EDA技术及其应用

第4章　应用VHDL设计数字系统

4.1 多路选择器的VHDL描述

4.1.1 多路选择器的VHDL描述

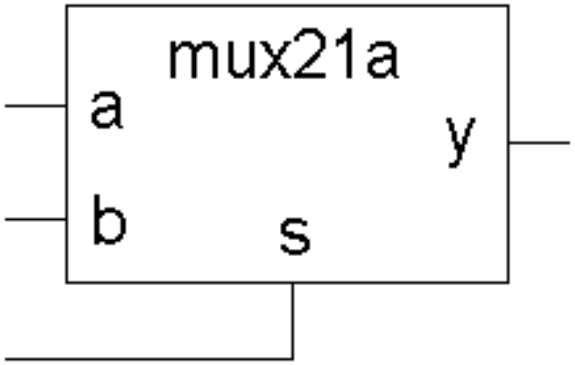


图4-1 mux21a实体

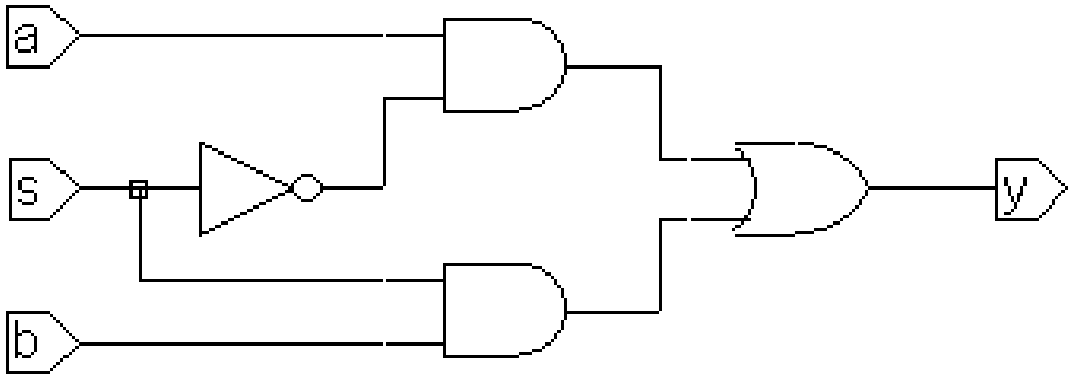


图4-2 mux21a结构体

4.1 多路选择器的VHDL描述

4.1.1 多路选择器的VHDL描述

【例4-1】

```
ENTITY mux21a IS
 PORT ( a, b : IN  BIT;
        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 ;
```

4.1 多路选择器的VHDL描述

4.1.1 多路选择器的VHDL描述

【例4-2】

```
ENTITY mux21a IS
 PORT ( a, b : IN  BIT;
        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 ;
```

4.1 多路选择器的VHDL描述

4.1.1 多路选择器的VHDL描述

【例4-3】

```
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 ;
```

4.1 多路选择器的VHDL描述

4.1.1 多路选择器的VHDL描述

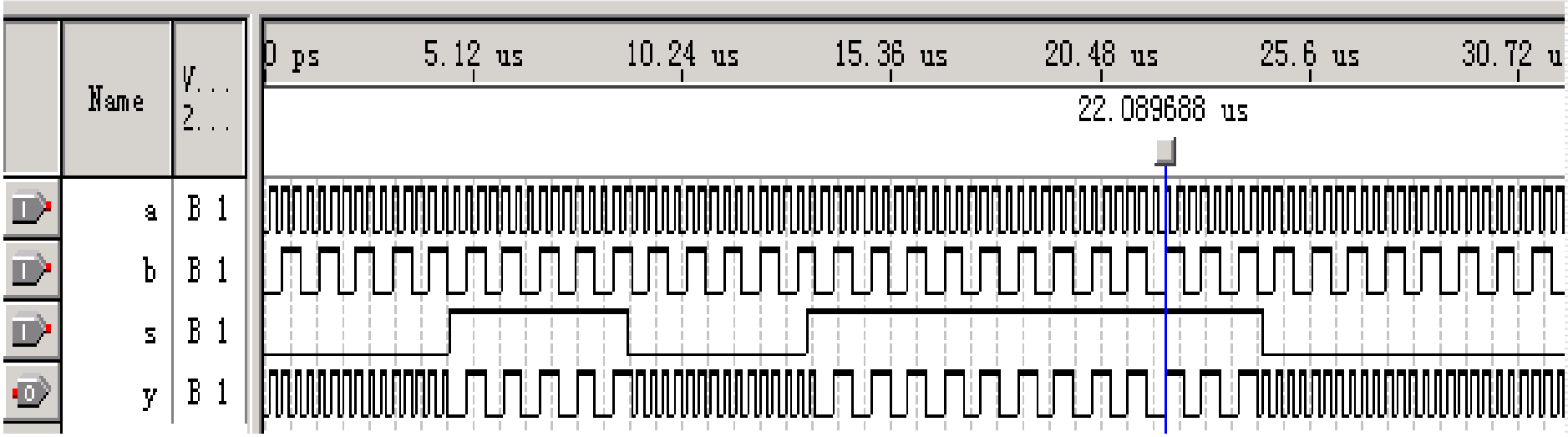


图4-3 mux21a功能时序波形

4.1 多路选择器的VHDL描述

4.1.2 语句结构和语法说明

1. 实体表达

【例4-4】

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

4.1 多路选择器的VHDL描述

4.1.2 语句结构和语法说明

2. 实体名

3. 端口语句和端口信号名

4. 端口模式

5. 数据类型

4.1 多路选择器的VHDL描述

6. 结构体表达

【例4-5】

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

4.1 多路选择器的VHDL描述

7. 赋值符号和数据比较符号

```
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 . .
```

8. 逻辑操作符

9. 条件语句

4.1 多路选择器的VHDL描述

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 ;
```

4.1 多路选择器的VHDL描述

11. 进程语句和顺序语句

12. 文件取名和存盘

4.2 寄存器描述的VHDL程序

4.2.1 D触发器的描述

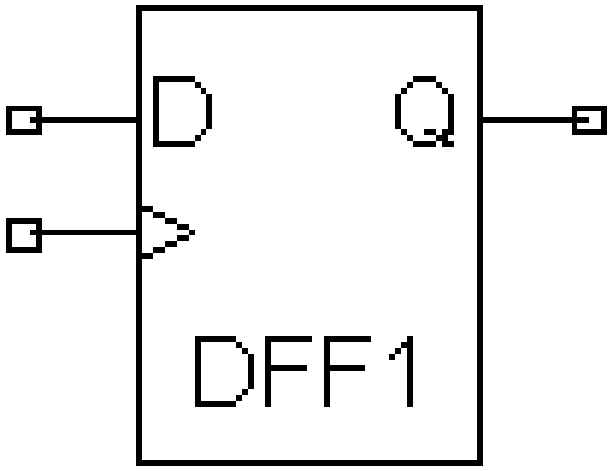


图4-4 D触发器

4.2 寄存器描述的VHDL程序

【例4-6】

```
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（双横线--是注释符号）
    END bhv;
```

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

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','-');
```

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

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 ;
```

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

3. 信号定义和数据对象

4. 上升沿检测表式和信号属性函数**EVENT**

5. 不完整条件语句与时序电路

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

5. 不完整条件语句与时序电路

【例4-7】

```
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 ;
```

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

5. 不完整条件语句与时序电路

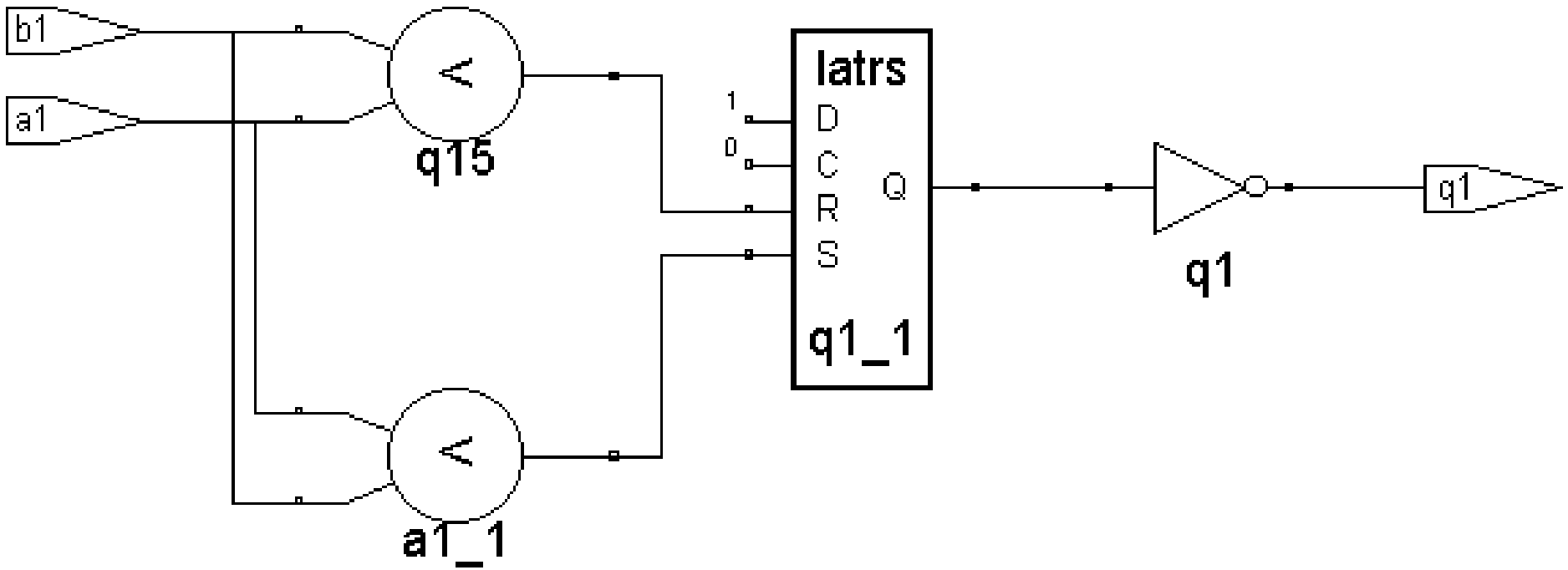


图4-5 例4-7的电路图（Synplify综合）

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

5. 不完整条件语句与时序电路

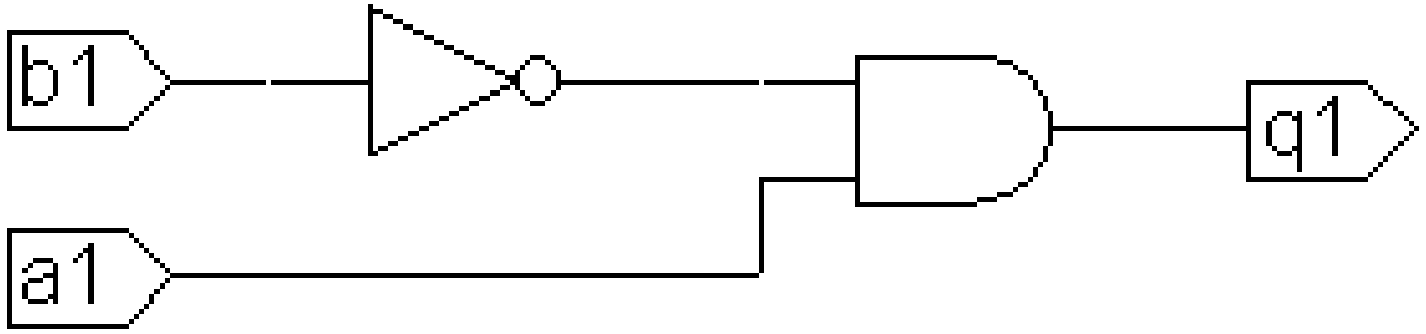


图4-6 例4-8的电路图（Synplify综合）

4.2 寄存器描述的VHDL程序

4.2.2 VHDL描述的语言现象说明

5. 不完整条件语句与时序电路

【例4-8】

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

4.2 寄存器描述的VHDL程序

4.2.3 实现时序电路的不同表述

【例4-9】

