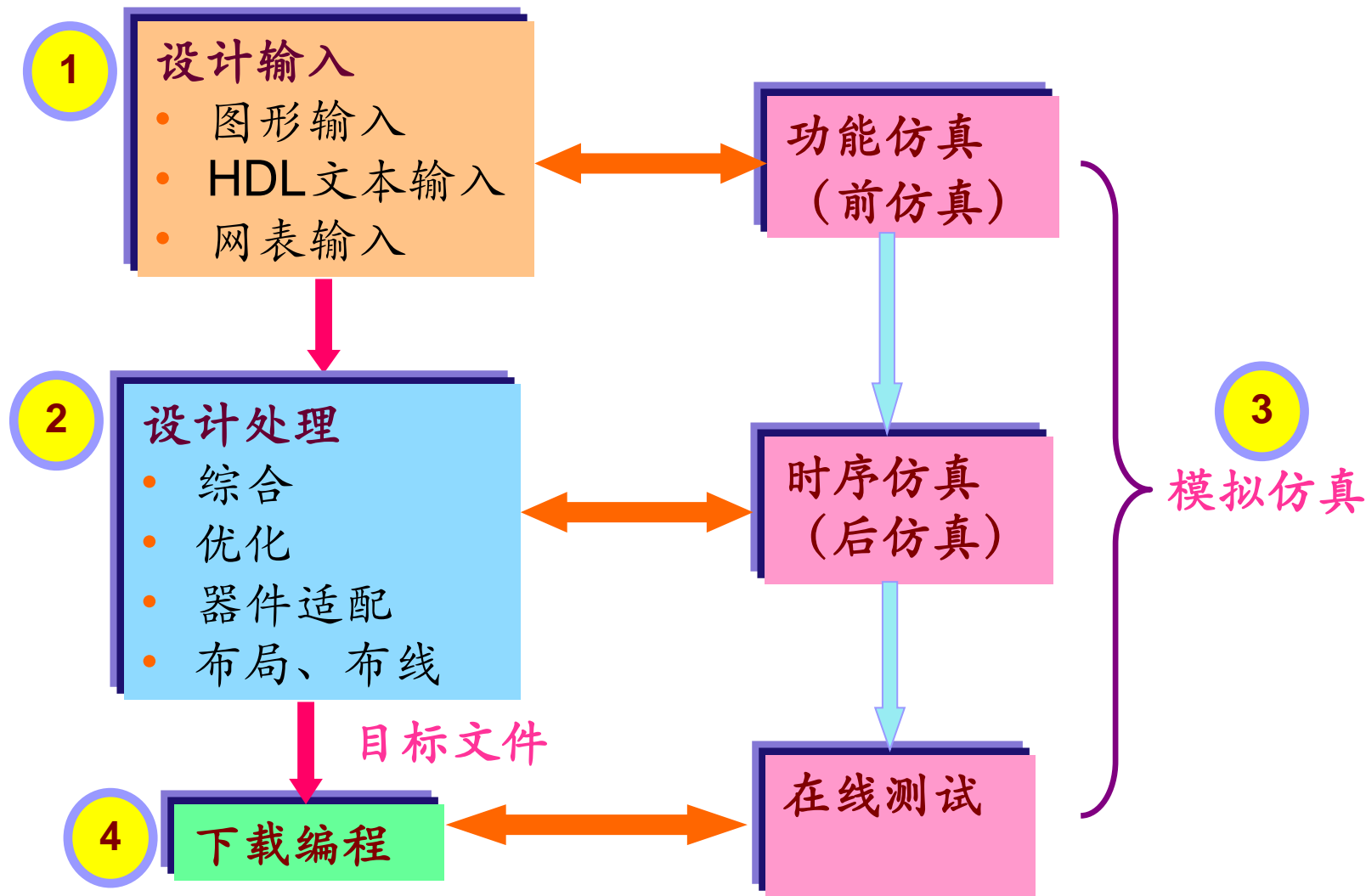


# Verilog HDL—设计实例

## 设计要求

- ◆ 设计一个数字系统，使其：（1）按**1Hz**的频率从**0**开始加**1**计数，当计到**99**时，再来一个时钟信号则产生**进位**信号，并**清零**，然后又从**0**开始加**1**计数；  
（2）具有异步清零功能；（3）两位计数结果用两个**数码管**显示，进位信号用一个**LED**显示。
- ◆ 实验板采用Altera公司的**EP1C20**开发板，系统时钟为**50MHz**，FPGA器件为**EP1C20F400C7**。

# 用PLD实现数字系统的设计流程





# 利用Quatus II 进行PLD设计的流程

1. 在资源管理器下创建一个**工作目录**



2. 在Quatus II中创建一个**工程**

工程名最好与**顶层文件**同名



3. 子模块设计

每个子**模块**用HDL语言或图形方式描述，编译、**仿真**，生成模块符号。



4. 顶层设计

创建**顶层图形文件**或**顶层文本文件**；编译，仿真。



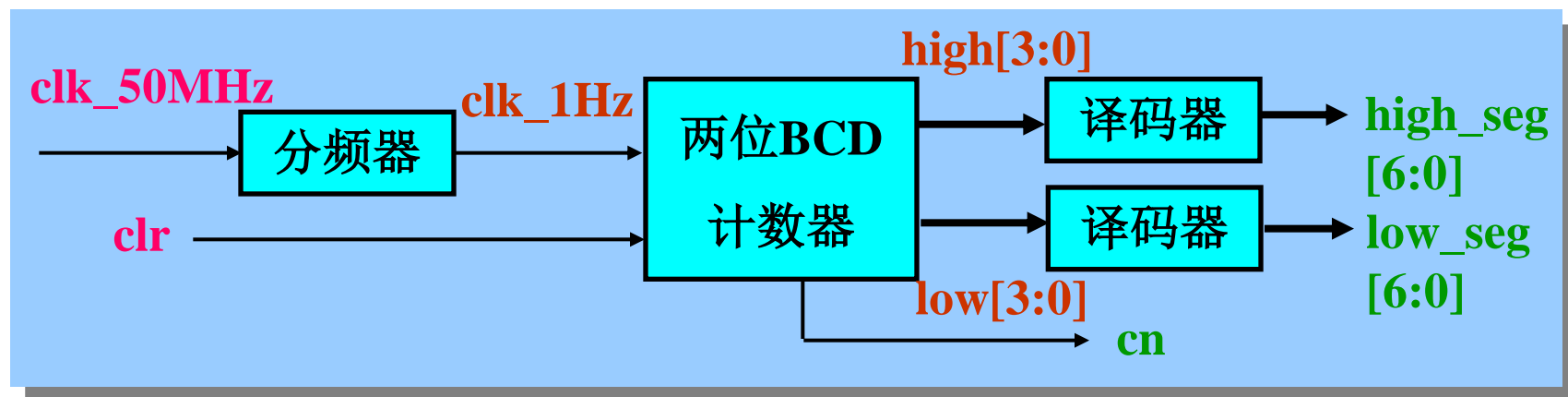
5. 给输入、输出引脚**分配引脚**号码，编程下载。

# 设计思路、功能框图

## ❖ 设计思路——自顶向下的设计方法：

- ◆ 需要一个分频器，将50MHz分频为1Hz；
- ◆ 需要一个两位（个位和十位）BCD计数器，按0→1→2→...→99→0→1→2→... 循环计数；
- ◆ 需要一个译码器，将BCD计数器的4位二进制数输出译码为7段显示器的7段电平输入。

## ❖ 功能框图





# 设计步骤

最好每个工程都有自己的工作目录！

- ❖ **第1步**：在资源管理器下创建一个工作目录 **counter\_7seg**。
- ❖ **第2步**：启动Quatus II，执行菜单命令 “ **File > New Project Wizard**”，创建一个工程，工程名为 **counter\_7seg**。

