EDA技术实用教程

第5章

VHDL设计进阶

5.1 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" ; --定义常数为标准位矢类型
CONSTANT  DATAIN : INTEGER := 15 ;              --定义常数为整数类型
```

5.1 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0~15
VARIABLE  d : STD LOGIC := '1';  --变量d定义为标准逻辑位数据类型，初始值是1
```

目标变量名 := 表达式 ;

```
VARIABLE  x, y : INTEGER RANGE 15 DOWNTO 0; --分别定义变量 x 和 y 为整数类型
VARIABLE  a, b : STD_LOGIC_VECTOR(7 DOWNTO 0) ;
x := 11 ;                                  --整数直接赋值
y := 2 + x ;                               --运算表达式赋值，y 也是整数变量
a := b                                     --b 向 a 赋值
a(0 TO 5) := b(2 TO 7) ;                   --位矢量类型赋值
```

5.1 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;  -- AFTER是关键词
```

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

5.1 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

表 5-1　信号与变量赋值语句功能的比较

比较对象	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，有延时	立即赋值，无延时
与 Verilog 对比	信号赋值类似于非阻塞式赋值	变量赋值类似于阻塞式赋值

5.1 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

【例 5-1】使变量赋值的时序模块设计

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

【例 5-2】使用信号赋值的时序模块设计

```
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】使信号赋值的时序模块设计

```
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 ;
```

【例 5-4】使变量赋值的时序模块设计

```
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 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

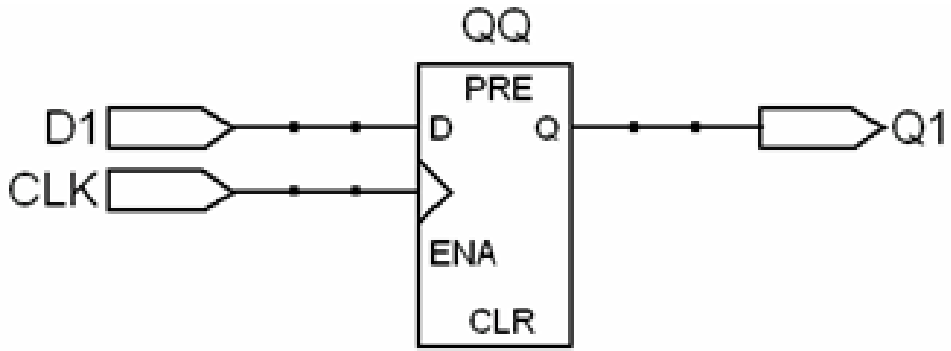


图 5-1 D 触发器电路

5.1 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

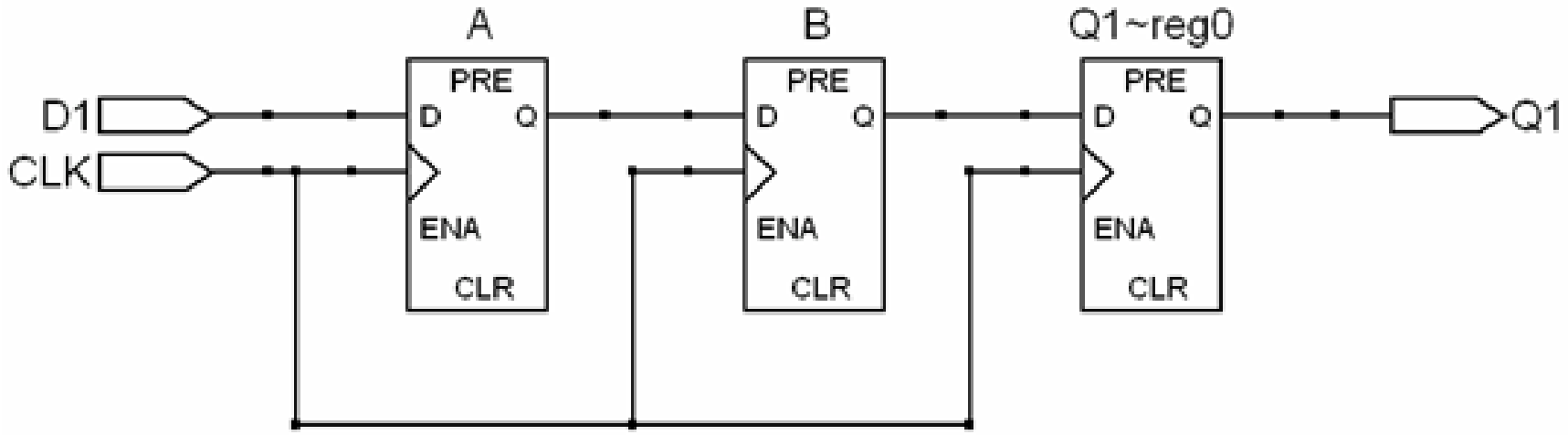


图 5-2 例 5-3 的 RTL 电路图

5.1 数 据 对 象

【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;
```