```
...
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 ;
```

4.2 寄存器描述的VHDL程序

4.2.3 实现时序电路的不同表述

【例4-10】

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

4.2 寄存器描述的VHDL程序

【例4-11】

```
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) -- 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 ;
```

4.2 寄存器描述的VHDL程序

4.2.3 实现时序电路的不同表述

【例4-12】

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

4.2 寄存器描述的VHDL程序

4.2.3 实现时序电路的不同表述

【例4-13】

```
...
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 ;
```

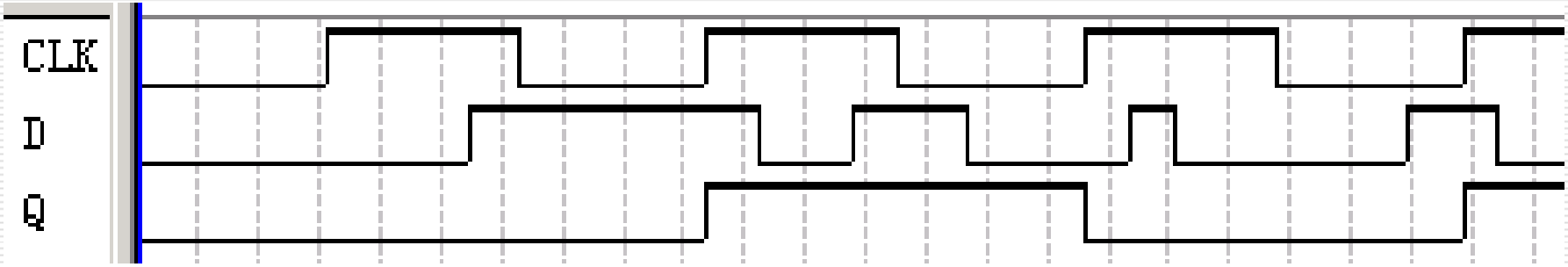


图4-7 例4-13的时序波形

4.2 寄存器描述的VHDL程序

4.2.3 实现时序电路的不同表述

【例4-14】

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

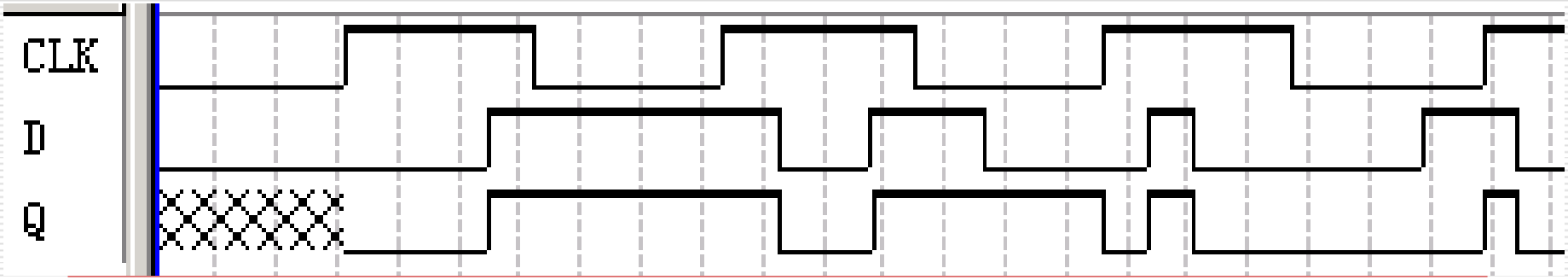


图4-8 例4-14的时序波形

4.2 寄存器描述的VHDL程序

4.2.4 异步时序电路设计

【例4-15】

```
 ...
  ARCHITECTURE bhv OF MULTI_DFF IS
    SIGNAL Q1,Q2 : STD_LOGIC;
  BEGIN
PRO1: PROCESS (CLK)
     BEGIN
      IF  CLK'EVENT AND CLK='1'
        THEN  Q1 <= NOT (Q2 OR A);
      END IF;
     END PROCESS ;
PRO2: PROCESS (Q1)
     BEGIN
      IF  Q1'EVENT AND Q1='1'
        THEN  Q2 <= D;
      END IF;
     END PROCESS ;
QQ <= Q2 ;
   ...
```

4.2 寄存器描述的VHDL程序

4.2.4 异步时序电路设计

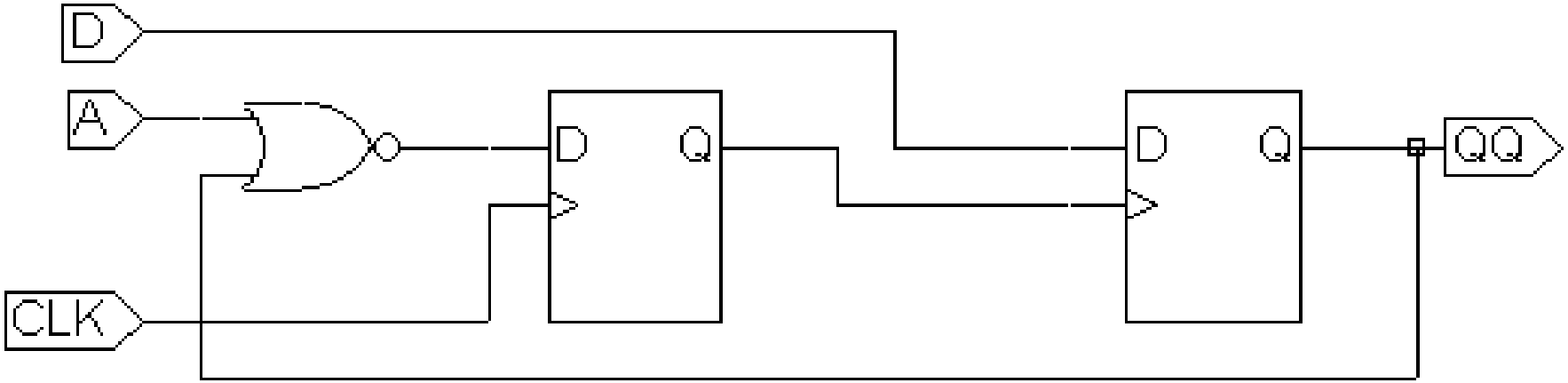


图4-9 例4-15综合后的电路（Synplify综合）

4.3 1位全加器的VHDL描述

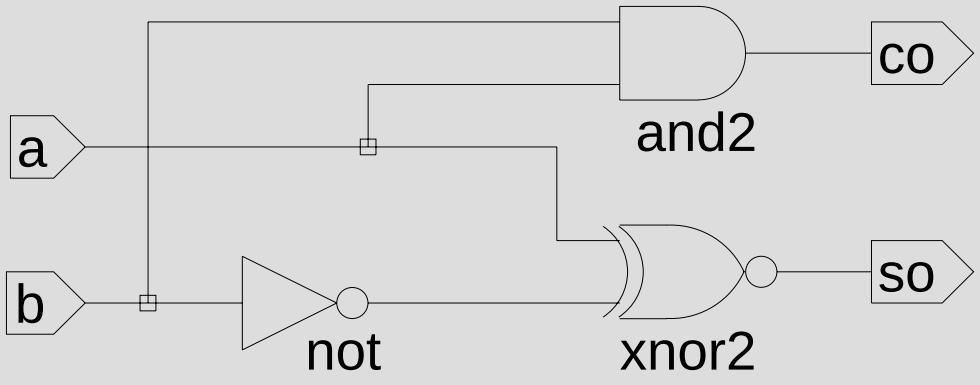
4.3.1 半加器描述

【例4-16】

```
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;
```

4.3 1位全加器的VHDL描述

4.3.1 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4-10 半加器h_adder电路图及其真值表

4.3 1位全加器的VHDL描述

4.3.1 半加器描述

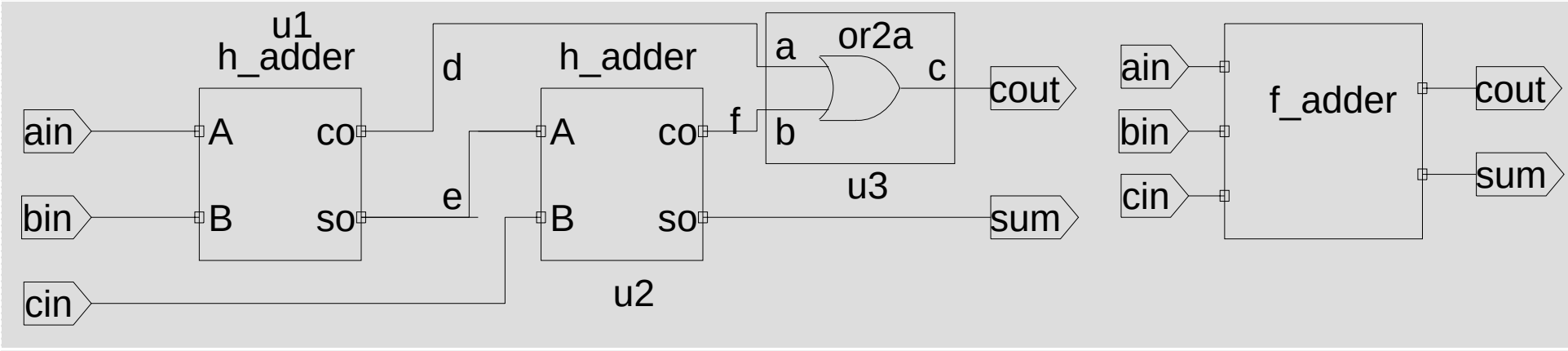


图4-11 全加器f_adder电路图及其实体模块

【例4-17】

```
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 ;
```

4.3 1位全加器的VHDL描述

4.3.1 半加器描述

【例4-18】

```
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4-19】

```
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;
```

4.3 1位全加器的VHDL描述

4.3.2 CASE语句

1. CASE语句

```
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 ;
```

WHEN OTHERS => <顺序语句>;

4.3 1位全加器的VHDL描述

4.3.2 CASE语句

2. 标准逻辑矢量数据类型

```
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);
```

4.3 1位全加器的VHDL描述

4.3.2 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
```

4.3 1位全加器的VHDL描述

4.3.3 全加器描述和例化语句

```
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,...);

