



第4章

时序仿真与硬件实现

4.1 VHDL程序输入与仿真测试

4.1.1 编辑和输入设计文件

(1) 新建一个文件夹。

(2) 输入源程序。

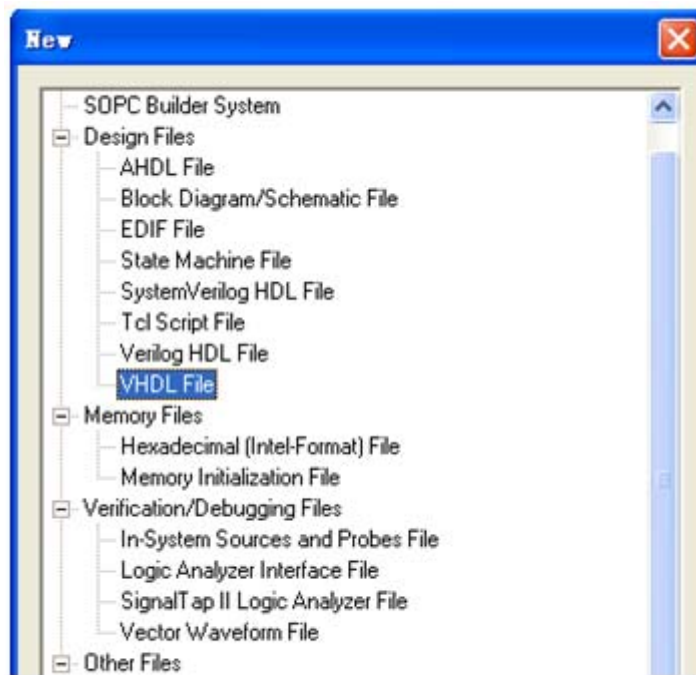


图 4-1 选择编辑文件类型

4.1 VHDL程序输入与仿真测试

4.1.1 编辑和输入设计文件

- (1) 新建一个文件夹。
- (2) 输入源程序。
- (3) 文件存盘。

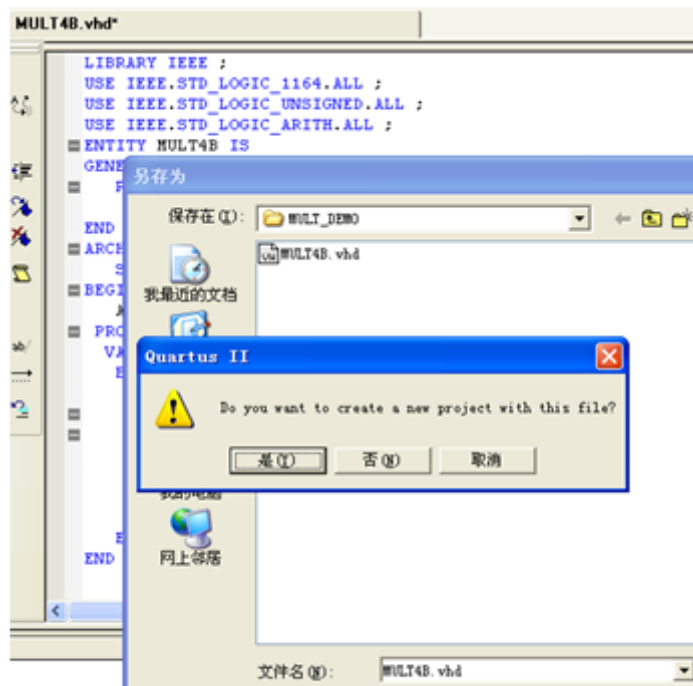


图 4-2 编辑输入源程序并存盘

4.1 VHDL程序输入与仿真测试

4.1.2 创建工程

(1) 打开并建立新工程管理窗口。

(2) 将设计文件加入工程中。



图 4-3 利用 New Project Wizard 创建工程 MULT4B

4.1 VHDL程序输入与仿真测试

4.1.2 创建工程

(3) 选择目标芯片。

(4) 工具设置。

(5) 结束设置。

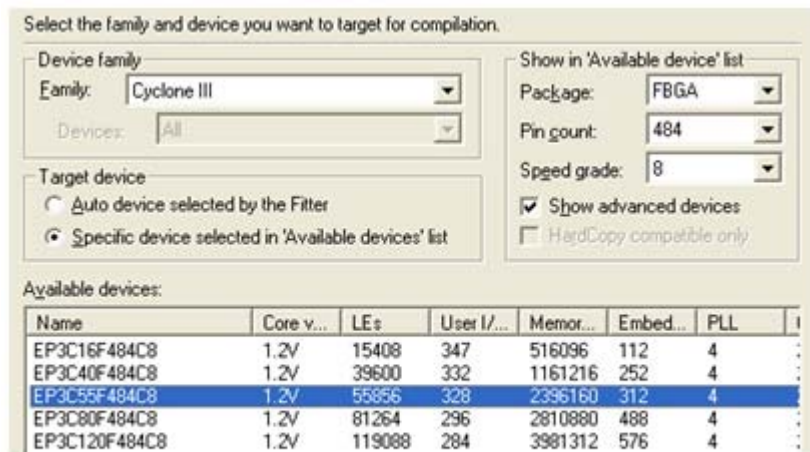


图 4-4 选择目标器件 EP3C55F484C8

4.1 VHDL程序输入与仿真测试

4.1.3 全程编译前约束项目设置

(1) 选择FPGA目标芯片。

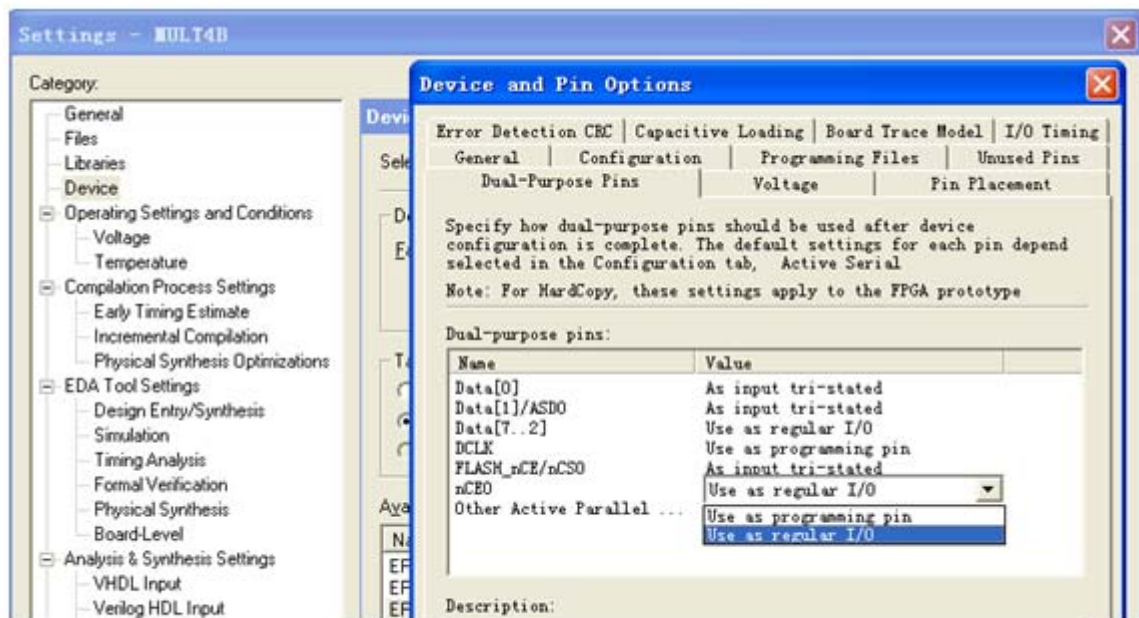


