, 则选择 “Block Diagram / Schematic File”。点 “OK” 后, 随即出现相应的编辑窗口



若是打开已有文件，则使用 File 菜单中的“Open”命令。通过键盘输入或浏览找到要打开的文件。文件打开后，即可进行编辑。其间可从别处复制一些内容（文本或图形）过来。在编辑语言文本时还可以使用针对某种语言的结构或语句模板。做法是，打开 Edit 菜单，选择“Insert Template”，在出现的对话框中先选择语言，再选择结构或语句。对于原理图文件（Block Diagram /Schematic File，扩展名是“.bdf”，在编辑时应安排好电路的输入输出引脚（pin）。在 Quartus II 5.0 中，引脚和逻辑器件的地位一样，属于符号，在符号库中被安排在文件夹 Primitives 的子文件夹 Pin 下。共有三种引脚：bidir、input 和 output。注意，双向信号（如接口数据总线）应安排 bidir 引脚。关于原理图文件的基本编辑方法，请看下一节。完成编辑后，通过 File 菜单中的“Save”或“Save AS”命令，将文件存到项目文件夹中。

### 1.3.3 选择器件

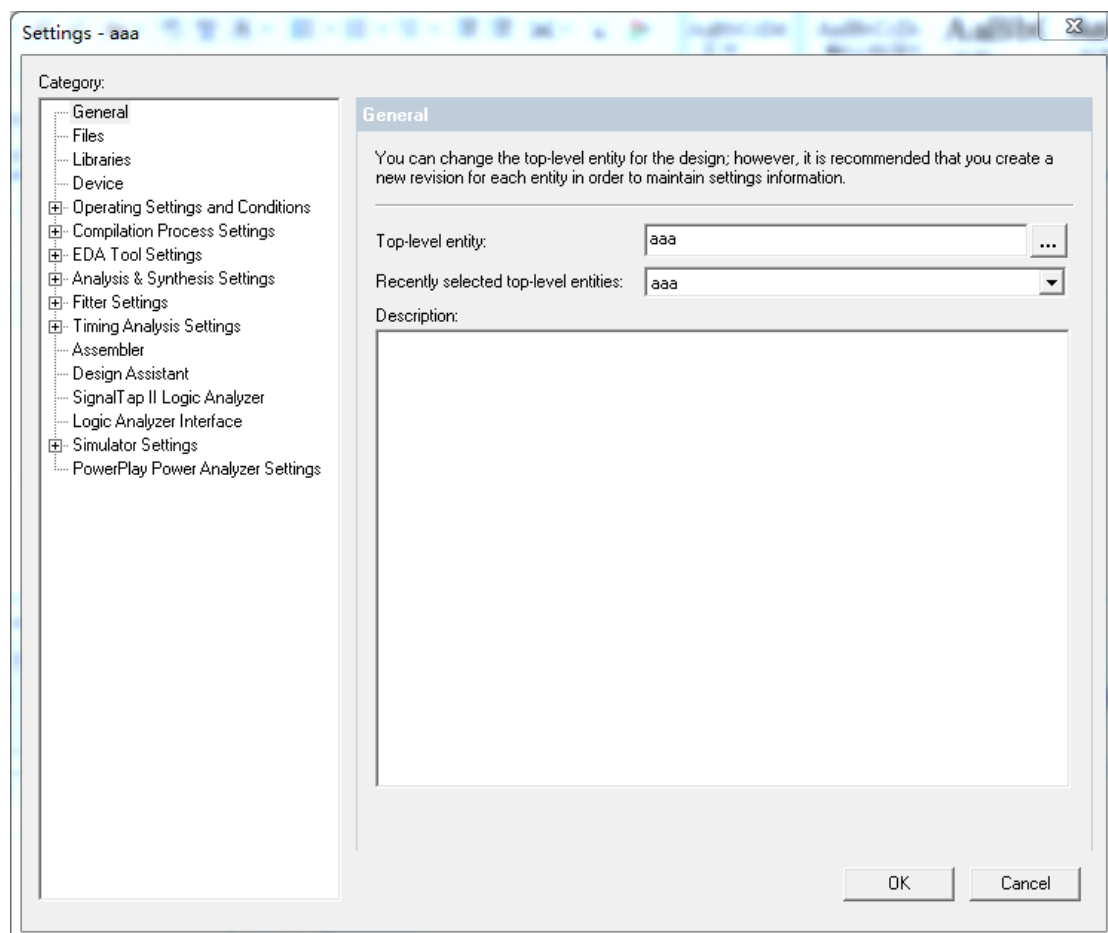
这一步指明项目目标代码要下载到的器件的种类和具体型号（实际是根据器件生成目标代码）。打开 Assignments 菜单，选择命令“Device”，弹出下面的对话框



本次实验中使用的FPGA板型号为Cyclone II 系列芯片的 EP2C5T144C8。

#### 1.3.4 规定项目顶层入口

这一步在建项目时做也可，放到现在做，好处是可根据项目当前实际情况确定入口（Entity）在上述对话框。（Settings）Category 框中选择 General, 或者打开 Assignments 的菜单，选择命令“Settings”，均会出现如图的对话框画面。



在 Top-Level entity 框中输入顶层入口：对于顶层是描述语言文件，可从文件中“entity xxx is”一行复制入口；对于顶层是原理图文件，可取该文件的文件名（不包括扩展名.bdf）。显然，如果仅一层，则该层就是顶层。

### 1.3.5 编译

单击工具按钮，或打开 Processing 菜单，选择命令“Start Compilation”，可启动编译。对于大项目，也可逐个文件编译，这时选择 Processing 菜单中的“Analyze Current File”命令。如果编译发现有错（error），则必须再编辑，排除之。而警告（warning）可以不予理睬，当然，最好也排除掉。

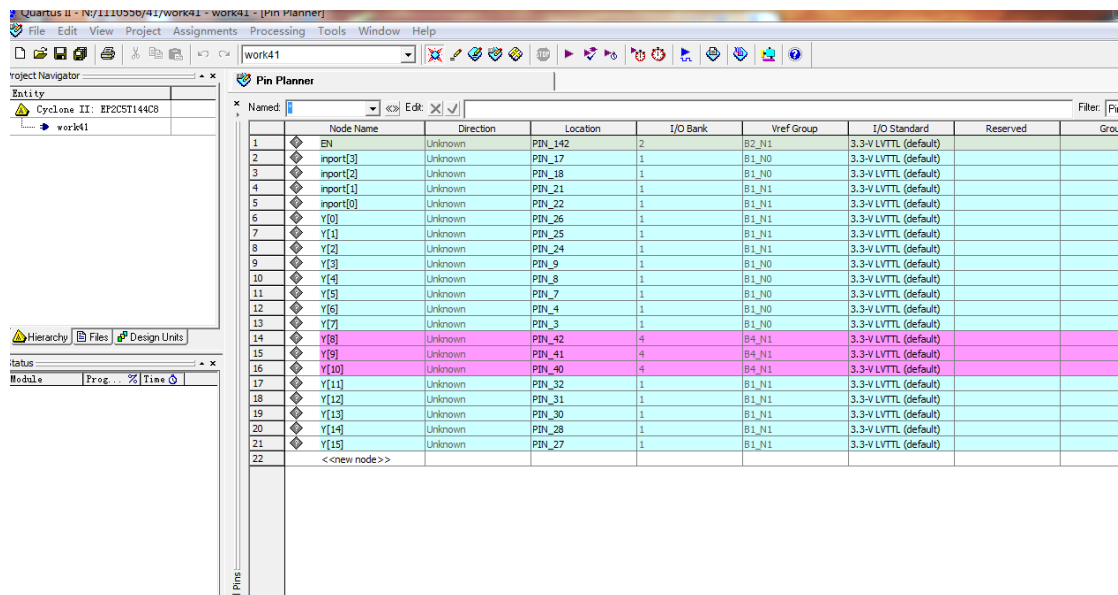
### 1.3.6 分配引脚

这一步使所编逻辑的输入输出与 FPGA 芯片引脚建立对应关系，以便与外部电路相连。注意，这一步仅针对顶层文件。对语言文件，需为每一个被定义的端口（port）分配一个FPGA 引脚；对于原理图文件，需为每一个准备与外部相连的引脚分配一个 FPGA 引脚。

在编译后再进行引脚分配的好处是，可从列表中选择待分配信号，而不必由键盘输入。启动分配引脚的方法是，打开 Processing 菜单，选择命令“Pins”将弹出的窗口。（Assignment Editor）拉长或最大化，如图所示。双击 To 下面的“<<new>>”，会出现 port 信号列表或原理图输入输出引脚信号列表，从中选择一个信号。再双击 Location 下面的“<<new>>”，会出现 FPGA 芯片引脚列表，根据自己的安排做选择。该步重复进行，直至所有的 port 信号或原理图引脚信号均分配了引脚。

如果新项目与原先某个项目的引脚分配相似，可再启动一个 Quartus 例程，通过表格选取和复制的方法，复制引脚分配。

提醒实验者注意，分配完引脚后最好再仔细检查一遍，以免出错。



### 1.3.7 再次编译一遍

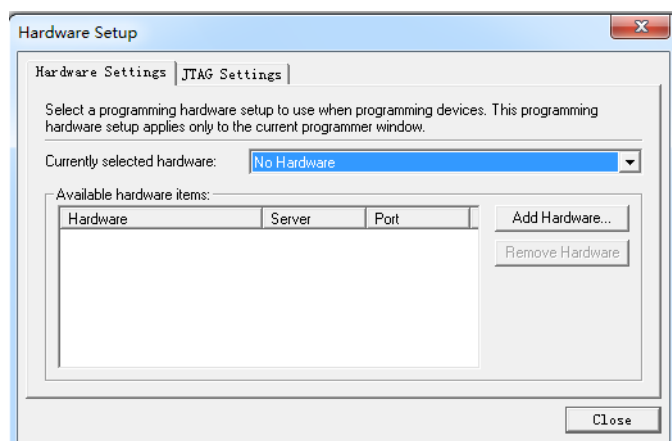
将整个项目再编译一遍，这是工程会把分配的引脚加入进去，若无错，则项目构建成功，相应文件（包括下载文件）均成功建立。