4.4 计数器设计

【例4-20】

```
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;
```

4.4 计数器设计

4.4.1 4位二进制加法计数器设计

4.4.2 整数类型

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

4.4 计数器设计

4.4.3 计数器设计的其他表述方法

【例4-21】

```
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;
```

4.4 计数器设计

4.4.3 计数器设计的其他表述方法

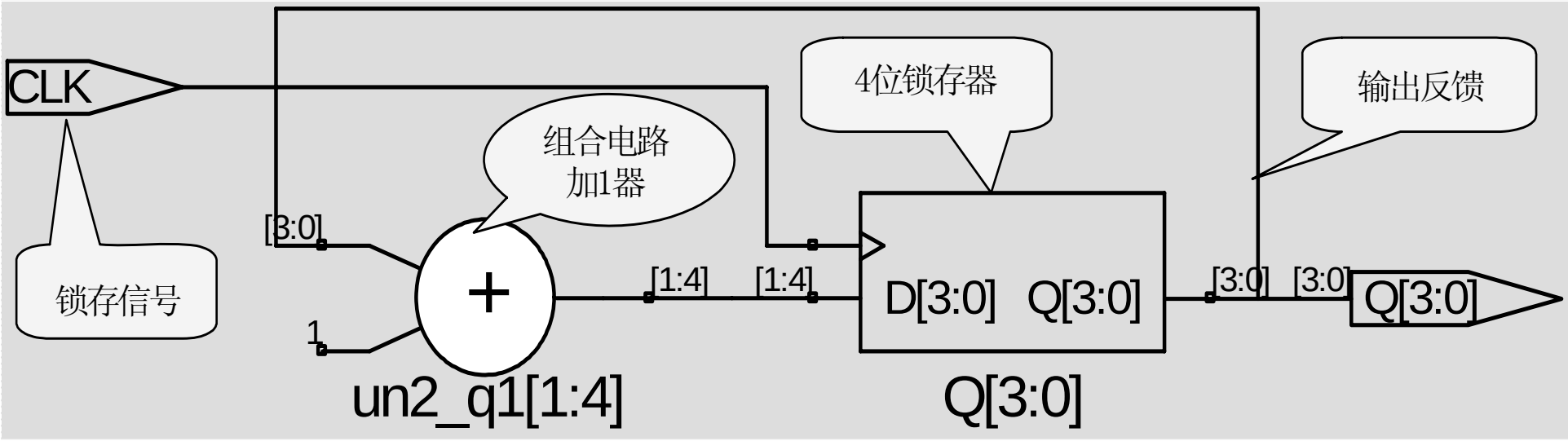


图4-12 4位加法计数器RTL电路（Synplify综合）

4.4 计数器设计

4.4.3 计数器设计的其他表述方法

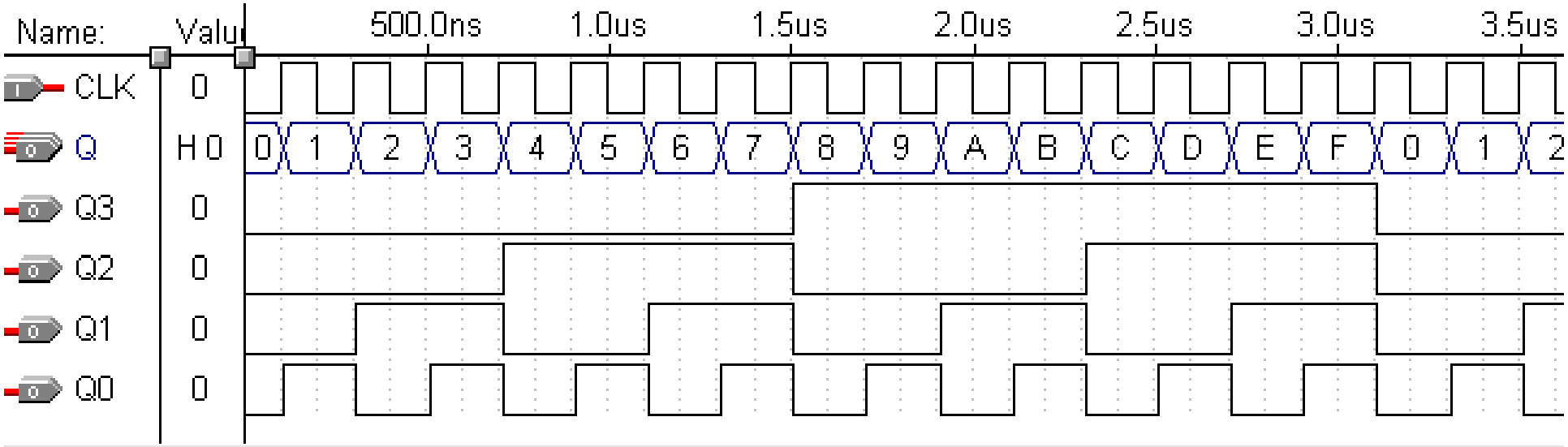


图4-13 4位加法计数器工作时序

4.5 一般加法计数器设计

【例4-22】

```
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
      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');          --大于9，计数值清零
         END IF;
       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;
```

4.5 一般加法计数器设计

4.5.1 相关语法说明

1. 变量　　“:= ”　　a:= ’1’

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) ;
```

4.5 一般加法计数器设计

4.5.2 程序分析

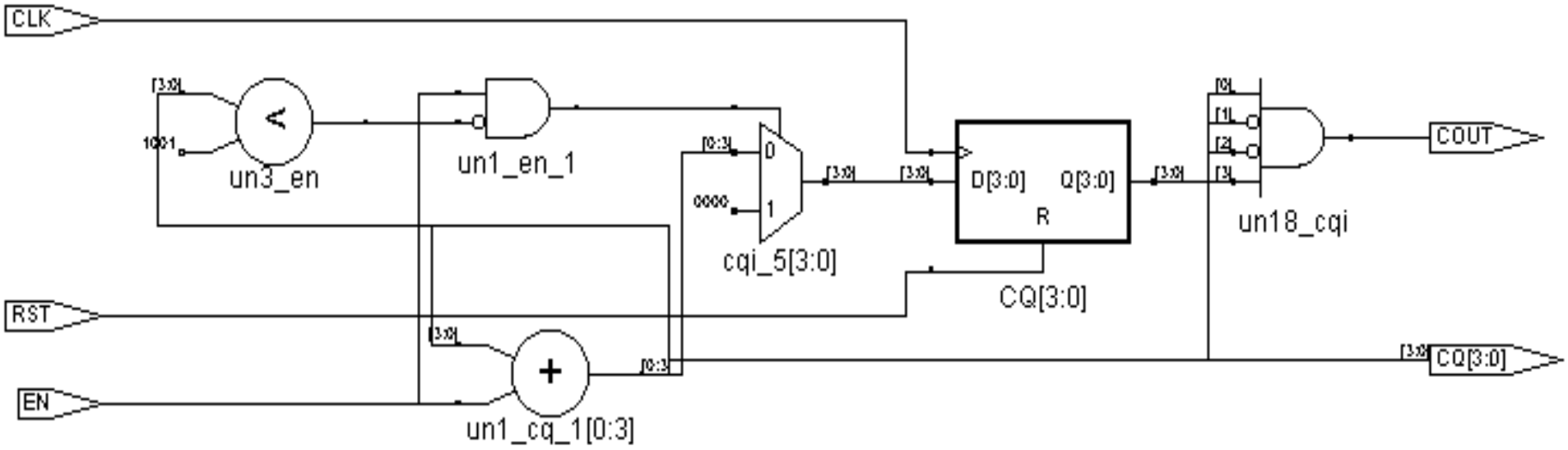


图4-14 例4-22的RTL电路（Synplify综合）

4.5 一般加法计数器设计

4.5.2 程序分析

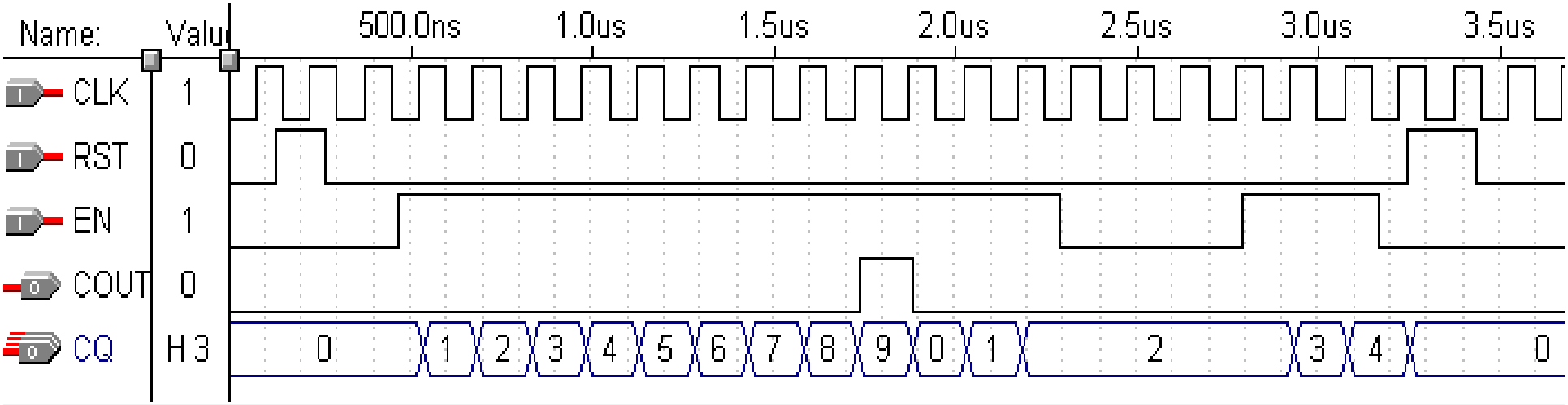


图4-15 例4-22的工作时序

4.5 一般加法计数器设计

4.5.3 含并行置位的移位寄存器设计

【例4-23】

