现代计算机组成原理

潘 明　潘 松 编著

科学出版社

第 2 章

VHDL与Quartusll应用

2.1 VHDL基本语句语法

2.1.1 组合电路的VHDL描述

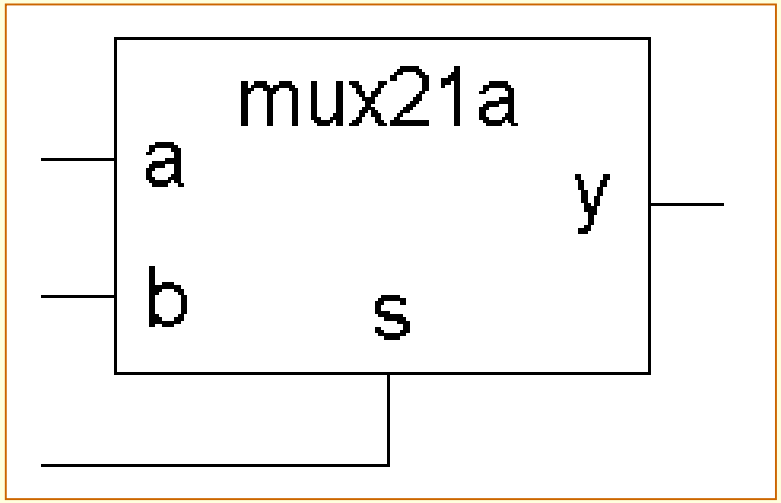


图2-1 mux21a实体

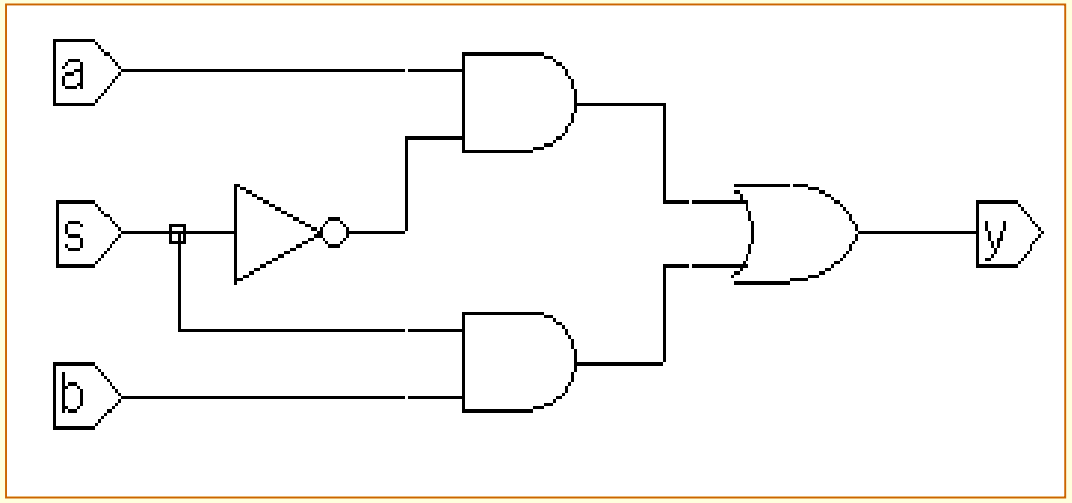


图2-2 mux21a结构体

2.1 VHDL基本语句语法

2.1.1 组合电路的VHDL描述

【例2-1】

```
ENTITY mux21a IS
  PORT ( a, b, s: IN  BIT;
                y : OUT BIT  );
END ENTITY mux21a;
ARCHITECTURE one OF mux21a IS
   BEGIN
     y <= a  WHEN  s = '0'  ELSE  b  ;
END ARCHITECTURE one ;
```

2.1 VHDL基本语句语法

2.1.1 组合电路的VHDL描述

【例2-2】

```
ENTITY mux21a IS
 PORT ( a, b, s: IN  BIT;
              y : OUT BIT  );
END ENTITY mux21a;
ARCHITECTURE one OF mux21a IS
      SIGNAL d,e :  BIT;
  BEGIN
d <= a AND (NOT S) ; e <= b AND s ;  y <= d OR e  ;
  END ARCHITECTURE one ;
```

2.1 VHDL基本语句语法

2.1.1 组合电路的VHDL描述

【例2-3】

```
 ENTITY mux21a IS
  PORT ( a, b, s: IN  BIT;
                y : OUT BIT  );
END ENTITY mux21a;
ARCHITECTURE one OF mux21a IS
 BEGIN
   PROCESS (a,b,s)
BEGIN
     IF s = '0'  THEN   y <= a ;  ELSE  y <= b ;
END IF;
   END PROCESS;
END ARCHITECTURE one ;
```

2.1 VHDL基本语句语法

2.1.1 组合电路的VHDL描述

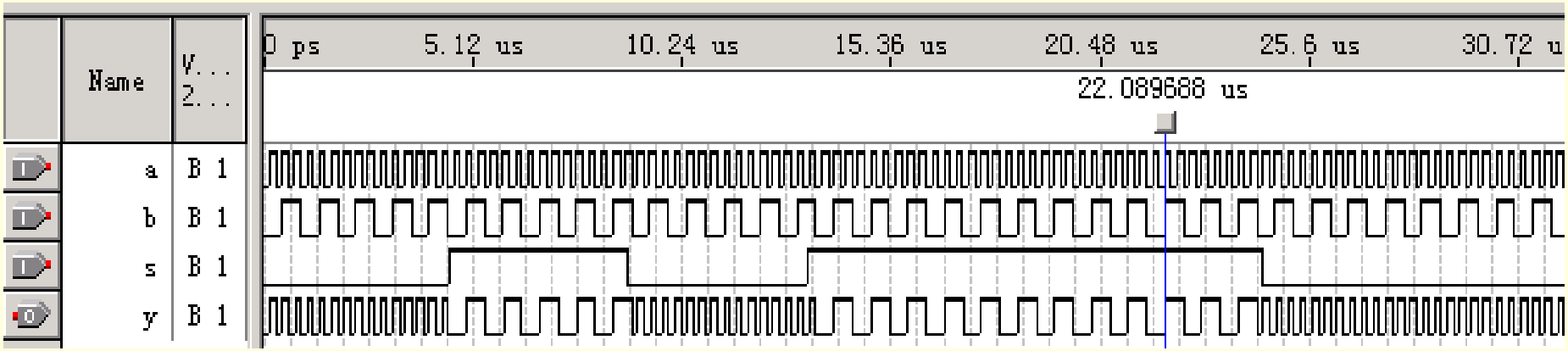


图2-3 mux21a功能时序波形

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

1. 实体表达

【例2-4】

```
ENTITY  e_name  IS
PORT ( p_name :  port_m   data_type;
              ...
            p_namei : port_mi   data_type );
END ENTITY e_name;
```

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

2. 实体名

e_name

3. 端口语句和端口信号名

PORT( ) “; ” p_name

4. 端口模式

“IN”、 “OUT”、 “INOUT”、 “BUFFER”

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

5. 数据类型 data_type

INTEGER类型、BOOLEAN类型、STD_LOGIC类型、BIT类型

6. 结构体表达

【例2-5】

```
ARCHITECTURE arch_name OF e_name IS
  [说明语句]
BEGIN
  (功能描述语句)
END ARCHITECTURE arch_name ;
```

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

7. 赋值符号和数据比较符号

y<= a

WHEN_ELSE　　s='0'

IF a THEN ...　-- 注意，a的数据类型必须是boolean

IF (s1='0')AND(s2='1')OR(c<b+1) THEN ..

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

8. 逻辑操作符

AND(与)、 OR(或)、 NAND(与非)、 NOR(或非)、
XOR(异或)、 XNOR(同或)、 NOT(取反)

逻辑操作符所要求的操作数(操作对象)的数据类型有3种：

BIT、 BOOLEAN、 STD_LOGIC。

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

9. 条件语句 **IF_THEN_ELSE**

IF语句必须以语句**“END IF；”**结束

10. WHEN_ELSE条件信号赋值语句

赋值目标 <= 表达式 **WHEN** 赋值条件 **ELSE**

表达式 **WHEN** 赋值条件 **ELSE**

…

表达式；

```
z  <= a WHEN p1 = '1' ELSE
      b WHEN p2 = '1' ELSE
           c ;
```

2.1 VHDL基本语句语法

2.1.2 VHDL语句结构

11. 进程语句和顺序语句

```
IF_THEN_ELSE_END IF;
PROCESS... END PROCESS
```

12. 文件取名和存盘

“.vhd”　　adder_f.vhd

mux21a.vhd

2.2 时序电路描述

2.2.1 D触发器描述

从VHDL的描述上看，与例2-3相比，例2-6多了4个部分：

(1) 由**LIBRARY**引导的库的说明部分。

(2) 使用了另一种数据类型：**STD_LOGIC**。

(3) 定义了一个内部节点信号：**SIGNAL**。

(4) 使用了一种新的条件判断表式：

CLK'EVENT AND CLK = '1'。

【例2-6】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF1 IS
 PORT (CLK : IN STD_LOGIC ;
       D : IN STD_LOGIC ;
       Q : OUT STD_LOGIC );
END ;
ARCHITECTURE bhv OF DFF1 IS
 SIGNAL Q1 : STD_LOGIC ;   --类似于在芯片内部定义一个数据的暂存节点
 BEGIN
  PROCESS (CLK,Q1)
   BEGIN
    IF  CLK'EVENT AND CLK = '1'
      THEN  Q1 <= D ;
    END IF;
   END PROCESS ;
Q <= Q1 ;--将内部的暂存数据向端口输出（--是注释符号）
    END bhv;
```

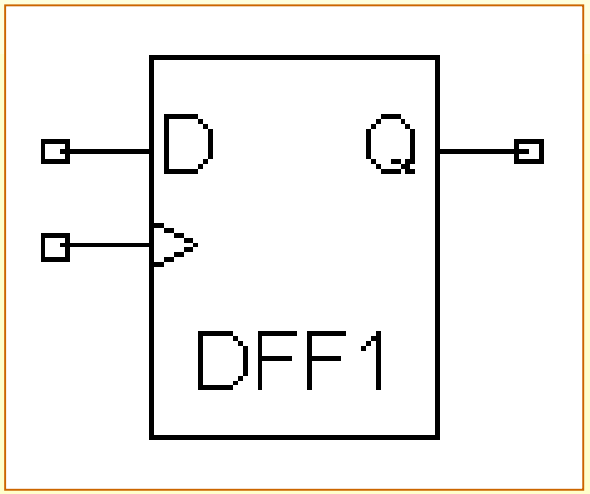


图2-4 D触发器

2.2 时序电路描述

2.2.2 时序描述相关语法规则

1. 标准逻辑位数据类型 STD_LOGIC

BIT数据类型定义：

TYPE BIT IS('0','1'); --只有两种取值

STD_LOGIC数据类型定义：

TYPE STD_LOGIC IS ('U','X','0','1','Z','W','L','H','-');

--有9种取值

2.2 时序电路描述

2.2.2 时序描述相关语法规则

2. 设计库和标准程序包

```
LIBRARY  WORK ;
LIBRARY  STD ;
USE STD.STANDARD.ALL ;
```

```
LIBRARY  <设计库名>;
    USE  < 设计库名>.<程序包名>.ALL ;
```

```
LIBRARY   IEEE ;
USE   IEEE.STD_LOGIC_1164.ALL ;
```

2.2 时序电路描述

2.2.2 时序描述相关语法规则

3. 信号定义和数据对象

```
SIGNAL Q1: STD_LOGIC;
```

4. 上升沿检测表式和信号属性函数 EVENT

```
CLK'EVENT AND CLK='1'
```

<信号名>'EVENT

```
clock'EVENT      clock'EVENT AND clock='1'
```

2.2 时序电路描述

5. 不完整条件语句与时序电路

【例2-7】

```
ENTITY COMP_BAD IS
  PORT( a1，b1  : IN BIT;
    q1  : OUT BIT    );
END ;
ARCHITECTURE one OF COMP_BAD IS
  BEGIN
   PROCESS (a1,b1)
BEGIN
IF  a1 > b1   THEN  q1 <= '1' ;
ELSIF a1 < b1 THEN  q1 <= '0' ;-- 未提及当a1=b1时，q1作何操作
END IF;
   END PROCESS ;
END ;
```

2.2 时序电路描述

5. 不完整条件语句与时序电路

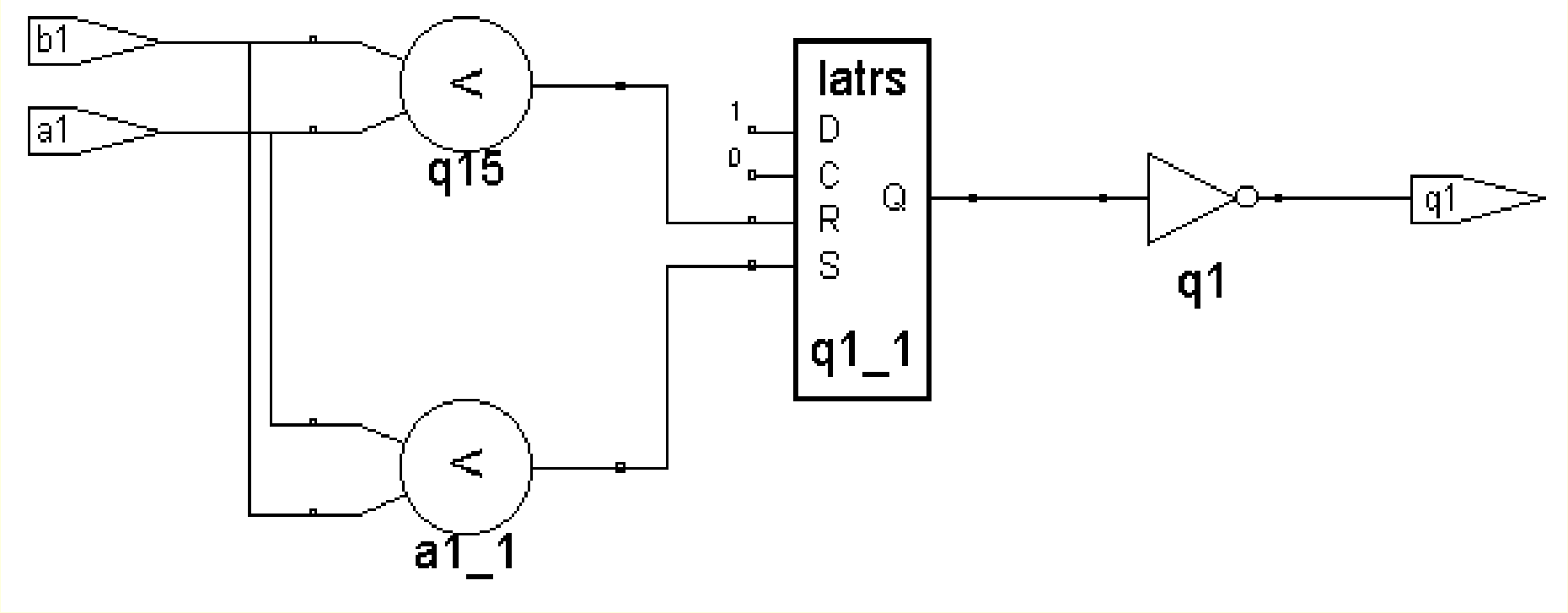


图2-5 例2-7的电路图

2.2 时序电路描述

5. 不完整条件语句与时序电路

【例2-8】

```
IF  a1 > b1  THEN  q1 <= '1' ;
ELSE   q1 <= '0' ;  END IF;
```

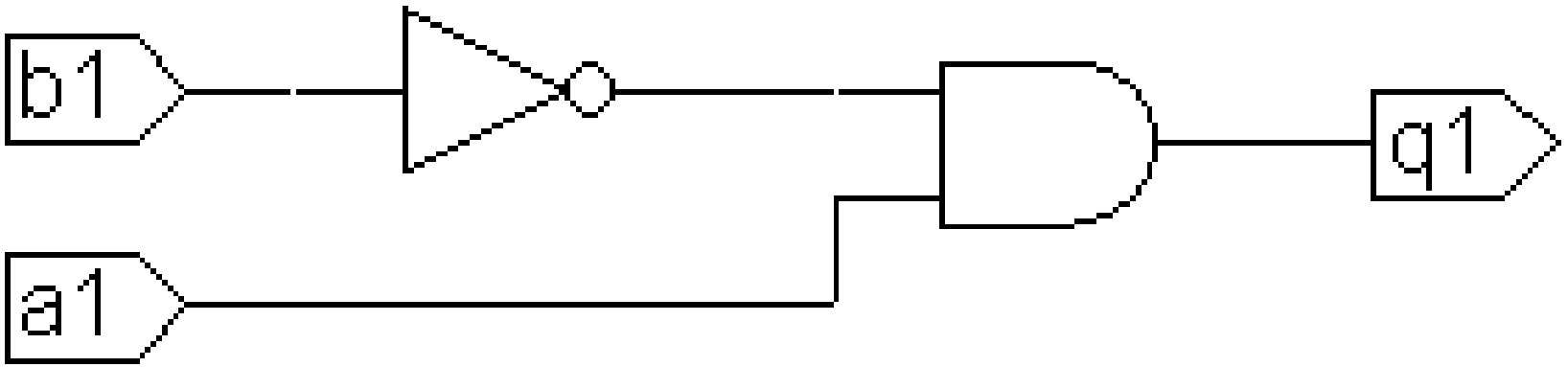


图2-6 例2-8的电路图

2.2 时序电路描述

2.2.3 实现时序电路的VHDL不同表述

【例2-9】

```
PROCESS (CLK)
  BEGIN
IF  CLK'EVENT AND (CLK='1') AND (CLK'LAST_VALUE='0')
        THEN  Q <= D ;     --确保CLK的变化是一次上升沿的跳变
     END IF;
END PROCESS ;
```

2.2 时序电路描述

2.2.3 实现时序电路的VHDL不同表述

【例2-10】

```
PROCESS (CLK)
  BEGIN
IF  CLK='1' AND CLK'LAST_VALUE='0'
        THEN  Q <= D ;
     END IF;
END PROCESS ;
```

【例2-11】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF3 IS
 PORT (CLK，  D : IN STD_LOGIC ;
        Q : OUT STD_LOGIC );
END ;
ARCHITECTURE bhv OF DFF3 IS
  SIGNAL Q1 : STD_LOGIC;
BEGIN
  PROCESS (CLK)
   BEGIN
   IF  rising_edge(CLK) -- 注意使用此函数必须打开STD_LOGIC_1164程序包
   THEN  Q1 <= D ;
   END IF;
    END PROCESS ;
   Q <= Q1 ;          --在此，赋值语句可以放在进程外，作为并行赋值语句
END ;
```

2.2 时序电路描述

2.2.3 实现时序电路的VHDL不同表述

【例2-12】

```
PROCESS
  BEGIN
    wait until CLK = '1'    ;       --利用wait语句
     Q <= D ;
END PROCESS;
```

2.2 时序电路描述

2.2.3 实现时序电路的VHDL不同表述

【例2-13】

```
PROCESS (CLK)
    BEGIN
     IF  CLK = '1'
     THEN  Q <= D ;      --利用进程的启动特性产生对CLK的边沿检测
     END IF;
END PROCESS ;
```

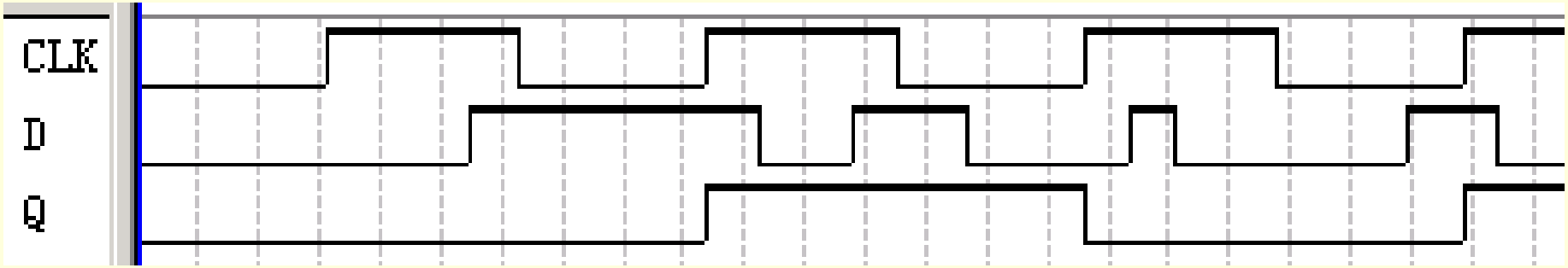


图2-7 例2-13的时序波形

2.2 时序电路描述

2.2.3 实现时序电路的VHDL不同表述

【例2-14】

```
PROCESS (CLK, D)  BEGIN
    IF  CLK = '1'                --电平触发型寄存器
    THEN  Q <= D ;
    END IF;
END PROCESS ;
```

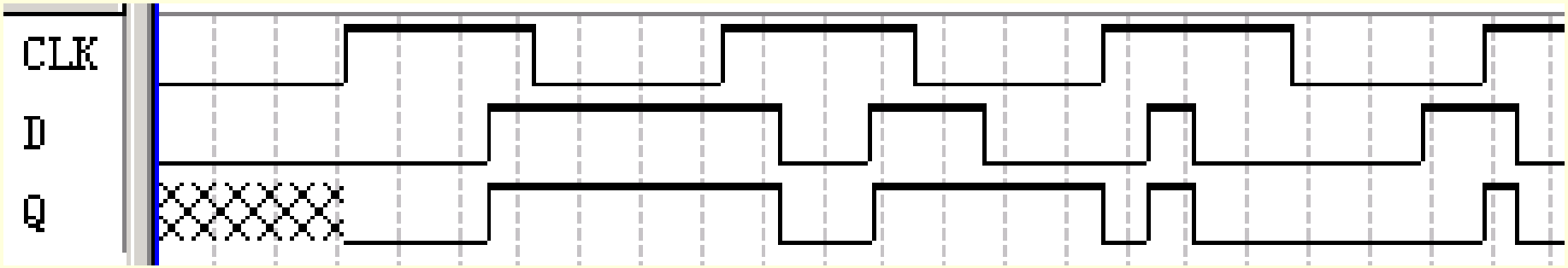


图2-8 例2-14的时序波形

2.3 全加器描述及相关语法

2.3.1 半加器描述

【例2-16】

```
LIBRARY  IEEE;     --半加器描述(1)：布尔方程描述方法
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY h_adder IS
  PORT (a, b : IN STD_LOGIC;
        co, so : OUT STD_LOGIC);
END ENTITY  h_adder;
ARCHITECTURE fh1 OF h_adder  is
BEGIN
  so <= NOT(a XOR (NOT b)) ;    co <= a AND b ;
END ARCHITECTURE fh1;
```