### 1.3.8 下载运行

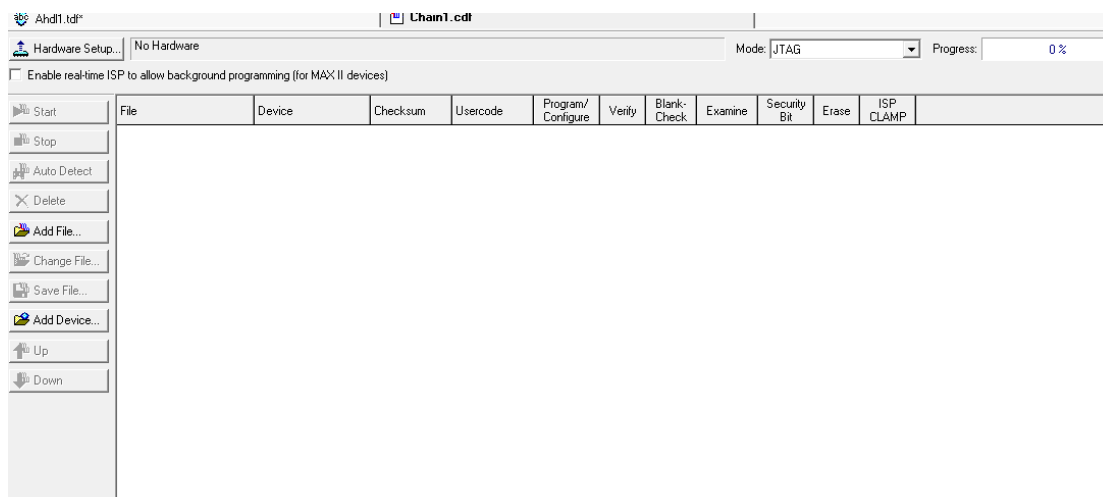
需要把FPGA板先通过USB连入电脑。

在 Quartus 窗口中，第一次下载前需要进行编程硬件设置。打开菜单 Tools，选择“Programmer”或点击工具按钮，会出现下载窗口。单击左上方的“Hardware Setup”按钮，出现如图 所示的对话框





单击 Start 按钮，便开始下载，Progress 条显示出下载进度，Quartus II 窗口下部的信息窗口会显示相关信息。



## 第二章 常用集成电路简介

### 2.1 概述

本章主要介绍几种将在接下来的实验中涉及的几类以 74 系列为代表的常用门电路，具体包括选择器、编码器、译码器、计数器、寄存器、锁存器、触发器，需要了解掌握其逻辑功能及工作方式。对于其中的某些元件，我们仅需掌握其使用方式并在实验中作为一个模块整体使用，而对于一些较为简单的元件，我们要力求使用图形设计方法与 VHDL 编程去实现其功能，并以此深入理解自底向上的模块化设计方法。

### 2.2 选择器

选择器(data selector)是根据给定的输入地址代码，从一组输入信号中选出指定的一个送至输出端的组合逻辑电路。

在本实验中常涉及的为 74151，八选一数据选择器，通过 A、B、C 三位数选择让 Y 输出  $D_0$ - $D_7$  中某个输入的值。（具体可参见附录二）

### 2.3 编/译码器

编码器(encoder)、译码器(decoder)是将信号或数据进行编制、转换为可用以通讯、传输和存储的信号形式的设备。

在本实验中常涉及的编码器是 74148，8-3 优先编码器，将 8 线的输入编码成 3 线的二进制数输出，但是输入的信号存在优先级，即输入信号从  $I_7$  到  $I_0$  的顺序依次优先级降低，当高优先级的信号有效时，低优先级的信号则不起作用。（具体可参见附录二）

在本实验中常涉及的译码器是 74138，3 线 8 路译码器，将三线的输入译码成 8 线中的某个选通信号，常用作地址译码，比如使用 16 位地址线的低 3 位作为输入，将某个地址对应某个模块的使能信号，即可通过该地址使用对应模块。（具体参见附录二）

### 2.4 计数器

计数器(counter)是一个用以实现计数功能的时序器件，它不仅可以用来记忆脉冲的个数，还常用于数字系统的定时、分频和执行数字运算以及其它特定的逻辑功能。

常用的计数器如 74192，同步十进制可逆计数器，具有双时钟输入十进制可逆计数功能；异步并行置数功能；保持功能和异步清零功能。

## 2.5 寄/锁存器

寄存器(register)中用的记忆部件是触发器, 每个触发器只能存一位二进制码。

移位寄存器具有数码寄存和移位两个功能, 在移位脉冲的作用下, 数码如向左移一位, 则称为左移, 反之称为右移。移位寄存器具有单向移位功能的称为单向移位寄存器, 即可向左移也可向右移的称为双向移位寄存器。

在本实验中常涉及的为 **74194**, 4 位双向通用移位寄存器。

锁存器(latch)是一种对脉冲电平敏感的存储单元电路, 它们可以在特定输入脉冲电平作用下改变状态。锁存, 就是把信号暂存以维持某种电平状态。锁存器的最主要作用是缓存, 其次完成高速的控制器与慢速的外设的不同步问题, 再其次是解决驱动的问题, 最后是解决一个 I/O 口既能输出也能输入的问题。锁存器是利用电平控制数据的输入, 它包括不带使能控制的锁存器和带使能控制的锁存器。

在本实验中常涉及的为 **74244**, 八路正相缓冲器/线路驱动器, 具有三态输出。

### 第三章 VHDL 编程简介

#### 3.1 概述

VHDL 即 VHSIC Hardwarter Description Language

其中 VHSIC 即 Very High speed integrated circuit

VHDL 是美国国防部在 20 世纪 80 年代初为实现其高速集成电路硬件 VHSIC 计划提出的描述语言；IEEE 从 1986 年开始致力于 VHDL 标准化工作，融合了其它 ASIC 芯片制造商开发的硬件描述语言的优点，于 93 年形成了标准版本（IEEE. std\_1164）；1995 年，我国国家技术监督局推荐 VHDL 做为电子设计自动化硬件描述语言的国家标准。

#### 3.2 基本语法简介

**library ieee;**

--库声明，声明工程中用到的库，这里声明的是 IEEE 库

**use ieee.std\_logic\_1164.all;**

--包声明，声明工程中用到的包，这里声明的是 IEEE 的 STD\_LOGIC\_1164 包

entity

单体：它负责宣告一个硬件的外部输入与输出，一个简单的范例（尖括号内为必填，方括号内为可选）：

**entity <实体名称> is**

**port(**

**a : IN STD\_LOGIC;**

**b : OUT STD\_LOGIC**

**);**

**end [实体名称];**

architecture

架构：它负责实现内部的硬件电路。

**architecture <结构体名称> of <实体名称> is**

**begin**

--此处可编写结构体内部操作

**end [结构体名称];**

configuration

配置：用来描述各种层与层的连接关系以及实体与结构体之间的关系，

**configuration [配置名] of [实体名] is**

--配置说明

**end [配置名];**

#### 3.3 VHDL 编程示例

##### 3.3.1 编写触发器示例

```

library ieee;
use ieee.std_logic_1164.all;
entity test is
  port(
    d      : in   std_logic;
    clk    : in   std_logic;
    q      : out  std_logic);
end test;
architecture trigger of test is
  signal q_temp:std_logic;
begin
  q<=q_temp;
  process(clk)
  begin
    if clk'event and clk='1' then
      q_temp<=d;
    end if;
  end process;
end trigger;
configuration d_trigger of test is
--配置, 将结构体配置给实体, 配置名为 d_trigger
  for trigger
  end for;
end d_trigger;

```

### 3.3.2 编写偶数分频器示例

```

Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;

Entity fdiv is
  generic(N: integer:=6);      --rate=N, N 是偶数
  port(
    clk_in: IN std_logic;
    clk_out: OUT std_logic
  );
End fdiv;
Architecture a of fdiv is
  signal cnt: integer range 0 to n-1;
Begin
  process(clk_in) --计数
  begin
    if(clk_in'event and clk_in='1') then
      if(cnt<n-1) then
        cnt <= cnt+1;
      else
        cnt <= 0;
      end if;
    end if;
  end process;
end process;

```

```
process(cnt) --根据计数值, 控制输出时钟脉冲的高、低电平
begin
    if(cnt<n/2) then
        clkout <= '1';
    else
        clkout <= '0';
    end if;
end process;

End a;
```

## II 项目实例

### 第四章 实验准备及注意事项

1. 实验前, 需将 FPGA 板的地 (GND) 和实验箱的地 (GND) 连通, 将FPGA板的+5V和实验箱的+5V连通, (注意, 对实验箱, 切不可错接到 GND 旁边的 VCC插孔, 否则会损坏 FPGA 芯片以及用户实验板上的电路)。

2. 连线必须在关闭用户实验板和 FPGA 板电源的情况下进行。下载电缆插头也不得带电插拔, 否则会损坏 FPGA 芯片。

3. 为了延长连接线的使用寿命, 在插拔连接线时, 不要直接拽线, 而是抓住插头的顶部操作。

4. 实验所用的 74 系列集成电路逻辑可在符号库文件夹 “others” 的子文件夹 “maxplus2” 下找到。

5. 在 Quartus II 中, 节点名 (node name) 中不允许含符号 “#”, 所以, 对于 IO 读/写的节点 (一段连线) 或引脚命名时直接用 IOR 或 IOW 即可。

6. 本系统 FPGA 板所用 FPGA 芯片是 Altera 公司的 **Cyclone II 系列芯片的 EP2C5T144C8**。

7. 在为顶层文件分配引脚时, 应选用 FPGA 板上标注编号的引脚。FPGA板中部分引脚已有特殊用途, 使用请注意避开使用这些引脚。(参见FPGA的EP2C5管脚手册)