若要打开一个已有的工程，则执行 “ **File > Open Project ...**”命令。

- ❖ **第3步**：设计子模块

模块名与程序文件名同名！

## (1) 编辑

执行菜单命令 “ **File > New**”，新建一个文本文件 **f50MHz\_to\_1Hz.v**，采用Verilog HDL语言描述。

## 第3步：设计子模块（1/8）

(2) 将该子模块设为顶层实体

一定要做！

执行 “**Project > Set as Top-level Entity**” 命令

(3) 编译

执行 “**Processing > Start > Start Analysis & Synthesis**” 命令，  
或单击工具条上的分析与综合快捷按钮 ，进行**分析与综合**。

(4) 产生功能仿真用网表文件

执行 “**Processing > Generate Functional Simulation Netlist**”  
命令

**注意：**

1. 对于子模块，进行**分析与综合**即可，不必进行全编译（执行 “**Processing > Start Compilation**” 命令）——否则太慢！
2. 若要**进行功能仿真**，必须执行第（4）步！



## 子模块1—分频电路

### ❖ 【模块1】 50MHz到1Hz的分频电路

模块名(同文件名)

```
module f50MHz_to_1Hz(clk_1KHz,clk_1Hz,clkin);  
    output clk_1KHz,clk_1Hz; // 输出端口声明  
    input clkin;              // 输入端口声明  
    reg[15:0] count1;  
    reg[9:0] count2;  
    reg clk_1KHz,clk_1Hz;  
    parameter count_width=50000;
```

**注意：**模块名可以由字母、数字、下划线和\$符号组成，但第一个字符不能是数字或\$符号！

## 子模块1 一分频电路（续）

续前页

```
always @(posedge clkin)           // (1) 50MHz到1KHz的分频
begin
    if(count1[15:0]==count_width-1)
        count1[15:0]=0;
    else
        count1[15:0]=count1[15:0]+1;
    clk_1KHz=count1[15];           // 计数器最高位作为时钟信号输出
end
always @(posedge clk_1KHz) // (2) 1KHz到1Hz的分频
begin
    if(count2 == 1000-1) count2 = 0;
    else count2 = count2+1;
    clk_1Hz=count2[9];
end
endmodule
```





## 子模块2—两位BCD计数器

### ❖ 【模块2】 两位BCD计数器（加1）

```
module bcd_counter(high,low,cn,clr,clk);  
    output[3:0] high,low;    // 高4位（十位）输出和低4位（个位）输出  
    output cn;                // 高4位的进位  
    input clr,clk;  
    reg[3:0] high,low;  
    reg cn;  
    always @(posedge clk or posedge clr)  
        begin  
            if (clr)           // 异步清零  
                begin  
                    cn=1'b0;    // 进位信号也必须清零！  
                    high[3:0]=4'd0;  
                    low[3:0]=4'd0;  
                end  
        end
```

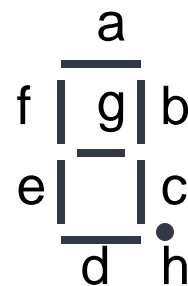
## 子模块2—两位BCD计数器（续）

```
else      // 计数，采用2个if语句的嵌套
begin
    if(low[3:0]==4'd9)      // 低4位是否为9?
    begin
        low[3:0]=4'd0;
        if(high[3:0]==4'd9) // 高4位是否为9?
        begin
            high[3:0]=4'd0;
            cn=1'b1;      // 如果高4位、低4位均为9，则产生进位
        end
    else
        high[3:0]=high[3:0]+4'd1;
    end
    else
    begin
        low[3:0]=low[3:0]+4'd1;cn=1'b0;
    end
end
end
endmodule
```

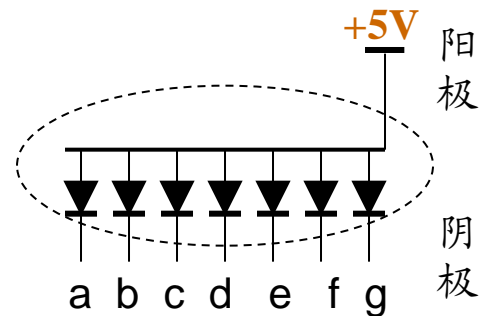
## 子模块3—7段译码器

❖ **【模块3】** 7段LED显示器（共阳极）译码器。

```
module decoder_7seg (segment,data);
    output[6:0] segment;
    // segment[6:0]对应a,b,c,d,e,f,g
    input[3:0] data;
    reg [6:0] segment;
    always @(data)
        begin
            case (data)
                4'd0:segment = 7'b0000001;
                4'd1:segment = 7'b1001111;
                .....
                4'd9:segment = 7'b0000100;
                default:segment = 7'b1111111;
                //当输入不是0~9时，数码管不亮
            endcase
        end
endmodule
```