【例2-17】

```
LIBRARY  IEEE;  --半加器描述(2)：真值表描述方法
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY h_adder IS
PORT (a, b : IN STD_LOGIC;
   co, so : OUT STD_LOGIC);
END ENTITY h_adder;
ARCHITECTURE fh1 OF h_adder  is
SIGNAL abc : STD_LOGIC_VECTOR(1 DOWNTO 0) ; --定义标准逻辑位矢量数据类型
BEGIN
 abc <= a & b ;  --a相并b，即a与b并置操作
PROCESS(abc)
 BEGIN
  CASE abc IS     --类似于真值表的CASE语句
   WHEN "00" => so<='0'; co<='0' ;
   WHEN "01" => so<='1'; co<='0' ;
   WHEN "10" => so<='1'; co<='0' ;
   WHEN "11" => so<='0'; co<='1' ;
   WHEN OTHERS => NULL ;
  END CASE;
END PROCESS;
END ARCHITECTURE fh1 ;
```

2.3 全加器描述及相关语法

2.3.1 半加器描述

【例2-18】

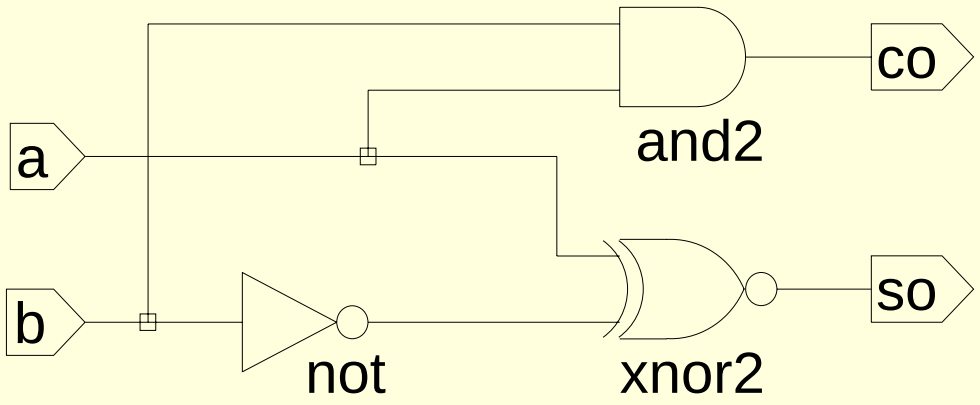
```
LIBRARY  IEEE ;    --或门逻辑描述
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY or2a IS
  PORT (a, b :IN STD_LOGIC;  c : OUT STD_LOGIC );
END ENTITY or2a;
ARCHITECTURE one OF or2a IS
  BEGIN
  c <= a OR b ;
END ARCHITECTURE one ;
```

【例2-19】

```
LIBRARY  IEEE;  --1位二进制全加器顶层设计描述
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY f_adder IS
  PORT (ain，bin，cin  : IN STD_LOGIC;
          cout，sum   : OUT STD_LOGIC );
END ENTITY f_adder;
ARCHITECTURE fd1 OF f_adder IS
  COMPONENT h_adder                --调用半加器声明语句
    PORT (  a，b :  IN STD_LOGIC;
      co，so :  OUT STD_LOGIC);
  END COMPONENT ;
  COMPONENT or2a
     PORT (a，b : IN STD_LOGIC;
             c : OUT STD_LOGIC);
  END COMPONENT;
SIGNAL d，e，f  :  STD_LOGIC; --定义3个信号作为内部的连接线。
 BEGIN
  u1 : h_adder PORT MAP(a=>ain，b=>bin，co=>d，so=>e);  --例化语句
  u2 : h_adder PORT MAP(a=>e，  b=>cin，  co=>f，so=>sum);
  u3 :  or2a  PORT MAP(a=>d，  b=>f，    c=>cout);
END ARCHITECTURE fd1;
```

2.3 全加器描述及相关语法

2.3.1 半加器描述



a	b	so	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

图2-10 半加器h_adder电路图及其真值表

2.3 全加器描述及相关语法

2.3.1 半加器描述

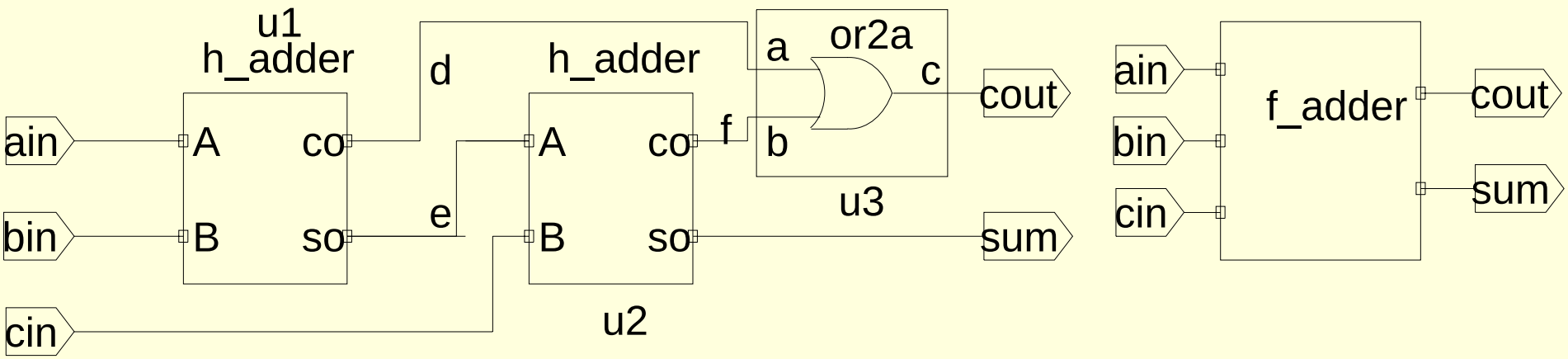


图2-11 全加器f_adder电路图及其实体模块

2.3 全加器描述及相关语法

2.3.2 CASE语句

1. CASE语句

一般表述

CASE <表达式> **IS**

When <选择值或标识符> => <顺序语句>; ... ; <顺序语句> ;

When <选择值或标识符> => <顺序语句>; ... ; <顺序语句> ;

...

WHEN OTHERS => <顺序语句>;

END CASE ;

2.3 全加器描述及相关语法

2.3.2 CASE语句

2. 标准逻辑矢量数据类型

STD_LOGIC_VECTOR

```
B : OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) ;
```

或 `SIGNAL A : STD_LOGIC_VECTOR(1 TO 4)`

```
B <= "01100010" ;                -- B(7)为 '0'
B(4 DOWNTO 1) <= "1101" ;        -- B(4)为 '1'
B(7 DOWNTO 4) <= A ;             -- B(6)等于 A(2)
```

```
SIGNAL C : BIT_VECTOR(3 DOWNTO 0);
```

2.3 全加器描述及相关语法

2.3.2 CASE语句

3. 并置操作符 &

```
SIGNAL a : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
SIGNAL d : STD_LOGIC_VECTOR (1 DOWNTO 0) ;
...
a  <= '1' & '0' & d(1) & '1'  ;  -- 元素与元素并置，并置后的数组长度为4
...
IF a & d = "101011"  THEN ... -- 在IF条件句中可以使用并置符
```

2.3 全加器描述及相关语法

2.3.3 例化语句

```
COMPONENT 元件名 IS
    PORT  (端口名表) ;
END COMPONENT 文件名 ;
```

```
COMPONENT h_adder
    PORT (  c, d :  IN STD_LOGIC;
            e, f :  OUT STD_LOGIC);
```

例化名：元件名 PORT MAP([端口名 =>] 连接端口名,...);

2.4 计数器设计及相关语法

【例2-20】

```
ENTITY CNT4 IS
  PORT ( CLK : IN BIT ;
           Q  : BUFFER INTEGER RANGE 15 DOWNTO 0  ) ;
  END ;
ARCHITECTURE bhv OF CNT4 IS
  BEGIN
  PROCESS (CLK)
    BEGIN
      IF  CLK'EVENT AND CLK = '1'  THEN
        Q <= Q + 1 ;
      END IF;
  END PROCESS ;
END bhv;
```

2.4 计数器设计及相关语法

1．程序说明

Q<=Q + 1

2．数据类型说明

1　　十进制整数

0　　十进制整数

35　　十进制整数

10E3　十进制整数，等于十进制整数**1000**

16#D9#　　十六进制整数，等于十六进制整数**D9H**

8#720#　　八进制整数，等于八进制整数**720O**

2#11010010#　　二进制整数，等于二进制整数**11010010B**

2.4 计数器设计及相关语法

【例2-21】

3．计数器的其他表述方法

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY CNT4 IS
PORT ( CLK : IN STD_LOGIC ;
          Q  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)  ) ;
END ;
ARCHITECTURE bhv OF CNT4 IS
SIGNAL Q1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
  PROCESS (CLK)
  BEGIN
      IF  CLK'EVENT AND CLK = '1'  THEN  Q1 <= Q1 + 1 ;
      END IF;
  END PROCESS ;
      Q <= Q1 ;
END bhv;
```

2.5 一般计数器设计

【例2-22】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT10 IS
   PORT (CLK,RST,EN : IN STD_LOGIC;
                CQ : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
COUT : OUT STD_LOGIC  );
END CNT10;
ARCHITECTURE behav OF CNT10 IS
BEGIN
  PROCESS(CLK, RST, EN)
    VARIABLE  CQI : STD_LOGIC_VECTOR(3 DOWNTO 0);
  BEGIN
    IF RST = '1' THEN   CQI := (OTHERS =>'0') ;  --计数器异步复位
```

(接下页）

2.5 一般计数器设计

(接上页)

```
ELSIF CLK'EVENT AND CLK='1' THEN          --检测时钟上升沿
      IF EN = '1' THEN                    --检测是否允许计数（同步使能）
       IF CQI < 9 THEN   CQI := CQI + 1;       --允许计数, 检测是否小于9
         ELSE    CQI := (OTHERS =>'0'); END IF;  --大于9，计数值清零
      END IF;
    END IF;
     IF CQI = 9 THEN COUT <= '1';             --计数大于9，输出进位信号
      ELSE    COUT <= '0';  END IF;
      CQ <= CQI;      --将计数值向端口输出
  END PROCESS;
END behav;
```

2.5 一般计数器设计

2.5.1 相关语法说明

1. 变量

```
VARIABLE  CQI : STD_LOGIC_VECTOR(3 DOWNTO 0)
```

2. 省略赋值操作符 (OTHERS=>X)

```
SIGNAL   d1 : STD_LOGIC_VECTOR(4 DOWNTO 0);
VARIABLE a1 : STD_LOGIC_VECTOR(15 DOWNTO 0);
   ...
  d1 <= (OTHERS=>'0');   a1 := (OTHERS=>'0') ;
```

```
d1 <= (1=>e(3),3=>e(5), OTHERS=>e(1) );
```

```
f <= e(1) & e(5) & e(1) & e(3) & e(1) ;
```