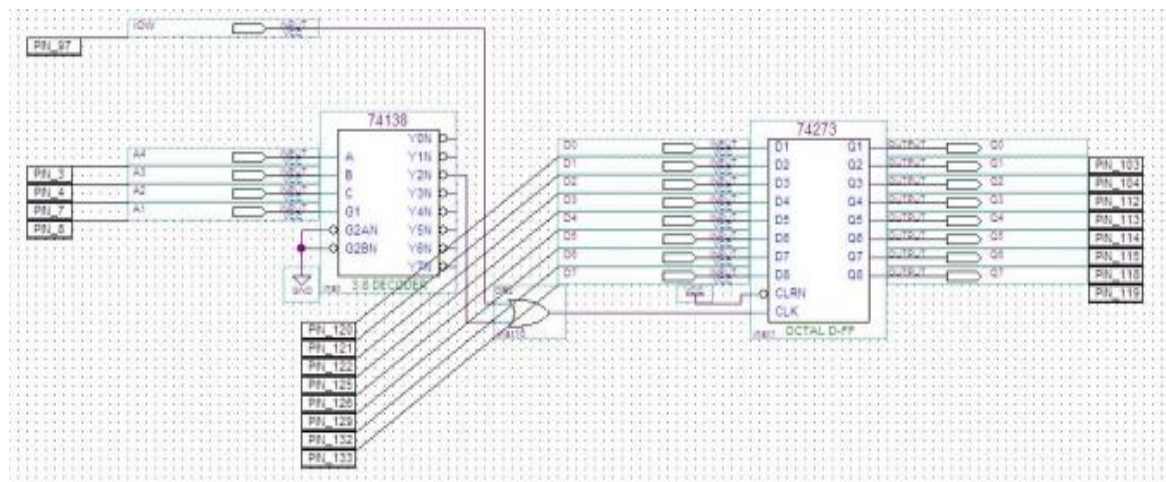
8. 对未使用的引脚, 应设置为 “As input tri-stated”, 以避免 FPGA 板或用户实验板上的电路损坏。设置的方法是, 打开 Assignments 菜单, 选择 “Device”, 在出现的对话框中单击 “Device & pin option”按钮, 在新出现的对话框中点击 “Unused Pins ” 选项卡 (标签), 选择 “As input tri-stated”。



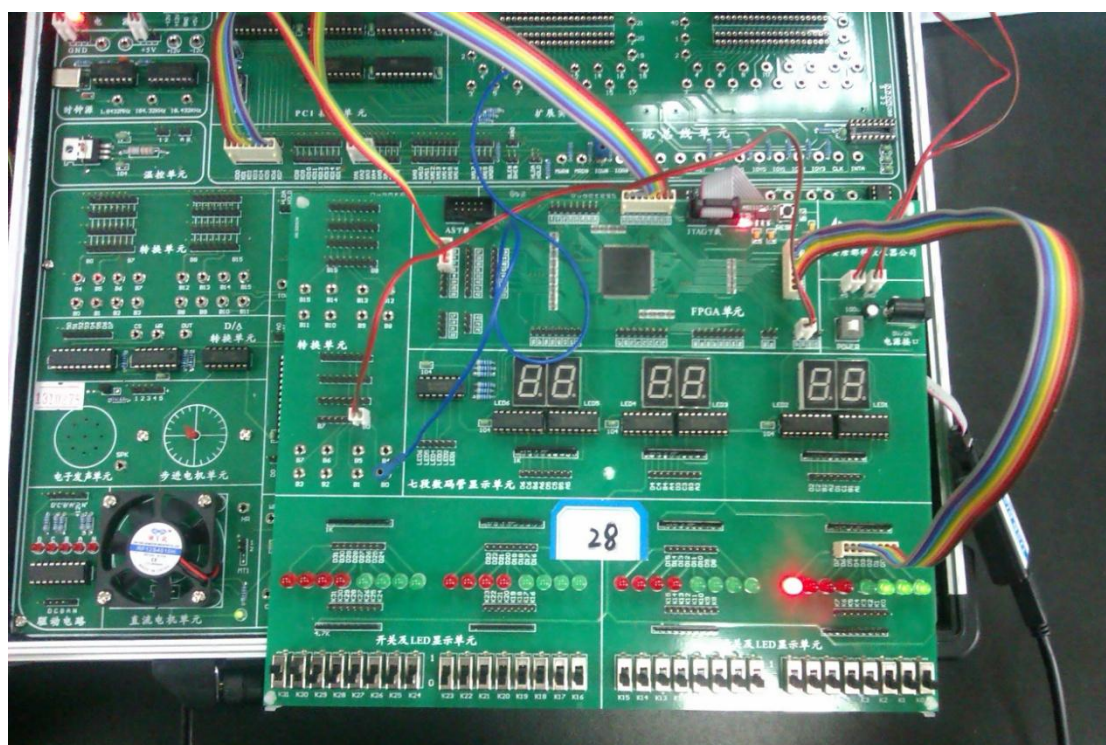
## 第五章 示例实验

### 5.1 地址译码及无条件输出实验

该实验采用图形设计方法，通过 74138 进行地址译码，通过 74273 进行锁存，实现将由总线指定地址的输出锁存显示的功能。



连接说明：





信号定义	IOW (写信号)	A1~A4 (地址信号)	D0~D7 (输入数据)	Q0~Q7 (输出显示)
FPGA 引脚	PIN_97	PIN_3、4、7、8	PIN_120、121、122、125、126、129、132、133	PIN_103、104、112、113、114、115、116、119
实验板连接	实验箱总线单元 IOW 口 (通过转换单元将引脚转换)	实验箱总线单元 XA1~XA4	实验箱总线单元 XD0~XD7	FPGA 板或实验箱的 LED 灯显示

下载说明:

1. 将 FPGA 板的电源 (+5V 与 GND) 连接至实验箱的相应口, 将 JTAG 线连接 FPGA 板与电脑, 打开实验箱与 FPGA 板的电源。
2. 打开项目文件下的 quartus 工程文件, 单击 tools 中的 programmer, 确认 FPGA 板硬件连接后点击 start 进行下载 (此工程已经预先 pin 完引脚并编译成了 sof 文件, 下载该 sof 文件即可, 故不需进行其他操作)。

运行说明:

进入电脑中预装的 TD-PIT 程序, 打开项目文件下的 c 文件, 如图所示, 进行编译、链接、运行, 可观察到 LED 灯显示相应数值 (示例中给出的是 0x87)。



```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <ctype.h>
#include <process.h>

void main()
{
    outp(0x300A, 0x87);
}

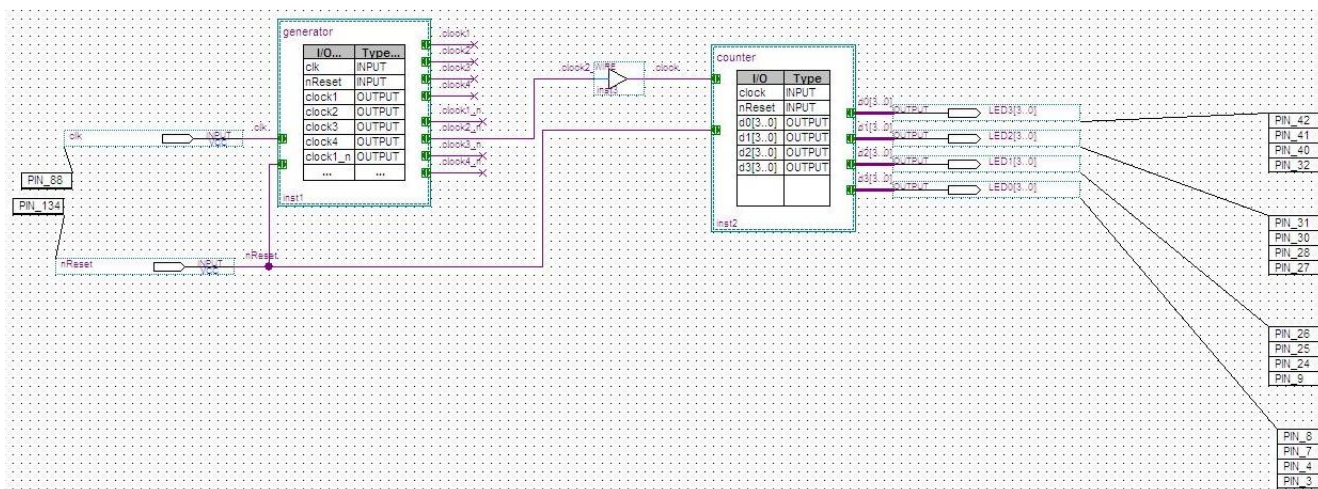
```

该实验的地址译码所使用的地址是 0x300A, 实际只进行了部分译码, 即 A1~A4 四位地址, 即下图所示标红四位, 作为 A、B、C、G1 的输入信号, 查阅 74138 真值表 (参见附录二) 可知当输入的 A、B、C、G1 位 0、1、0、1 时, 选通 Y2N (输出低电平), 即上述电路图中 74138 输出与写信号 (低电平) 或后作为 74273 的时钟信号 (上升沿有效)。

