EDA 技术实用教程

第 4 章

VHDL设计初步

4.1 多路选择器的VHDL描述

4.1.1 2选1多路选择器的VHDL描述

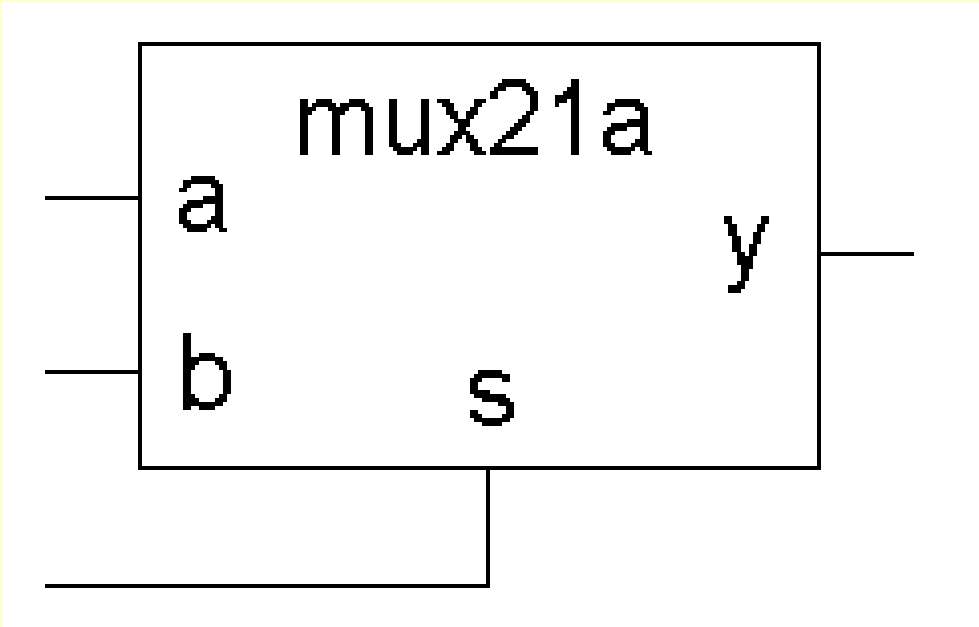


图4-1 mux21a实体

4.1 多路选择器的VHDL描述

4.1.1 2选1多路选择器的VHDL描述

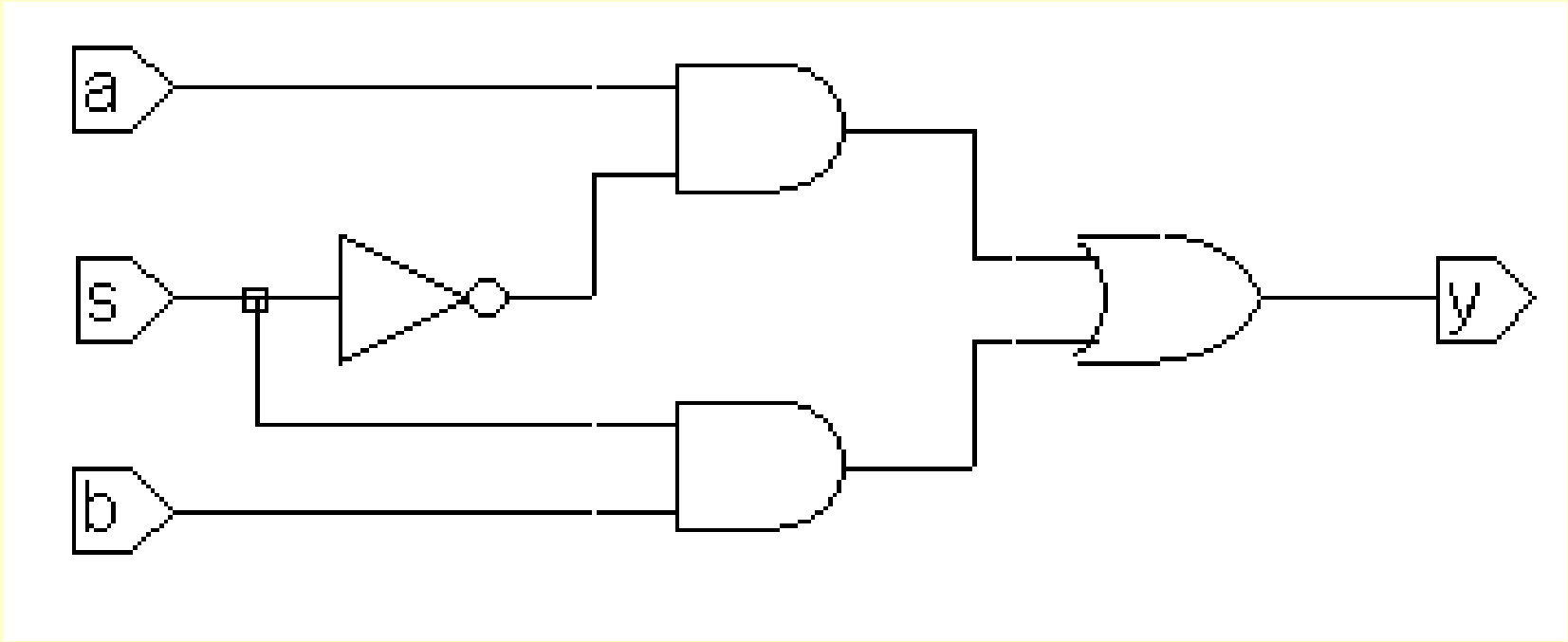


图4-2 mux21a结构体

4.1 多路选择器的VHDL描述

4.1.1 2选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 2选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 2选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 2选1多路选择器的VHDL描述