7段LED显示  
器（数码管）



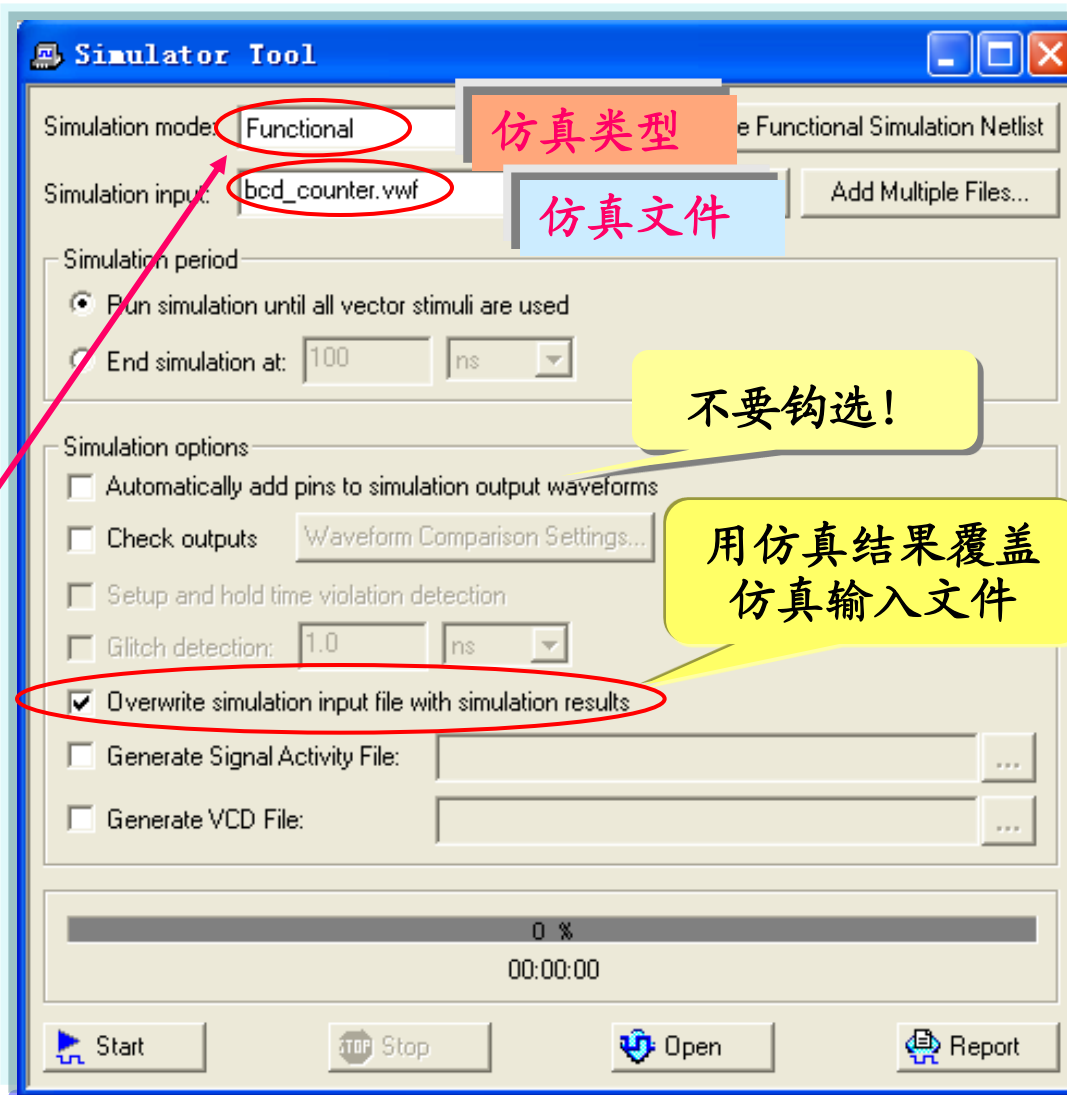
共阳极结构

## 第3步：设计子模块之（5）功能仿真

◆ 检验逻辑功能是否正确

◆ 以**bcd\_counter.v**为例

- 执行“**File > New**”命令，新建一个仿真波形文件**.vwf**（如**bcd\_counter.vwf**）；
- 在**Edit**菜单中设置结束时间**End Time**（这里为**120s**）、网格大小**Grid Size**（如**1s**），编辑输入波形（如这里给时钟信号**clk**设置时钟周期为**1s**）；
- 执行“**Processing > Simulation Tool**”命令，打开仿真器工具窗口，选择“**Functional**”。





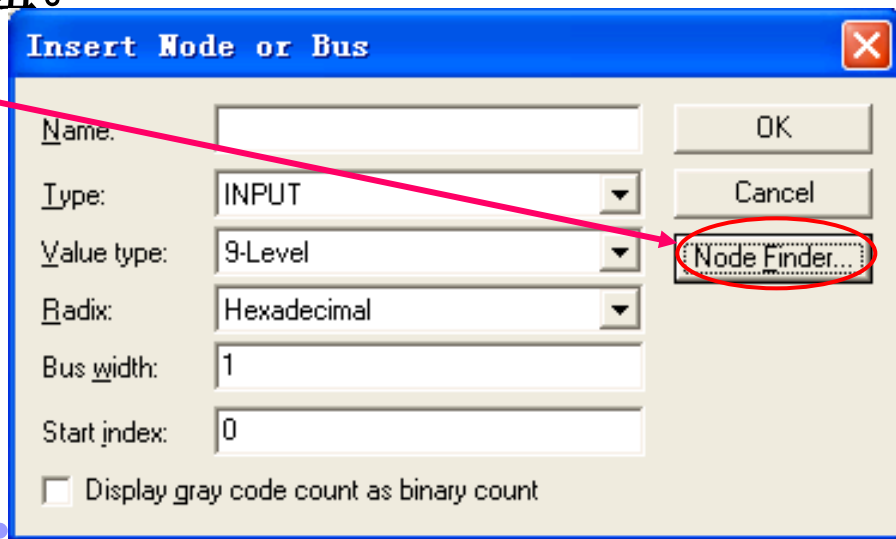
# 创建仿真文件的方法

1. 建立一个仿真文件 (.vwf)
  - (1) 执行**File>New**命令
  - (2) 设置仿真的结束时间 (用**Edit>End Time**命令)
  - (3) 设置网格间距 (用**Edit > Grid Size**命令)
2. 将当前工程的端口信号选入波形文件中
  - (1) 打开**Node Finder** 窗口
  - (2) 列出设计中的所有节点名 (输入、输出或者组)
  - (3) 选择**必要**的节点加入波形文件中
3. 编辑输入波形 (输入激励信号)  
即对输入引脚设置输入信号
4. 保存文件 (最好与设计文件名同名)

# 将当前工程的端口信号选入波形文件中

## 1) 打开Node Finder 窗口（有两种方法）

- ◆ 方法一：执行“**View > Utility Windows > Node Finder**”命令，弹出Node Finder界面；
- ✓ 方法二：在波形编辑器左边Name列的空白处单击右键，在弹出菜单中选择“**Insert Node or Bus...**”命令，或双击左键，则弹出对话框Insert Node or Bus；在其中选择“**Node Finder...**”按钮。



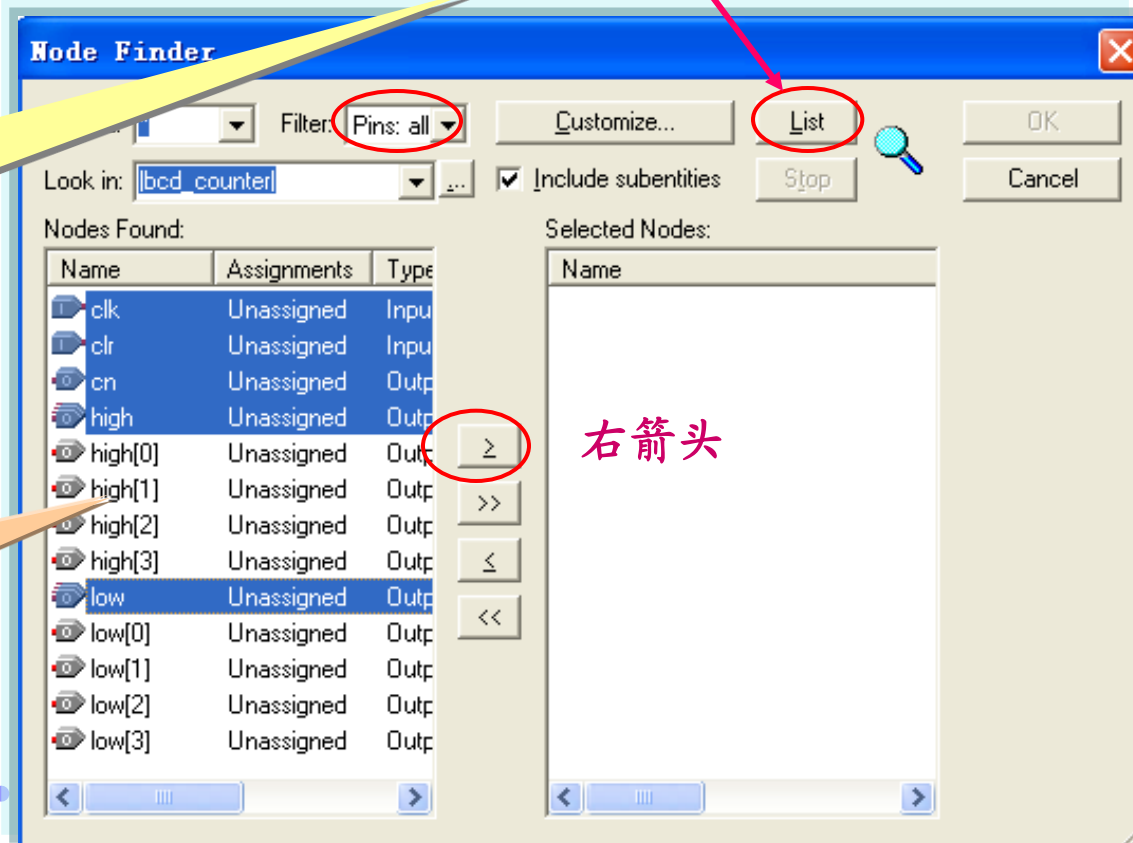
## 2) 列出设计中的所有节点名

### 2) 列出设计中的所有节点名

- 在Node Finder界面中，在Filter列表中选择Pins:all，在Named栏中键入“\*”，然后单击List按钮，则在“Nodes Found”中会出现所有节点名。

