EDA技术及其应用

第5章　VHDL设计技术深入

5.1 深入讨论数据对象

5.1.1 常数

CONSTANT 常数名：数据类型 **:=** 表达式；

```
CONSTANT  FBT : STD_LOGIC_VECTOR := "010110" ;
                                          -- 标准位矢类型
CONSTANT  DATAIN : INTEGER := 15 ;          -- 整数类型
```

第1句定义常数FBT的数据类型是STD_LOGIC_VECTOR，它等于“010110”；

第2句定义常数DATAIN的数据类型是整数INTEGER，它等于15 。

5.1 深入讨论数据对象

5.1.2 变量

VARIABLE 变量名：数据类型 := 初始值；

```
VARIABLE  a : INTEGER RANGE 0 TO 15 ;
                                --变量a定义为常数，取值范围是0到5
VARIABLE  d : STD_LOGIC := '1' ;
                                --变量a定义为标准逻辑位类型，初始值是1
```

目标变量名 := 表达式；

```
VARIABLE  x, y : INTEGER RANGE 15 DOWNTO 0 ;
                                --定义变量x和y为整数类型
VARIABLE  a, b : STD_LOGIC_VECTOR(7 DOWNTO 0) ;
x := 11 ;
y := 2 + x ;                    -- 运算表达式赋值，y 也是实数变量
a := b                          --b向a赋值
a(0 TO 5) :=  b(2 TO 7) ;
```

5.1 深入讨论数据对象

5.1.3 信号

SIGNAL 信号名： 数据类型 := 初始值 ；

目标信号名 <= 表达式 AFTER 时间量;

```
SIGNAL a，b，c，y，z: INTEGER ;
  …
PROCESS (a，b，c)
BEGIN
 y <= a + b ;
 z <= c – a ;
 y <= b ;
END PROCESS ;
```

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

表5-1 信号与变量赋值语句功能的比较

	信号SIGNAL	变量VARIABLE
基本用法	用于作为电路中的信号连线	用于作为进程中局部数据存储单元
适用范围	在整个结构体内的任何地方都能适用	只能在所定义的进程中使用
行为特性	在进程的最后才对信号赋值	立即赋值

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

【例5-1】

```
...
 ARCHITECTURE bhv OF DFF3 IS
  BEGIN
  PROCESS (CLK)
   VARIABLE QQ : STD_LOGIC ;
   BEGIN
    IF  CLK'EVENT AND CLK='1'
      THEN QQ := D1 ;
    END IF;
 END PROCESS ;
     Q1 <= QQ;
 END ;
```

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

【例5-2】

```
...
ARCHITECTURE bhv OF DFF3 IS
  SIGNAL QQ : STD_LOGIC ;
 BEGIN
 PROCESS (CLK)
  BEGIN
   IF  CLK'EVENT AND CLK='1'
    THEN  QQ <= D1 ;
   END IF;
END PROCESS ;
    Q1 <= QQ;
END ;
```

【例5-3】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF3 IS
PORT ( CLK,D1 : IN STD_LOGIC ;
       Q1  : OUT STD_LOGIC  ) ;
END ;
 ARCHITECTURE bhv OF DFF3 IS
  SIGNAL A,B : STD_LOGIC ;
 BEGIN
 PROCESS (CLK) BEGIN
   IF  CLK'EVENT AND CLK = '1'
    THEN
     A <= D1 ;
     B <= A ;
    Q1 <= B ;
   END IF;
 END PROCESS ;
END ;
```

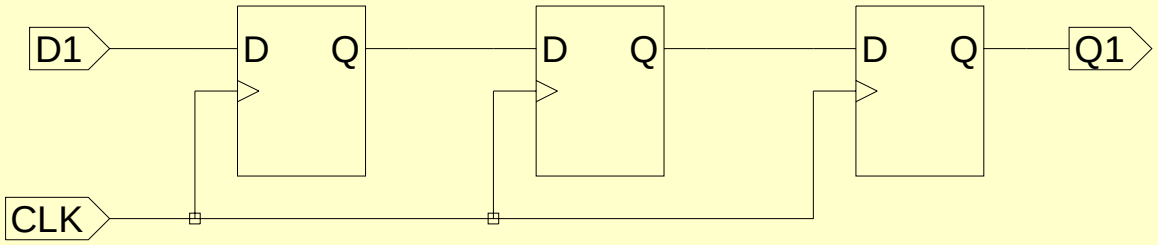

图5-1 例5-3的RTL电路

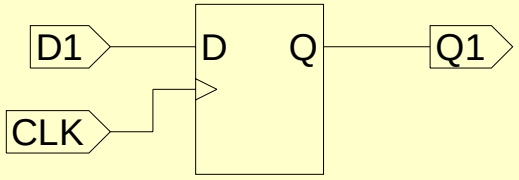

图5-2 D触发器电路

【例5-4】

```
LIBRARY IEEE ;
 USE IEEE.STD_LOGIC_1164.ALL ;
 ENTITY DFF3 IS
 PORT ( CLK,D1 : IN STD_LOGIC ;
          Q1 : OUT STD_LOGIC  ) ;
 END ;
  ARCHITECTURE bhv OF DFF3 IS
   BEGIN
   PROCESS (CLK)
    VARIABLE A,B : STD_LOGIC ;
    BEGIN
     IF  CLK'EVENT AND CLK = '1'
       THEN
       A := D1 ;
       B := A ;
      Q1 <= B ;
     END IF;
  END PROCESS ;
    END ;
```

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

【例5-5】

```
SIGNAL in1, in2, e1, ... : STD_LOGIC;
...
PROCESS(in1, in2, ...)
VARIABLE c1, ... : STD_LOGIC_VECTOR(3 DOWNTO 0);
 BEGIN
   IF in1 = '1' THEN ...          -- 第 1 行
      e1 <= "1010";               -- 第 2 行
         ...
   IF in2 = '0' THEN ...          -- 第 15+n 行
         ...
      c1 := "0011";               -- 第 30+m 行
         ...
   END IF;
 END PROCESS;
```

【例5-6】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
PORT (i0, i1, i2, i3, a, b : IN STD_LOGIC;
     q : OUT STD_LOGIC);
END mux4;
ARCHITECTURE body_mux4 OF mux4 IS
signal muxval : integer range 7 downto 0;
BEGIN
process(i0,i1,i2,i3,a,b)
begin
                  muxval <= 0;
if (a = '1') then   muxval <= muxval + 1; end if;
if (b = '1') then   muxval <= muxval + 2; end if;
case muxval is
    when 0 => q <= i0;
    when 1 => q <= i1;
    when 2 => q <= i2;
    when 3 => q <= i3;
    when others => null;
end case;
end process;
END body_mux4;
```

