现代EDA综合实验系统使用说明书

南昌大学通信实验中心

目　录

第一章 MAX PLUS II 软件的使用

1.1 MAX PLUS II 软件介绍

1. S-Site 授权 及 PLS-WEB 特点

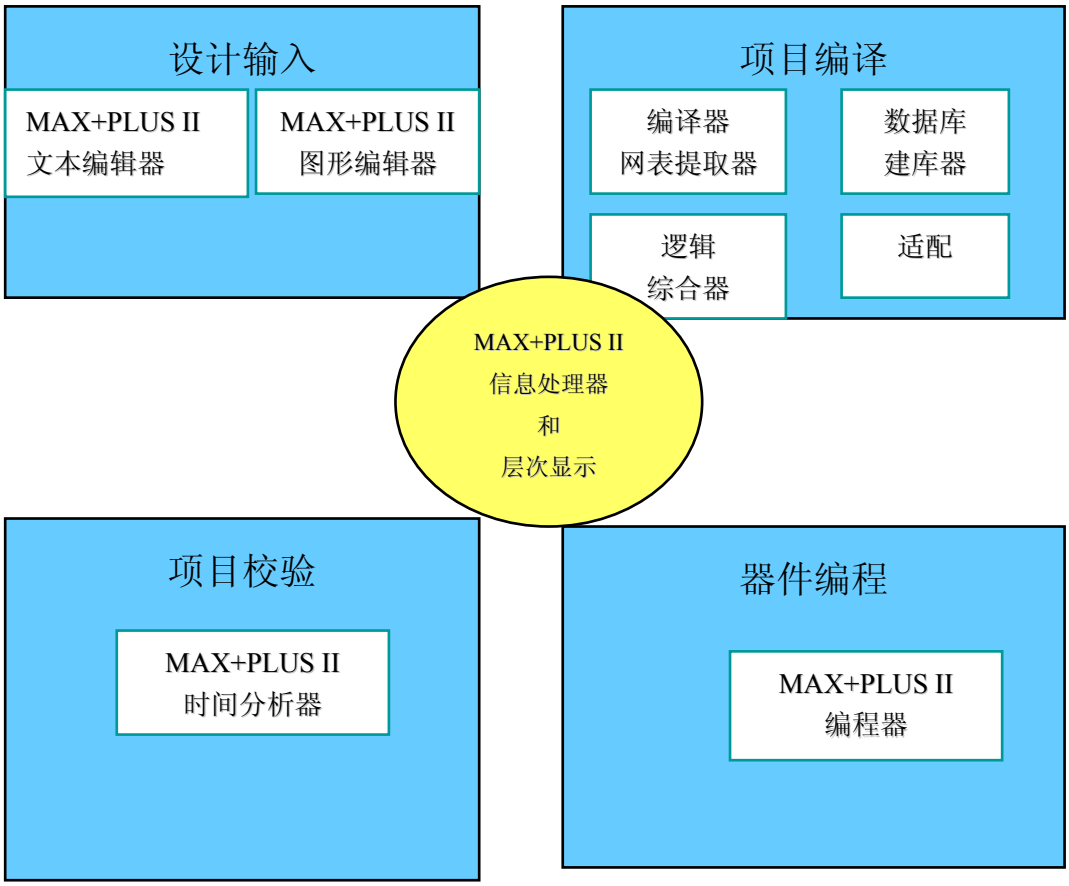


图 1.1

ES-Site & PLS-WEB 允许用户使用 Classic 系列, MAX5000 系列, MAX7000(S) 系列以及EPM9320, EPF8282A/EPF8452A,EPF6016, EPF10K10器件完成设计。

2. Max+Plus II 的安装

第一次运行 MAX+PLUS II ，双击 MAX+PLUS II 图标 或在 开始 菜单内选择 MAX+PLUS II 项，开始运行 MAX+PLUS II。

第一次运行 MAX+PLUS II 时， 将会出现如下的窗口， 选择 **ES site License** 按钮。

申请授权代码：您可以通过访问 Alter 公司 的 www 站点:
http://www.Altera.com 获得授权代码。

3. MAX+PLUS II 管理器窗口

ES-Site 授权有效后，您将返回到 MAX+PLUS II 管理器窗口：

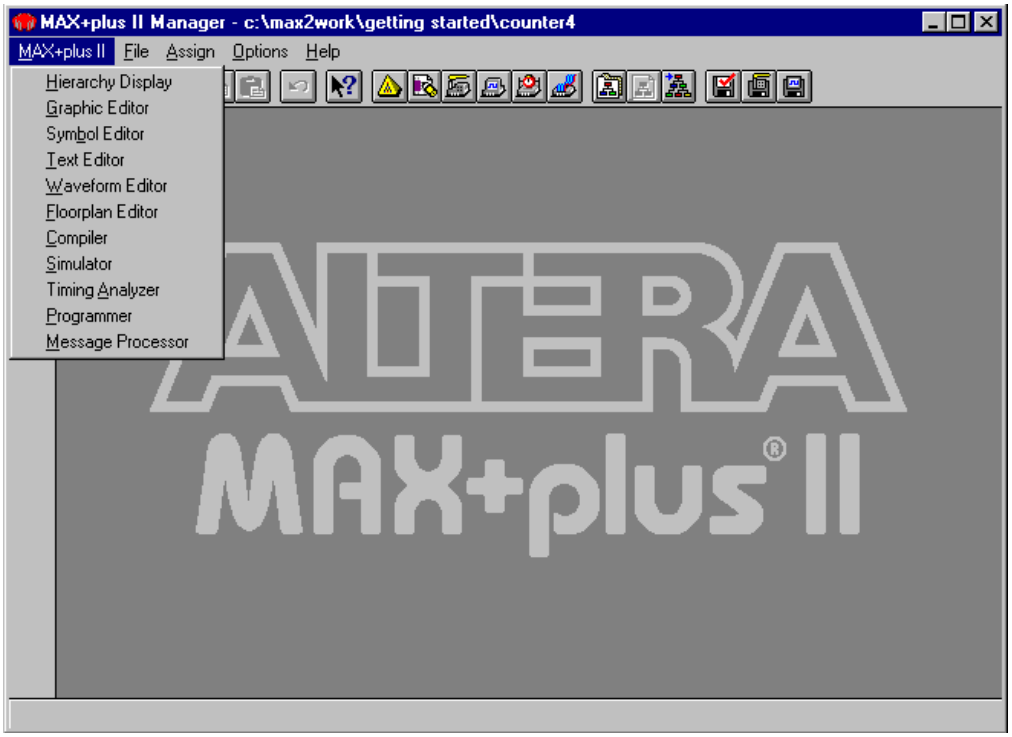


图 1.2

1. 2 原理图输入法

下面以一个十进制的设计为例，详细介绍原理图输入的设计方法。

1）指定设计项目文件名

MAX PLUS II 编译器的工作对象是项目，所以在进行一个设计时，首先要指定该设计的项目名称，并且要保证一个设计项目中所有的文件都出现在该目录的层次结构中。对于每个新的项目，应该建立一个独立的子目录，不能在根目录下直接建立项目设计。如果这个子项目不存在，MAX PLUS II 将自动创建。注意：每个设计必须有一个项目名，并且要保证项目名与设计文件名一致。

选中菜单项：File/Project/Name,在图 1.3 中的 Project Name 对话框中键入：e:\exam\demo1。按 OK 键，出现图 1.3 中的小对话框，回答是（Y），建立子目录：
e:\exam。并且工程名为 demo1。

2）建立一个新的原理图文件

1. 在 File 菜单中，选择 New…，出现图 1.4，

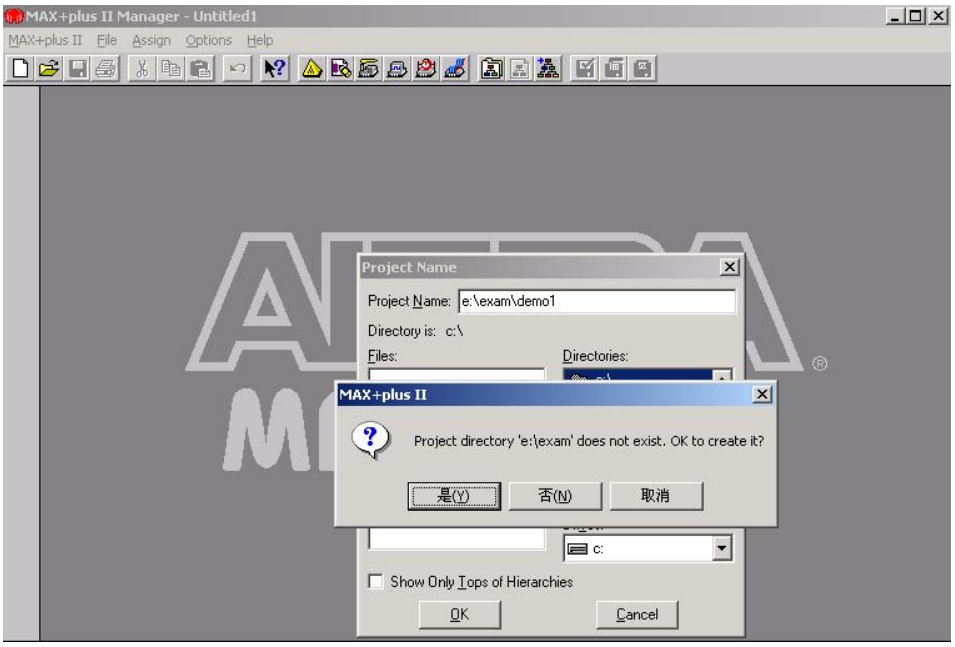


图 1.3

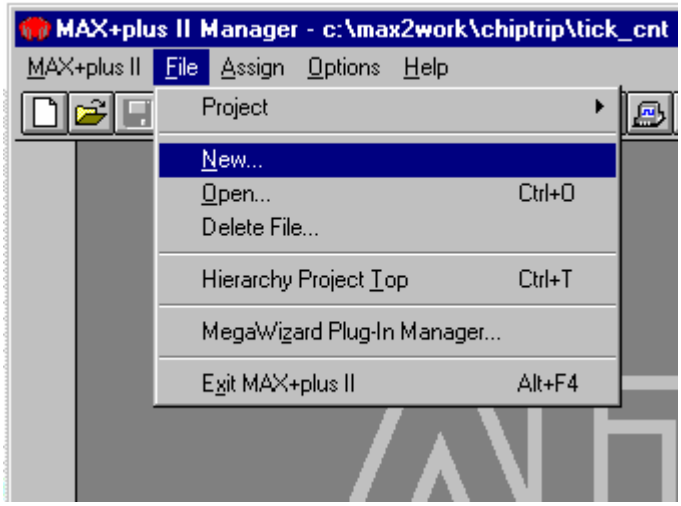


图 1.4

2. 选择 Graphic Editor，出现图 1.5，然后按下 **OK** 按钮， 将会出现一个无标题的图形编辑窗口,如图 1.6 所示。

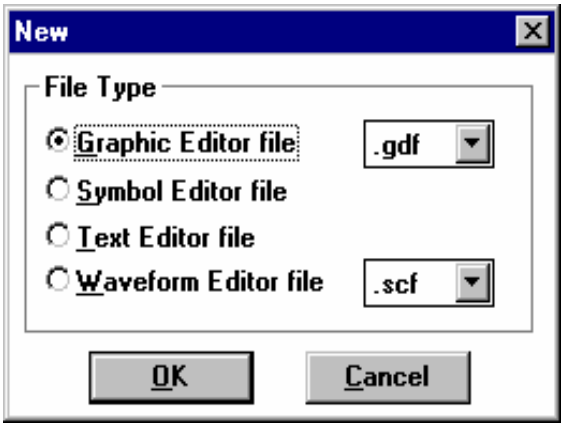

New
File Type
Graphic Editor file
.gdf
Symbol Editor file
Text Editor file
Waveform Editor file
.scf
OK
Cancel

图 1.5

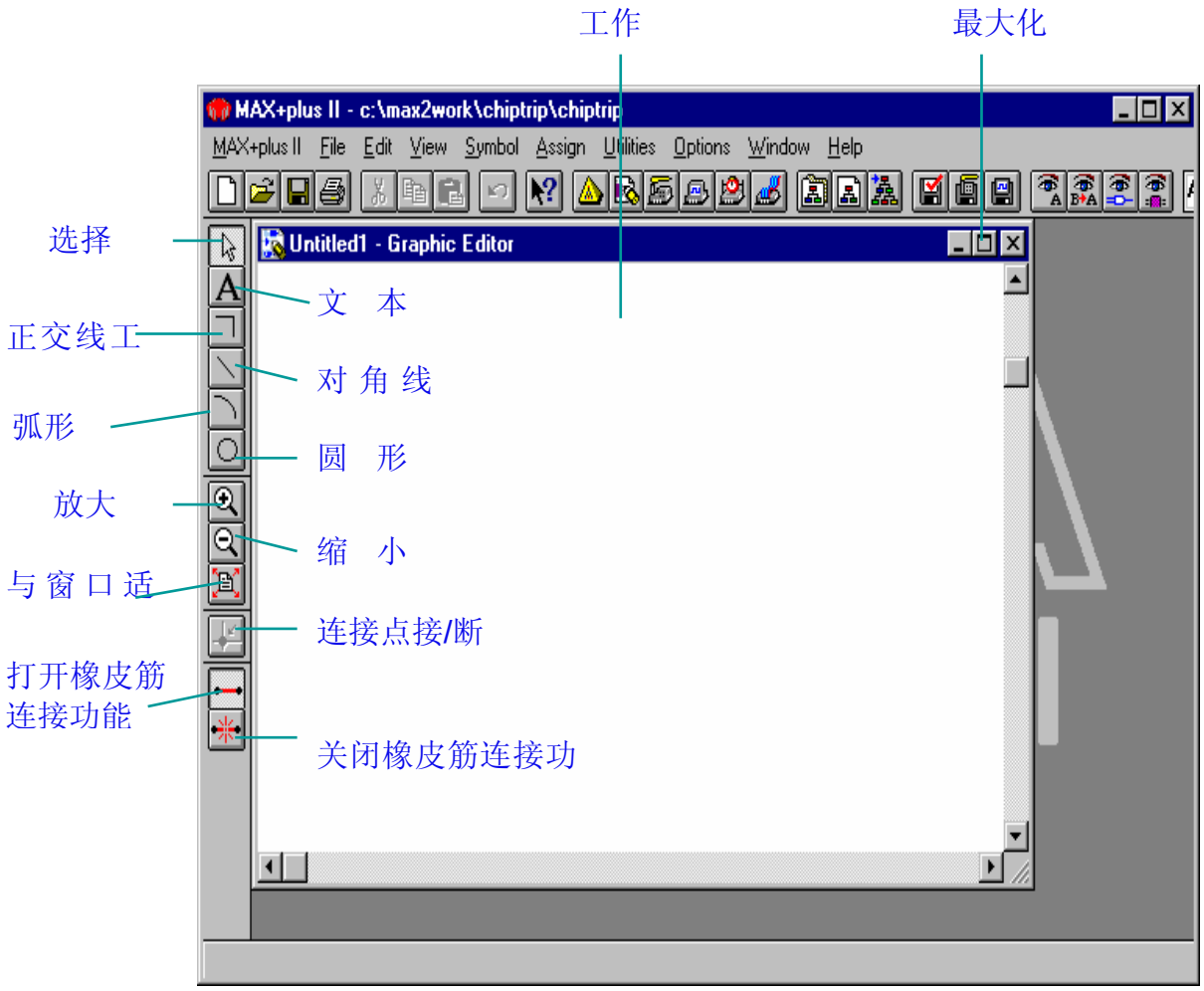

工作
最大化
MAX+plus II - c:\max2work\chiptrip\chiptrip
MAX+plus II File Edit View Symbol Assign Utilities Options Window Help
Untitled1 - Graphic Editor
选择
文 本
正交线工
对 角 线
弧形
圆 形
放大
缩 小
与 窗 口 适
连接点接/断
打开橡皮筋
连接功能
关闭橡皮筋连接功

图 1.6

点击 Assign\Device 菜单，如图 1.7；选择器件，如 EP1K100QC208-3，按 OK。

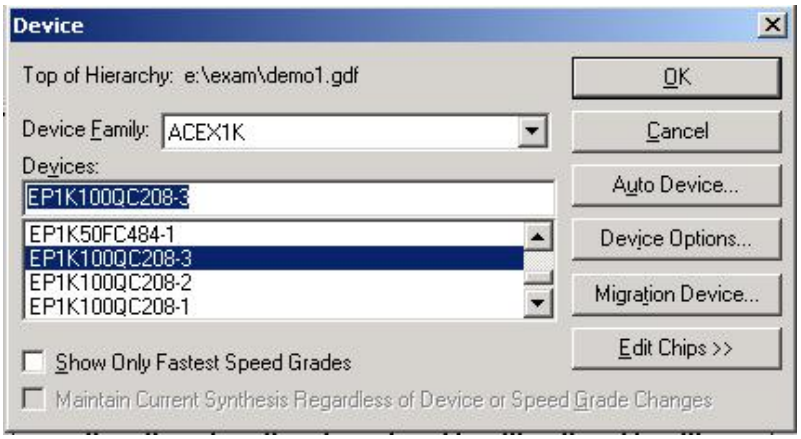


图 1.7

3.选择采单 File\Save,出现图 1.8 的窗口。按 OK，即将 demo1.gdf 文件保存到当前项目的子目录下。

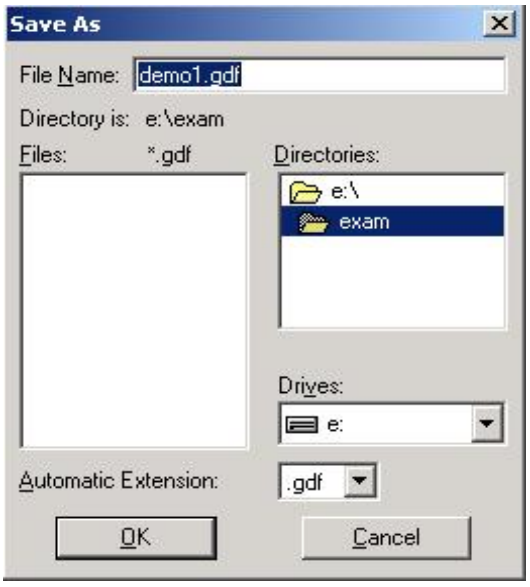


图 1.8

4．输入 Altera 图元，选择工具按钮有效时，在图形编辑器窗口的空白处单击鼠标左键以确定输入位置，然后选择 **Enter Symbol**，或双击鼠标左键。将出现一个 **Enter Symbol** 对话框，如图 1.9 所示。在 **symbol Libraries** 框中 选择 “..\maxplus2\max2lib\prim” 。

所有的 Altera 图元以列表方式显示出来，选择您想输入的图元，如 nand2，然后选择 **OK。**

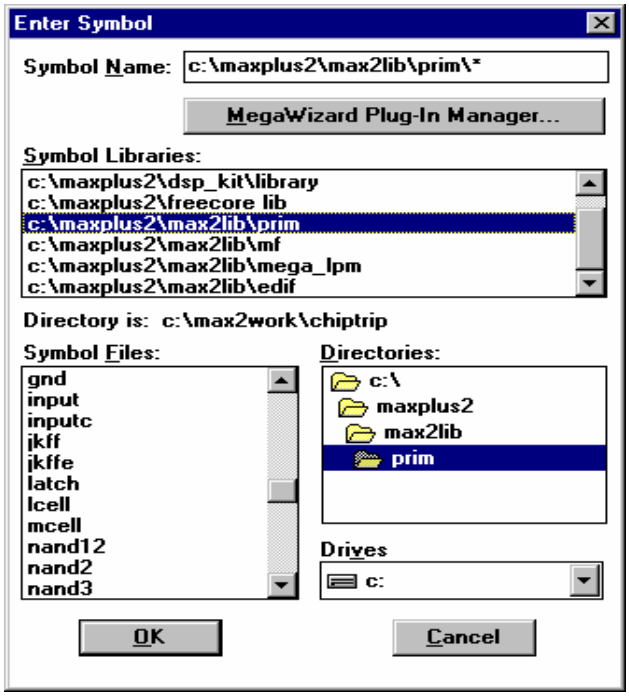


图 1.9

点击 nand2 符号，即选定这个符号。按住鼠标左键并移动到适合的位置。在 nand2 符号上，右击鼠标，通过 Rotate,Flip Horizontal 或 Flip Vertical 项，可分别对 nand2 符号进行旋转，水平或垂直镜像操作。

