EDA 技术实用教程

第 6 章

VHDL设计进阶

6.1 数据对象

6.1.1 常数

常数定义的一般表述如下：

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
```

6.1 数据对象

6.1.2 变量

定义变量的一般表述如下：

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定义为标准逻辑位数据类型, 初始值是1
```

变量赋值的一般表述如下：

目标变量名 **:=** 表达式

6.1 数据对象

6.1.3 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 ;
```

6.1 数据对象

6.1.4 进程中的信号与变量赋值

表6-1　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

6.1 数据对象

6.1.4 进程中的信号与变量赋值

【例6-1】

```
. . .
  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;
```

6.1 数据对象

6.1.4 进程中的信号与变量赋值

【例6-2】

```
. . .
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;
```

6.1.4 进程中的信号与变量赋值

【例6-3】

```
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;
```

6.1 数据对象

6.1.4 进程中的信号与变量赋值

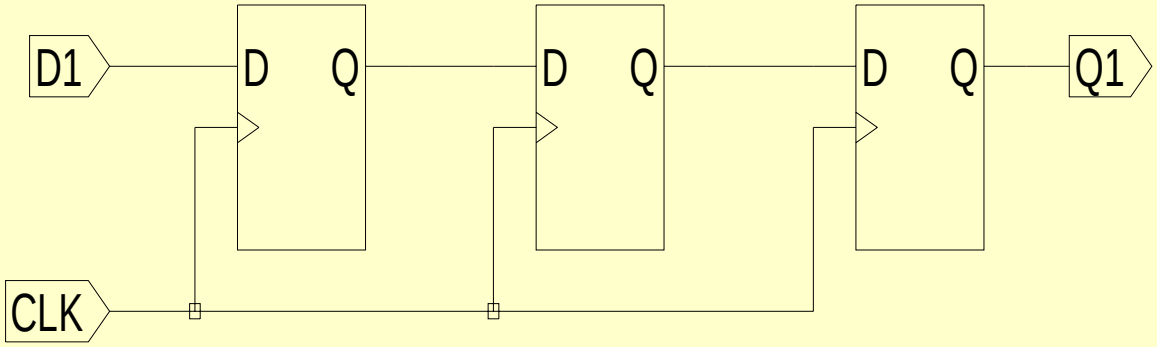

图6-1 例6-3的RTL电路

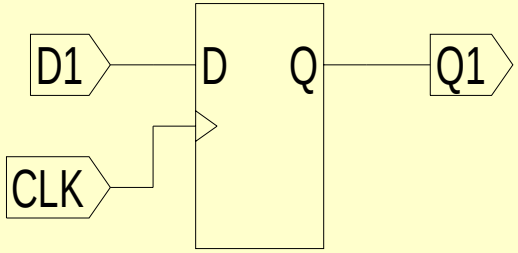

图6-2 D触发器电路

6.1.4 进程中的信号与变量赋值

【例6-4】

```
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 ;
```

6.1 数据对象

6.1.4 进程中的信号与变量赋值

【例6-5】

```
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6-6】

```
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;
```