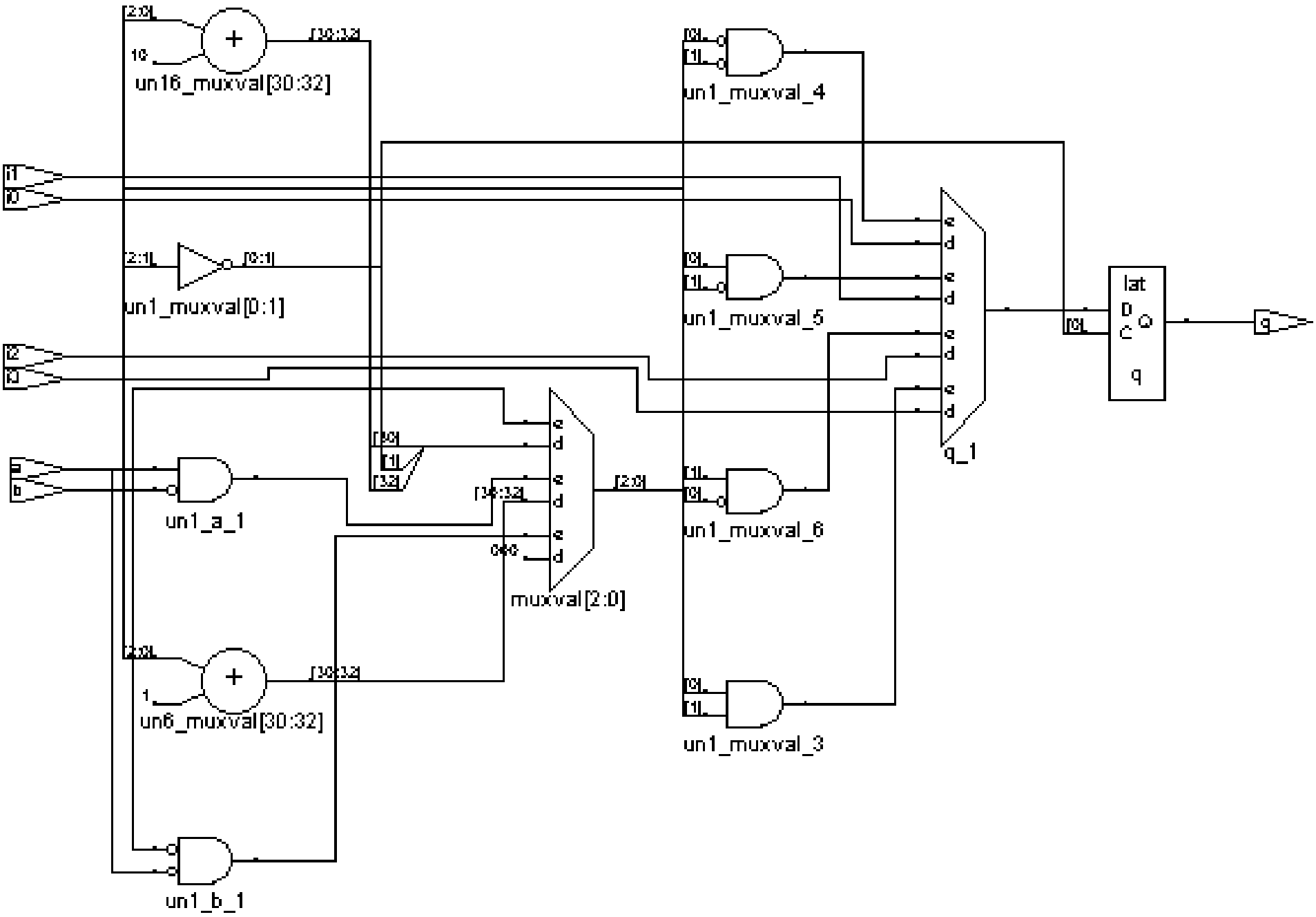

图5-3 例5-6的RTL电路（Synplify综合）

【例5-7】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
PORT (i0, i1, i2, i3, a, b : IN STD_LOGIC;
     q : OUT STD_LOGIC);
END mux4;
ARCHITECTURE body_mux4 OF mux4 IS
BEGIN
process(i0,i1,i2,i3,a,b)
variable muxval : integer range 7 downto 0;
begin
                  muxval := 0;
if (a = '1') then   muxval := muxval + 1;  end if;
if (b = '1') then   muxval := muxval + 2;  end if;
case muxval is
    when 0 => q <= i0;
    when 1 => q <= i1;
    when 2 => q <= i2;
    when 3 => q <= i3;
    when others => null;
end case;
end process;
END body_mux4;
```

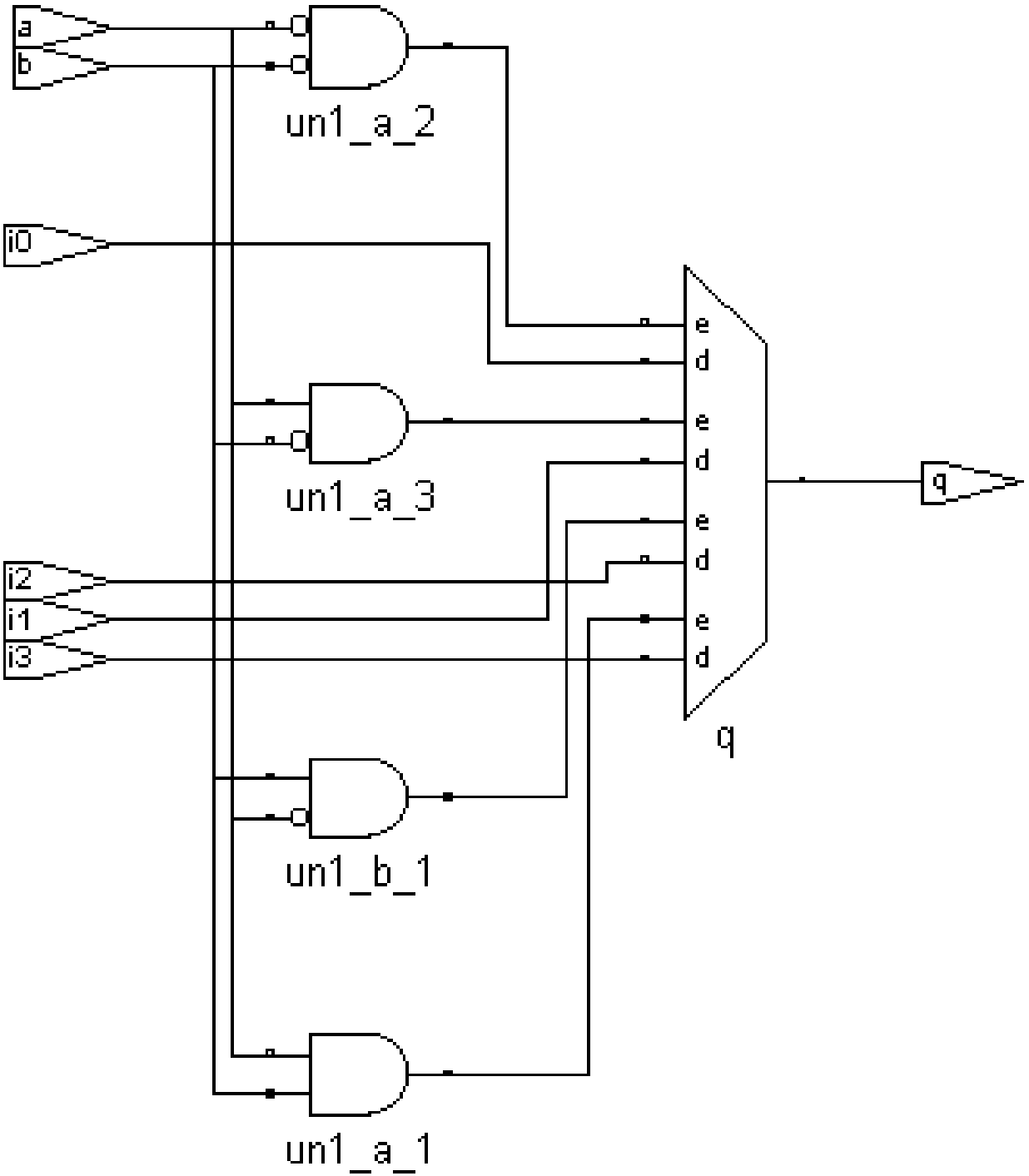

图5-4 例5-7的RTL电路（Synplify综合）

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

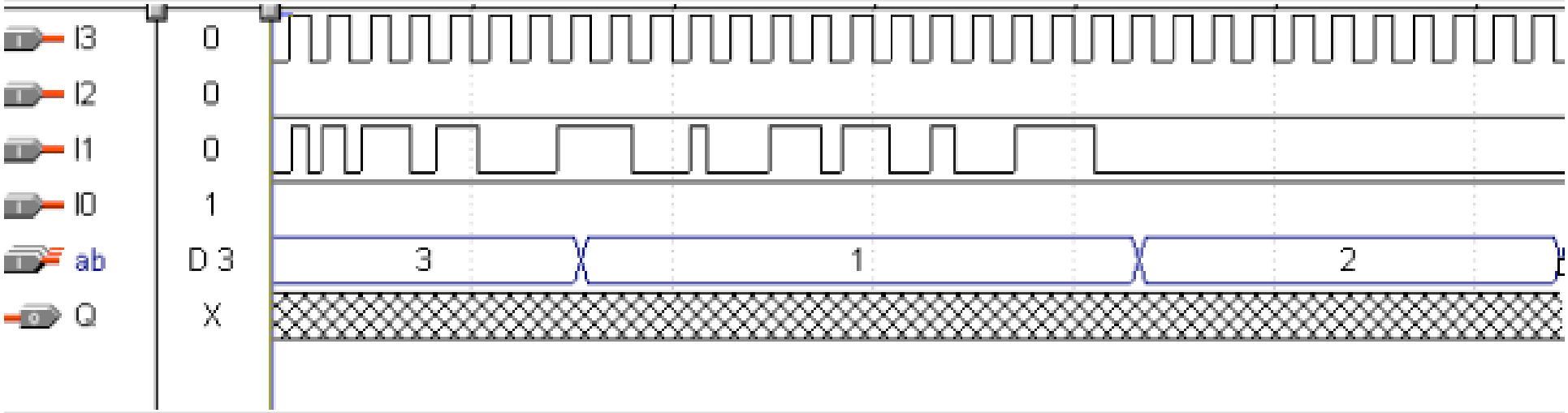

图5-5 例5-6中错误的工作时序

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

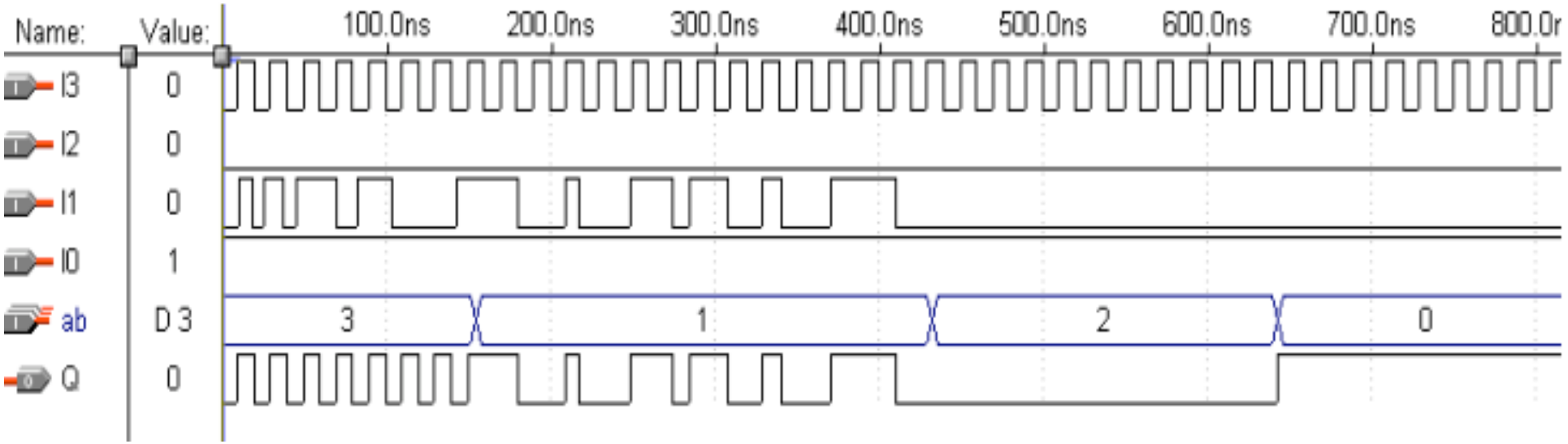

图5-6 例5-7中正确的工作时序

【例5-8】

```
Library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SHIFT IS
          PORT (CLK,C0 :  IN STD_LOGIC;  --时钟和进位输入
                              MD   :  IN STD_LOGIC_VECTOR(2 DOWNTO 0); --移位模式控制字
                              D    :  IN STD_LOGIC_VECTOR(7 DOWNTO 0); --待加载移位的数据
                              QB   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); --移位数据输出
                              CN          : OUT STD_LOGIC);                --进位输出
END ENTITY;
ARCHITECTURE BEHAV OF SHIFT IS
            SIGNAL REG  : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL   CY : STD_LOGIC ;
 BEGIN
PROCESS (CLK,MD,C0)
  BEGIN
   IF CLK'EVENT AND CLK = '1' THEN
    CASE MD IS
     WHEN "001" =>  REG(0) <= C0 ;
REG(7 DOWNTO 1) <= REG(6 DOWNTO 0); CY <= REG(7);--带进位循环左移
WHEN "010" =>      REG(0) <= REG(7);
REG(7 DOWNTO 1) <= REG(6 DOWNTO 0);            --自循环左移
WHEN "011" =>      REG(7) <= REG(0);
REG(6 DOWNTO 0) <= REG(7 DOWNTO 1);            --自循环右移
WHEN "100" =>      REG(7) <= C0 ;
REG(6 DOWNTO 0) <= REG(7 DOWNTO 1); CY <= REG(0);--带进位循环右移
WHEN "101" =>  REG(7 DOWNTO 0) <=   D(7 DOWNTO 0); --加载待移数
WHEN OTHERS => REG <= REG ;  CY <= CY ;            --保持
END CASE;
  END IF;
 END PROCESS;
QB(7 DOWNTO 0) <= REG(7 DOWNTO 0); CN <= CY;       --移位后输出
END BEHAV;
```

5.1 深入讨论数据对象

5.1.4 进程中的信号与变量赋值

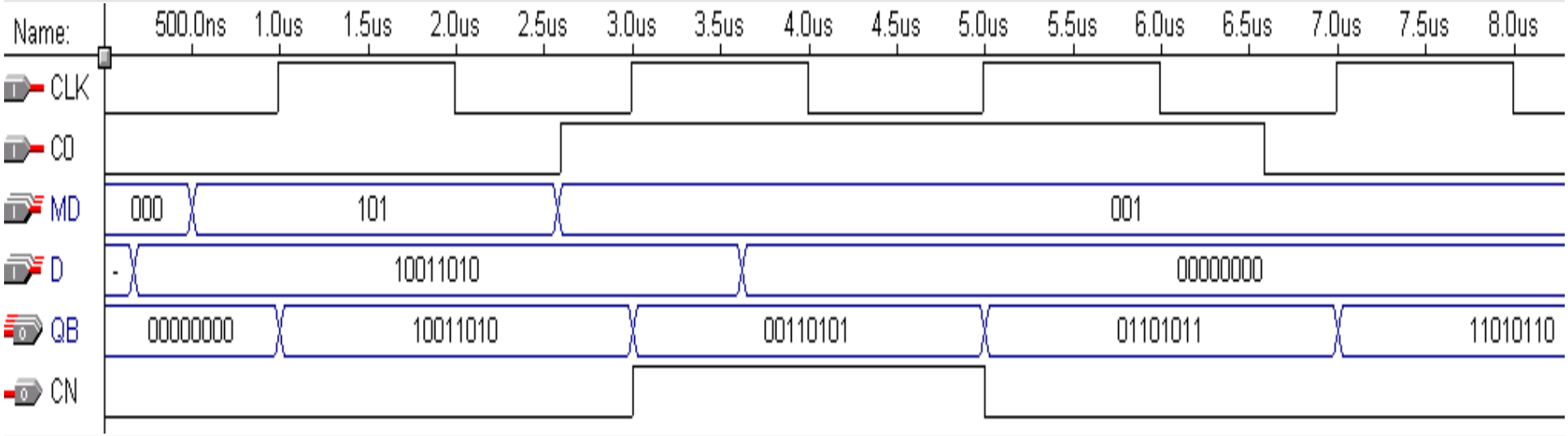

图5-7 例5-8中带进位循环左移仿真波形（MD="001"）

5.2 双向和三态电路信号赋值

5.2.1 三态门设计

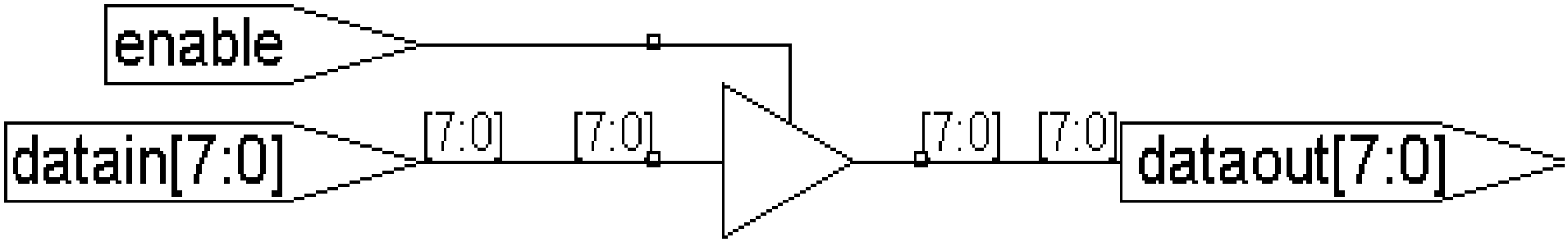

图5-8 8位3态控制门电路（Synplify综合）

5.2 双向和三态电路信号赋值

【例5-9】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY tri_s IS
   port (    enable : IN STD_LOGIC;
             datain : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            dataout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
END tri_s ;
ARCHITECTURE bhv OF tri_s IS
BEGIN
PROCESS(enable,datain)
  BEGIN
  IF enable = '1' THEN dataout <= datain ;
    ELSE dataout <="ZZZZZZZZ" ;
  END IF ;
END PROCESS;
END bhv;
```