5.1 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

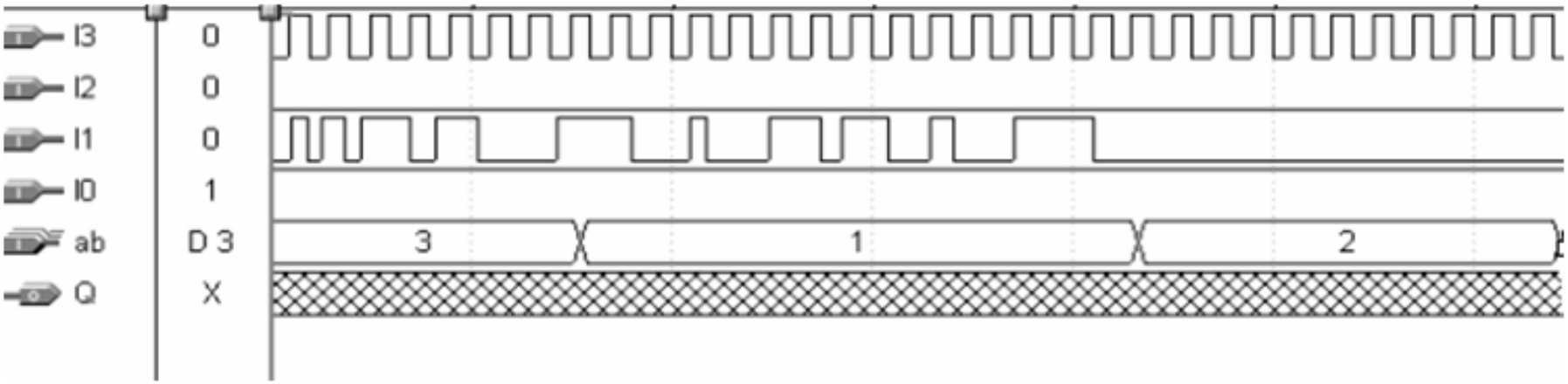


图 5-3 程序例 5-6 的错误工作时序

【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;
```

5.1 数 据 对 象

5.1.4 进程中的信号赋值与变量赋值

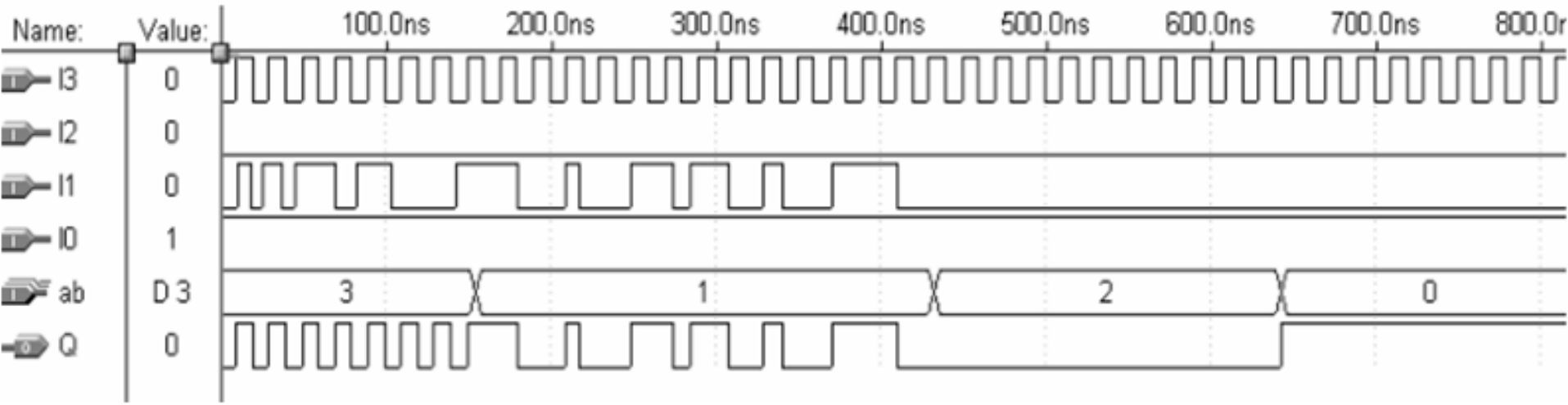


图 5-4 程序例 5-7 的正确工作时序

5.2 VHDL设计实例及其语法内涵

5.2.1 含同步并行预置功能的**8**位移位寄存器设计

【例 5-8】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SHFT IS
    PORT (  CLK, LOAD : IN STD_LOGIC;
                DIN  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                DOUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                  QB : OUT STD_LOGIC  );
END SHFT;
ARCHITECTURE behav OF SHFT IS
 SIGNAL REG8 : STD_LOGIC_VECTOR(7 DOWNTO 0);
    BEGIN
    PROCESS (CLK, LOAD)
    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;
     END PROCESS;
        QB <= REG8(0);     DOUT<=REG8;
END behav;
```