```
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;
```

4.5 一般加法计数器设计

4.5.3 含并行置位的移位寄存器设计

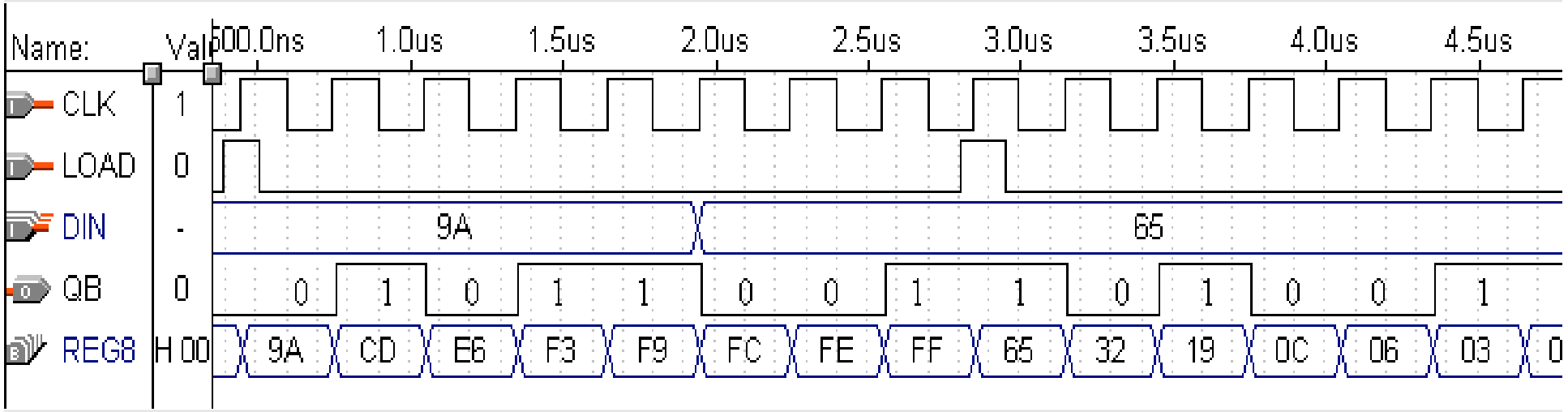


图4-16 例4-23的工作时序

4.6 基于Quartus II的VHDL文本输入设计

4.6.1 建立工作库文件夹和编辑设计文件

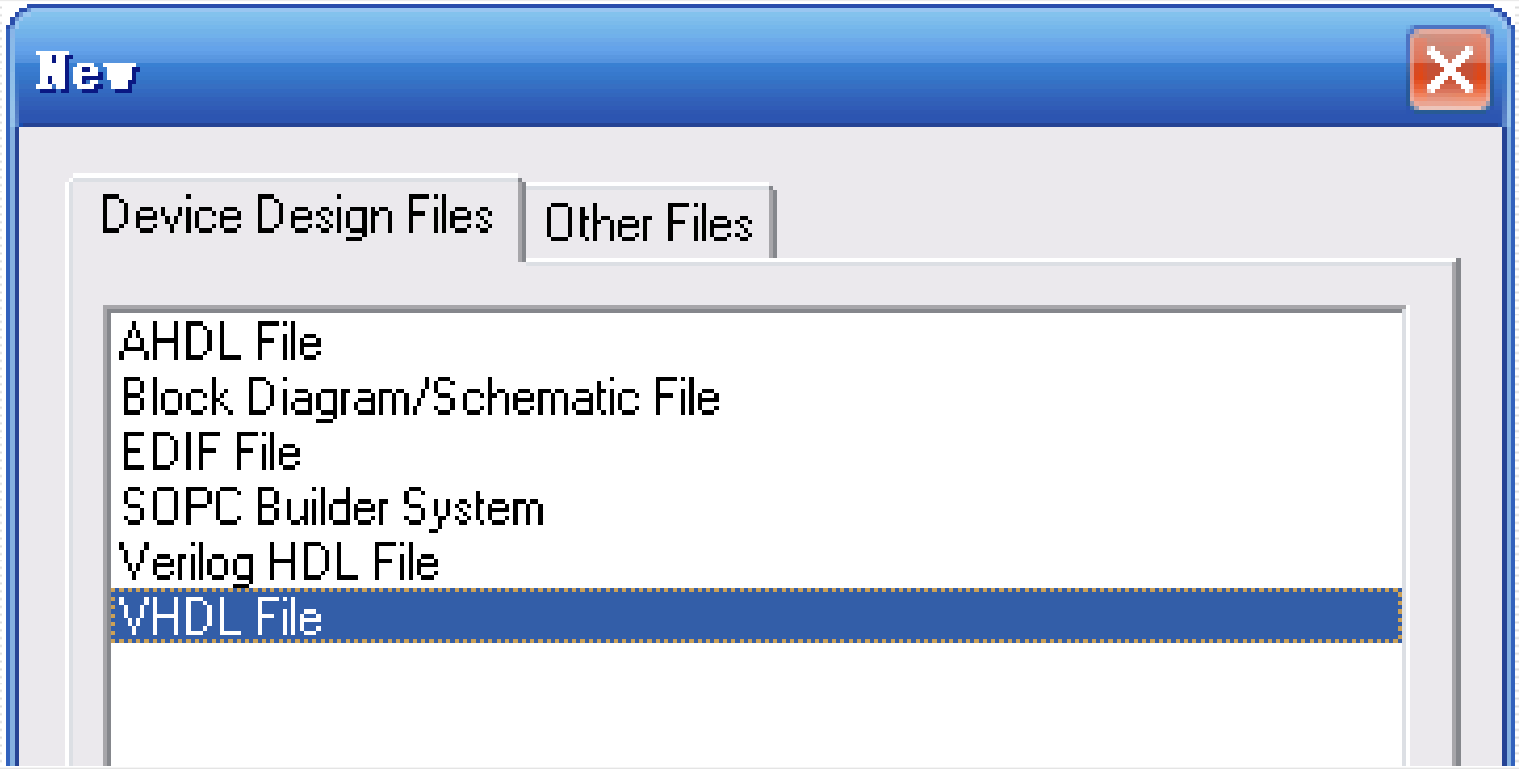


图4-17 选择VHDL文本编辑器

4.6 基于Quartusll的VHDL文本输入设计

4.6.1 建立工作库文件夹和编辑设计文件

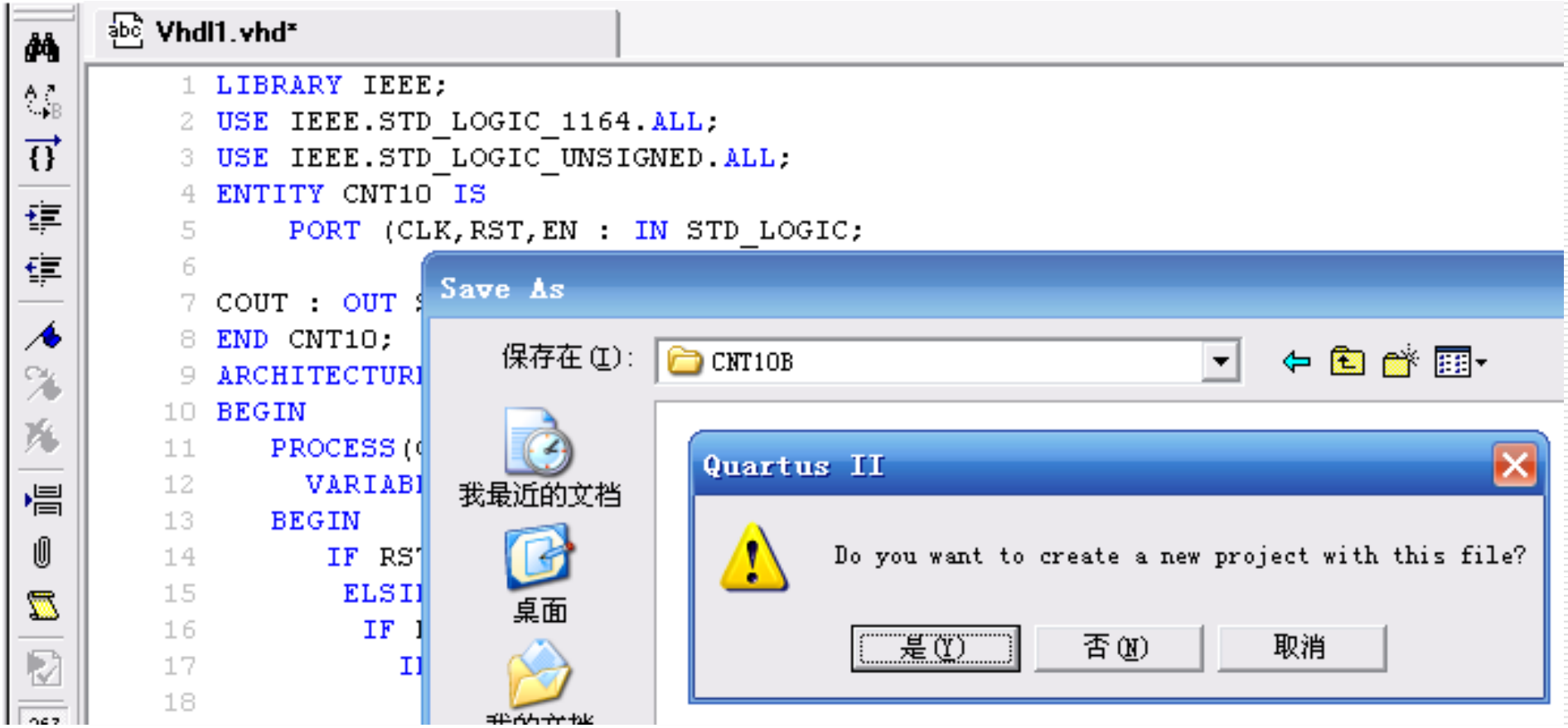


图4-18 选择编辑文件的语言类型，键入源程序并存盘

4.6 基于Quartusll的VHDL文本输入设计

4.6.2 创建工程

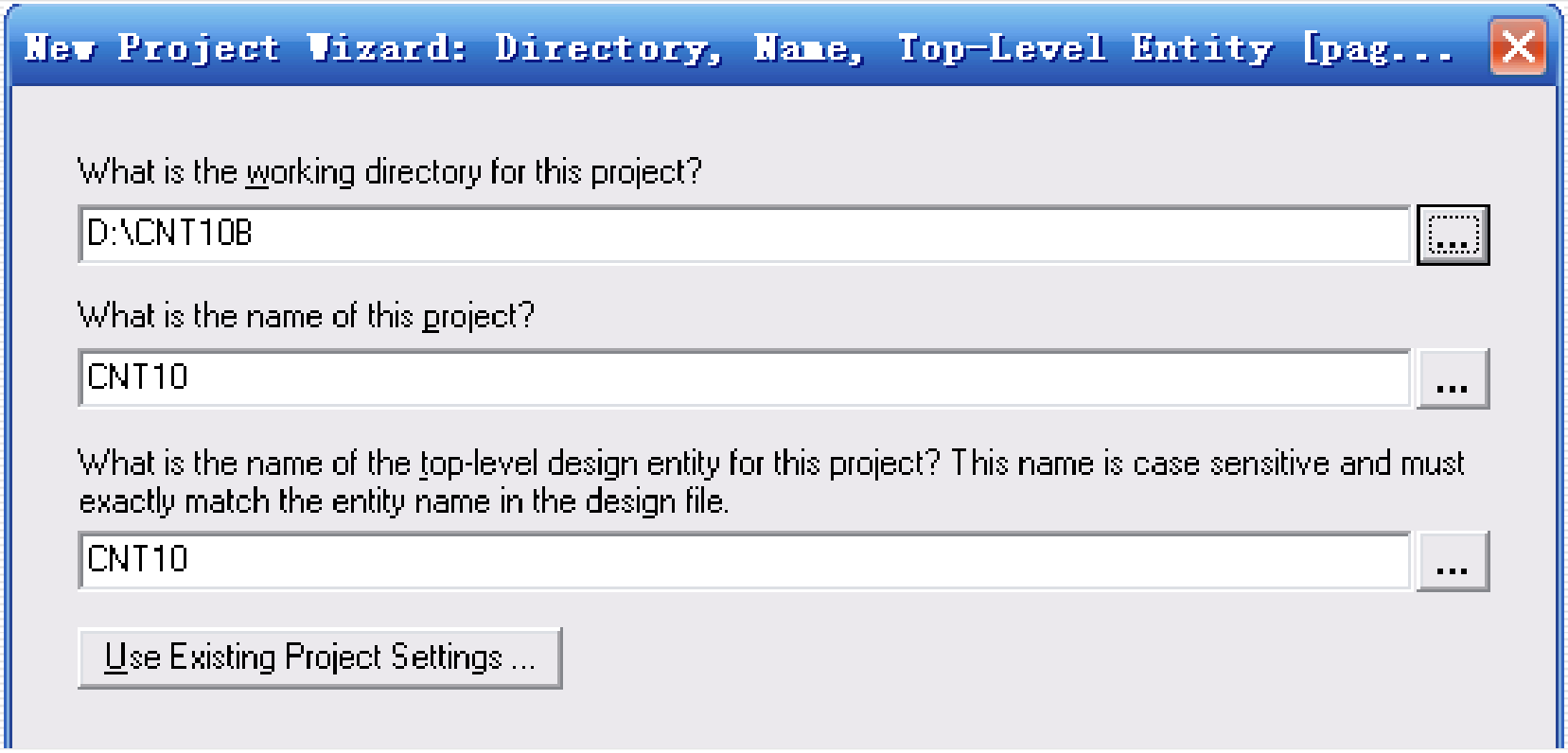


图4-19 利用“New Preject Wizard”创建工程cnt10

4.6 基于Quartusll的VHDL文本输入设计

4.6.2 创建工程

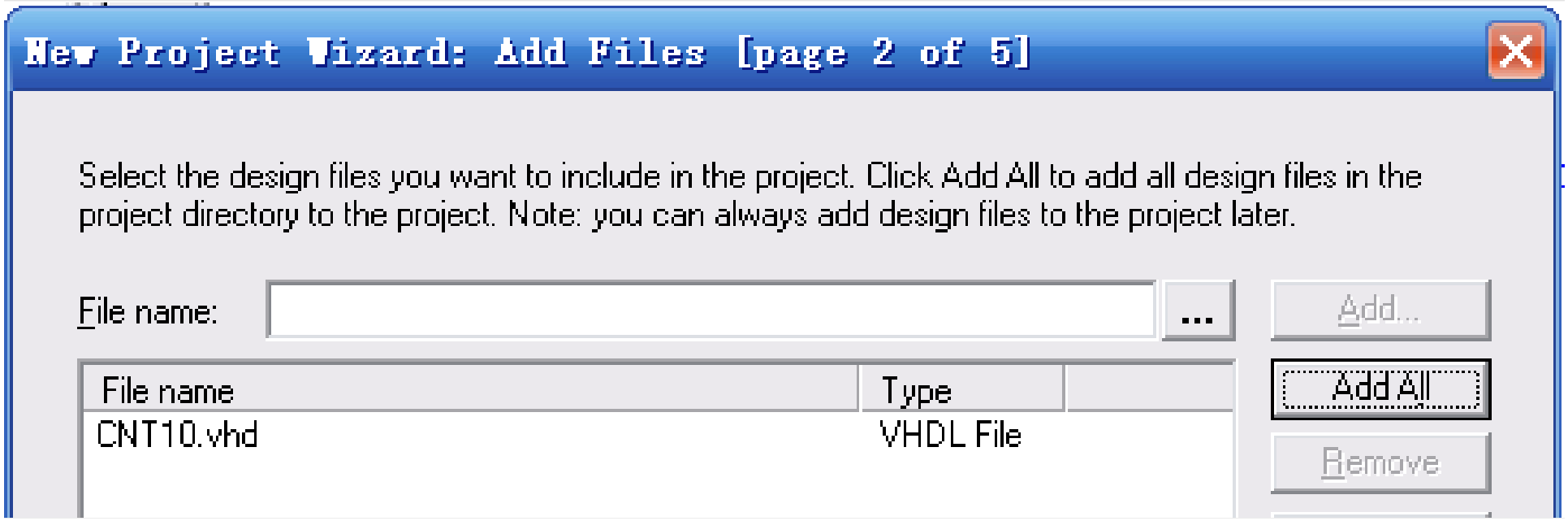


图4-20 将所有相关的文件都加入进此工程

4.6 基于Quartusll的VHDL文本输入设计

4.6.3 全程编译

4.6.4 时序仿真

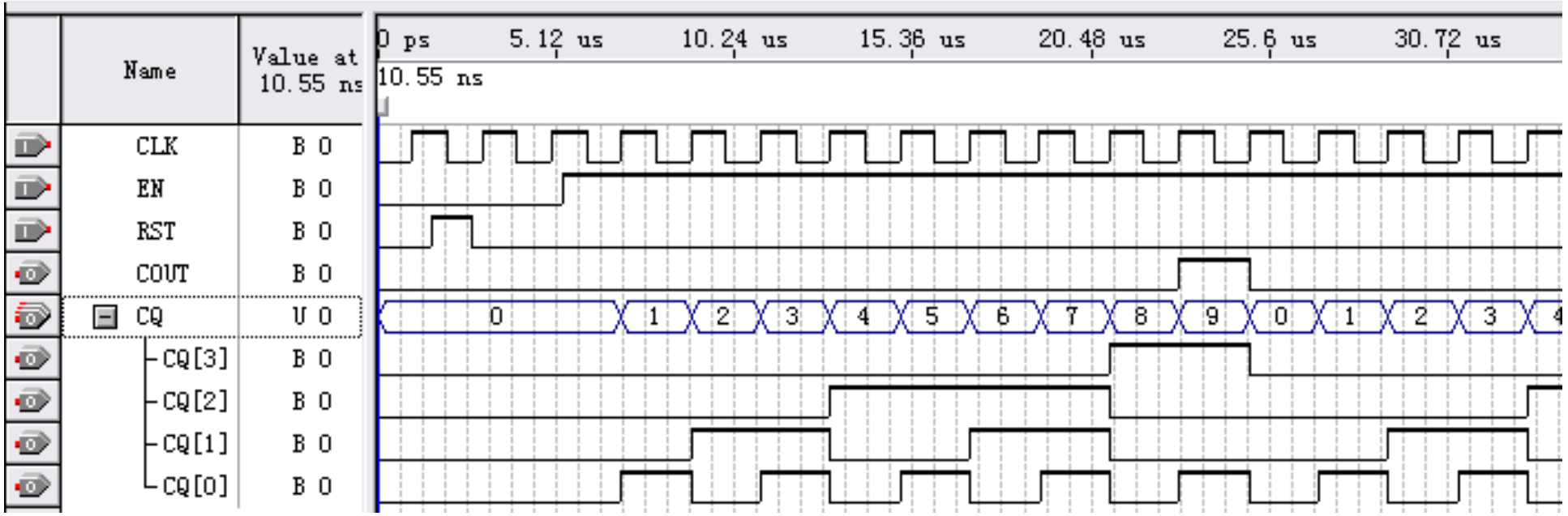


图4-21例4-22的时序仿真波形图

4.6 基于Quartusll的VHDL文本输入设计

4.6.5 应用RTL电路图观察器

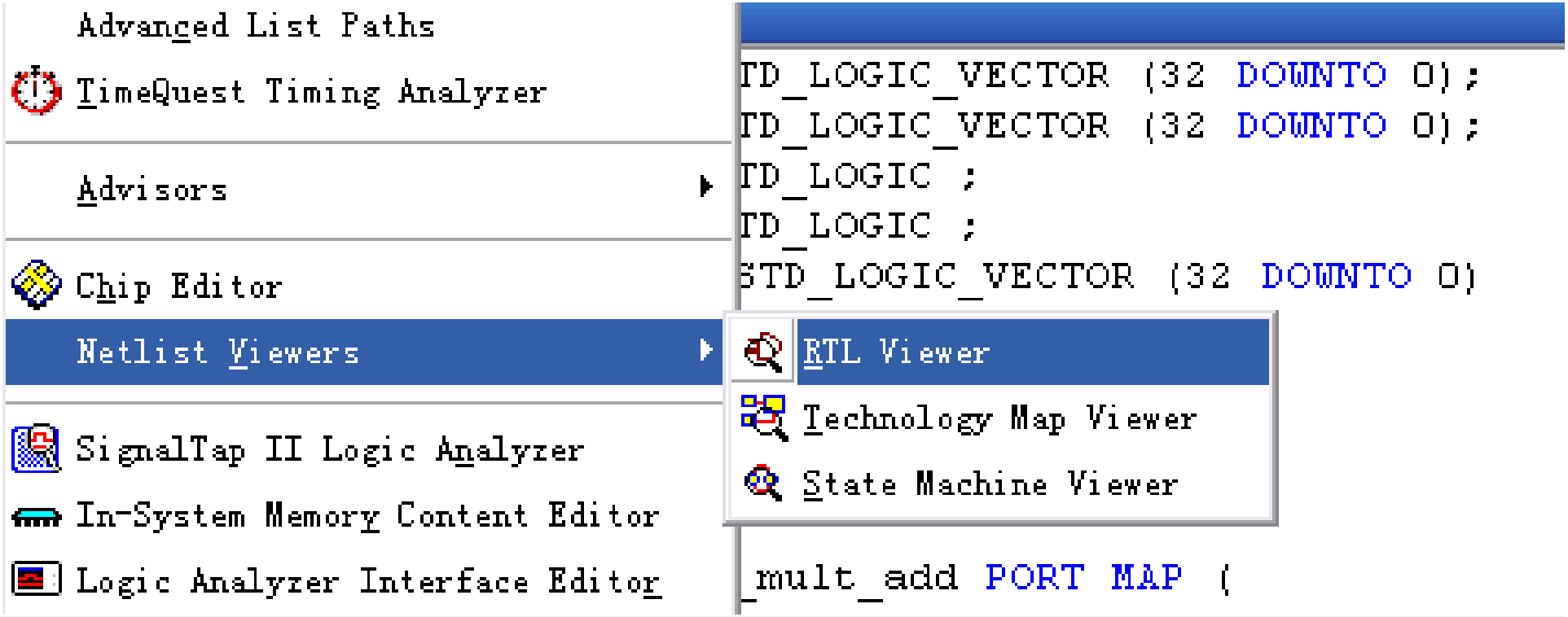


图4-22 RTL Viewer选择

4.6 基于Quartusll的VHDL文本输入设计

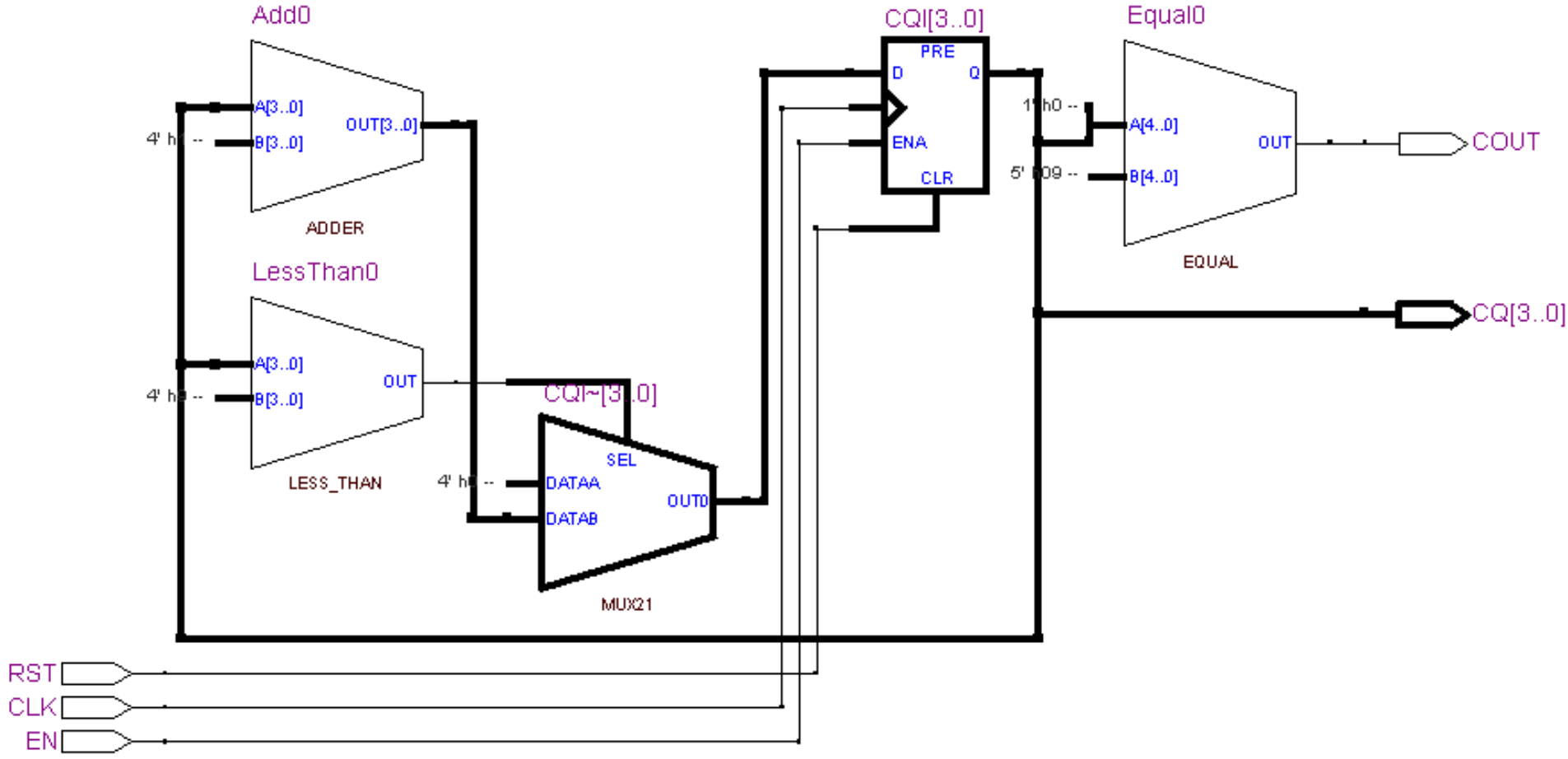


图4-23 例4-22的RTL图

4.6 基于Quartusll的VHDL文本输入设计

4.6.6 硬件测试器

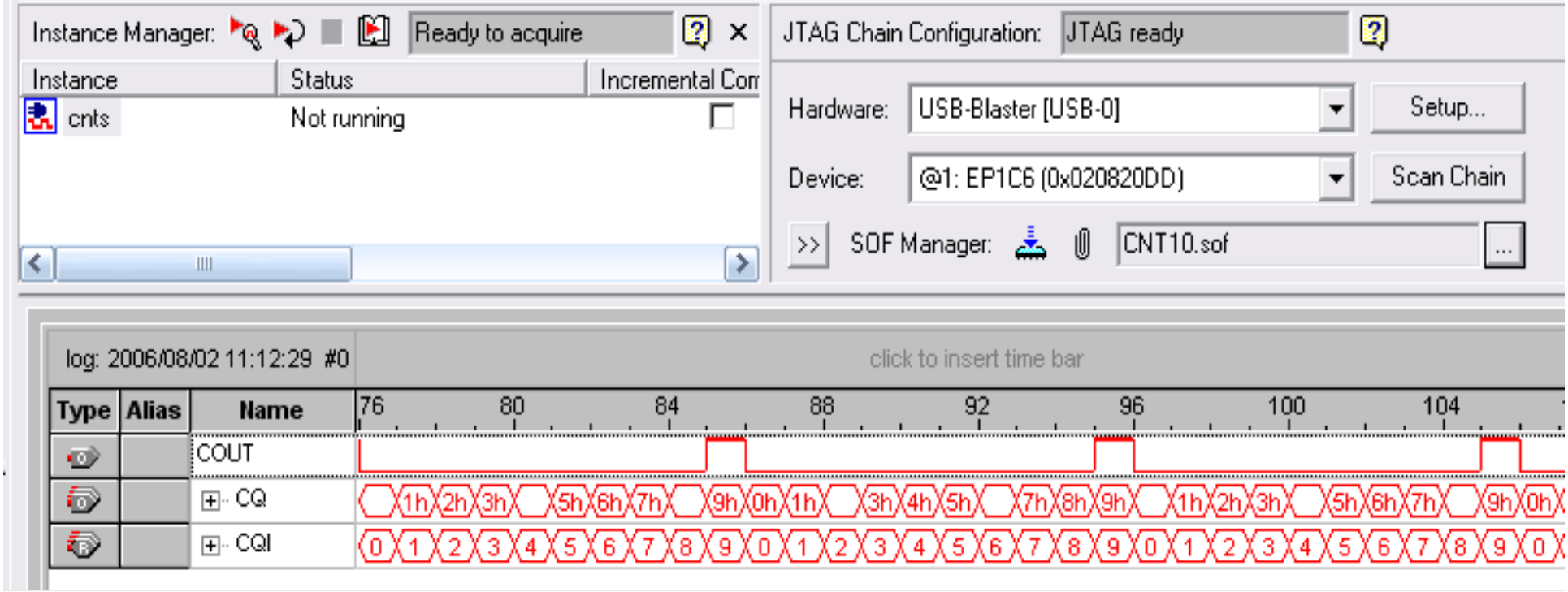