是当前工程  
顶层实体的  
节点

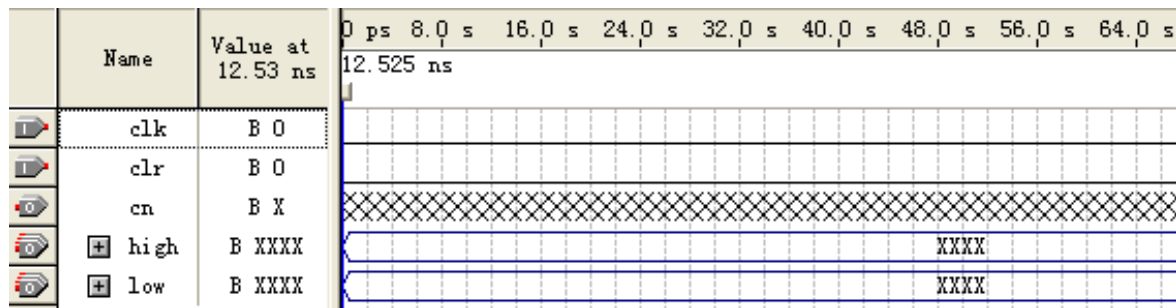
总线信号中的单个  
信号一般不选



### 3) 选择必要的节点加入波形文件中

#### 3) 选择必要的节点加入波形文件中

- ◆ 在**Node Found**栏中选择要加入波形文件中的节点（可用**Shift**键或**Ctrl**键选择多个连续或不连续的节点），单击**右箭头**，将所选择节点送入“**Selected Nodes**”栏中。
- ◆ 单击“**OK**”，则所选的信号和组出现在波形编辑器中。



- 未编辑的**输入**信号波形默认为**低**电平；
- **输出**信号和**隐含节点**默认为**未定义（X）**电平。


- ❖ 总线信号最好不要选择单个信号，而是选择一组信号！
- ❖ 一般将输入信号放在波形编辑器中的上方，输出信号放在下方——便于观察波形！

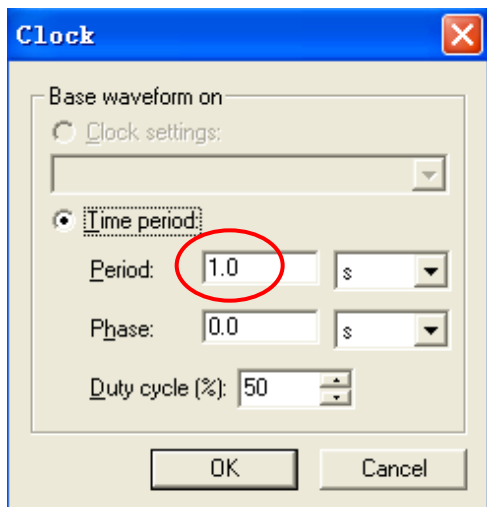


# 编辑输入波形（输入激励信号）

## ❖ 指定输入节点的逻辑电平变化

### ① 编辑时钟信号波形

- ◆ 在时钟节点名（**clk**）上单击右键，在弹出菜单中选择“**Value>Clock...**”命令，或选择时钟节点名后直接单击图形工具按钮 **Overwrite Clock** ，则弹出**Clock**对话框；
- ◆ 在其中设置**时钟周期**（如这里为**1s**）、**相位**和**占空比**（默认为**50%**）

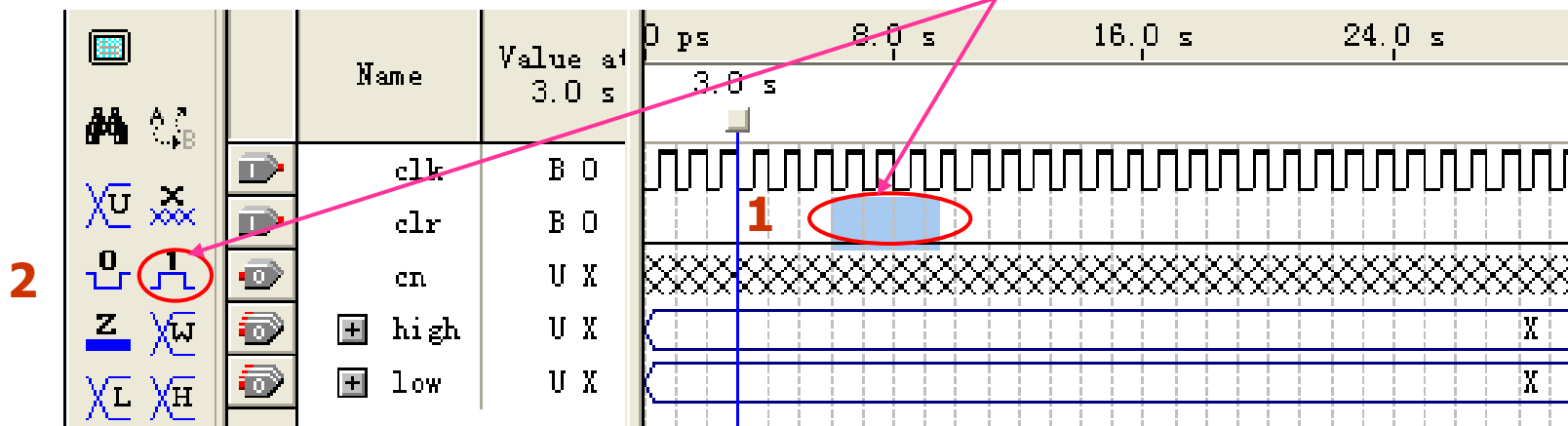


- ❖ 如果某信号在**Value**栏显示为“**A**”，则选中该信号，右击，在弹出菜单中选择“**Properties**”，将**Radix**设为“**Binary**”（如**clk**、**clr**）或“**Hexadecimal**”或“**Unsigned Decimal**”（如**high[3:0]**和**low[3:0]**）

# 编辑任意信号波形

## ② 编辑clr信号波形


- ◆ 在波形编辑器中选择输入节点**clr**，单击窗口左部的图形工具按钮（**0**或**1**），编辑**整个**波形；
- ◆ 或拖动鼠标选定信号在某个时间段的**区域**，单击合适的图形工具按钮；或在选中区域上单击右键，在**Value**菜单中选择需要设置的值，编辑**该段**波形。



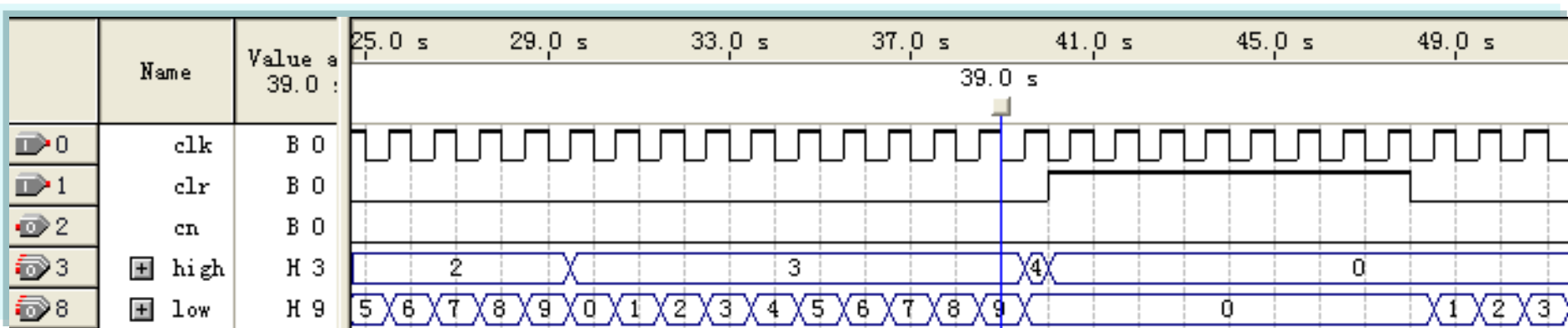
将bcd\_counter.vwf某一段clr设置为1

## 进行功能仿真

❖ 执行 “**Processing > Simulation Tool**”命令，打开仿真器工具窗口，进行**功能**仿真。

- ❖ 建议初学Quartus II时，不要直接执行 “**Processing > Start Simulation**”命令，或直接单击工具条上的仿真快捷按钮（因为这样是对已打开且上次刚刚仿真过的文件进行仿真）！
- ❖ 而应执行 “**Processing > Simulation Tool**”命令，打开仿真器工具窗口，在其中设置仿真类型为Timing或Functional，并确认 “**Simulation input**”栏中显示的是当前需要仿真的波形文件，再单击 “**Start**”，开始仿真。