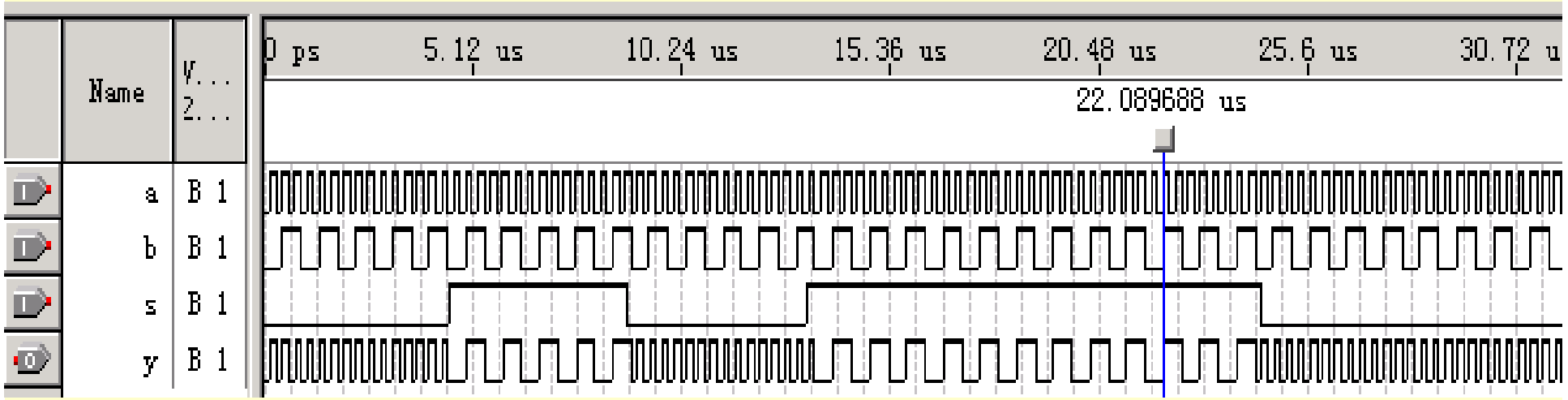


图4-3 mux21a功能时序波形

4.1 多路选择器的VHDL描述

4.1.2 相关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;
```

2. 实体名

3. 端口语句和端口信号名

4.1 多路选择器的VHDL描述

4.1.2 相关语句结构和语法说明

4. 端口模式

- **IN** 输入端口，定义的通道为单向只读模式
- **OUT** 输出端口，定义的通道为单向输出模式
- **INOUT** 定义的通道确定为输入输出双向端口
- **BUFFER** 缓冲端口，其功能与INOUT类似

4.1 多路选择器的VHDL描述

4.1.2 相关语句结构和语法说明

5. 数据类型

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

4.1.2 相关语句结构和语法说明

7. 赋值符号和数据比较符号

赋值符 “<=”

表式中的等号“=”没有赋值的含义，只是一种数据比较符号。

```
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 ..
```

4.1 多路选择器的VHDL描述

4.1.2 相关语句结构和语法说明

8. 逻辑操作符

AND、OR、NOT

9. 条件语句

IF_THEN_ELSE

IF语句必须以语句 **“END IF；”**结束

4.1 多路选择器的VHDL描述

4.1.2 相关语句结构和语法说明

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

4.1.2 相关语句结构和语法说明

11. 进程语句和顺序语句

在一个结构体中可以包含任意个进程语句结构，所有的进程语句都是并行语句，而由任一进程**PROCESS**引导的语句（包含在其中的语句）结构属于顺序语句。

12. 文件取名和存盘

4.2 寄存器描述及其VHDL语言现象

4.2.1 D触发器的VHDL描述

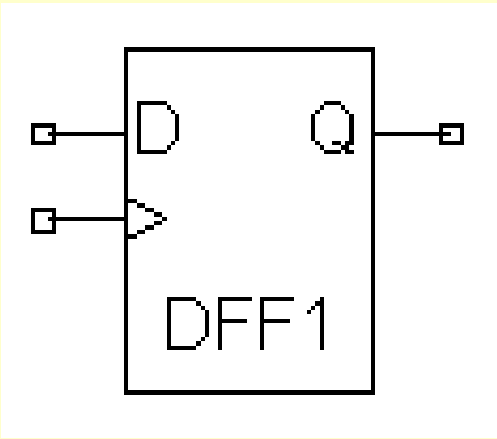


图4-4 D触发器

【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 寄存器描述及其VHDL语言现象

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 寄存器描述及其VHDL语言现象

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 ;
```