图 4-5 选择配置器件的工作方式



4.1 VHDL程序输入与仿真测试

4.1.3 全程编译前约束项目设置

- (2) 选择配置器件的工作方式。
- (3) 选择配置器件和编程方式。
- (4) 选择目标器件引脚端口状态。
- (5) 对双功能引脚进行设置。

4.1 VHDL程序输入与仿真测试

4.1.4 全程综合与编译

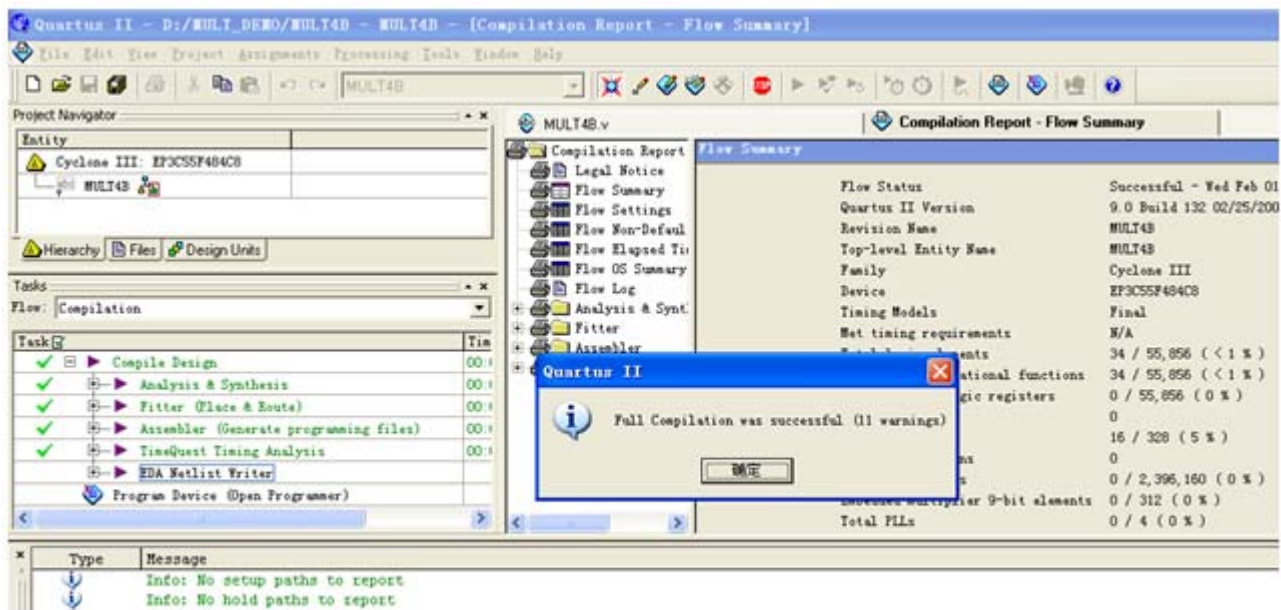


图 4-6 全程编译无错后的报告信息

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

(1) 打开波形编辑器。

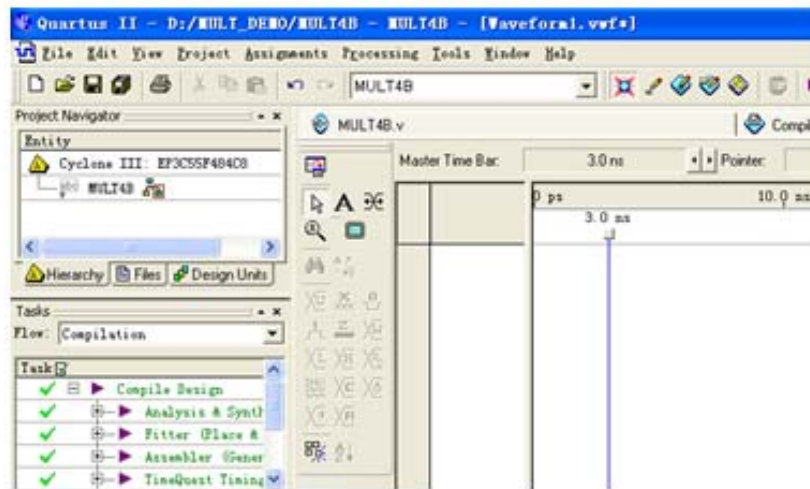


图 4-7 激励信号波形编辑器窗口

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

(2) 设置仿真时间区域。



图 4-8 设置仿真时间长度

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

(3) 波形文件存盘。

(4) 将工程**MULT4B**的端口信号节点选入波形编辑器中。

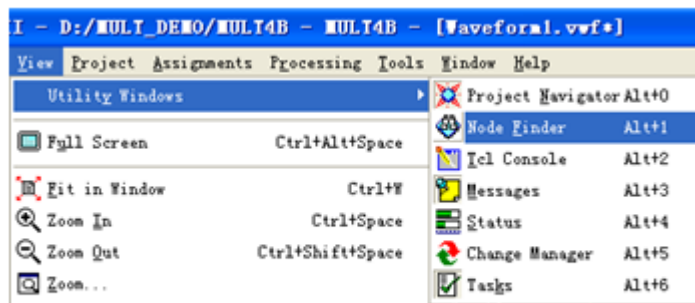


图 4-9 打开信号节点查询窗口