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

【例5-10】

```
library ieee;
use ieee.std_logic_1164.all;
entity tri_state is
port (control : in std_logic;
     in1: in std_logic_vector(7 downto 0);
      q : inout std_logic_vector(7 downto 0);
      x : out std_logic_vector(7 downto 0));
end tri_state;
architecture body_tri of tri_state is
begin
process(control,q,in1)
begin
if (control = '0') then   x <= q  ;
else      q <= in1; x<="ZZZZZZZZ" ;
end if;
end process;
end body_tri;
```

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

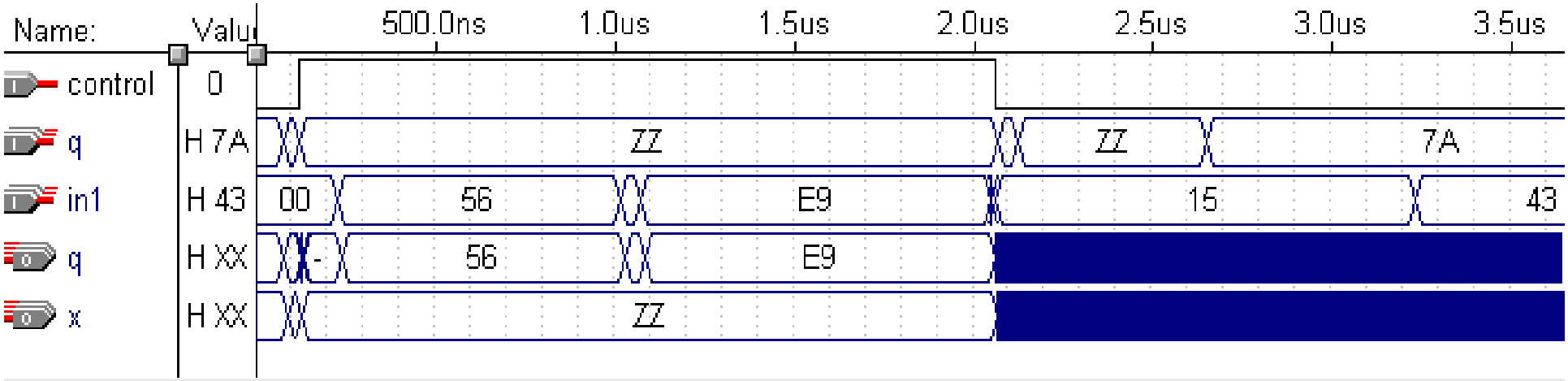

图5-9 例5-10的仿真波形图

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

【例5-11】

```
(以上部分同上例)
process(control,q,in1)
begin
if (control='0') then  x <= q ; q <= "ZZZZZZZZ";
                else  q <= in1; x <="ZZZZZZZZ";
    end if;
end process;
end body_tri;
```