# 仿真波形



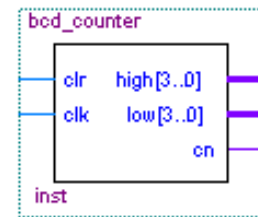
**bcd\_counter.vwf**

**注意：**执行“**View>Fit in Window**”命令，可以使波形缩小到窗口大小，以便于观察整个波形！


## 第3步：设计子模块之（6）创建模块符号

### （6）创建模块符号（文件后缀为.bsf）

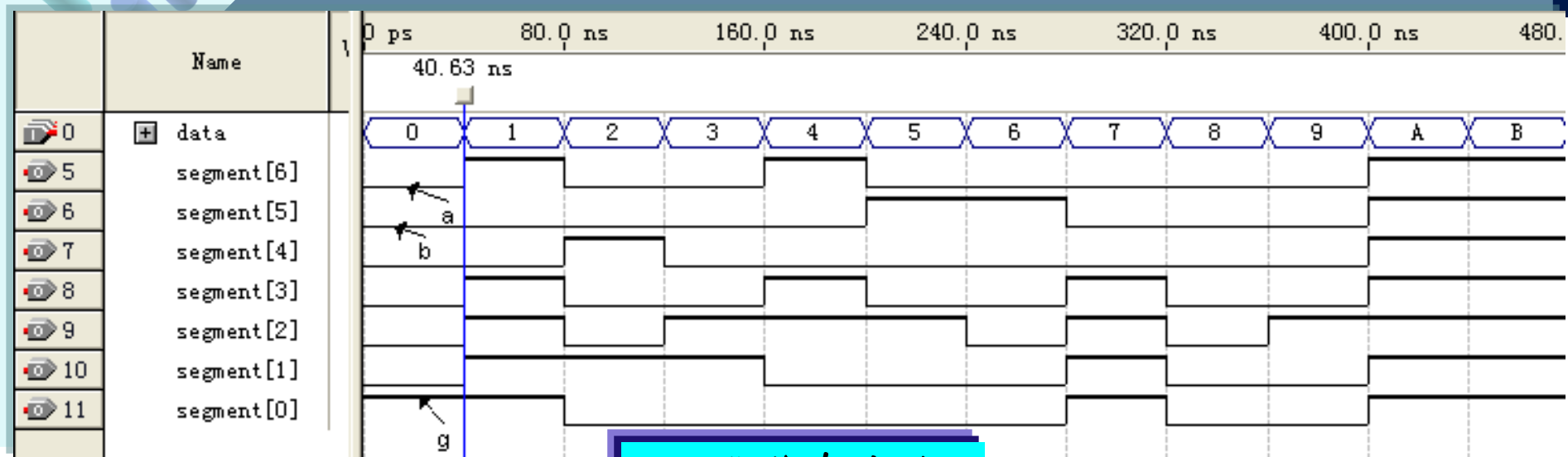
- ◆ 可以为设计好的子模块文件（可以是文本文件或原理图文件）生成一个模块符号文件（**Block Symbol Files, .bsf**），供顶层图形文件调用。
- ◆ **创建逻辑符号**：仿真通过后，执行“**File > Create/Update > Create Symbol Files for Current File**”菜单命令。
- ◆ **调用逻辑符号**：在原理图中双击鼠标左键，打开“**Symbol**”对话框；展开“**Project**”，选择刚创建的模块符号，单击“**OK**”按钮；移动鼠标，将模块放置到适当位置。
- ◆ **编辑逻辑符号**：用“**File>Open**”命令打开.bsf文件；或者在原理图中选中符号，执行“**Edit > Edit Selected Symbol**”菜单命令，或右击该符号，选择“**Edit Selected Symbol**”命令，进入符号编辑界面，修改后保存。



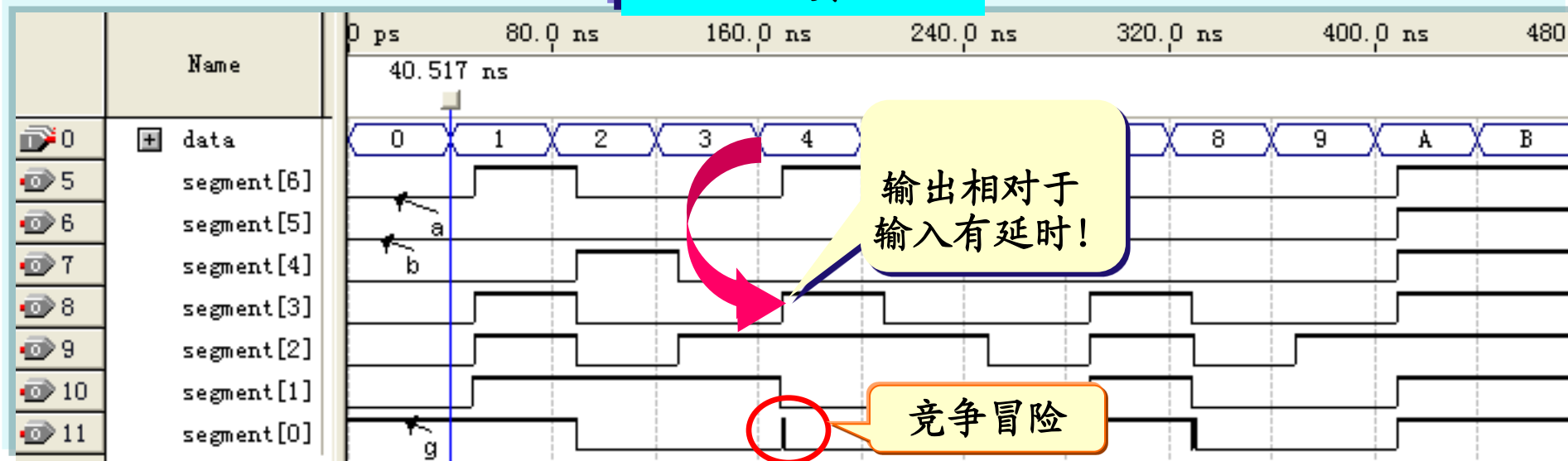
# 进行时序仿真的方法

- ❖ **时序仿真**检验逻辑**功能**是否正确，并验证器件配置后的**时序**关系是否正确。
  - ❖ 若要采用不同的仿真类型，则需重新选择编译的“**Processing**”菜单，对工程重新进行编译！
    - ◆ 如果要进行功能仿真，一定要先执行 “**Processing > Start > Start Analysis & Synthesis**”命令，进行**分析与综合**；执行 “**Processing > Generate Functional Simulation Netlist**”命令，产生**功能仿真网表**文件；
    - ◆ 如果要进行时序仿真，一定要先执行 “**Processing > Start Compilation**”命令，进行**全编译**。
- ① 将要仿真的模块（如decoder\_7seg.v）设为顶层实体；
  - ② 执行 “**Processing > Start Compilation**”命令，或单击 “全编译” 按钮 ，进行**全编译**；
  - ③ 执行 “**Processing > Simulation Tool**”命令，打开仿真器工具窗口，选择仿真类型为 “**Timing**”，进行时序仿真。