2.5 一般计数器设计

2.5.2 程序功能和语法分析

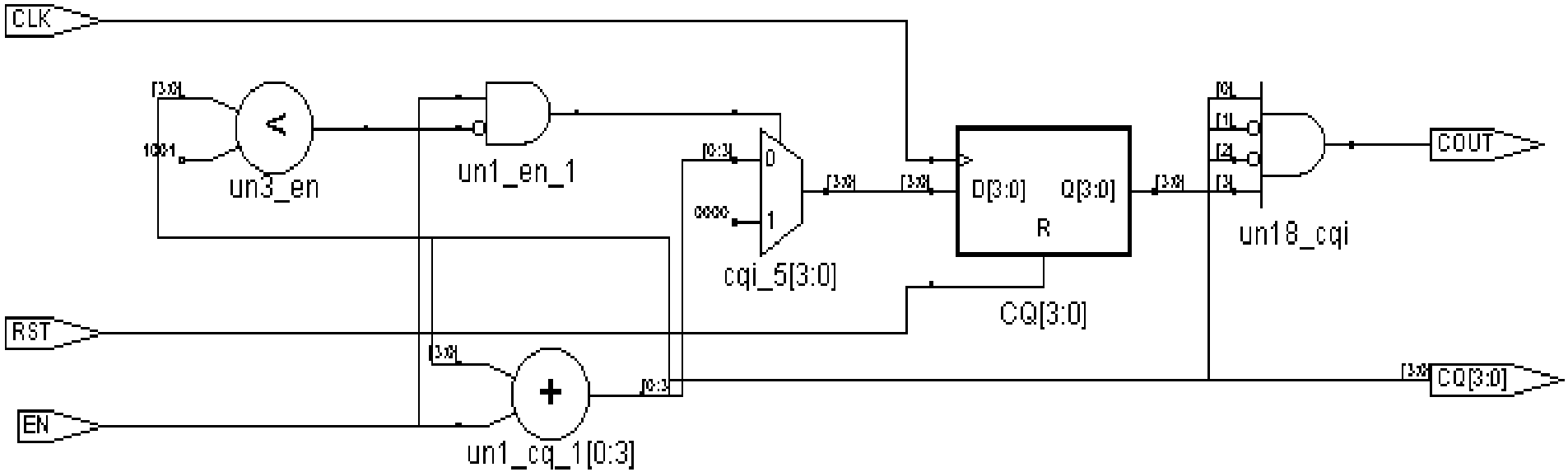


图2-12 例2-22的RTL电路

2.5 一般计数器设计

2.5.2 程序功能和语法分析

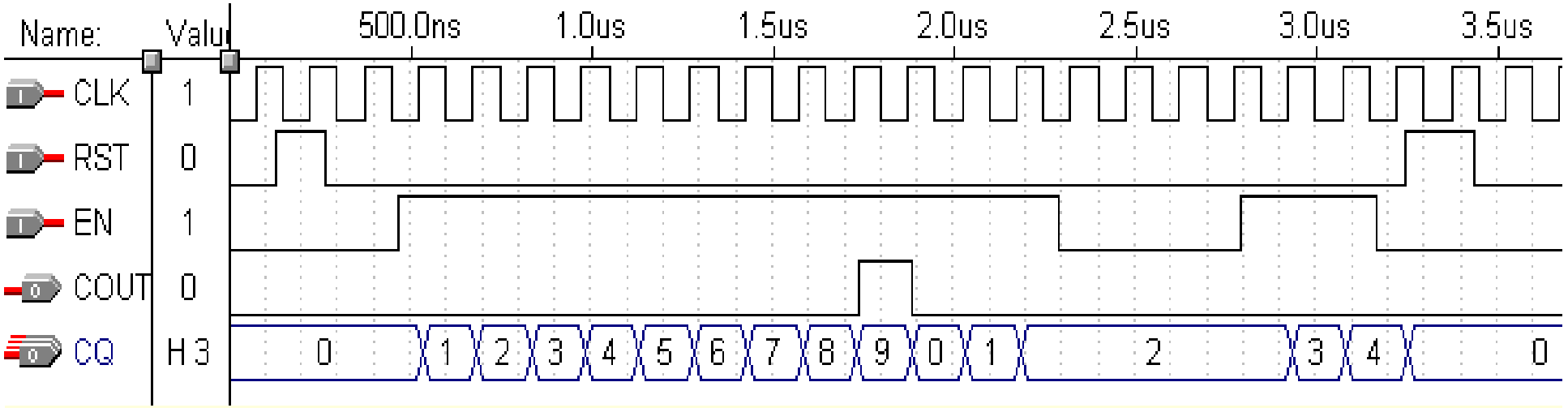


图2-13 例2-22的工作时序

2.5 一般计数器设计

2.5.3 含并行置位的移位寄存器设计

【例2-23】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SHFRT IS                -- 8位右移寄存器
   PORT (  CLK，  LOAD : IN STD_LOGIC;
                DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                 QB : OUT STD_LOGIC  );
END SHFRT;
ARCHITECTURE behav OF SHFRT IS
   BEGIN
   PROCESS (CLK, LOAD)
    VARIABLE REG8 : STD_LOGIC_VECTOR(7 DOWNTO 0);
   BEGIN
      IF CLK'EVENT AND CLK = '1' THEN
        IF LOAD = '1' THEN  REG8 := DIN;  --由（LOAD='1'）装载新数据
          ELSE  REG8(6 DOWNTO 0) := REG8(7 DOWNTO 1); END IF;
      END IF;
        QB <= REG8(0); -- 输出最低位
   END PROCESS;
END behav;
```

2.5 一般计数器设计

2.5.3 含并行置位的移位寄存器设计

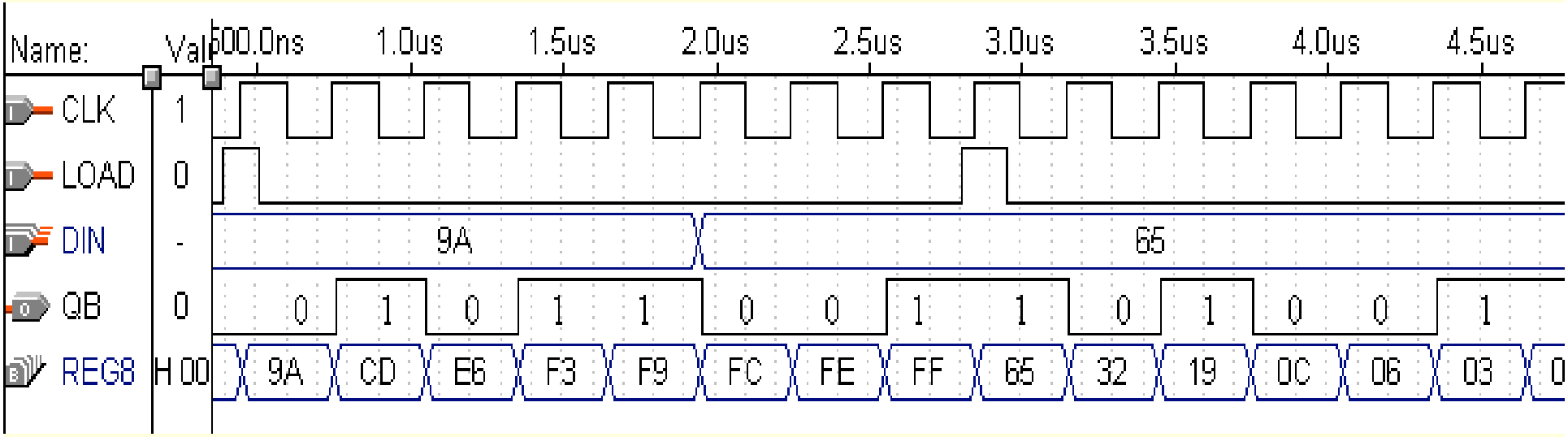


图2-14 例2-23的工作时序

2.6 QuartusII6.0使用向导

2.6.1 建立工作库文件夹和编辑设计文件

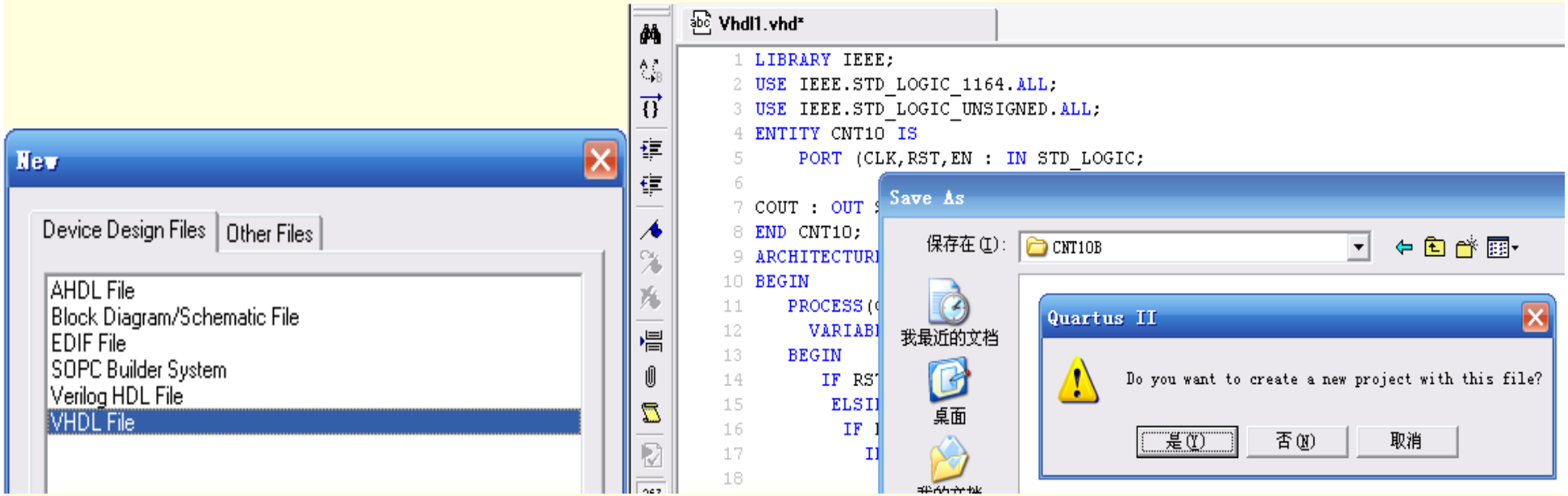


图2-15 选择编辑文件的语言类型，键入源程序并存盘

2.6 QuartusII6.0使用向导

2.6.2 创建工程

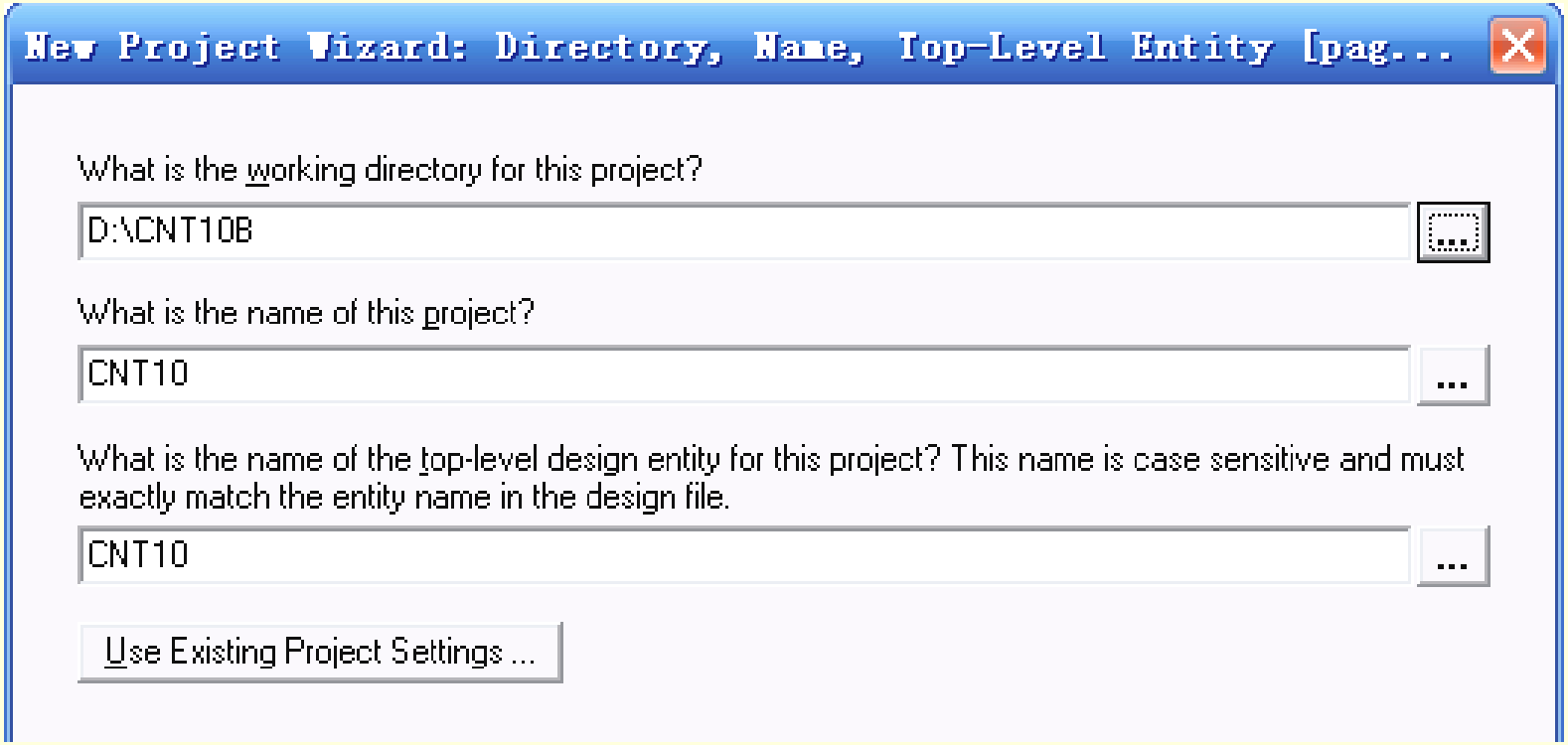


图2-16 利用“New Preject Wizard”创建工程cnt10

2.6 QuartusII6.0使用向导

2.6.2 创建工程

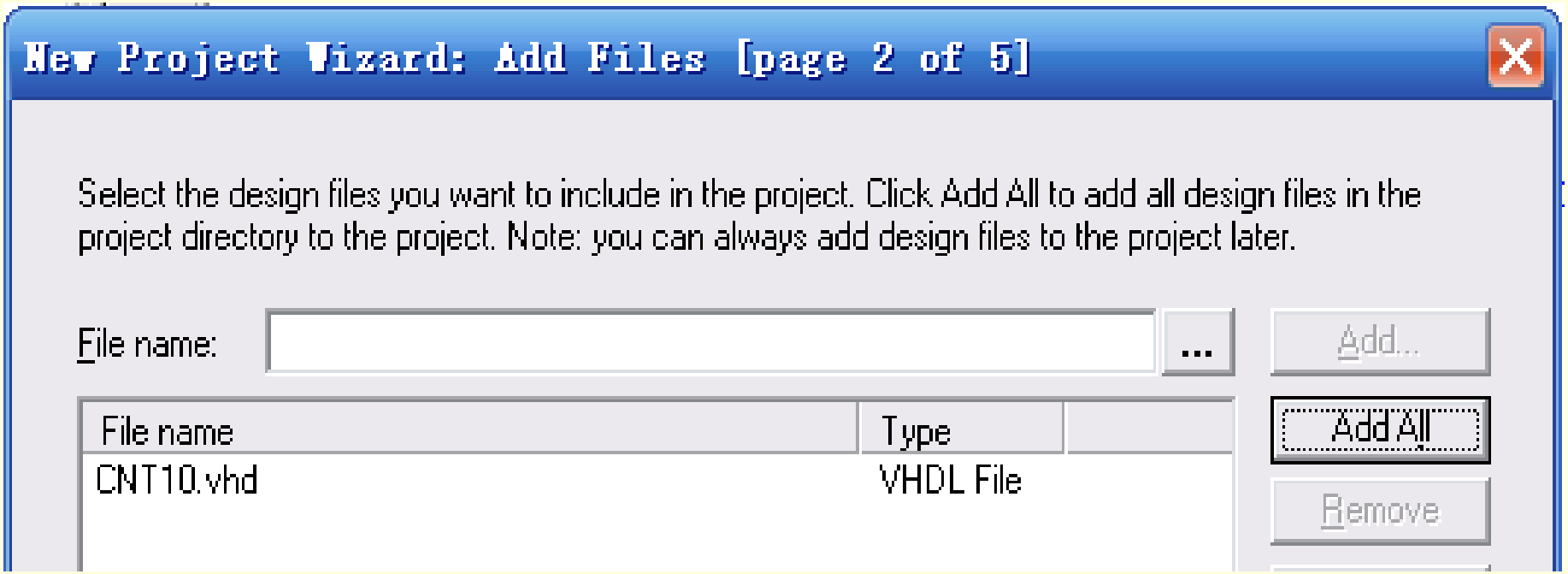


图2-17 将所有相关的文件都加入进此工程

2.6 QuartusII6.0使用向导

2.6.2 创建工程