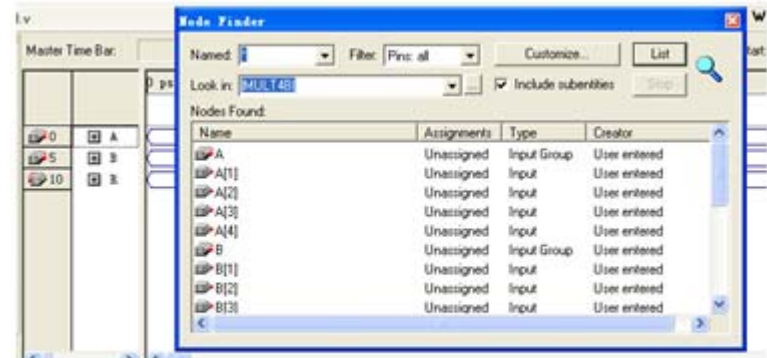


图 4-10 向波形编辑器拖入信号节点

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

(5) 总线数据格式设置和参数设置。

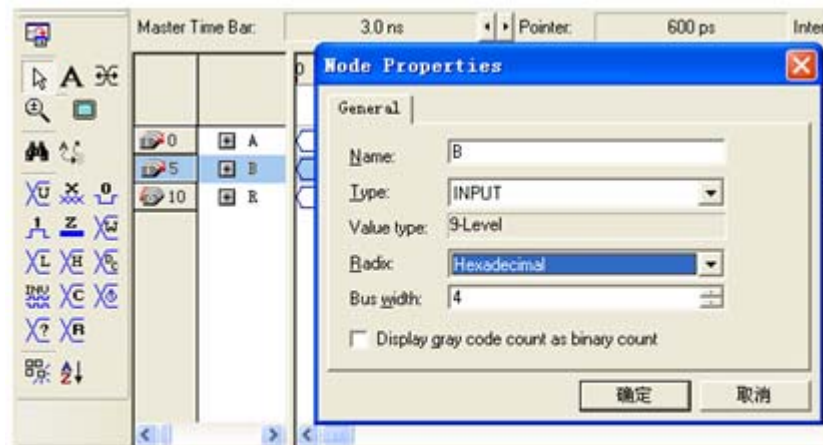


图 4-11 设置好的激励波形图，及选择总线数据格式

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

(6) 编辑输入波形数据(输入激励信号)。

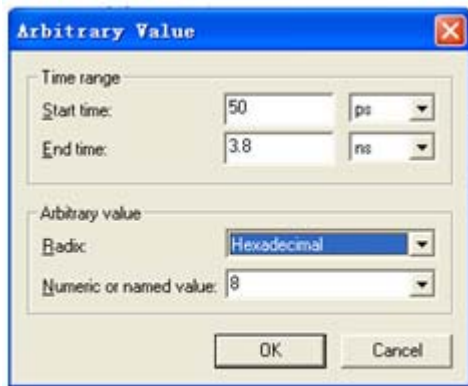


图 4-12 设置输入数据

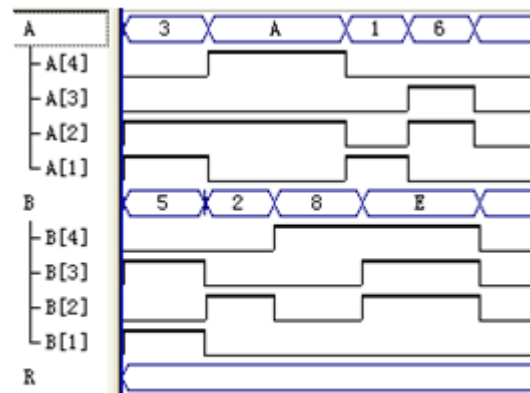


图 4-13 编辑好的 vwf 文件

4.1 VHDL程序输入与仿真测试

4.1.5 仿真测试

- (7) 仿真器参数设置。
- (8) 启动仿真器。
- (9) 观察仿真结果。

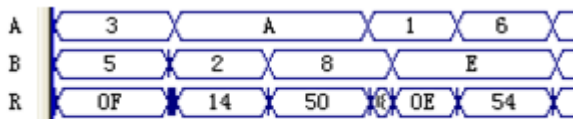


图 4-14 仿真波形输出(十六进制类型)

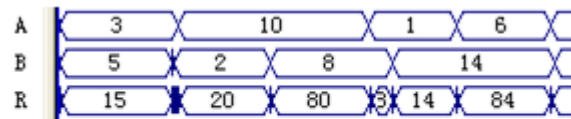


图 4-15 仿真波形输出(十进制类型)