# decoder\_7seg.v功能仿真与时序仿真比较



功能仿真波形



时序仿真波形



## 第4步：设计顶层图形文件（1/5）

### ❖第4步：设计顶层图形文件

**顶层图形设计文件**即是把一个设计的各个子模块符号放在一个图形文件中，以描述设计的总体功能。

采用图形设计文件，便于表达各个子模块的连接关系和芯片内部逻辑到引脚的接口。

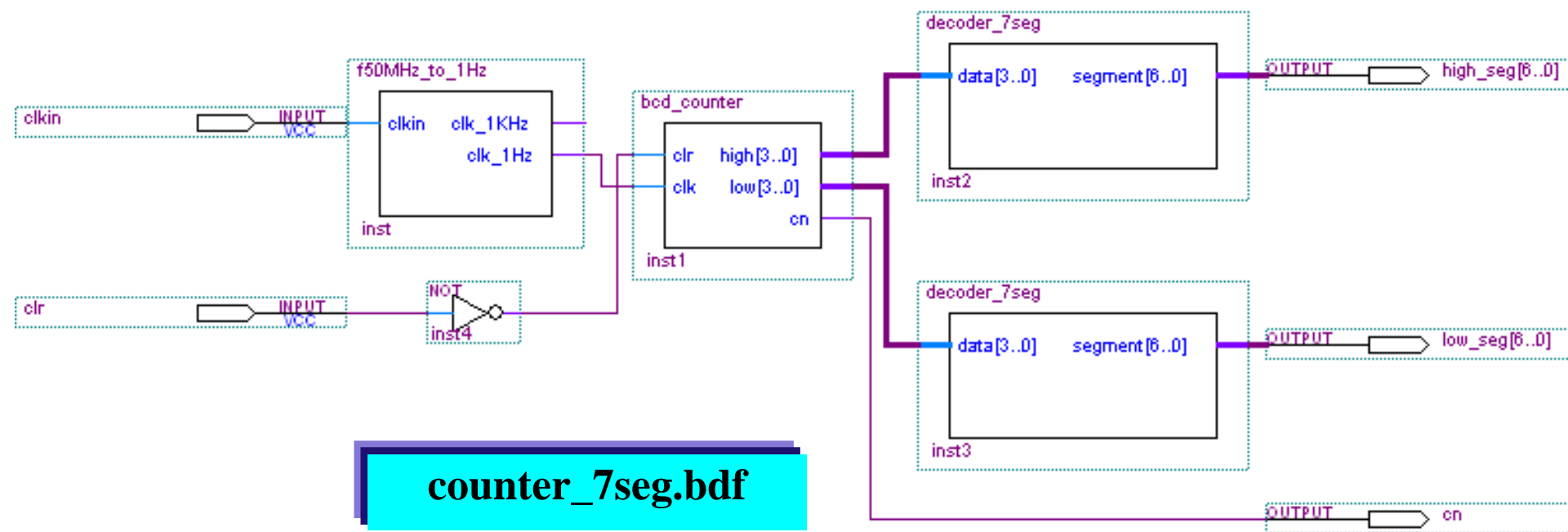
#### (1) 创建一个顶层图形文件**counter\_7seg.bdf**

- ◆ 执行**File-New**命令，打开“**New**”对话框；
- ◆ 选择“**Device Design Files**”标签下的“**Block Diagram/Schematic File**”；
- ◆ 单击**OK**，打开图形编辑器。
- ◆ 将各模块符号放到图中，添加输入、输出引脚，连线。
- ◆ 用**File > Save As...**命令保存文件，文件名后缀为**.bdf**。



# 顶层图形文件counter\_7seg.bdf

- **放置模块符号**：双击图形窗口的空白处，弹出**Symbol**对话框，在“Library”栏中展开“**Project**”文件夹，在其中选择模块符号（如“bcd\_counter”）；单击“OK”。
- **添加输入、输出引脚**：双击鼠标左键，打开“**Symbol**”对话框，在“Name”框中键入“**INPUT**”（或“**OUTPUT**”），单击“OK”。



counter\_7seg.bdf



## 设计顶层图形文件（2/5）

千万别忘了！

(2) 将该图形文件设置为顶层实体

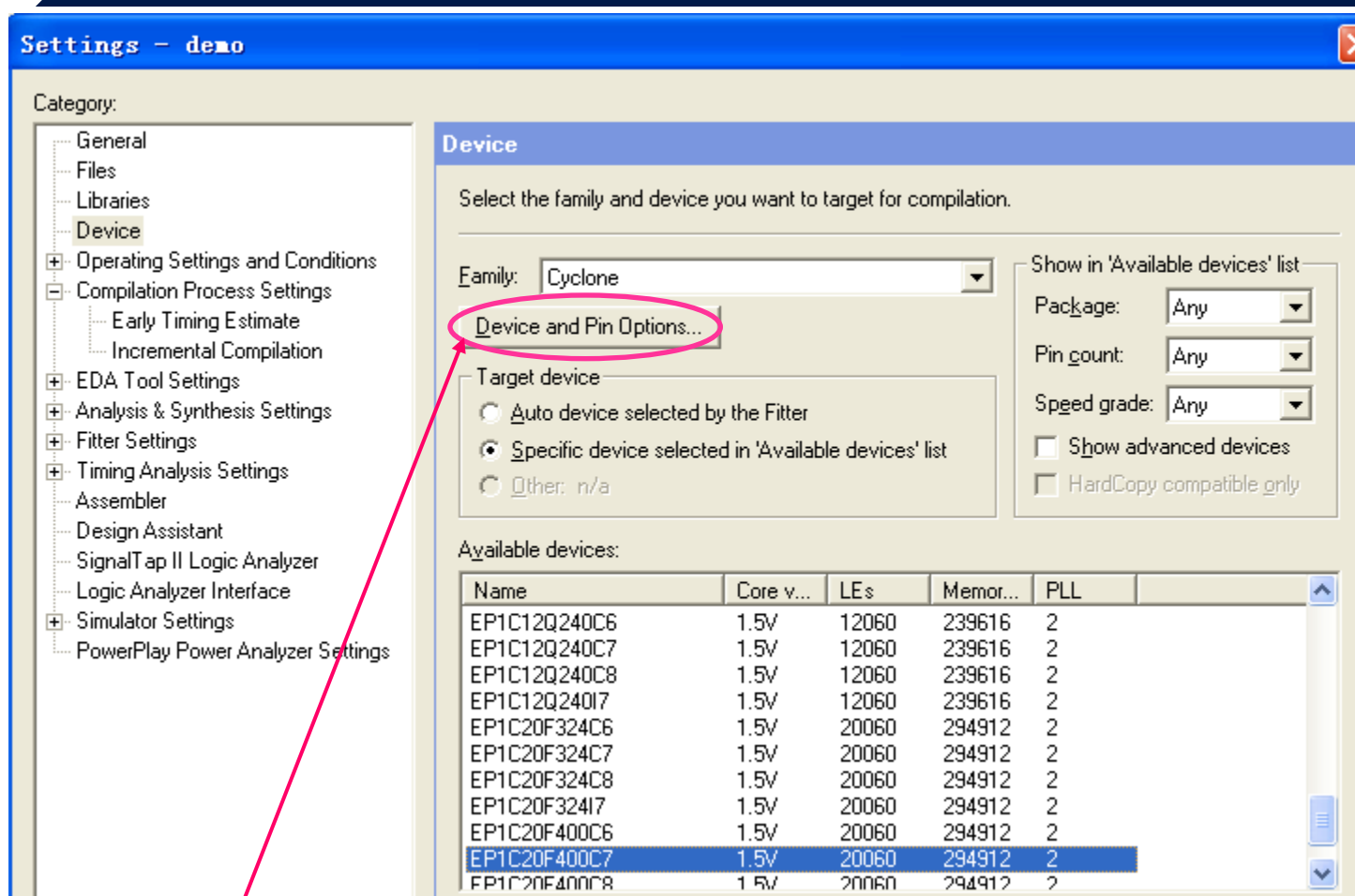
执行“**Project> Set as Top-Level Entity**”命令，或在  
**Project Navigator**中右击该文件名，选择“**Set as Top-Level Entity**”。