使用库和程序包的一般定义表式是：

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

4.2 寄存器描述及其VHDL语言现象

4.2.2 VHDL描述的语言现象说明

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

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 寄存器描述及其VHDL语言现象

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

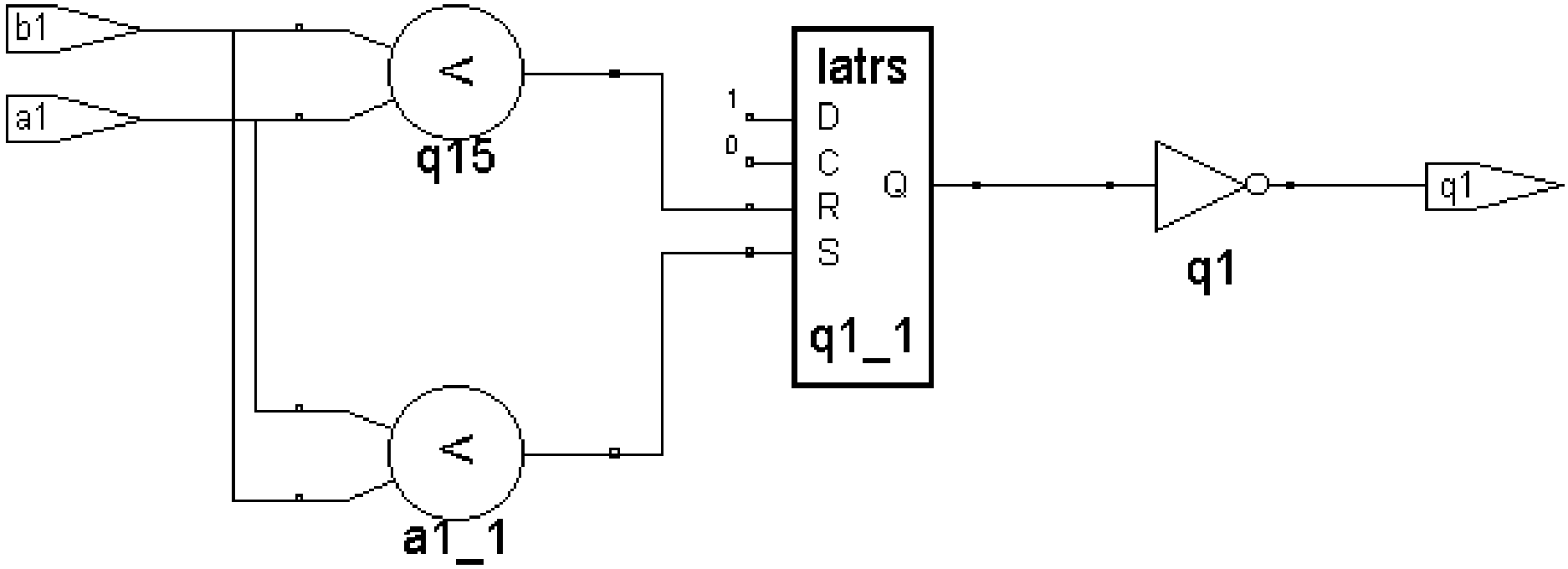


图4-5 例4-7的电路图（Synplify综合）

4.2 寄存器描述及其VHDL语言现象

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;
...
```

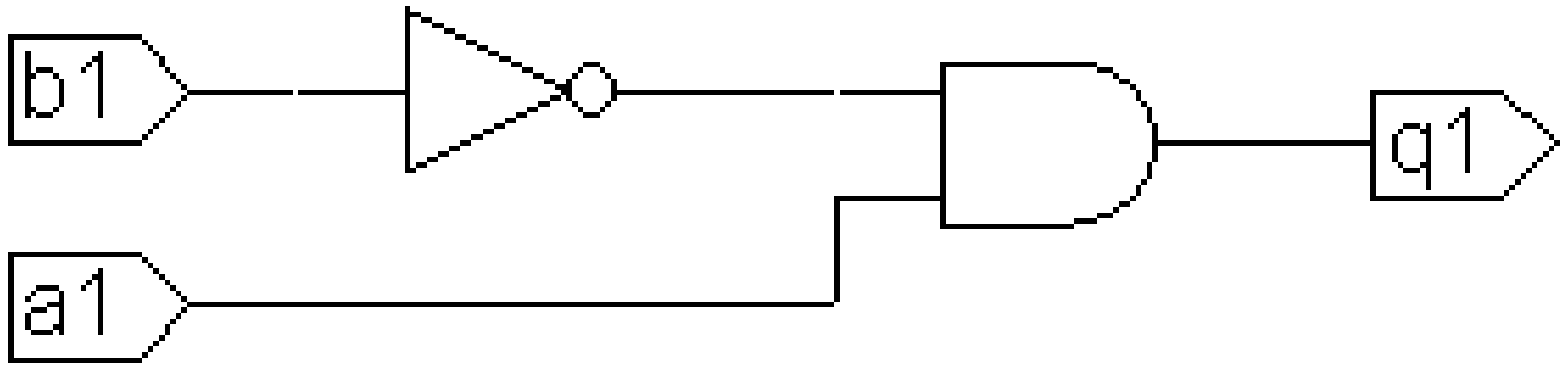


图4-6 例4-8的电路图（Synplify综合）

4.2 寄存器描述及其VHDL语言现象

4.2.3 实现时序电路的VHDL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 寄存器描述及其VHDL语言现象

4.2.3 实现时序电路的VHDL不同表述

【例4-10】

```
...
PROCESS (CLK)
  BEGIN
IF  CLK='1' AND CLK'LAST_VALUE='0'    --同例3-9
        THEN  Q <= D ;
     END IF;
END PROCESS ;
```

【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 寄存器描述及其VHDL语言现象

4.2.3 实现时序电路的VHDL不同表述

【例4-12】

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

4.2.3 实现时序电路的VHDL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 ;
```