5.2 VHDL设计实例及其语法内涵

5.2.1 含同步并行预置功能的8位移位寄存器设计

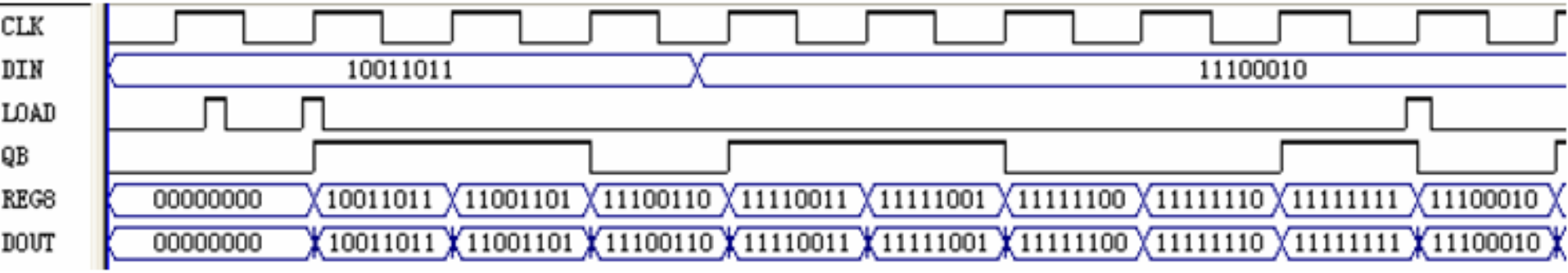


图 5-5　例 5-8 的工作时序

5.2 VHDL设计实例及其语法内涵

5.2.2 移位模式可控的8位移位寄存器设计

【例 5-9】

```
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)
```

接下页

5.2 VHDL设计实例及其语法内涵

接上页

```
    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;
```

5.2 VHDL设计实例及其语法内涵

5.2.2 移位模式可控的8位移位寄存器设计

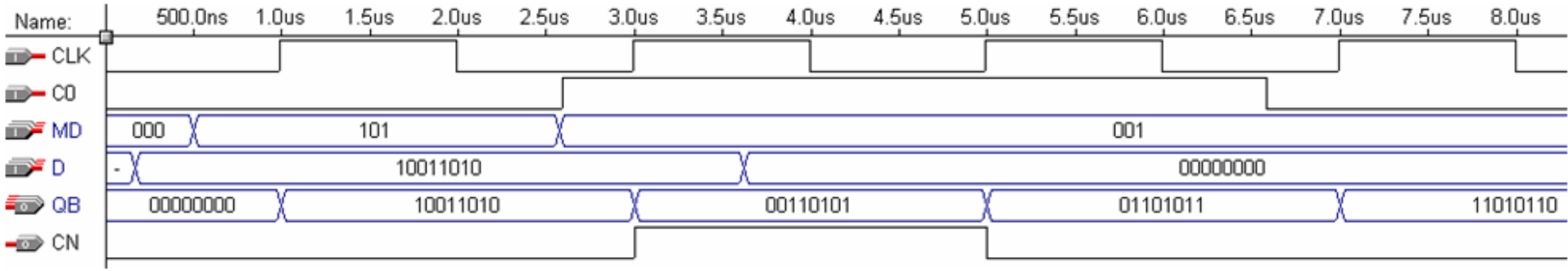


图 5-6 例 5-9 中带进位循环左移仿真波形(MD = “001”)

5.2.3 位矢中‘1’码个数统计电路设计

【例 5-10】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNTC IS
PORT (DIN  : IN STD_LOGIC_VECTOR(7 downto 0);
      CNTH : OUT STD_LOGIC_VECTOR(3 downto 0));
END CNTC;
ARCHITECTURE BHV OF CNTC IS
BEGIN
process(DIN)
VARIABLE Q : STD_LOGIC_VECTOR(3 downto 0);
begin
   Q := "0000";
   FOR n in 0 to 7 LOOP  --n是LOOP的循环变量
      IF (DIN(n)='1')  THEN  Q:=Q+1;   END IF;
   END LOOP;
   CNTH<=Q;
end process;
END BHV;
```

5.2 VHDL设计实例及其语法内涵

5.2.3 位矢中‘1’码个数统计电路设计

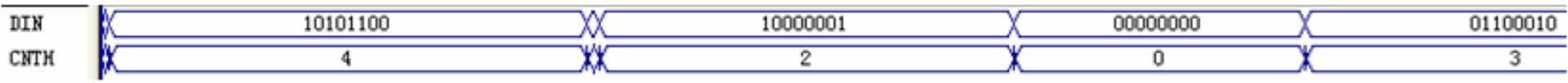


图 5-7 例 5-10 的仿真波形

5.2 VHDL设计实例及其语法内涵

5.2.3 位矢中'1'码个数统计电路设计

LOOP语句的常用表达方式有两种：

```
[ LOOP 标号: ] LOOP
    顺序语句
END LOOP [ LOOP 标号 ];
```

（1）单个LOOP语句

```
...
L2 : LOOP
      a := a+1;
     EXIT L2 WHEN a >10 ;              -- 当 a 大于 10 时跳出循环
END LOOP L2;
...
```

（2）FOR_LOOP语句

```
[LOOP 标号: ] FOR 循环变量, IN  循环次数范围  LOOP
        顺序语句
END LOOP [LOOP 标号];
```

5.2 VHDL设计实例及其语法内涵

5.2.3 位矢中‘1’码个数统计电路设计

【例 5-11】

```
SIGNAL a, b, c : STD_LOGIC_VECTOR (1 TO 3);
...
FOR n IN 1 To 3 LOOP
a(n) <= b(n) AND c(n);
END LOOP;
```