【例6-7】

```
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;
```

6.1.4 进程中的信号与变量赋值

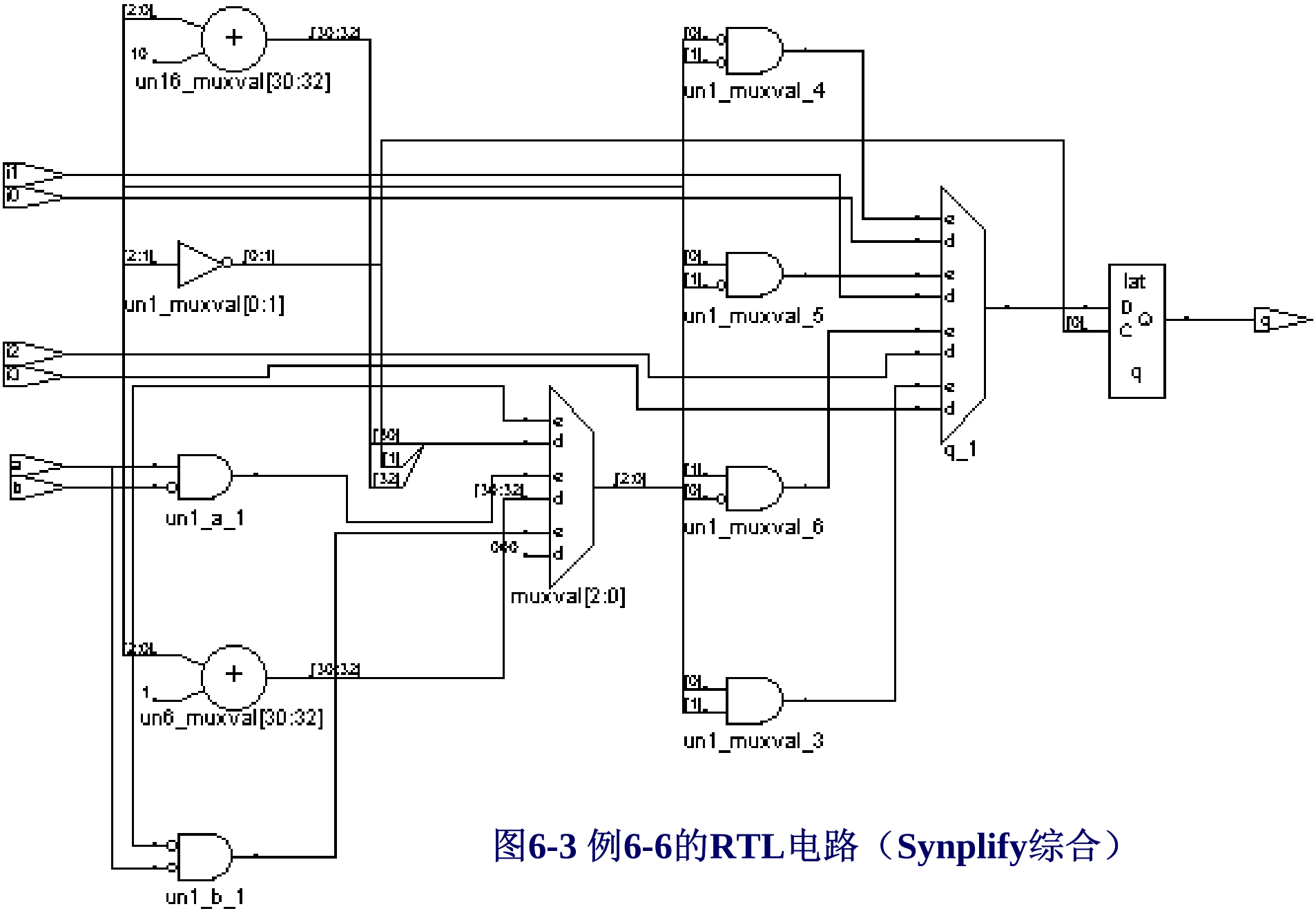

图6-3 例6-6的RTL电路（Synplify综合）

6.1.4 进程中的信号与变量赋值

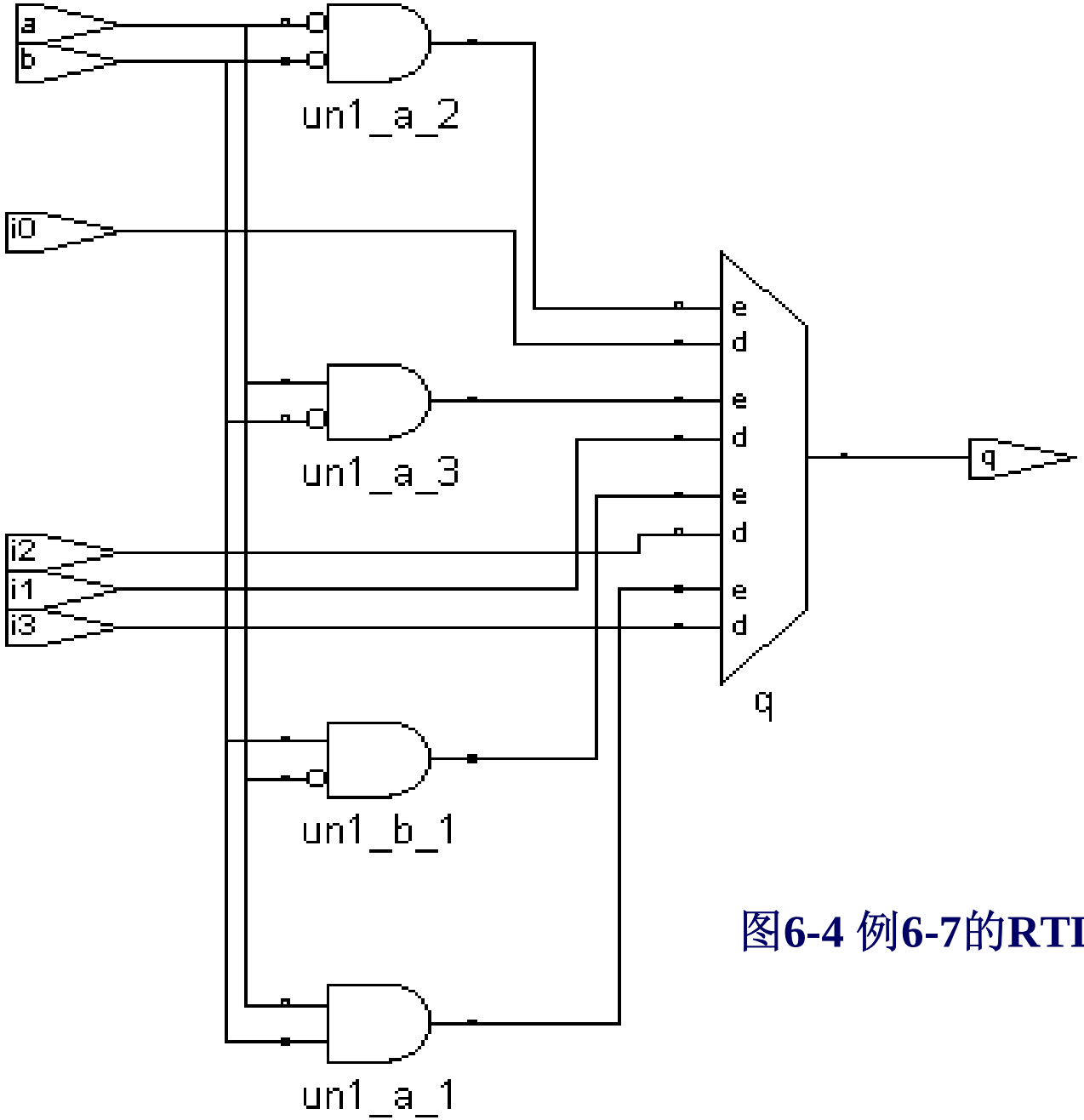

图6-4 例6-7的RTL电路（Synplify综合）

6.1 数据对象

6.1.4 进程中的信号与变量赋值

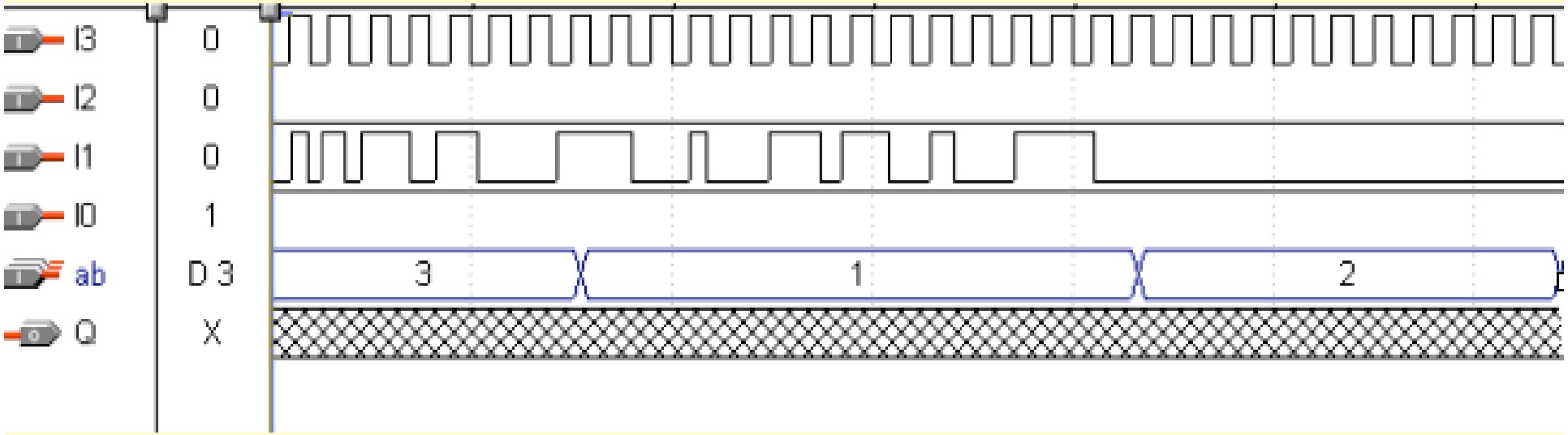

图6-5 例6-6中错误的工作时序

6.1 数据对象

6.1.4 进程中的信号与变量赋值

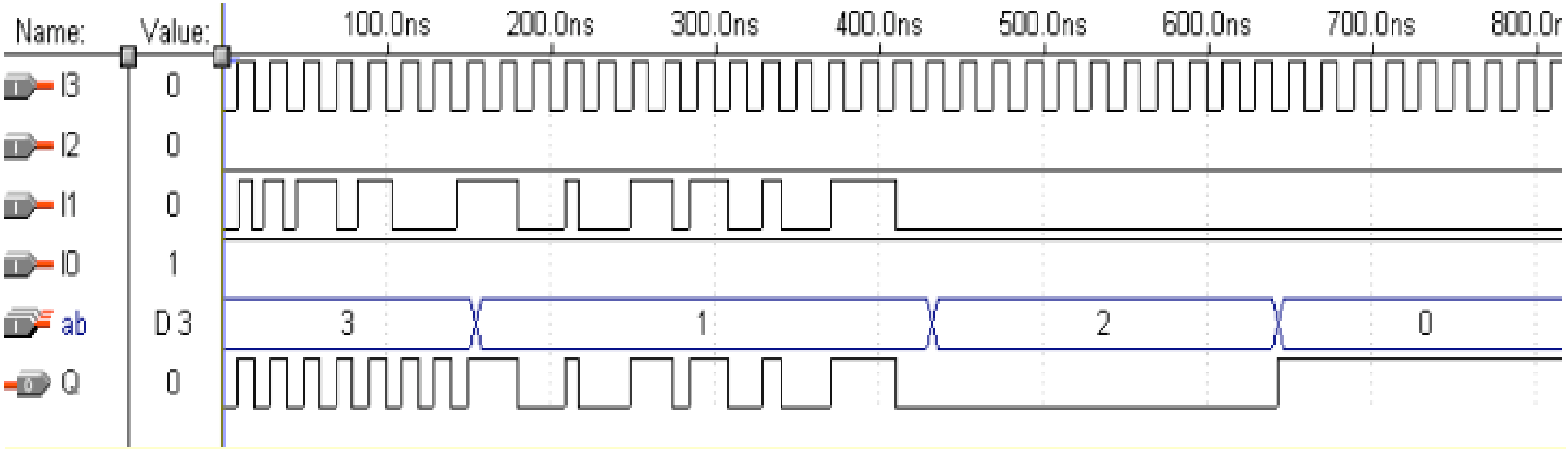

图6-6 例6-7中正确的工作时序

【例6-8】

```
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);
```

（接下页）

KX 康芯科技

（接上页）

```