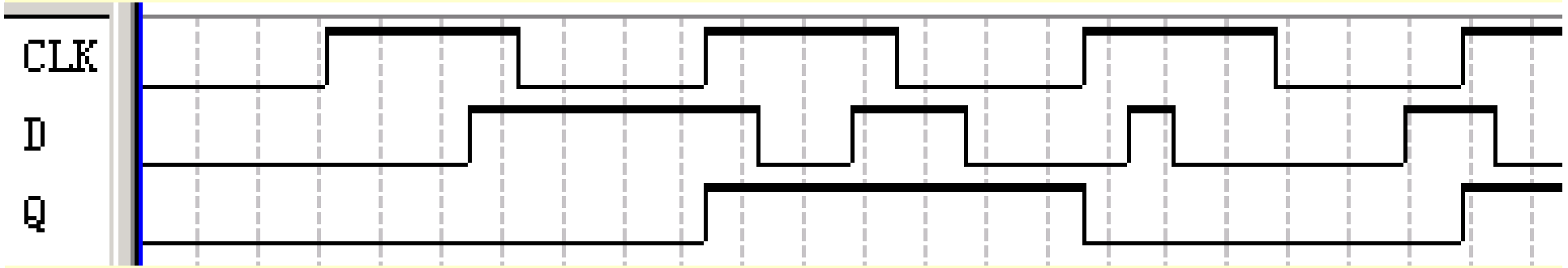


图4-7 例4-13的时序波形

4.2.3 实现时序电路的VHDL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 ;
```

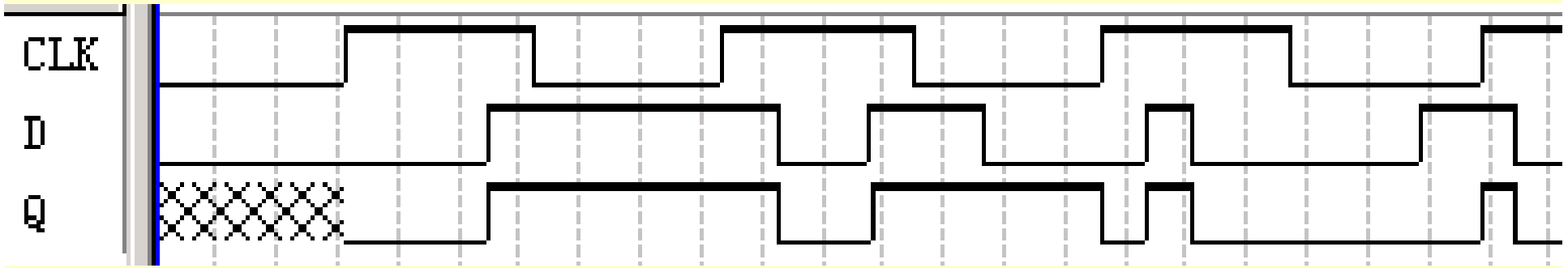


图4-8 例4-14的时序波形

4.2.4 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 寄存器描述及其VHDL语言现象

4.2.4 异步时序电路设计

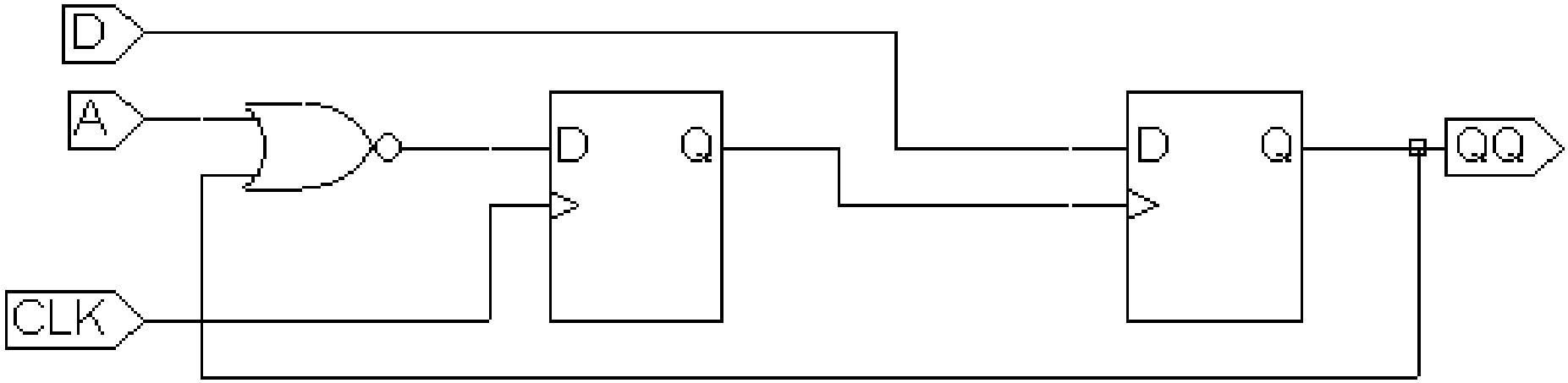
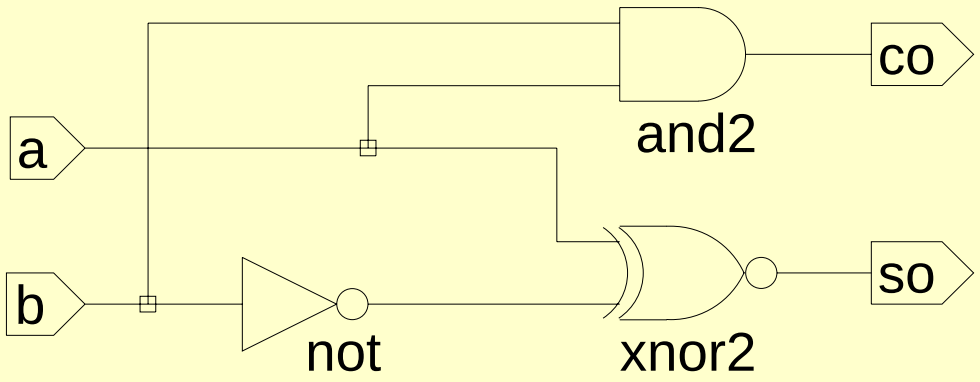


图4-9 例4-15综合后的电路（Synplify综合）

4.3 1位二进制全加器的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位二进制全加器的VHDL描述

4.3.1 半加器描述

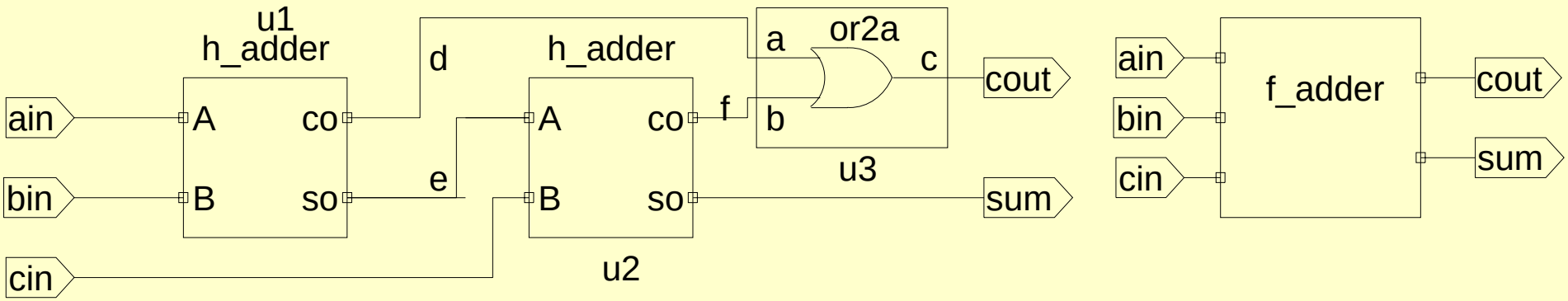


图4-11 全加器f_adder电路图及其实体模块

4.3 1位二进制全加器的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;
```