图4-24下载cnt10.sof并启动SignalTap II，测试输出数据

4.6 基于QuartusII的VHDL文本输入设计

4.6.6 硬件测试器

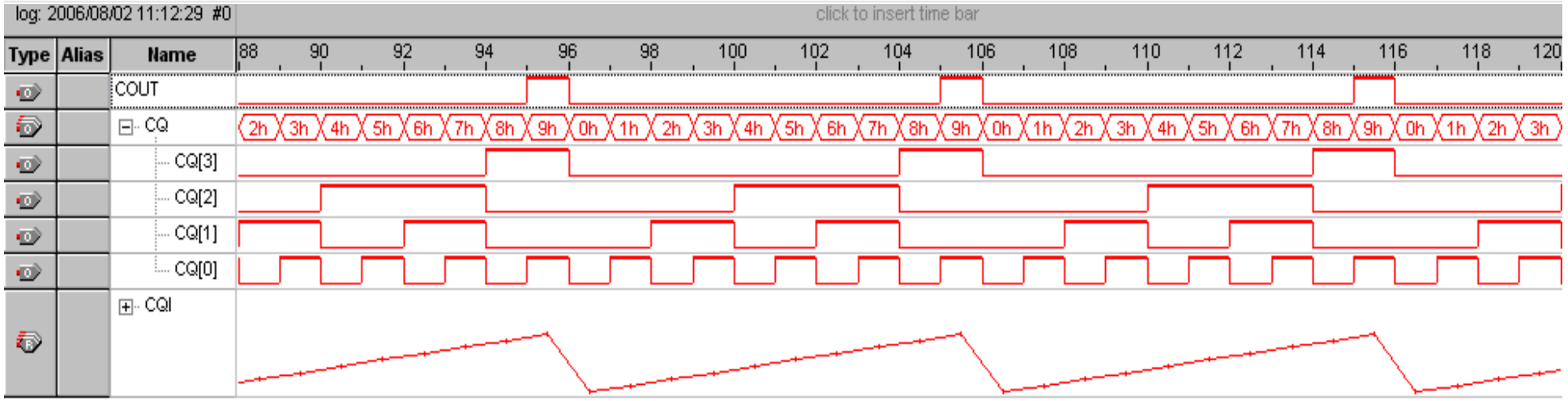


图4-25 SignalTap II数据窗设置后的信号波形

习 题

4-1. 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

4-2. 图4-26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。

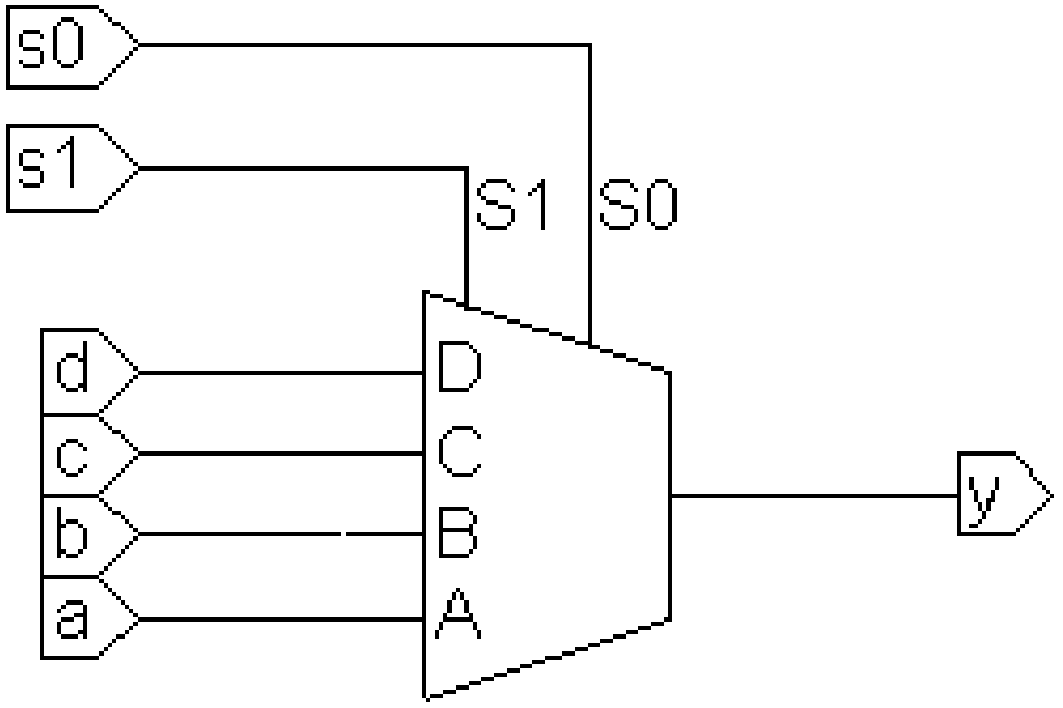


图4-26 4选1多路选择器

习 题

4-3. 图4-27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。

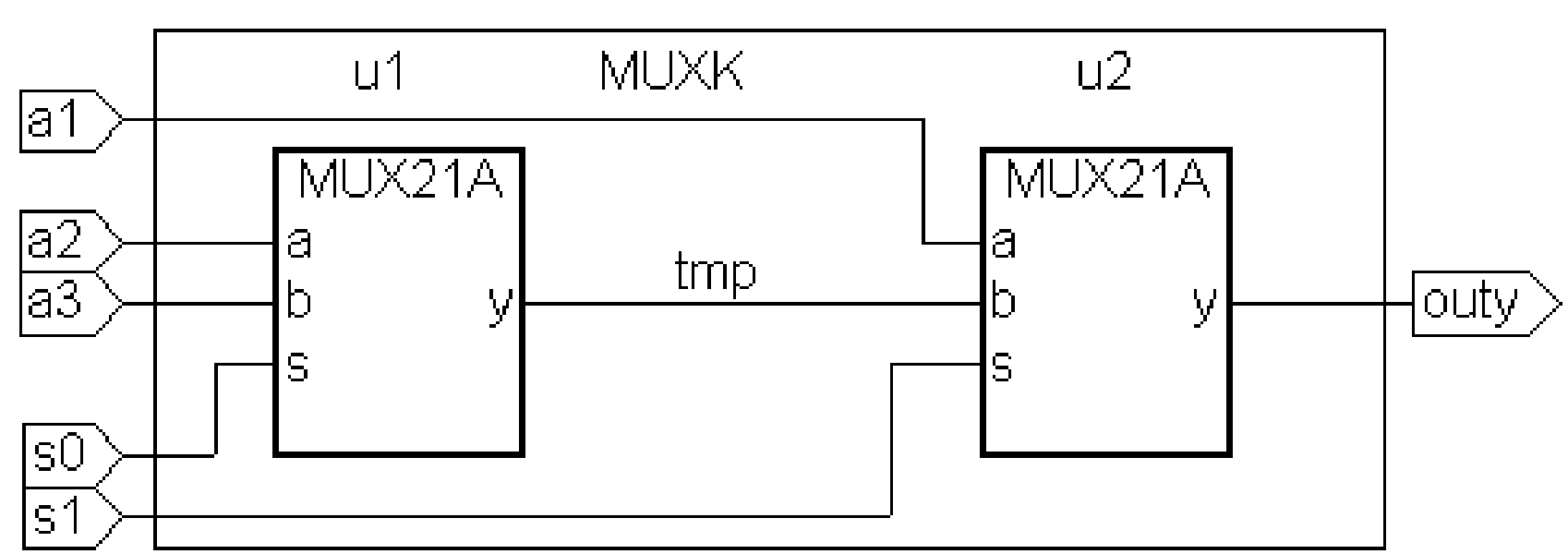


图4-27 双2选1多路选择器

习 题

4-4. 图4-28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。

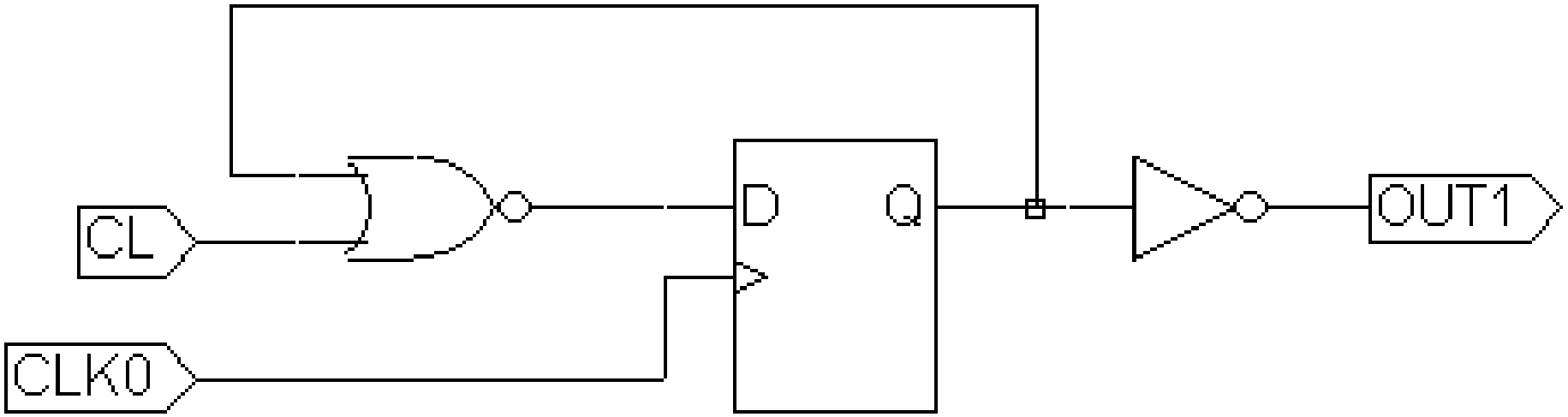


图4-28 时序电路图

习 题

4-5. 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4-29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)。

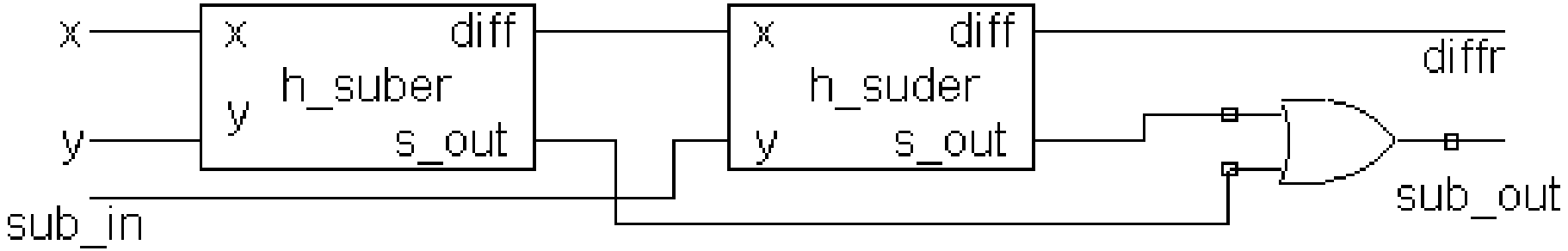


图4-29 1位全减器

习 题

4-6. 根据图**4-30**，写出顶层文件**MX3256.VHD**的**VHDL**设计文件。

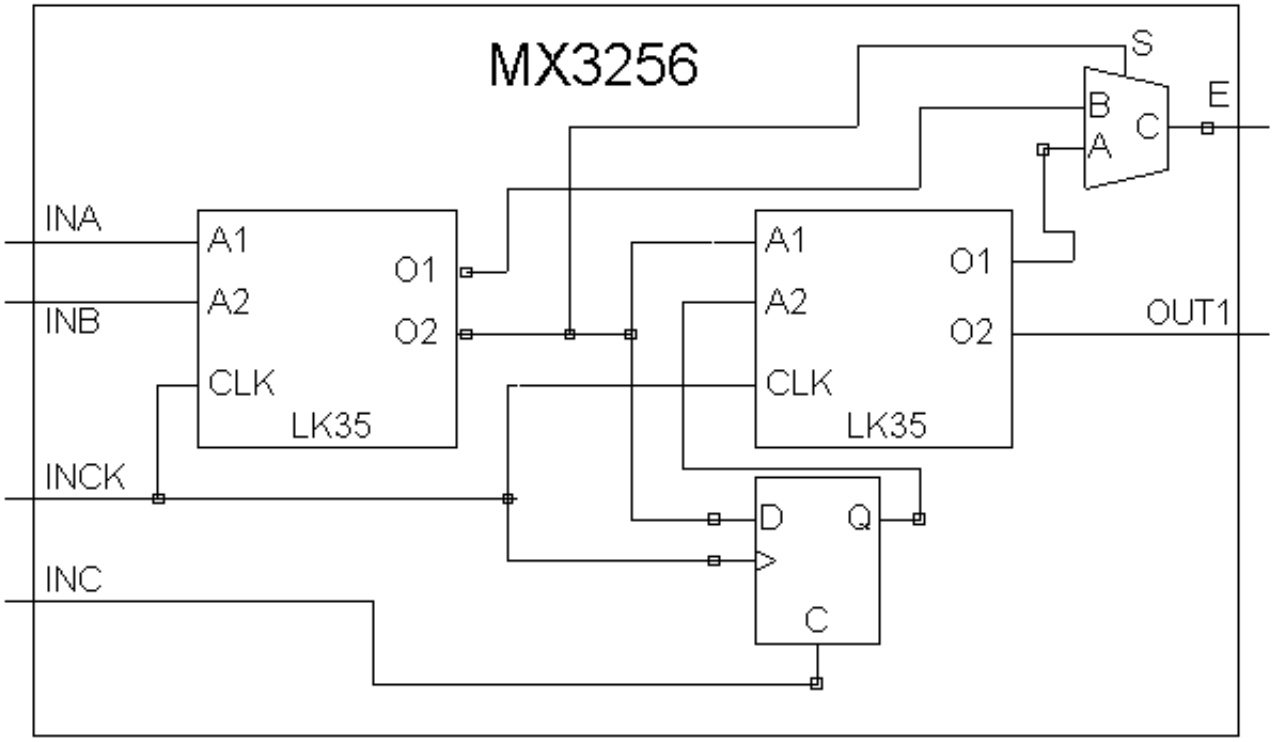


图**4-30** 题**4-6**电路图

4-7. 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。

实验与实践

4-1. 基于VHDL的组合电路的设计