  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;
```

6.1 数据对象

6.1.4 进程中的信号与变量赋值

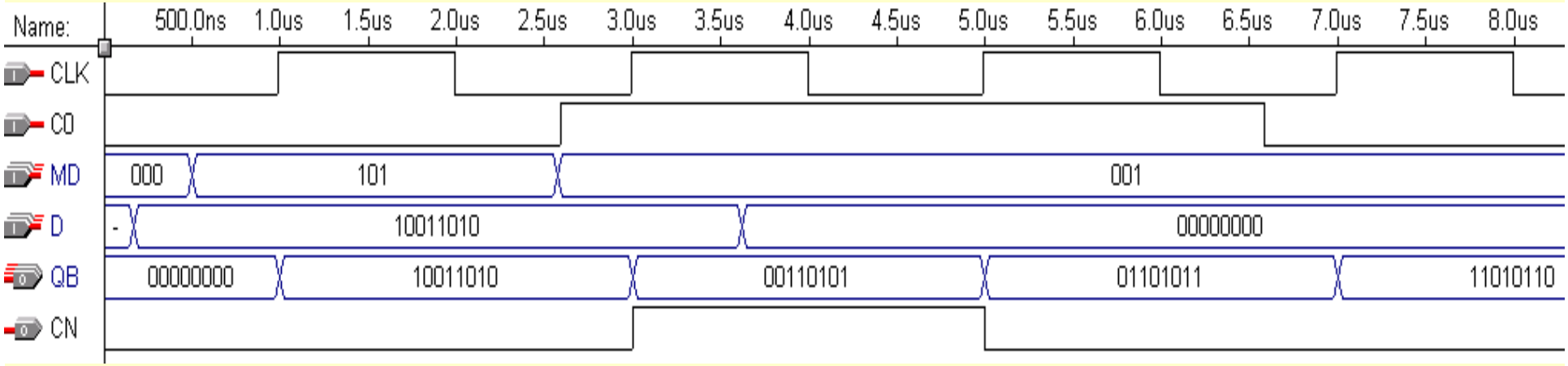

图6-7 例6-8中带进位循环左移仿真波形（MD="001"）

6.2 双向和三态电路信号赋值例解

6.2.1 三态门设计

【例6-9】

```
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;
```

6.2 双向和三态电路信号赋值例解

6.2.1 三态门设计

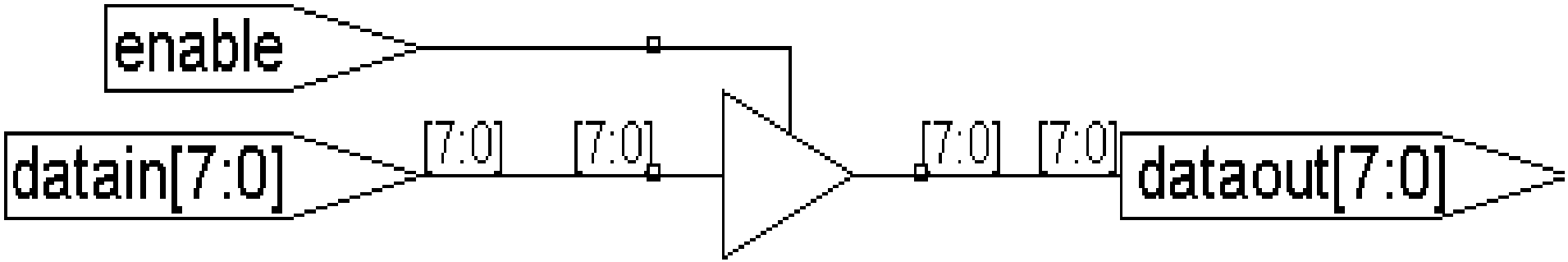

图6-8 8位3态控制门电路（Synplify综合）

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

【例6-10】

```
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;
```

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

【例6-11】