【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位二进制全加器的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位二进制全加器的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 ;
```

4.3 1位二进制全加器的VHDL描述

4.3.2 CASE语句

2. 标准逻辑矢量数据类型

STD_LOGIC_VECTOR　　STD_LOGIC

在使用STD_LOGIC_VECTOR中，必须注明其数组宽度，即位宽，如：

```
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)
```

4.3 1位二进制全加器的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位二进制全加器的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

表面上，**BUFFER**具有双向端口**INOUT**的功能，但实际上其输入功能是不完整的，它只能将自己输出的信号再反馈回来，并不含有IN的功能。

表式**Q <= Q + 1**的右项与左项并非处于相同的时刻内，对于时序电路，除了传输延时外，前者的结果出现于当前时钟周期；后者，即左项要获得当前的**Q + 1**，需等待下一个时钟周期。

4.4 计数器设计

4.4.2 整数类型

Q : BUFFER INTEGER RANGE 15 DOWNTO 0;

整数常量的书写方式示例

```
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
```

Q : BUFFER NATURAL RANGE 15 DOWNTO 0;

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

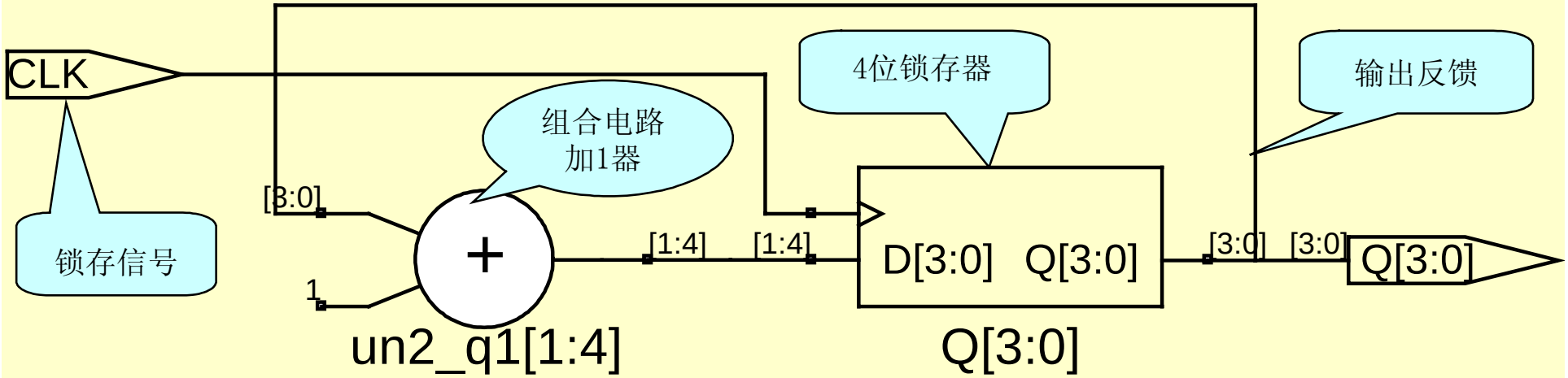


图4-12 4位加法计数器RTL电路（Synplify综合）

4.4 计数器设计

4.4.3 计数器设计的其他表述方法

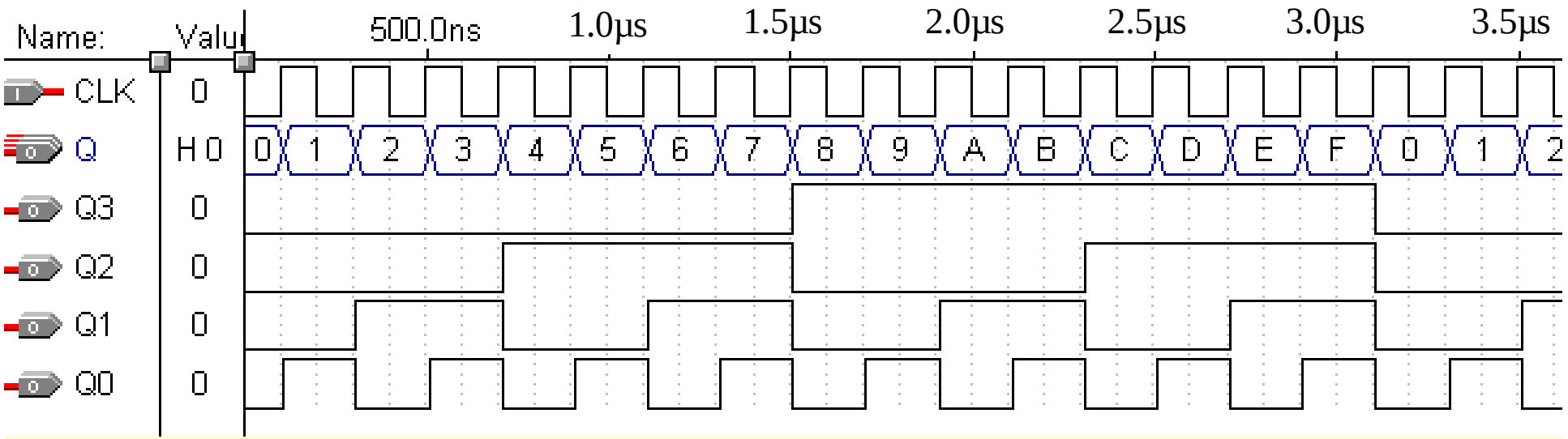


图4-13 4位加法计数器工作时序

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
```