```
…
  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；
```

实验与实践

4-2. 时序电路的设计

4-3. 含异步清**0**和同步时钟使能的加法计数器的设计

4-4. 数控分频器的设计

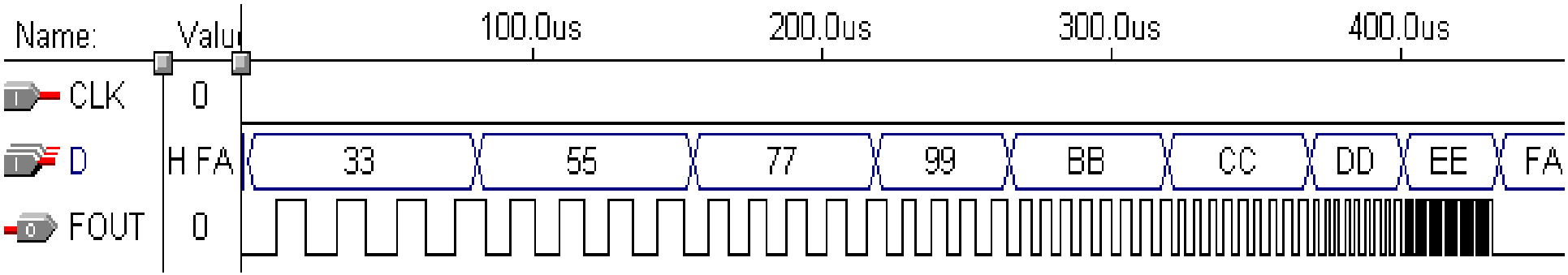


图**4-31** 当给出不同输入值**D**时，**FOUT**输出不同频率**(CLK**周期**=50ns)**

【例4-24】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY DVF IS
   PORT (   CLK  : IN STD_LOGIC;
            D  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
           FOUT : OUT STD_LOGIC  );
END;
ARCHITECTURE one OF DVF IS
   SIGNAL   FULL : STD_LOGIC;
BEGIN
  P_REG: PROCESS(CLK)
   VARIABLE CNT8 : STD_LOGIC_VECTOR(7 DOWNTO 0);
   BEGIN
     IF CLK'EVENT AND CLK = '1' THEN
         IF CNT8 = "11111111" THEN
         CNT8 := D;     --当CNT8计数计满时，输入数据D被同步预置给计数器CNT8
          FULL <= '1'; --同时使溢出标志信号FULL输出为高电平
            ELSE   CNT8 := CNT8 + 1;  --否则继续作加1计数
                  FULL <= '0';       --且输出溢出标志信号FULL为低电平
         END IF;
     END IF;
    END PROCESS P_REG ;
   P_DIV: PROCESS(FULL)
    VARIABLE CNT2 : STD_LOGIC;
   BEGIN
   IF FULL'EVENT AND FULL = '1' THEN
    CNT2 := NOT CNT2; --如果溢出标志信号FULL为高电平，D触发器输出取反
       IF CNT2 = '1' THEN  FOUT <= '1'; ELSE FOUT <= '0';
      END IF;
   END IF;
    END PROCESS P_DIV ;
END;
```