5．输入 74 系列的符号：MAX+PLUS II 为实现不同的逻辑功能提供了许多符号，如：图元符号 ,兆功能符号和宏功能符号。在图形编辑器文件中可直接使用以上符号。74 系列符号的输入方法和上页图元输入的方法相同。当 **Enter Symbol** 对话框出现后，在 **symbol Libraries** 对话框中选择 “..\maxplus2\max2lib\mf”路径。

在 **Symbol Files** 对话框中，选择您需要的 74 系列符号，如 74161 等。

注意：输入 LPM 符号：nlpm (library parameterized megafunction) 符号的输入方法与先前符号的输入方法相同。

在 **Enter Symbol** 对话框出现后，在 **symbol Libraries** 框中选择 “..\maxplus2\max2lib\mega_lpm” 路径。

在 **Symbol Files** 框中选择您需要的 lpm 符号。

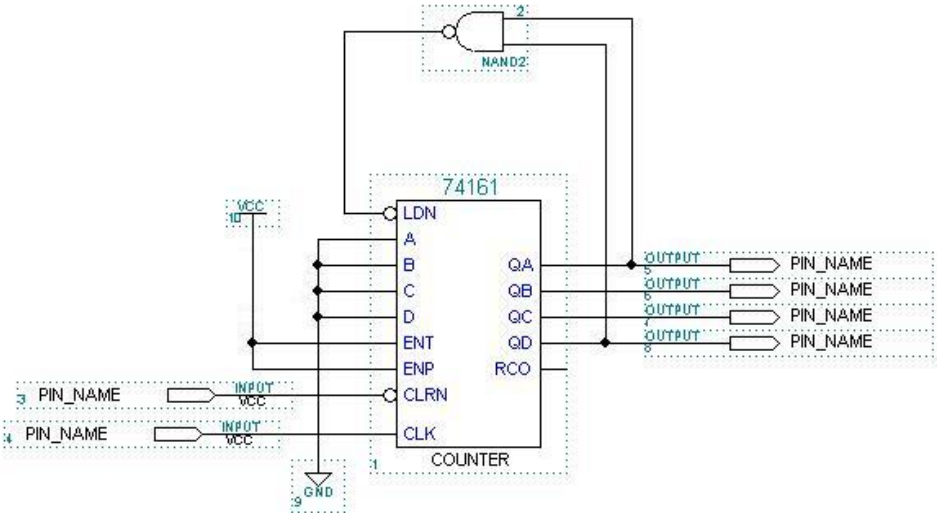
双击参数框 (位于符号的右上角)，输入您需要的 lpm 的参数。在 **Port Status** 框中选择 **Unused** ，可将您不需要的信号去掉。

6.输入 Input 和 Output 引脚：在原理图的空白处双击鼠标即可显示 Enter Symbol 对话框，在对话框中输入 INPUT，选 OK，即显示 INPUT 符号；在对话框中输入 OUTPUT，选 OK，即显示 OUTPUT 符号。

在 INPUT 符号上同时按下 Ctrl 键和鼠标左键，拖动鼠标到该符号的下方再放开，就复制了该符号。

地和电源的符号同上面方法，输入 GND 和 VCC。

7. **连线：**如果需要连接两个端口，将您的鼠标移到其中一个端口，则鼠标自动变为‘+’形状。 一直按住鼠标的左键并将鼠标拖到第二个端口。放开左键 ，则一条连接线被画好了。如果您需要删除一根连接线，单击这根连接线并按 **Del** 键。如图 1.10。



如 图 1.10

8. **为管脚和节点命名：** 在管脚上的 PIN_NAME 处双击鼠标左键，然后输入名字。 选中需命名的线，然后输入名字，如 CLK，CLR，Q3，Q2，Q1，Q0 等。 对 n 位宽的总线 A 命名时，您可以采用 A[n-1..0] 形式，其中单个信号用 A0, A1, A2, ….., An 形式 。

最后完成的电路图如图 1.11 所示。

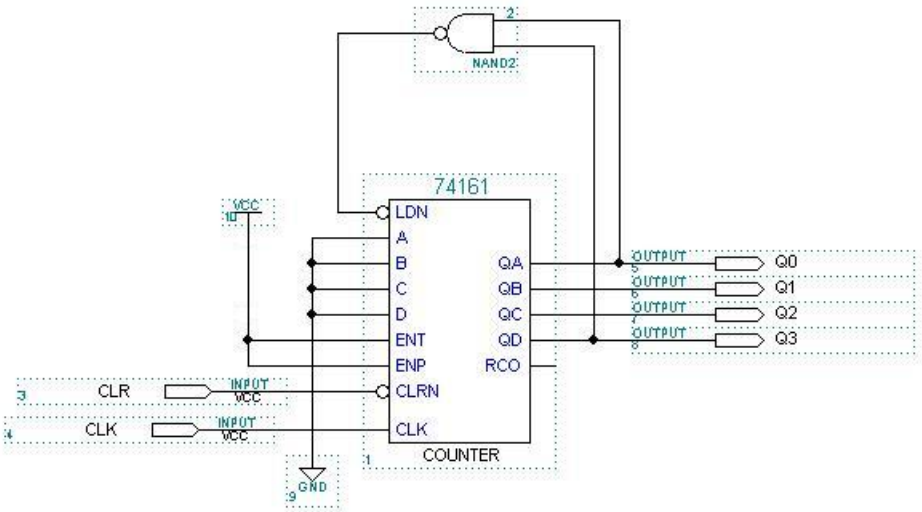


图 1.11

9.保存原理图：选择 **File** 菜单中的 **Save As** 项. 将出现 **Save As** 对话框。

10.编译：在 **MAX+PLUS II** 菜单内选择 **Compiler** 项。则出现编译器窗口，按 Start 开始编译，如图 1.12 所示。

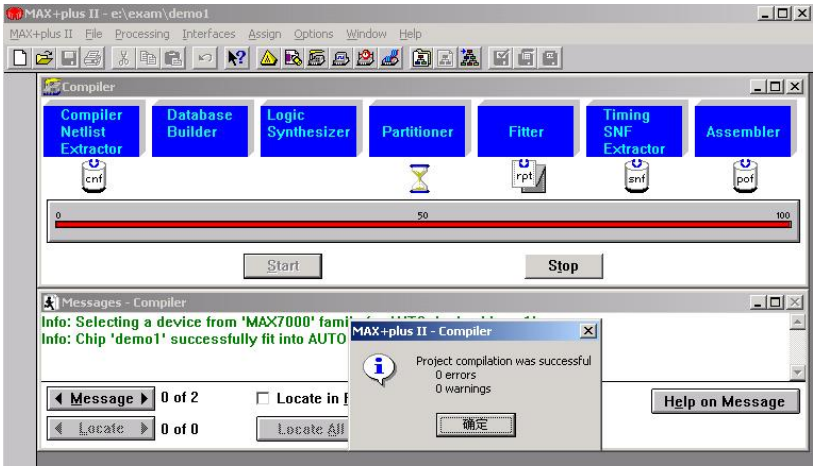


图 1.12

MAX+PLUS II 编译器将检查项目是否有错，并对项目进行逻辑综合，然后配置到一个 Altera 器件中，同时将产生报告文件、编程文件和用于时间仿真用的输出文件。但是，在开始编译前，我们还必须设定一些别的选项 。

11.时序模拟仿真：

建立波形输入文件，在 File/New 菜单下选择 Waveform Editor file .scf,如图 1.13。

图 1.13

波形编辑器窗口如图 1.14。在 Name 下方单击鼠标右键，出现浮动菜单，选择 Enter Nodes from SNF……，进入如图 1.15，单击 List,则在 Available Nodes &Groups 中出现所有输入，输出节点名称，选择需要仿真的节点名称，单击=>按钮，则它们被添加到 Selected Nodes & Groups 中，按 OK。

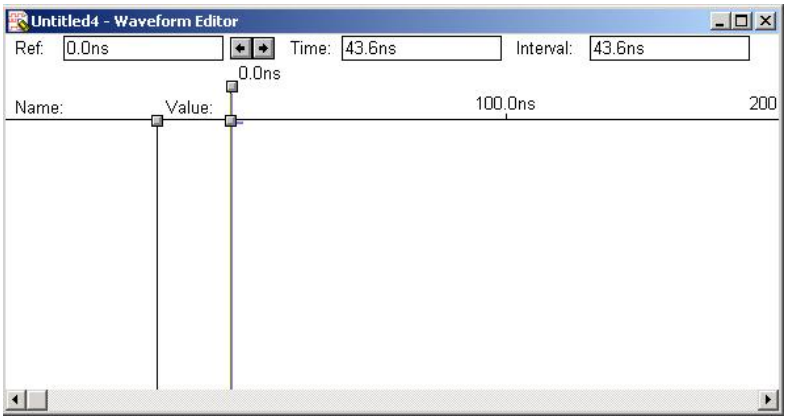


图 1.14

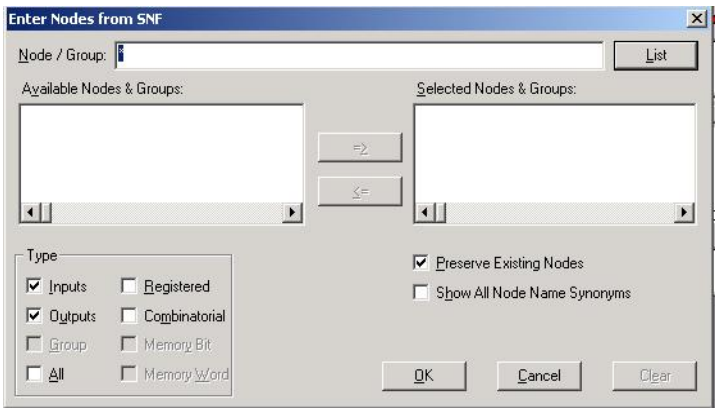


图 1.15

12.编辑输入节点波形：在菜单 Options 中选择 Snap to Grip,Show Grid,并打开 Grid Size……，设置网格大小，如 20.0ns,并点击 File/End Time 设置模拟时间的长短，如 2μs 。

如图 1.16，编辑 CLK 波形，点击 Name 中的 CLK，然后，右击采单选中 Overwrite 命令来编辑波形。

编辑 CLR 波形，点击 Name 中的 CLR，然后点击图形工具按钮  ，则用高电平覆盖了整个 CLR 波形，在 CLR 的起始地方按下左键，拖动鼠标到 80.0ns 处再松开，然后点击图形工具按钮  ，则该区间设置为低电平。用同样的方法，可以设置任意区间为高或为低电平。

设置完成后，保存。即点击 File/Save 保存仿真文件，后缀名为 scf。

然后点击 File/Close，关闭波形编辑器窗口。

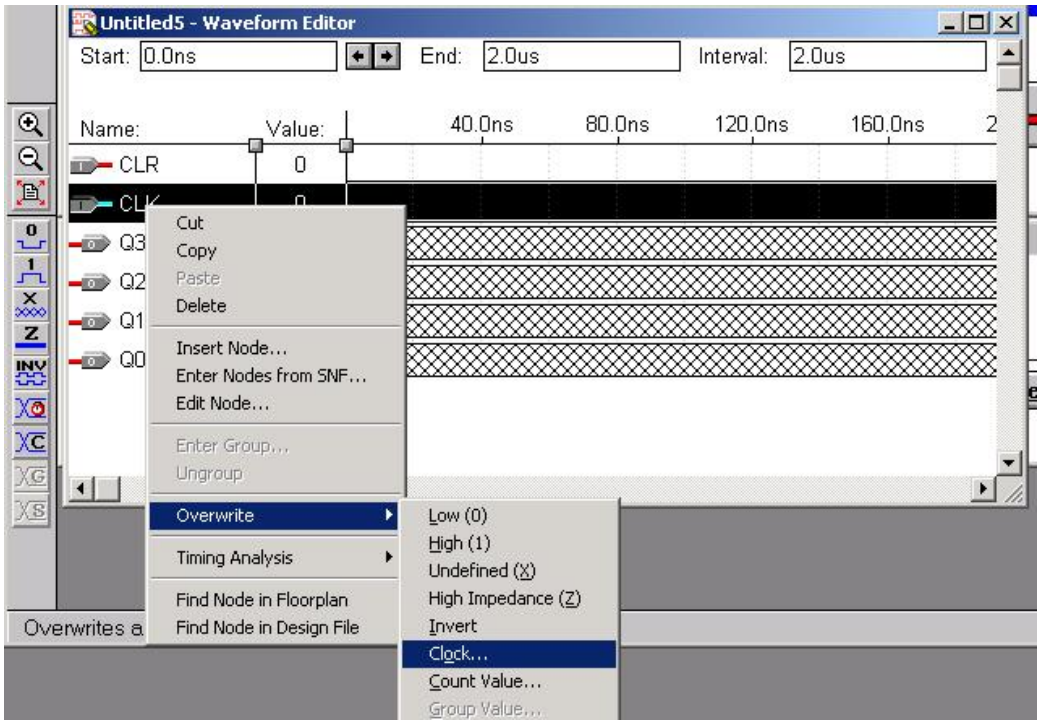


图 1.16

13．时序模拟仿真：选择 MAX PLUS II\Simulator 菜单，点击 Start,开始模拟仿真。

仿真完成后，单击 Open SCF，打开仿真结果，如图 1.17。

可以将 Q3，Q2，Q1，Q0 选为一个组，方法是如同资源管理器下选多个文件，首先选中 Q3，按住 Shift,再依次选中 Q2，Q1，Q0，然后按右键，在弹出的窗口中，选择 Enter Group…，将会出现如图 1.18 所示的显示。如果波形没有完全显示出来，可以用  或  来缩小或放大显示波形。

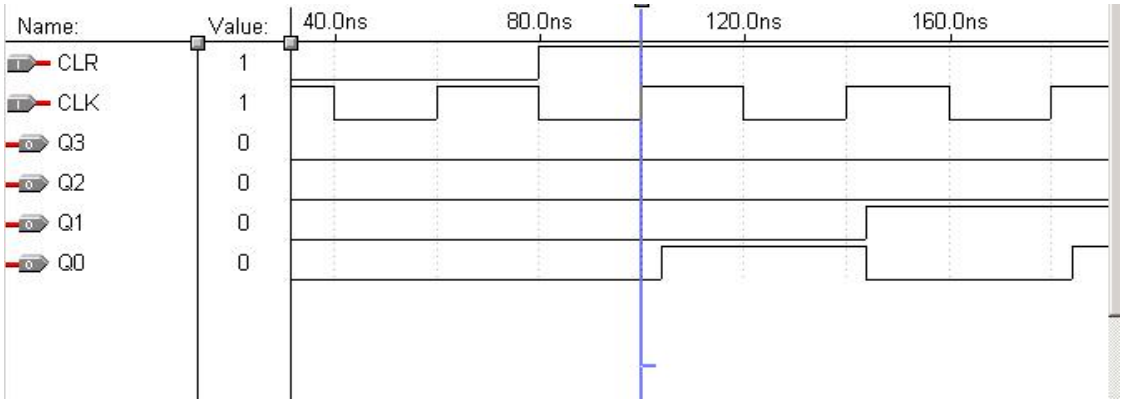


图 1.17

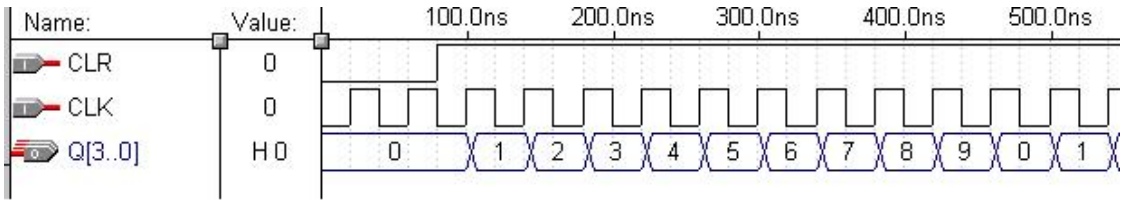


图 1.18

14．管脚分配：仿真正确后，就可以准备下载到 FPGA/CPLD 芯片中进行验证了。

在 **MAX+PLUS II** 菜单内选择 Floorplan Editor 菜单，则显示该设计项目的信号列表和目标芯片的管脚，如图 1.19。

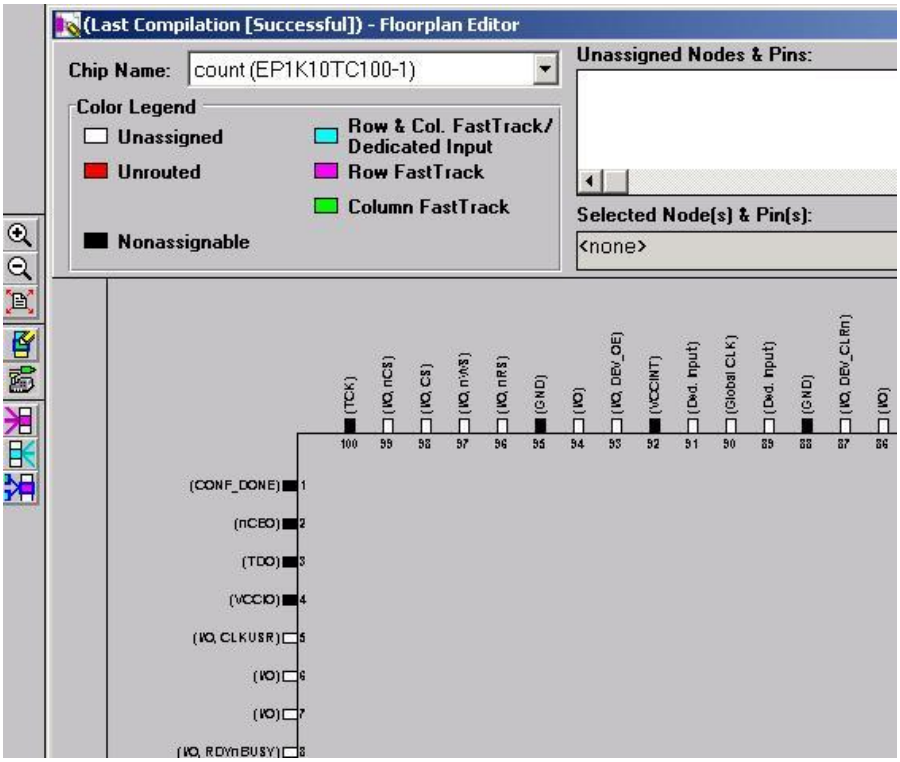


图 1.19

按  键，所有输入和输出出现在 Unassigned Nodes 栏中内，可以用手动方式分配管脚。

方法：用鼠标左键按住某输入/输出口名称，拖到下面芯片的某一管脚上，松开鼠标，并完成了一个管脚的分配。

在本实验指导书中，管脚的分配参看第二章的实验装置的下载板的引脚表。

管脚分配完成后，再进行一次编译。

15．下载：

用并口连接线将计算机与实验箱的下载口相连，打开实验相电源。

在 **MAX+PLUS II** 菜单内选择 Programmer 菜单,出现如图 1.20。在 Hardware Type 中选择 ByteBlaster(MV)。