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

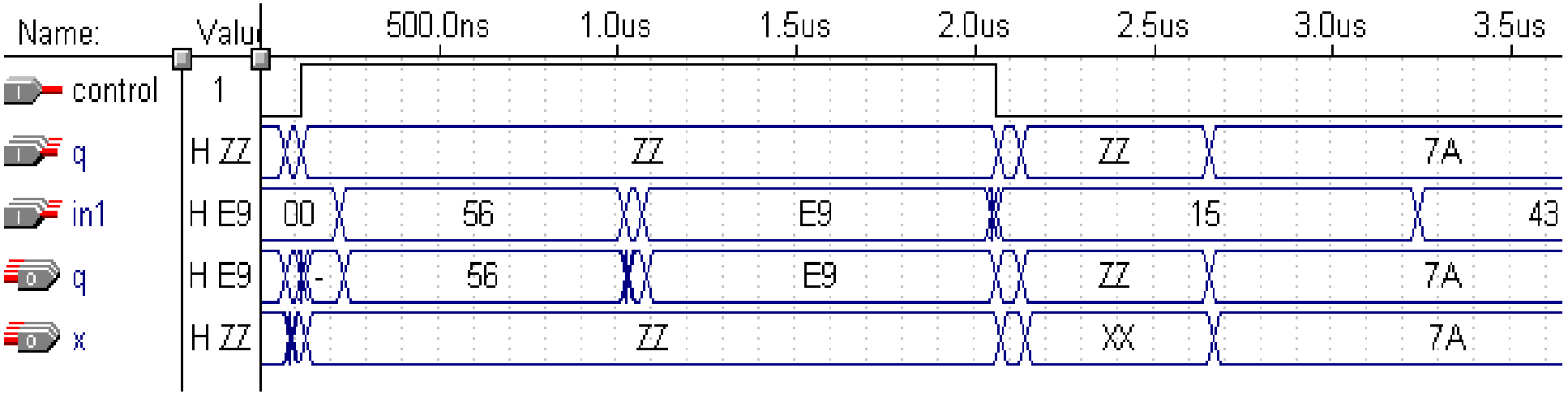

图5-10 例5-11的仿真波形图

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

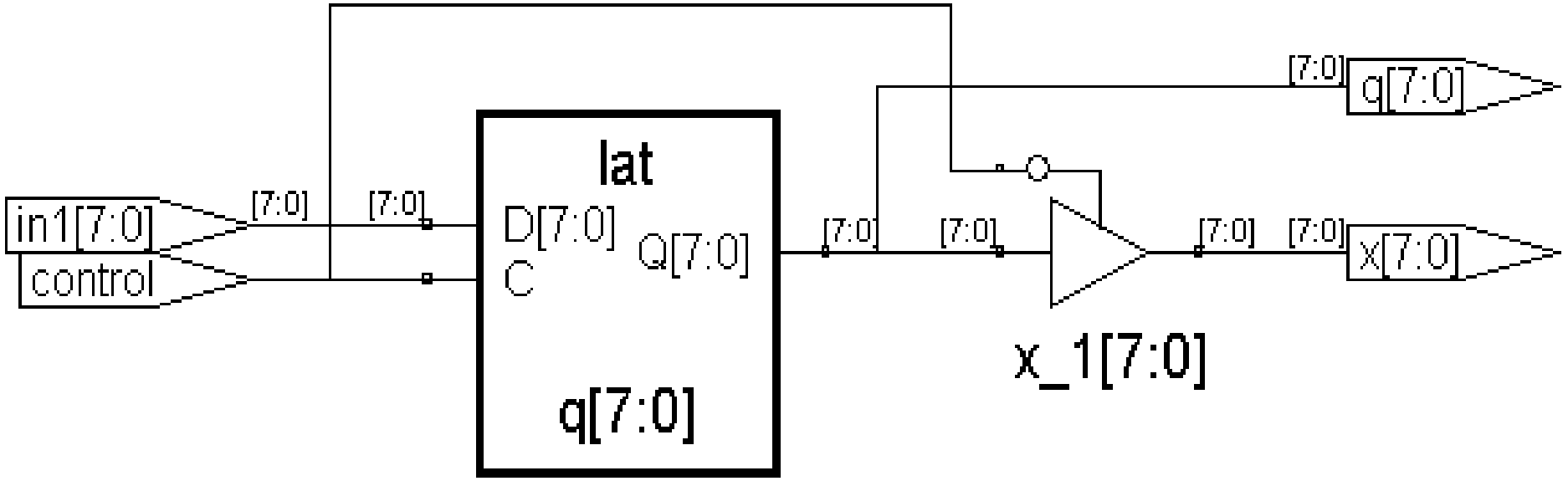

图5-11 例5-10的综合结果（Synplify综合）

5.2 双向和三态电路信号赋值

5.2.2 双向端口设计

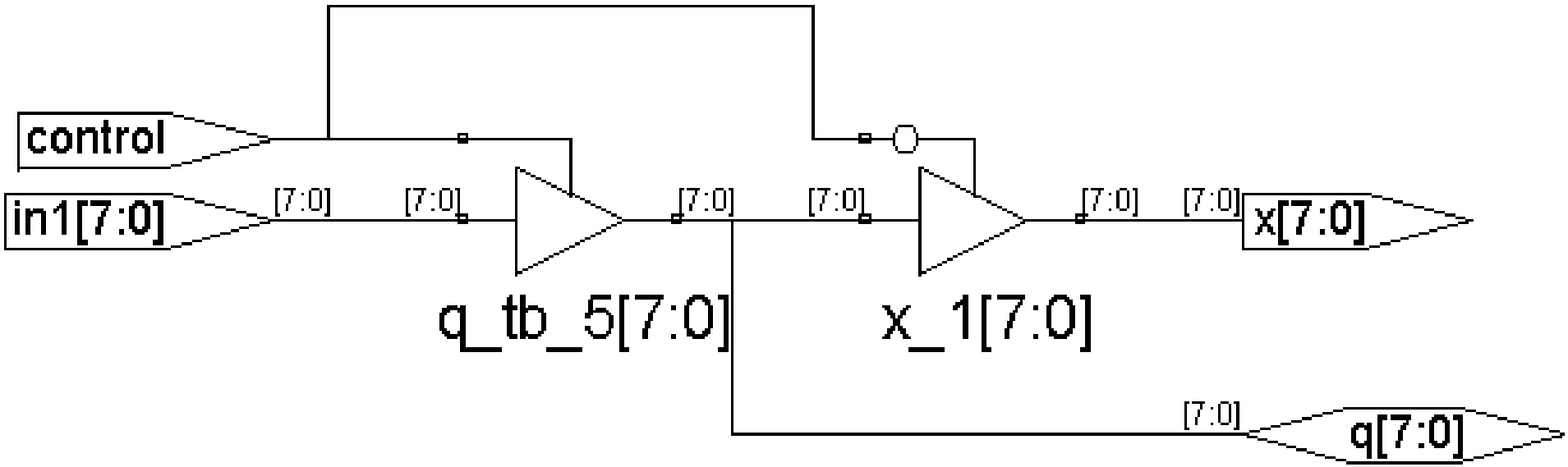

图5-12 例5-11的综合结果（Synplify综合）

5.2.3 三态总线电路设计

【例5-12】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY tristate2 IS
   port ( input3, input2, input1, input0 :
          IN STD_LOGIC_VECTOR (7 DOWNTO 0);
   enable : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
   output : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END tristate2 ;
ARCHITECTURE multiple_drivers OF tristate2 IS
BEGIN
PROCESS(enable,input3, input2, input1, input0 )
  BEGIN
  IF enable = "00" THEN output <= input3 ;
     ELSE output <=(OTHERS => 'Z');
  END IF ;
  IF enable = "01" THEN output <= input2 ;
     ELSE output <=(OTHERS => 'Z');
  END IF ;
  IF enable = "10" THEN output <= input1 ;
     ELSE output <=(OTHERS => 'Z');
  END IF ;
  IF enable = "11" THEN output <= input0 ;
     ELSE output <=(OTHERS => 'Z');
  END IF ;
END PROCESS;
END multiple_drivers;
```

5.2 双向和三态电路信号赋值

5.2.3 三态总线电路设计

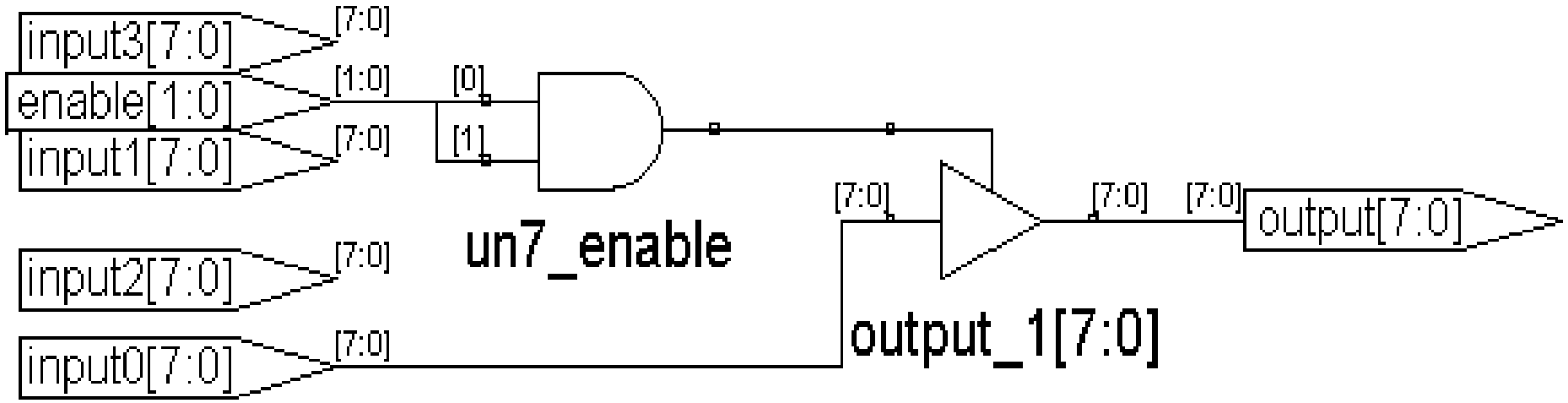

图5-13 例5-12错误的综合结果（Synplify综合结果）

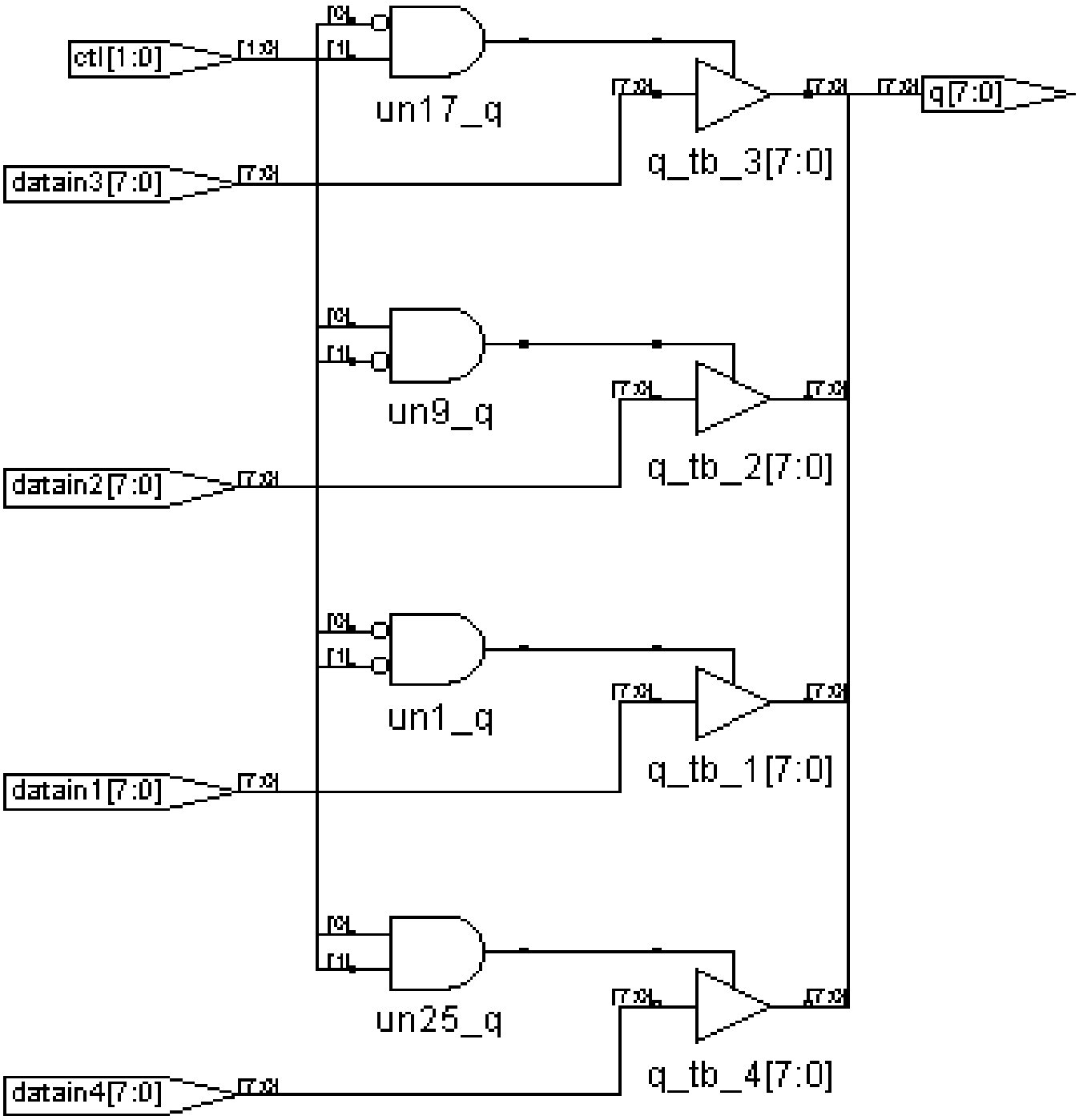

图5-14 例5-13正确的综合结果（Synplify综合结果）

5.3 IF语句概述

(1) IF 条件句 Then
顺序语句
END IF ;

(2) IF 条件句 Then
顺序语句
ELSE
顺序语句
END IF ;

(3) IF 条件句 Then
IF 条件句 Then
...
END IF
END IF

(4) IF 条件句 Then
顺序语句
ELSIF 条件句 Then
顺序语句
...
ELSE
顺序语句
END IF

5.3 IF语句概述

【例5-14】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY control_stmts IS
PORT (a, b, c: IN BOOLEAN;
      output: OUT BOOLEAN);
END control_stmts;
ARCHITECTURE example OF control_stmts IS
BEGIN
  PROCESS (a, b, c)
    VARIABLE n: BOOLEAN;
  BEGIN
    IF  a  THEN   n := b;   ELSE   n := c;
    END IF;
    output <= n;
  END PROCESS;
END example;
```

5.3 IF语句概述

表5-2 8线-3线优先编码器真值表

输入								输出		
din0	din1	din2	din3	din4	din5	din6	din7	output0	output1	output2
x	x	x	x	x	x	x	0	0	0	0
x	x	x	x	x	x	0	1	1	0	0
x	x	x	x	x	0	1	1	0	1	0
x	x	x	x	0	1	1	1	1	1	0
x	x	x	0	1	1	1	1	0	0	1
x	x	0	1	1	1	1	1	1	0	1
x	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	1

注：表中的“x”为任意，类似VHDL中的“—”值。

【例5-15】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY coder IS
 PORT (    din : IN STD_LOGIC_VECTOR(0 TO 7);
        output : OUT STD_LOGIC_VECTOR(0 TO 2)  );
END coder;
ARCHITECTURE behav OF coder IS
 SIGNAL SINT : STD_LOGIC_VECTOR(4 DOWNTO 0);
 BEGIN
  PROCESS (din)
 BEGIN
    IF (din(7)='0') THEN  output <= "000" ;
  ELSIF (din(6)='0') THEN  output <= "100" ;
  ELSIF (din(5)='0') THEN  output <= "010" ;
  ELSIF (din(4)='0') THEN  output <= "110" ;
  ELSIF (din(3)='0') THEN  output <= "001" ;
  ELSIF (din(2)='0') THEN  output <= "101" ;
  ELSIF (din(1)='0') THEN  output <= "011" ;
                   ELSE  output <= "111" ;
 END IF ;
  END PROCESS ;
 END behav;
```

5.4 深入了解进程语句

5.4.1 进程语句格式

PROCESS语句结构的一般表达格式如下

[进程标号:] PROCESS [(敏感信号参数表)] [IS]
[进程说明部分]
BEGIN
　　顺序描述语句
END PROCESS [进程标号];

5.4 深入了解进程语句

5.4.2 进程结构组成

进程说明部分

顺序描述语句部分

- 信号赋值语句
- 变量赋值语句
- 进程启动语句
- 子程序调用语句
- 顺序描述语句
- 进程跳出语句

敏感信号参数表

5.4 深入了解进程语句

5.4.3 进程要点

◎基于CPU的纯软件的VHDL行为仿真运行方式；
◎基于VHDL综合器的综合结果所可能实现的运行方式；
◎基于最终实现的硬件电路的运行方式。

1. PROCESS为一无限循环语句

5.4 深入了解进程语句

2. PROCESS中的顺序语句具有明显的顺序/并行运行双重性