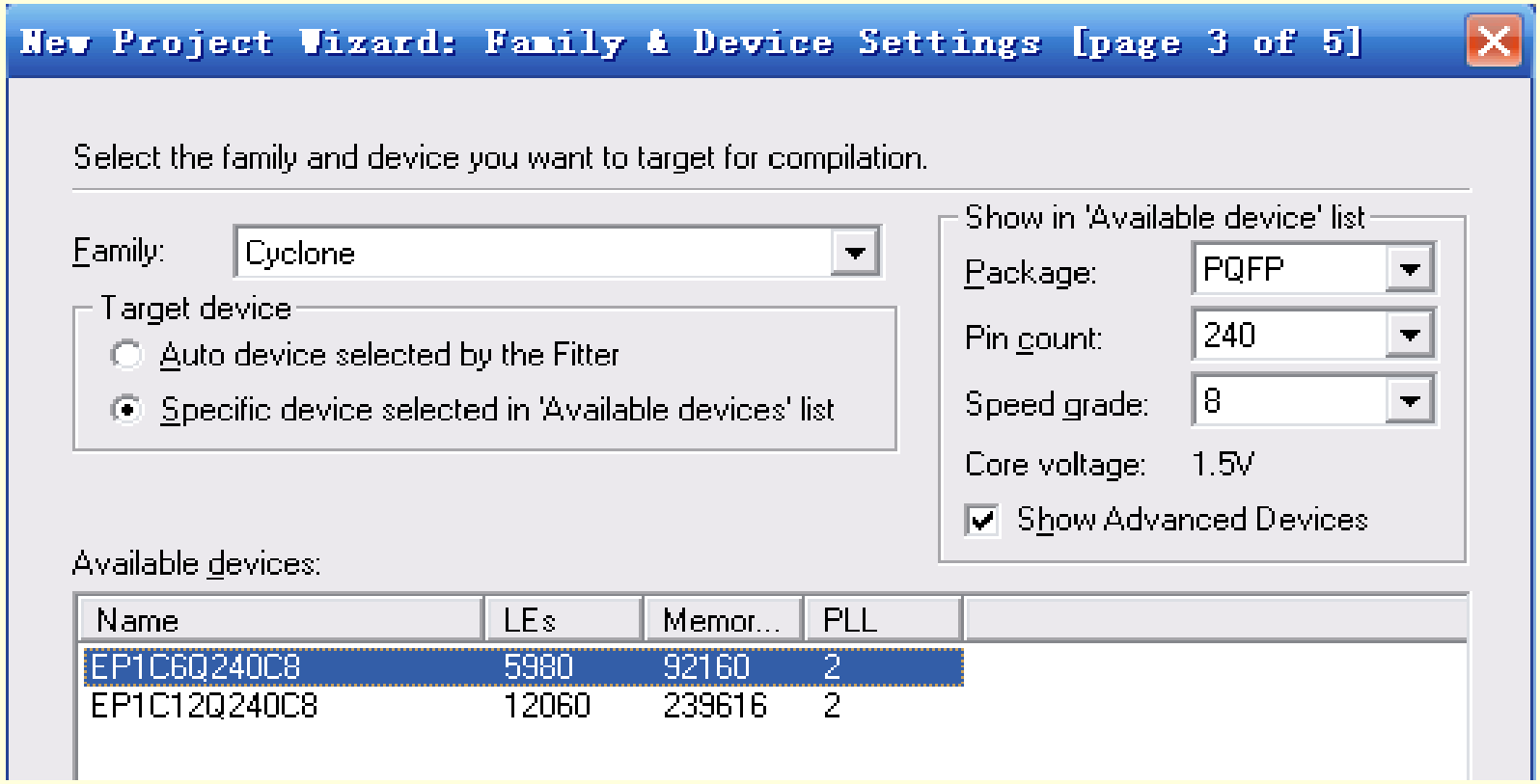


图2-18 选择目标器件EP1C6Q240C8

2.6 QuartusII6.0使用向导

2.6.3 编译前设置

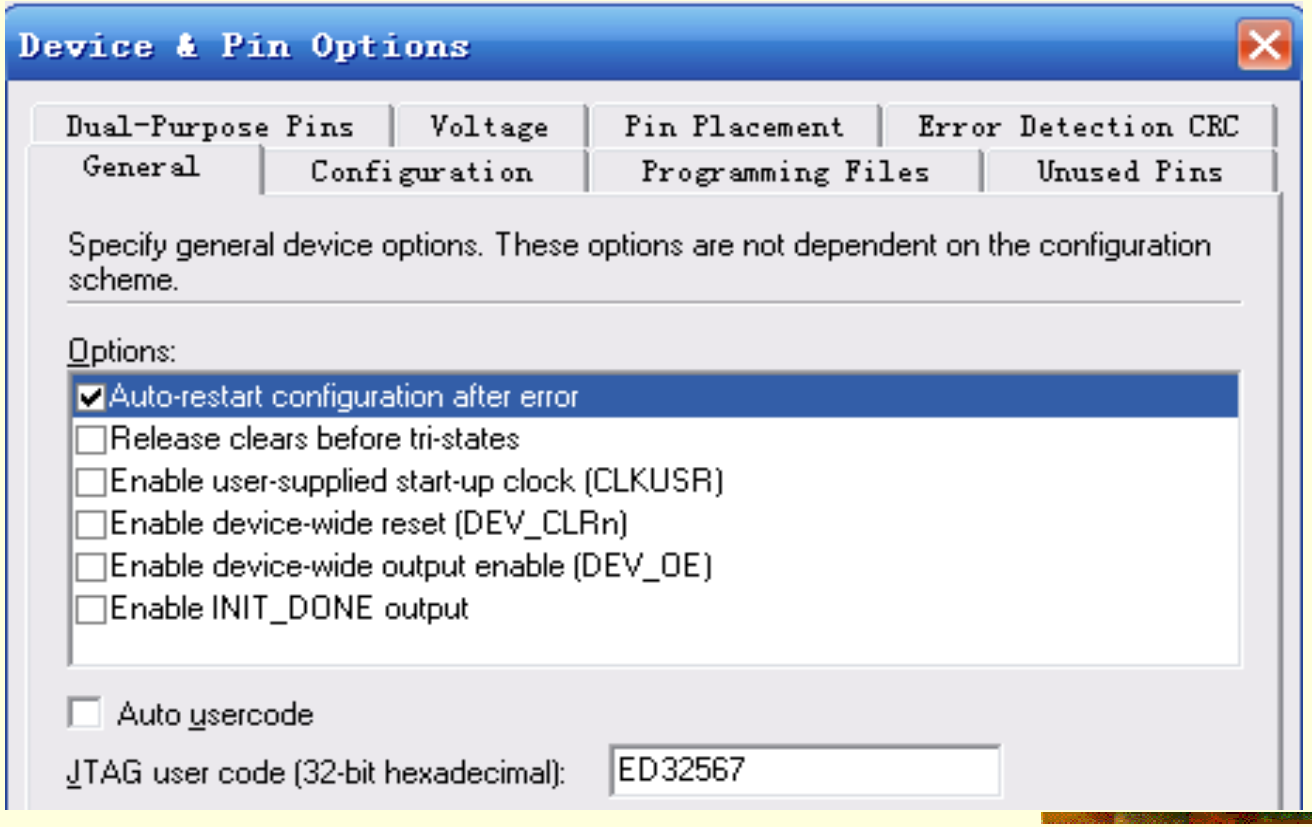


图2-19选择配置器件的工作方式

2.6 QuartusII6.0使用向导

2.6.3 编译前设置

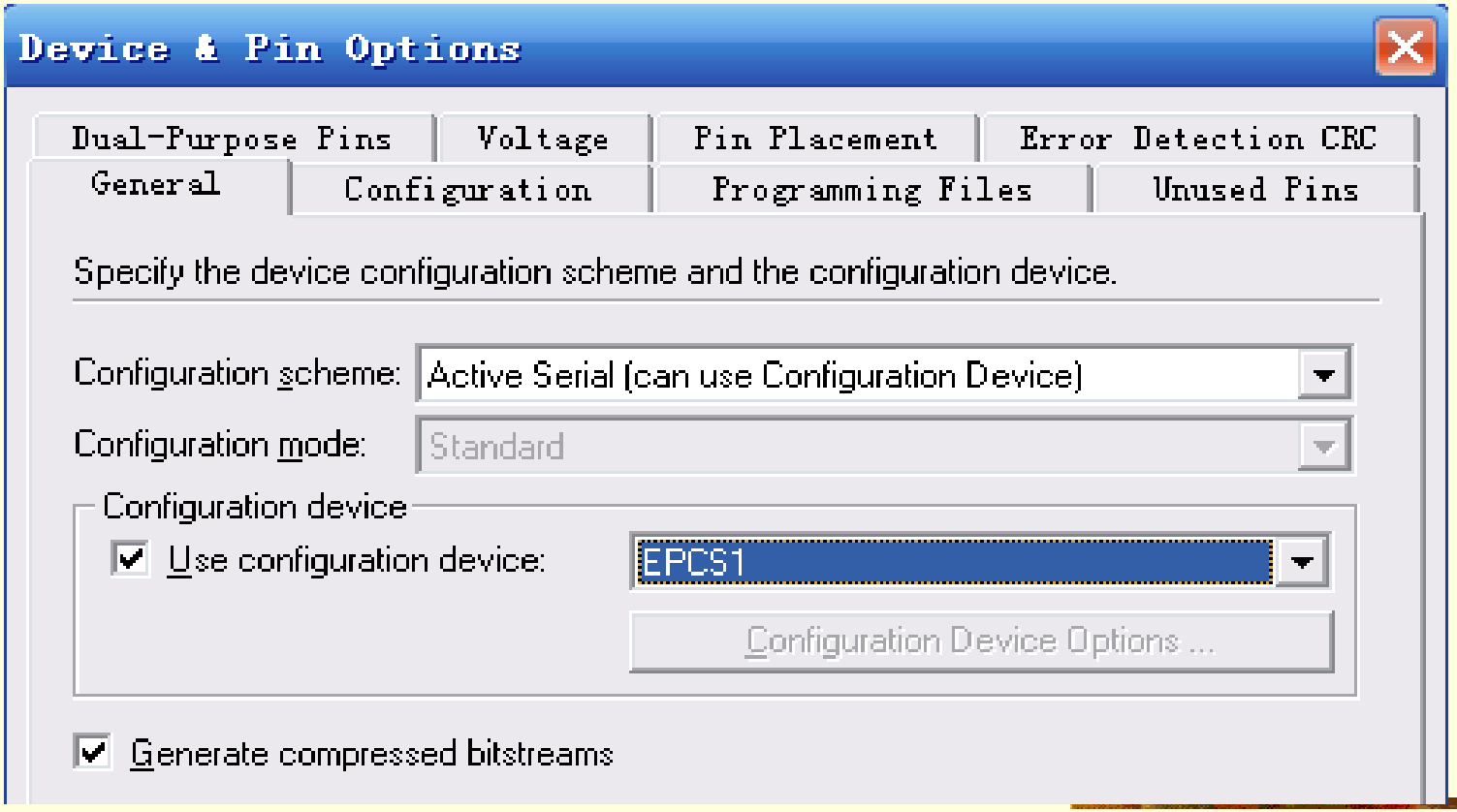


图2-20 选择配置器件和编程方式

2.6.4 全程编译

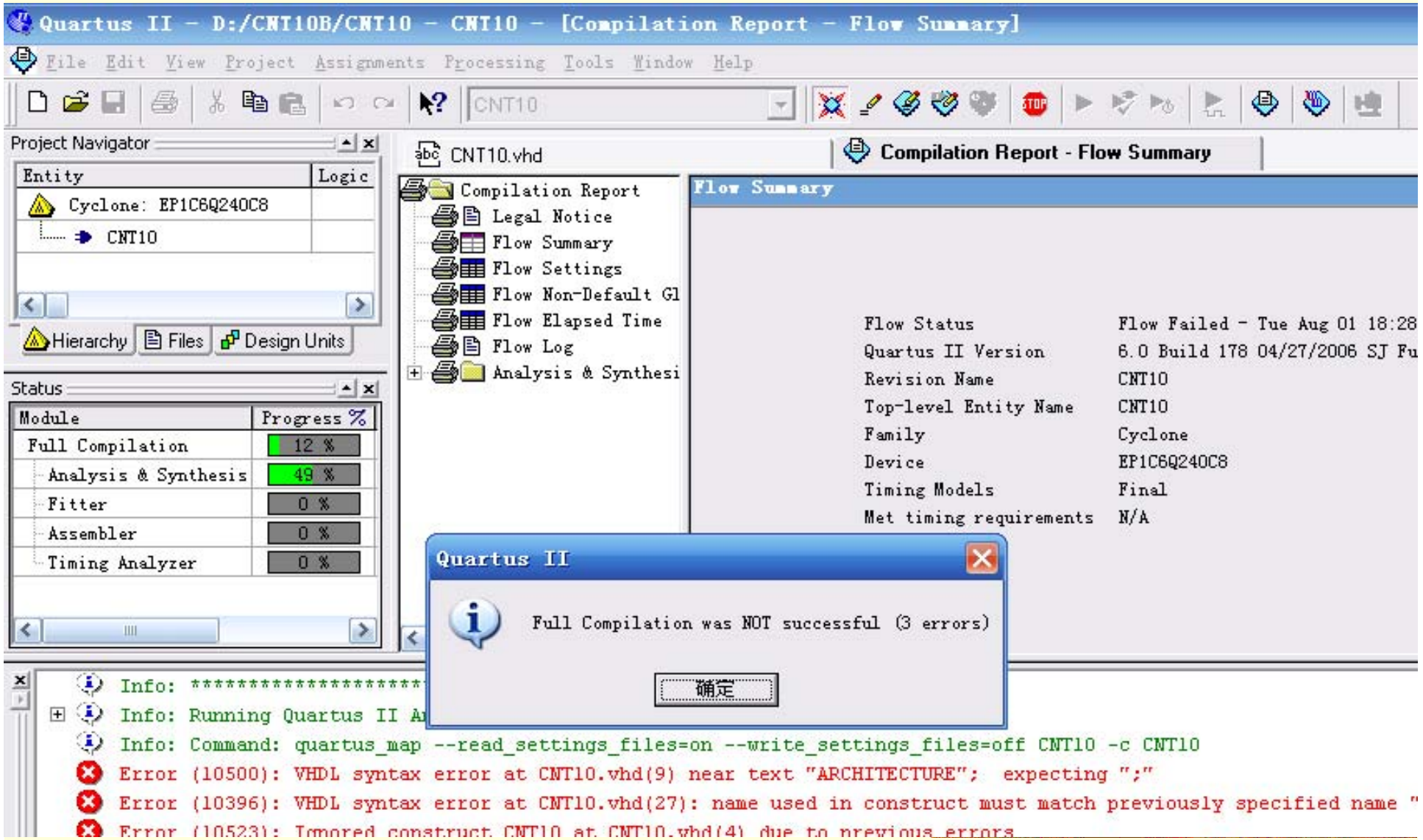


图2-21 全程编译后出现报错信息

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

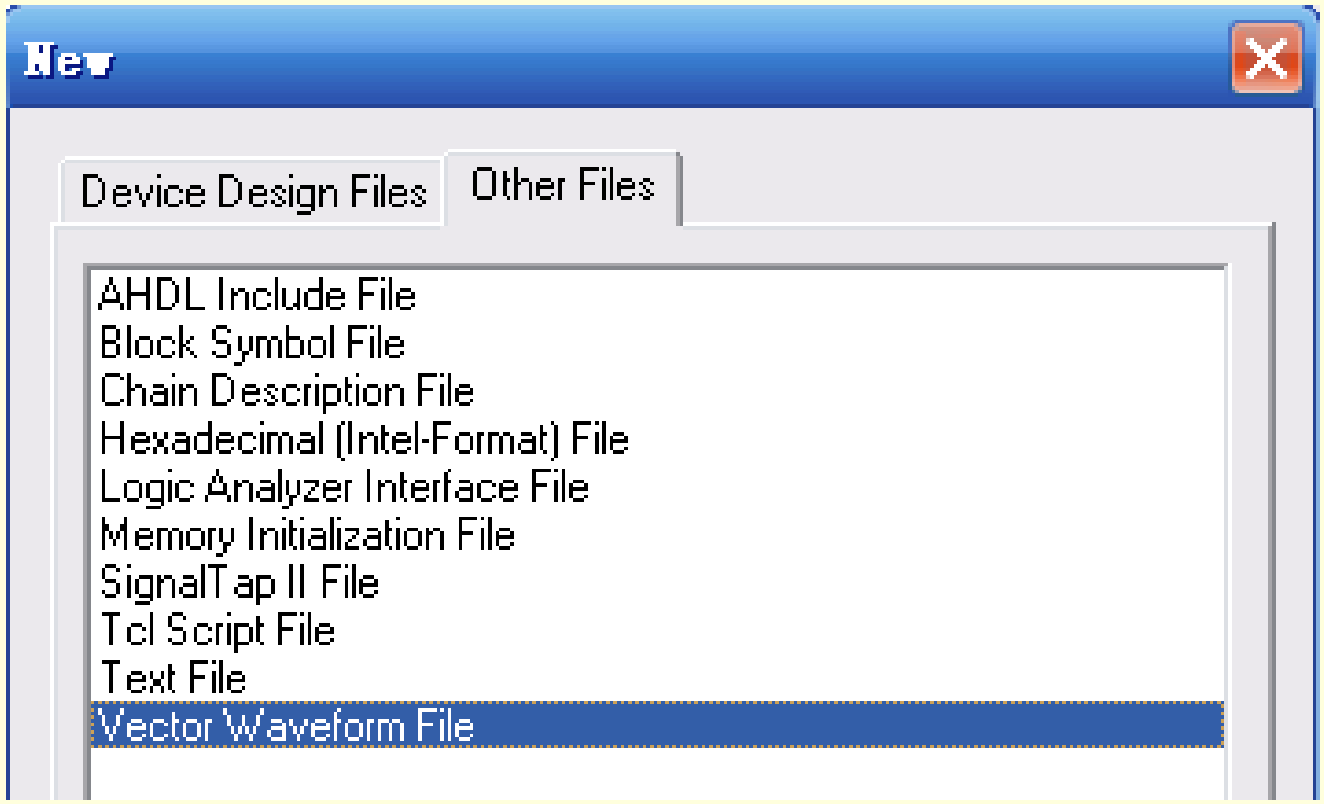


图2-22 选择编辑矢量波形文件

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

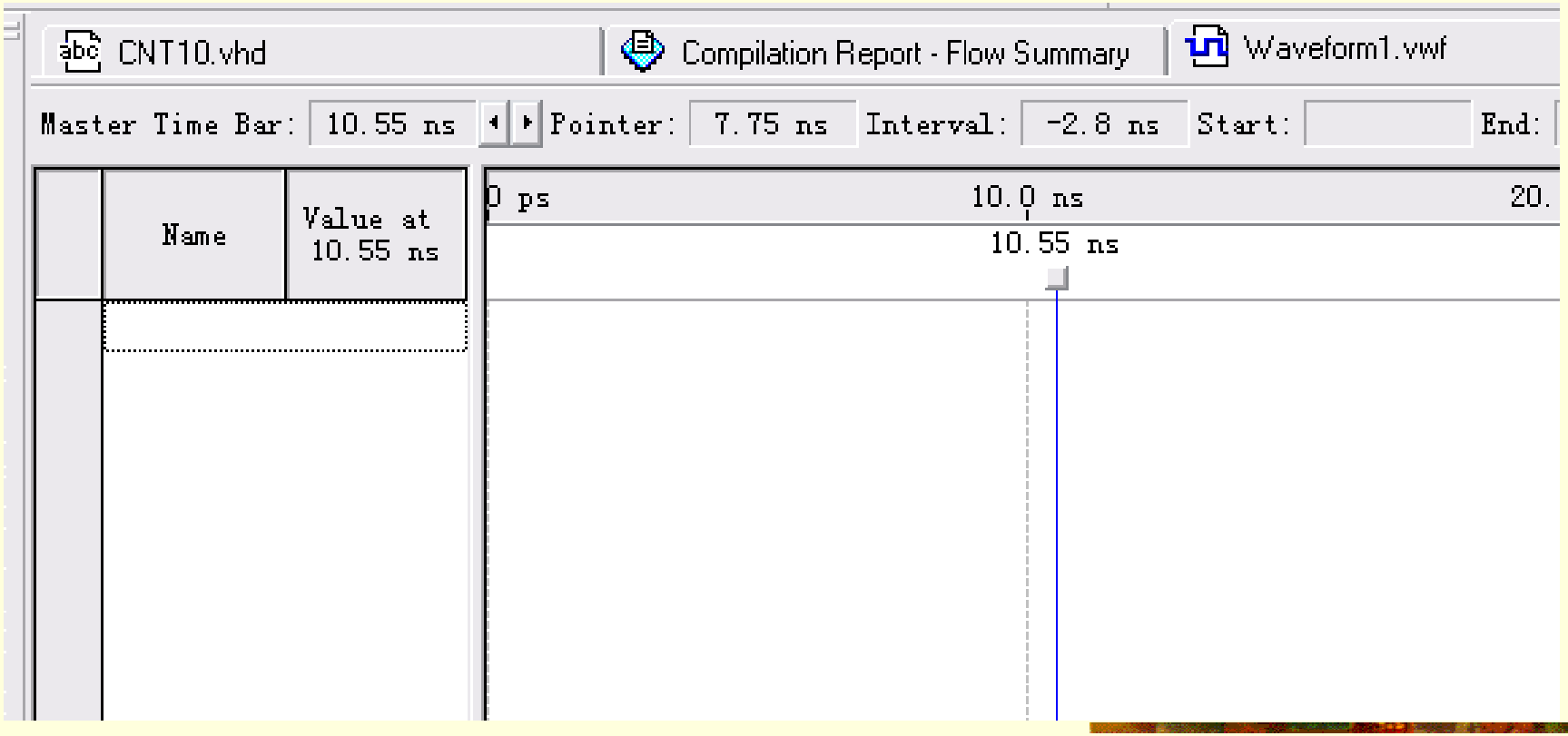


图2-23 波形编辑器

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

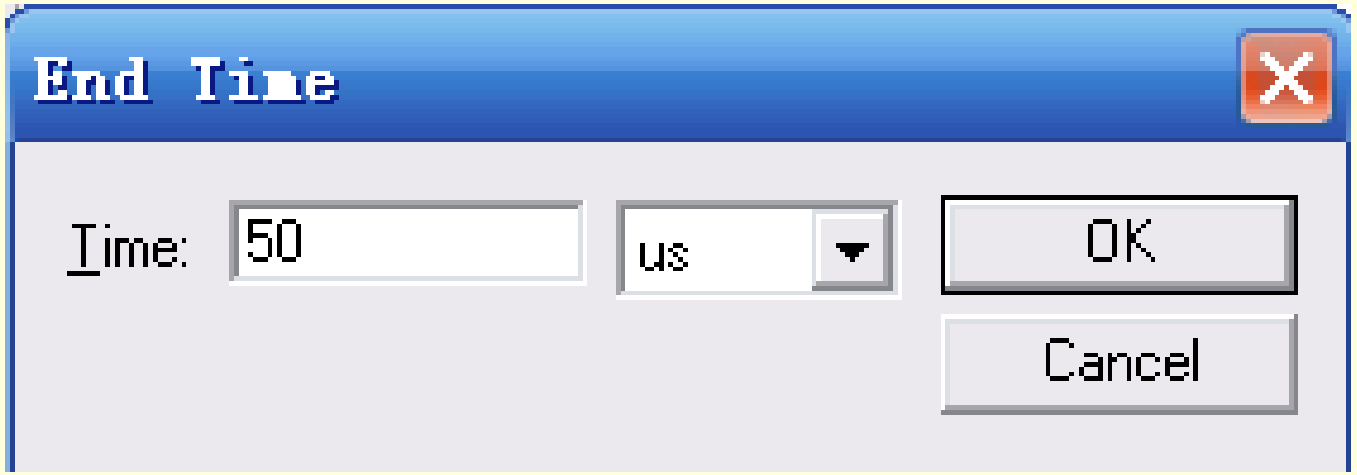


图2-24 设置仿真时间长度

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

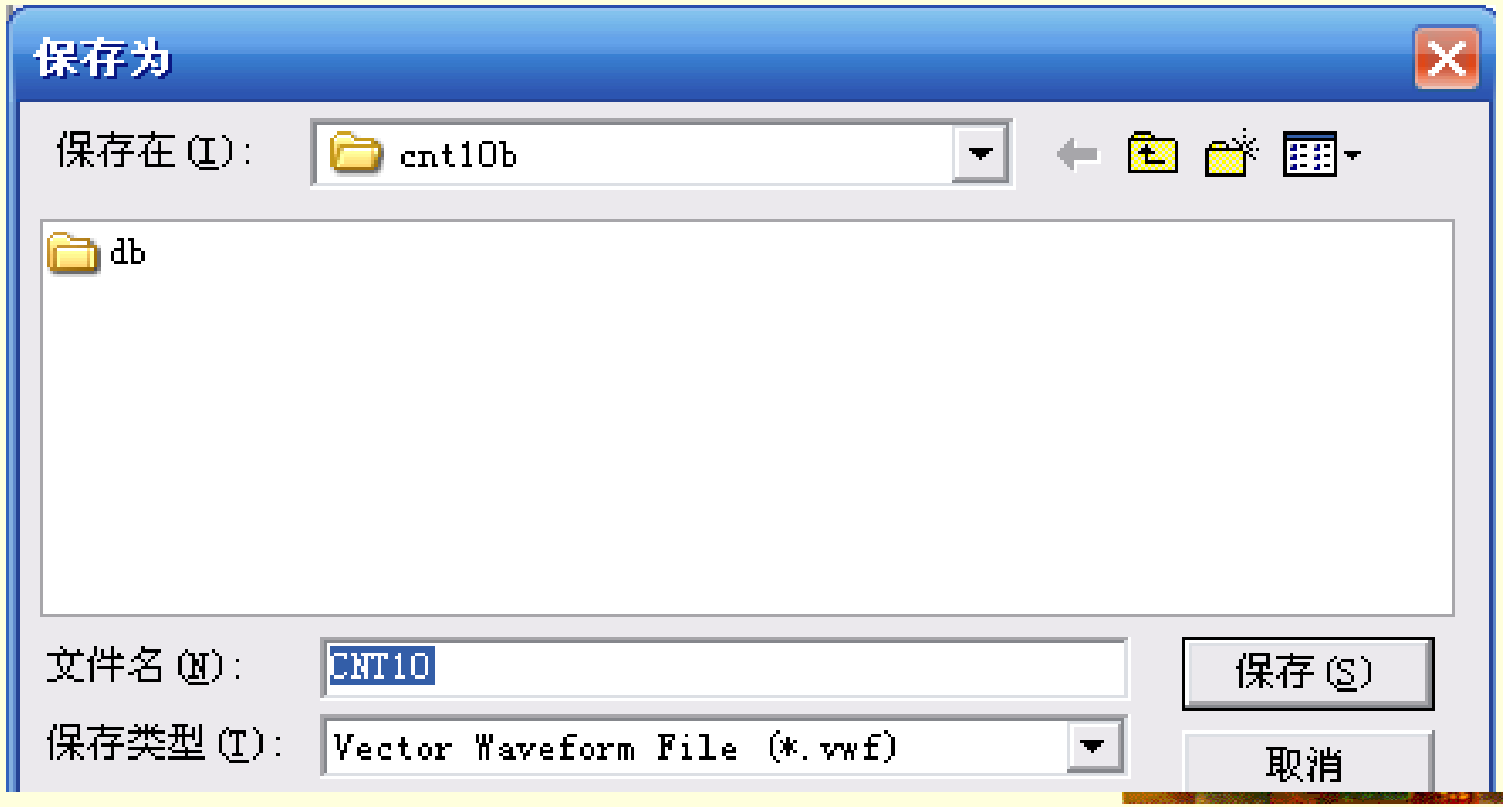


图2-25 vwf激励波形文件存盘

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

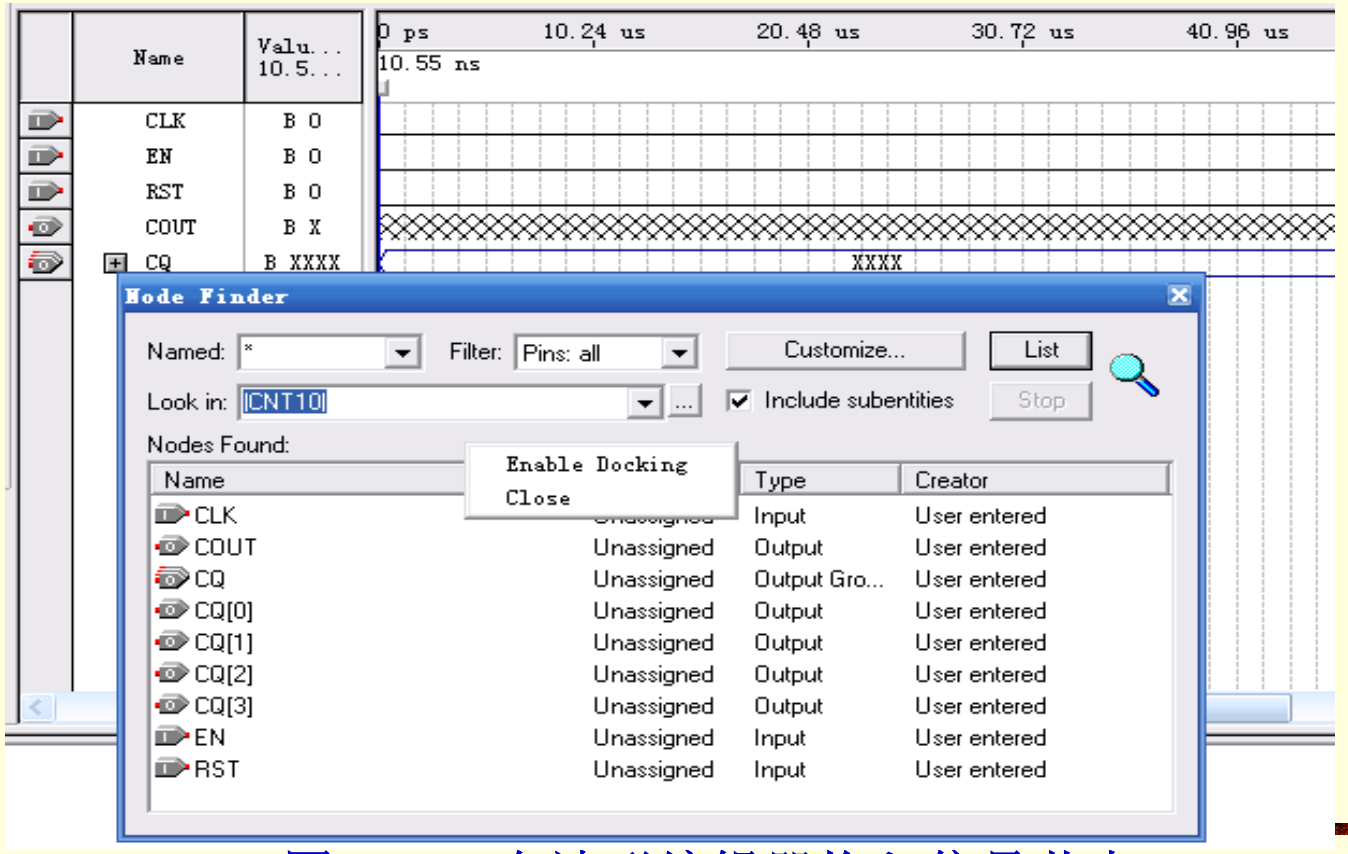


图2-26 向波形编辑器拖入信号节点

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

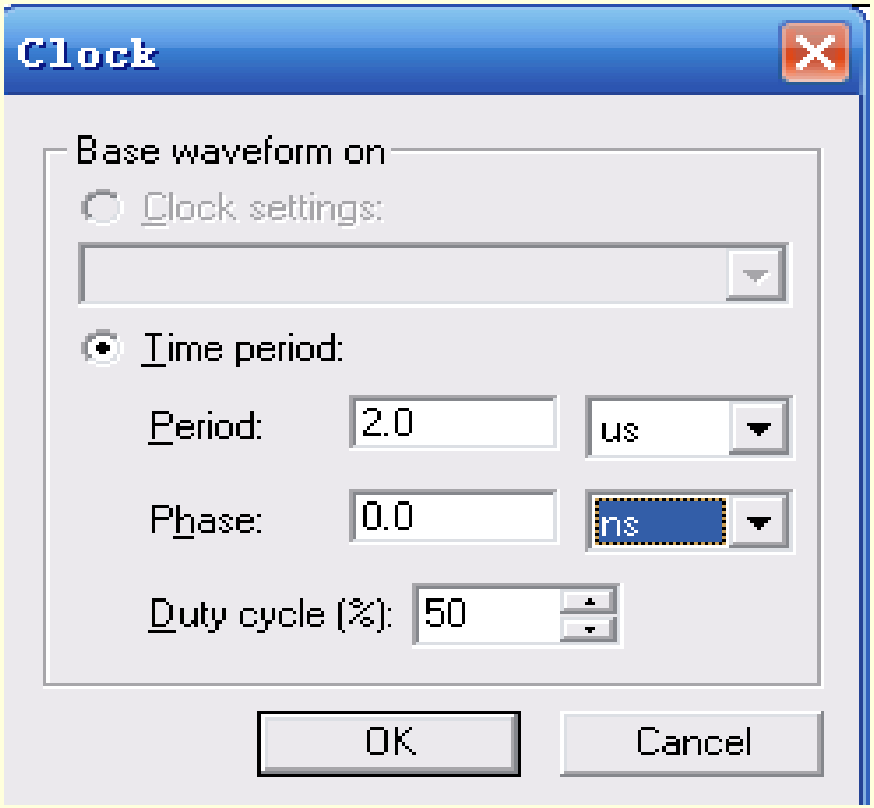


图2-27 设置时钟CLK的周期

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

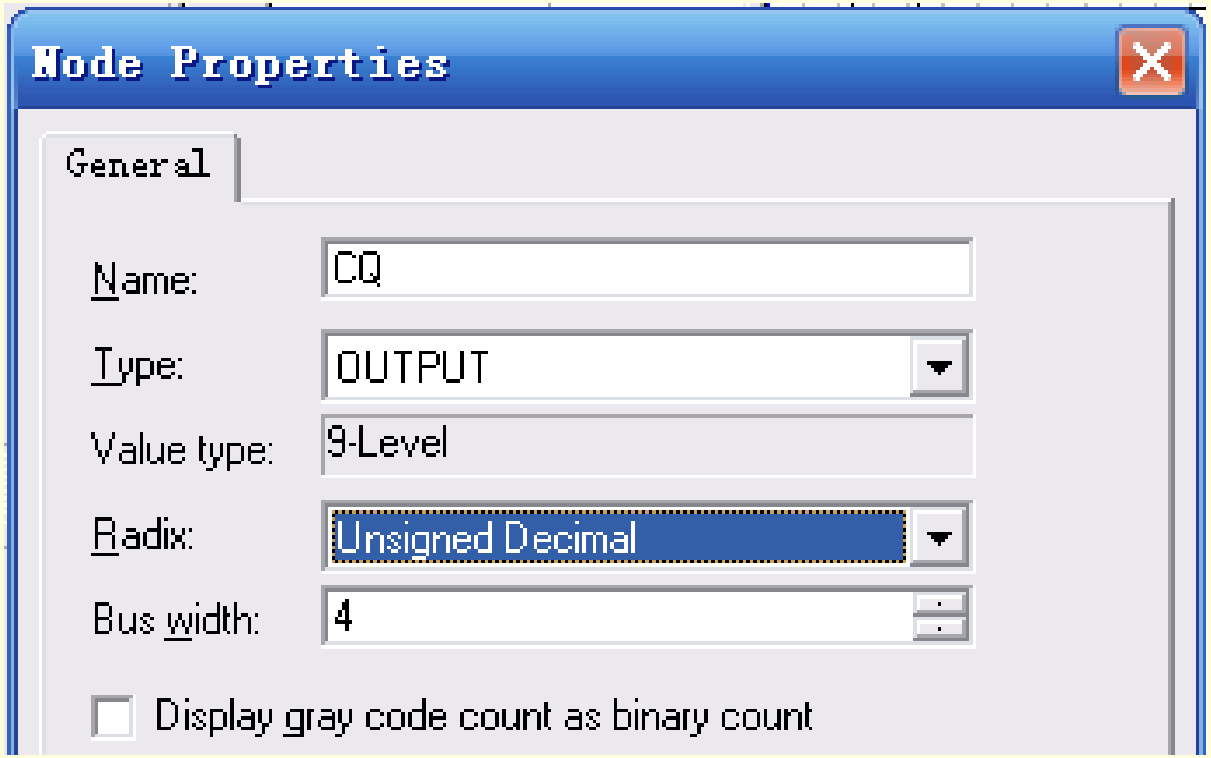


图2-28 选择总线数据格式

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

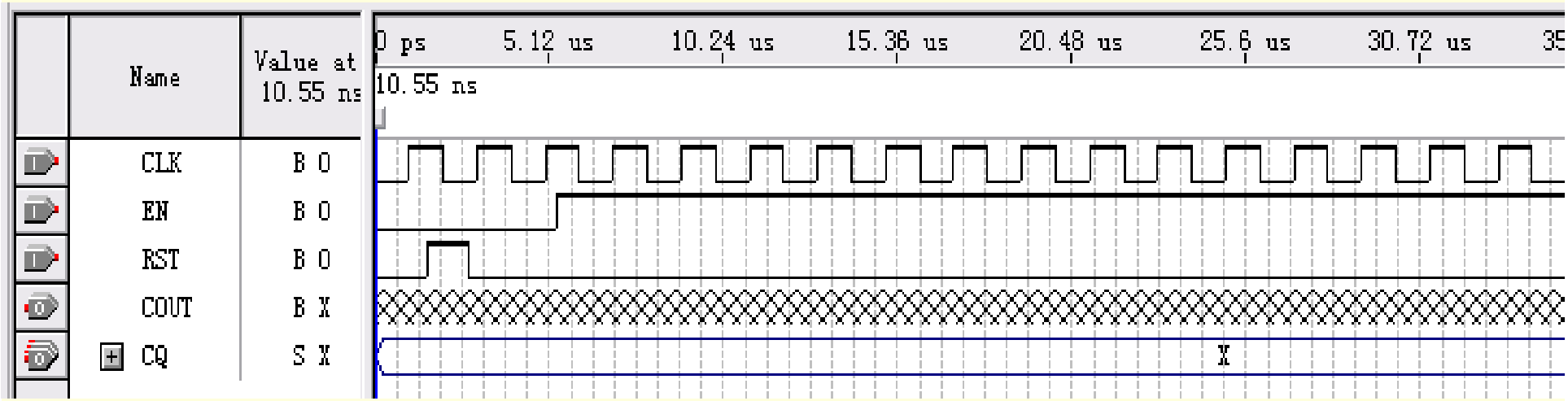


图2-29设置好的激励波形图

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