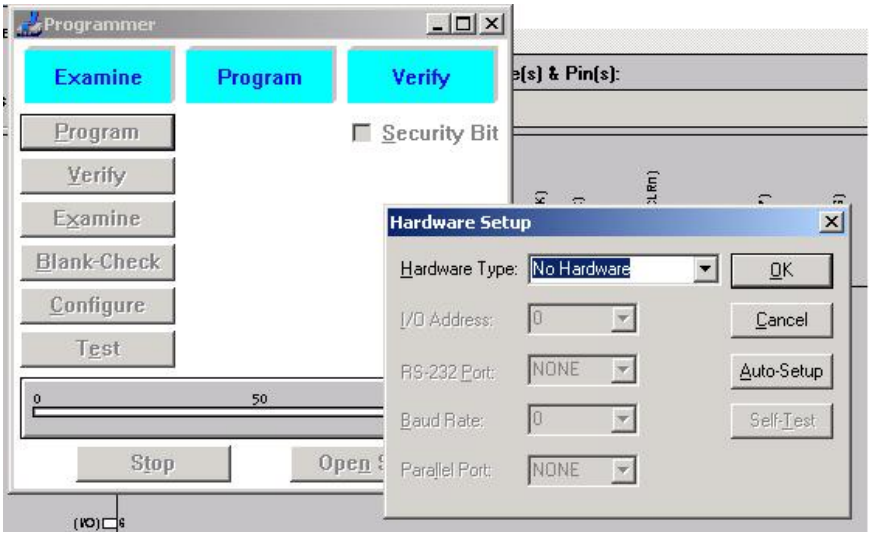


图 1.20

选中下载文件，demo1.sof,按 Configure 键即可完成下载。

16．实验验证：根据设置的管脚，在实验箱上，进行相应的操作，验证实验的结果。

1. 2 Verilog HDL 语言设计法

同原理图设计方法，首先建立工程文件。

点击 File/New 菜单，选择 Text Editor file,按 OK。在文本输入窗口中输入如下程序。

```
module count(clk,q,ncr);
  output[3:0]q;
  reg[3:0]q;

  input clk,ncr;

  always @(posedge clk or negedge ncr)
   begin
    if(~ncr)q<=0;
      else begin
    if (q[3:0]==4'b1001) q<=0;
       else
         q<=q+1;
      end
   end
```

endmodule

输入完后，点击 File/Save as ,出现图 1.21。输入文件名 count.v，按 OK。

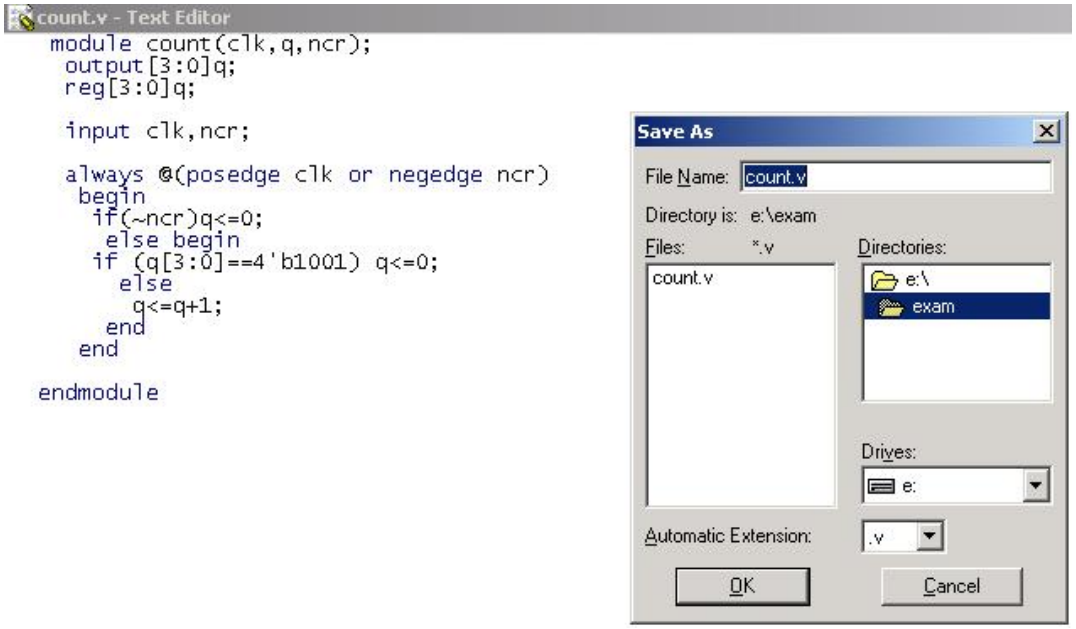


图 1.21

其它的编译、仿真、管脚分配和下载与原理图设计法相同。仿真波形如图 1.22。

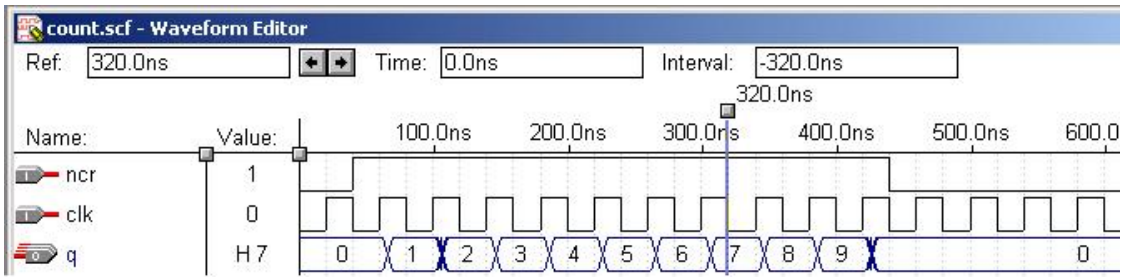


图 1.22

Verilog HDL 语言描述的模块 (如 count.v) ，可以生成一个符号，放在用户库中，供其它原理图输入文件调用，调用的方法与从器件库中取元件的方法相同。原理图和 Verilog HDL 语言描述的文本文件都可生成功能模块，方法是：点击 File/create default symbol 菜单即可。

第二章　现代 EDA 技术综合实验系统 PH-IV 型

现代 EDA 技术综合实验系统（以下简称实验系统）是依据目前我国工科“数字电子技术基础”、“CPLD/FPGA 与 ASIC 原理与应用”、“单片机原理与应用”、“电子技术基础课程设计”教学大纲的要求，以及在总结我校多年的实验教学和大规模可编程逻辑器件的实验教学的基础上，并结合目前电子电路发展的最新技术、最新设计思想和最先进的教学方法和教学内容，采用全开放式设计的方法，由华中科技大学（原华中理工大学）电信系国家工科电工电子教学基地研制开发的。

一、 EDA 技术综合实验系统的主要结构和特点 ：

1．EDA 技术综合实验系统主要由以下几部分组成：

1） 综合实验系统主板。

2） 高性能，大功率输出直流电源。

3） 下载实验电路板。

4） 通用下载电缆。

5） 高性能接插件。

实验系统主板的示意图如图 2.1 所示，主要包括以下几个部分：

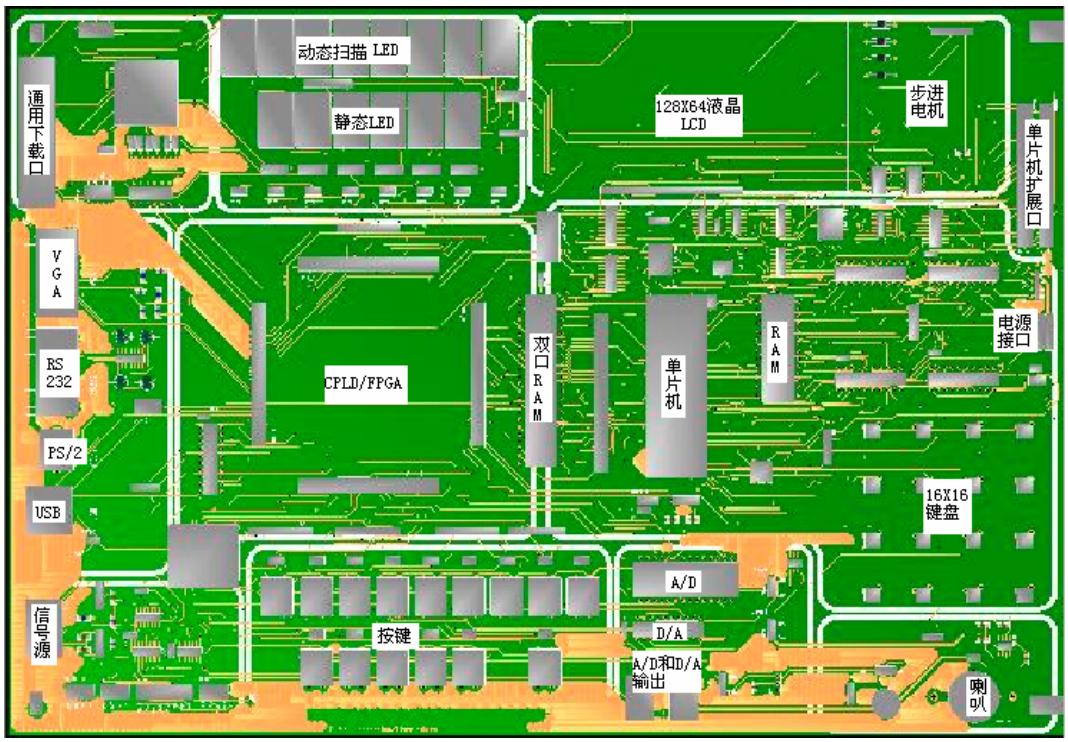


图 2. 1　实验系统主板的示意图

1） 直流电源 +5V （3A）；+12V （0.5A）；－12V（0.5A）。

2） 平台上集成了万能下载编程器，它除提供给板上的 CPLD/FPGA 和 isp 单片机 下载外，还有对外接口，可编程用户自己设计的可编程逻辑器件。

3） A/D，D/A 转换实验的输入/输出接口（含 A/D，D/A 芯片）。

4） RS232 接口；

5） VGA 接口；

6） PS/2 接口；

7） 信源工作频率：50MHz-2Hz；

8） 9 个琴键按键，5 个电平按键，1 个脉冲按键。

9） 8 个七段码共阴极显示器（扫描方式）。7 个七段码共阴极显示器（译码方式）。

10）8 个发光二极管；

11）单片机及其接口；

12）USB 接口

14）128X64 LCD 液晶显示

二．主要特点：

1） 通过大量的教学实践证明，该实验系统完全可以电子技术课程的四层次实验教学的需要，即基础实验、设计性实验、综合设计性实验和研究性实验。同时，利用该综合实验系统也可进行电子技术基础课程设计和学生进行二课活动，电子大赛和毕业设计。

2） 该实验系统将可编程器件（LATTICE/VANTIS、XILINX 和 ALTERA 等公司生产的 CPLD/FPGA 可编程逻辑器件）的下载实验板模块化,能完成这些公司的器件的设计和实验教学。同时将 isp 单片机的下载编程电缆和 CPLD/FPGA 的下载编程电缆结合成通用下载电缆，无需仿真器就可完成单片机的调试实验。

3）本实验教学系统由于集大规模可编程逻辑器件和 isp 单片机于一体，可以完成大规模可编程逻辑器件的实验教学和单片机与 CPLD/FPGA 的综合应用的实验教学； 该实验系统采用模块化，全开放的方式设计，设计灵活，可测试性，可扩充性的设计模式，使学生能利用该平台进行第二次开发，做出难度较大的综合和研究性实验和学生进行毕业设计。

三．实验系统的基本功能和资源配置

1） 实验系统主板提供的基本功能说明：

15 个按键：即**九个琴键按键**（按住琴键，对应输出指示的红色二极管亮，表示输出高电平，松开琴键，对应输出指示的红色二极管灭，表示输出为低电平）。**五个电平按键**（按下键，对应输出指示的红色二极管亮，表示输出高电平，再按下键，对应输出指示的红色二极管灭，表示输出为低电平）。**一个脉冲模式按键**： 当按下此按键，对应输出 20ms 的脉冲电平。各按键均已用软件消抖。

8 个共阴数码管，其中 7 个作为数码显示用(显示采用扫描和自动灭零技术，当输入的四位二进制数大于 1001 时，灭灯)，通过跳线选择供 CPLD/FPGA 或单片机使用。

6 个共阴数码管，其中 7 个作为数码显示用(自动灭零技术，当输入的四位二进制数大于 1001 时，灭灯)，直接与 CPLD/FPGA 连接。用户直接提供 6 个数码管的 4 位 BCD 码输入，共 24 位。

8 个发光二极管（输入高电平时，二极管发亮）。

4 组时钟输入（时钟频率从 50MHz 到 2Hz）即 CLK1、CLK2、CLK3 、CLK4。其中任何一组只能用一个跳线帽接通时钟信号，绝对不能在一组上插两个或两个以上的跳线帽接通两个以上的时钟。

一个蜂鸣器和一个喇叭（由跳线 K1 接通）；

一个串行通信接口，通过跳线选择供 CPLD/FPGA 或单片机使用。

电源输入电压：交流 220V±10V。

电源输出电压：＋5 V（2A）和±１２V(0.5A)。＋5 V 的地和±１２V 地。

2）跳线说明

jump1:

7	5	3	1
8	6	4	2

(1,2)对 XILINX 的 CPLD/FPGA 进行配置。
(3,4)对 LATTICE 的 CPLD 进行配置。
(5,6)对 ALTERA 的 CPLD/FPGA 进行配置。
(7,8)对 89S5X 进行配置。

Jump2:

2	4	6	8
1	3	5	7

(2，4)（5，7）单片机串口与 RS232 的端口相连。
(1，3)（6，8）CPLD/FPGA 串行信号与 RS232 的端口相连。

(1，2)（7，8）单片机与 CPLD/FPGA 的串口对连。

Jump3:

1	2

(1,2)提供 USB 外设的电源

Jump4:

1
2
3

(1,2) 选通 AD0809
（2，3）停用 0809

Jump5:

1
2
3

(1,2) 选通 AD0832
（2，3）停用 0832

MCU JP1:

全接左：由 CPLD/FPGA 控制扫描数码管显示
全接右：由单片机控制扫描数码管显示

MCU JP2:

按下：单片机下载状态
弹上：单片机运行状态

MCU JP3:

全接左：使用 IIC 端口连接
全接右：使用扫描键盘

MCU JP4:

全接左：由 CPLD/FPGA 控制步进电机驱动信号
全接右：由单片机控制步进电机驱动信号

下面介绍 ALTERA 公司的 EP1K30 芯片的下载实验板的引脚分配， 如表 1：

F1 下载实验板 1K30 的引脚号与实验系统的连接关系

功能	脚号	功能	脚号	功能	脚号	功能	脚号
Clk1	55	A1(低)	118	SL(A/D)	95	双口 RAM	
Clk2	54	B1	121	RD(A/D)	92		
Clk3	59	C1	120	EN(A/D)	91	CE	113
CLK4	56	D1(高)	128	D7(A/D)	82	WR	112
蜂鸣器	64	A2	122	D6(A/D)	81	RD	111
喇叭	65	B2	131	D5(A/D)	86	D7	96

琴键 1	44	C2	130	D4(A/D)	83	D6	98
琴键 2	46	D2	133	D3(A/D)	88	D5	97
琴键 3	47	A3	132	D2(A/D)	87	D4	100
琴键 4	48	B3	136	D1(A/D)	90	D3	99
琴键 5	49	C3	135	D0(A/D)	89	D2	102
琴键 6	51	D3	138	WR(D/A)	67	D1	101
琴键 7	60	A4	137	D7(D/A)	80	D0	110
琴键 8	62	B4	141	D6(D/A)	79	A10	109
琴键 9	63	C4	140	D5(D/A)	78	VGA	
电平 1	37	D4	143	D4(D/A)	73	RED1	19
电平 2	38	A5	142	D3(D/A)	72	RED0	20
电平 3	39	B5	7	D2(D/A)	70	GRN1	22
电平 4	41	C5	144	D1(D/A)	69	GRN0	21
电平 5	42	D5	9	D0(D/A)	68	BLU1	26
脉冲	43	A6	8	RxD	29	BLU0	23
LED8	18	B6	11	TxD	30	VG13	28
LED7	14	C6	10	PS1	32	VG14	27
LED6	17	D6	13	PS5	31		
LED5	12	LED2	117	T1			
LED4	119	LED1	114				
LED3	116						

F1 下载实验板 1K100 的引脚号与实验系统的连接关系

功能	脚号	功能	脚号	功能	脚号	功能	脚号
Clk1	75	A1(低)	172	SL(A/D)	121	双口 RAM	
Clk2	78	B1	173	RD(A/D)	119		
Clk3	79	C1	174	EN(A/D)	120	CE	159
CLK4	80	D1(高)	175	D7(A/D)	103	WR	160
蜂鸣器	86	A2	176	D6(A/D)	104	RD	158
喇叭	89	B2	177	D5(A/D)	111	Busy	157
琴键 1	60	C2	179	D4(A/D)	112	D7	126
琴键 2	61	D2	180	D3(A/D)	113	D6	127
琴键 3	63	A3	186	D2(A/D)	114	D5	128
琴键 4	64	B3	187	D1(A/D)	115	D4	131
琴键 5	65	C3	189	D0(A/D)	116	D3	132
琴键 6	67	D3	190	WR(D/A)	93	D2	133
琴键 7	68	A4	191	D7(D/A)	101	D1	134
琴键 8	69	B4	192	D6(D/A)	99	D0	135

琴键 9	70	C4	193	D5(D/A)	100	A10	136
电平 1	53	D4	196	D4(D/A)	96	A9	139
电平 2	54	A5	195	D3(D/A)	97	A8	140
电平 3	55	B5	198	D2(D/A)	94	A7	141
电平 4	56	C5	197	D1(D/A)	95	A6	142
电平 5	57	D5	200	D0(D/A)	92	A5	143
脉冲	58	A6	199	RxD	36	A4	144
LED8	208	B6	203	TxD	31	A3	147
LED7	206	C6	202	T1	122	A2	148
LED6	207	D6	205	INT1	125	A1	149
LED5	204	VGA		扫描		A0	150
LED4	170	RED1	17	DX	11	通用口	
LED3	169	RED0	18	CX	8	SPIO1	44
LED2	168	GRN1	24	BX	9	SPIO2	45
LED1	167	GRN0	25	AX	10	SPIO3	46
PS2		BLU1	26	Y3	14	SPIO4	47
PS1	37	BLU0	27	Y2	13		
PS5	38	VG13	28	Y1	12		
		VG14	29	小数点	15		

四．本实验系统及附件

1． 综合实验系统主平台（内含大功率高性能专用电源一个） 1 台
2. ALTERA 下载实验板（含 EP1K30 芯片 1 片或 EP1K30 芯片 1 片） 1 块
3. 通用下载电缆 1 根
4.《CPLD/FPGA 实验指导书 》 1 本
6. 电源线 1 根
7.其它附件

基础性实验

实验一 组合逻辑 3 - 8 译码器的设计

（MaxplusII 软件的基本操作与应用）

一、实验目的：

1、掌握组合逻辑电路的设计方法。
2、掌握组合逻辑电路的静态测试方法。
3、初步掌握 Max+PlusII 软件的基本操作与应用。
4、初步了解可编程器件的设计全过程。