```
a(1)<=b(1) AND c(1);
a(2)<=b(2) AND c(2);
a(3)<=b(3) AND c(3);
```

5.2 VHDL设计实例及其语法内涵

5.2.4 三态门设计

【例 5-12】

```
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 VHDL设计实例及其语法内涵

5.2.4 三态门设计

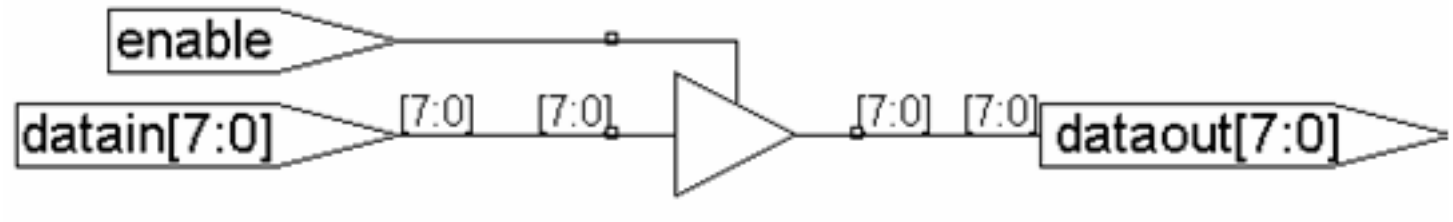


图 5-8　8 位三态控制门电路

5.2 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

【例 5-13】

```