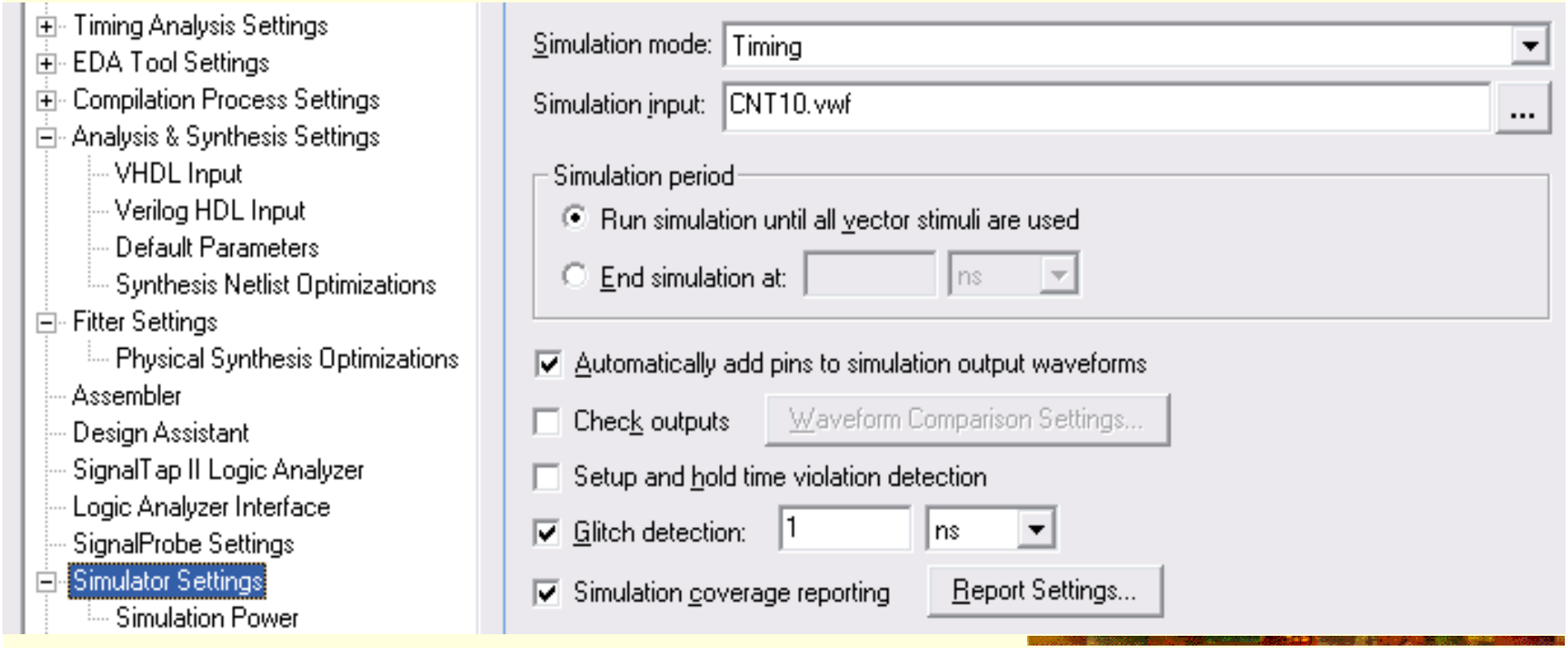


图2-30 选择仿真控制

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

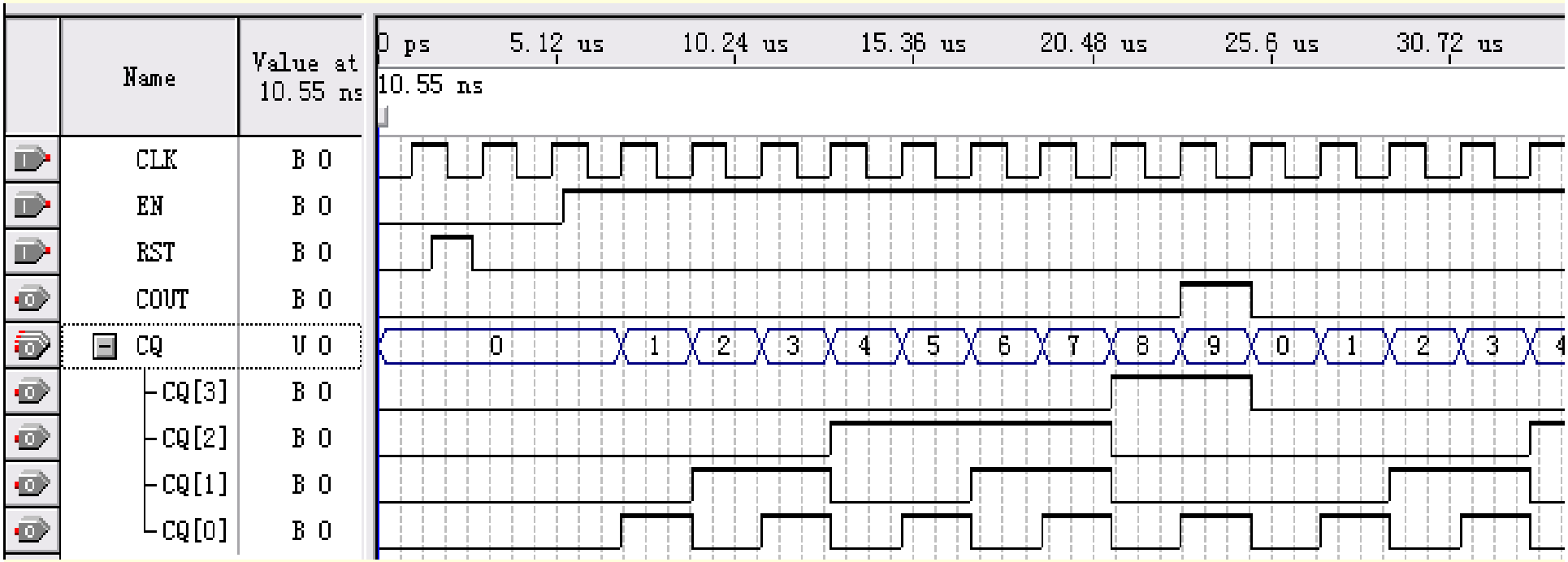


图2-31 仿真波形输出

2.6 QuartusII6.0使用向导

2.6.5 时序仿真

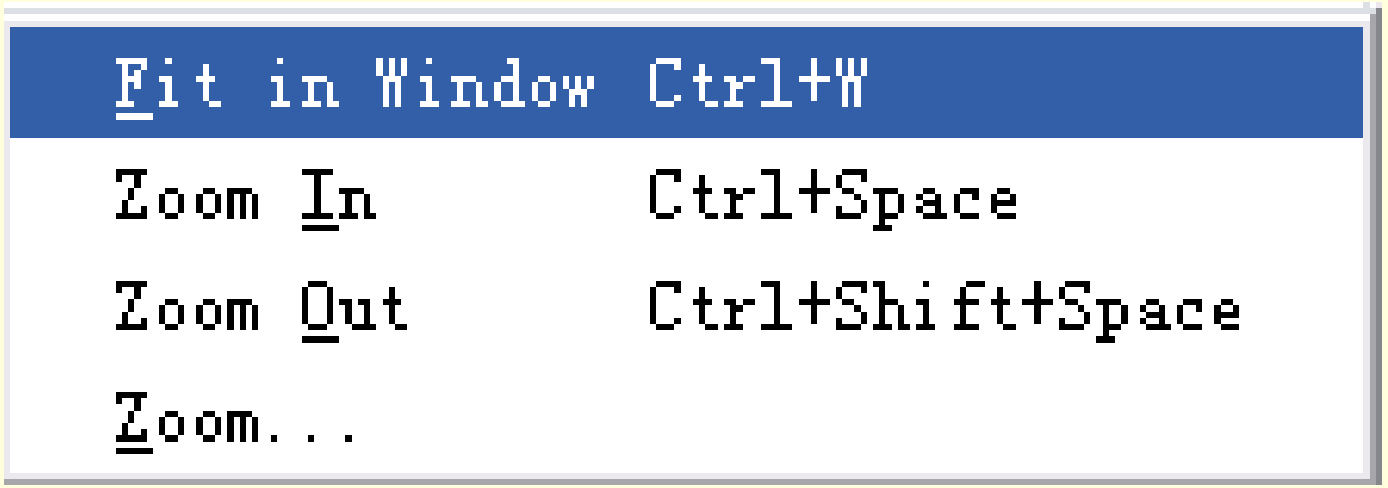


图2-32 选择全时域显示

2.6 QuartusII6.0使用向导

2.6.6 应用RTL电路图观察器

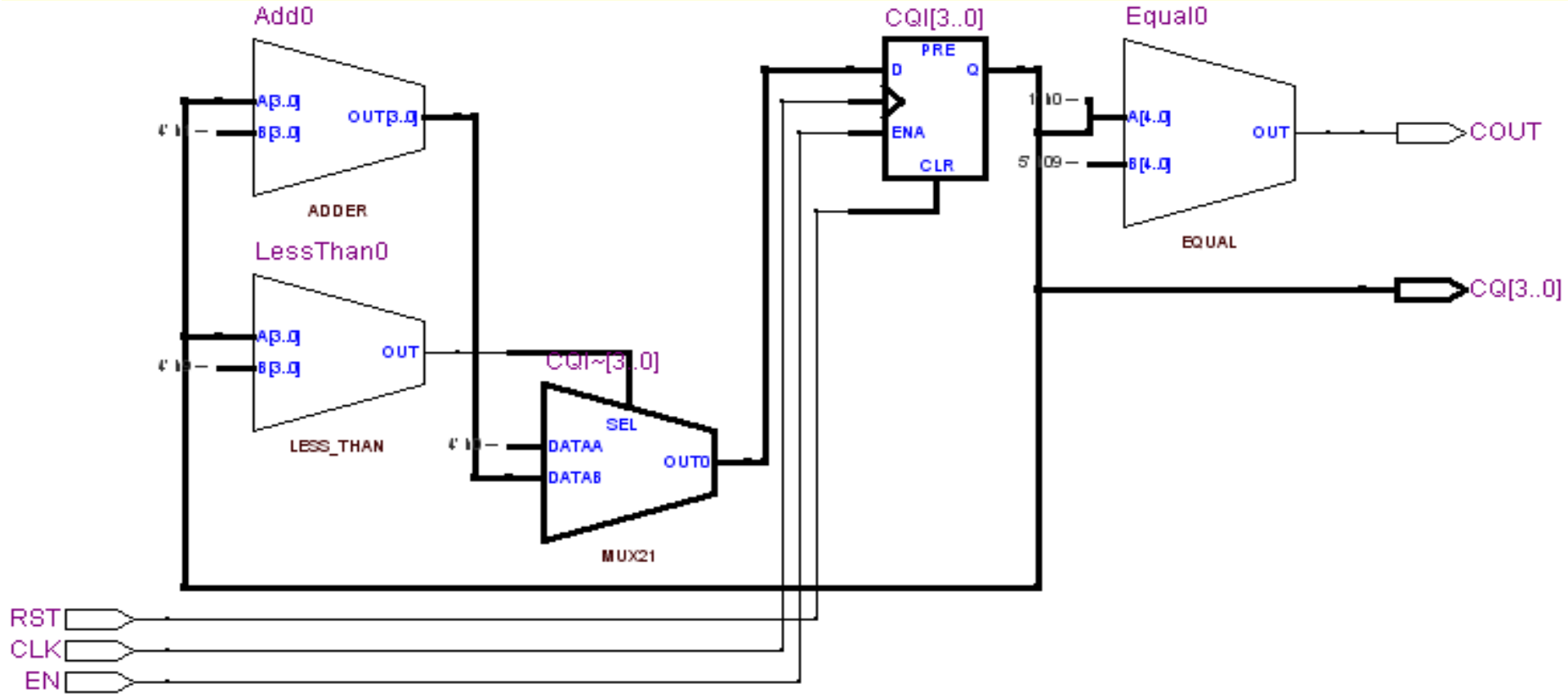


图2-33 cnt10工程的RTL电路图

2.6 QuartusII6.0使用向导

2.6.7 引脚锁定设置和下载

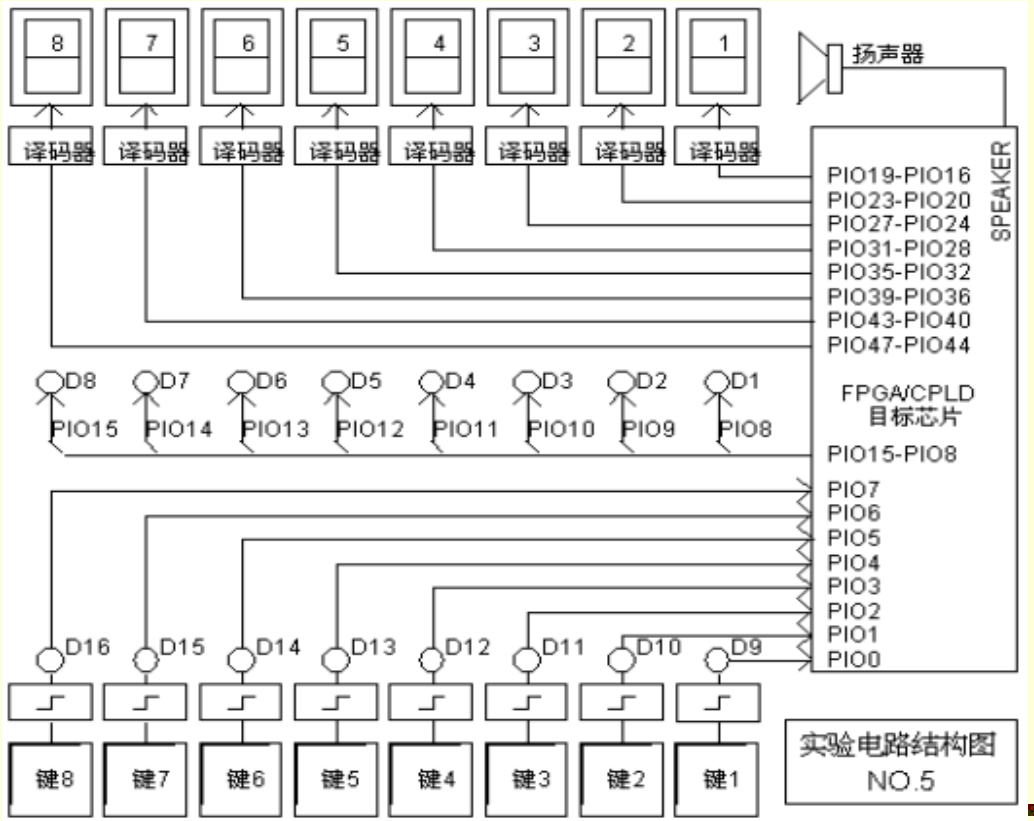


图2-34 GW48实验系统模式5实验电路图

2.6 QuartusII6.0使用向导

2.6.7 引脚锁定设置和下载

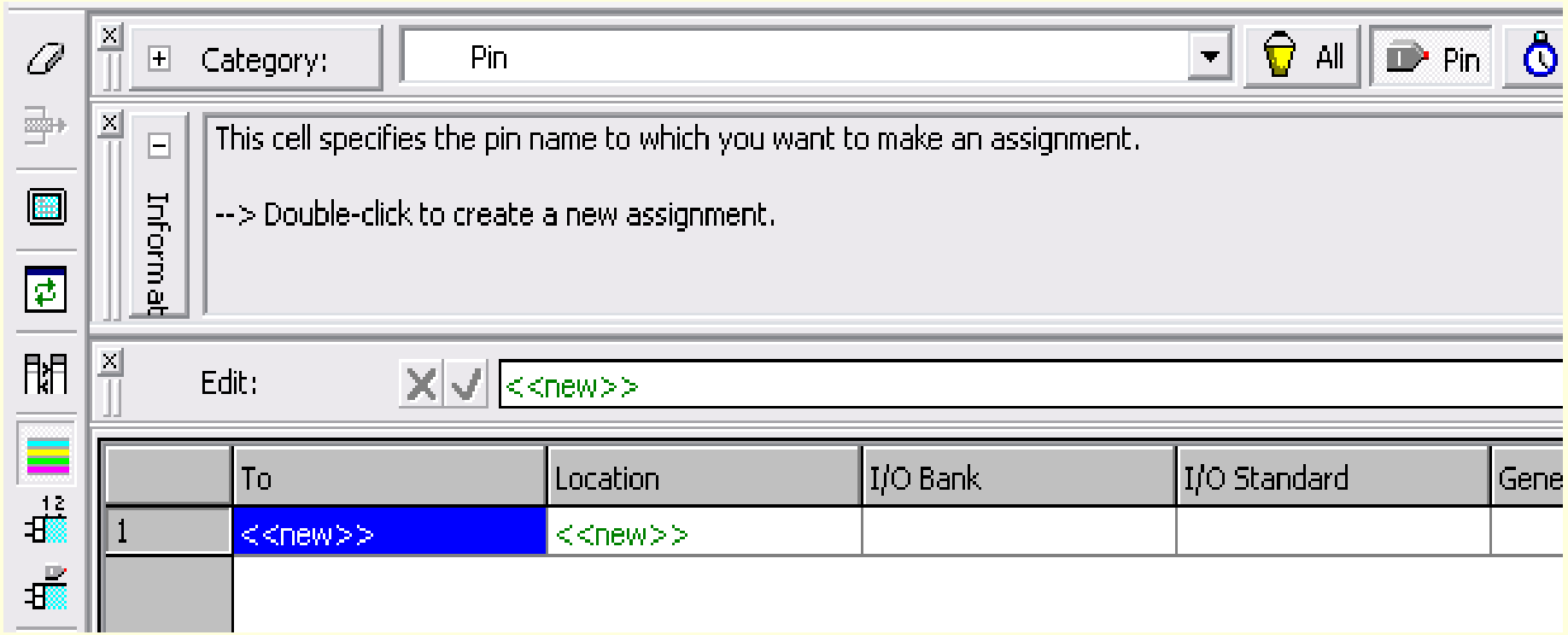


图2-35 Assignment Editor编辑器

2.6 QuartusII6.0使用向导

2.6.7 引脚锁定设置和下载

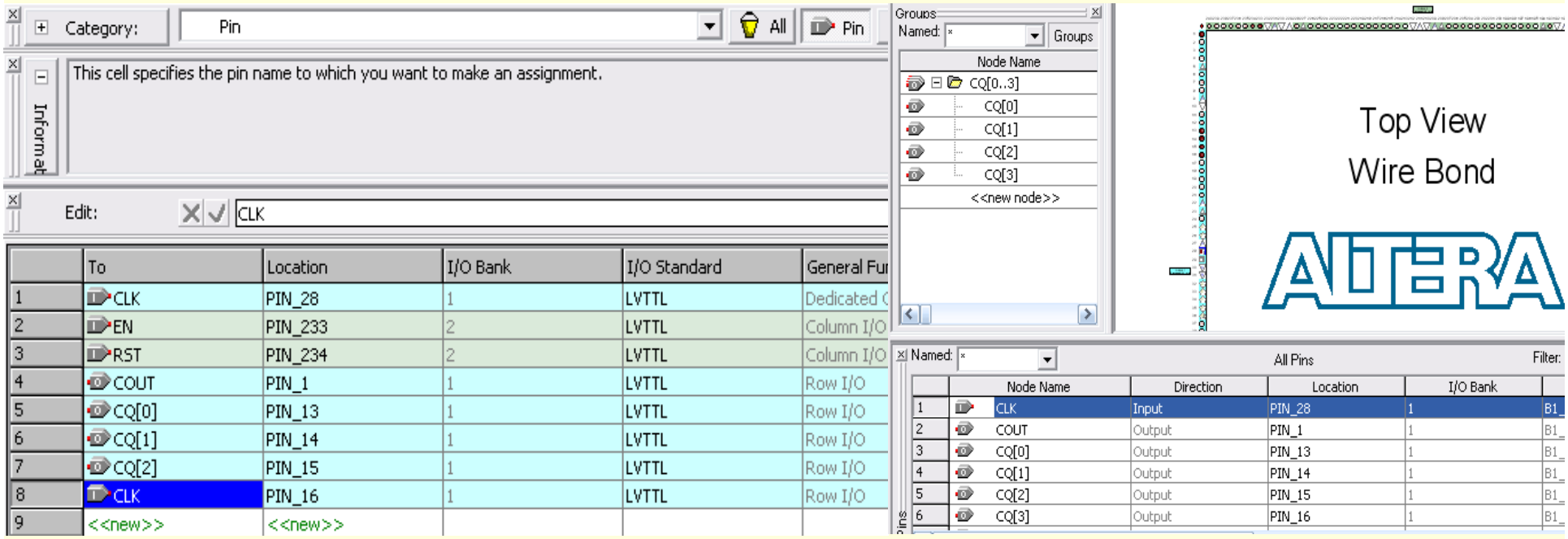


图2-36 两种引脚锁定对话框

2.6 QuartusII6.0使用向导

2.6.8 配置文件下载

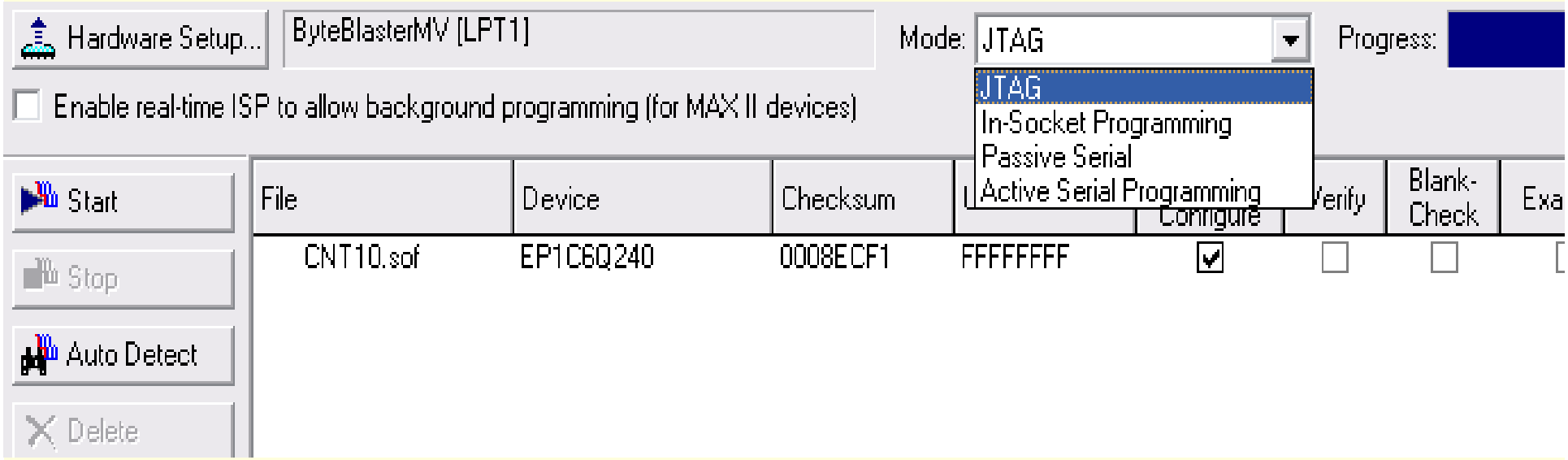


图2-37 选择编程下载文件

2.6 QuartusII6.0使用向导

2.6.8 配置文件下载

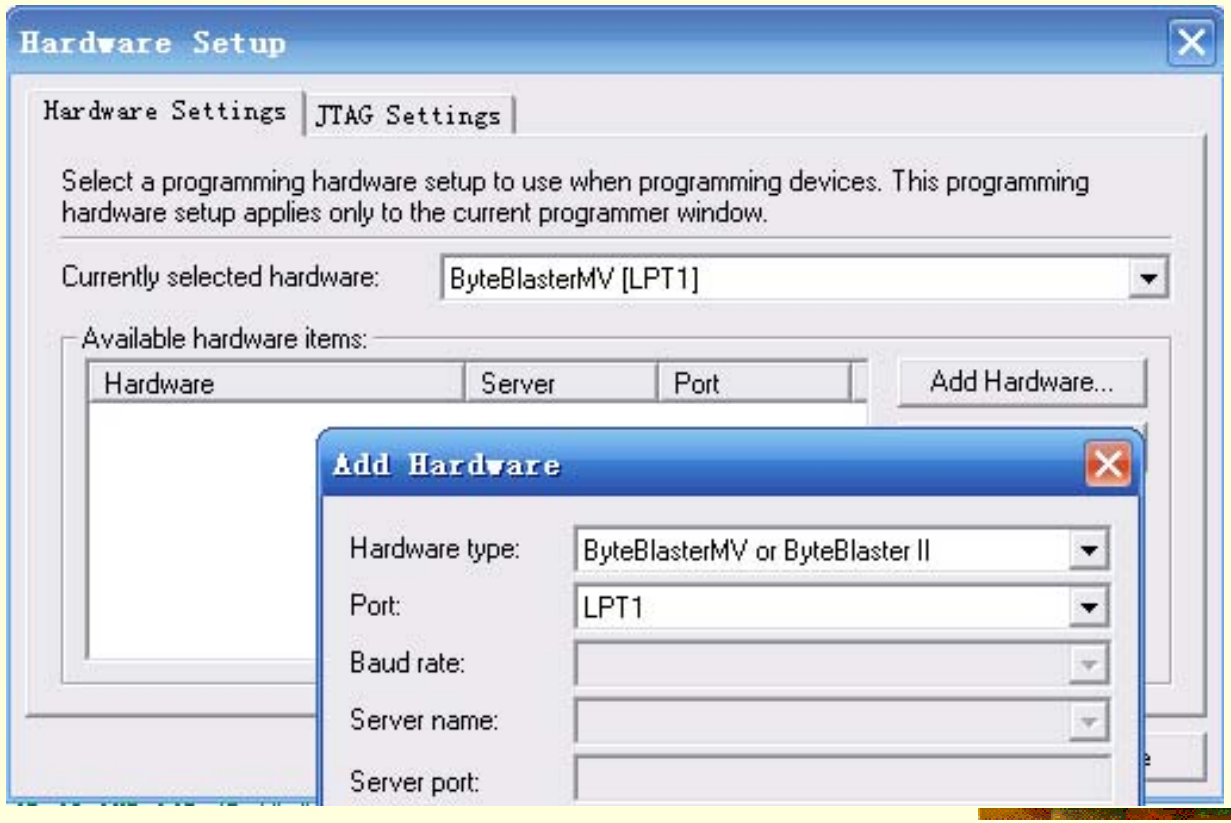


图2-38加入编程下载方式

2.6 QuartusII6.0使用向导

2.6.8 配置文件下载

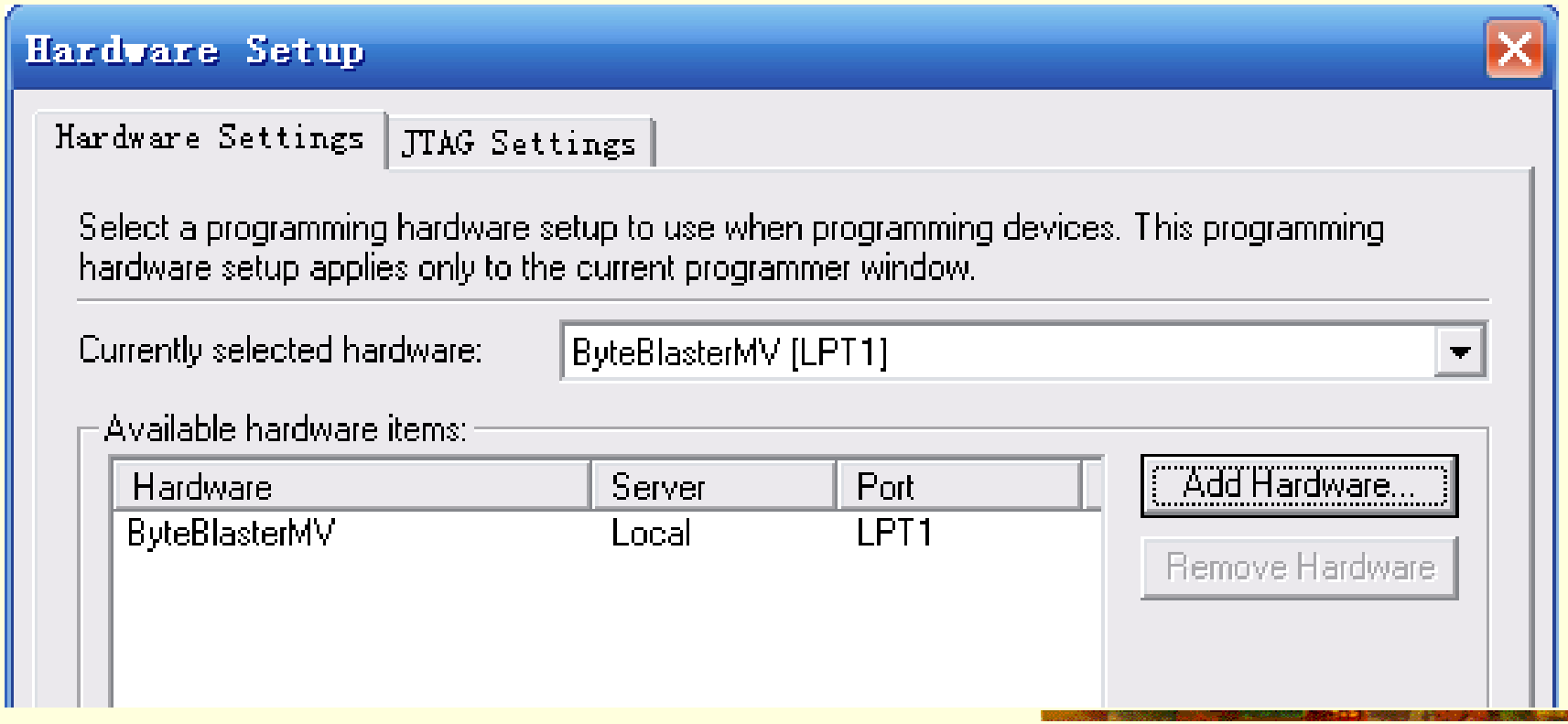


图2-39 双击选中的编程方式名

2.6 QuartusII6.0使用向导

2.6.9 AS模式编程配置器件

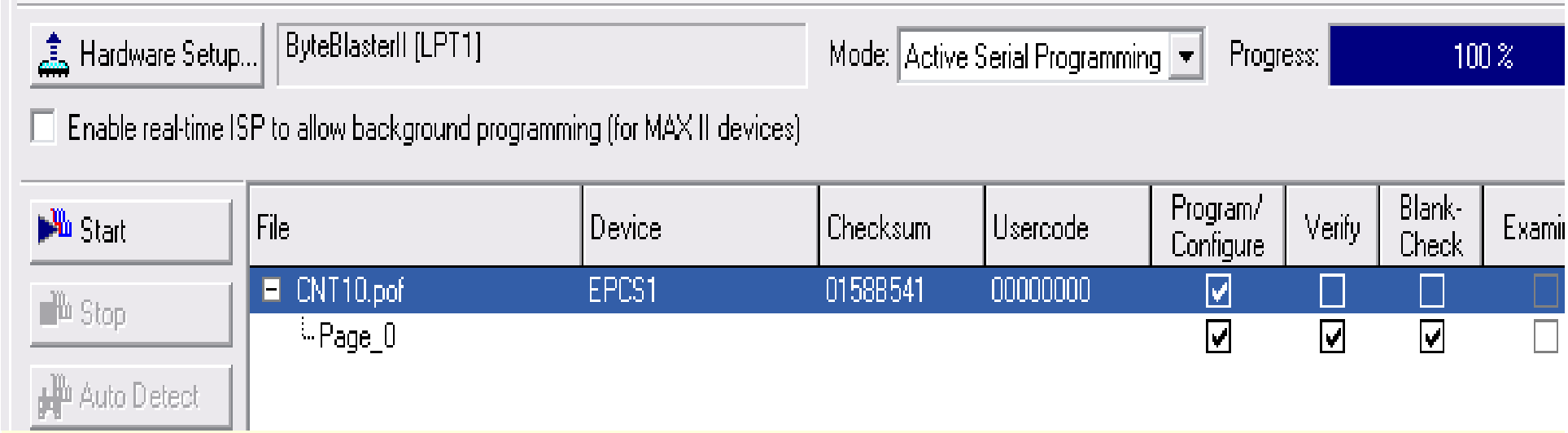


图2-40 ByteBlaster II接口AS模式编程窗口

2.6 QuartusII6.0使用向导

2.6.10 JTAG间接模式编程配置器件

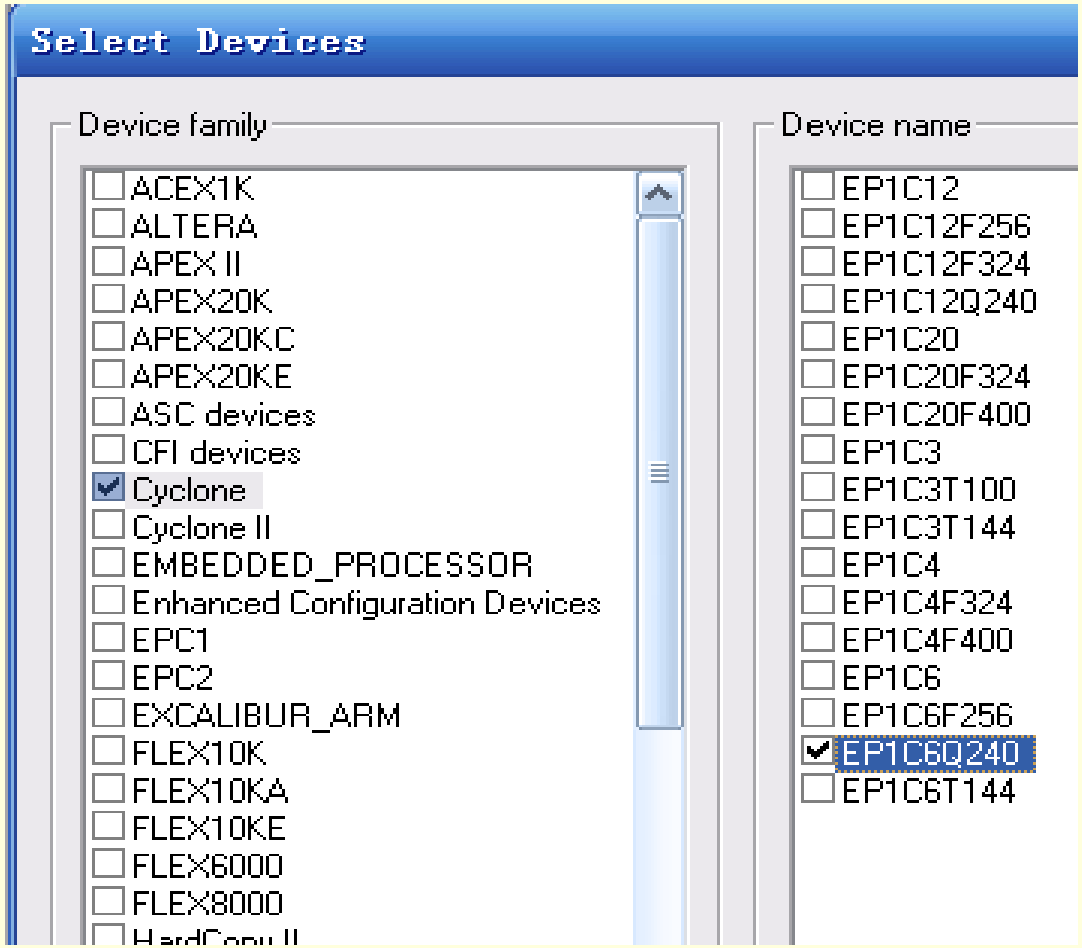


图2-41 选择目标器件EP1C6Q240

2.6 QuartusII6.0使用向导

2.6.10 JTAG间接模式编程配置器件

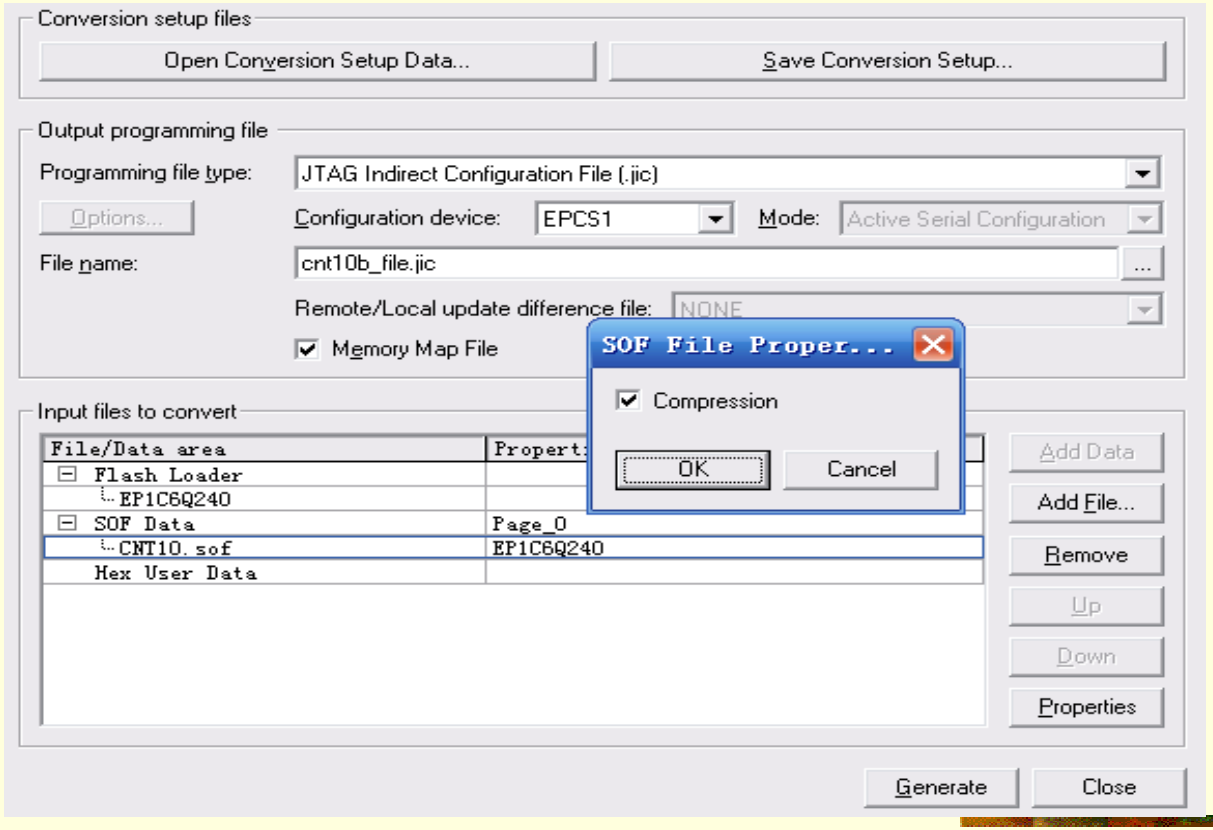