(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;
```

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

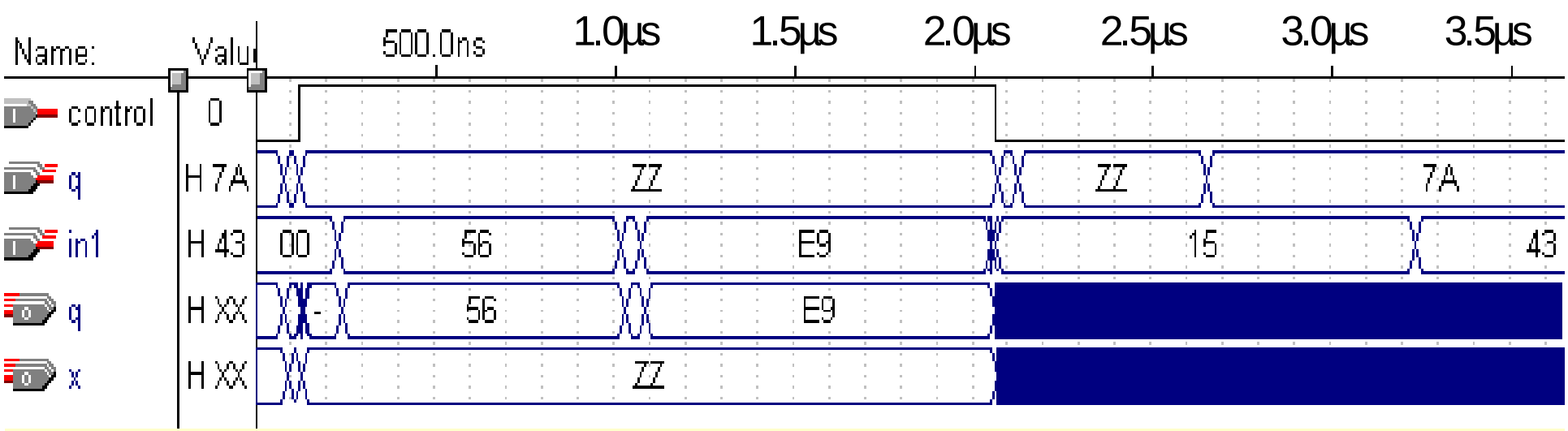

图6-9 例6-10的仿真波形图

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

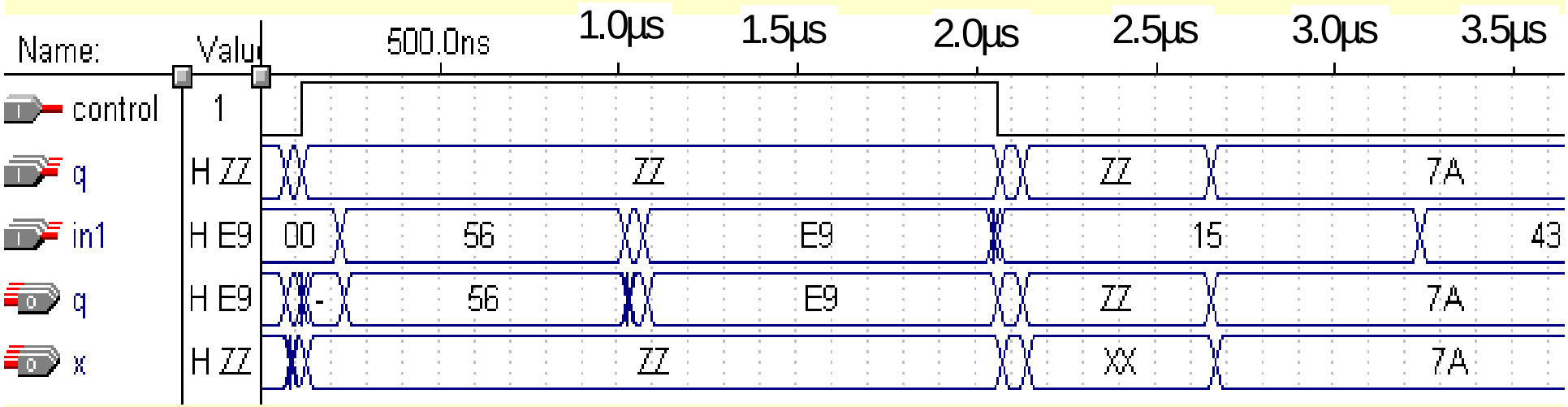

图6-10 例6-11的仿真波形图

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

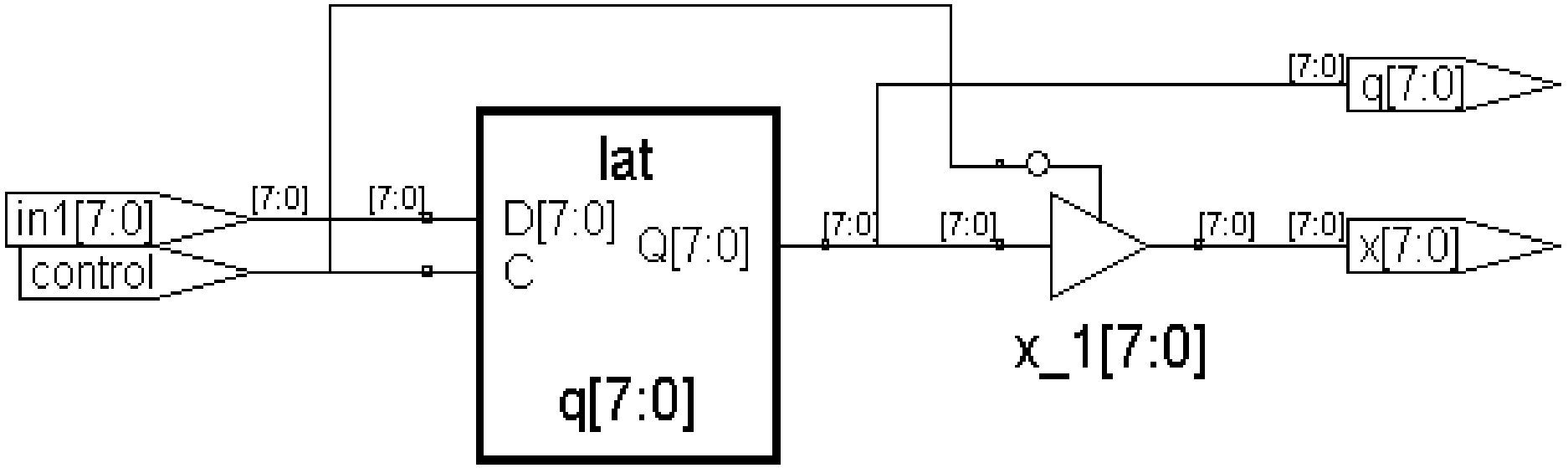

图6-11 例6-10的综合结果（Synplify综合）

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

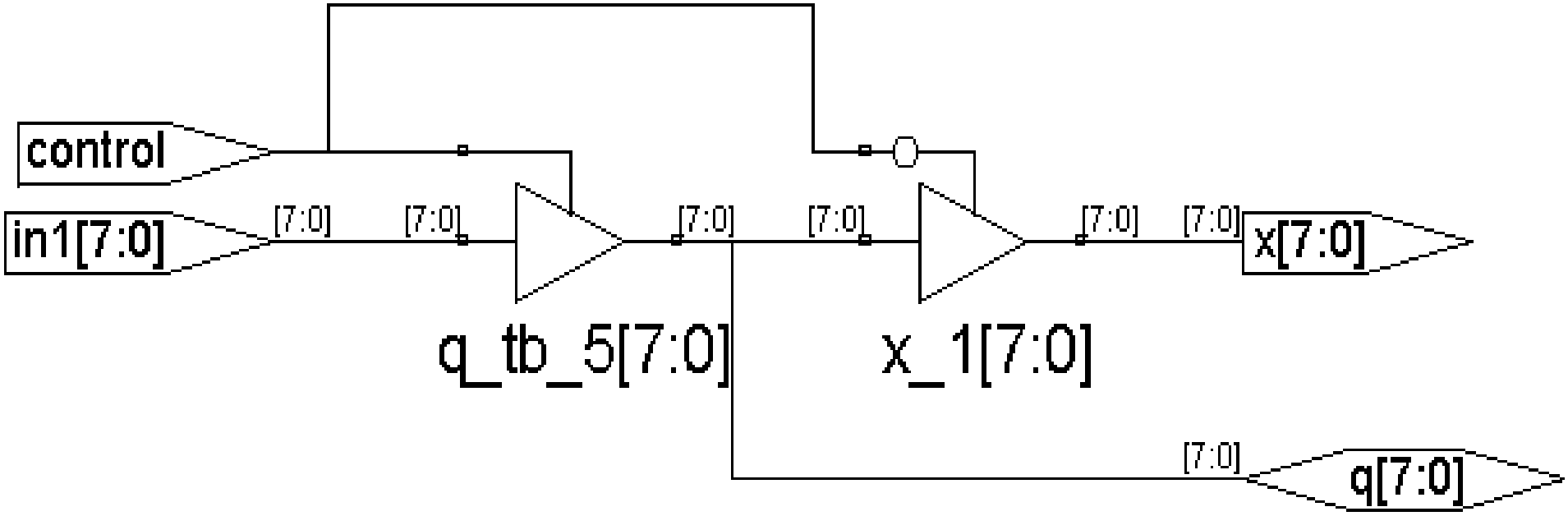

图6-12 例6-11的综合结果（Synplify综合）

6.2.3 三态总线电路设计

【例6-12】