实验与实践

4-5 乐曲硬件演奏电路设计

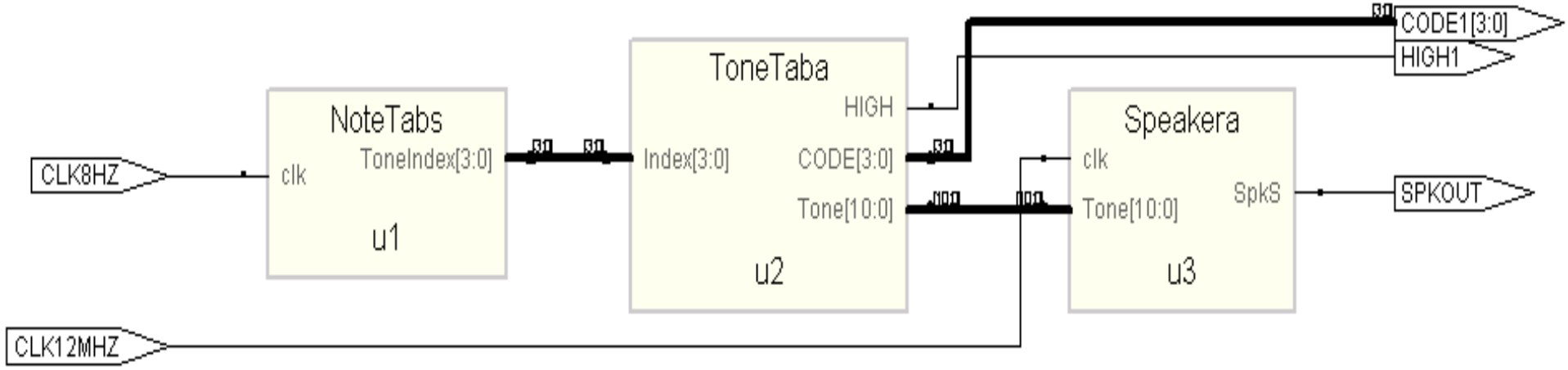


图4-32 硬件乐曲演奏电路结构（Synplify综合）

【例4-25】

```
LIBRARY IEEE; -- 硬件演奏电路顶层设计
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY Songer IS
   PORT (   CLK12MHZ : IN STD_LOGIC;          --音调频率信号
         CLK8HZ   : IN STD_LOGIC;            --节拍频率信号
         CODE1  : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);-- 简谱码输出显示
          HIGH1  : OUT STD_LOGIC; --高8度指示
         SPKOUT   : OUT STD_LOGIC );--声音输出
 END;
ARCHITECTURE one OF Songer IS
  COMPONENT NoteTabs
   PORT ( clk    : IN STD_LOGIC;
      ToneIndex : OUT STD_LOGIC_VECTOR (3 DOWNTO 0) );
   END COMPONENT;
   COMPONENT ToneTaba
      PORT ( Index :  IN  STD_LOGIC_VECTOR (3 DOWNTO 0) ;
           CODE  : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0) ;
           HIGH  : OUT STD_LOGIC;
           Tone  : OUT  STD_LOGIC_VECTOR (10 DOWNTO 0) );
   END COMPONENT;
   COMPONENT Speakera
      PORT  ( clk  : IN STD_LOGIC;
            Tone : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
            SpkS : OUT STD_LOGIC  );
   END COMPONENT;
   SIGNAL Tone : STD_LOGIC_VECTOR (10 DOWNTO 0);
   SIGNAL ToneIndex : STD_LOGIC_VECTOR (3 DOWNTO 0);
 BEGIN
u1 : NoteTabs  PORT MAP (clk=>CLK8HZ, ToneIndex=>ToneIndex);
u2 : ToneTaba PORT MAP (Index=>ToneIndex,Tone=>Tone,CODE=>CODE1,HIGH=>HIGH1);
u3 : Speakera PORT MAP(clk=>CLK12MHZ,Tone=>Tone, SpkS=>SPKOUT );
END
```