图2-42 选定SOF文件后，选择文件压缩

2.6 QuartusII6.0使用向导

2.6.10 JTAG间接模式编程配置器件

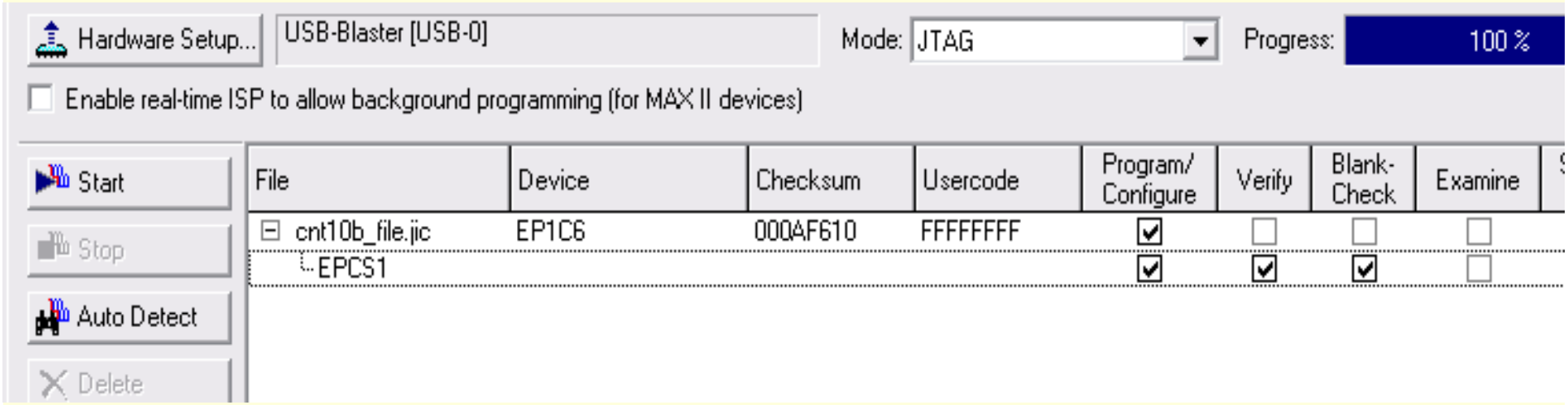


图2-43 用JTAG模式对配置器件EPCS1进行间接编程

2.7 嵌入式逻辑分析仪使用方法

1．打开SignalTap II编辑窗

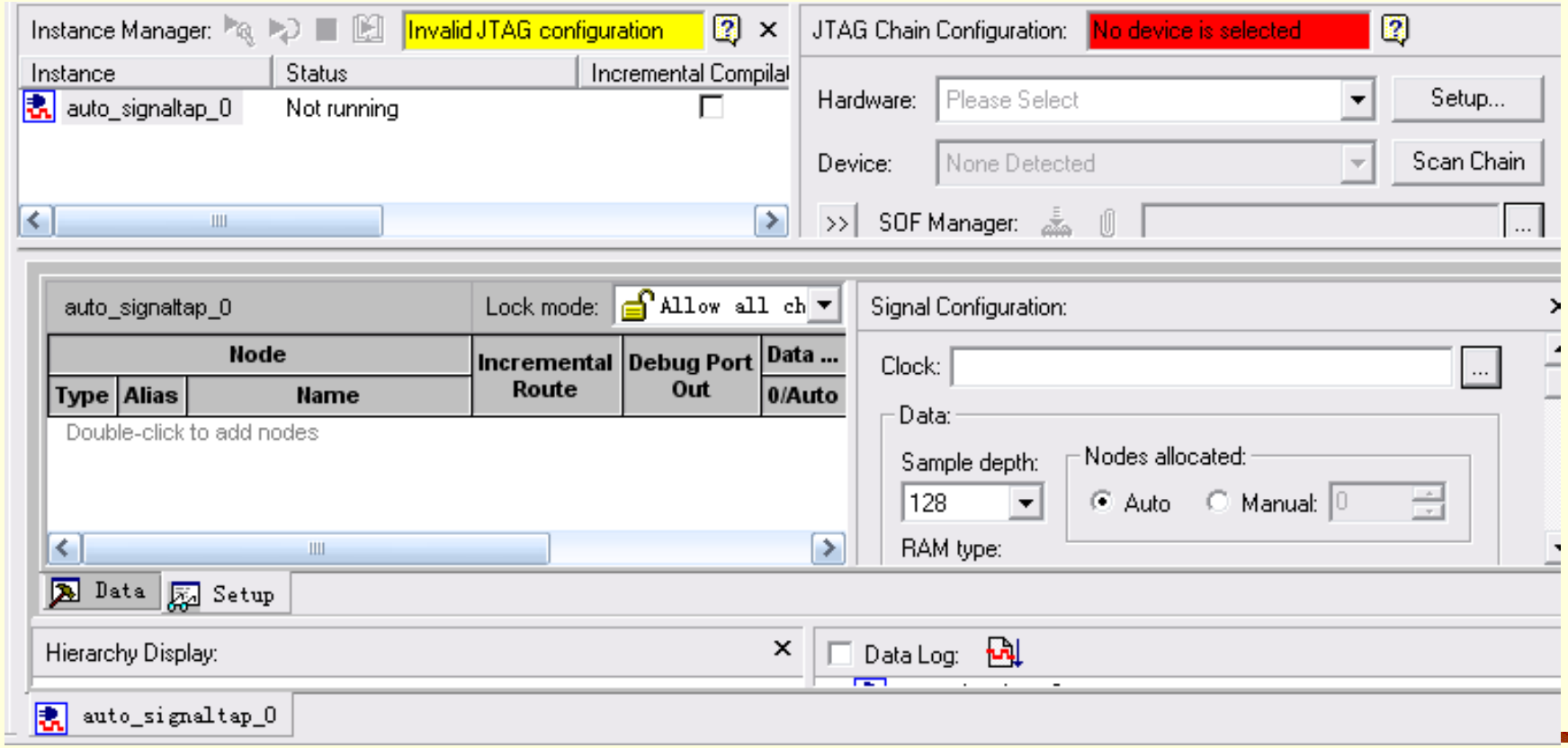


图2-44 SignalTap II编辑窗

2.7 嵌入式逻辑分析仪使用方法

2．调入待测信号

3．SignalTap II参数设置

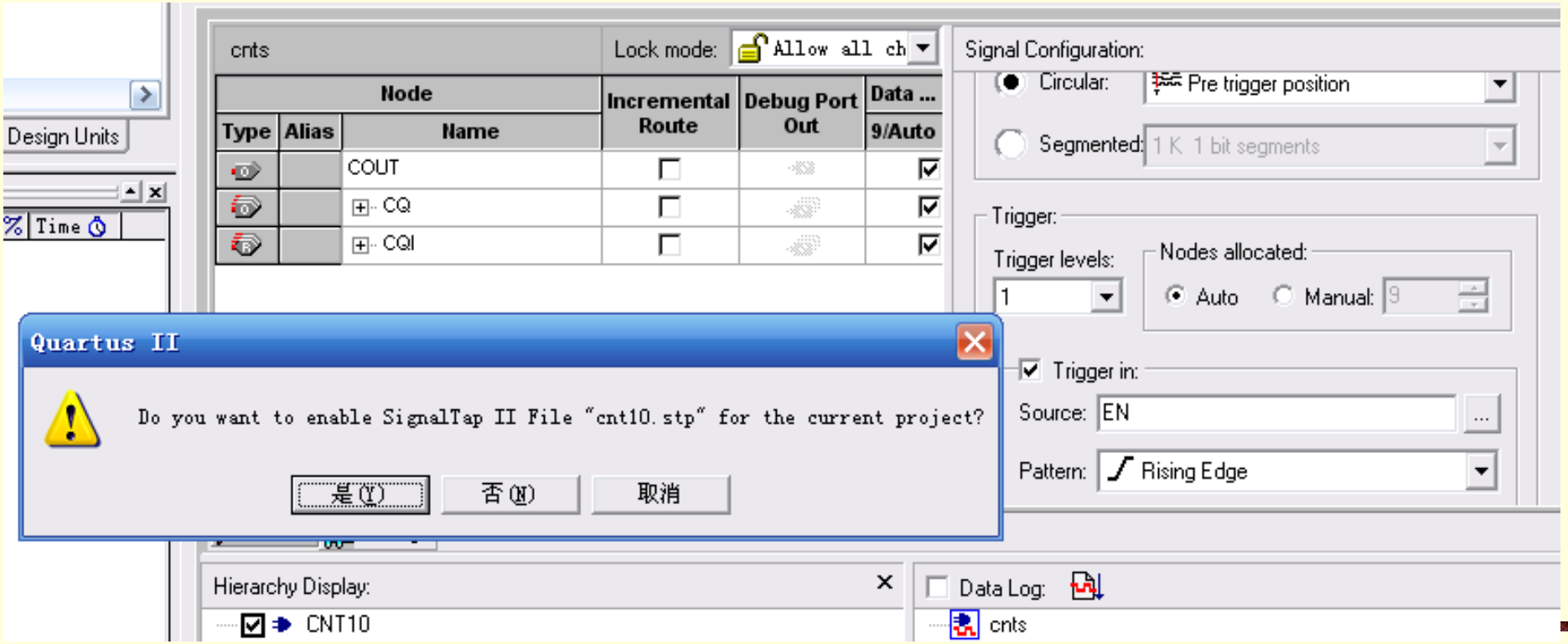


图2-45 SignalTap II编辑窗

2.7 嵌入式逻辑分析仪使用方法

4．文件存盘

5．编译下载

6．启动SignalTap II进行采样与分析

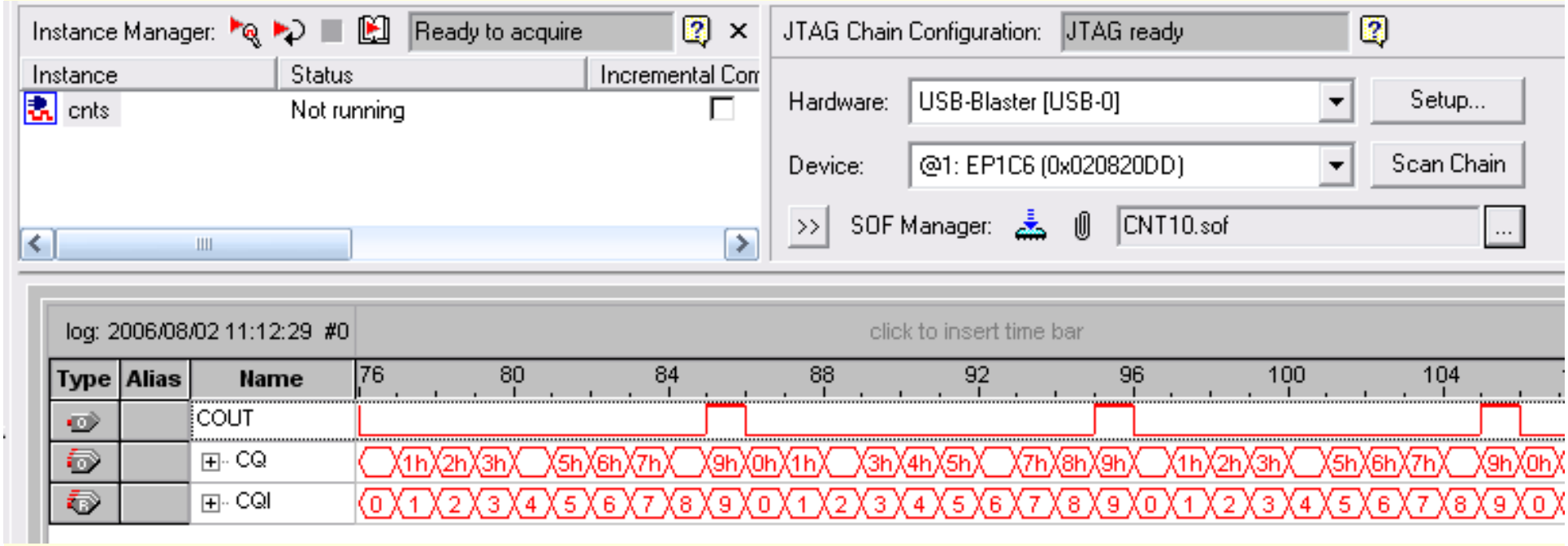


图2-46下载cnt10.sof并准备启动SignalTap II

2.7 嵌入式逻辑分析仪使用方法

6．启动SignalTap II进行采样与分析

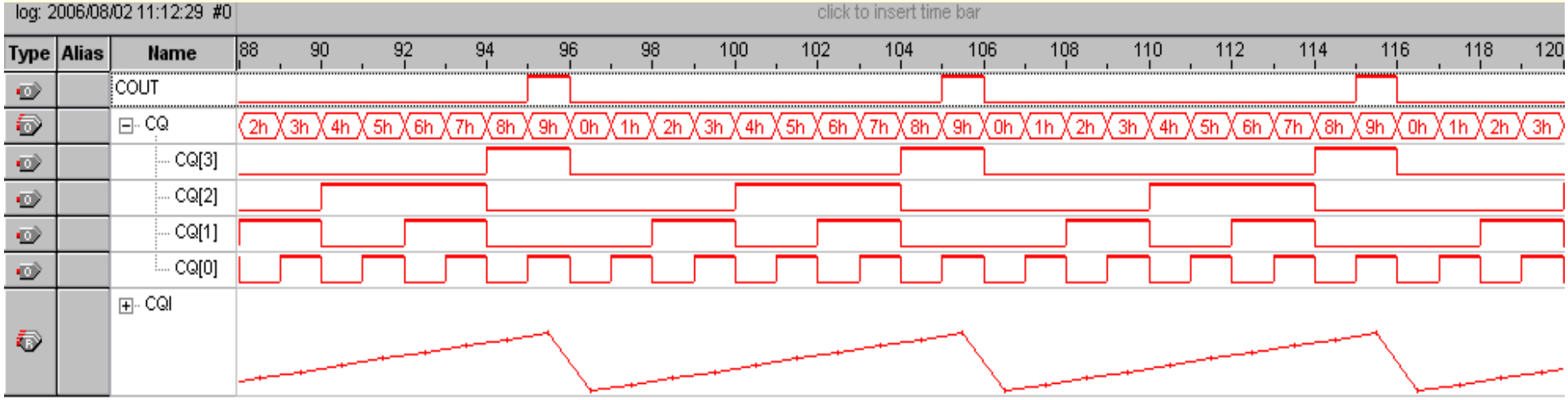


图2-47 SignalTap II数据窗设置后的信号波形

2.8 原理图输入设计方法

1. 为本项工程设计建立文件夹

2. 输入设计项目和存盘

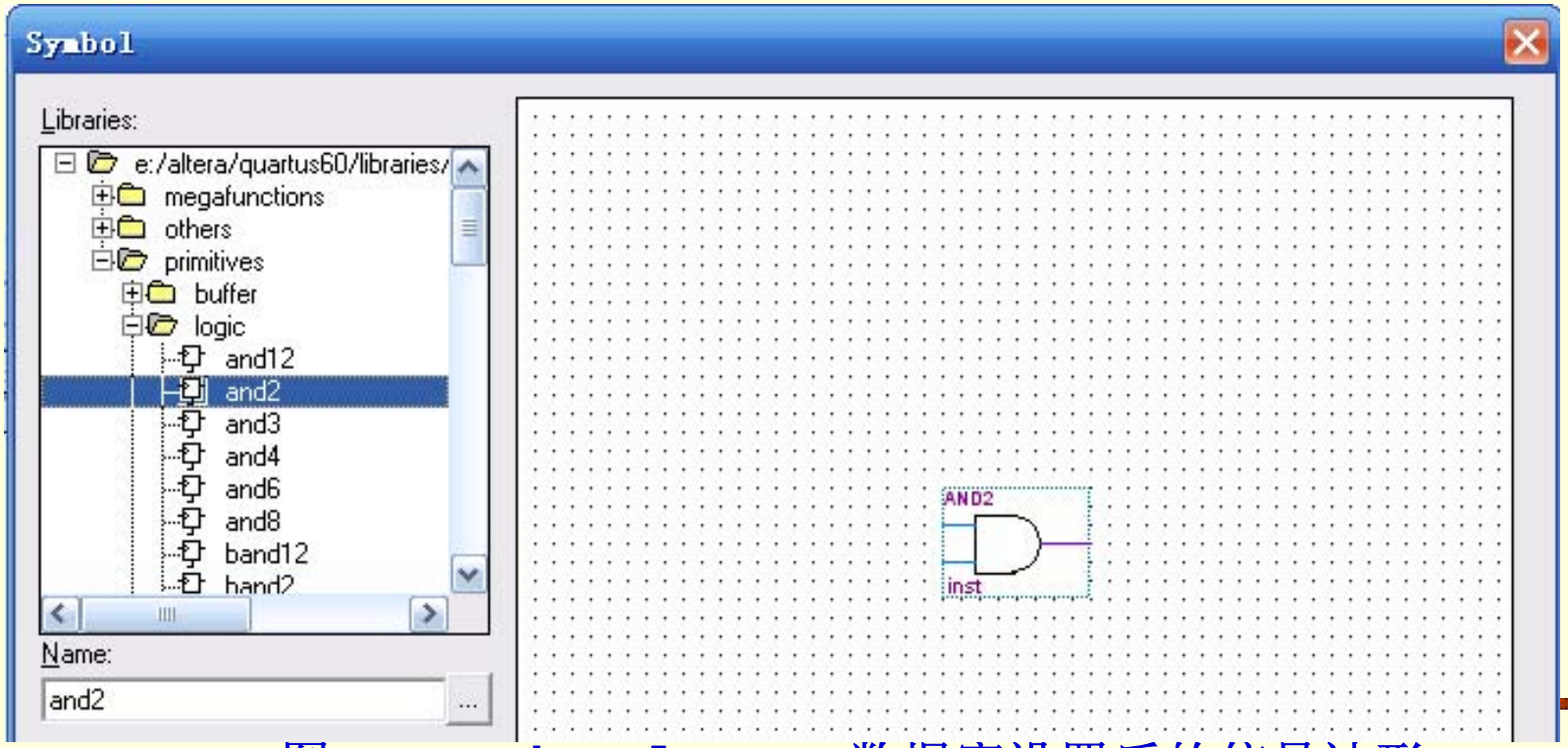


图2-47 SignalTap II数据窗设置后的信号波形

2.8 原理图输入设计方法

3. 将设计项目设置成可调用的元件

4. 设计全加器顶层文件

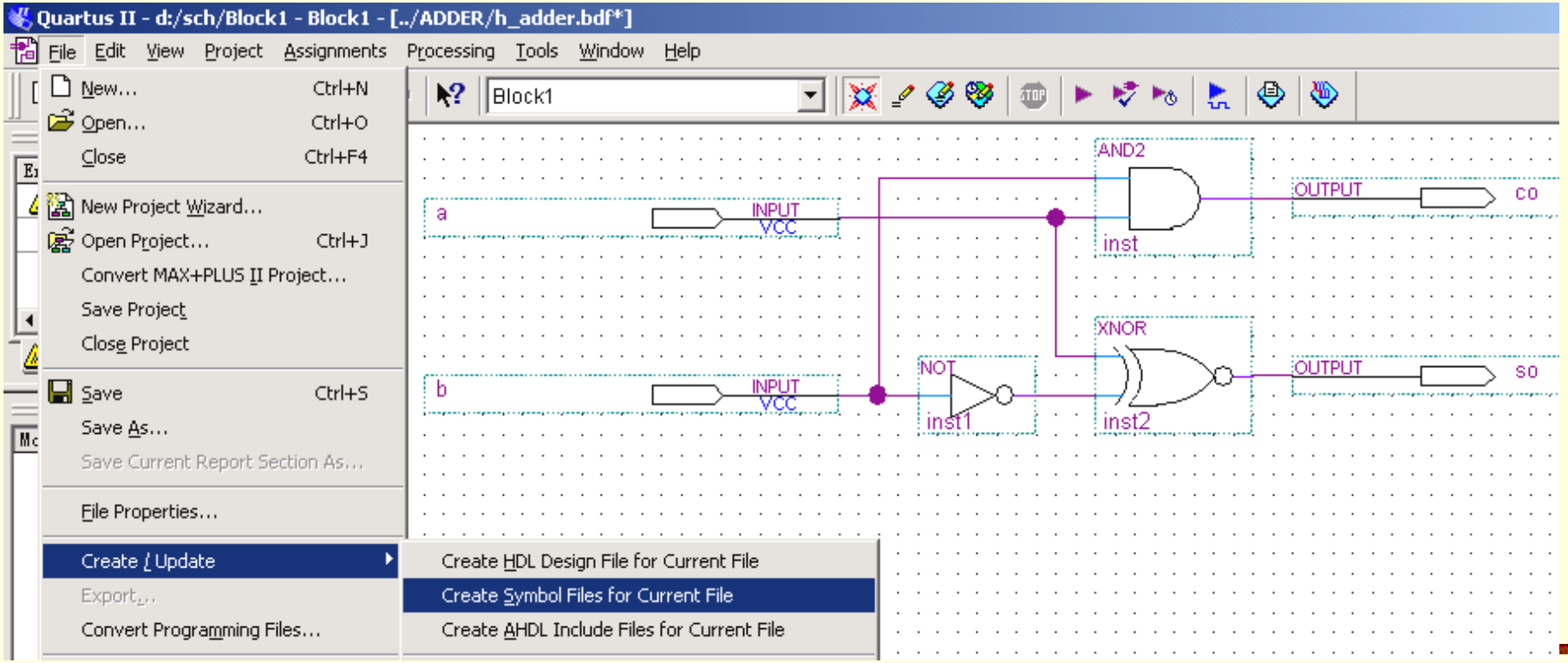


图2-49 将所需元件全部调入原理图编辑窗并连接好

2.8 原理图输入设计方法

4. 设计全加器顶层文件

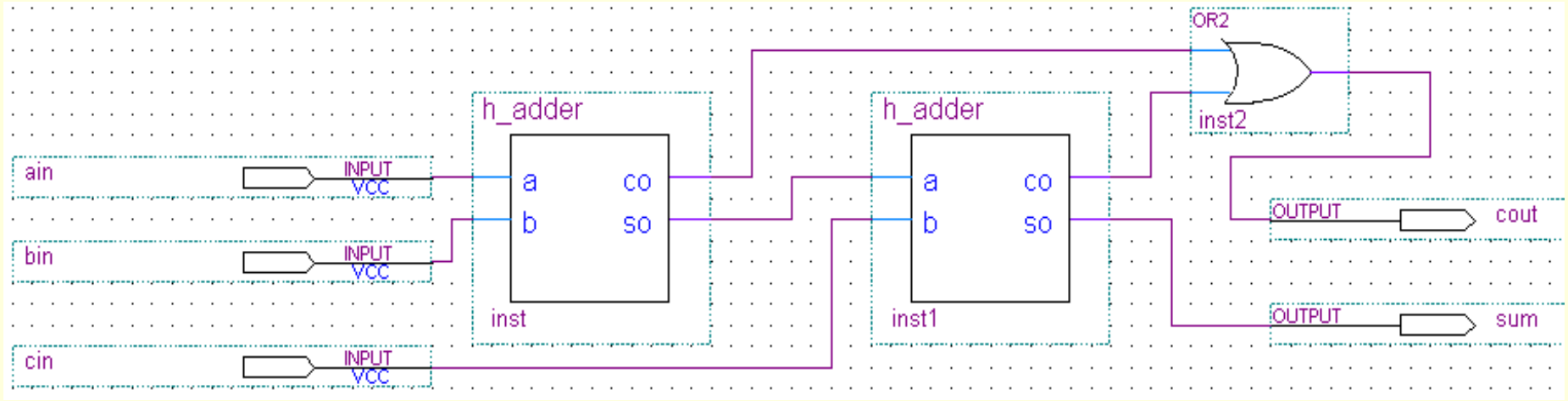


图2-50 连接好的全加器原理图f_adder.bdf

2.8 原理图输入设计方法

5. 将设计项目设置成工程和时序仿真

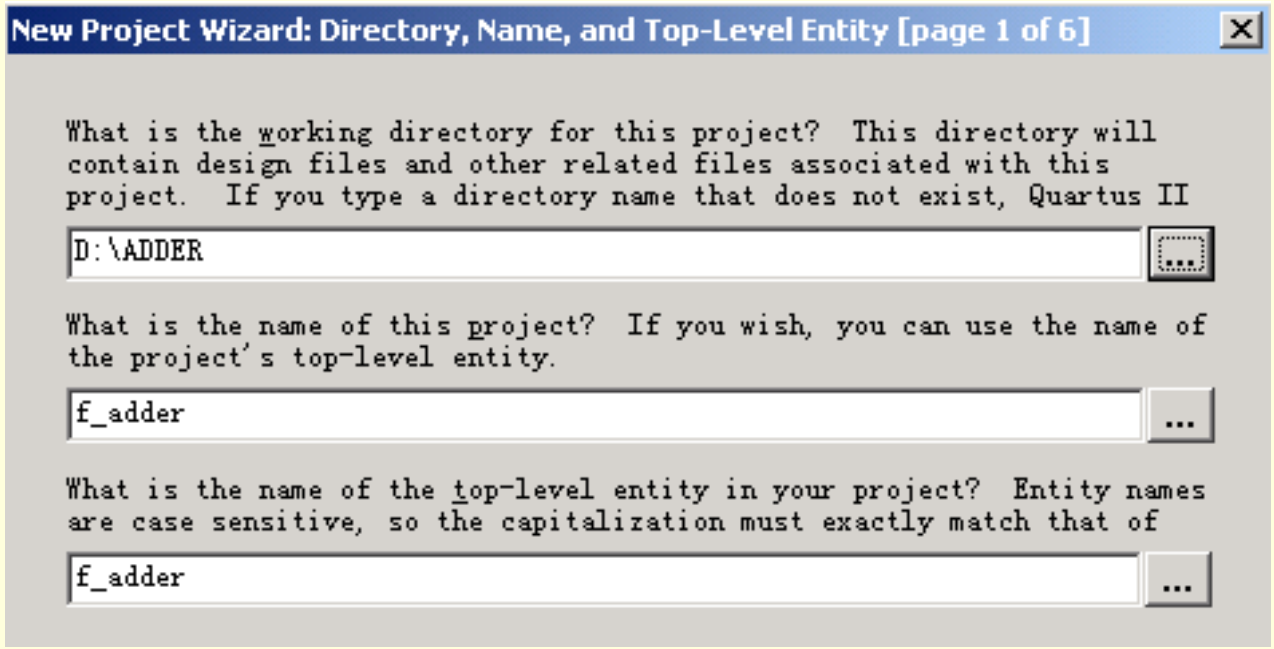


图2-51 f_adder.bdf工程设置窗

2.8 原理图输入设计方法

5. 将设计项目设置成工程和时序仿真

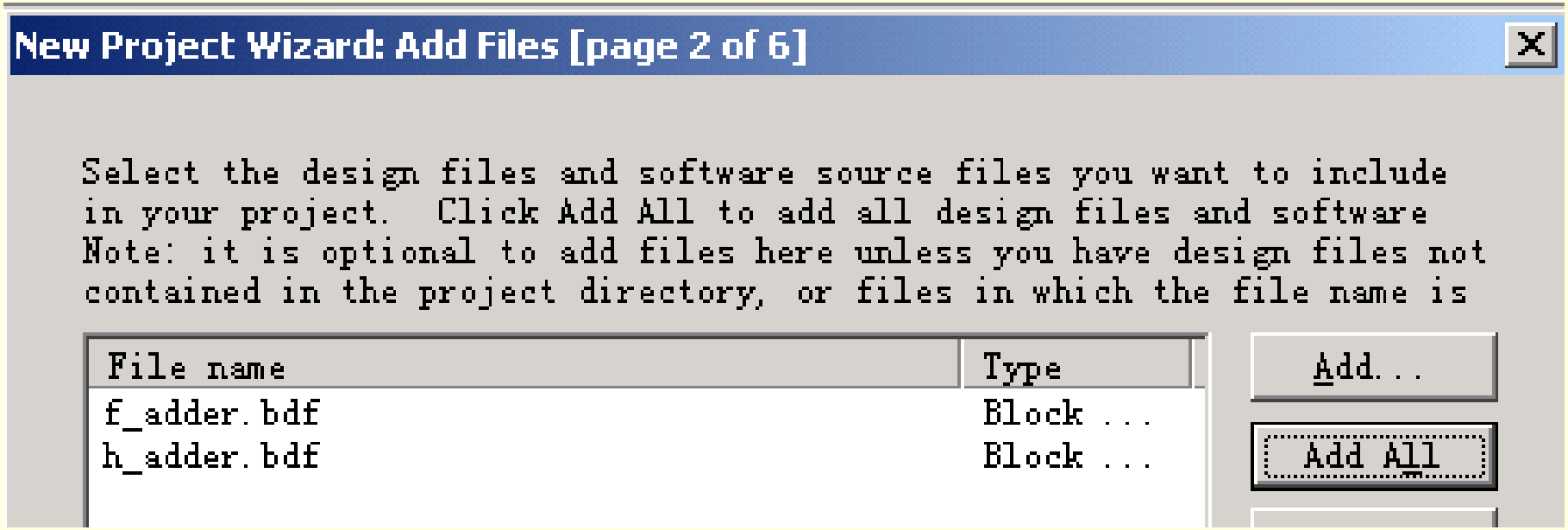


图2-52 加入本工程所有文件

2.8 原理图输入设计方法

5. 将设计项目设置成工程和时序仿真

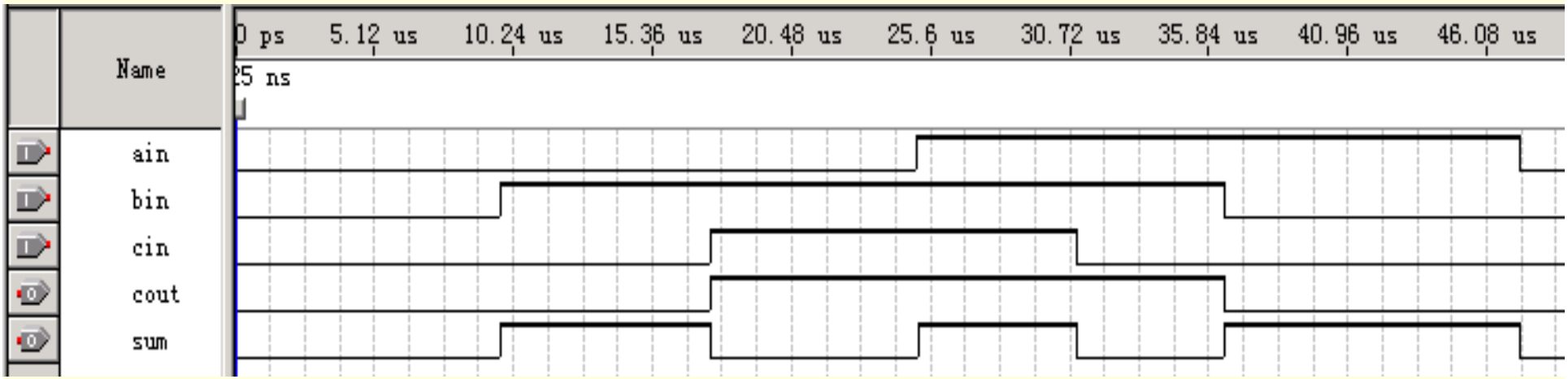


图2-53 全加器工程f_adder的仿真波形

习 题

2-1. 画出与下例实体描述对应的原理图符号元件：