```
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 ;
```

（接下页）

6.2.3 三态总线电路设计

（接上页）

```
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;
```

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

【例6-13】（注：MaxplusII不支持本例）

```
library ieee;
use ieee.std_logic_1164.all;
entity tri2 is
port (ctl : in  std_logic_vector(1 downto 0);
         datain1，  datain2，datain3，  datain4 :
              in std_logic_vector(7 downto 0);
        q : out std_logic_vector(7 downto 0)   );
end tri2;
architecture body_tri of tri2 is
begin
 q <= datain1   when ctl="00" else  (others =>'Z') ;
 q <= datain2   when ctl="01" else  (others =>'Z') ;
 q <= datain3   when ctl="10" else  (others =>'Z') ;
 q <= datain4   when ctl="11" else  (others =>'Z') ;
end body_tri;
```

6.2 双向和三态电路信号赋值例解

6.2.2 双向端口设计

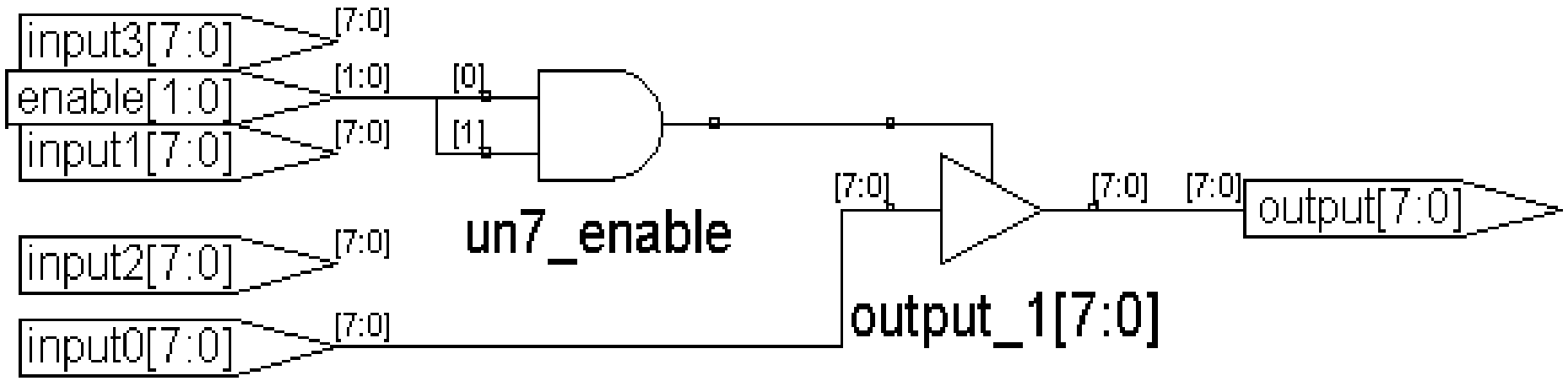

图6-13 例6-12错误的综合结果（Synplify综合结果）

6.2.2 双向端口设计

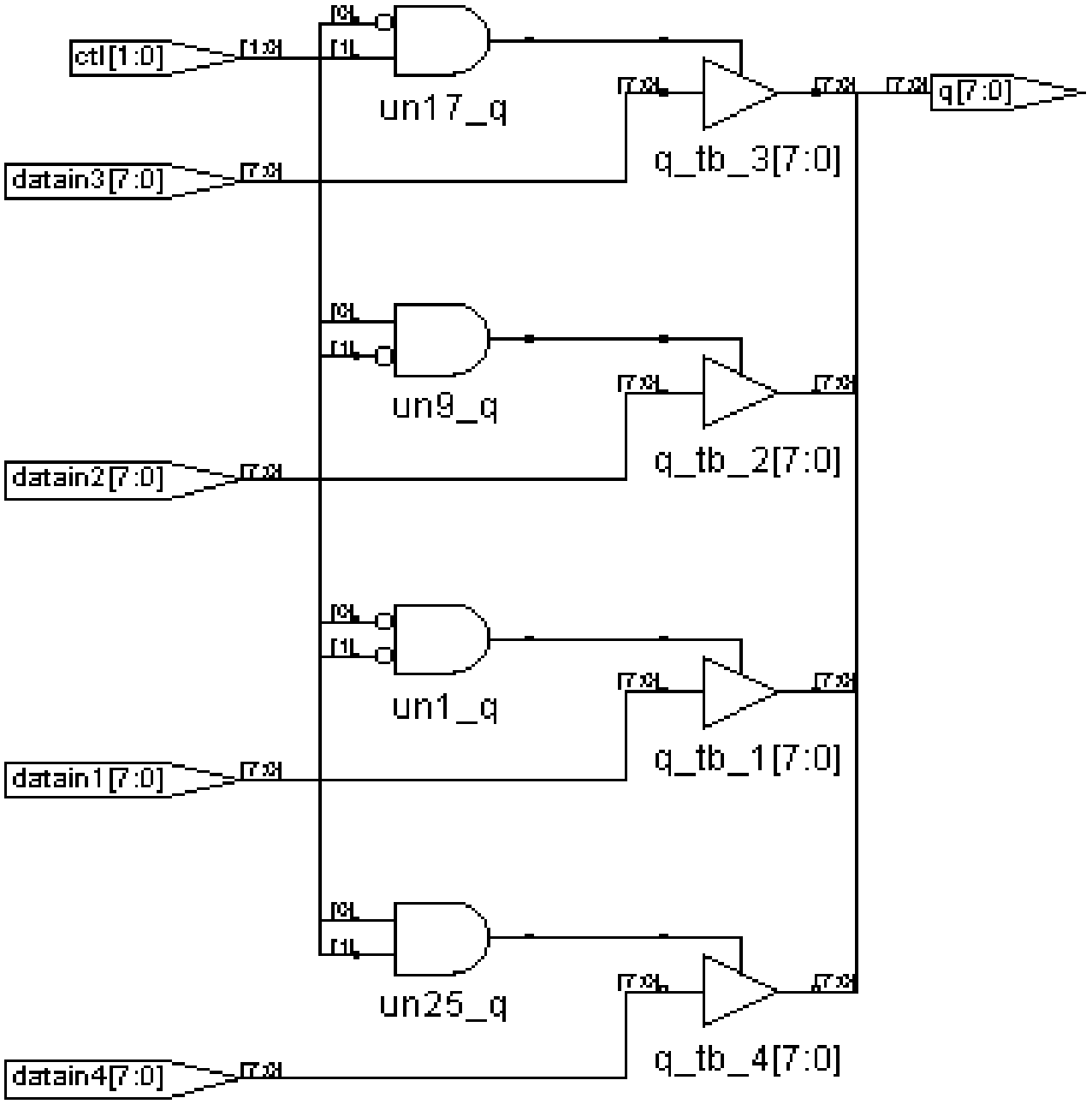

图6-14 例6-13正确的综合结果（Synplify综合结果）

6.3 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

6.3 IF语句概述

【例6-14】