二、实验步骤：

（一）设计输入：

1 、软件的启动：单击“开始”进入“程序”选中“Max+PlusII 10.2”，打开“”MaxplusII 软件，如图 4.1-1 所示。

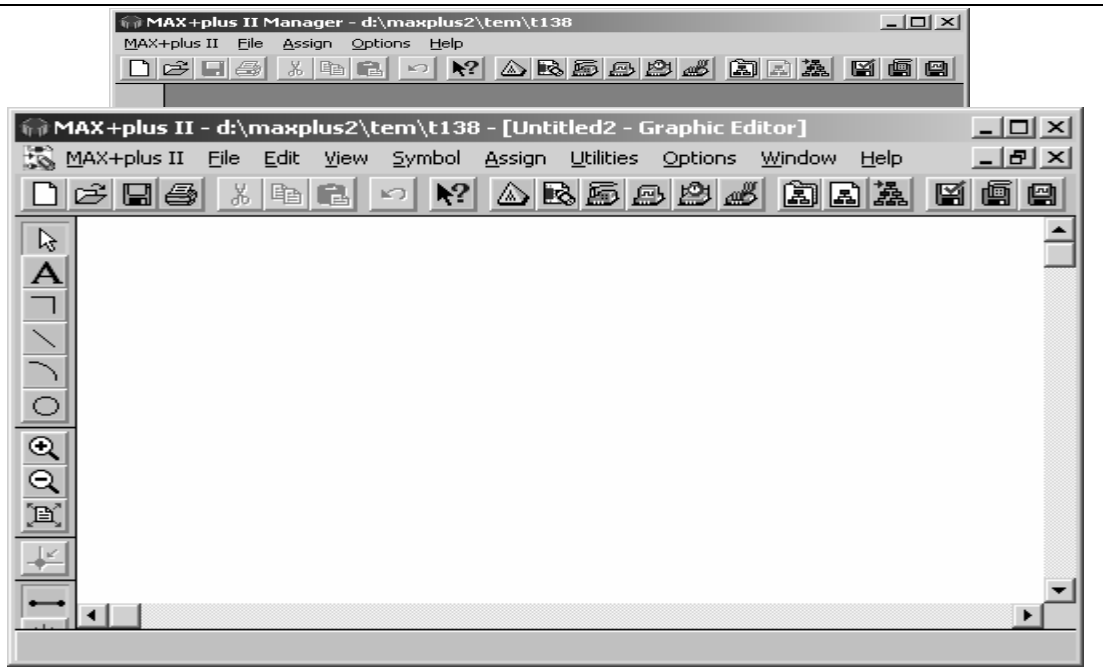


图 4.1-3

2、启动 File\New 菜单，弹出设计输入选择窗口，如图 4.1-2 所示：

3、选择 Graphic Editor File，单击 OK，打开原理图编辑器，进入原理图设计输入电路编辑状态，如图 4.1-3 所示：

4、设计输入

1）放置一个器件在原理图上

a、在原理图的空白处双击鼠标右键，出现图 4.1-4：

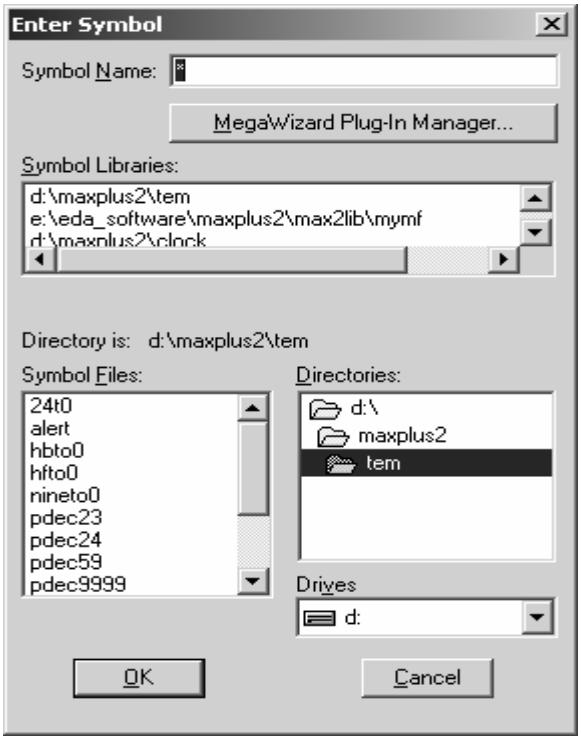


图 4.1-4

b、在光标处输入元件名称（如：input，output，and2，and3，nand2，or2，not，xor，dff 等）或用鼠标点击库元件，按下 OK 即可。

c、如果安放相同的元件，只要按住 Ctrl 键，同时用鼠标按左键拖动该元件复制即可。

d、一个完整的电路包括：输入端口 input、电路元件集合、输出端口 output。

e、图 4.1-5 为 3 - 8 译码器元件安放结果。

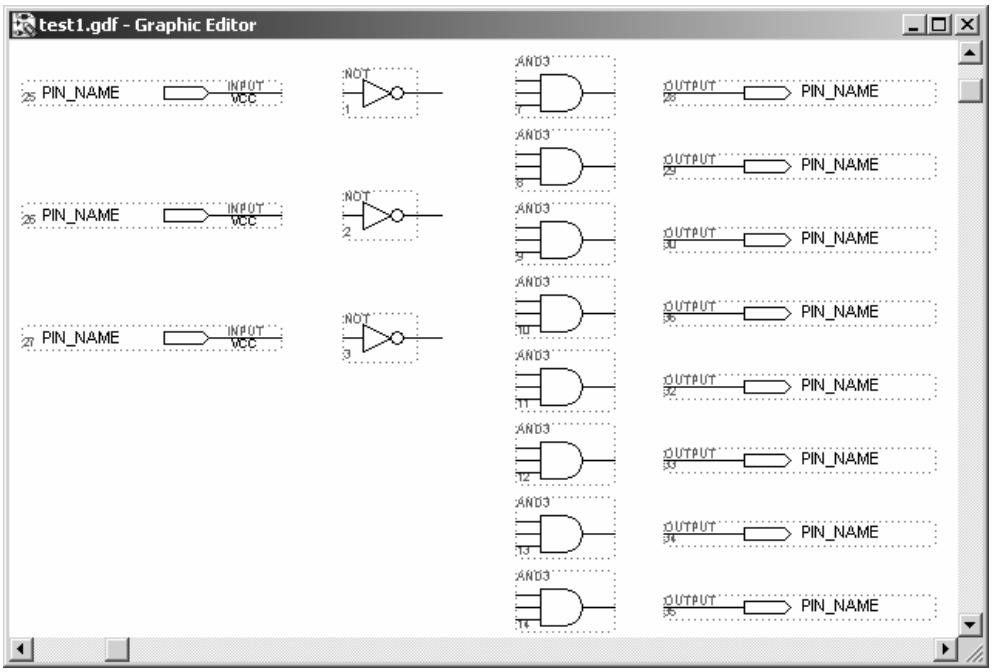


图 4.1-5

2 ）添加连线到器件的引脚上：把鼠标移到元件引脚附近，则鼠标自动由箭头变为十字，按住鼠标左键拖动，即可画出连线。 3 - 8 译码器原理图连线后如图 4.1-6 所示。

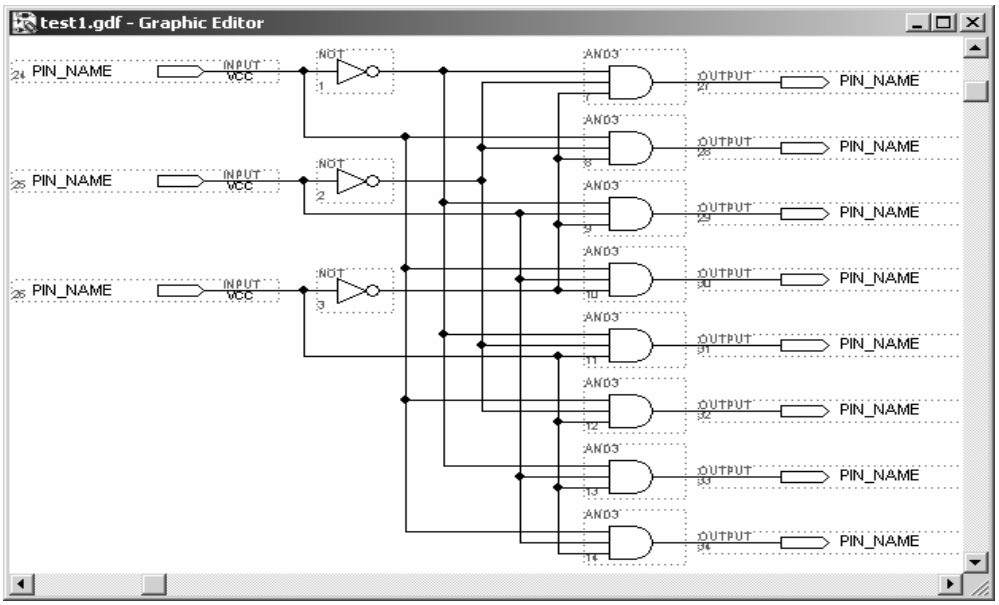


图 4.1-6

3）标记输入/输出端口属性

分别双击输入端口的“PINNAME”，当变成黑色时，即可输入标记符并回车确认；输出端口标记方法类似。本译码器的三输入端分别标记为：A、B、C；其八输出端分别为：Q0、Q1、Q2、Q3、Q4、Q5、Q6、Q7。如图 4.1-7 所示。

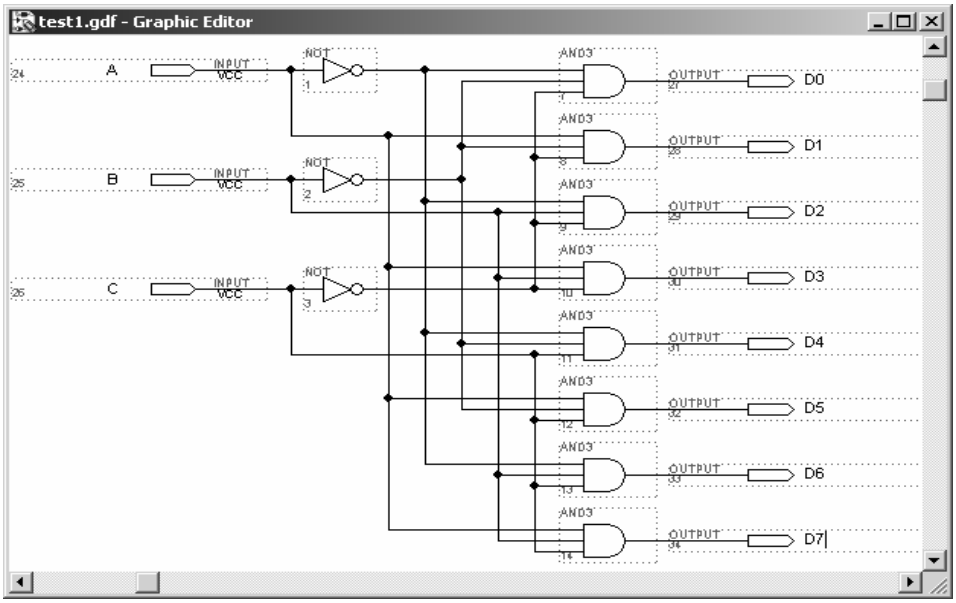


图 4.1-7

4）保存原理图

单击保存按钮图表，对于新建文件，出现类似文件管理器图框，请选择保存路径/文件名称保存原理图，原理图的扩展名为.gdf，本实验中取名为 test1.gdf。（**注意：新建项目，一定要建立一个专门的文件夹保存项目文件，在编译过程中将有大量新文件产生。**）

5）点击 File\Project\Set project to current File 设置此项目为当前项目文件，如图 4.1-8 所示。**注意此操作在你打开几个原有项目文件时尤为重要，否则编译时容易出错** 。

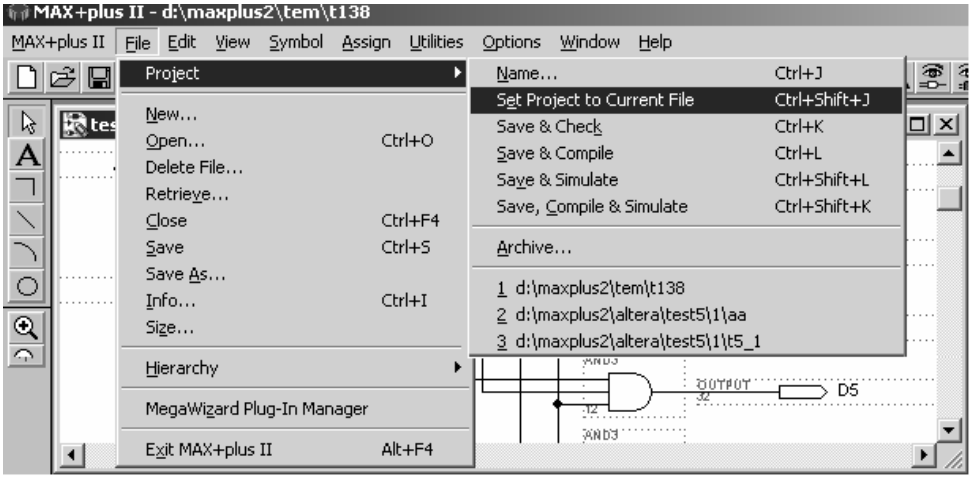


图 4.1-8

至此，你已完成了一个电路的原理图的设计输入过程。

（二）电路的编译与适配

1、选择芯片型号

选择当前项目文件欲设计实现的实际芯片进行编译适配，单击 Assign|Device 菜单选择芯片，如图 4.2-1 所示。

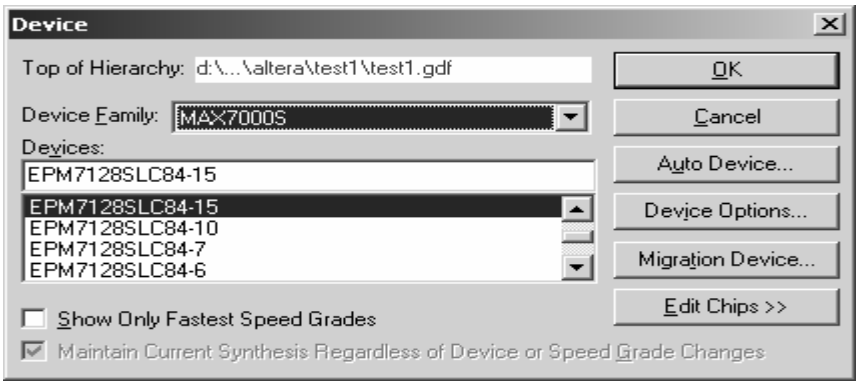


图 4.2-1

如果此时不选择适配芯片的话，该软件将自动把所有适合本电路的芯片一一进行编译适配，这将费你许多时间。该例程中我们选用 FPGA 芯片来实现，如用 ACEX1K 系列的 EP1K30TC144-3 芯片；同样也可以用其他 CPLD/FPGA 芯片来实现，你只需在下面的对话框中指出具体芯片型号即可。注意如果将该列表下方标有“Show only Fastest Speed Grades”选项的“√”消去，以便显示出所有速度级别的器件。完成选择后单击“OK”按钮。

2、编译适配

启动 MaxplusII\Compiler 菜单，按 Start 开始编译，并显示编译结果，生成下载文件。如果编译时选择的芯片是 CPLD，则生成*.pof 文件；如果是 FPGA 芯片，则生成*.sof 文件，以被硬件下载编程时调用。同时生成*.rpt 报告文件，可详细察看编译结果。如果有错误待修改后再进行编译适配，如图 4.2-2 所示。注意此时在主菜单栏里的 Processing 菜单下有许多编译时的选项，视实际情况选择设置。

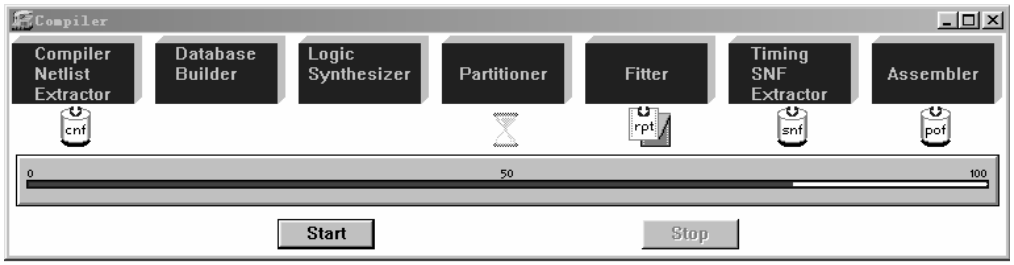


图 4.2-2

如果你设计的电路顺利地通过了编译，在电路不复杂的情况下，就可以对芯片进行编程下载，测试硬件。如果你的电路有足够复杂，那么其仿真就显得非常必要。

（三）电路仿真与时序分析

MaxplusII 教学版软件支持电路的功能仿真（或称前仿真）和时序分析（或称后仿真）。众所周知，开发人员在进行电路设计时，非常希望有比较先进的高效的仿真工具出现，这将为你的设计过程节约很多时间和成本。由于 EDA 工具的出现，和它所提供的强大的（在线）仿真功能迅速地得到了电子工程设计人员的青睐，这也是当今 EDA(CPLD/FPGA)技术非常火

爆的原因之一。下面就 MaxpluII 软件仿真功能的基本应用在本实验中作一初步介绍，在以后的实验例程中将不再一一介绍。

一）添加仿真激励波形

1、启动 MaxplusII\Wavefrom Editor 菜单，进入波形编辑窗口，如图 4.3-1 所示。

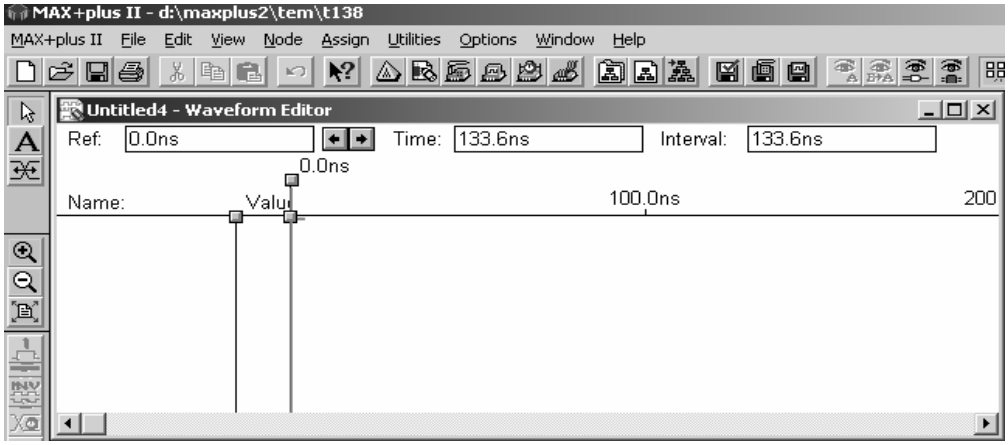


图 4.3-1

2 、将鼠标移至空白处并单击右键，出现如图 4.3-2 所示对话窗口。

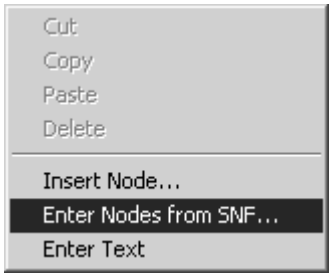


图 4.3-2

3 、选择 Enter Nodes from SNF 选项，并按左键确认，出现 4.3-3 所示对话框，单击“List”和“=>”按钮，选择欲仿真的 I/O 管脚。