0x300A —————> (二进制) 0011 0000 0000 **1010**

## 5.2 计数器实验

该实验采用图形设计方法与VHDL编程相结合,实现了一个四位的十进制的计数器。



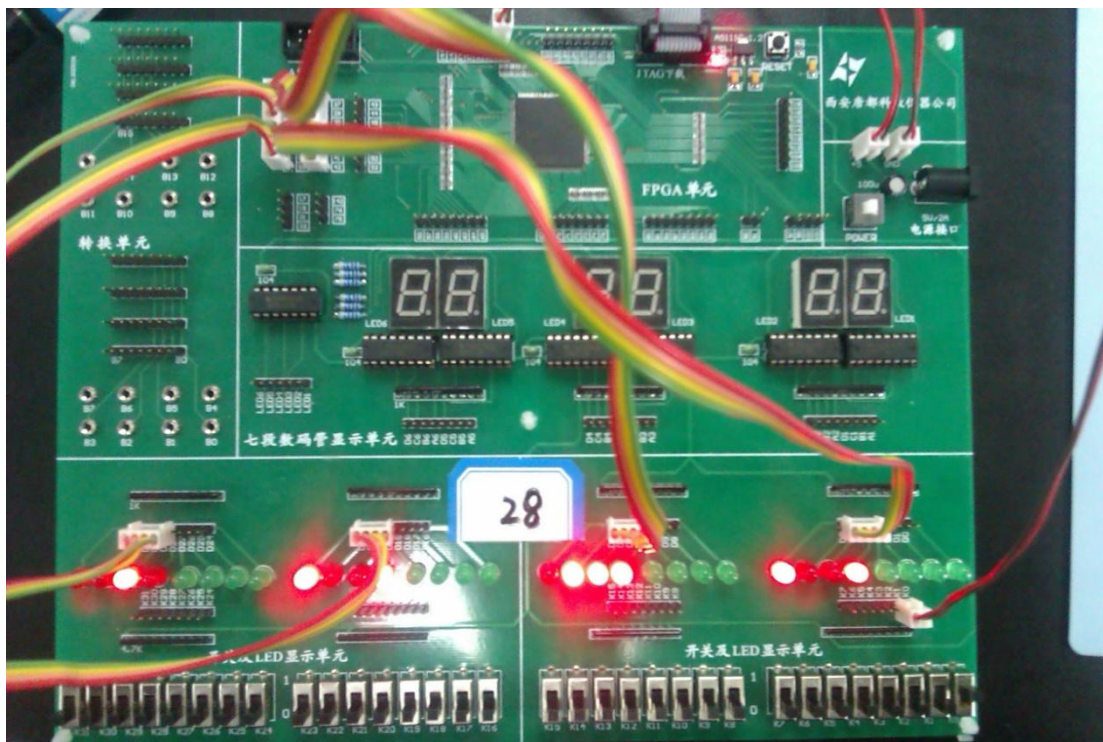
如上图所示, 点开 **generator** (分频器, 用来将时钟信号的频率变换) 与 **counter** (计数器) 可以看到内部的 VHDL 语言描述的电路逻辑。

Named	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	Reserved	Group
1	clk	Input	PIN_88	3	B3_N1	3.3-V LVTTTL (default)		
2	LED0[3]	Output	PIN_88	IOBANK_3	Dedicated Clock	CLK7, LVDSCLK0n, Input		LED0[3..0]
3	LED0[2]	Output	PIN_89	IOBANK_3	Dedicated Clock	CLK6, LVDSCLK3p, Input		LED0[3..0]
4	LED0[1]	Output	PIN_90	IOBANK_3	Dedicated Clock	CLK5, LVDSCLK2n, Input		LED0[3..0]
5	LED0[0]	Output	PIN_91	IOBANK_3	Dedicated Clock	CLK4, LVDSCLK2p, Input		LED0[3..0]
6	LED1[3]	Output	PIN_92	IOBANK_3	Row I/O	LVDS35n		LED1[3..0]
7	LED1[2]	Output	PIN_93	IOBANK_3	Row I/O	LVDS35p, DPCLK7/DQS0R/CQ1R		LED1[3..0]
8	LED1[1]	Output	PIN_94	IOBANK_3	Row I/O	LVDS34n		LED1[3..0]
9	LED1[0]	Output	PIN_96	IOBANK_3	Row I/O	LVDS34p		LED1[3..0]
10	LED2[3]	Output	PIN_31	1	B1_N1	3.3-V LVTTTL (default)		LED2[3..0]
11	LED2[2]	Output	PIN_30	1	B1_N1	3.3-V LVTTTL (default)		LED2[3..0]
12	LED2[1]	Output	PIN_28	1	B1_N1	3.3-V LVTTTL (default)		LED2[3..0]
13	LED2[0]	Output	PIN_27	1	B1_N1	3.3-V LVTTTL (default)		LED2[3..0]
14	LED3[3]	Output	PIN_42	4	B4_N1	3.3-V LVTTTL (default)		LED3[3..0]
15	LED3[2]	Output	PIN_41	4	B4_N1	3.3-V LVTTTL (default)		LED3[3..0]
16	LED3[1]	Output	PIN_40	4	B4_N1	3.3-V LVTTTL (default)		LED3[3..0]
17	LED3[0]	Output	PIN_32	1	B1_N1	3.3-V LVTTTL (default)		LED3[3..0]
18	nReset	Input	PIN_134	2	B2_N1	3.3-V LVTTTL (default)		
19	<<new node>>							

如上图为该实验工程的引脚连接, 其中 **clk** 信号使用的 **PIN\_88** 引脚为 FPGA 板内置的时钟信号, 故本实验不需接额外的时钟信号。

实验的另一个输入是 **PIN\_134**, 为该计时器的重置/使能信号, 我们将它接 FPGA 板或实验箱上的一个逻辑开关即可。

下图是该实验连接图: (为便于观察, 将四组四位信号输出全部接了 FPGA 板上的红色 LED 灯组, 连接时请特别注意每一组中四位的高低位与四组的高低位顺序)

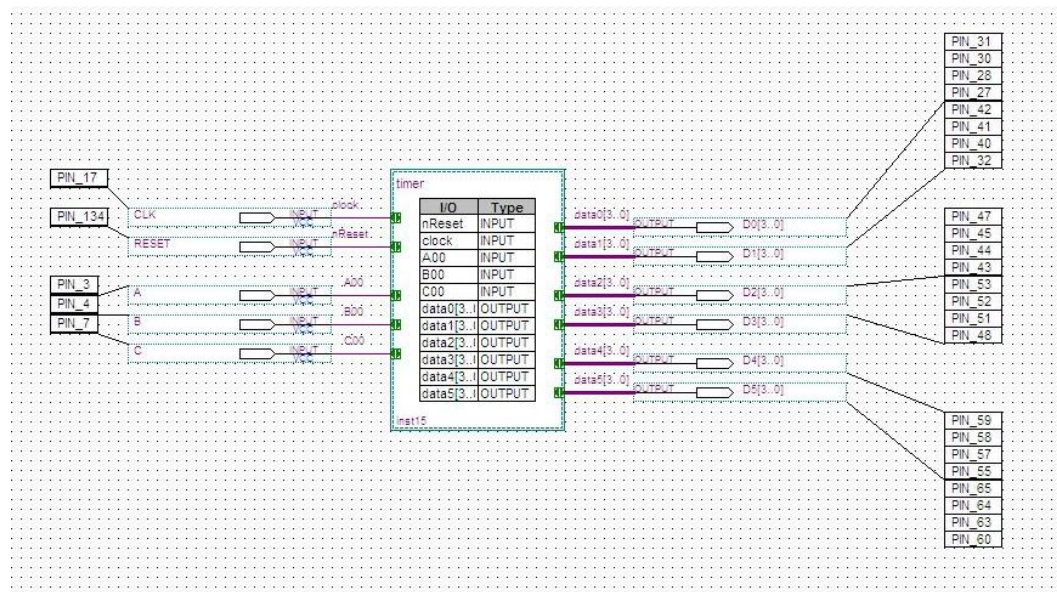


仿照实验 5.1 中完成连接与下载之后，将接 PIN\_134 的逻辑开关推至 1，即可观察到计数器开始计数，每一组四位从 0000 自增至 1001 后进位。



### 5.3 时分秒显示实验

该实验使用采用图形设计方法与 VHDL 编程结合，实现了一个显示时分秒的时钟。

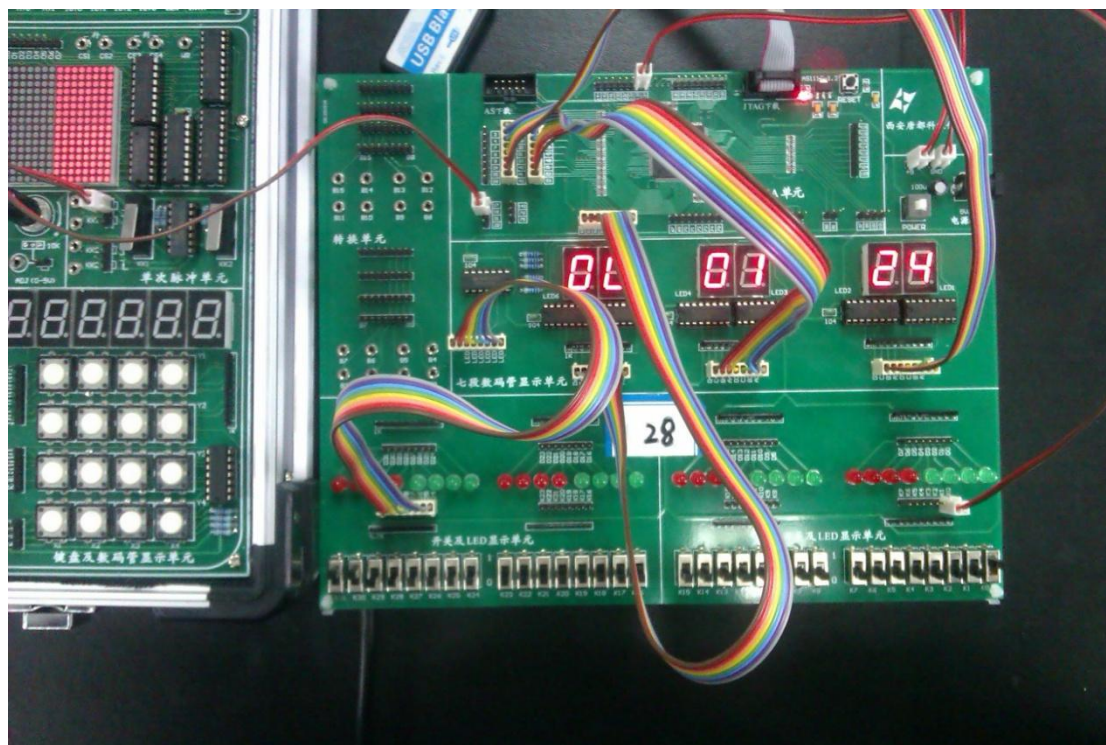


如上图中 timer 单元采用 VHDL 编写。

CLK 信号为输入的时钟信号，可考虑使用实验板上的时钟信号作为输入（在本示例中使用了单次脉冲信号作为输入），RESET 信号仍仿照实验 5.2 使用逻辑开关置位。

（A、B、C 输入为预留的控制位，在本次实验中暂未用到，无需连接）

实验连接图如下所示：



图中将 PIN\_17 引脚接了实验箱的单次脉冲单元 KK1，即每单击一下单次脉冲单元的拨片，输出一个时钟信号，时分秒计数增加 1 秒（思考：如何仿照实验 5.2 中的分频器实现频率为 1HZ 的时钟信号作为该时分秒的时钟信号输入？）