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 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

【例 5-14】

（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 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

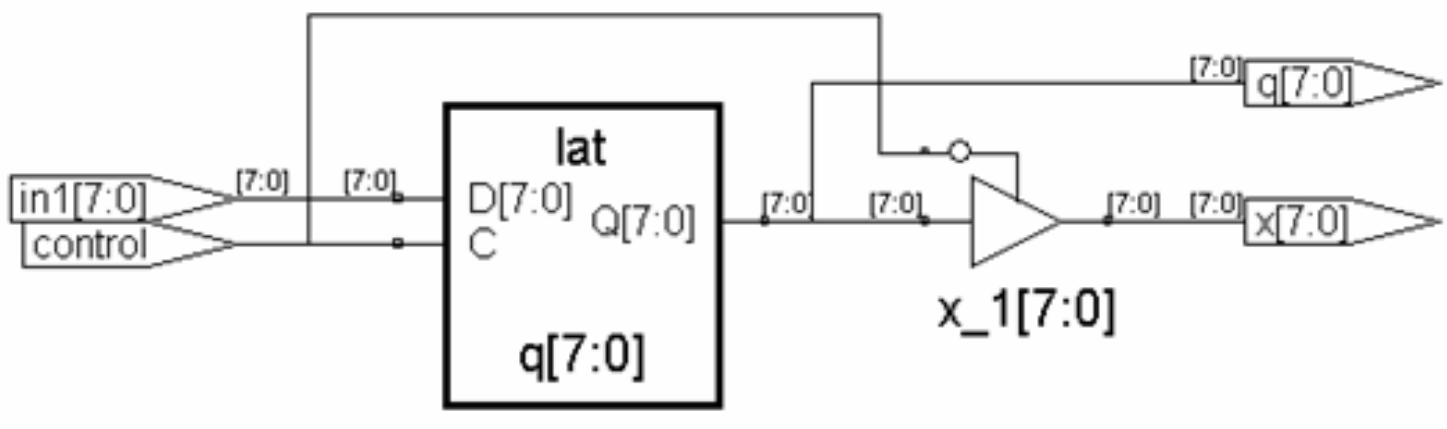


图 5-9 例 5-13 的综合结果

5.2 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