4.1.6 RTL图观察器应用

4.2 引脚锁定与硬件测试

4.2.1 引脚锁定

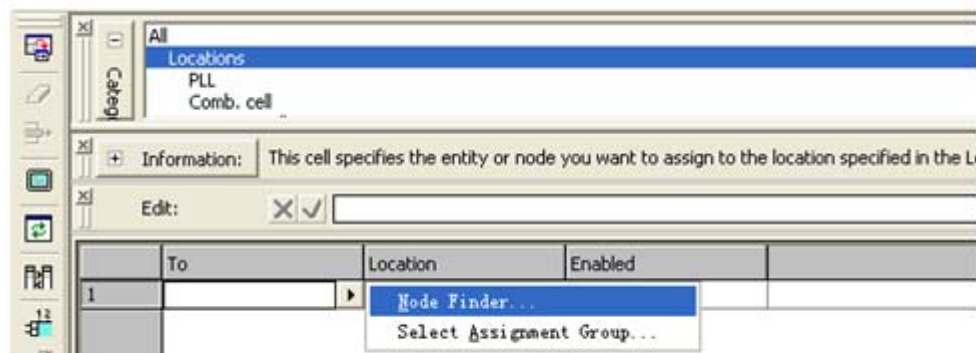


图 4-16 利用 Assignment Editor 编辑器锁定 FPGA 引脚

4.2 引脚锁定与硬件测试

4.2.1 引脚锁定

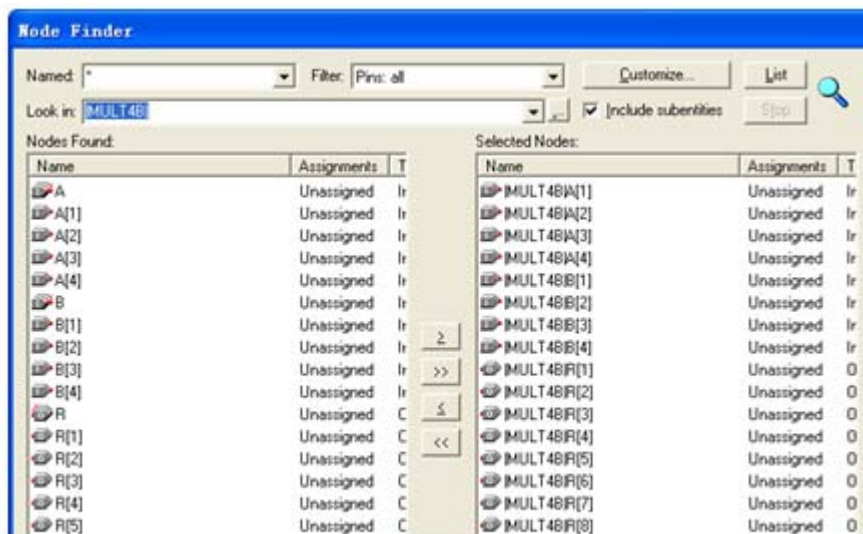


图 4-17 利用 Node Finder 工具选择需要锁定引脚的信号

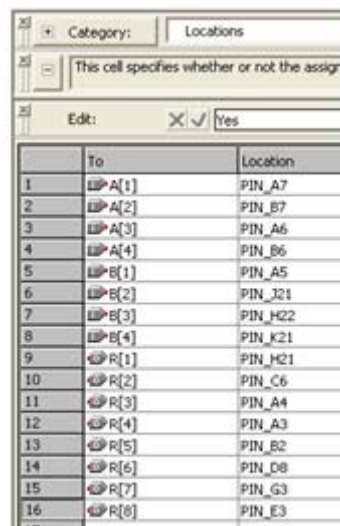


图 4-18 引脚锁定窗口

4.2 引脚锁定与硬件测试

4.2.2 编译文件下载

(1) 打开编程窗和配置文件。

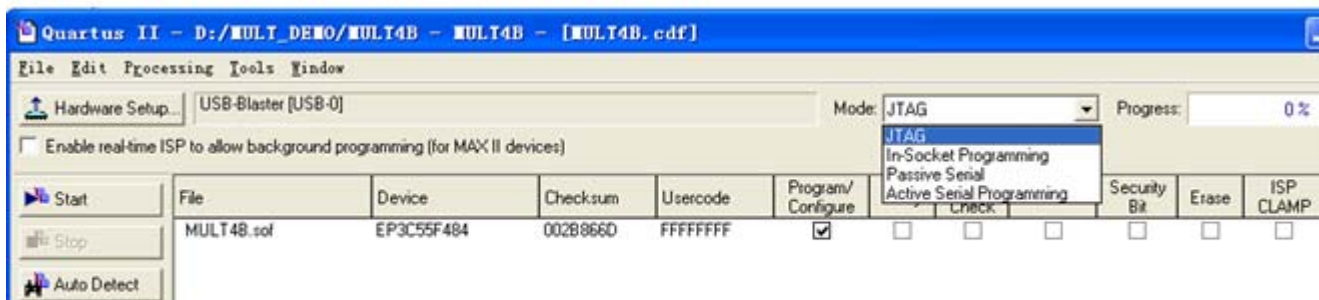


图 4-19 选择 JTAG 编程模式

4.2 引脚锁定与硬件测试

4.2.2 编译文件下载

(2) 设置编程器。



图 4-20 加入编程下载方式

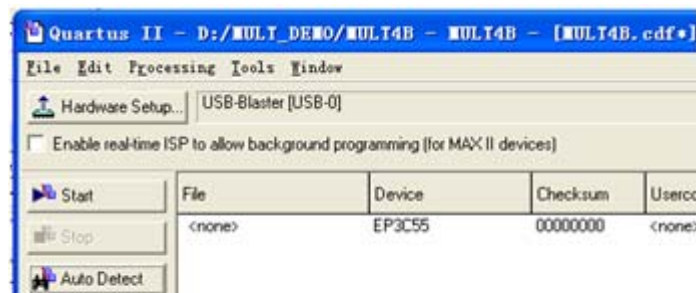


图 4-21 检测实验板上的 FPGA 器件

(3) 硬件测试。

4.2 引脚锁定与硬件测试

4.2.3 JTAG间接编程模式

1. 将SOF文件转化为JTAG间接配置文件

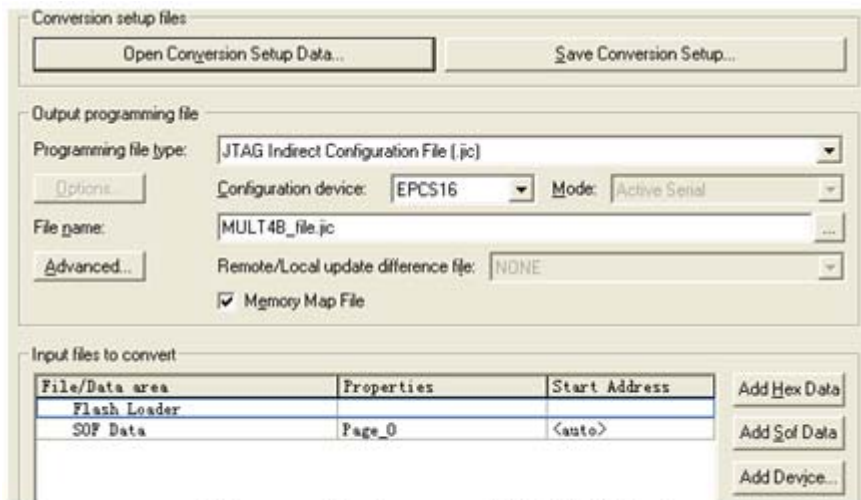


图 4-22 设定 JTAG 间接编程文件



图 4-23 选择目标器件 EP3C55

4.2 引脚锁定与硬件测试

4.2.3 JTAG间接编程模式

1. 将SOF文件转化为JTAG间接配置文件