另外需要注意的是 FPGA 板七段数码管显示模块的接线，图中的 LED1~LED6 为 LED 显示的使能信号（低电平有效），我们将它们接逻辑开关，并全部置位 0。电路图右侧三组输出从下到上依次为两位的时分秒显示信号，分别接数码管模块的六组输入即可（注意顺序）。

完成连接和下载之后将 RESET 信号置 1，拨动单次脉冲单元的拨片，即可观察到时分秒时钟显示逐秒递增。

## III 实验安排

### 第六章 基础实验题

#### 6.1 第一类实验题

本类实验是用图形设计方法在 FPGA 芯片内构建基于计算机接口的实验逻辑，需要编程驱动。

##### 6.1.1 在 FPGA 芯片内用图形设计方法实现一个无条件输入输出端口（包括端口地址译码电路）。

将该端口的输出接用户实验板的 LED 显示模块，以便显示送到该端口的数据。说明：地址译码器请选用 74138，端口所用的寄存器请选用 74273，该端口的数据来自用户实验板的数据总线。（该实验已经在示例实验5.1中给出，要求更改译码的地址重新完成）。

说明：

计算机寻址为所有实验的核心，正确理解并掌握计算机寻址对于整个学习至关重要。

端口或元器件都有一个接口叫“使能端”，意思是这个接口有效，本元器件才能工作，可以理解为开关。

寻址的过程是，当 20 位地址总线上的信号恰好为该元器件预设的地址时，元器件的使能端为有效。一般通过一系列与非门，最终连到使能端。

具体结合实验器材介绍寻址方法，方便使用。

首先点开 TD-PIT 的“端口资源”，发现实验器材提供的地址空间：

IOY0: 3000H ---303FH

IOY1: 3040H ---307FH

IOY2: 3080H ---30BFH

IOY3: 30C0H ---30FFH

对应实验板上的接口 IOY0—IOY3（注意：这四个端口为“低电平”有效），我们也可以绕过 IOY 口对实验板上的 XA0-XA20 的地址线进行直接使用，本次将直接使用地址线进行地址译码。（但是所能使用的地址范围仍然只有 3000H 到 30FFH，这是程序中的限制，不过本次进行地址的部分译码，只会使用到其中的几位，参见示例实验 5.1）

##### 6.1.2 在 FPGA 芯片内用图形设计方法实现一个无条件输入端口（包括端口地址译码电路）。

端口所用的三态门可用 74244，该端口的数据输入来自用户实验板的手动数字量输入模块，读入的结果在机器屏幕上显示出来。







### 6.1.3 在 FPGA 芯片内用图形设计方法实现一个 8 位循环右移寄存器（包括端口地址译码电路，移位寄存器可选用 74194）。

该移位寄存器的初值来自用户实验板的数据总线，由 VC 测试程序设定，每按一次回车键，右移一位。将该移位寄存器的输出接用户实验板的 LED 显示模块，以便观察运行情况。

说明：

74194 是 4 位双向通用移位寄存器，其输入输出关系为。

**TRUTH TABLE**

CLEAR	INPUTS									OUTPUTS			
	MODE		CLOCK	SERIAL		PARALLEL				QA	QB	QC	QD
	S1	S0		LEFT	RIGHT	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X		X	X	X	X	X	X	QA0	QB0	QC0	QD0
H	H	H		X	X	a	b	c	d	a	b	c	d
H	L	H		X	H	X	X	X	X	H	QAn	QBn	QCn
H	L	H		X	L	X	X	X	X	L	QAn	QBn	QCn
H	H	L		H	X	X	X	X	X	QBn	QCn	QDn	H
H	H	L		L	X	X	X	X	X	QBn	QCn	QDn	L
H	L	L	X	X	X	X	X	X	X	QA0	QB0	QC0	QD0

X: Don't Care : Don't Care

a ~ d : The level of steady state input voltage at input A ~ D respectively

QA0 ~ QD0 : No change

QAn ~ QDn : The level of QA, QB, QC, respectively, before the most recent positive transition of the clock.

由表可知，在 S1、S0 为 H、H 的情况下，CLK 上升沿来时，74194 进行并行输入。由表还可看出，假定 QD 为最高位（通常如此），则在 S1、S0 为 H、L 的情况下，CLK 上升沿来时，寄存器右移一位，最高位的状态取决于 SLSI（左端串行输入）的电平；而在 S1、S0 为 L、H 的情况下，CLK 上升沿来时寄存器左移一位，最低位的状态取决于 SRSI（右端串行输入）的电平。通常清除端 CLRN 接高电平，在进行电路设计时直接连 VCC 即可。

## 6.2 第二类实验题

本类实验是用 VHDL 编程方法在 FPGA 芯片内构建独立逻辑实验（与计算机接口无关）。



### 6.2.1 在 FPGA 芯片内用硬件描述语言描述一个 3-8 译码器。

利用用户实验板上的资源测试该逻辑：将使能端和代码输入端接用户实验板上的手动开关，译码输出接 LED 显示模块。进行手动操作，观察该译码器的输入输出特性。

### 6.2.2 在 FPGA 芯片内用硬件描述语言描述一个 8-3 优先编码器

利用用户实验板上的资源测试该逻辑：输入代码端接用户实验板上的手动开关，输出接 LED 显示模块。进行手动操作，观察该编码器的输入输出特性。

说明：“编码”是“译码”的逆行为，实现编码的逻辑称为“编码器”。编码器有多个输入，多个输出。普通编码器在同一时刻最多只能有一个输入信号有效，当所有输入均无效时，规定产生某种输出。而优先编码器允许几个输入同时有效，但各个输入的优先权不同，编码器自动产生当前优先权最高的输入的编码。本题要实现的编码器有 8 个（即 8 位）输入，3 个输出。

注意：

该编码器没有使能端。

该逻辑在测试时，对表中最下面两行，均会看到所接的 3 个 LED 全亮。然而，编码器实际输出的电位是不相同的，可用万用表测量：“1”信号对应 3.2 V 左右，“2”对应 1.2 V 左右（在编码器输出与 LED 模块连线的情况下）。

### 6.2.3 在 FPGA 芯片内用硬件描述语言描述一个具有清除端（CLRN）的 8 位通用寄存器

该寄存器在 CLRN 为低电平时清 0（输出变全 0），在 CLRN 为高电平的情况下，接受脉冲输入端（CLK）上升沿来时进行数据写入，并立即出现在输出端。

利用用户实验板上的资源测试该逻辑（清 0 和数据写入）。

### 6.2.4 在 FPGA 芯片内用硬件描述语言构建一个信号发生器，分别产生周期为 2 秒、1 秒、0.5 秒和 0.25 秒的方波

使用实验箱或 FPGA 板上的任意时钟输出作为输入，将这些输出信号接 LED 显示模块，以便观察其周期。



说明：可参考之前介绍的分频器，即从一个时钟频率进行计数，按特定周期/频率输出方波即可。（参考示例实验 5.2 中的分频器部分与 VHDL 编程示例 3.3.2 分频器示例）

### 6.3 第三类实验题

本类实验是用图形设计方法与VHDL编程相结合在 FPGA 芯片内构建独立逻辑实验（与计算机接口无关）。

#### 6.3.1 设计一个 4 线-16 线译码器。

其输入输出特性如下表所示。

使能	输入代码				输 出															
EN	D	C	B	A	Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

注：X 表示取值任意

用FPGA板或者实验箱中的数字开关为其提供输入代码和使能控制信号，译码结果在FPGA或实验箱中的LED灯上显示出来。

#### 6.3.2 设计一个可循环左/右移，并可进行并行输入的 8 位移位寄存器。

用单脉冲作为并行输入或移位的控制脉冲。

提示：可考虑使用 4 位移位寄存器 74194。（参见附录二）

### 6.3.3 设计 8086 逻辑地址到物理地址的转换电路。

逻辑地址的段地址和段内偏移量均由逻辑开关提供，转换成的 20 位物理地址不要求寄存，直接送LED显示。

提示：可考虑使用 4 位加法器 7483。（参见附录二）

### 6.3.4 设计一个能显示时、分、秒的实时时钟

通过FPGA板上的数码管显示。初始时间为0 时 0 分 0 秒。  
（参考示例实验5.3）

对这部分实验的实验报告要求是，对每一实验题：

- 打印原理图或.vhd 文件，若含二者，则均打印；
- 说明引脚分配以及与用户实验板之间的连接；
- 提交项目和测试程序

## 第七章 自主设计实验提示

该章主要为使用实验箱与FPGA板进行自主设计实验提供一个方向性提示，不提供具体的实验过程，鼓励灵活运用实验仪器的各个部分，将之前学习的每个模块设计的内容融合创新，设计出具有特定功能的电路模块。

### 7.1 电子表

以示例实验5.3为基础，编写其中A、B、C的控制位逻辑，可以实现一个可以进行时间调节的电子表。

### 7.2 电子琴

以示例实验 5.2 中的分频器为基础可以实现输出特定频率方波的电路，利用实验箱的电子发声单元，即可将这些特定频率的方波输出为特定音频的声音，以 4\*4 的小键盘作为电子琴键，即可实现一个简易电子琴。

### 7.3 红绿灯

以示例实验 5.2 中的计数器为基础，即可实现一个按周期变化的红绿灯，接 LED 显示，并可同时仿照示例实验 5.3 为这个红绿灯提供按秒的数字显示的倒计时。

## 附录

### 附录一 VHDL 语法速查

#### 1.数据对象

##### 1.1 常量 Constant

常量是对某一常量名赋予一个固定的值，而且只能赋值一次。通常赋值在程序开始前进行，该值的数据类型则在说明语句中指明。

Constant 常数名: 数据类型: =表达式

Constant Vcc: real:=5.0; --定义 Vcc 的数据类型是实数，赋值为 5.0V

Constant bus\_width: integer := 8; --定义总线宽度为常数 8

##### 1.2 变量 Variable

变量只能在进程语句、函数语句和过程语句结构中使用。变量的赋值是直接的，非预设的，分配给变量的值立即成为当前值，变量不能表达“连线”或存储元件，不能设置传输延迟量。

变量定义语句:

**Variable 变量名 : 数据类型 := 初始值;**

**Variable count: integer 0 to 255:=20 ; -- 定义 count 整数变量，变化范围 0~255，初始值为 20。**

变量赋值语句:

**目标变量名 := 表达式;**

**x:=10.0; --实数变量赋值为 10.0**

**Y:=1.5+x; --运算表达式赋值，注意表达式必须与目标变量的数据类型相同**

**A(3 to 6):=("1101"); --位矢量赋值**

##### 1.3 信号 Signal

信号表示逻辑门的输入或输出，类似于连接线，也可以表达存储元件的状态。信号通常在构造体、程序包和实体中说明。

信号定义语句:

**Signal 信号名: 数据类型 := 初始值**

**Signal clock : bit :='0'; --定义时钟信号类型，初始值为 0**

**Signal count : BIT\_VECTOR(3 DOWNT0 0); --定义 count 为 4 位位矢量**

信号赋值语句:

**目标信号名 <= 表达式;**

**x<=9;**

**Z<=x after 5 ns; -- 在 5ns 后将 x 的值赋予 z**

#### 2.数据类型

##### 2.1 布尔: (Boolean)

TYPE BOOLEAN IS (FALSE, TRUE); -- 取值为 FALSE 和 TRUE，不是数值，不能运算，一般用于关系运算符

## 2.2 位:(Bit)

**TYPE BIT IS ('0','1');** --取值为 0 和 1, 用于逻辑运算

## 2.3 位矢量:(Bit\_Vector)

**TYPE BIT\_VECTOR IS ARRAY (Natural range<>) OF BIT;** -- 基于

Bit 类型的数组, 用于逻辑运算

**SIGNAL a : Bit\_Vector(0 TO 7) ; SIGNAL a : Bit\_Vector ( 7 DOWN TO 0)**

## 2.4 字符: (Character)

**TYPE CHARACTER IS (NUL, SOH,STX, ..., ", '!',...);** --通常用 ‘ ’ 引起来, 区分大小写;

## 2.5 字符串: (String)

**ARIABLE string\_var: STRING (1 TO 7);**

**ring\_var:="A B C D" ;** -- 通常用""引起来, 区分大小写 ;

## 2.6 整数: (Integer)

取值范围  $-(2^{31}-1) \sim (2^{31}-1)$ , 可用 32 位有符号的二进制数表示

**variable a : integer range -63 to 63**

# 3.数据对象操作

## 3.1 属性

属性提供的是关于信号、类型等的指定特性。

3.1.1 event: 若属性对象有事件发生, 则生成布尔值 “true”, 常用来检查时钟边沿是否有效。

上升沿: **Clock' EVENT AND Clock='1'**

3.1.2 range: 生成一个限制性数组对象的范围

' range: “0 to n” ;

' reverse\_range: “n downto 0”

3.1.3 left: 生成数据类型或数据子类型的左边界值;

## 3.2 运算符

非运算、算术运算符 ( NOT, \*\*, ABS)

乘法运算符 (/ , MOD, REM, \* )

正负运算符: +, -,

加减、并置运算符: +, -, &

关系运算符: =, /=, <, >, <=, >=

逻辑运算符: AND, OR, NAND, NOR, XNOR, NOT, XOR

优先级自上而下降低

## 4.语句

### 4.1 并行语句

在结构体中的执行是同时进行，执行顺序与书写顺序无关。

并行信号赋值语句

简单赋值语句

**目标信号名 <= 表达式 (目标信号的数据类型与右边表达式一致)**

选择信号赋值语句

**WITH 选择表达式 SELECT**

**赋值目标信号 <= 表达式 1 WHEN 选择值 1,**

**表达式 2 WHEN 选择值 1,**

**表达式 n WHEN OTHERS ;**

选择值要覆盖所有可能情况，若不能一一指定，用 OTHERS 为其他情况找个出口；选择值必须互斥，不能出现条件重复或重叠的情况。

条件信号赋值语句

**赋值目标信号 <= 表达式 1 WHEN 赋值条件 1 ELSE**

**表达式 2 WHEN 赋值条件 2 ELSE**

**表达式 n WHEN 赋值条件 n ELSE**

**表达式 ;**

各赋值语句有优先级的差别，按书写顺序从高到低排列；各赋值条件可以重叠。

### 4.2 进程语句

进程语句定义顺序语句模块，用于将从外部获得的信号值或内部的运算数据向其他的信号进行赋值。

进程本身是并行语句，但内部是顺序语句；进程只有在特定的时刻（敏感信号发生变化）才会被激活。

**[进程标号:] PROCESS (敏感信号参数表)**

**[声明区] ;**

**BEGIN**

**顺序语句**

**END PROCESS [进程标号] ;**

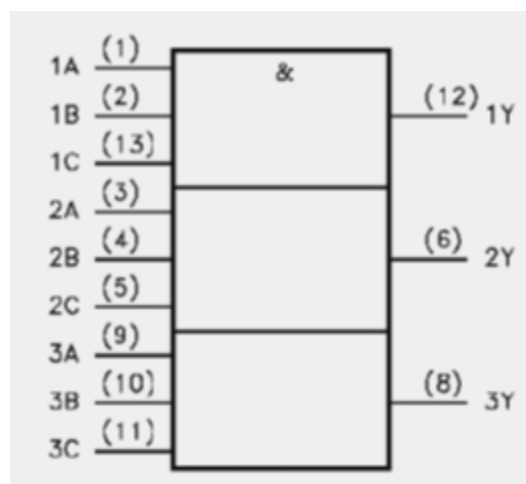
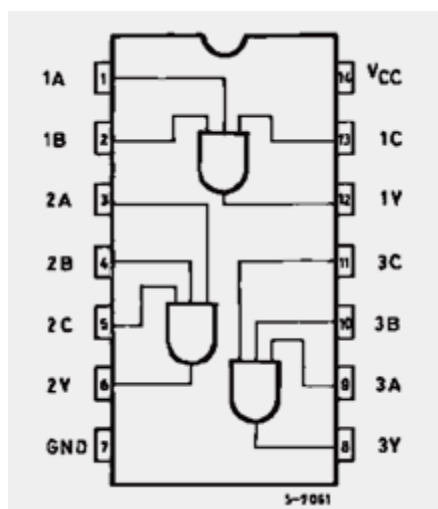
一个进程可以有多个敏感信号，任一敏感信号发生变化都会激活进程

声明区中声明在进程中起作用的局部变量

在每个上升沿启动一次进程（执行进程内所有的语句）。

## 附录二 常用电路逻辑速查

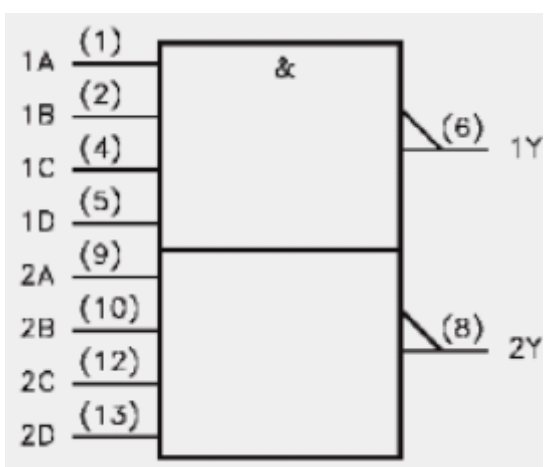
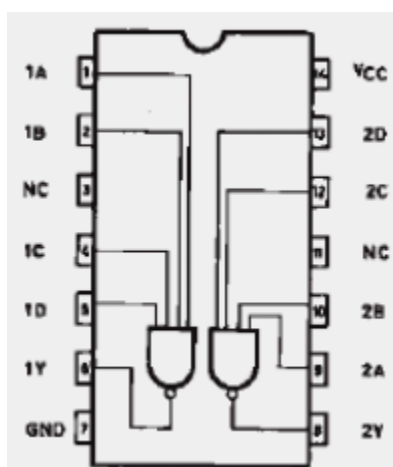
## 1.74LS11——3 输入端 3 与门管脚图及逻辑功能表



A	B	C	Y
L	X	X	L
X	L	X	L
X	X	L	L
H	H	H	H

X : Don't Care

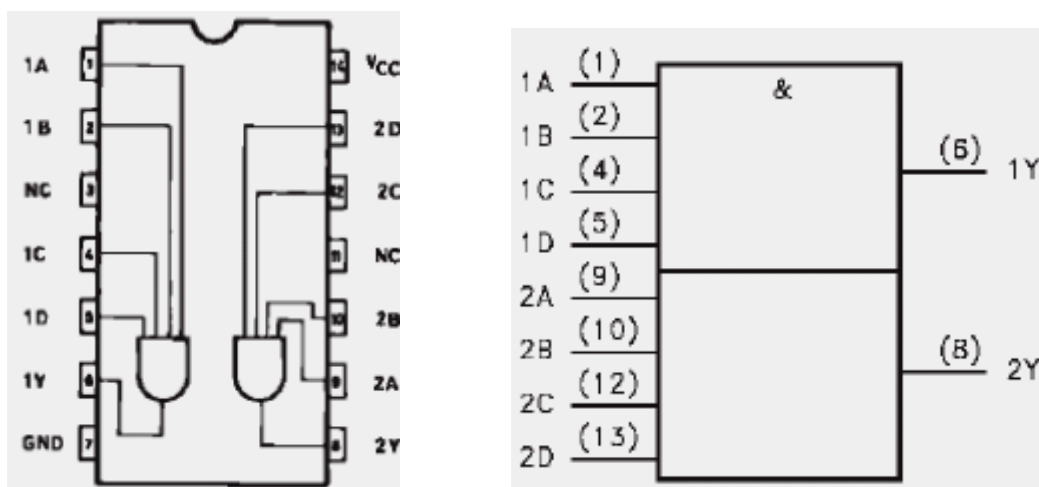
## 2.74LS20——4 输入端双与非门管脚图及逻辑功能表



TRUTH TABLE

A	B	C	D	Y
L	X	X	X	H
X	L	X	X	H
X	X	L	X	H
X	X	X	L	H
H	H	H	H	L

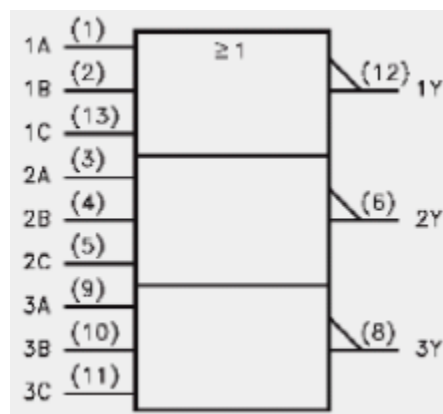
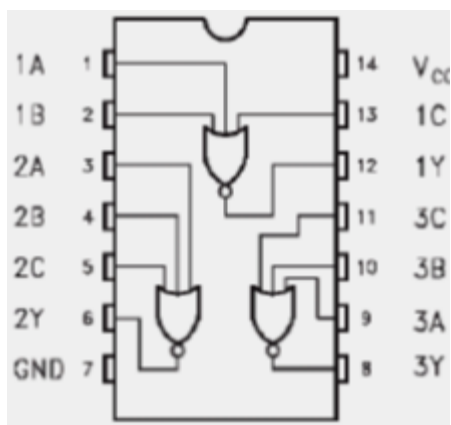
3.74LS21——4 输入端双与门管脚图及逻辑功能表