(3) 进行编译器选项设置

执行“**Assignments> Settings...**”命令。

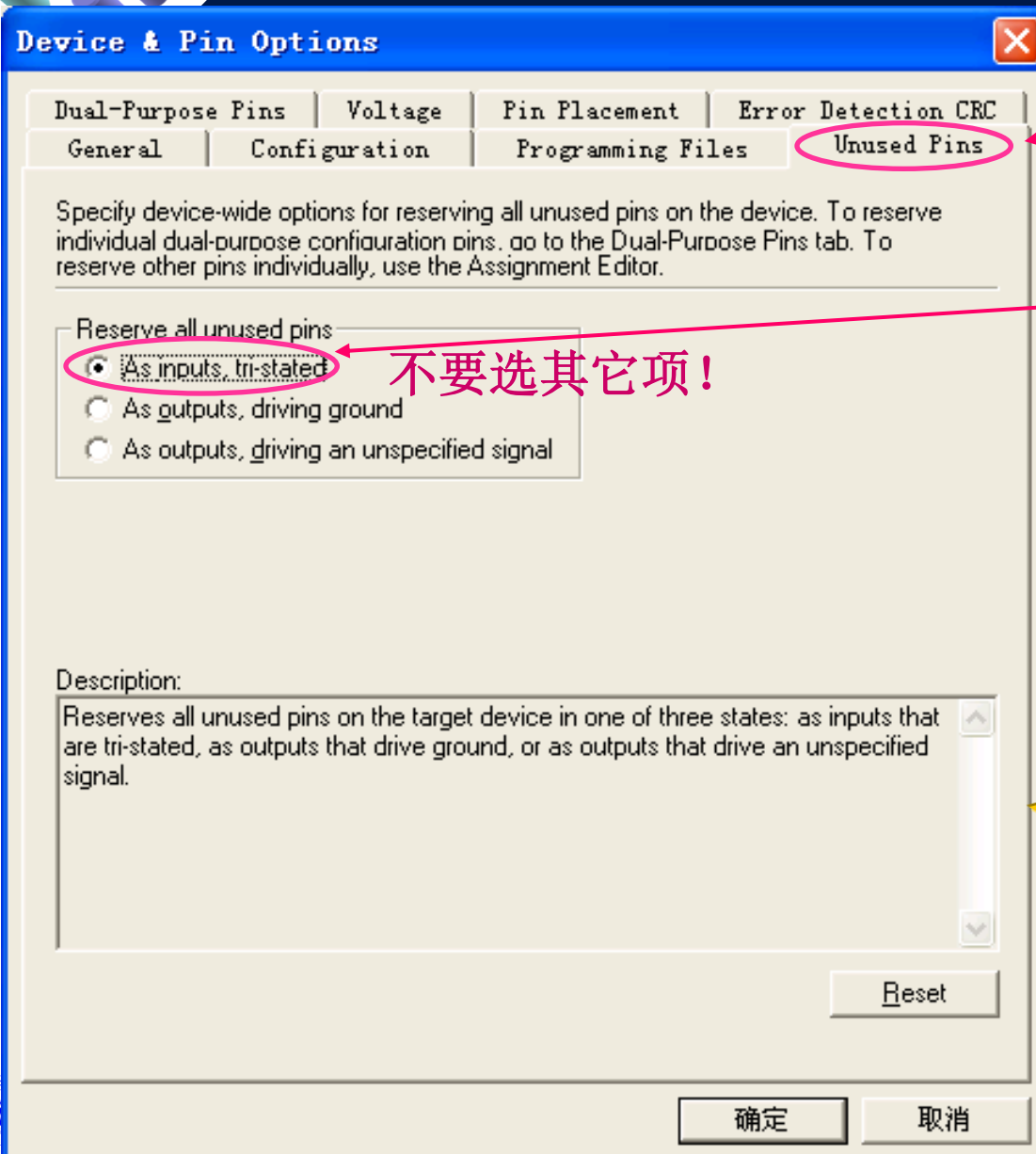
- ① 在**Settings**对话框的**Category**栏中选择“**Device**”，在  
**Device**页面中指定**目标器件**（一定要与实际器件型号完全一致！本例为**EP1C20F400C7**）；

## 设计顶层图形文件 (3/5)



② 将未使用引脚指定为输入：然后在Device页面中单击 “**Device & Pin Options**”按钮，打开 “**Device & Pin Options**” 对话框；

## 设计顶层图形文件（4/5）



选择“**Unused Pins**”标签，在“**Reserve all unused pins**”域中选择“**As inputs, tri-stated**”单选钮。

千万不要将未使用引脚指定为输出，否则无法成功下载设计！

## 设计顶层图形文件（5/5）

③ 进行编译过程设置等（在**Settings**对话框的**Category**栏中选择“**Compilation Process Settings**”

- 采取**智能编译技术**：勾选“**Use Smart compilation**”选项；
- 采取**增量编译技术**：选中“**Incremental compilation**”下的“**Full incremental compilation**”单选钮。

为使重编译速度加快，一定选中此项！

（4）全编译

执行“**Processing > Start Compilation**”命令，或单击工具条上的全编译快捷按钮。



（5）仿真（如果必要的话）

❖ 如果系统分频级数太多，而各子模块已通过仿真的话，则不必对顶层设计文件进行仿真，否则仿真时间太长！

## 第5步：分配引脚号、编程下载

❖ 第5步：给输入、输出引脚分配引脚号码，编程下载

(1) 对顶层图形文件counter\_7seg.bdf 进行引脚锁定；

前锁定：执行 “**Assignnments > Assignment Editor**”菜单命令

信号名

引脚号

	To	Location	I/O Bank	I/O Standard	General Function	Special Function	Reserved	Enabled
1	clkln	PIN_K5	1	3.3-V LVTTTL	Dedicated Clock	CLK0/LVDSCLK1p		Yes
2	clr	PIN_W3	4	3.3-V LVTTTL	Column I/O	LVDS128p		Yes
3	high_seg[6]	PIN_T5	4	3.3-V LVTTTL	Column I/O	LVDS126p/DQ1B7		Yes
4	high_seg[5]	PIN_U5	4	3.3-V LVTTTL	Column I/O	LVDS126n/DQ1B6		Yes
5	high_seg[4]	PIN_V5	4	3.3-V LVTTTL	Column I/O	LVDS125p		Yes
6	high_seg[3]	PIN_W5	4	3.3-V LVTTTL	Column I/O	LVDS125n		Yes
7	high_seg[2]	PIN_T6	4	3.3-V LVTTTL	Column I/O	DPCLK7/DQ51B		Yes
8	high_seg[1]	PIN_T7	4	3.3-V LVTTTL	Column I/O	VREF2B4		Yes
9	high_seg[0]	PIN_W6	4	3.3-V LVTTTL	Column I/O	LVDS124p		Yes
10	low_seg[6]	PIN_U6	4	3.3-V LVTTTL	Column I/O	LVDS123p/DQ1B4		Yes
11	low_seg[5]	PIN_V6	4	3.3-V LVTTTL	Column I/O	LVDS123n		Yes
12	low_seg[4]	PIN_W7	4	3.3-V LVTTTL	Column I/O	LVDS122p		Yes
13	low_seg[3]	PIN_Y7	4	3.3-V LVTTTL	Column I/O	LVDS122n		Yes
14	low_seg[2]	PIN_R7	4	3.3-V LVTTTL	Column I/O	LVDS121p/DQ1B3		Yes
15	low_seg[1]	PIN_T8	4	3.3-V LVTTTL	Column I/O	LVDS121n/DQ1B2		Yes
16	low_seg[0]	PIN_V7	4	3.3-V LVTTTL	Column I/O	LVDS120p/DQ1B1		Yes
17	cn	PIN_E14	2	3.3-V LVTTTL	Column I/O	LVDS56n		Yes
18	<<new>>	<<new...>						

# 前锁定方法

❖ **前锁定**：在Assignment Editor中完成引脚锁定

(1) 打开分配编辑器

- ◆ 执行 “**Assignnments > Assignment Editor**” 菜单命令，然后在Category栏中选择**pin**；
- ◆ 或直接选择 “**Assignnments > Pins**” 菜单命令。