接下页

4.5 一般加法计数器设计

```
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

```
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) ;
```

4.5 一般加法计数器设计

4.5.2 程序分析

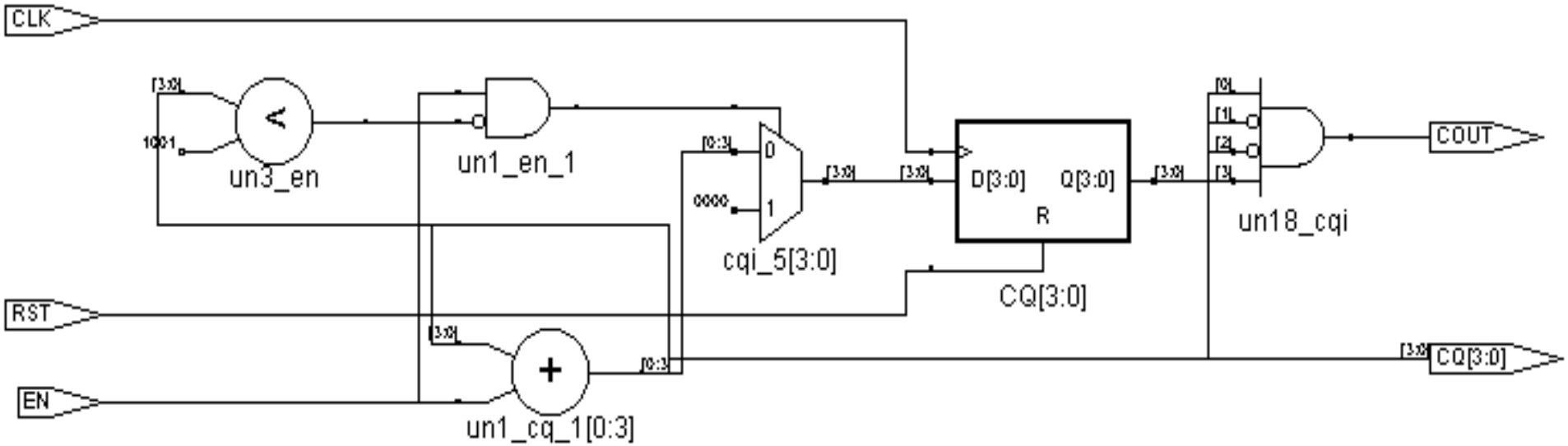


图4-14 例4-22的RTL电路（Synplify综合）

4.5 一般加法计数器设计

4.5.2 程序分析

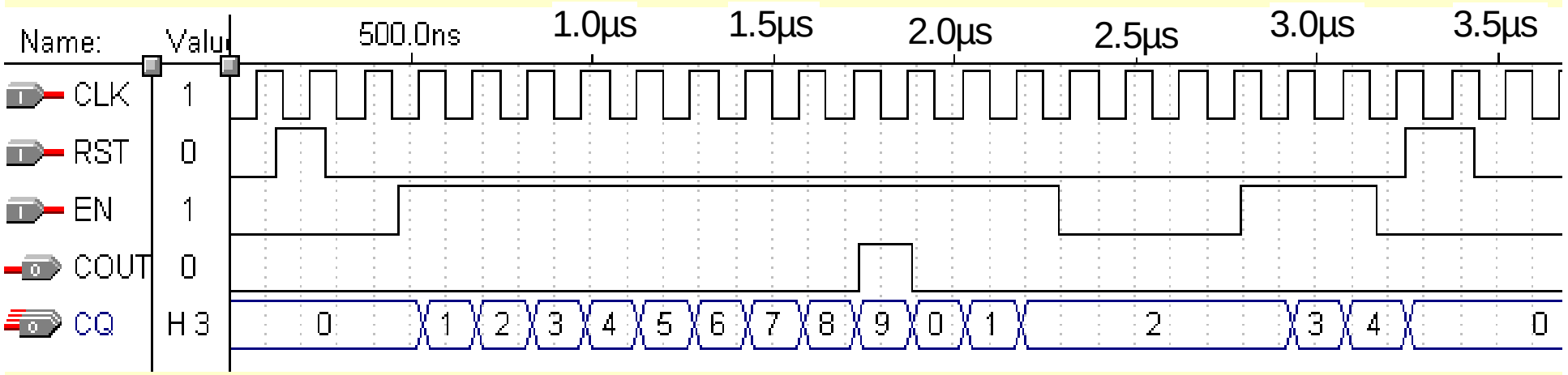


图4-15 例4-22的工作时序

4.5.3 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