```
PROCESS(abc)
 BEGIN
  CASE abc IS
   WHEN "0000" => so<="010";
   WHEN "0001" => so<="111";
   WHEN "0010" => so<="101";
   ...
   WHEN "1110" => so<="100";
   WHEN "1111" => so<="000";
   WHEN OTHERS => NULL;
  END CASE;
END PROCESS;
```

3. 进程必须由敏感信号的变化来启动

5.4 深入了解进程语句

4. 进程语句本身是并行语句

【例5-16】

```
ENTITY mul IS
PORT (a, b, c, selx, sely  : IN BIT;
                data_out : OUT BIT  );
END mul;
ARCHITECTURE ex OF mul IS
 SIGNAL  temp : BIT;
BEGIN
p_a: PROCESS (a, b, selx)
     BEGIN
       IF (selx = '0') THEN  temp <= a;  ELSE    temp <= b;
     END IF;
 END PROCESS p_a ;
  p_b :  PROCESS(temp, c, sely)
     BEGIN
    IF (sely = '0')  THEN  data_out  <= temp; ELSE   data_out <= c;
    END IF;
 END PROCESS p_b ;
END ex;
```

5.4 深入了解进程语句

5.4.3 进程要点

5. 信号是多个进程间的通信线

6. 一个进程中只允许描述对应于一个时钟信号的同步时序逻辑

5.5 并行语句特点

```
data1 <= a AND b ;
data2 <= c ;
```

【例5-17】

```
ARCHITECTURE dataflow OF mux IS
   SIGNAL seiect ： INTEGER RANGE 15 DOWNTO 0;
BEGIN
Select <= 0 WHEN s0='0' AND s1='0' ELSE
          1 WHEN s0='1' AND s1='0' ELSE
          2 WHEN s0='0' AND s1='1' ELSE
          3 ;
     x <= a WHEN select=0  ELSE
          b WHEN select=1  ELSE
          c WHEN select=2  ELSE
          d ;
```

5.6 仿真延时

5.6.1 固有延时

```
z <= x XOR y AFTER 5ns ;

z <= x XOR y ;

B <= A AFTER 20ns ; --固有延时模型
```

5.6 仿真延时

5.6.2 传输延时

B <= TRANSPORT A AFTER 20 ns;-- 传输延时模型

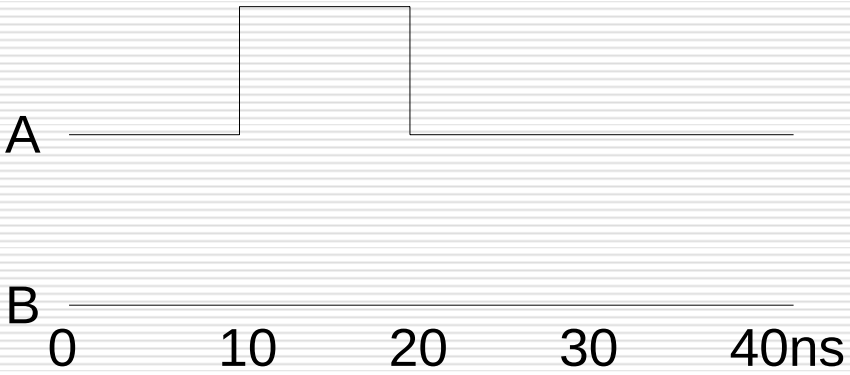

图**5-15** 固有延时输入输出波形

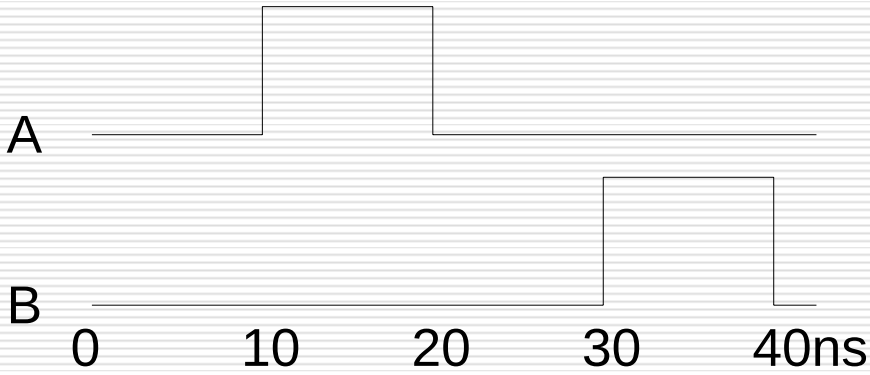

图**5-16**传输延时输入输出波形

5.6 仿真延时

5.6.3 仿真 δ

VHDL仿真器和综合器将自动为系统中的信号赋值配置一足够小而又能满足逻辑排序的延时量，即仿真软件的最小分辨时间，这个延时量就称为仿真δ（Simulation Delta），或称 δ 延时，从而使并行语句和顺序语句中的并列赋值逻辑得以正确执行。由此可见，在行为仿真、功能仿真乃至综合中，引入 δ 延时是必需的。仿真中，δ 延时的引入由EDA工具自动完成，无需设计者介入。

5.7 实体与相关语句语法

5.7.1 实体语句结构

```
ENTITY 实体名 IS
    [GENERIC ( 参数名：数据类型 );]
    [PORT ( 端口表 );]
END ENTITY 实体名;
```

5.7 实体与相关语句语法

5.7.2 参数传递说明语句

参数传递说明语句(GENERIC语句)是一种常数参数的端口界面，常以一种说明的形式放在实体或块结构体前的说明部分。

GENERIC([常数名：数据类型 [：设定值]
{；常数名：数据类型 [：设定值] })；

【例5-18】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY andn IS
 GENERIC (  n : INTEGER );                    --定义类属参量及其数据类型
  PORT(a : IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);--用类属参量限制矢量长度
        c : OUT STD_LOGIC);
END;
ARCHITECTURE behav OF andn IS
   BEGIN
    PROCESS (a)
       VARIABLE int : STD_LOGIC;
    BEGIN
      int := '1';
        FOR i IN a'LENGTH - 1 DOWNTO 0 LOOP  --循环语句
          IF a(i)='0' THEN   int := '0';
          END IF;
        END LOOP;
          c <=int ;
    END PROCESS;
END;
```

【例5-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY exn IS
  PORT(d1,d2,d3,d4,d5,d6,d7 : IN STD_LOGIC;
                    q1,q2  : OUT STD_LOGIC);
END;
 ARCHITECTURE exn_behav OF exn IS
    COMPONENT andn              --调用例10-1的元件调用声明
        GENERIC (  n : INTEGER);
       PORT(a : IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
            C : OUT STD_LOGIC);
     END COMPONENT ;
 BEGIN
 u1: andn GENERIC MAP (n =>2) --参数传递映射语句，定义类属变量，n赋值为2
             PORT MAP (a(0)=>d1,a(1)=>d2,c=>q1);
  u2: andn GENERIC MAP (n =>5)    -- 定义类属变量，n赋值为5
             PORT MAP (a(0)=>d3,a(1)=>d4,a(2)=>d5,
                       a(3)=>d6,a(4)=>d7, c=>q2);
   END;
```

5.7 实体与相关语句语法

5.7.3 参数传递映射语句

GENERIC MAP(类属表)

【例5-20】

```
LIBRARY IEEE;                          --待例化元件
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_arith.ALL;
USE IEEE.STD_LOGIC_unsigned.ALL;
ENTITY addern IS
   PORT (a, b: IN STD_LOGIC_VECTOR;
      result: out STD_LOGIC_VECTOR);
END addern;
 ARCHITECTURE behave OF addern IS
  BEGIN
     result <= a + b;
 END;
```

【例5-21】