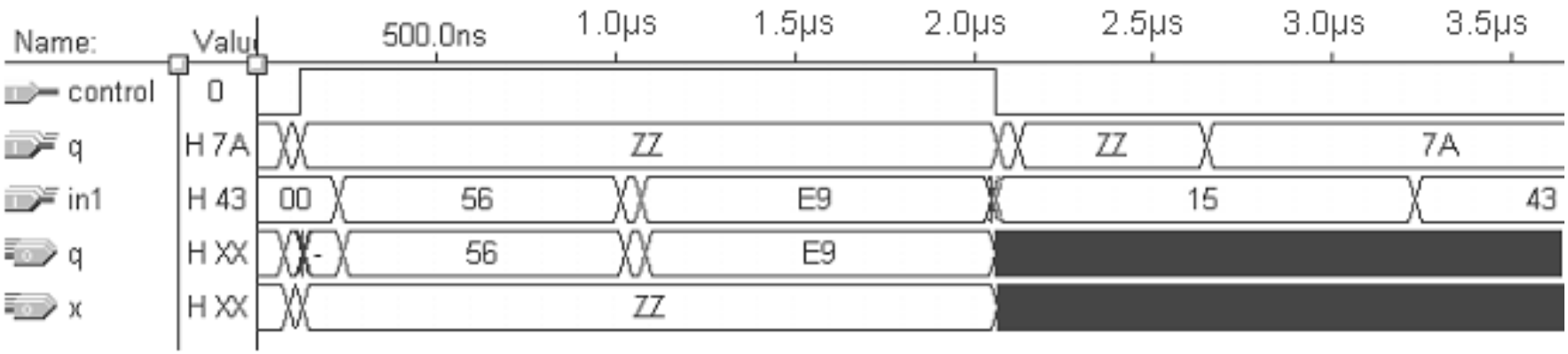


图 5-10 例 5-13 的仿真波形图

5.2 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

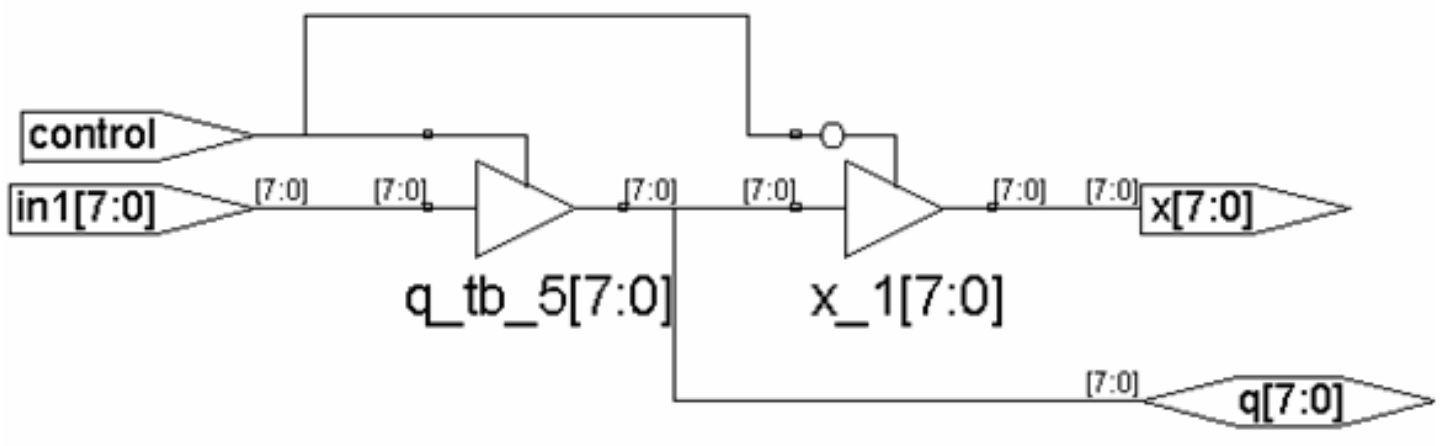


图 5-11 例 5-14 的综合结果

5.2 VHDL设计实例及其语法内涵

5.2.5 双向端口的设计方法

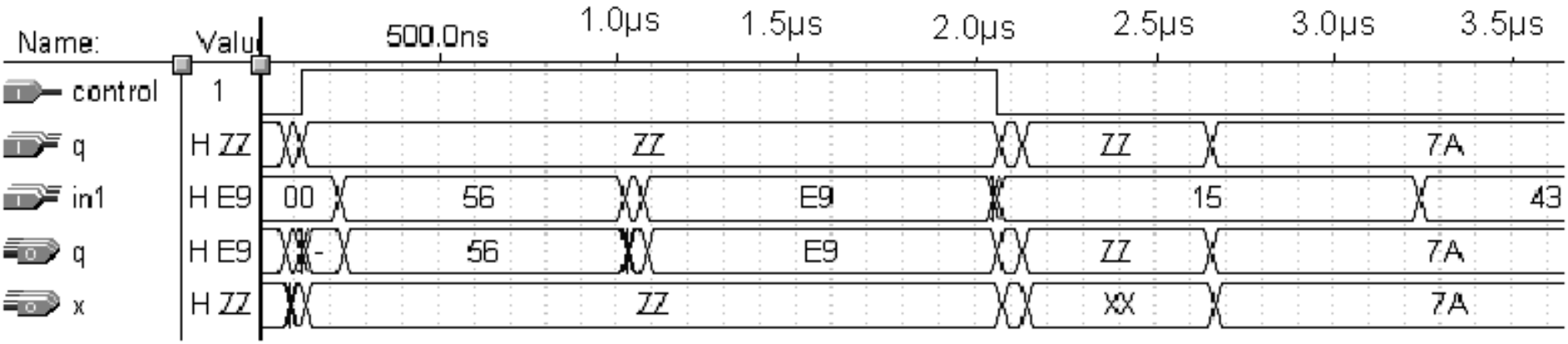


图 5-12 例 5-14 的仿真波形图

5.2.6 三态总线电路设计

【例 5-15】