```
  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;
```

6.3 IF语句概述

表6-2 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6-15】

```
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;
```

6.4 进程语句归纳

6.4.1 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];
```

6.4 进程语句归纳

6.4.2 进程结构组成

进程说明部分 → 数据类型、常数、变量、属性、子程序

顺序描述语句部分 → 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

6.4 进程语句归纳

6.4.3 进程要点

1. **PROCESS为一无限循环语句**
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;
```

6.4 进程语句归纳

6.4.3 进程要点

3. 进程必须由敏感信号的变化来启动

4. 进程语句本身是并行语句

【例6-16】

```
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;
```

6.4 进程语句归纳

6.4.3 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

6.5 并行语句例解

【例6-17】

```
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 ;
. . .
```

6.6 仿真延时

6.6.1 固有延时

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

z <= x XOR y ;

B <=  A AFTER 20ns ; --固有延时模型
```

6.6 仿真延时

6.6.2 传输延时

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

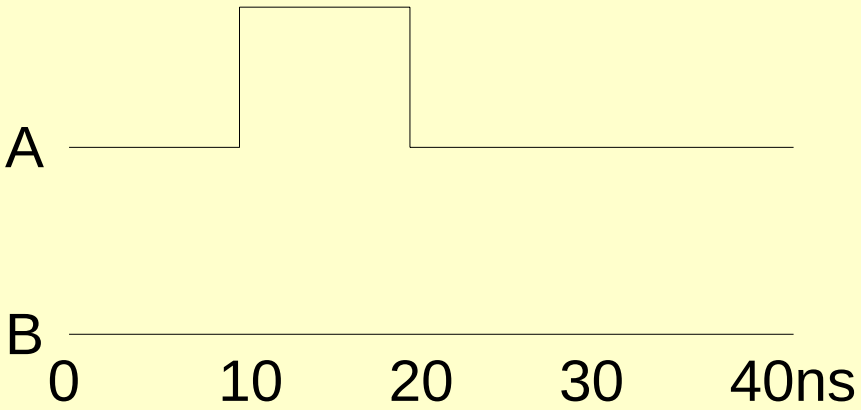

图6-15 固有延时输入输出波形

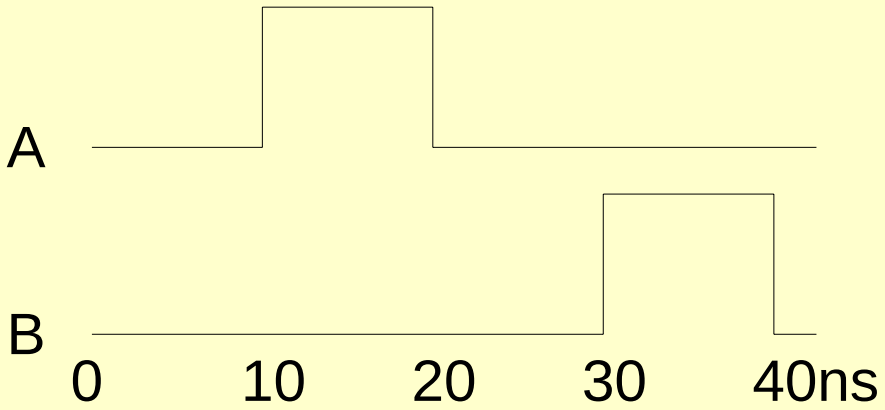

图6-16传输延时输入输出波形

6.6 仿真延时

6.6.3 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。

习 题

6-1. 什么是固有延时？什么是惯性延时？

6-2. δ 是什么？在**VHDL**中，δ 有什么用处？

6-3. 哪些情况下需要用到程序包**STD_LOGIC_UNSIGNED**？试举一例。

6-4. 说明信号和变量的功能特点，应用上的异同点。

6-5. 在**VHDL**设计中，给时序电路清**0**(复位)有两种方法，它们是什么？

6-6. 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。

6-7. 什么是重载函数？重载算符有何用处？如何调用重载算符函数？

习 题

6-8. 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;
```

程序2：

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

习 题

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

6-9. 根据例**4-23**设计**8**位左移移位寄存器，给出时序仿真波形。

6-10. 将例**6-12**中的**4**个**IF**语句分别用**4**个并列进程语句表达出来。

实 验 与 设 计

6-1. 七段数码显示译码器设计

(1) 实验目的：学习7段数码显示译码器设计；学习VHDL的CASE语句应用及多层次设计方法。

(2) 实验原理：7段数码是纯组合电路，通常的小规模专用IC，如74或4000系列的器件只能作十进制BCD码译码，然而数字系统中的数据处理和运算都是2进制的，所以输出表达都是16进制的，为了满足16进制数的译码显示，最方便的方法就是利用译码程序在FPGA/CPLD中来实现。例6-18作为7段译码器，输出信号LED7S的7位分别接如图6-18数码管的7个段，高位在左，低位在右。例如当LED7S输出为“1101101”时，数码管的7个段：g、f、e、d、c、b、a分别接1、1、0、1、1、0、1；接有高电平的段发亮，于是数码管显示“5”。注意，这里没有考虑表示小数点的发光管，如果要考虑，需要增加段h，例6-18中的`LED7S:OUT STD_LOGIC_VECTOR(6 DOWNTO 0)`应改为 `... (7 DOWNTO 0)` 。

实 验 与 设 计

(3) 实验内容1：说明例**6-18**中各语句的含义，以及该例的整体功能。在**QuartusII**上对该例进行编辑、编译、综合、适配、仿真，给出其所有信号的时序仿真波形。

提示：用输入总线的方式给出输入信号仿真数据，仿真波形示例图如图**6-17**所示。

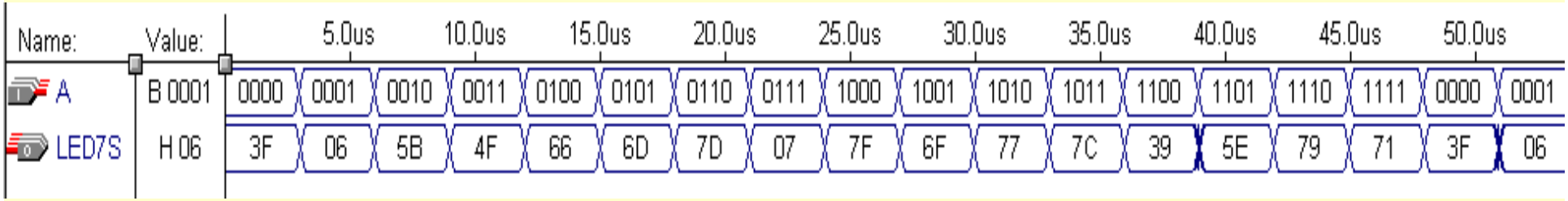

图6-17 7段译码器仿真波形

【例6-18】