(2) 选择引脚名

- ◆ 在Assignment Editor界面中，双击 “**To**” 所在列的 <new> 单元，从下拉框中选择引脚名

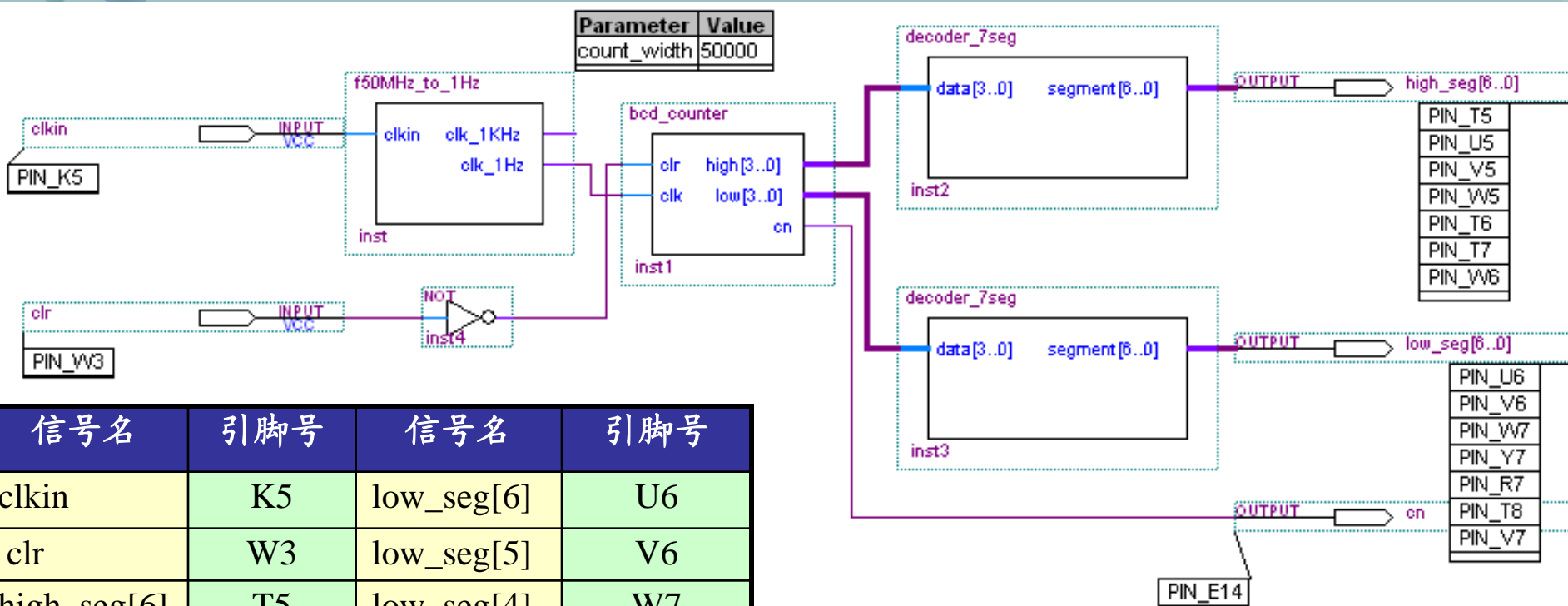
(3) 指定引脚号

- ◆ 双击 “**Location**” 所在列的 <new> 单元，从下拉框中指定引脚号；或单击该引脚名所对应的Location <new> 单元，直接键入引脚号

(4) 完成设计中所有引脚的指定，**保存分配**

一定要做！

# 引脚锁定后counter\_7seg.bdf



信号名	引脚号	信号名	引脚号
clk	K5	low_seg[6]	U6
clr	W3	low_seg[5]	V6
high_seg[6]	T5	low_seg[4]	W7
high_seg[5]	U5	low_seg[3]	Y7
high_seg[4]	V5	low_seg[2]	R7
high_seg[3]	W5	low_seg[1]	T8
high_seg[2]	T6	low_seg[0]	V7
high_seg[1]	T7	cn	E14
high_seg[0]	W6		

## 引脚锁定后counter\_7seg.bdf

- ❖ 如果图中没有显示出引脚锁定情况，可以执行“**View>Show Pin and Location Assignments**”命令。



## 重新编译和编程下载

(2) 全编译，生成编程目标文件；

(3) 编程下载；



连接开发板，执行“**Tools > Programmer**”命令，或单击编程快捷按钮，在编程器窗口中选中“**Program/Configure**”复选框；单击**Start**按钮，开始编程下载。若完成编程，则在**Message**窗口中显示“**Configuration succeeded**”。

(4) 验证设计。

利用实验板上的按钮**SW0**产生清零信号，观察数码管和**LED**的显示，看是否与预定的功能相符。

❖ 若在**Hardware type**列表中没有所需的编程硬件（本实例为**USB- Blaster**），则必须按下面的步骤安装编程硬件的驱动程序！然后在编程器窗口中单击“**Hardware Setup**”按钮，添加硬件（本实例为**USB- Blaster (USB-0)**）

课后练习

独立完成此设计实例的全过程



# 安装编程硬件的驱动程序

- ◆ 当第一次插上编程硬件时，会自动弹出“找到新的硬件向导”对话框，提示“Windows可以连接到Windows Update以搜索软件吗？”，选择“否，暂时不”，单击“下一步”；
- ◆ 选择“从列表或指定位置安装（高级）”单选钮，单击“下一步”；
- ◆ 选择“在这些位置上搜索最佳驱动程序”单选钮，并选择“在搜索中包括这个位置”复选框，单击“浏览”按钮，指定硬件驱动程序的路径为“C:\altera\Quartus72\quartus\drivers\usb-blaster\x32\”，单击“下一步”；
- ◆ 弹出一个“硬件安装”提示框，单击“仍然继续”，开始安装；
- ◆ 安装完毕时，单击“完成”。

实验时我们使用KX-DN7实验台，硬件类型为ByteBlasterMV|[LPT1]，驱动程序在..... drivers\win2000下

## 引脚锁定的第2种方法—后锁定

### 步骤

- ❖ 每个设计工程都有一个设置文件`.qsf` (Quartus Settings File), 包含所有的设置参数, 如器件型号、引脚锁定、速率及面积的比例、时间参数的要求和布线设置等。
- ❖ **后锁定**——在**编译后**通过修改`.qsf`文件 (其名与顶层实体同名) 中的适配结果来修改引脚锁定。

1. 打开设置文件`.qsf`文件 (其名与**工程**同名);
2. 在Pin & Location Assignments处用复制、粘贴的方法增加对输出、输入引脚的定义语句, 或直接修改引脚号或信号名;
3. 保存修改后的`.qsf`文件。
4. 对原下载用设计文件**重新编译**, 以生成新的编程目标文件! (`.sof`, `.pof`, `.hex`)。

千万别  
忘了哦!

# 引脚锁定信息

## 修改counter\_7seg.qsf文件

```
# Pin & Location Assignments
# =====
set_location_assignment PIN_K5 -to clk_in
set_location_assignment PIN_U3 -to clr
set_location_assignment PIN_T5 -to high_seg[6]
set_location_assignment PIN_U5 -to high_seg[5]
set_location_assignment PIN_V5 -to high_seg[4]
set_location_assignment PIN_W5 -to high_seg[3]
set_location_assignment PIN_T6 -to high_seg[2]
set_location_assignment PIN_T7 -to high_seg[1]
set_location_assignment PIN_W6 -to high_seg[0]
set_location_assignment PIN_U6 -to low_seg[6]
set_location_assignment PIN_V6 -to low_seg[5]
set_location_assignment PIN_W7 -to low_seg[4]
set_location_assignment PIN_Y7 -to low_seg[3]
set_location_assignment PIN_R7 -to low_seg[2]
set_location_assignment PIN_T8 -to low_seg[1]
set_location_assignment PIN_V7 -to low_seg[0]
set_location_assignment PIN_E14 -to cn
```

引脚号

信号名

在哪里可以方便地查看引脚锁定情况呢？