```
LIBRARY IEEE;                    --顶层设计
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_ LOGIC_ arith. ALL;
USE IEEE.STD_ LOGIC_ unsigned. ALL;
ENTITY adders IS
   GENERIC (msb_ operand: INTEGER := 15; msb_ sum:
INTEGER :=15);
   PORT( b: IN STD_LOGIC_VECTOR (msb_ operand DOWNTO 0);
        result: OUT STD_LOGIC_VECTOR (msb_ sum DOWNTO 0));
END adders;
 ARCHITECTURE behave OF adders IS
   COMPONENT addern
    PORT (   a, b: IN STD_LOGIC_VECTOR;
          result: OUT STD_LOGIC_VECTOR);
   END COMPONENT;
   SIGNAL a: STD_LOGIC_VECTOR (msb_ sum /2 DOWNTO 0);
   SIGNAL twoa: STD_LOGIC_VECTOR (msb_ operand DOWNTO 0);
  BEGIN
   twoa <= a & a;
  U1: addern PORT MAP (a => twoa , b => b, result => result);
  U2: addern  PORT MAP (a=>b (msb_ operand downto
msb_operand/2 +1),
        b=>b(msb_operand/2 downto 0), result => a);
END behave;
```

5.7 实体与相关语句语法

5.7.3 参数传递映射语句

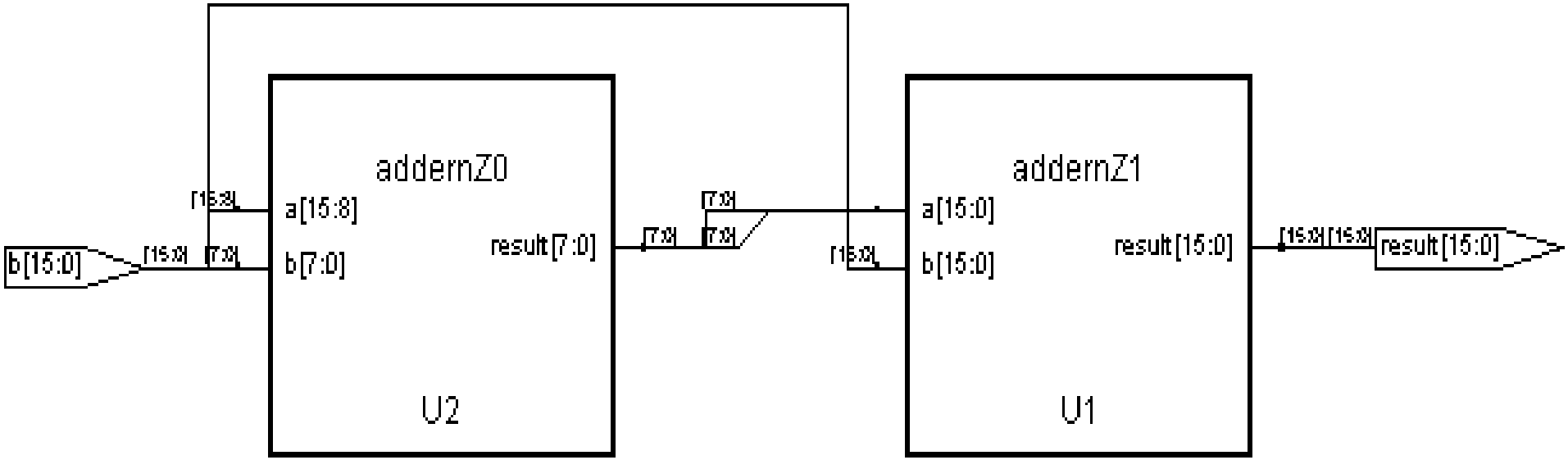

图5-17 例5-21的RTL电路图（Synplify综合）

5.8 直接数字综合器(DDS)设计

$$S_{out} = A\sin\omega t = A\sin(2\pi f_{out} t) \quad (5\text{-}1)$$

$$\theta = 2\pi f_{out} t \quad (5\text{-}2)$$

$$\Delta\theta = 2\pi f_{out} T_{clk} = \frac{2\pi f_{out}}{f_{clk}} \quad (5\text{-}3)$$

$$\frac{B_{\Delta\theta}}{2^N} = \frac{f_{out}}{f_{clk}}, \quad B_{\Delta\theta} = 2^N \cdot \frac{f_{out}}{f_{clk}} \quad (5\text{-}4)$$

$$S_{out} = A\sin(\theta_{k-1} + \Delta\theta) = A\sin\left[\frac{2\pi}{2^N} \cdot \left(B_{\theta_{k-1}} + B_{\Delta\theta}\right)\right] = Af_{sin}\left(B_{\theta_{k-1}} + B_{\Delta\theta}\right) \quad (5\text{-}5)$$

$$B_{\theta_{k-1}} \approx \frac{\theta_{k-1}}{2\pi} \cdot 2^N \quad (5\text{-}6)$$

5.8 直接数字综合器(DDS)设计

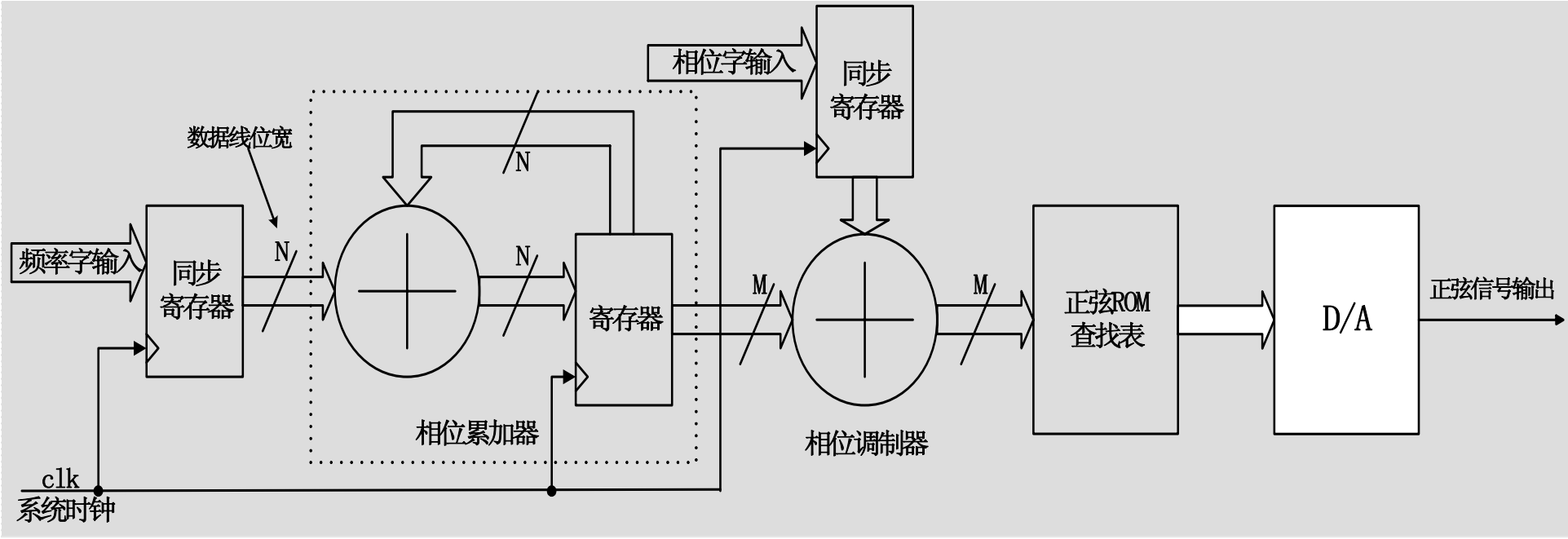

图5-18 基本DDS结构

5.8 直接数字综合器(DDS)设计

关于基本DDS结构的常用参量计算

(1) DDS的输出频率fout。

$$f_{out} = \frac{B_{\Delta\theta}}{2^N} \cdot f_{clk} \qquad (5\text{-}7)$$

(2) DDS的频率分辨率。

$$f_{out} = \frac{f_{clk}}{2^N} \qquad (5\text{-}8)$$

(3) DDS的频率输入字计算。

$$B_{\Delta\theta} = 2^N \cdot \frac{f_{out}}{f_{clk}}$$

5.8 直接数字综合器(DDS)设计

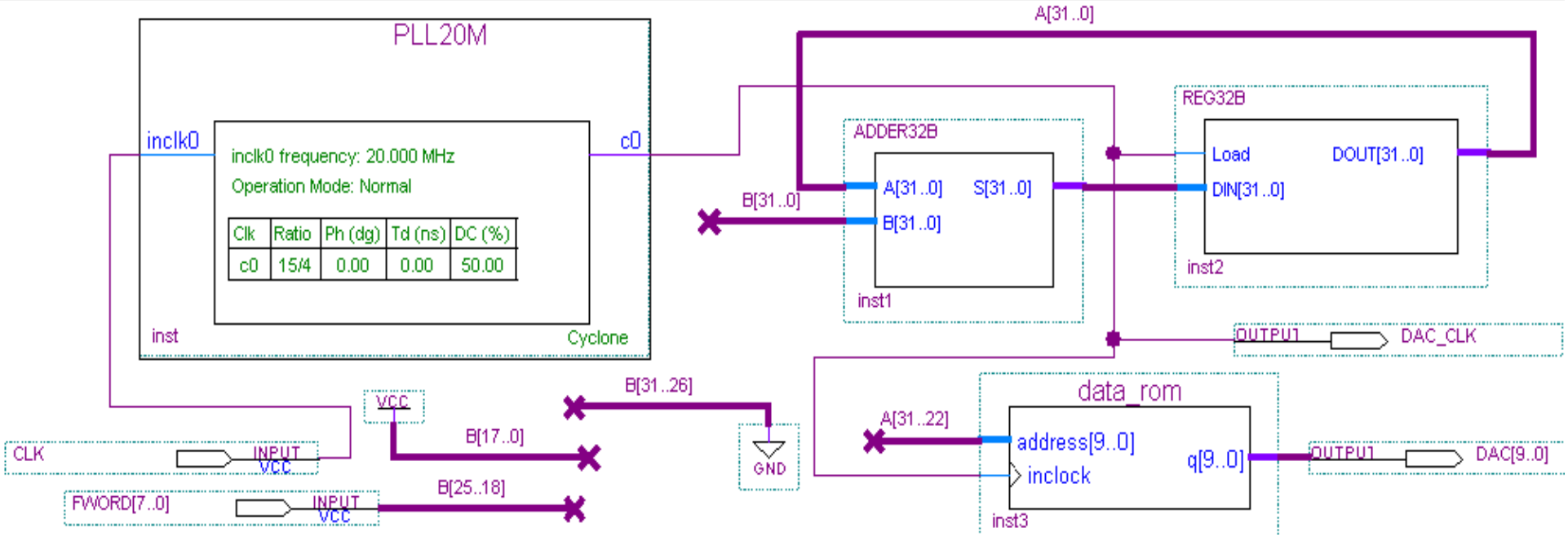

图5-19 DDS．vhd顶层原理图

5.8 直接数字综合器(DDS)设计

【例5-22】

```
LIBRARY ieee ; --波形数据ROM
USE ieee.std_logic_1164.all;
LIBRARY altera_ mf;
USE altera _ mf. Altera _ mf _ components .all;
ENTITY data _ rom IS
PORT
(   address       : IN STD_LOGIC_VECTOR (9 DOWNTO 0);
    inclock       : IN STD_LOGIC ;
    q             : OUT STD_LOGIC_VECTOR (9 DOWNTO 0));
END data _ rom;
 . . .
    init _file => "./data/LUT10X10.mif", --波形数据初始化文件路径
    lpm _hint => "ENABLE_RUNTIME_MOD=YES,  INSTANCE_NAME=rom2",
. . .
END;
```

5.8 直接数字综合器(DDS)设计

【例5-23】

```
LIBRARY IEEE;  --32位加法器模块
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ADDER32B IS
   PORT (  A : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
           B : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
           S : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)    );
END ADDER32B;
ARCHITECTURE behav OF ADDER32B IS
   BEGIN
           S <= A + B;
END behav;
```

5.8 直接数字综合器(DDS)设计

【例5-23】

```
LIBRARY IEEE;  --32位加法器模块
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ADDER32B IS
   PORT (  A : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
           B : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
           S : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)    );
END ADDER32B;
ARCHITECTURE behav OF ADDER32B IS
   BEGIN
           S <= A + B;
END behav;
```

5.8 直接数字综合器(DDS)设计

【例5-24】--32位寄存器模块

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG32B IS
   PORT (  Load : IN STD_LOGIC;
           DIN : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
          DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
END REG32B;
ARCHITECTURE behav OF REG32B IS
BEGIN
   PROCESS (Load, DIN)
  BEGIN
  IF Load‘ EVENT AND Load = '1' THEN    -- 时钟到来时，锁存输入数据
         DOUT <= DIN;
      END IF;
   END PROCESS;
END behav;
```

5.8 直接数字综合器(DDS)设计

【例5-25】rom_data.mif 10位正弦波数据文件，读者可用MATLAB/DSP Builder生成

```
WIDTH=10;
DEPTH=1024;
ADDRESS_RADIX=DEC;
DATA_RADIX=DEC;
CONTENT BEGIN
  0 : 513; 1 : 515; 2 : 518; 3 : 521; 4 : 524; 5 : 527;  6 : 530; 7 : 533;
  8 : 537; 9 : 540; 10 : 543; 11 : 546; 13 : 549; 13 : 552; 14 : 555;
  . . . . . .（略去部分数据）
 1018 : 493; 1019 : 496; 1020 : 499; 1021 : 502; 1022 : 505; 1023 :
508;
END;
```

习 题

5-1. 说明实体，设计实体概念。

5-2. 举例说明GENERIC说明语句和GENERIC映射语句有何用处，并举例说明。

5-3. 说明端口模式INOUT和BUFFER有何异同点。

5-4. 表式C <= A + B中，A、B和C的数据类型都是STD_LOGIC_VECTOR，是否能直接进行加法运算？说明原因和解决方法。

5-5. VHDL中有哪3种数据对象？详细说明它们的功能特点以及使用方法，举例说明数据对象与数据类型的关系。

5-6. 能把任意一种进制的值向一整数类型的数据对象赋值吗？如果能，怎样做？

5-7. 数据类型BIT、INTEGER和BOOLEAN分别定义在哪个库中？

习 题

5-8. 回答有关Bit和Boolean数据类型的问题：
(1) 解释Bit和Boolean类型的区别；
(2) 对于逻辑操作应使用哪种类型？
(3) 关系操作的结果为哪种类型？
(4) IF语句测试的表达式是哪种类型？
5-9. 用两种方法设计8位比较器，比较器的输入是两个待比较的8位数A=[A7..A0]和B=[B7..B0]，输出是 D、E、F。当A=B时D=1；当A>B时E=1；当A<B时F=1。第一种设计方案是常规的比较器设计方法，即直接利用关系操作符进行编程设计；第二种设计方案是利用减法器来完成，通过减法运算后的符号和结果来判别两个被比较值的大小。对两种设计方案的资源耗用情况进行比较，给以解释。
5-10. 什么是固有延时？什么是惯性延时？
5-11. δ 是什么？在VHDL中，δ 有什么用处？