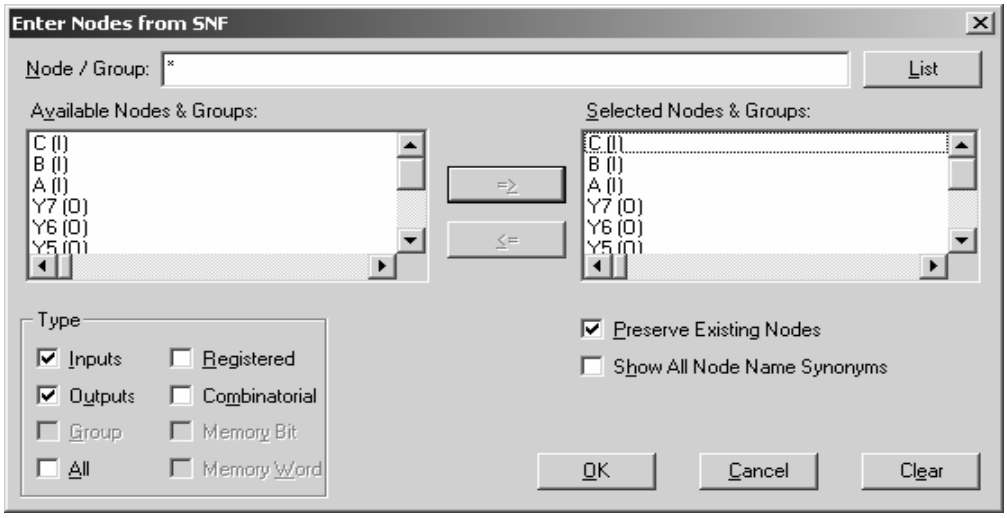


图 4.3-3

4 、单击 OK 按钮，列出仿真电路的输入、输出管脚图，如图 4.3-4 所示。在本电路中，3-8 译码器的输出为网格，表示未仿真前输出是未知的。

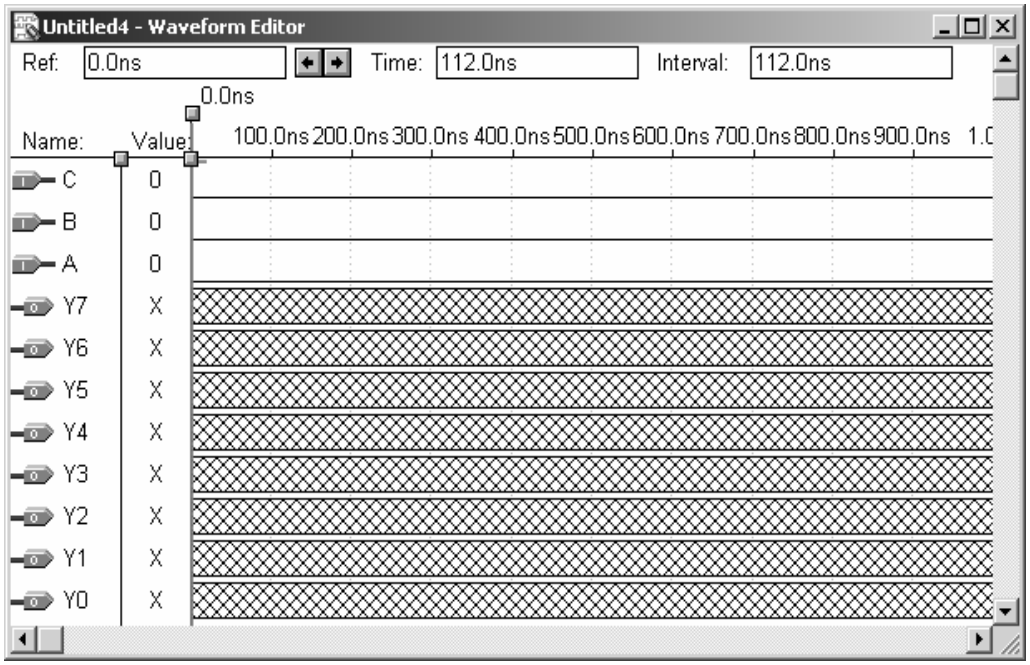


图 4.3-4

5 、调整管脚顺序，符合常规习惯，调整时只需选中某一管脚（如 A）并按住鼠标左键拖到相应的位置即可完成。调整后如图 4. 3 - 5 所示。

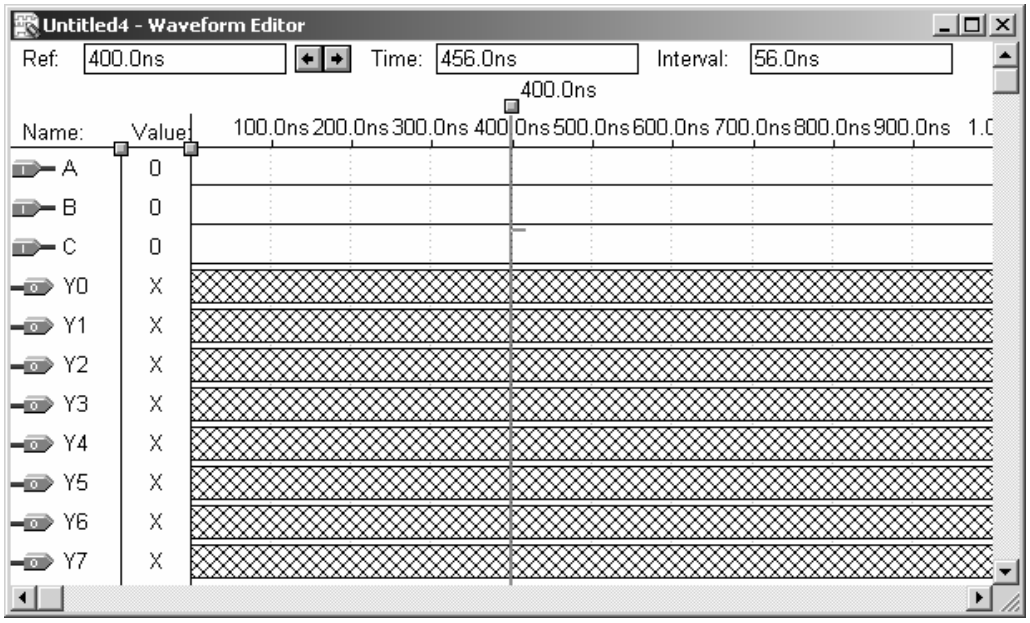


图 4.3-5

6 、准备为电路输入端添加激励波形。选中欲添加信号的管脚，窗口左边的信号源即可

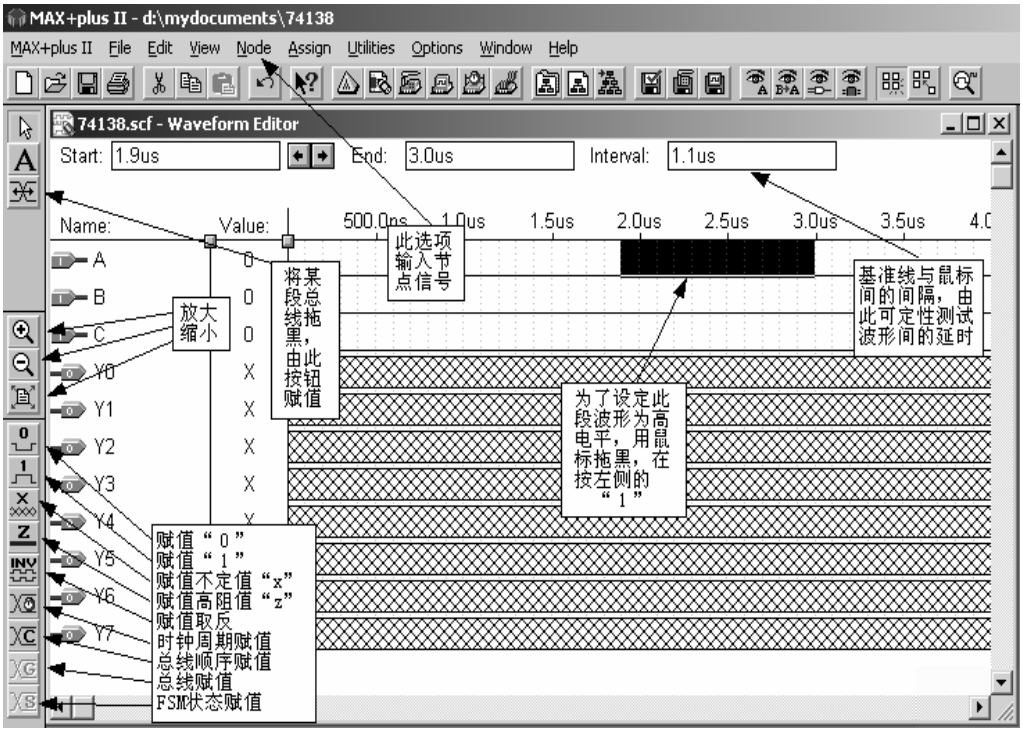


图 4.3-6

变成可操作状态，如图 4. 3 - 6 中箭头和圆括号所示。根据实际要求选择信号源种类，在本电路中选择时钟信号就可以满足仿真要求。

7 、选择仿真时间：视电路实际要求确定仿真时间长短，如图 4. 3 - 7 所示。本实验中，我们选择软件的默认时间 1us 就能观察到 3-8 译码器的 8 个输出状态。

8 、为 A、B、C 三输入端添加信号：先选中 A 输入端“  ”，然后再点击窗口左侧的时钟信号源图标“  ”添加激励波形，出现图 4.3-8 所示的对话窗口。

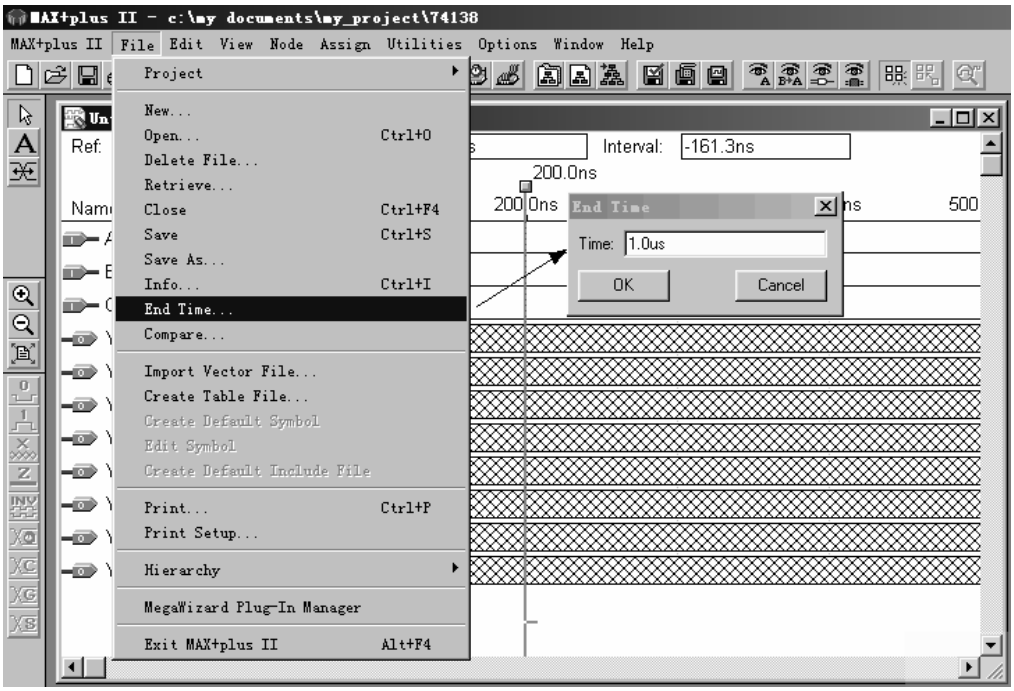


图 4.3-7

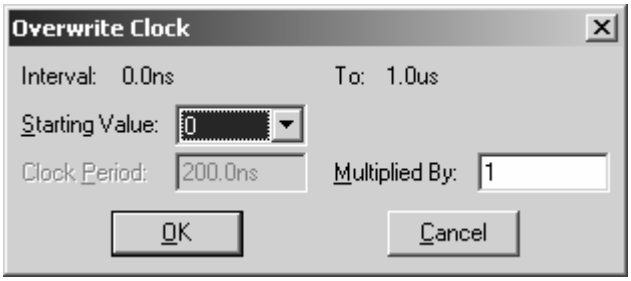


图 4.3-8

9 、在本例程中，我们选择初始电平为“0”，时钟周期倍数为“1”（时钟周期倍数只能为 1 的整数倍）并按 OK 确认。经上述操作我们已为 A 输入端添加完激励信号，点击全屏显示如图 4. 3 - 9 所示。

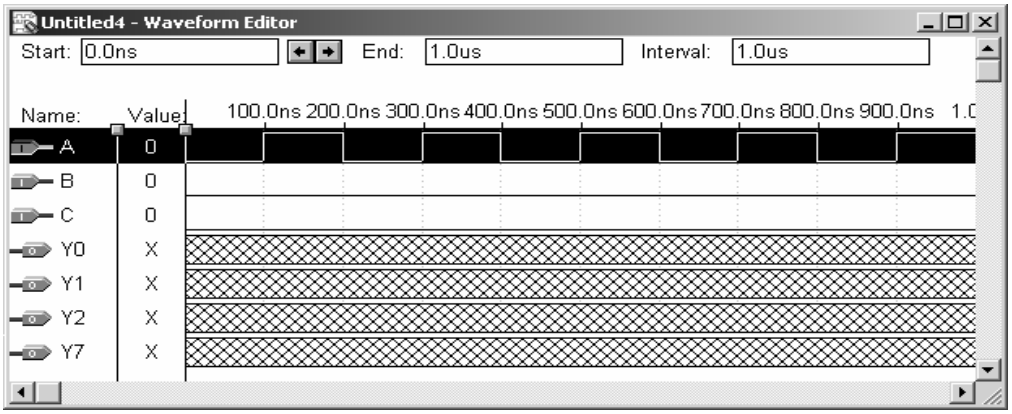


图 4.3-9

10、根据电路要求编辑另外两路输入端激励信号波形，在本实验中，3-8 译码器的 A、B、C 三路信号的频率分别为 1、2、4 倍关系，其译码输出顺序就符合我们的观察习惯。按上述方法为 B、C 两路端口添加波形后单击左边全屏显示图表“”，三路激励信号的编辑结果为图 4.3-10 所示。

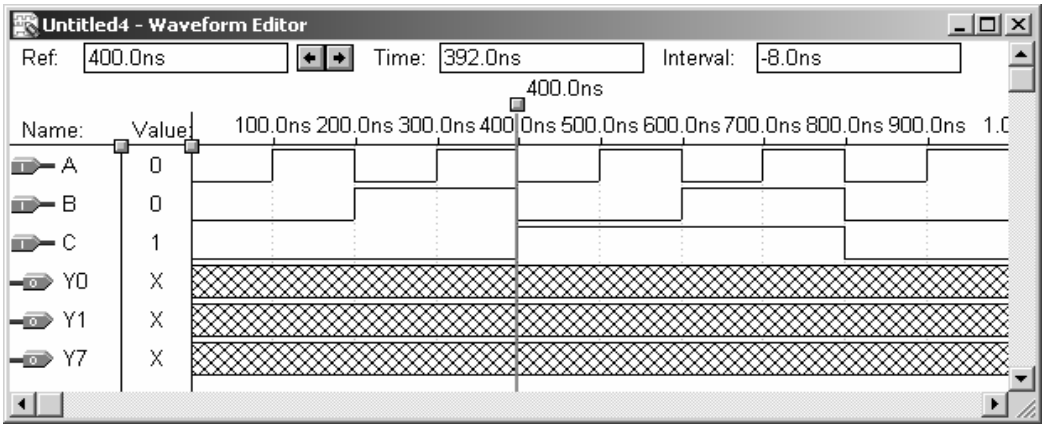


图 4.3-10

11、 保存激励信号编辑结果：使用 File | Save ，或关闭当前波形编辑窗口时均出现图 4. 3 -11 会话框，注意此时文件名不要随意改动，单击 OK 按钮保存激励信号波形。