图 4-24 加入 SOF 文件，并选择压缩模式

4.2 引脚锁定与硬件测试

4.2.3 JTAG间接编程模式

2. 下载JTAG间接配置文件。

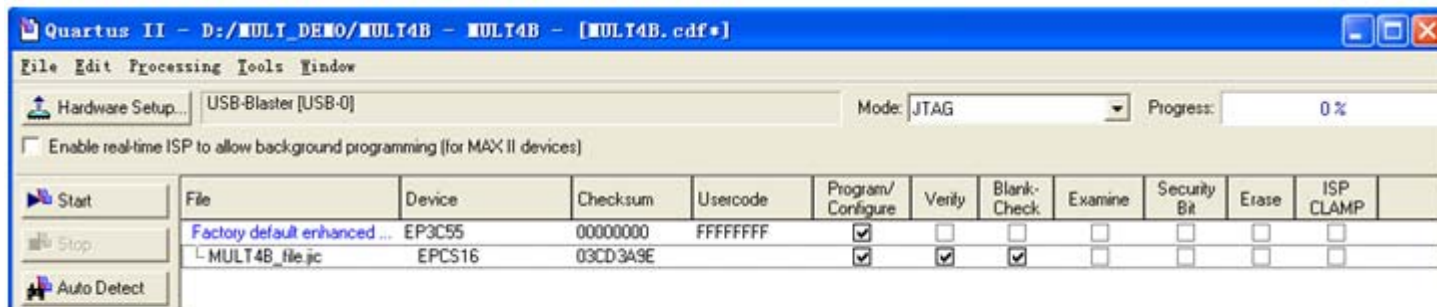


图 4-25 用 JTAG 模式对配置器件 EPCS16 进行编程

4.2.4 USB-Blaster驱动程序安装方法

4.3 电路原理图设计流程

1. 建立原理图文件工程和仿真



图 4-26 选择打开元件输入窗

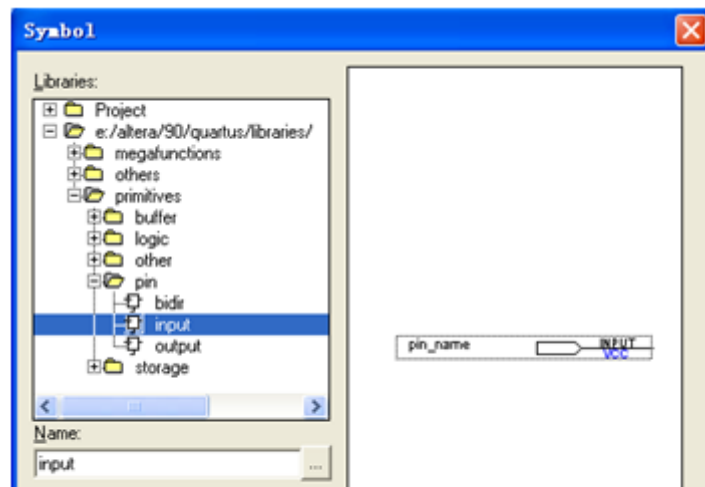


图 4-27 在元件输入对话框输入引脚

4.3 电路原理图设计流程

1. 建立原理图文件工程和仿真

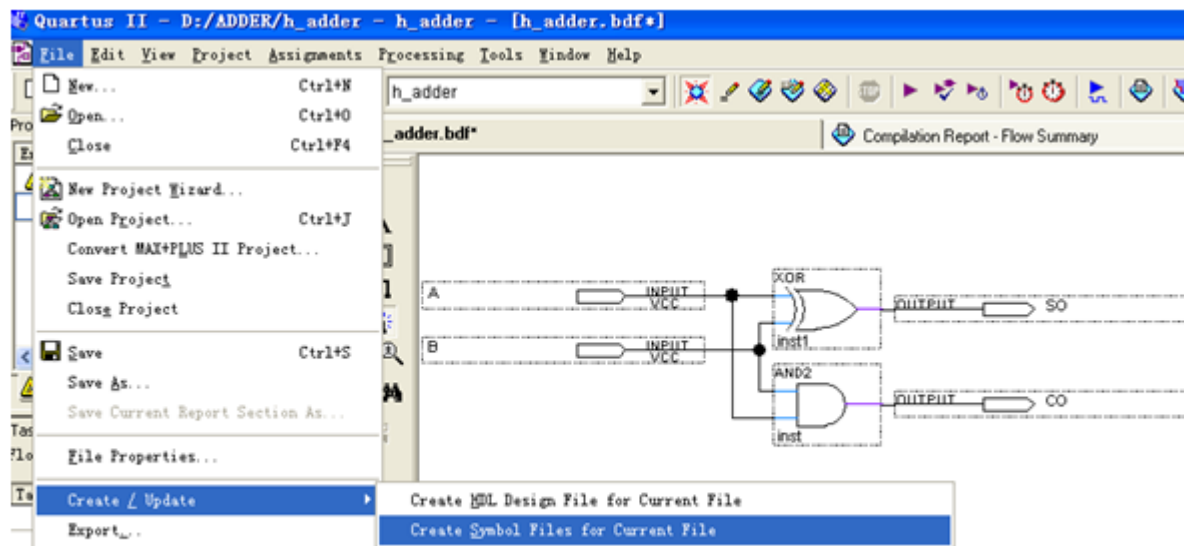


图 4-28 完成设计，并将半加器封装成一个元件，以便在更高层设计中调用

4.3 电路原理图设计流程

2. 将设计项目设置成可调用的元件

3. 设计全加器顶层文件

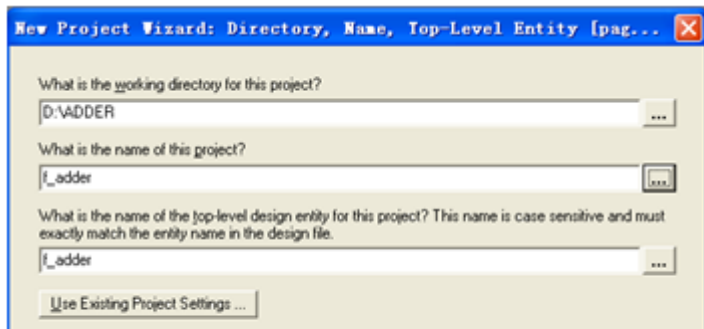


图 4-29 全加器 f_adder.bdf 工程设置

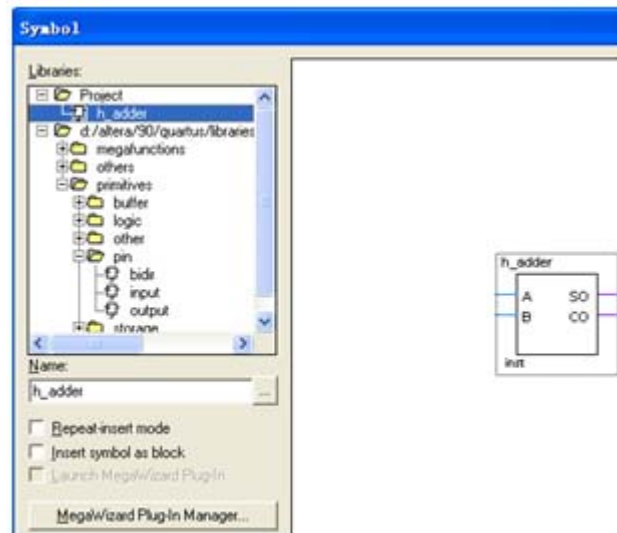


图 4-30 在 f_adder 工程下加入半加器

4.3 电路原理图设计流程

4. 对设计项目进行时序仿真

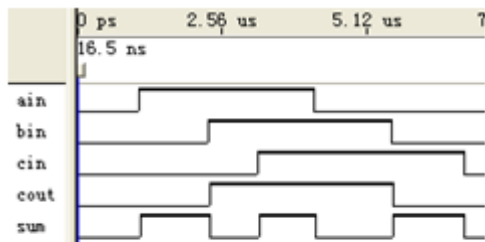
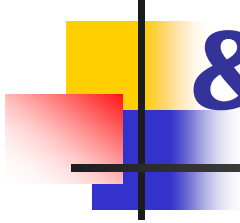


图 4-31 全加器的仿真波形

5. 硬件测试

4.4 HDL版本设置及Analysis & Synthesis功能



4.5 利用属性表述实现引脚锁定

【例 4-1】

```
ARCHITECTURE ONE OF MULT4B IS
    attribute chip_pin : string; -- chip_pin 被定义为字符串数据类型 string
    attribute chip_pin of A : signal is "B6,A6,B7,A7" ;
    attribute chip_pin of B : signal is "K21,H22,J21,A5" ;
    attribute chip_pin of R : signal is "E3,G3,D8,B2,A3,A4,C6,H21" ;
    SIGNAL A0 : std_logic_vector(2*S DOWNT0 1);
BEGIN

    attribute chip_pin of EN : signal is "B6";
```

Edit -> Insert Template -> VHDL -> Synthesis Attributes;

4.6 keep属性应用

【例 4-2】

```
ARCHITECTURE fd1 OF f_adder IS
```

```
...
```

```
SIGNAL net1,net2,net3 : STD_LOGIC;
```

```
attribute keep : boolean; --由于“true”的类型是布尔类型 boolean
```

```
attribute keep of net1 : signal is true;
```

```
BEGIN
```