```
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 设 计

(4) 实验内容2：引脚锁定及硬件测试。建议选**GW48**系统的实验电路模式**6**（参考附录图**6**），用数码**8**显示译码输出**(PIO46-PIO40)**，键**8**、键**7**、键**6**和键**5**四位控制输入，硬件验证译码器的工作性能。

(5) 实验内容3：用第**4**章介绍的例化语句，按图**6-19**的方式连接成顶层设计电路（用**VHDL**表述），图中的**CNT4B**是一个**4**位二进制加法计数器，可以由例**4-22**修改获得；模块**DECL7S**即为例**6-18**实体元件，重复以上实验过程。注意图**6-20**中的**tmp**是**4**位总线，**led**是**7**位总线。对于引脚锁定和实验，建议选电路模式**6**，用数码**8**显示译码输出，用键**3**作为时钟输入**(**每按**2**次键为**1**个时钟脉冲**)**，或直接接时钟信号**clock0**。

(8) 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、硬件测试和实验过程；设计程序、程序分析报告、仿真波形图及其分析报告。

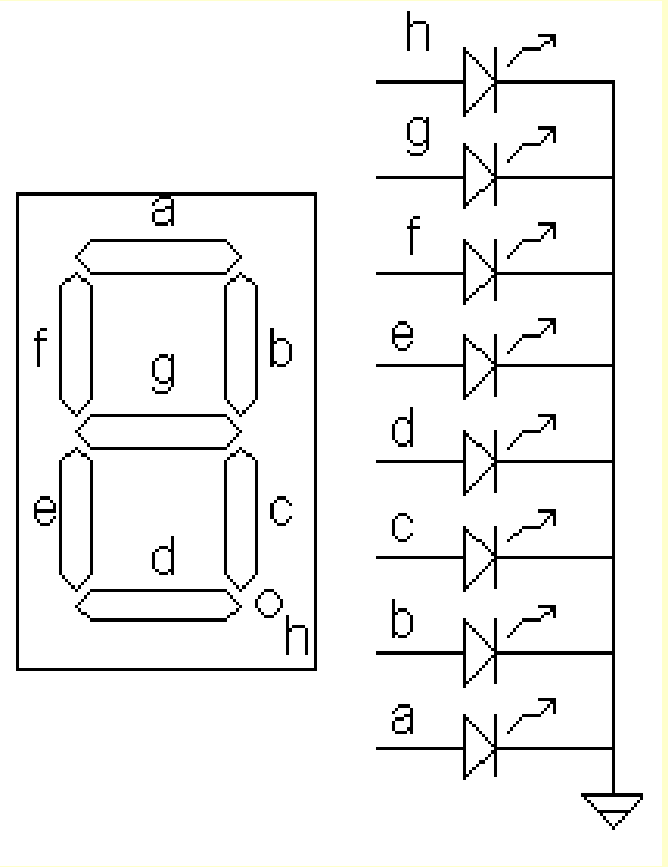

图6-18共阴数码管及其电路

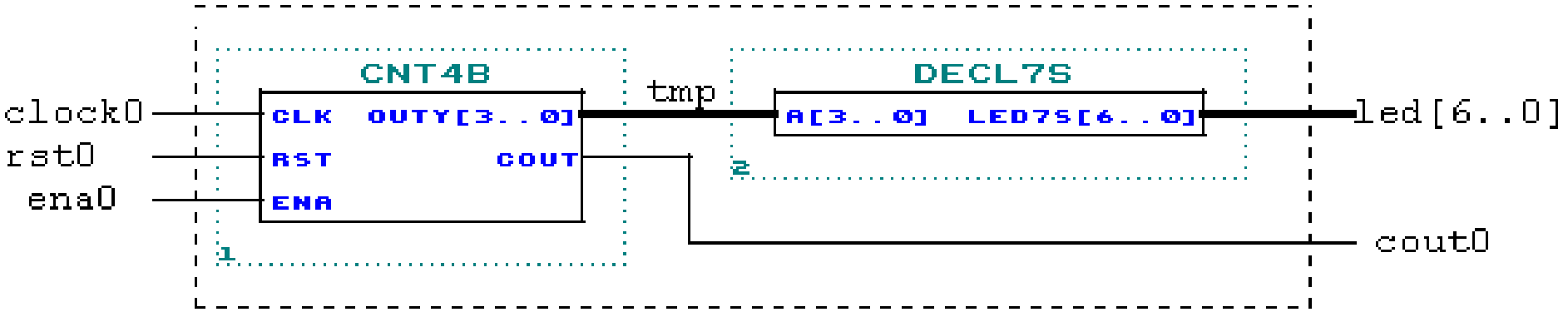

图6-19 计数器和译码器连接电路的顶层文件原理图

KX 康心科技

实 验 与 设 计

6-2. 八位数码扫描显示电路设计

(1) 实验目的：学习硬件扫描显示电路的设计。

(2) 实验原理：图6-20所示的是8位数码扫描显示电路，其中每个数码管的8个段：h、g、f、e、d、c、b、a（h是小数点）都分别连在一起，8个数码管分别由8个选通信号k1、k2、…k8来选择。被选通的数码管显示数据，其余关闭。如在某一时刻，k3为高电平，其余选通信号为低电平，这时仅k3对应的数码管显示来自段信号端的数据，而其它7个数码管呈现关闭状态。根据这种电路状况，如果希望在8个数码管显示希望的数据，就必须使得8个选通信号k1、k2、…k8分别被单独选通，并在此同时，在段信号输入口加上希望在该对应数码管上显示的数据，于是随着选通信号的扫变，就能实现扫描显示的目的。