习 题

5-12. 哪些情况下需要用到程序包**STD_LOGIC_UNSIGNED**？试举一例。

5-13. 在**VHDL**设计中，给时序电路清**0**(复位)有两种方法，它们是什么？

5-14. 哪一种复位方法必须将复位信号放在敏感信号表中？给出这两种电路的**VHDL**描述。

5-15. 判断下面**3**个程序中是否有错误，若有则指出错误所在，并给出完整程序。

程序**1**：

```
Signal A, EN : std_logic;
Process (A, EN)
    Variable B : std_logic;
Begin
if EN = 1 then   B <= A;  end if;
end process;
```

习 题

程序2：

```
Architecture one of sample is
   variable a, b, c : integer;
begin
   c <= a + b;
end;
```

程序3：

```
library ieee;
use ieee.std_logic_1164.all;
entity mux21 is
   port ( a, b : in std_logic; sel : in std_logic; c : out std_logic;);
end sam2;
architecture one of mux21 is
begin
if sel = '0' then   c := a; else    c := b;  end if;
end two;
```

习 题

5-16. 根据例**4-23**设计**8**位左移移位寄存器，给出时序仿真波形。

5-17. 将例**5-12**中的**4**个**IF**语句分别用**4**个并列进程语句表达出来。

5-18. 进程有哪几种主要类型？不完全组合进程是由什么原因引起的？有什么特点？如何避免？

5-19. 给触发器复位的方法有哪两种？如果时钟进程中用了敏感信号表，哪种复位方法要求把复位信号放在敏感信号表中？

5-20. 为什么说一条并行赋值语句可以等效为一个进程？如果是这样的话，该语句怎样实现敏感信号的检测？

习 题

5-21. 下述VHDL代码的综合结果会有几个触发器或锁存器？

程序1：

```
architecture rtl of ex is
   signal a, b: std_logic_vector(3 downto 0);
begin
    process(clk)
    begin
       if clk = '1' and clk'event then
            if q(3) /= '1' then  q <= a + b;
             end if;
          end if;
    end process;
end rtl;
```

程序2：

```
architecture rtl of ex is
    signal a, b: std_logic_vector(3 downto 0);
begin
    process (clk)
        variable int : std_logic_vector(3 downto 0);
    begin
        if clk ='1' and clk‘ event then
            if int(3) /= '1' then  int := a + b ; q <= int;
             end if;
        end if;
    end process;
end rtl  ;
```

程序3：

```
architecture rtl  of ex is
    signal a, b，c, d, e: std_logic_vector(3 downto  0);
begin
    process (c, d, e, en)
    begin
        if en ='1'  then  a <= c ;  b <= d;
          else    a <= e;
        end if;
    end process;
end rtl;
```

习 题

5-22. 将以下程序段转换为WHEN_ELSE语句：

```
PROCESS (a, b, c, d)
 BEGIN
 IF a= '0' AND b='1' THEN  next1 <= "1101" ;
    ELSIF  a='0' THEN  next1 <= d ;
 ELSIF  b='1' THEN  next1 <= c ;
     ELSE
     Next1 <= "1011" ;
 END  IF;
END PROCESS;
```

习 题

5-23. 说明以下两程序有何不同，哪一电路更合理？试画出它们的电路。

程序1：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY EXAP IS  PORT ( clk,a,b  : IN STD_LOGIC;
                       y : OUT STD_LOGIC );
END EXAP ;
 ARCHITECTURE behav OF EXAP IS
 SIGNAL x : STD_LOGIC;
BEGIN
 PROCESS
 BEGIN
  WAIT UNTIL CLK ='1' ;
      x <= '0';   y <= '0';
  IF a = b THEN    x <= '1';
  END IF;
  IF x='1' THEN    y <= '1' ;
  END IF ;
 END PROCESS  ;
END behav;
```

程序2：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY EXAP IS  PORT ( clk,a,b  : IN STD_LOGIC;
                       y : OUT STD_LOGIC );
END EXAP ;
 ARCHITECTURE behav OF EXAP IS
 BEGIN
 PROCESS
  VARIABLE x : STD_LOGIC;
 BEGIN
  WAIT UNTIL CLK ='1' ;
     x := '0';    y <= '0';
   IF a = b THEN  x := '1';
  END IF;
  IF x='1' THEN   y <= '1' ;
  END IF ;
 END PROCESS  ;
END behav;
```