A	B	C	D	Y
L	X	X	X	L
X	L	X	X	L
X	X	L	X	L
X	X	X	L	L
H	H	H	H	H

4.74LS27——3 输入端三或非门管脚图及逻辑功能表



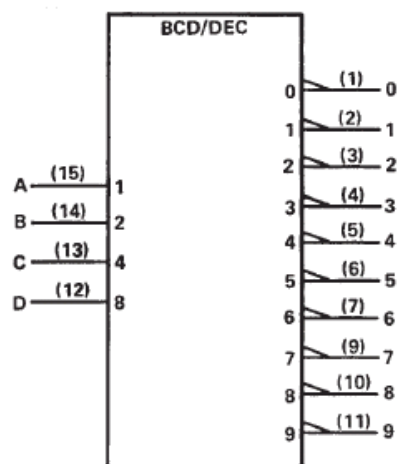
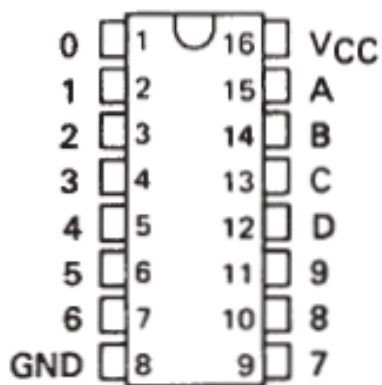


TRUTH TABLE

A	B	C	Y
L	L	L	H
H	X	X	L
X	H	X	L
X	X	H	L

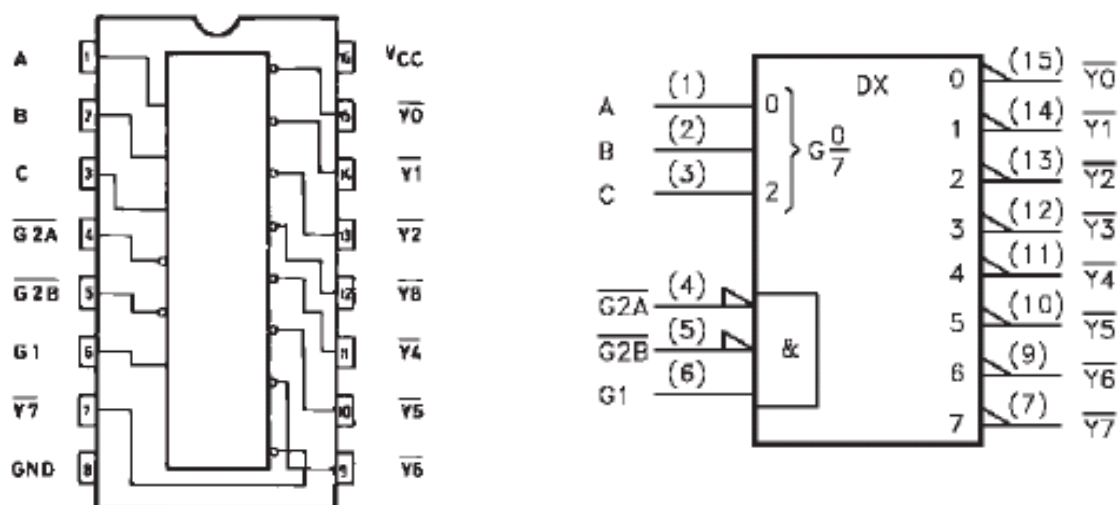
X : Don't Care

## 5.74LS42——BCD/十进制译码器管脚图及逻辑功能表



NO.	BCD INPUT				DECIMAL OUTPUT									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

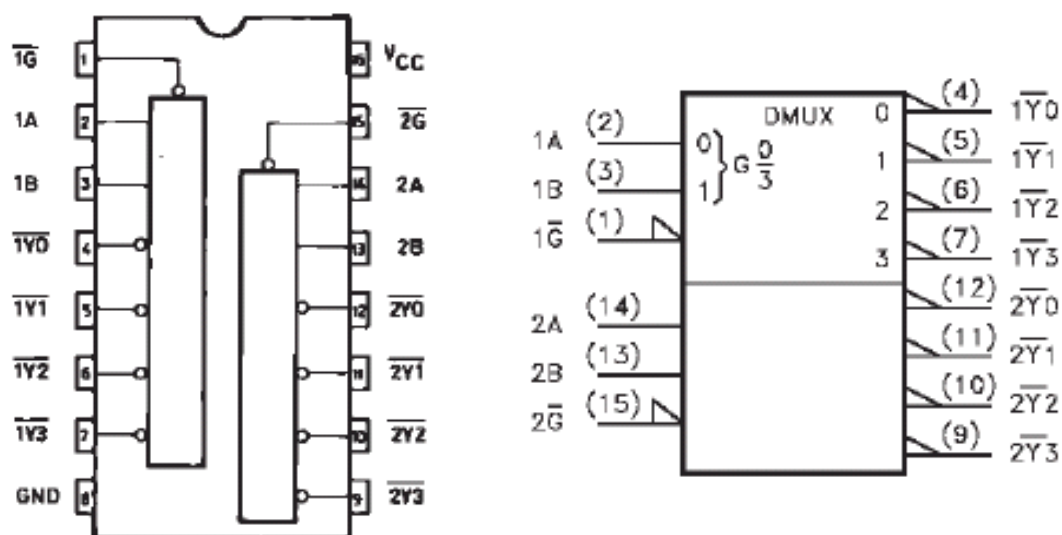
6.74LS138——3-8 线译码器管脚图及逻辑功能表



INPUTS						OUTPUTS							
ENABLE			SELECT										
$\overline{G2B}$	$\overline{G2A}$	G1	C	B	A	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$
X	X	L	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
H	X	X	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

X : Don't Care

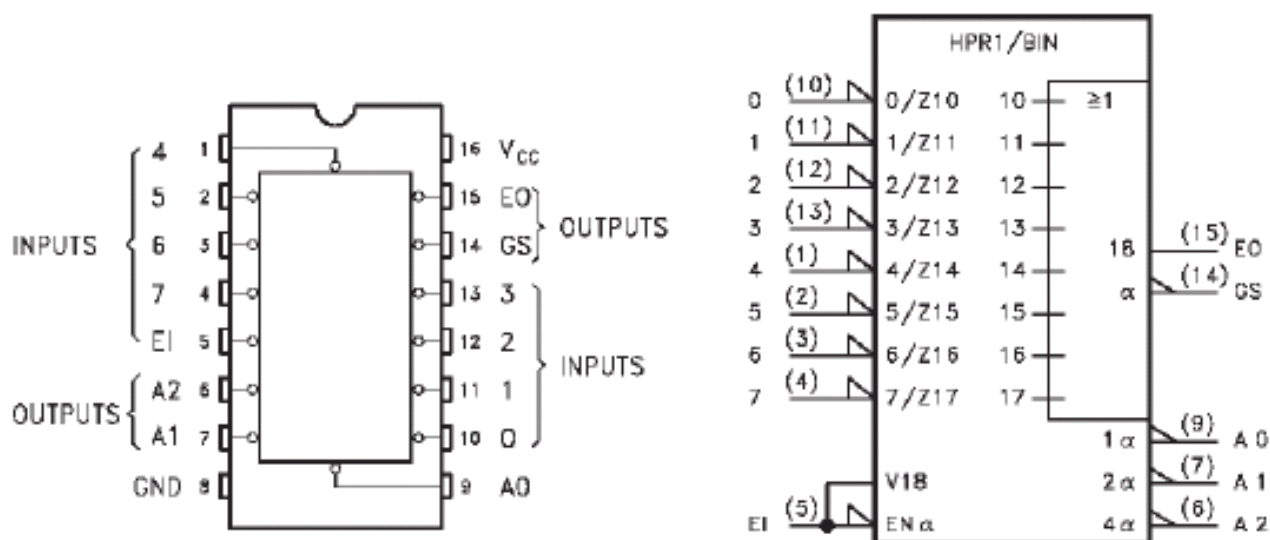
## 7.74LS139——双 2-4 线译码器管脚图及逻辑功能表



TRUTH TABLE

INPUTS			OUTPUTS				SELECTED OUTPUT
ENABLE	SELECT		$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	
$\overline{G}$	B	A					
H	X	X	H	H	H	H	NONE
L	L	L	L	H	H	H	$\overline{Y}_0$
L	L	H	H	L	H	H	$\overline{Y}_1$
L	H	L	H	H	L	H	$\overline{Y}_2$
L	H	H	H	H	H	L	$\overline{Y}_3$