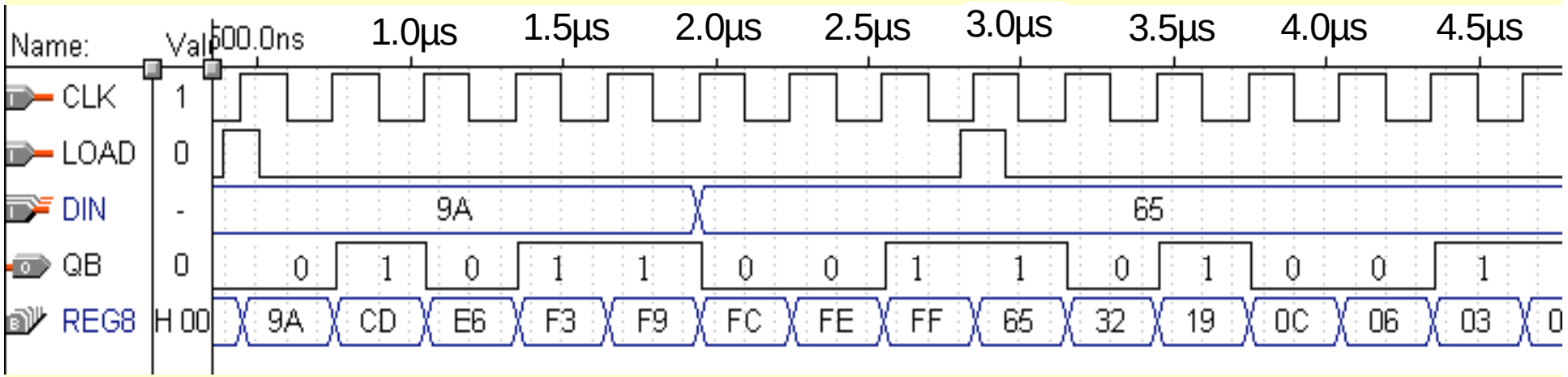


图4-16 例4-23的工作时序

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4-17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。

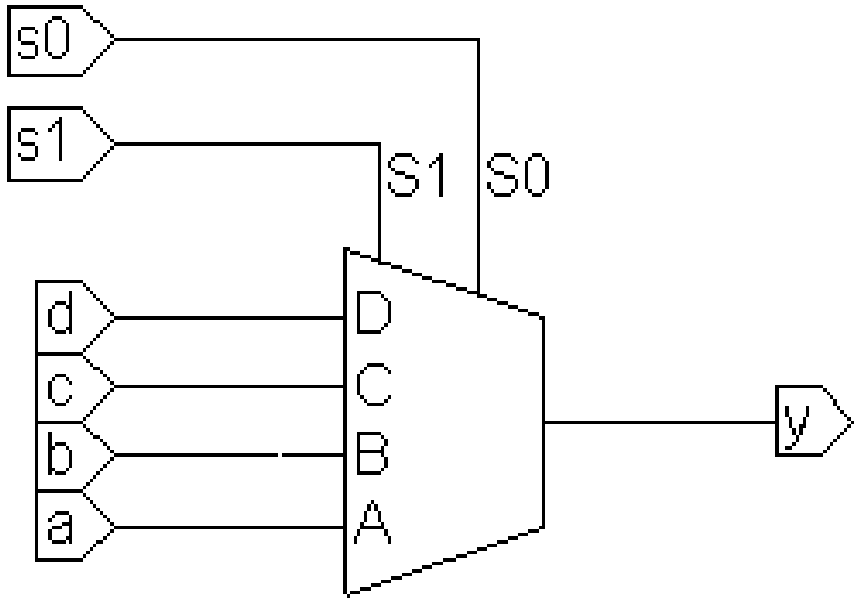


图4-17 4选1多路选择器

习 题

4-3. 图4-18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。

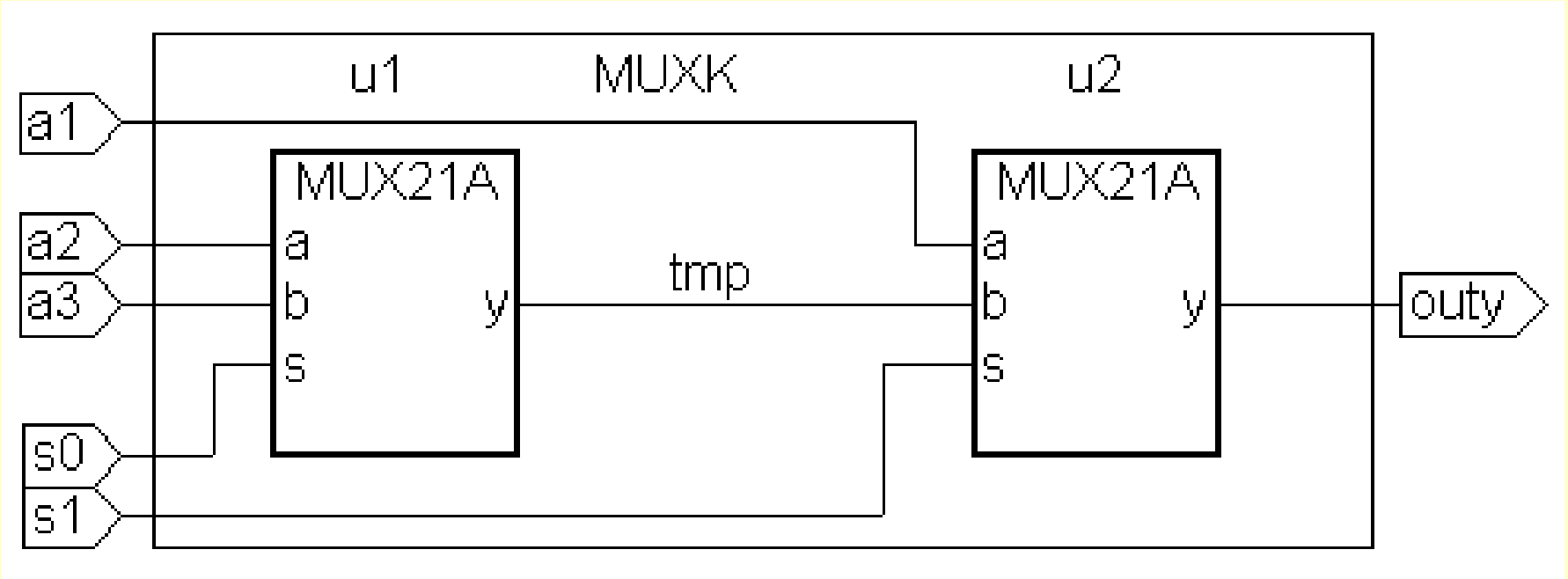


图4-18 双2选1多路选择器

习 题

4-4. 图4-19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。