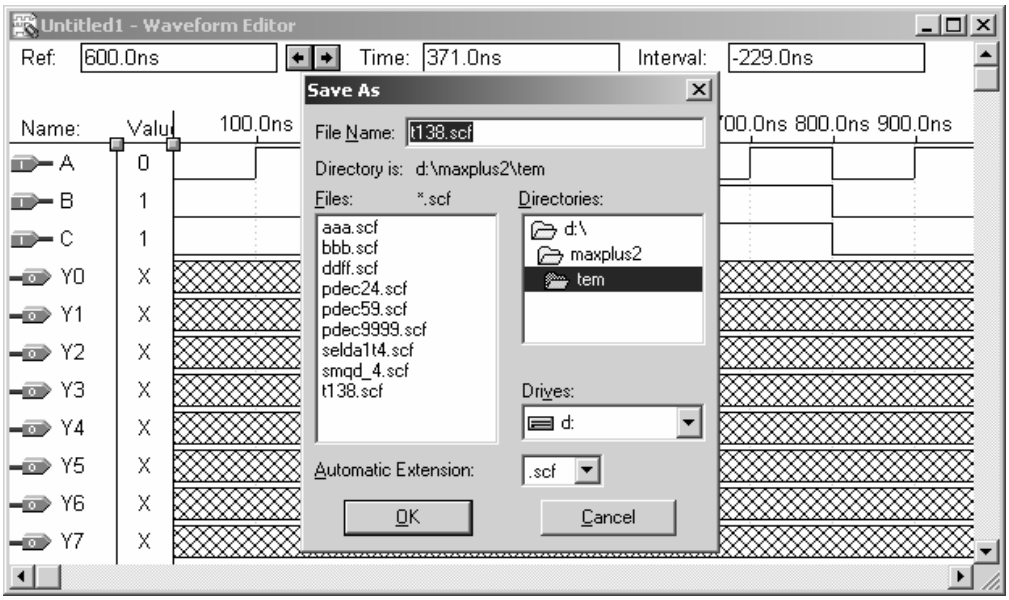


图 4.3-11

二）电路仿真

电路仿真有前仿真（功能仿真）和后仿真（时序仿真）两种，时序仿真覆盖了功能仿真，在该例程中我们直接使用时序仿真。读者可以自行使用功能仿真，对比其区别。

1 、选择 Maxplus2|Simulator 菜单，弹出其对话窗口，如图 4.3-12 所示。

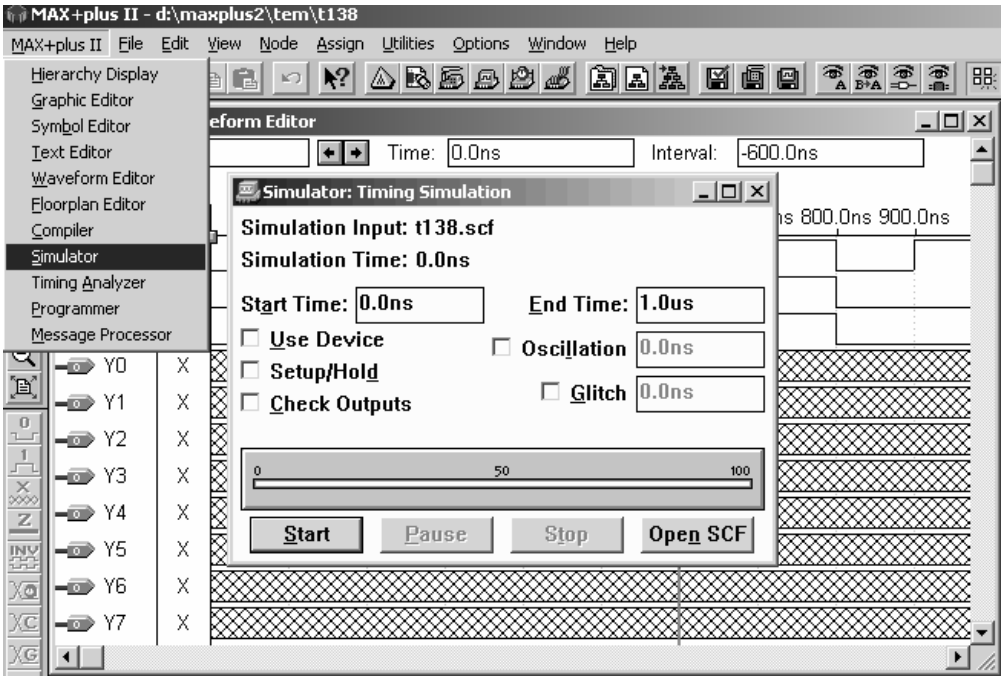


图 4.3-12

2、确定仿真时间，End Time 为“1”的整数倍。注意：如果在添加激励信号完成后设置

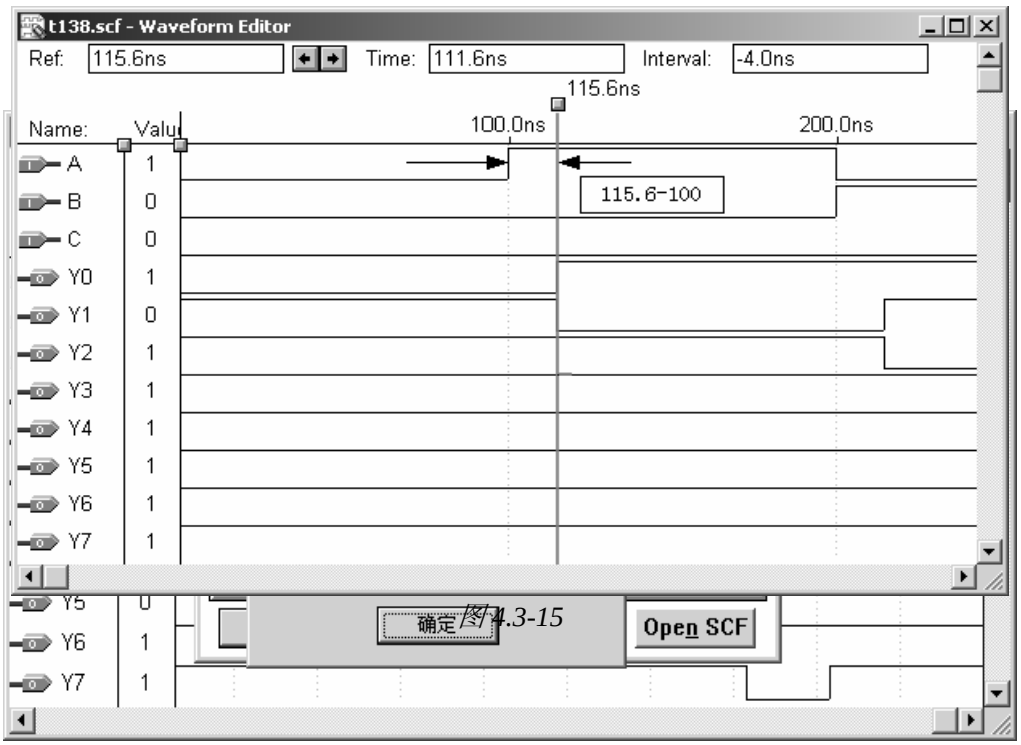


图 4.3-15

图 4.3-13

结束时间的话，此时仿真窗口中就不能修改 End Time 参数了。在该例程中，我们使用的是默认时间，单击 Start 开始仿真，如有出错报告，请查找原因，一般是激励信号添加有误。本电路仿真结果报告中无错误、无警告，如下图 4.3-13 所示。

3、观察电路仿真结果，请单击“确定”后单击激励输出波形文件“Open SCF”图标。如图 4. 3 -14 所示。

4、从上图可见，我们所设计的 3-8 译码器顺利地通过了仿真,设计完全正确。点击“”将上图放大，仔细观察一下电路的时序，在空白处单击鼠标的右键，出现测量标尺，然后将标尺拖至欲测量的地方，查看延时情况，如图 4.3-15 所示。

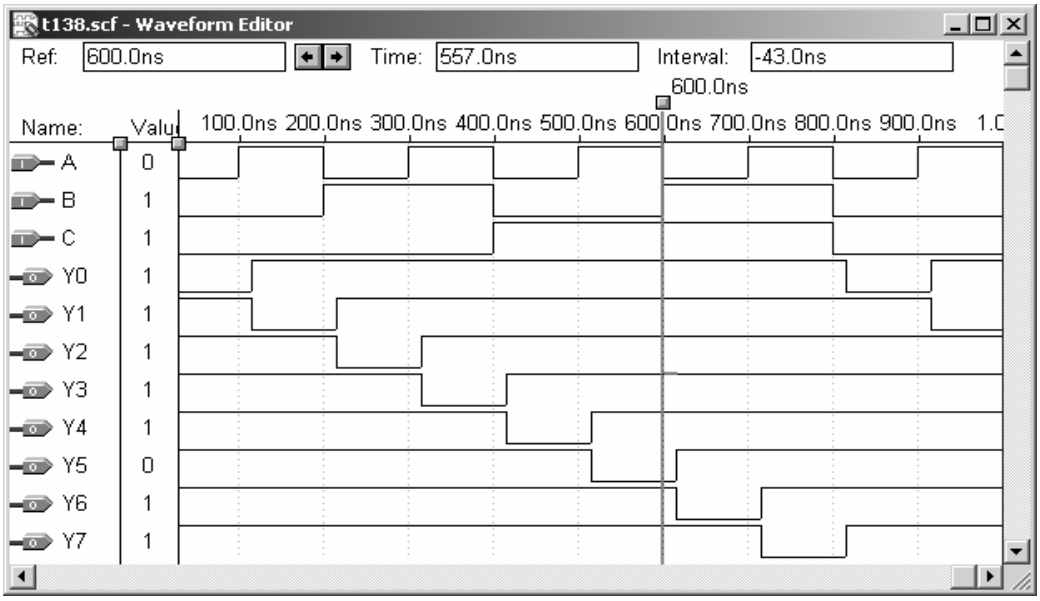


图 4.3-14

从上图可以看到，我们这个电路在实际工作时，激励输出有 15.6 个 ns 的延迟时间。至此，你已完成和掌握了软件的仿真功能。

（四）管脚的重新分配与定位：

启动 MaxplusII\Floorplan Editor 菜单命令，（或按“”快捷图标）出现图 4.4-1 所示的芯片管脚自动分配画面，点击“”图标，所有管脚将会在“ Unassigned Nodes & Pins: ”中显示。

读者可在芯片的空白处试着双击鼠标左键，你会发现这样的操作可在芯片和芯片内部之

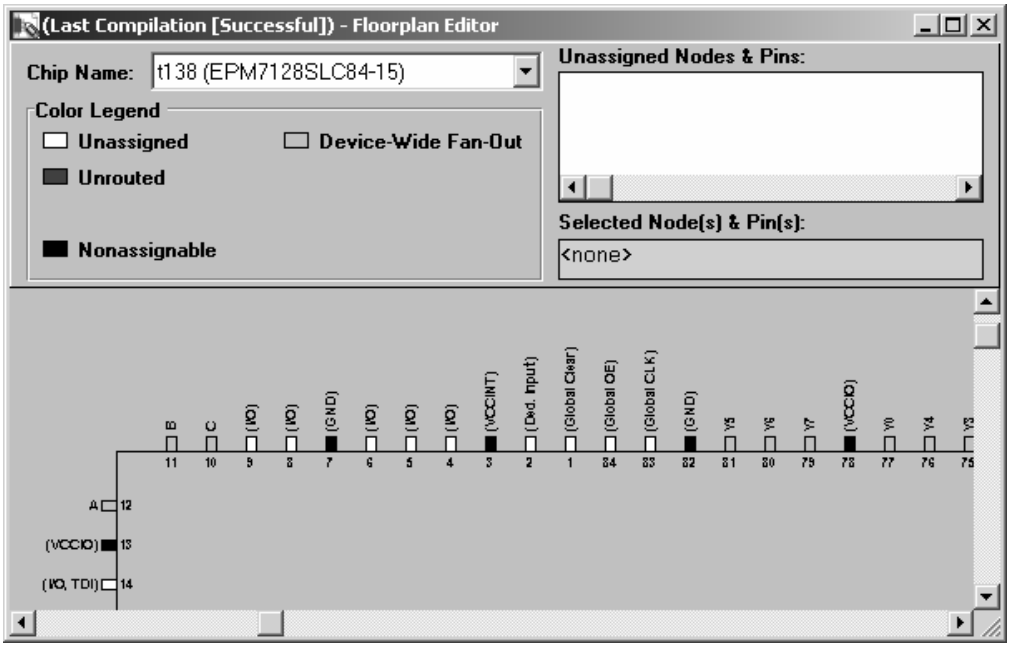


图 4.4-1

间进行切换，可观察到芯片内部的逻辑块等。

Foolrplan Editor 展示的是该设计项目的管脚分配图。这是由软件自动分配的。用户可随意改变管脚分配，以方便与你的外设电路进行匹配。管脚编辑过程如下：

1 、按下窗口左边手动分配图标“”，所有管脚将会出现在窗口中，如图 4.4 - 2 中箭头所指。

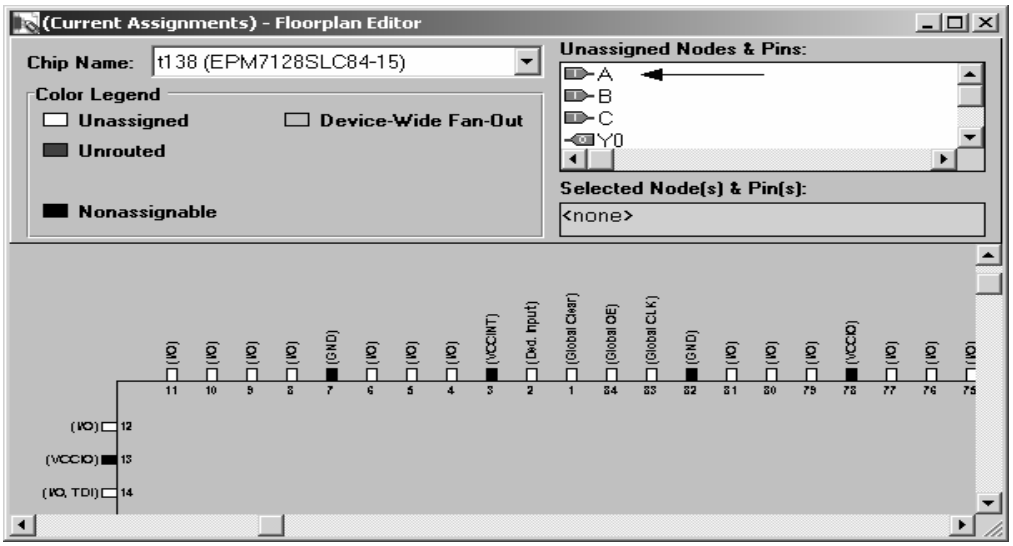


图 4.4-2

2、用鼠标按住某输入|输出端口，并拖到下面芯片的某一管脚上，松开鼠标左键，便可完成一个管脚的重新分配（读者可以试着在管脚之间相互拖曳，你会觉得非常方便）。注意：芯片上有一些特定的管脚不能被占用，进行管脚编辑时一定要注意。另外，在芯片器件选择

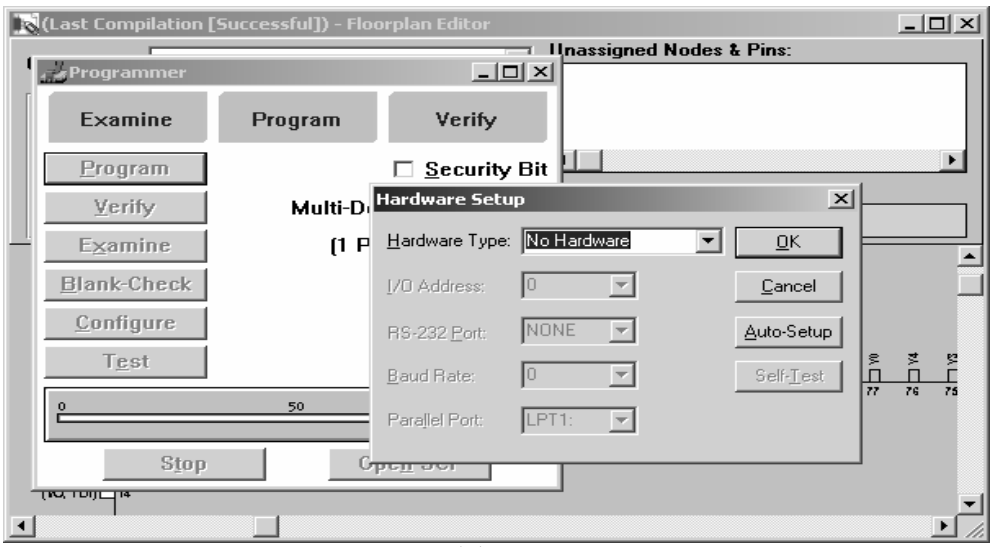


图 4.5-1

中，如果选的时 Auto，则不允许对管脚进行再分配。当你对管脚进行二次调整以后，一定要再编译一次，否则程序下载以后，其管脚功能还是当初的自动分配状态。

（五）、器件下载编程与硬件实现

一）实验电路板上的连线

用三位拨码开关代表译码器的输入端 A、B、C，将之分别与 EP1K30TC144-3 芯片的相应管脚相连；用 LED 灯来表示译码器的输出，将 Q0...Q7 对应的管脚分别与 8 只 LED 灯相连。试验结果如下：

A	B	C	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
0	0	0	亮	灭	灭	灭	灭	灭	灭	灭
1	0	0	灭	亮	灭	灭	灭	灭	灭	灭
0	1	0	灭	灭	亮	灭	灭	灭	灭	灭
1	1	0	灭	灭	灭	亮	灭	灭	灭	灭
0	0	1	灭	灭	灭	灭	亮	灭	灭	灭
1	0	1	灭	灭	灭	灭	灭	亮	灭	灭
0	1	1	灭	灭	灭	灭	灭	灭	亮	灭
1	1	1	灭	灭	灭	灭	灭	灭	灭	亮

二）器件编程下载

1、启动 MaxplusII\Programmer 菜单，如果是第一次启用的话，将出现如图 4. 5 - 1 所示的对话框，请你填写硬件类型，请选择“ByteBlaster(MV)”并按下 OK 确认即可。

2、启用 JTAG\Multi-Device JTAG Chain Setup...菜单项，按 Select Programming File...按钮，选择要下载的*.pof 文件。然后按 Add 加到文件列表中，如图 4. 5 - 2 所示（如果编译时选择的是 FPGA 芯片，此时要选择的下载文件为*.sof）如果不是当前要下载编程的文件的话，请使用 Delete 将其删除。

3、选择完下载文件后，单击 OK 确定，出现下图 4.5- 3 所示的下载编程界面。

4、单击 Pogram 按钮，进行下载编程，如不能正确下载，请点击图 4. 5 - 2 的 Detect JTAG chain info 按钮进行 JTAG 测试，查找原因，直至完成下载，最后按 OK 退出。至此，你已经完成了可编程器件的从设计到下载实现的整个过程。

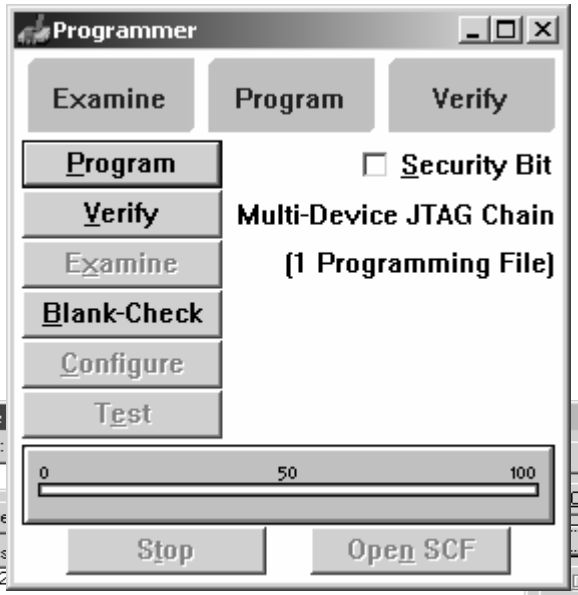


图 4.5-3

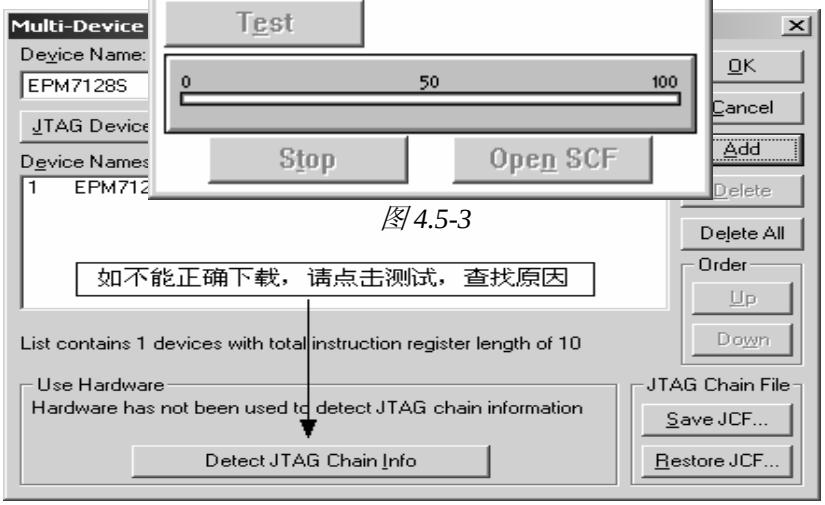


图 4.5-2

5、结合电路功能，观察设计实现的正确结果。

说明：通过对本实验的学习，相信读者对 MaxplusII 软件已经有了一定的认识，同样对 CPLD/FPGA 可编程器件的整个设计过程也有了一个完整的概念和思路。