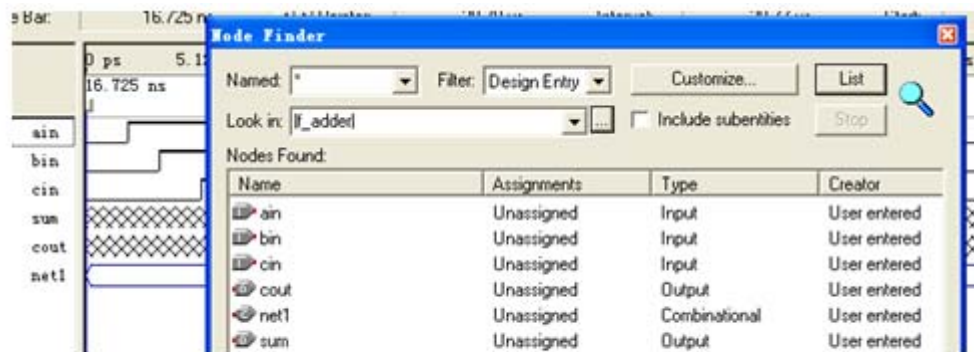


图 4-32 向仿真激励信号波形编辑窗调入信号 net1

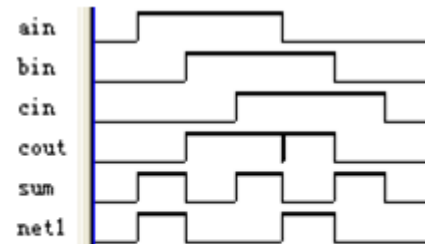


图 4-33 例 4-2 的仿真波形



4.7 SignalProbe使用方法

1. 按常规流程完成设计仿真和硬件测试

2. 设置SignalProbe Pins

4.7 SignalProbe使用方法

3. 编译SignalProbe Pins测试信息并下载测试

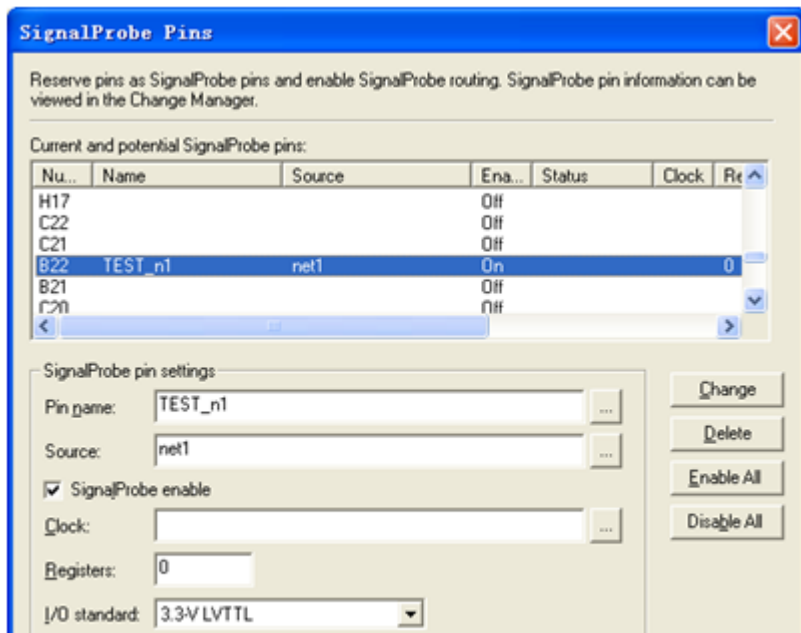


图 4-34 在 SignalProbe 对话框设置探测信号 net1

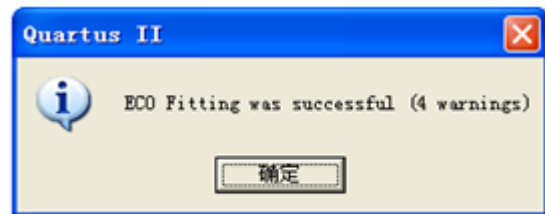


图 4-35 ECO 文件编译成功

4.8 宏模块逻辑功能查询

```

FUNCTION 74138 (g1, g2an, g2bn, c, b, a)
  RETURNS (y0n, y1n, y2n, y3n, y4n, y5n, y6n, y7n);

```

Inputs						Outputs							
Enable		Select				Y0N	Y1N	Y2N	Y3N	Y4N	Y5N	Y6N	Y7N
G1	G2*	C	B	A									
X	H	X	X	X	H	H	H	H	H	H	H	H	
L	X	X	X	X	H	H	H	H	H	H	H	H	
H	L	L	L	L	L	H	H	H	H	H	H	H	
H	L	L	L	H	H	L	H	H	H	H	H	H	
H	L	L	H	L	H	H	L	H	H	H	H	H	
H	L	L	H	H	H	H	H	L	H	H	H	H	
H	L	H	L	L	H	H	H	H	L	H	H	H	
H	L	H	L	H	H	H	H	H	H	L	H	H	
H	L	H	H	L	H	H	H	H	H	H	L	H	
H	L	H	H	H	H	H	H	H	H	H	H	L	

* G2 = G2AN + G2BN

图 4-36 74138 真值表



实验

4-1. 多路选择器设计实验

4-2. 8位加法器设计实验

4-3. 8位硬件乘法器设计实验

实验

4-4. 十六进制7段数码显示译码器设计

表 4-1 7 段译码器真值表

输入码	输出码	代表数据
0000	0111111	0
0001	0000110	1
0010	1011011	2
0011	1001111	3
0100	1100110	4
0101	1101101	5
0110	1111101	6
0111	0000111	7
1000	1111111	8
1001	1101111	9
1010	1110111	A
1011	1111100	B
1100	0111001	C