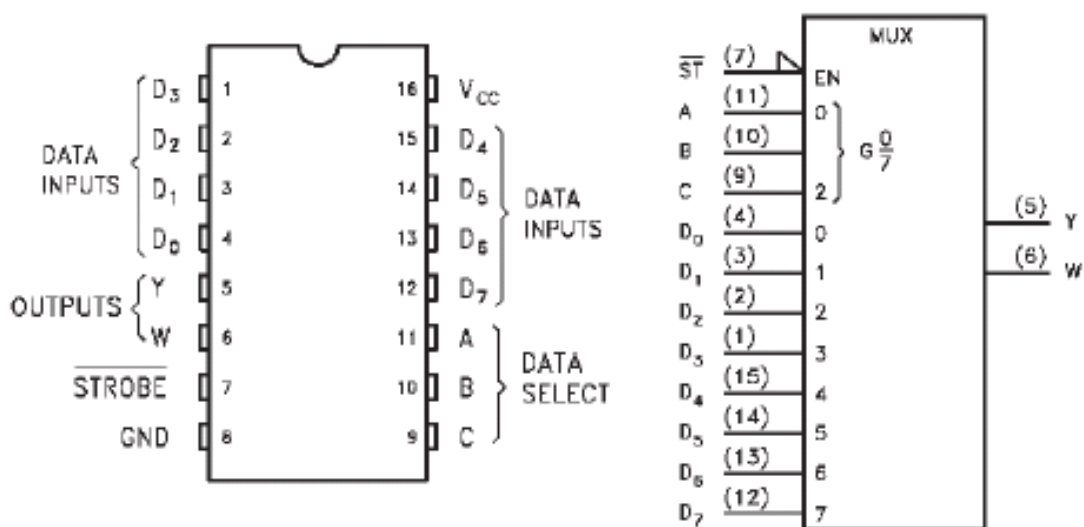
8.74LS148——8-3 线优先编码器管脚图及逻辑功能表



INPUTS									OUTPUTS				
E1	0	1	2	3	4	5	6	7	A2	A1	A0	GS	E0
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

X: Don't Care

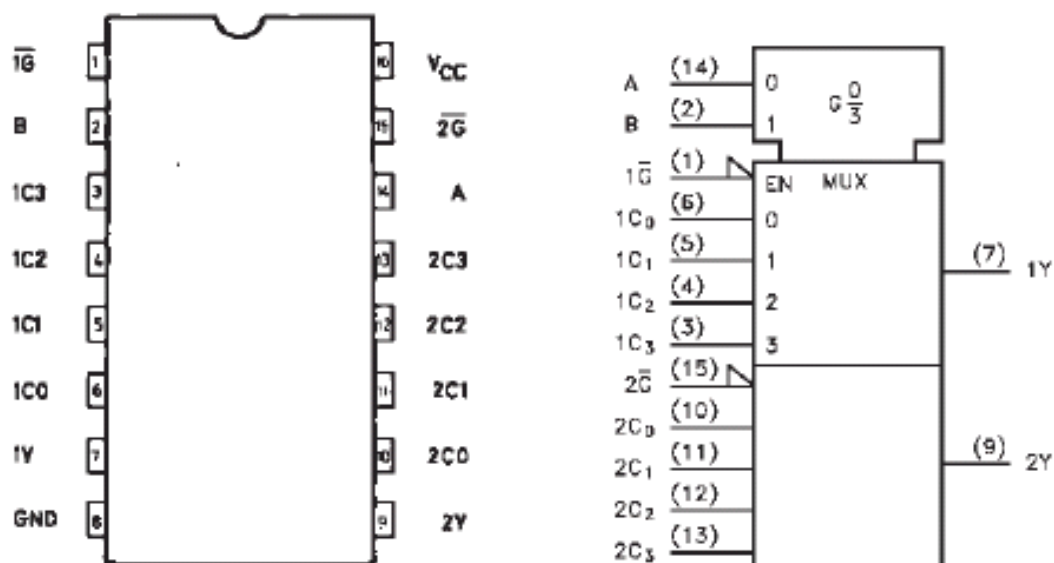
## 9.74LS151——八选一数据选择器管脚图及逻辑功能表



INPUTS				OUTPUTS	
SELECT			$\overline{\text{STROBE}}$	Y	W
C	B	A	S		
X	X	X	H	L	H
L	L	L	L	D <sub>0</sub>	$\overline{\text{D}}_0$
L	L	H	L	D <sub>1</sub>	$\overline{\text{D}}_1$
L	H	L	L	D <sub>2</sub>	$\overline{\text{D}}_2$
L	H	H	L	D <sub>3</sub>	$\overline{\text{D}}_3$
H	L	L	L	D <sub>4</sub>	$\overline{\text{D}}_4$
H	L	H	L	D <sub>5</sub>	$\overline{\text{D}}_5$
H	H	L	L	D <sub>6</sub>	$\overline{\text{D}}_6$
H	H	H	L	D <sub>7</sub>	$\overline{\text{D}}_7$

X: Don't Care

10.74LS153——双 4 选 1 数据选择器管脚图及逻辑功能表



TRUTH TABLE

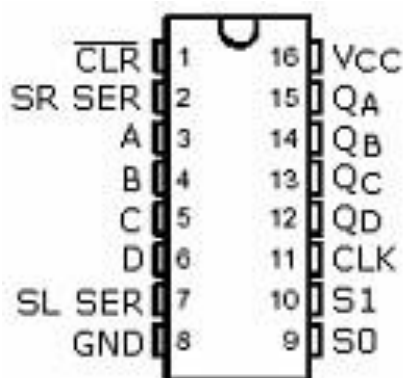
SELECT INPUTS		DATA INPUTS				STROBE	OUTPUT Y
B	A	C <sub>0</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	$\overline{G}$	
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

X : Don't Care

## 11.74LS47——4 线 7 段显示译码器，低电平有效，驱动共阳数码管

Decimal or Function	Inputs							Outputs						
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI/RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H
$\overline{BI}$	X	X	X	X	X	X	L	H	H	H	H	H	H	H
$\overline{RBI}$	H	L	L	L	L	L	L	H	H	H	H	H	H	H
$\overline{LT}$	L	X	X	X	X	X	H	L	L	L	L	L	L	L

## 12.74LS194——4 位移位寄存器管脚图及逻辑功能



功能表

输				入				输 出					
清零	模式		时钟 CLK	串行 SER		并 行				Q <sub>A</sub>	Q <sub>B</sub>	Q <sub>C</sub>	Q <sub>D</sub>
$\overline{\text{CLR}}$	S <sub>1</sub>	S <sub>0</sub>		左 SL	右 SR	A	B	C	D				
L	×	×	×	×	×	×	×	×	×	L	L	L	L
H	×	×	L	×	×	×	×	×	×	Q <sub>AO</sub>	Q <sub>BO</sub>	Q <sub>CO</sub>	Q <sub>DO</sub>
H	H	H	↑	×	×	a	b	c	d	a	b	c	d
H	L	H	↑	×	H	×	×	×	×	H	Q <sub>An</sub>	Q <sub>Bn</sub>	Q <sub>Cn</sub>
H	L	H	↑	×	L	×	×	×	×	L	Q <sub>An</sub>	Q <sub>Bn</sub>	Q <sub>Cn</sub>
H	H	L	↑	H	×	×	×	×	×	Q <sub>Bn</sub>	Q <sub>Cn</sub>	Q <sub>Dn</sub>	H
H	H	L	↑	L	×	×	×	×	×	Q <sub>Bn</sub>	Q <sub>Cn</sub>	Q <sub>Dn</sub>	L
H	L	L	×	×	×	×	×	×	×	Q <sub>AO</sub>	Q <sub>BO</sub>	Q <sub>CO</sub>	Q <sub>DO</sub>

a、b、c、d=分别为 A、B、C 或 D 输入端上稳定状态输入的电平。

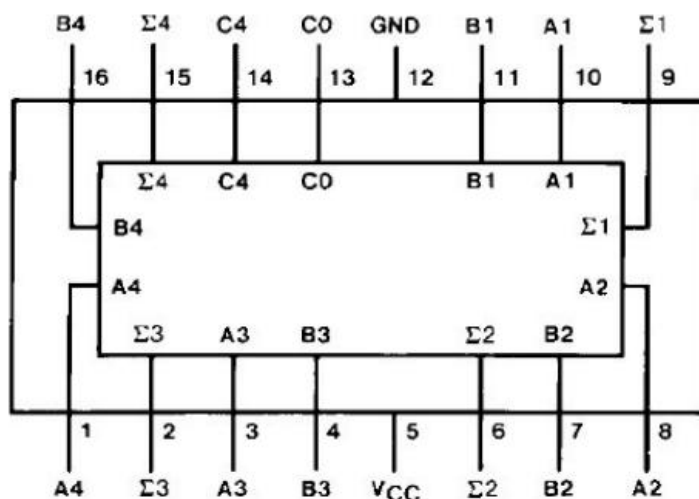
$Q_{AO}$ 、 $Q_{BO}$ 、 $Q_{CO}$ 、 $Q_{DO}$ =在已建立稳定状态输入条件之前  $Q_A$ 、 $Q_B$ 、 $Q_C$ 、 $Q_D$  相应的电平。

$Q_{An}$ 、 $Q_{Bn}$ 、 $Q_{Cn}$ 、 $Q_{Dn}$ =在时钟最新  $\uparrow$  跃变之前的  $Q_A$ 、 $Q_B$ 、 $Q_C$ 、 $Q_D$  的电平。

H=高电平 L=低电平 X=不定  $\uparrow$ =从低电平转换到高电平

### 13.74LS83——4 位加法器

Connection Diagram





Truth Table

Inputs				Outputs					
				When C0 = L			When C0 = H		
A1	B1	A2	B2	$\Sigma 1$	$\Sigma 2$	C2	$\Sigma 1$	$\Sigma 2$	C2
A3	B3	A4	B4	$\Sigma 3$	$\Sigma 4$	C4	$\Sigma 3$	$\Sigma 4$	C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	L	H	L	H	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	L	L	H
H	L	H	L	H	H	L	L	L	H
L	H	H	L	L	L	H	H	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	L	L	H
H	L	L	H	H	H	L	L	L	H
L	H	L	H	L	L	H	H	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	H	L	H	L	H	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

H = HIGH Level, L = LOW Level

Input conditions at A1, B1, A2, B2, and C0 are used to determine outputs  $\Sigma 1$  and  $\Sigma 2$  and the value of the internal carry C2. The values at C2, A3, B3, A4, and B4 are then used to determine outputs  $\Sigma 3$ ,  $\Sigma 4$ , and C4.