管脚定义说明：输入： A --------电平 1 （37）

B--------电平 2 （38）

C ------电平 3 （39）

输出： Q0 -----LED1 （114）

Q1 -----LED2 （117）

Q2 -----LED3 （116）
Q3 -----LED4 （119）
Q4 -----LED5 （12）
Q5 -----LED6 （17）
Q6 -----LED7 （14）
Q07-----LED8 （18）

实验二 计数器模块设计

一、实验目的

1、了解时序电路的经典设计方法（D 触发器、JK 触发器和一般逻辑门组成的时序逻辑电路）。

2、了解同步计数器，异步计数器的使用方法。

3、了解同步计数器通过清零阻塞法和预显数法得到循环任意进制计数器的方法。

4、理解时序电路和同步计数器加译码电路的联系，设计任意编码计数器。

5、了解同步芯片和异步芯片的区别。

二、硬件要求

主芯片 Altera EP1K30TC144-3，时钟，四位八段数码管

三、实验参考程序

```
//模块：counter10
//功能：一位十进制双向计数，异步复位，异步置数，带进位信号

module counter10(clk,rst,enb,down_up,load,pre_data,dataout,carry);
    input clk,rst,enb,down_up,load;                    //enb 信号低有效？
    input [3:0] pre_data;
    output [3:0] dataout;
    output carry;
    reg [3:0] dataout;
    reg carry;
```

```
always@(posedge clk or posedge rst or posedge load)
begin
if(rst)                                                    //采用异步复位
      begin
      dataout<=4'b0000;
      carry<=0;
      end
else if(load)
      dataout<=pre_data;
else if(!enb)
      if(!down_up)
            begin
            if(dataout==9)
                  begin
                  dataout <= 0;
                  carry <= 1;
                  end
            else
                  begin
                  dataout <= dataout + 1;
                  carry <= 0;
                  end
            end
      else if(down_up)
            begin
            if(dataout==0)
                  begin
                  dataout <= 9;
                  carry <= 1;
                  end
            else
                  begin
                  dataout <= dataout - 1;
                  carry <= 0;
                  end
            end
      else
            begin
            dataout<=4'b0000;
```

```
                carry<=0;
                end
        end
endmodule

//两位十进制计数器。练习使用例化语句。
//不保证加、减计数切换时输出的前后一致
/*-------------------------------------------------------------------------------------------------------------
                        输入输出说明
电平 1：计数使能   (37)                          电平 2：打开预置数状态  (38)
电平 3：加、减计数选择   (39)
琴键 1：复位     (44)                            琴键 2：增加预置数值     (46)
琴键 3：改变预置位  (47)                         琴键 4：装载预置数值     (48)
LED1：  进位信号     (114)                       LED2：输入信号          （117）
LED3    溢出信号   （116）
CLK3：  clkset 2Hz          (59)                 CLK4:  clock   4Hz       (56)
数码管 1：data1 (118,121,120,128)                数码管 2：data2 (122,131,130,133)
数码管 3：data3 (132,136,135,138)                数码管 4：predata1 (137,141,140,143)
数码管 5：predata2（142，7，144，9）              数码管 6：predata3（8，11，10，13）
-------------------------------------------------------------------------------------------------------------*/
`include  "counter10.v"
`include  "set_data.v"
module
dec_counter(clock,G_reset,enable,down_up,load,pre_inc,pre_select,pre_set,clkset,over,set,carry,
                        predata2,predata1,   data2,data1);
//                      predata3,predata2,predata1,   data3,data2,data1);
    input clock,G_reset,enable,down_up,load,pre_inc,pre_select,pre_set,clkset;
    output [3:0] predata2,predata1,data2,data1;
    reg      [3:0] predata2,predata1,data2,data1;
//  output [3:0] predata3,predata2,predata1,data3,data2,data1;
//  reg      [3:0] predata3,predata2,predata1,data3,data2,data1;
    output over,set,carry;
    reg      over;
    wire     set,carry,carry1;          /* 加 carry2 信号 */
    assign set=pre_set;
    assign carry=carry1;
    set_data   SD(clkset,G_reset,load,pre_inc,pre_select,predata3,predata2,predata1);
    counter10 C1(clock,G_reset,enable,down_up,pre_set,predata1,data1,carry1);
    counter10 C2(carry1,G_reset,enable,down_up,pre_set,predata2,data2          );
```

```
//    counter10 C3(carry2,G_reset,enable,down_up,pre_set,predata3,data3);
      //溢出判断
      always@(posedge clock)
      if(down_up)
            if( data2==0 && data1==0 ) over<=1;          /* 加 data3 信号 */
            else over<=0;
      else
            if( data2==9 && data1==9 ) over<=1;          /* 加 data3 信号 */
            else over<=0;
endmodule
```

综合性实验

实验三 红绿交通灯控制设计

一条主干道和支干道汇合成十字路口，每个入口处设置背靠背一对信号灯：红、绿、黄、左拐各四盏(黄灯可省略)。要求任意两条行驶线之间不能出现交叉，同时由于所有交通工具都靠前进方向的右边行驶。故交通灯的控制要实现：

1、 主干道、支干道的通行时间完全区分开；

2、 两条干道上的直通行驶时间和左拐时间也要完全区分开。

实验要求

信号灯交换次序为：主干道、支干道交替允许通行。假设各状态允许时间如下表：

主干道	支干道	通行时间
绿灯亮，允许通行	红灯亮，禁止通行	40s
黄灯亮	红灯亮，禁止通行	5s
左拐灯亮，可左行	红灯亮，禁止通行	15s
黄灯亮	红灯亮，禁止通行	5s
红灯亮，禁止通行	绿灯亮，允许通行	30s
红灯亮，禁止通行	黄灯亮	5s
红灯亮，禁止通行	左拐灯亮，可左行	15s
红灯亮，禁止通行	黄灯亮	5s

要求依次设定定时电路，完成四种信号灯的交通灯控制器设计。

参考程序如下：

```
/*------------------------------------------------ 输出定义:
   1.主干道红、黄、绿、左拐信号  lampa1 lampa2 lampa3 lampa4
   2.支干道红、黄、绿、左拐信号  lampb1 lampb2 lampb3 lampb4
   3.主干道计时显示              acounth acountl
   4.支干道计时显示              bcounth bcountl
    设计思路:
   1.用于倒数计时的 BCD 码计数器  numa     numb
   2.某一段计数结束时的状态判断   tempa    tempb
   3.8 个输出状态的循环           counta countb
------------------------------------------------------------*/
module traffic(clk,enb,acounth,acountl,bcounth,bcountl,
                lampa1,lampa2,lampa3,lampa4,
                lampb1,lampb2,lampb3,lampb4);
  input clk,enb;
  output [3:0] acounth,acountl,bcounth,bcountl;
  output lampa1,lampa2,lampa3,lampa4,lampb1,lampb2,lampb3,lampb4;
  reg tempa,tempb;
  reg [2:0] counta,countb;
  reg [7:0] numa,numb;
  reg [7:0] ared,ayellow,agreen,aleft,bred,byellow,bgreen,bleft;
  reg lampa1,lampa2,lampa3,lampa4,lampb1,lampb2,lampb3,lampb4;
  assign {acounth,acountl}=numa;
  assign {bcounth,bcountl}=numb;
  /*预置倒数计时器的计时时间*/
  always@(enb)
  if(!enb)
      begin
      ared    <=8'b01010101;        // 55 秒＝byellow+bleft+bgreen
      ayellow<=8'b00000101;         //   5 秒
      agreen <=8'b01000000;         // 40 秒
      aleft   <=8'b00010101;        // 15 秒
      bred    <=8'b01100101;        // 65 秒＝ayellow+aleft+agreen
      byellow<=8'b00000101;         //   5 秒
      bleft   <=8'b00010101;        // 15 秒
      bgreen <=8'b00110000;         // 30 秒
      end
```

```
/*主干道的时序处理*/
always@(posedge clk)
begin
if(enb)
    begin
    if(tempa)
        //倒数计时过程，计时寄存器 numa 通过 wire 直接输出
        begin
        if(numa>0)
            if(numa[3:0]==0)
                begin
                numa[3:0]<=9;
                numa[7:4]<=numa[7:4]-1;
                end
            else
                numa[3:0]<=numa[3:0]-1;
        if(numa==0)
            tempa<=0;
        end
    else
        //每段计时结束时，置 tempa 低电位，进入状态循环判定，并同时获得新的 numa
        begin
        tempa<=1;
        case(counta)
            0:                                                  begin numa<=agreen;{lampa1,lampa2,lampa3,lampa4}<=2;counta<=1;end
            1:                                                  begin numa<=ayellow;{lampa1,lampa2,lampa3,lampa4}<=4;counta<=2;end
            2:                                                  begin numa<=aleft;{lampa1,lampa2,lampa3,lampa4}<=1;counta<=3;end
            3:                                                  begin numa<=ayellow;{lampa1,lampa2,lampa3,lampa4}<=4;counta<=4;end
            4:                                                  begin numa<=ared;{lampa1,lampa2,lampa3,lampa4}<=8;counta<=0;end
        default: {lampa1,lampa2,lampa3,lampa4}<=8;
        endcase
        end
    end
```

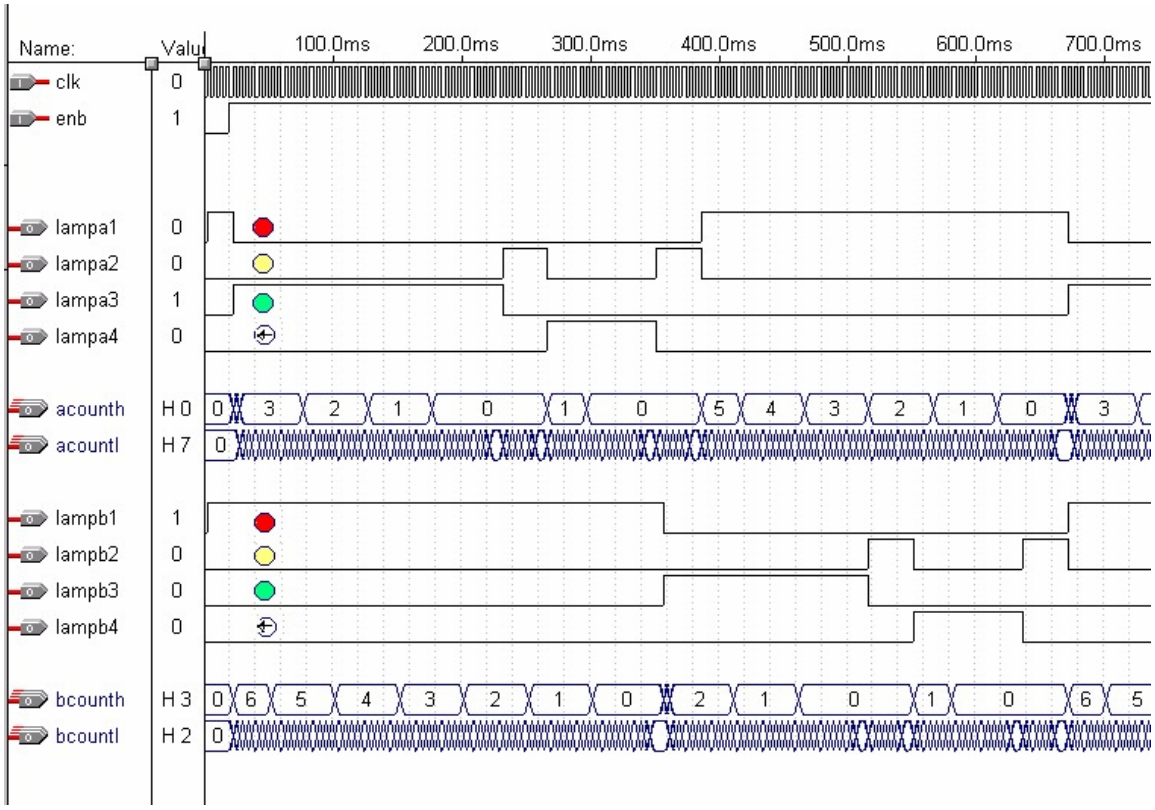
```
    else                    //无使能信号时的状态置位
        begin
        {lampa1,lampa2,lampa3,lampa4}<=4'b1000;
        counta<=0;
        tempa<=0;
        end
    end

    /*支干道的时序处理*/
    always@(posedge clk)
    begin
    if(enb)
        begin
        if(tempb)
            //倒数计时过程，计时寄存器 numa 通过 wire 直接输出
            begin
            if(numb>0)
                if(numb[3:0]==0)
                    begin
                    numb[3:0]<=9;
                    numb[7:4]<=numb[7:4]-1;
                    end
                else
                    numb[3:0]<=numb[3:0]-1;
            if(numb==0)
                tempb<=0;
            end
        else
            //每段计时结束时，置 tempb 低电位，进入状态循环判定，并同时获得新
的 numb                    begin
            tempb<=1;
            case(countb)
                0:                                                  begin
numb<=bred;{lampb1,lampb2,lampb3,lampb4}<=8;countb<=1;end
                1:                                                  begin
numb<=bgreen;{lampb1,lampb2,lampb3,lampb4}<=2;countb<=2;end
                2:                                                  begin
numb<=byellow;{lampb1,lampb2,lampb3,lampb4}<=4;countb<=3;end
                3:                                                  begin
```

```
numb<=bleft;{lampb1,lampb2,lampb3,lampb4}<=1;countb<=4;end
                4:                                                    begin
numb<=byellow;{lampb1,lampb2,lampb3,lampb4}<=4;countb<=0;end
            default:{lampb1,lampb2,lampb3,lampb4}<=8;
            endcase
            end
        end
    else                    //无使能信号时的状态置位
        begin
        {lampb1,lampb2,lampb3,lampb4}<=4'b1000;
        countb<=0;
        tempb<=0;
        end
    end
endmodule
```

【仿真结果】

注: 在使能信号为 0 时，默认得交通灯状态为全部红灯。



管脚定义说明：

acounth：数码管 2 (122,131,130,133)　　acountl ： 数码管 1（118，121，120，128）

bcounth: 数码管 6 （8，11，10，13）　　bcountl: 数码管 5（142，7，144，9）

clk: CLK3 (59)　　enb: 电平 1 （37）

lampa1:LED4 (119)　　lampa2: LED2 (117)

lampa3:LED3 (116)　　lampa4: LED1 (114)

lampb1:LED5 (12)　　lampb2:LED7 (14)

lampb3:LED6 (17)　　lampb4:LED8 (18)

实验四 频率计设计

```
/*-----------------------------------------------------
频率计设计的主要参考程序如下，实际应用还要配合模块设计。
CLK2: 1KHz                CLK3: 被测频率
Ena:  检测读数开关
-----------------------------------------------------*/
module cnt10(clkin,clr,ena,cq,carry_out);
        input clkin,clr,ena;
        output [3:0] cq;
        output carry_out;
        reg [3:0] cq;
        reg carry_out;
        always@(posedge clr or posedge clkin)
        begin
          if(clr)
            begin
            cq<=0;
            carry_out<=0;
            end
        else
             if(ena==1)
               if(cq==9)
                 begin
                   cq<=0;
                   carry_out<=1;
                 end
               else
                   begin
```

```
                cq<=cq+1;
                carry_out<=0;
               end
        end
  endmodule

module reg32b(load,din,clr,dout);
        input load,clr;
        input   [31:0] din;
        output [23:0] dout;

        reg      [23:0] dout;
        always@(posedge clr or posedge load)
          begin
            if(clr)
              dout<=0;
            else
              dout<=din[23:0];
           end
 endmodule
            module testctl(clr,clk,clr_cnt,tsten,load);
         input clr,clk;
         output tsten,load,clr_cnt;
         reg      tsten,load,clr_cnt;

         reg [9:0] count;

         always@(posedge clk or posedge clr)
          begin
          if(clr)
              begin
              count<=0;
              tsten<=1;
              clr_cnt<=1;
              end
          else if (count==1023)
                  begin
                  load<=1;
                  tsten<=0;
```

```
                clr_cnt<=1;
                end
        else
                begin
                 count<=count+1;
                 load<=0;
                 tsten<=1;
                 clr_cnt<=0;
                 end
        end
endmodule
```

管脚定义说明：

clkin: CLK2 (54)　　clr: CLK3

ena：琴键 1 （44）

数码管 1：cq0 (118,121,120,128)　　数码管 2：cq1 (122,131,130,133)

数码管 3：cq3 (132,136,135,138)　　数码管 4： (137,141,140,143)

数码管 5： （142，7，144，9）　　数码管 6：（8，11，10，13）

实验五 乒乓球游戏设计

两人乒乓游戏机是用 8～16 个发光二极管代表乒乓球台，用点亮的发光二极管按一定的方向移动来表示球的运动。在游戏机两侧各设置一个击发键，在甲方发球局，当甲拨动开关时，靠近甲的第一盏 LED 亮，然后各 LED 由甲向乙依次点亮，代表乒乓球在运动。当”球”行至靠近乙的第一盏 LED 时，若乙准确击球，则球”反弹”，若失误则甲的记分牌自动加一分。重新发球，比赛继续。每赛四球发球局轮换一次。按下清零键，则双方记分牌清零，开始新局。

实验要求：

按照满 11 分为一局进行比赛，用 Verilog 语言设计乒乓球游戏机。能够实现裁判与记分、击球发音、重新开局。

```
/*信号定义列表
        in1 in2：甲、乙双方击球信号
          o1~o8：乒乓球轨迹模拟显示信号
    right1 right2：甲、乙方发球权拥有信号
    score1 score2：甲、乙方得分信号
*/
```

```
module
pingpong(clk,cp1k,rst,ready,beepon,in1,in2,o1,o2,o3,o4,o5,o6,o7,o8,pickball,
    right1,right2,score1l,score1h,score2l,score2h,beep);
input clk,cp1k,rst,ready,beepon,in1,in2;
output o1,o2,o3,o4,o5,o6,o7,o8,right1,right2,pickball,beep;
output [3:0] score1l,score1h,score2l,score2h;
reg o1,o2,o3,o4,o5,o6,o7,o8,right1,right2,pickball;
reg o0,o9;
reg [3:0] score1l,score1h,score2l,score2h;
reg ldir,rdir,temp;
reg [2:0] count;
reg [1:0] act1,act2;
reg in_act1,in_act2;
wire beep;
/*挥拍时发出蜂鸣*/
assign beep=(in_act1||in_act2)&&cp1k&&beepon;
/*游戏过程的主时序*/
always@(posedge clk)
begin
/*按下挥拍键之后的动作处理：把电平散列为脉冲*/
if(in1)
    if(act1<2)
        act1=act1+1;
    if(act1==2)
        act1=0;
if(act1>0 && in1)
    in_act1=1;
else
    in_act1=0;

if(in2)
    if(act2<2)
        act2=act2+1;
    if(act2==2)
        act2=0;
if(act2>0 && in2)
    in_act2=1;
else
```

```
        in_act2=0;

/*复位及初始化*/
if(rst==1)
        begin
        {score1h,score1l}=0;
        {score2h,score2l}=0;
        count=0;
        end
if({o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}==10'b0000000000)
        begin
        right1=1;
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0100000000;
        end

/*发球、拍球的过程;在这里球才能改变运行方向*/
if(o1&&in_act1)
        begin
        {o1,o2,o3,o4,o5,o6,o7,o8}=8'b01000000;
        rdir=1;
        ldir=0;
        end
else if(o8&&in_act2)
        begin
        {o1,o2,o3,o4,o5,o6,o7,o8}=8'b00000010;
        rdir=0;
        ldir=1;
        end

/*乒乓球的运行*/
else if(rdir && !o1 && !o9)
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}={o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}>>1;
else if(ldir && !o8 && !o0)
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}={o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}<<1;

/*一球结束的处理*/
else if(o9||o0)
        begin
        if(!pickball)
```