```
ENTITY buf3s IS                          -- 实体1： 三态缓冲器
   PORT (input : IN STD_LOGIC ;     -- 输入端
         enable : IN STD_LOGIC ;     -- 使能端
         output : OUT STD_LOGIC ) ;  -- 输出端
END buf3x ;
ENTITY mux21 IS       --实体2： 2选1多路选择器
   PORT (in0,  in1, sel :  IN STD_LOGIC;
          output : OUT STD_LOGIC);
```

习 题

2-2. 图2-54所示的是4选1多路选择器，试分别用IF_THEN语句和CASE语句的表达方式写出此电路的VHDL程序。选择控制的信号s1和s0的数据类型为STD_LOGIC_VECTOR；当s1='0'，s0='0'；s1='0'，s0='1'；s1='1'，s0='0'和s1='1'，s0='1'分别执行y<=a、y<=b、y<=c、y<=d。

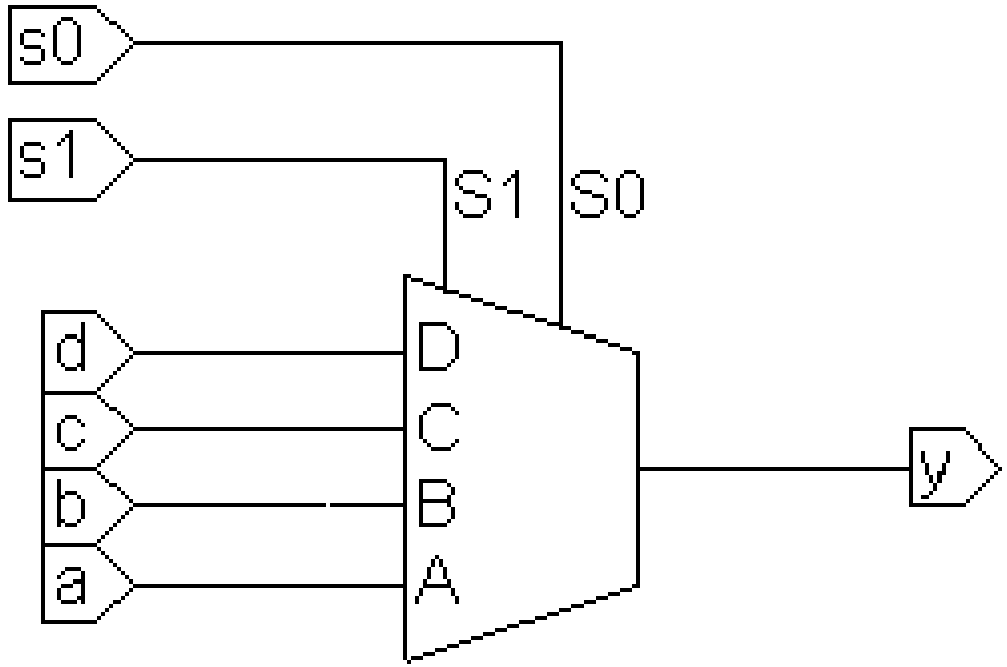


图2-54 4选1多路选择器

习 题

2-3. 图2-55所示的是双2选1多路选择器构成的电路MUXK，对于其中MUX21A，当s='0'和'1'时，分别有y<='a'和y<='b'。试在一个结构体中用两个进程来表达此电路，每个进程中用CASE语句描述一个2选1多路选择器MUX21A。

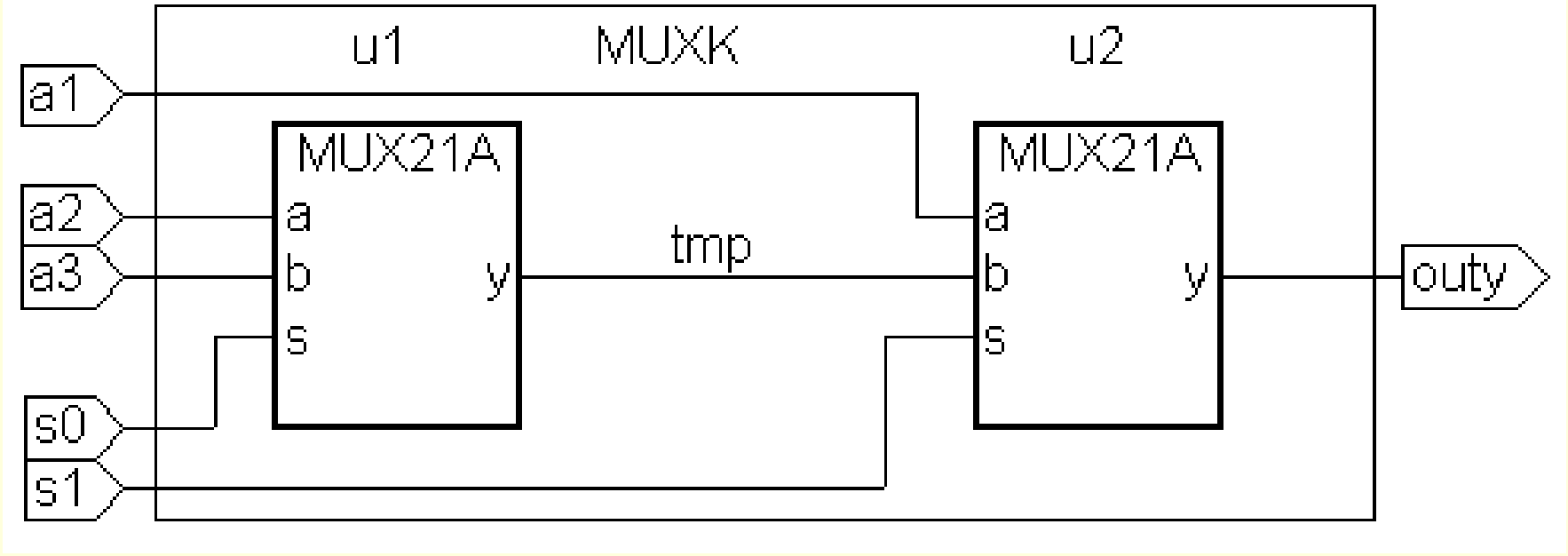


图2-55 双2选1多路选择器

习 题

2-4. 给出1位全减器的VHDL描述。要求：

(1) 首先设计1位半减器，然后用例化语句将它们连接起来，图2-56中h_suber是半减器，diff是输出差，s_out是借位输出，sub_in是借位输入。

(2) 以1位全减器为基本硬件，构成串行借位的8位减法器，要求用例化语句来完成此项设计(减法运算是 x – y - sun_in = diffr)。

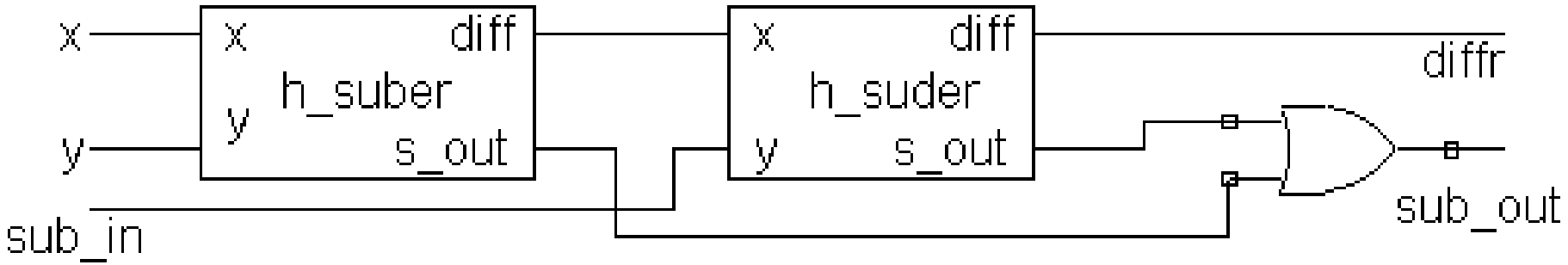


图2-56 1位全减器

习 题

2-5. 图2-57是一个含有上升沿触发的D触发器的时序电路，试写出此电路的VHDL设计文件。

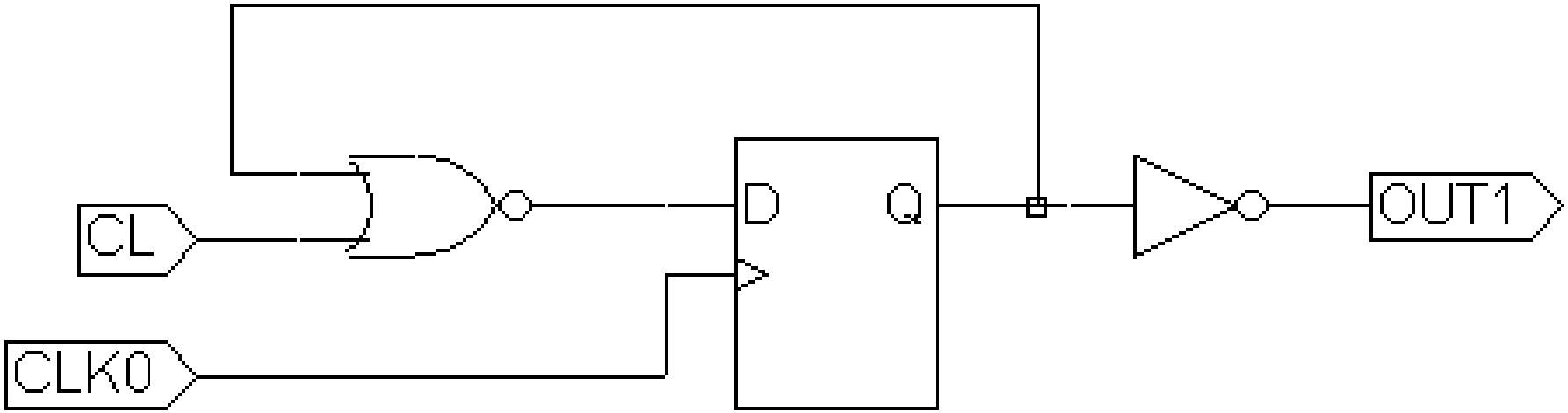


图2-57 时序电路图

习 题

2-6. 根据图2-58，写出顶层文件MX3256.VHD的VHDL设计文件。

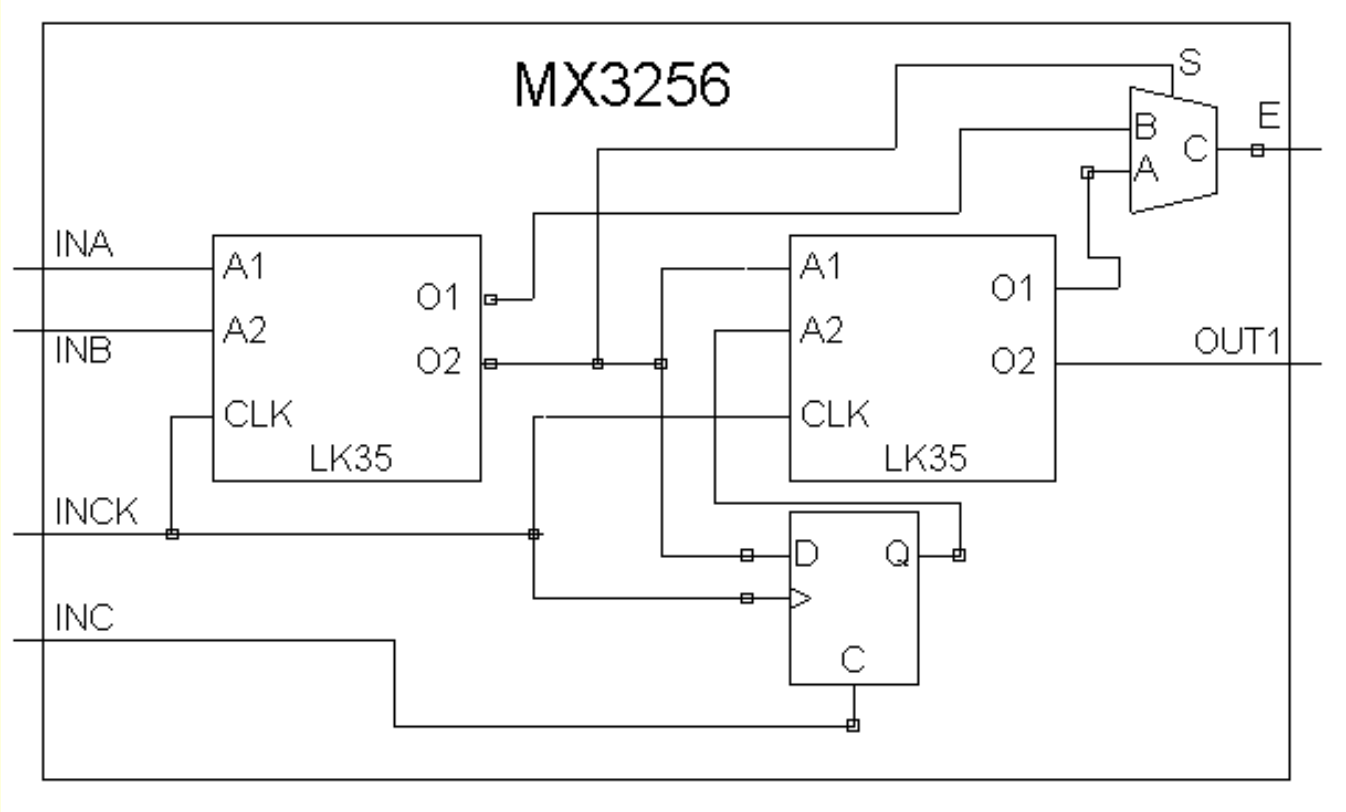


图2-58 题2-6电路图

习 题

2-7. 设计含有异步清零和计数使能的**16**位二进制加减可控计数器。

2-8. 归纳利用QuartusII进行VHDL文本输入设计的流程：从文件输入一直到SignalTap II测试。

2-9. 由图2-46、2-47，详细说明工程设计cnt10的硬件工作情况。

2-10. 如何为设计中的SignalTap II加入独立采用时钟？试给出完整的程序和对它的实测结果。

2-11. 参考Quartus II的Help，详细说明Assignments菜单中Settings对话框的功能。

（1）说明其中的Timing Requirements & Qptions的功能、使用方法和检测途径。

（2）说明其中的Compilation Process的功能和使用方法。

（3）说明Analysis & Synthesis Setting的功能和使用方法，以及其中的Synthesis Netlist Optimization的功能和使用方法。

（4）说明Fitter Settings中的Design Assistant和Simulator功能，举例说明它们的使用方法。

习 题

2-12. 概述**Assignments**菜单中**Assignment Editor**的功能，举例说明。

2-13. 用**74148**和与非门实现**8421BCD**优先编码器，用**3**片**74139**组成一个**5-24**线译码器。

2-14. 用**74283**加法器和逻辑门设计实现一位**8421BCD**码加法器电路，输入输出均是**BCD**码，**CI**为低位的进位信号，**CO**为高位的进位信号，输入为两个**1**位十进制数**A**，输出用**S**表示。

2-15. 设计一个**7**人表决电路，参加表决者**7**人，同意为**1**，不同意为**0**，同意者过半则表决通过，绿指示灯亮；表决不通过则红指示灯亮。

2-16. 设计一个周期性产生二进制序列**01001011001**的序列发生器，用移位寄存器或用同步时序电路实现，并用时序仿真器验证其功能。

实验与设计

实验2-1. 组合电路的设计

参考实验示例：

/CMPUT_EXPMT/CH2_Expt/ DEMO_21_mux21/mux21a

(1) 实验目的

熟悉**Quartus**Ⅱ的**VHDL**文本设计流程全过程，学习简单组合电路的设计、多层次电路设计、仿真和硬件测试。

(2) 实验任务1

首先利用**Quartus**Ⅱ完成**2**选**1**多路选择器（例**2-3**）的文本编辑输入**(mux21a.vhd)**和仿真测试等步骤，给出图**2-3**所示的仿真波形。最后在实验系统上进行硬件测试，验证本项设计的功能。

实验与设计

(3) 实验任务2: 将此多路选择器看成是一个元件**mux21a**，利用元件例化语句描述图**2-55**，并将此文件放在同一目录中。以下是部分参考程序：