实 验 与 设 计

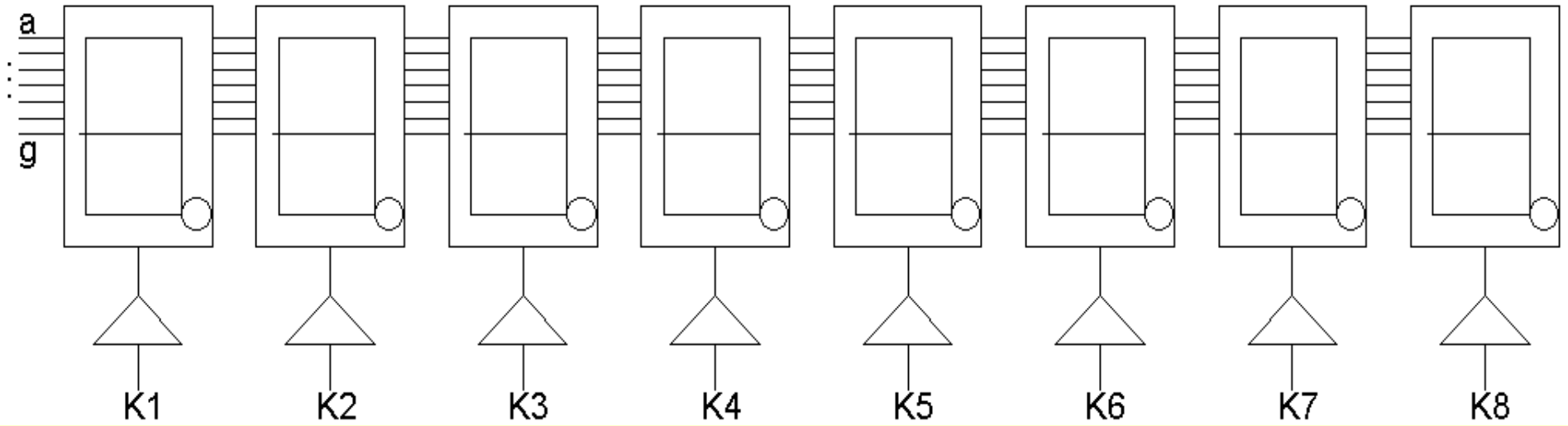

图6-20 8位数码扫描显示电路

【【例6-19】

```
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;
```

接下页

接上页

```
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 设 计

(3) 实验内容1：说明例**6-19**中各语句的含义，以及该例的整体功能。对该例进行编辑、编译、综合、适配、仿真，给出仿真波形。实验方式：若考虑小数点，**SG**的**8**个段分别与**PIO49**、**PIO48**、…、**PIO42**（高位在左）、**BT**的**8**个位分别与**PIO34**、**PIO35**、…、**PIO41**（高位在左）；电路模式不限，引脚图参考附录图**10**。将**GW48EDA**系统左下方的拨码开关全部向上拨，这时实验系统的**8**个数码管构成图**6-20**的电路结构，时钟**CLK**可选择**clock0**，通过跳线选择**16384Hz**信号。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。

(4) 实验内容2：修改例**6-19**的进程**P1**中的显示数据直接给出的方式，增加**8**个**4**位锁存器，作为显示数据缓冲器，使得所有**8**个显示数据都必须来自缓冲器。缓冲器中的数据可以通过不同方式锁入，如来自**A/D**采样的数据、来自分时锁入的数据、来自串行方式输入的数据，或来自单片机等。

实 验 与 设 计

6-3. 数控分频器的设计

(1) 实验目的：学习数控分频器的设计、分析和测试方法。

(2) 实验原理：数控分频器的功能就是当在输入端给定不同输入数据时，将对输入的时钟信号有不同的分频比，数控分频器就是用计数值可并行预置的加法计数器设计完成的，方法是将计数溢出位与预置数加载输入信号相接即可，详细设计程序如例**6-20**所示。

(3) 分析：根据图**6-21**的波形提示，分析例**6-20**中的各语句功能、设计原理及逻辑功能，详述进程**P_REG**和**P_DIV**的作用，并画出该程序的**RTL**电路图。

实 验 与 设 计

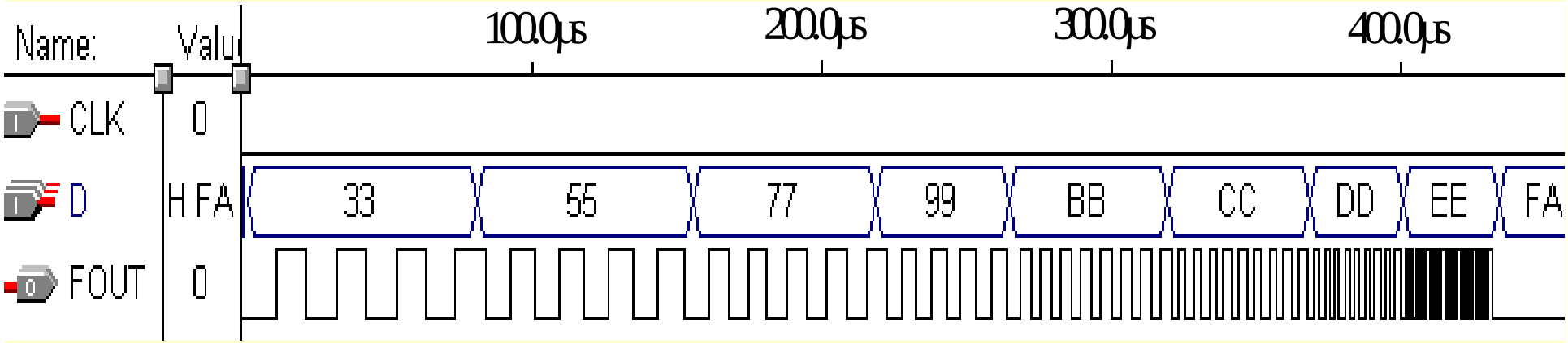

图6-21 当给出不同输入值D时，FOUT输出不同频率(CLK周期=50ns)

实 验 与 设 计

(4) 仿真：输入不同的CLK频率和预置值D，给出如图6-21的时序波形。

(5) 实验内容1：在实验系统上硬件验证例6-20的功能。可选实验电路模式1（参考附录图3）；键2/键1负责输入8位预置数D(PIO7-PIO0)；CLK由clock0输入，频率选65536Hz或更高(确保分频后落在音频范围)；输出FOUT接扬声器(SPKER)。编译下载后进行硬件测试：改变键2/键1的输入值，可听到不同音调的声音。

(6) 实验内容2：将例6-20扩展成16位分频器，并提出此项设计的实用示例，如PWM的设计等。

(7) 思考题：怎样利用2个例6-20给出的模块设计一个电路，使其输出方波的正负脉宽的宽度分别由可两个8位输入数据控制？

(8) 实验报告：根据以上的要求，将实验项目分析设计，仿真和测试写入实验报告。

【例6-20】

```
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;
```

接下页

接上页

```
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;
```

6-4. 32位并进/并出移位寄存器设计

仅用例6-8一个8位移位寄存器，再增加一些电路，如4个8位锁存器等，设计成为一个能为32位二进制数进行不同方式移位的移位寄存器。这个电路模型十分容易用到CPU的设计中。