【例4-26】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY Speakera IS
   PORT (   clk  : IN STD_LOGIC;
         Tone : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
         SpkS : OUT STD_LOGIC  );
END;
ARCHITECTURE one OF Speakera IS
   SIGNAL PreCLK, FullSpkS : STD_LOGIC;
BEGIN
 DivideCLK : PROCESS(clk)
      VARIABLE Count4 : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
   BEGIN
      PreCLK <= '0';  --  将CLK进行16分频，PreCLK为CLK的16分频
      IF Count4>11 THEN PreCLK <= '1';  Count4 := "0000";
      ELSIF clk'EVENT AND clk = '1' THEN  Count4 := Count4 + 1;
      END IF;
   END PROCESS;
   GenSpkS : PROCESS(PreCLK, Tone)-- 11位可预置计数器
      VARIABLE Count11 : STD_LOGIC_VECTOR (10 DOWNTO 0);
BEGIN
     IF PreCLK'EVENT AND PreCLK = '1' THEN
      IF Count11 = 16#7FF# THEN Count11 := Tone ; FullSpkS <= '1';
         ELSE Count11 := Count11 + 1; FullSpkS <= '0'; END IF;
      END IF;
  END PROCESS;
 DelaySpkS : PROCESS(FullSpkS)--将输出再2分频，展宽脉冲，使扬声器有足够功率发音
      VARIABLE Count2 : STD_LOGIC;
BEGIN
   IF FullSpkS'EVENT AND FullSpkS = '1' THEN  Count2 := NOT Count2;
         IF Count2 = '1' THEN  SpkS <= '1';
         ELSE SpkS <= '0';  END IF;
      END IF;
   END PROCESS;
END;
```

【例4-27】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ToneTaba IS
   PORT ( Index :  IN  STD_LOGIC_VECTOR (3 DOWNTO 0) ;
        CODE  : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0) ;
        HIGH  : OUT STD_LOGIC;
        Tone  : OUT  STD_LOGIC_VECTOR (10 DOWNTO 0) );
END;
ARCHITECTURE one OF ToneTaba IS
BEGIN
   Search : PROCESS(Index)
   BEGIN
      CASE Index IS    -- 译码电路，查表方式，控制音调的预置数
 WHEN "0000" => Tone<="11111111111"; CODE<="0000"; HIGH <='0';--2047
 WHEN "0001" => Tone<="01100000101"; CODE<="0001"; HIGH <='0';--773;
 WHEN "0010" => Tone<="01110010000"; CODE<="0010"; HIGH <='0';--912;
 WHEN "0011" => Tone<="10000001100"; CODE<="0011"; HIGH <='0';--1036;
 WHEN "0101" => Tone<="10010101101"; CODE<="0101"; HIGH <='0';--1197;
 WHEN "0110" => Tone<="10100001010"; CODE<="0110"; HIGH <='0';--1290;
 WHEN "0111" => Tone<="10101011100"; CODE<="0111"; HIGH <='0';--1372;
 WHEN "1000" => Tone<="10110000010"; CODE<="0001"; HIGH <='1';--1410;
 WHEN "1001" => Tone<="10111001000"; CODE<="0010"; HIGH <='1';--1480;
 WHEN "1010" => Tone<="11000000110"; CODE<="0011"; HIGH <='1';--1542;
 WHEN "1100" => Tone<="11001010110"; CODE<="0101"; HIGH <='1';--1622;
 WHEN "1101" => Tone<="11010000100"; CODE<="0110"; HIGH <='1';--1668;
 WHEN "1111" => Tone<="11011000000"; CODE<="0001"; HIGH <='1';--1728;
 WHEN OTHERS => NULL;
 END CASE;
 END PROCESS;
END;
```

【例4-28】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY NoteTabs IS
   PORT ( clk    : IN STD_LOGIC;
      ToneIndex : OUT STD_LOGIC_VECTOR (3 DOWNTO 0) );
END;
ARCHITECTURE one OF NoteTabs IS
COMPONENT MUSIC              --音符数据ROM
 PORT(address : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
           inclock : IN STD_LOGIC ;
                q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END COMPONENT;
   SIGNAL Counter :  STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
   CNT8 : PROCESS(clk, Counter)
   BEGIN
      IF Counter=138 THEN  Counter <= "00000000";
      ELSIF (clk'EVENT AND clk = '1') THEN Counter <= Counter+1;
 END IF;
   END PROCESS;
 u1 : MUSIC PORT MAP(address=>Counter , q=>ToneIndex,
inclock=>clk);
 END;
```

实验与实践

【例4-29】

```
WIDTH = 4 ; --“梁祝”乐曲演奏数据
DEPTH = 256 ;
ADDRESS_RADIX = DEC ;
DATA_RADIX = DEC ;
CONTENT BEGIN --注意实用文件中要展开以下数据，每一组占一行
00: 3 ; 01: 3 ; 02: 3 ; 03: 3; 04: 5; 05: 5; 06:  5;07: 6; 08: 8; 09: 8;
10: 8 ; 11: 9 ; 12: 6 ; 13: 8; 14: 5; 15: 5; 16: 12;17: 12;18: 12; 19:15;
20:13 ; 21:12 ; 22:10 ; 23:12; 24: 9; 25: 9; 26: 9; 27: 9; 28: 9; 29: 9;
30: 9 ; 31: 0 ; 32: 9 ; 33: 9; 34: 9; 35:10; 36: 7; 37: 7; 38: 6; 39: 6;
40: 5 ; 41: 5 ; 42: 5 ; 43: 6; 44: 8; 45: 8; 46: 9; 47: 9; 48: 3; 49: 3;
50: 8 ; 51: 8 ; 52: 6 ; 53: 5; 54: 6; 55: 8; 56: 5; 57: 5; 58: 5; 59: 5;
60: 5 ; 61: 5 ; 62: 5 ; 63: 5; 64:10; 65:10; 66:10; 67:12; 68: 7; 69: 7;
70: 9 ; 71: 9 ; 72: 6 ; 73: 8; 74: 5; 75: 5; 76: 5; 77: 5; 78: 5; 79: 5;
80: 3 ; 81: 5 ; 82: 3 ; 83: 3; 84: 5; 85: 6; 86: 7; 87: 9; 88: 6; 89: 6;
90: 6 ; 91: 6 ; 92: 6 ; 93: 6; 94: 5; 95: 6; 96: 8; 97: 8; 98: 8; 99: 9;
100:12 ;101:12 ;102:12 ;103:10;104: 9;105: 9;106:10;107: 9;108: 8;109: 8;
110: 6 ;111: 5 ;112: 3 ;113: 3;114: 3;115: 3;116: 8;117: 8;118: 8;119: 8;
120: 6 ;121: 8 ;122: 6 ;123: 5;124: 3;125: 5;126: 6;127: 8;128: 5;129: 5;
130: 5 ;131: 5 ;132: 5 ;133: 5;134: 5;135: 5;136: 0;137: 0;138: 0;
END ;
```

实验与实践

4-6. 混合输入设计实验

【例4-30】

```
LIBRARY IEEE;  --测频控制电路
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY FTCTRL IS
   PORT (CLKK : IN STD_LOGIC;              -- 1Hz
      CNT_EN : OUT STD_LOGIC;              -- 计数器时钟使能
     RST_CNT : OUT STD_LOGIC;               -- 计数器清零
      Load : OUT STD_LOGIC    );           -- 输出锁存信号
 END FTCTRL;
ARCHITECTURE behav OF FTCTRL IS
   SIGNAL Div2CLK : STD_LOGIC;
BEGIN
   PROCESS( CLKK )
   BEGIN
      IF CLKK'EVENT AND CLKK = '1' THEN     -- 1Hz时钟2分频
         Div2CLK <= NOT Div2CLK;
      END IF;
   END PROCESS;
   PROCESS (CLKK, Div2CLK)
   BEGIN
      IF CLKK='0' AND Div2CLK='0' THEN RST_CNT<='1';-- 产生计数器清零信号
       ELSE RST_CNT <= '0';  END IF;
   END PROCESS;
   Load  <= NOT Div2CLK;   CNT_EN <= Div2CLK;
END behav;
```

实验与实践

4-6. 混合输入设计实验

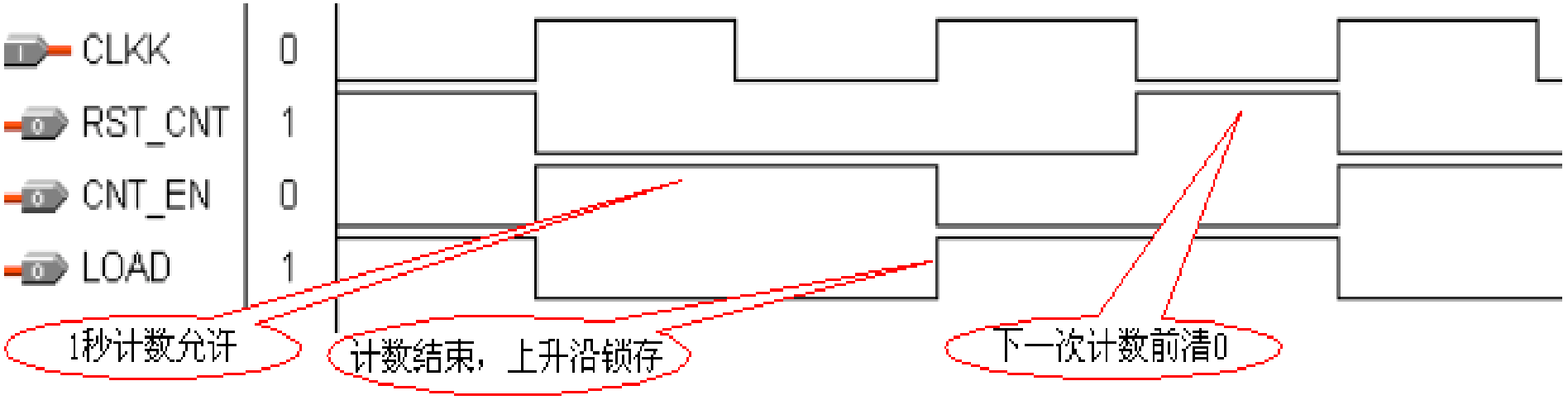


图4-32 频率计测频控制器FTCTRL测控时序图