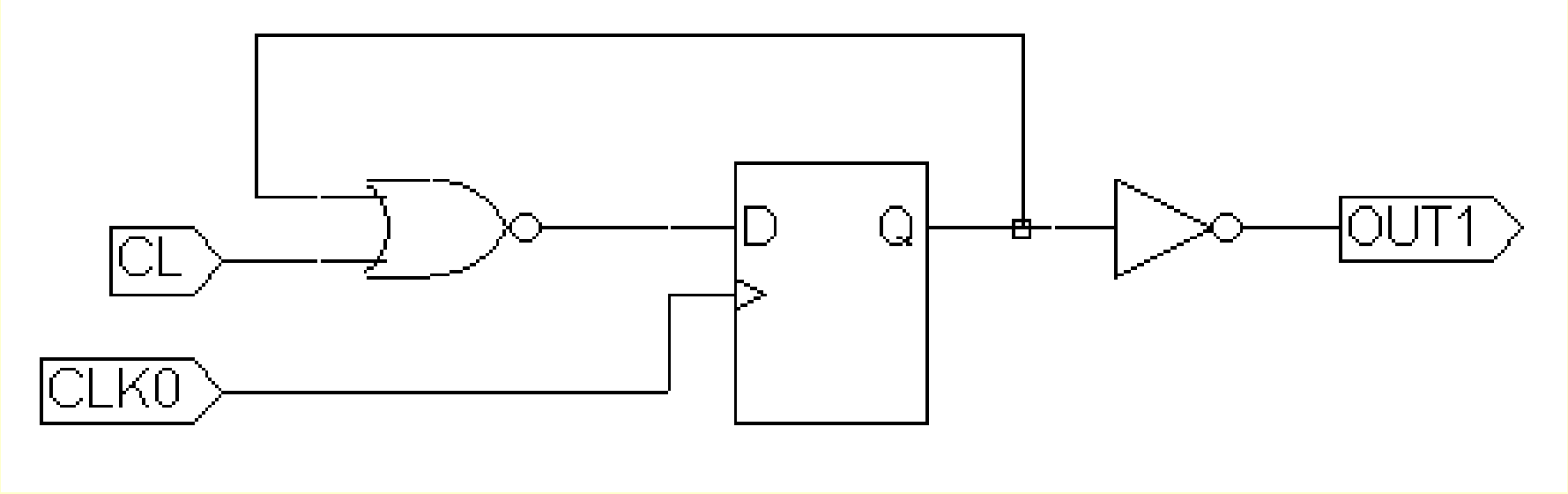


图4-19 时序电路图

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4-20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)。

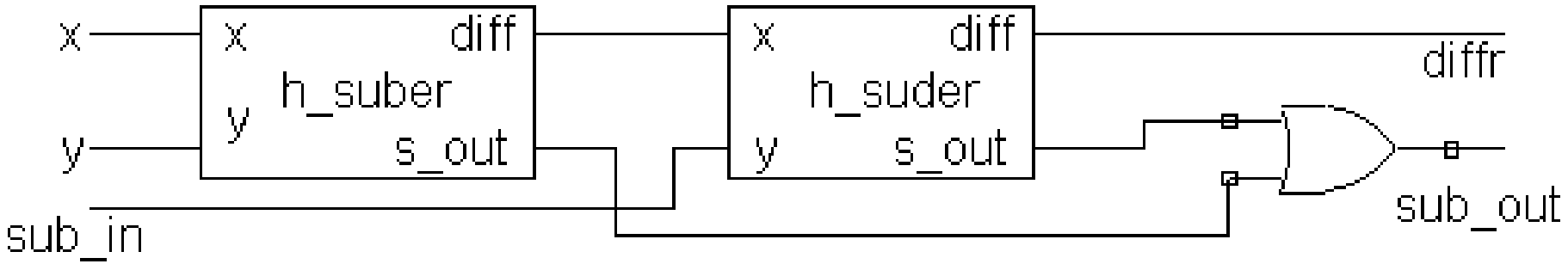


图4-19 时序电路图

习 题

4-6. 根据图4-21，写出顶层文件MX3256.VHD的VHDL设计文件。

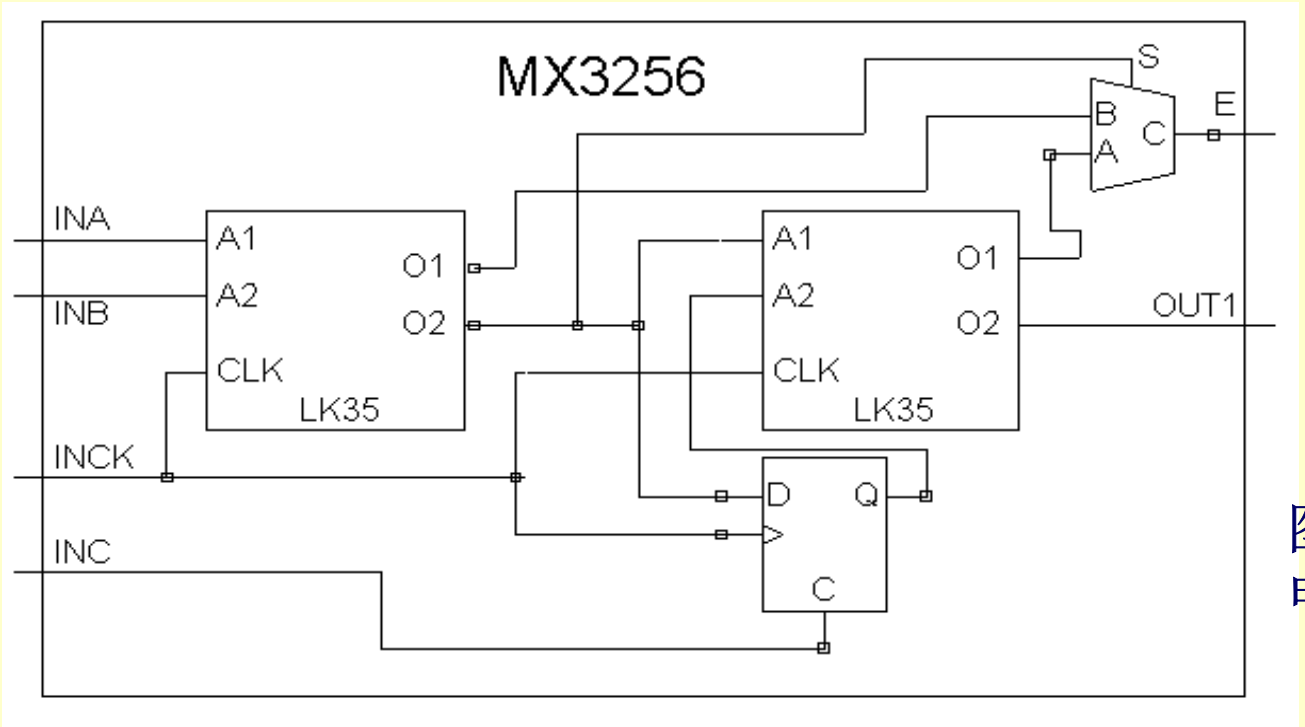


图4-21 题4-6 电路图

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。