实验与实践

5-1. 七段数码显示译码器设计

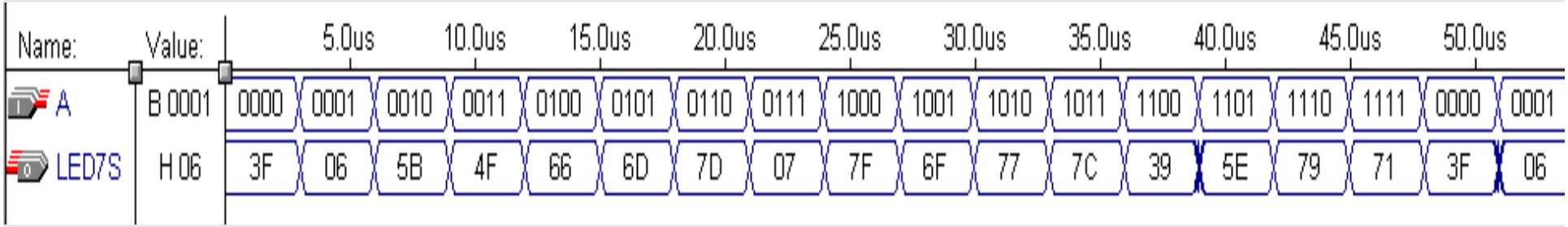

图5-20 7段译码器仿真波形

【例5-26】

```
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 ;
```

实验与实践

5-1. 七段数码显示译码器设计

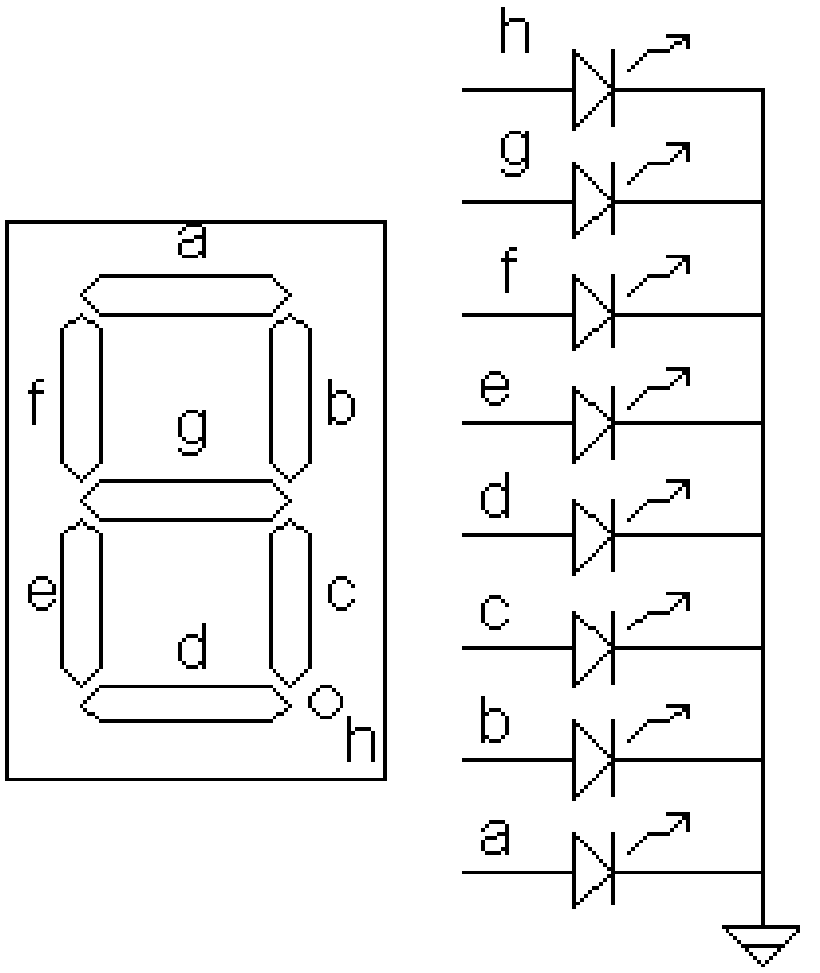

图5-21共阴数码管及其电路

实验与实践

5-1. 七段数码显示译码器设计

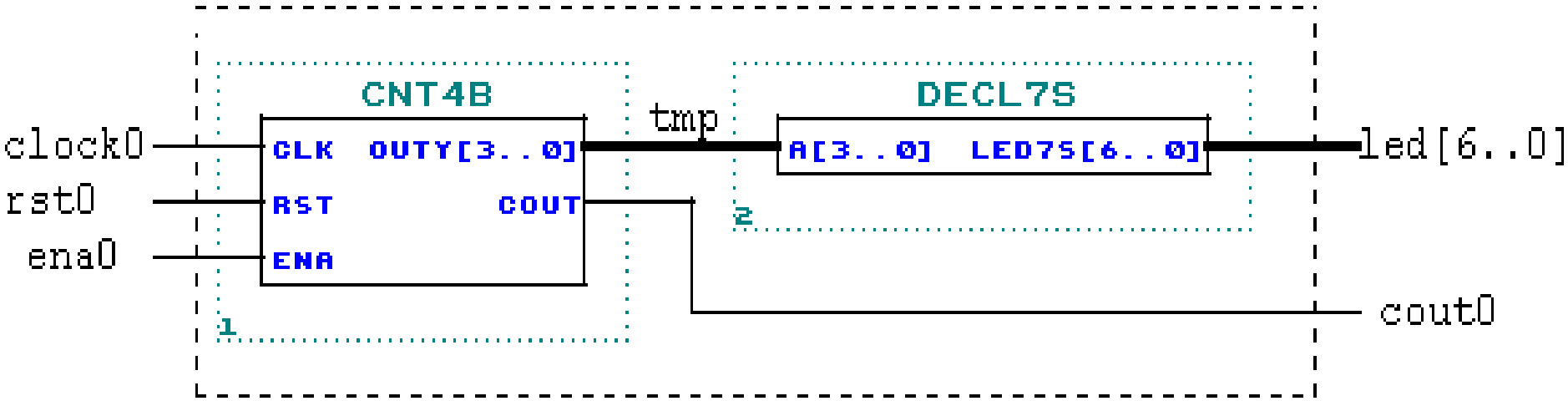

图5-22 计数器和译码器连接电路的顶层文件原理图

实验与实践

5-2. 八位数码扫描显示电路设计

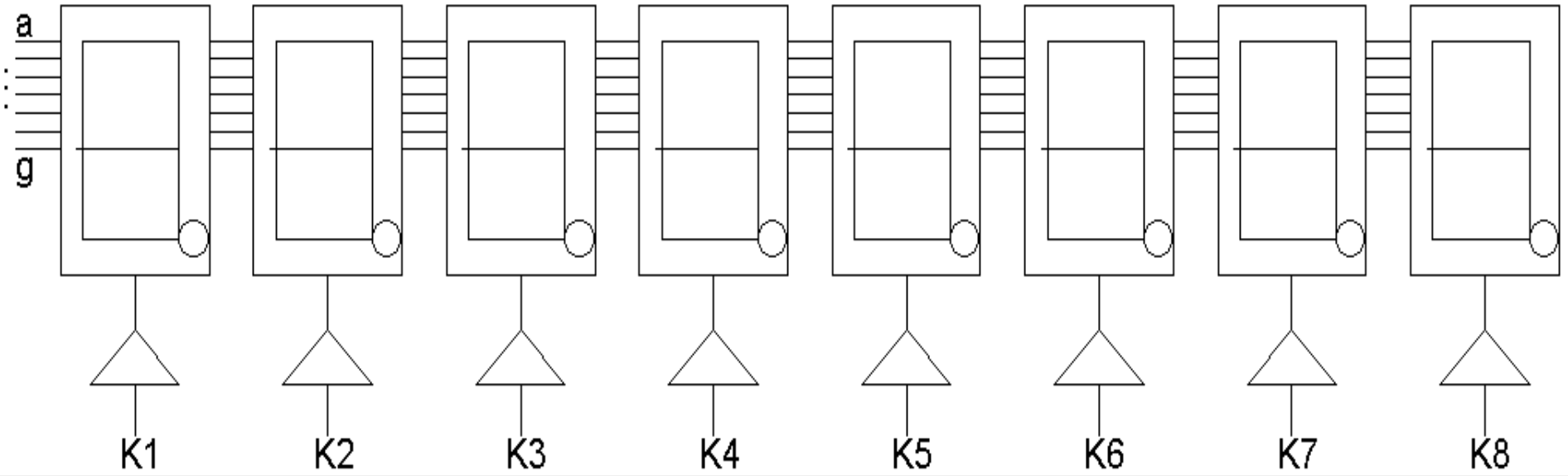

图5-23 8位数码扫描显示电路

【例5-27】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SCAN_LED IS
   PORT (   CLK   : IN STD_LOGIC;
         SG   : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);  --段控制信号输出
         BT   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );--位控制信号输出
 END;
ARCHITECTURE one OF SCAN_LED IS
   SIGNAL CNT8  : STD_LOGIC_VECTOR(2 DOWNTO 0);
   SIGNAL    A  : INTEGER RANGE 0 TO 15;
BEGIN
P1：PROCESS( CNT8 )
   BEGIN
     CASE  CNT8  IS
      WHEN "000" =>  BT <= "00000001" ; A <= 1 ;
      WHEN "001" =>  BT <= "00000010" ; A <= 3 ;
      WHEN "010" =>  BT <= "00000100" ; A <= 5 ;
      WHEN "011" =>  BT <= "00001000" ; A <= 7 ;
      WHEN "100" =>  BT <= "00010000" ; A <= 9 ;
      WHEN "101" =>  BT <= "00100000" ; A <= 11 ;
      WHEN "110" =>  BT <= "01000000" ; A <= 13 ;
      WHEN "111" =>  BT <= "10000000" ; A <= 15 ;
      WHEN OTHERS =>  NULL ;
     END CASE ;
   END PROCESS P1;
   P2：PROCESS(CLK)
     BEGIN
      IF CLK'EVENT AND CLK = '1' THEN CNT8 <= CNT8 + 1;
      END IF;
                END PROCESS P2 ;
    P3：PROCESS( A ) --译码电路
    BEGIN
     CASE  A  IS
      WHEN 0  => SG <= "0111111";  WHEN 1  => SG <= "0000110";
      WHEN 2  => SG <= "1011011";  WHEN 3  => SG <= "1001111";
      WHEN 4  => SG <= "1100110";  WHEN 5  => SG <= "1101101";
      WHEN 6  => SG <= "1111101";  WHEN 7  => SG <= "0000111";
      WHEN 8  => SG <= "1111111";  WHEN 9  => SG <= "1101111";
      WHEN 10 => SG <= "1110111";  WHEN 11 => SG <= "1111100";
      WHEN 12 => SG <= "0111001";  WHEN 13 => SG <= "1011110";
      WHEN 14 => SG <= "1111001";  WHEN 15 => SG <= "1110001";
      WHEN OTHERS =>  NULL ;
     END CASE ;
    END PROCESS P3;
END;
```

实验与实践

5-3. 32位并进/并出移位寄存器设计

5-4 直接数字频率合成器(DDS)设计

5-5 数字移相信号发生器设计

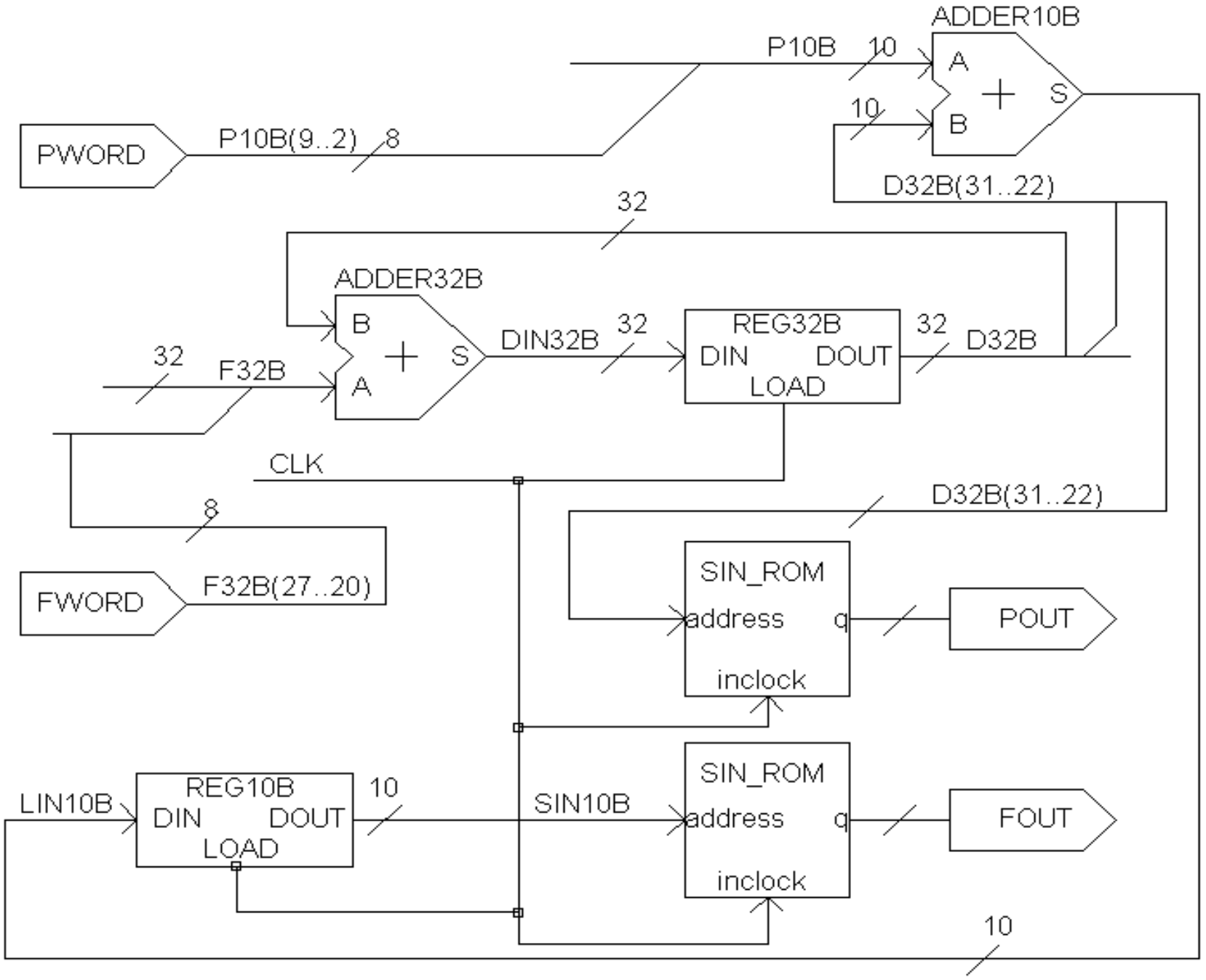

图5-24 数字移相信号发生器电路模型图

【例5-28】

```
LIBRARY IEEE;-- 数字移相信号发生器顶层设计文件，元件连接结构参考图5-24。
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY DDS_VHDL IS                                    -- 顶层设计
   PORT ( CLK : IN STD_LOGIC; --系统时钟
        FWORD : IN STD_LOGIC_VECTOR(7 DOWNTO 0);    --频率控制字
        PWORD : IN STD_LOGIC_VECTOR(7 DOWNTO 0);    --相位控制字
       FOUT : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);     --可移相正弦信号输出
       POUT : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)  ); --参考信号输出
 END;
ARCHITECTURE one OF DDS_VHDL IS
    COMPONENT REG32B                --32位锁存器
      PORT ( LOAD : IN STD_LOGIC;
            DIN : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
           DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
   END COMPONENT;
    COMPONENT REG10B                --10位锁存器
      PORT ( LOAD : IN STD_LOGIC;
            DIN : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
           DOUT : OUT STD_LOGIC_VECTOR(9 DOWNTO 0) );
   END COMPONENT;
   COMPONENT ADDER32B               --32位加法器
     PORT ( A : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
          B : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
          S : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)    );
   END COMPONENT;
```

(接下页)

```
COMPONENT ADDER10B              --10位加法器
    PORT (  A : IN  STD_LOGIC_VECTOR(9 DOWNTO 0);
            B : IN  STD_LOGIC_VECTOR(9 DOWNTO 0);
            S : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)    );
  END COMPONENT;
  COMPONENT SIN_ROM                 --10位地址10位数据正弦信号数据ROM
   PORT        ( address       : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
                 inclock       : IN STD_LOGIC ;
                 q             : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)       );
  END COMPONENT;
   SIGNAL F32B, D32B, DIN32B   : STD_LOGIC_VECTOR(31 DOWNTO 0);
   SIGNAL P10B, LIN10B, SIN10B : STD_LOGIC_VECTOR( 9 DOWNTO 0);
BEGIN
F32B(27 DOWNTO 20)<=FWORD ;     F32B(31 DOWNTO 28)<="0000";
F32B(19 DOWNTO 0)<="00000000000000000000" ;
P10B( 9 DOWNTO 2)<=PWORD ;     P10B( 1 DOWNTO 0)<="00" ;
u1 : ADDER32B  PORT MAP( A=>F32B,B=>D32B, S=>DIN32B );
u2 :  REG32B  PORT MAP( DOUT=>D32B,DIN=> DIN32B, LOAD=>CLK );
u3 : SIN_ROM  PORT MAP( address=>SIN10B, q=>FOUT, inclock =>CLK );
u4 : ADDER10B  PORT MAP( A=>P10B,B=>D32B(31 DOWNTO
22),S=>LIN10B );
u5 :  REG10B  PORT MAP( DOUT=>SIN10B,DIN=>LIN10B, LOAD=>CLK );
u6 : SIN_ROM  PORT MAP( address=>D32B(31 DOWNTO 22), q=>POUT,
inclock =>CLK );
END;
```