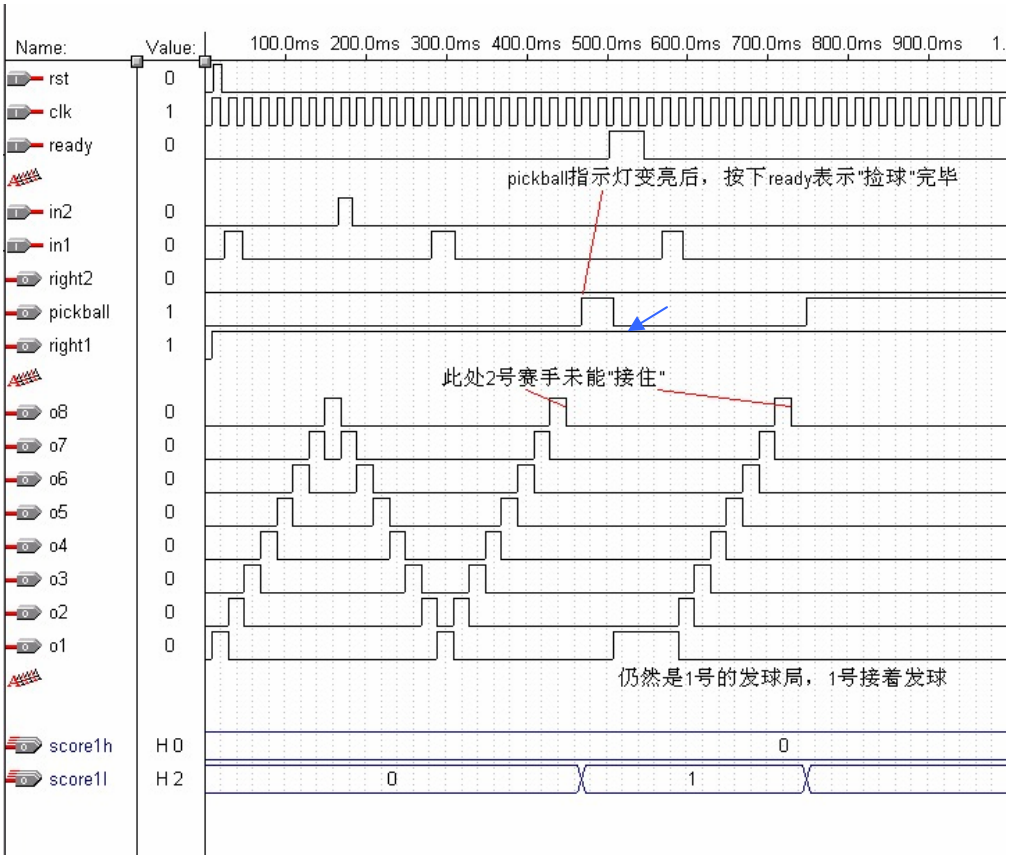
```
            begin
            rdir=0;
            ldir=0;

            if(count<4)             //判断是否发球局交替
            count=count+1;
            else if(count==4)
                begin
                count=0;
                temp=right2;
                right2=right1;
                right1=temp;
                end

if(o9)                  //判断得分的一方并更新记分版
    if(score1l[0] && score1l[3])            //1 号得分个位是否满 9
                begin
                score1l=0;
                score1h=1;
                end
            else if(score1h[0] && score1l[0])       //判断 1 号得分＝11？
                begin
                {score1h,score1l}=0;
                {score2h,score2l}=0;
                count=0;
                end
            else score1l=score1l+1;
else if(o0)
     if(score2l[0] && score2l[3])               //2 号得分个位是否满 9
                begin
                score2l=0;
                score2h=1;
                end
     else if(score2h[0] && score2l[0])          //判断 2 号得分＝11？
                begin
                {score1h,score1l}=0;
                {score2h,score2l}=0;
                count=0;
                end
```

```
                else score2l=score2l+1;
            end
        pickball=1;                    //捡球标志
        if(ready)
            pickball=0;
        if(!pickball)                  //把球置于发射位置
            if(right1 && !right2)
                {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0100000000;
            else if(right2 && !right1)
                {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0000000010;
        end
    end
    endmodule
```

【仿真结果】



/*信号定义列表

in1 in2：甲、乙双方击球信号

```
            o1~o8 : 乒乓球轨迹模拟显示信号
    right1 right2 : 甲、乙方发球权拥有信号
    score1 score2 : 甲、乙方得分信号
*/

module pingpong(clk,cp1k,rst,ready,beepon,in1,in2,o1,o2,o3,o4,o5,o6,o7,o8,pickball,
        right1,right2,score1l,score1h,score2l,score2h,beep);
    input clk,cp1k,rst,ready,beepon,in1,in2;
    output o1,o2,o3,o4,o5,o6,o7,o8,right1,right2,pickball,beep;
    output [3:0] score1l,score1h,score2l,score2h;
    reg o1,o2,o3,o4,o5,o6,o7,o8,right1,right2,pickball;
    reg o0,o9;
    reg [3:0] score1l,score1h,score2l,score2h;
    reg ldir,rdir,temp;
    reg [2:0] count;
    reg [1:0] act1,act2;
    reg in_act1,in_act2;
    wire beep;

    /*挥拍时发出蜂鸣*/
    assign beep=(in_act1||in_act2)&&cp1k&&beepon;

    /*游戏过程的主时序*/
    always@(posedge clk)
    begin

    //按下挥拍键之后的动作处理：把电平散列为脉冲
    if(in1)
        if(act1<2)
            act1=act1+1;
        if(act1==2)
            act1=0;
    if(act1>0 && in1)
        in_act1=1;
    else
        in_act1=0;

    if(in2)
        if(act2<2)
```

```
            act2=act2+1;
        if(act2==2)
            act2=0;
if(act2>0 && in2)
        in_act2=1;
else
        in_act2=0;

//复位及初始化
if(rst==1)
        begin
        {score1h,score1l}=0;
        {score2h,score2l}=0;
        count=0;
        end
if({o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}==10'b0000000000)
        begin
        right1=1;
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0100000000;
        end

//发球、拍球的过程;在这里球才能改变运行方向
if(o1&&in_act1)
        begin
        {o1,o2,o3,o4,o5,o6,o7,o8}=8'b01000000;
        rdir=1;
        ldir=0;
        end
else if(o8&&in_act2)
        begin
        {o1,o2,o3,o4,o5,o6,o7,o8}=8'b00000010;
        rdir=0;
        ldir=1;
        end

//乒乓球的运行
else if(rdir && !o1 && !o9)
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}={o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}>>1;
else if(ldir && !o8 && !o0)
```

```
        {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}={o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}<<1;

    //一球结束的处理
    else if(o9||o0)
        begin
        if(!pickball)
            begin
            rdir=0;
            ldir=0;

            if(count<4)
            count=count+1;
            else if(count==4)
                begin
                count=0;
                temp=right2;
                right2=right1;
                right1=temp;
                end

            if(o9)
                if(score1l[0] && score1l[3])
                    begin
                    score1l=0;
                    score1h=1;
                    end
                else if(score1h[0] && score1l[0])
                    begin
                    {score1h,score1l}=0;
                    {score2h,score2l}=0;
                    count=0;
                    end
                else score1l=score1l+1;
            else if(o0)
                if(score2l[0] && score2l[3])
                    begin
                    score2l=0;
                    score2h=1;
                    end
                else if(score2h[0] && score2l[0])
                    begin
```

```
                    {score1h,score1l}=0;
                    {score2h,score2l}=0;
                    count=0;
                    end
                else score2l=score2l+1;
            end
        pickball=1;                 //捡球标志
        if(ready)
            pickball=0;
        if(!pickball)               //把球置于发射位置
            if(right1 && !right2)
                {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0100000000;
            else if(right2 && !right1)
                {o0,o1,o2,o3,o4,o5,o6,o7,o8,o9}=10'b0000000010;
        end

    end
endmodule
```

管脚定义：

```
    right2 :    OUTPUT_PIN = 46;
    right1 :    OUTPUT_PIN = 47;
    pla2y3 :    OUTPUT_PIN = 138;
    pla2y2 :    OUTPUT_PIN = 135;
    pla2y1 :    OUTPUT_PIN = 136;
    pla2y0 :    OUTPUT_PIN = 132;
    pla1y3 :    OUTPUT_PIN = 13;
    pla1y2 :    OUTPUT_PIN = 10;
    pla1y1 :    OUTPUT_PIN = 11;
    pla1y0 :    OUTPUT_PIN = 8;
    score2h3 :      OUTPUT_PIN = 133;
    score2h2 :      OUTPUT_PIN = 130;
    score2h1 :      OUTPUT_PIN = 131;
    score2h0 :      OUTPUT_PIN = 122;
    score2l3 :OUTPUT_PIN = 128;
    score2l2 :OUTPUT_PIN = 120;
    score2l1 :OUTPUT_PIN = 121;
    score2l0 :OUTPUT_PIN = 118;
    score1l3 :OUTPUT_PIN = 143;
    score1l2 :OUTPUT_PIN = 140;
    score1l1 :OUTPUT_PIN = 141;
    score1l0 :OUTPUT_PIN = 137;
```

score1h3 : OUTPUT_PIN = 9;
score1h2 : OUTPUT_PIN = 144;
score1h1 : OUTPUT_PIN = 7;
score1h0 : OUTPUT_PIN = 142;
pickball : OUTPUT_PIN = 49;
o8 : OUTPUT_PIN = 18;
o7 : OUTPUT_PIN = 14;
o6 : OUTPUT_PIN = 17;
o5 : OUTPUT_PIN = 12;
o4 : OUTPUT_PIN = 119;
o3 : OUTPUT_PIN = 116;
o2 : OUTPUT_PIN = 117;
o1 : OUTPUT_PIN = 114;
beep : OUTPUT_PIN = 43;
rst : INPUT_PIN = 37;
ready : INPUT_PIN = 42;
in2 : INPUT_PIN = 63;
in1 : INPUT_PIN = 44;
cp1k : INPUT_PIN = 54;
clk : INPUT_PIN = 59;
beepon : INPUT_PIN = 38

实验五 抢答器设计

//程序：snatch
//功能：3 人竞赛抢答记分器

/*----------

输入输出说明

参赛选手按键
琴键 1：1 号选手抢答键　　琴键 3：2 号选手抢答键
琴键 5：3 号选手抢答键

主持人按键
琴键 6：抢答开始　　琴键 7：回答正确
琴键 8：回答错误　　琴键 9：限定时间内未作答
电平 5：复位初始化

----------*/

```
module snatch(rst,start,add,sub,back,in1,in2,in3,o1,o2,o3,o4,o5,o6,o7,o8,sclk,flashcp,bees,
              point1h,point1l,point2h,point2l,point3h,point3l);
```

```
input    rst,start,add,sub,back,in1,in2,in3,sclk,flashcp;
output o1,o2,o3,o4,o5,o6,o7,o8,bees;
reg      o1,o2,o3,o4,o5,o6,o7,o8,bees;
output [3:0] point1h,point1l,point2h,point2l,point3h,point3l;
reg      [3:0] point1h,point1l,point2h,point2l,point3h,point3l;

reg      [3:0] data1h,data1l,data2h,data2l,data3h,data3l;
reg      [2:0] num;
reg      [2:0] win,out;

reg      [4:0] state;
reg      [11:0] delay1,delay2,delay3;

parameter      Idle      = 5'd0,
               Detect    = 5'd1,
               Get1      = 5'd2,
               Get2      = 5'd3,
               Get3      = 5'd4,
               Alter1    = 5'd5,
               Alter2    = 5'd6,
               Alter3    = 5'd7,
               Ctdown8 = 5'd8,
               Ctdown7 = 5'd9,
               Ctdown6 = 5'd10,
               Ctdown5 = 5'd11,
               Ctdown4 = 5'd12,
               Ctdown3 = 5'd13,
               Ctdown2 = 5'd14,
               Ctdown1 = 5'd15,
               Ctdown0 = 5'd16,
               Timeover= 5'd17,
               No_ans    = 5'd18,
               Right     = 5'd19,
               Wrong     = 5'd20;

always@(posedge sclk)
if(!rst)
      begin
      data1h<=3; data1l<=0; data2h<=3; data2l<=0; data3h<=3; data3l<=0;
```

```
        {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000000;
        bees<=0;
        num<=0; win<=0;
        state<=Idle;
        end
    else
        case(state)
            Idle        : begin
                          num<=3'b000;
                          {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000000;
                          bees<=0;
                          case(win)
                              3'b100 : begin
                                       if(flashcp)
                                           begin point1h<=4'h8;     point1l<=4'h8; end
                                       else
                                           begin point1h<=4'hF;     point1l<=4'hF; end

                                       point2h<=data2h; point2l<=data2l;
                                       point3h<=data3h; point3l<=data3l;
                                       end
                              3'b010 : begin
                                       if(flashcp)
                                           begin point2h<=4'h8;     point2l<=4'h8; end
                                       else
                                           begin point2h<=4'hF;     point1l<=4'hF; end

                                       point1h<=data1h; point1l<=data1l;
                                       point3h<=data3h; point3l<=data3l;
                                       end
                              3'b001 : begin
                                       if(flashcp)
                                           begin point3h<=4'h8;     point3l<=4'h8; end
                                       else
                                           begin point3h<=4'hF;     point3l<=4'hF; end

                                       point1h<=data1h; point1l<=data1l;
                                       point2h<=data2h; point2l<=data2l;
```

```
                    end
        default     : begin
                        point1h<=data1h; point1l<=data1l;
                        point2h<=data2h; point2l<=data2l;
                        point3h<=data3h; point3l<=data3l;
                        end
        endcase
        if(start && win==3'b000)    state<=Detect;
        else                        state<=Idle;
        end

Detect  : if(in1 && out[2]==0 )     state<=Get1;
        else if(in2 && out[1]==0 )  state<=Get2;
        else if(in3 && out[0]==0 )  state<=Get3;
        else                state<=Detect;

Get1    : begin
        o3<=1;
        bees<=flashcp;
        num<=3'b100;

        delay1<=delay1+1;
        if(delay1==2400)
            begin
            delay1<=0;
            state<=Alter1;
            end
        else
            state<=Get1;
        end

Get2    :   begin
        o5<=1;
        bees<=flashcp;
        num<=3'b010;

        delay1<=delay1+1;
        if(delay1==2400)
            begin
```

```
                delay1<=0;
                state<=Alter2;
                end
            else
                state<=Get2;
            end

Get3        : begin
            o6<=1;
            bees<=flashcp;
            num<=3'b001;

            delay1<=delay1+1;
            if(delay1==2400)
                begin
                delay1<=0;
                state<=Alter3;
                end
            else
                state<=Get3;
            end

Alter1      : begin
            {o1,o2,o3,o4,o5,o6,o7,o8}<={8{flashcp}};
            bees<=0;
            point2h<=4'hF; point2l<=4'hF;
            point3h<=4'hF; point3l<=4'hF;

            delay2<=delay2+1;
            if(delay2==500)
                begin delay2<=0; state<=Ctdown8; end
            else
                state<=Alter1;
            end

Alter2      : begin
            {o1,o2,o3,o4,o5,o6,o7,o8}<={8{flashcp}};
            bees<=0;
            point1h<=4'hF; point1l<=4'hF;
```

```
                point3h<=4'hF; point3l<=4'hF;

                delay2<=delay2+1;
                if(delay2==500)
                    begin delay2<=0; state<=Ctdown8; end
                else
                    state<=Alter2;
                end

Alter3      : begin
                {o1,o2,o3,o4,o5,o6,o7,o8}<={8{flashcp}};
                bees<=0;
                point1h<=4'hF; point1l<=4'hF;
                point2h<=4'hF; point2l<=4'hF;

                delay2<=delay2+1;
                if(delay2==500)
                    begin delay2<=0; state<=Ctdown8; end
                else
                    state<=Alter3;
                end

Ctdown8     : begin
                {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b11111111;

                delay3<=delay3+1;
                if(delay3==1600)
                    begin delay3<=0; state<=Ctdown7; end
                else if(add)
                    state<=Right;
                else if(sub)
                    state<=Wrong;
                else
                    state<=Ctdown8;
                end

Ctdown7     : begin
                {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b01111111;
```

```
            delay3<=delay3+1;
            if(delay3==1400)
                begin delay3<=0; state<=Ctdown6; end
            else if(add)
                state<=Right;
            else if(sub)
                state<=Wrong;
            else
                state<=Ctdown7;
            end

Ctdown6     : begin
            {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00111111;

            delay3<=delay3+1;
            if(delay3==1400)
                begin delay3<=0; state<=Ctdown5; end
            else if(add)
                state<=Right;
            else if(sub)
                state<=Wrong;
            else
                state<=Ctdown6;
            end

Ctdown5     : begin
            {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00011111;

            delay3<=delay3+1;
            if(delay3==1400)
                begin delay3<=0; state<=Ctdown4; end
            else if(add)
                state<=Right;
            else if(sub)
                state<=Wrong;
            else
                state<=Ctdown5;
            end
```

```
Ctdown4    : begin
               {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00001111;

               delay3<=delay3+1;
               if(delay3==1400)
                    begin delay3<=0; state<=Ctdown3; end
               else if(add)
                    state<=Right;
               else if(sub)
                    state<=Wrong;
               else
                    state<=Ctdown4;
               end

Ctdown3    : begin
               {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000111;

               delay3<=delay3+1;
               if(delay3==1400)
                    begin delay3<=0; state<=Ctdown2; end
               else if(add)
                    state<=Right;
               else if(sub)
                    state<=Wrong;
               else
                    state<=Ctdown3;
               end

Ctdown2    : begin
               {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000011;

               delay3<=delay3+1;
               if(delay3==1500)
                    begin delay3<=0; state<=Ctdown1; end
               else if(add)
                    state<=Right;
               else if(sub)
                    state<=Wrong;
               else
```

```
                    state<=Ctdown2;
                end

        Ctdown1     : begin
                    {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000001;

                    delay3<=delay3+1;
                    if(delay3==1600)
                        begin delay3<=0; state<=Ctdown0; end
                    else if(add)
                        state<=Right;
                    else if(sub)
                        state<=Wrong;
                    else
                        state<=Ctdown1;
                    end

        Ctdown0     : begin
                    {o1,o2,o3,o4,o5,o6,o7,o8}<=8'b00000000;
                    state<=Timeover;
                    end

        Timeover    : begin                         //计时已完毕，等候主持人裁决得分。
                    if(back)
                        state<=No_ans;
                    else if(add)
                        state<=Right;
                    else if(sub)
                        state<=Wrong;
                    else
                        state<=Timeover;
                    end

        No_ans      : begin
                    case(num)
                        3'b100 : begin
                                    if(data1l!=0)       data1l<=data1l-5;
                                    else    if(data1h!=0)   begin   data1h<=data1h-1;
data1l<=5; end
```

```
                                    else                    begin   data1h<=4'hF;
data1l<=4'hF; out<=3'b100; end
                                    end
                                3'b010 : begin
                                    if(data2l!=0)       data2l<=data2l-5;
                                    else    if(data2h!=0)   begin   data2h<=data2h-1;
data2l<=5; end
                                    else                    begin   data2h<=4'hF;
    data2l<=4'hF; out<=3'b010; end
                                    end
                                3'b001 : begin
                                    if(data3l!=0)       data3l<=data3l-5;
                                    else    if(data3h!=0)   begin   data3h<=data3h-1;
data3l<=5; end
                                    else                    begin   data3h<=4'hF;
    data3l<=4'hF; out<=3'b001; end
                                    end
//                          default : ;
                            endcase

                            state<=Idle;
                            end

            Right       :   begin
                            case(num)
                                3'b100 : begin
                                    if(data1l!=5)       data1l<=data1l+5;
                                    else    if(data1h!=9)   begin   data1h<=data1h+1;
data1l<=0; end
                                    else                    begin       data1h<=4'h8;
data1l<=4'h8; win<=3'b100; end
                                    end
                                3'b010 : begin
                                    if(data2l!=5)       data2l<=data2l+5;
                                    else    if(data2h!=9)   begin   data2h<=data2h+1;
data2l<=0; end
                                    else                    begin       data2h<=4'h8;
data2l<=4'h8; win<=3'b010; end
                                    end
```

```
                    3'b001 : begin
                        if(data3l!=5)       data3l<=data3l+5;
                        else    if(data3h!=9)   begin   data3h<=data3h+1;
data3l<=0; end
                        else                    begin       data3h<=4'h8;
data3l<=4'h8; win<=3'b001; end
                        end
//                  default : ;
                    endcase

                    state<=Idle;
                    end

            Wrong       : begin
                    case(num)
                    3'b100 : begin
                        if(data1h!=0) data1h<=data1h-1;
                        else            begin  data1h<=4'hF;  data1l<=4'hF;
out<=3'b100; end
                        end
                    3'b010 : begin
                        if(data2h!=0) data2h<=data2h-1;
                        else            begin  data2h<=4'hF;  data2l<=4'hF;
out<=3'b010; end
                        end
                    3'b001 : begin
                        if(data3h!=0) data3h<=data3h-1;
                        else            begin  data3h<=4'hF;  data3l<=4'hF;
out<=3'b001; end
                        end
//                  default : ;
                    endcase

                    state<=Idle;
                    end

        default : state<=Idle;
        endcase
endmodule
```