```
...
  COMPONENT MUX21A
    PORT ( a，b，s：  IN  STD_LOGIC;
                y：  OUT STD_LOGIC);
  END COMPONENT ;
...
  u1 : MUX21A PORT MAP(a=>a2，b=>a3，s=>s0，y=>tmp);
  u2 : MUX21A PORT MAP(a=>a1，b=>tmp，s=>s1，y=>outy);
 END ARCHITECTURE BHV ;
```

按照本章给出的步骤对上例分别进行编译、综合、仿真。并对其仿真波形作出分析说明。

实验与设计

(4) 实验任务3　引脚锁定以及硬件下载测试。建议选实验电路模式5（附录图4），用键1(PIO0)控制s0；用键2(PIO1)控制s1；a3、a2和a1分别接clock5、clock0和clock2；输出信号outy仍接扬声器spker。通过短路帽选择clock0接256Hz信号，clock5接1024Hz，clock2接8Hz信号。最后进行编译、下载和硬件测试实验（通过选择键1、键2，控制s0、s1，可使扬声器输出不同音调）。

(5) 实验报告　根据以上的实验内容写出实验报告，包括程序设计、软件编译、仿真分析、硬件测试和详细实验过程；给出程序分析报告、仿真波形图及其分析报告。

(6) 附加内容

根据本实验以上提出的各项实验内容和实验要求，设计1位全加器。首先用Quartus Ⅱ完成以上给出的全加器的设计，包括仿真和硬件测试。实验要求分别仿真测试底层硬件或门和半加器，最后完成顶层文件全加器的设计和测试，给出设计原程序，程序分析报告、仿真波形图及其分析报告。

(7) 实验习题

以1位二进制全加器为基本元件，用例化语句写出8位并行二进制全加器的顶层文件，并讨论此加法器的电路特性。

实验与设计

实验2-2. 时序电路的设计

(1) 实验目的

熟悉Quartus Ⅱ的VHDL文本设计过程，学习简单时序电路的设计、仿真和测试。

(2) 实验任务1

设计触发器(使用例2-6)，给出程序设计、软件编译、仿真分析、硬件测试及详细实验过程。

(3) 实验任务2

设计锁存器(使用例2-14)，同样给出程序设计、软件编译、仿真分析、硬件测试及详细实验过程。

实验与设计

(4) 实验任务3

只用一个1位二进制全加器为基本元件和一些辅助的时序电路，设计一个8位串行二进制全加器，要求：1、能在8至9个时钟脉冲后完成8位二进制数（加数被加数的输入方式为并行）的加法运算，电路须考虑进位输入Cin和进位输出Cout；2、给出此电路的时序波形，讨论其功能，并就工作速度与并行加法器进行比较；3、在FPGA中进行实测。对于GW48实验系统，建议选择电路模式1（附录图2），键2，键1输入8位加数；键4，键3输入8位被加数；键8作为手动单步时钟输入；键7控制进位输入Cin；键9控制清0；数码6和数码5显示相加和；发光管D1显示溢出进位Cout。4、键8作为相加起始控制，同时兼任清0；工作时钟由clock0自动给出，每当键8发出一次开始相加命令，电路即自动相加，结束后停止工作，并显示相加结果。就外部端口而言，与纯组合电路8位并行加法器相比，此串行加法器仅多出一个加法起始/清0控制输入和工作时钟输入端。提示：此加法器有并/串和串/并移位寄存器各一。

(5) 实验报告　分析比较实验内容1和2的仿真和实测结果，说明这两种电路的异同点。

实验与设计

实验2-3. 含异步清0和同步时钟使能的加法计数器的设计

参考实验示例：**/CMPUT_EXPMT/CH2_Expt/ DEMO_23_cnt10/cnt10**

(1) 实验目的

学习计数器的设计、仿真和硬件测试，进一步熟悉**VHDL**设计技术。

(2) 实验原理

实验程序为例**2-22**，实验原理参考**2.5**节，设计流程参考本章。

(3) 实验任务1

在**Quartus Ⅱ** 上对例**2-22**进行编辑、编译、综合、适配、仿真。说明例中各语句的作用，详细描述示例的功能特点，给出其所有信号的时序仿真波形。

(4) 实验任务2

引脚锁定以及硬件下载测试。引脚锁定后进行编译、下载和硬件测试实验。将实验过程和实验结果写进实验报告。

实验与设计

(5) 实验任务3　使用SignalTap II对此计数器进行实时测试。

(6) 实验任务4　从设计中去除SignalTap II，要求全程编译后生成用于配置器件EPCS1编程的压缩POF文件，并使用USB-Blaster，通过JTAG间接模式对实验板上的EPCS1进行编程，最后进行验证。

(7) 实验任务5　为此项设计加入一个可用于SignalTapII采样的独立的时钟输入端（采用时钟选择clock0=12MHz，计数器时钟CLK分别选择256Hz、16384Hz、6MHz），并进行实时测试。

(8) 思考题　在例2-22中是否可以不定义信号 CQI，而直接用输出端口信号完成加法运算，即：

CQ <= CQ + 1？为什么？

(9) 实验报告　将实验原理、设计过程、编译仿真波形和分析结果、硬件测试实验结果写进实验报告。

实验与设计

实验2-4. 用原理图输入法设计8位全加器

(1) 实验目的

熟悉利用Quartus II 的原理图输入方法设计简单组合电路，掌握层次化设计的方法，并通过一个8位全加器的设计把握利用EDA软件进行原理图输入方式的电子线路设计的详细流程。

(2) 实验原理

一个8位全加器可以由8个1位全加器构成，加法器间的进位可以串行方式实现，即将低位加法器的进位输出cout与相临的高位加法器的最低进位输入信号cin相接。而一个1位全加器可以按照6.1节介绍的方法来完成。

实验与设计

(3) 实验任务1

完成半加器和全加器的设计，包括原理图输入、编译、综合、适配、仿真、实验板上的硬件测试，并将此全加器电路设置成一个硬件符号入库。键1、键2、键3(PIO0/1/2)分别接ain、bin、cin；发光管D2、D1(PIO9/8)分别接sum和cout。

(4) 实验任务2

建立一个更高层次的原理图设计，利用以上获得的1位全加器构成8位全加器，并完成编译、综合、适配、仿真和硬件测试。建议选择电路模式1（附录图2）；键2、键1输入8位加数；键4、键3输入8位被加数；数码6/5显示加和；D8显示进位cout。

(5) 实验报告

详细叙述8位加法器的设计流程；给出各层次的原理图及其对应的仿真波形图；给出加法器的时序分析情况；最后给出硬件测试流程和结果。