1101	1011110	D
1110	1111001	E
1111	1110001	F

【例 4-3】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DECL7S IS
    PORT ( A : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          LED7S : OUT STD_LOGIC_VECTOR(6 DOWNTO 0) ) ;
END ;
ARCHITECTURE one OF DECL7S IS
BEGIN
    PROCESS( A ) BEGIN
        CASE A IS
            WHEN "0000" => LED7S <= "0111111" ;
            WHEN "0001" => LED7S <= "0000110" ;
            WHEN "0010" => LED7S <= "1011011" ;
            WHEN "0011" => LED7S <= "1001111" ;
            WHEN "0100" => LED7S <= "1100110" ;
            WHEN "0101" => LED7S <= "1101101" ;
            WHEN "0110" => LED7S <= "1111101" ;
            WHEN "0111" => LED7S <= "0000111" ;
            WHEN "1000" => LED7S <= "1111111" ;
            WHEN "1001" => LED7S <= "1101111" ;
            WHEN "1010" => LED7S <= "1110111" ;
            WHEN "1011" => LED7S <= "1111100" ;
            WHEN "1100" => LED7S <= "0111001" ;
            WHEN "1101" => LED7S <= "1011110" ;
            WHEN "1110" => LED7S <= "1111001" ;
            WHEN "1111" => LED7S <= "1110001" ;
            WHEN OTHERS => NULL ;
        END CASE ;
    END PROCESS ;
END ARCHITECTURE one ;
```

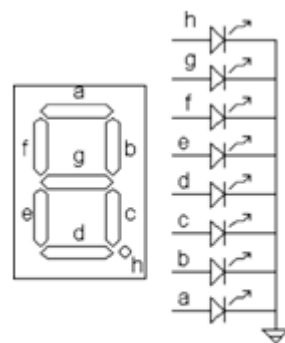


图 4-38 共阴数码管