```
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 VHDL设计实例及其语法内涵

5.2.6 三态总线电路设计

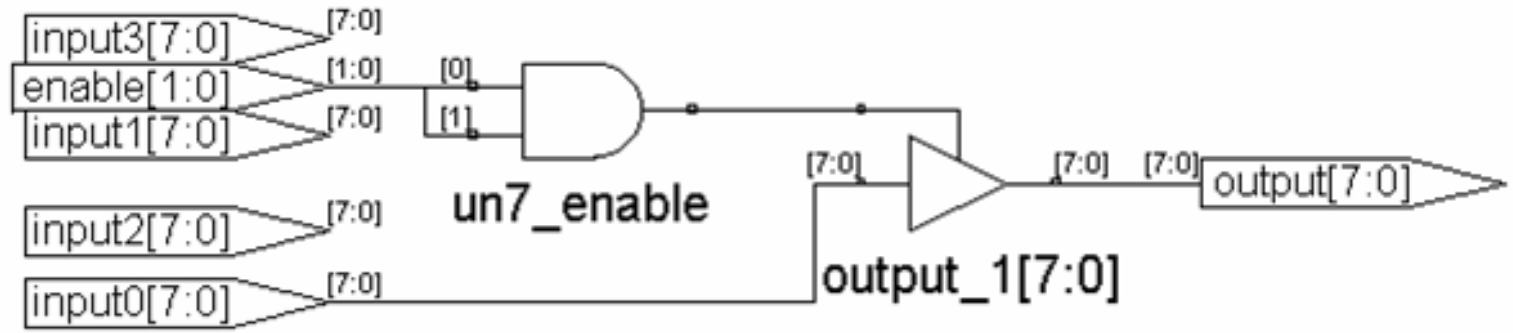


图5-13 例5-15错误的综合结果

5.2 VHDL设计实例及其语法内涵

5.2.6 三态总线电路设计

【例 5-16】

```
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;
```

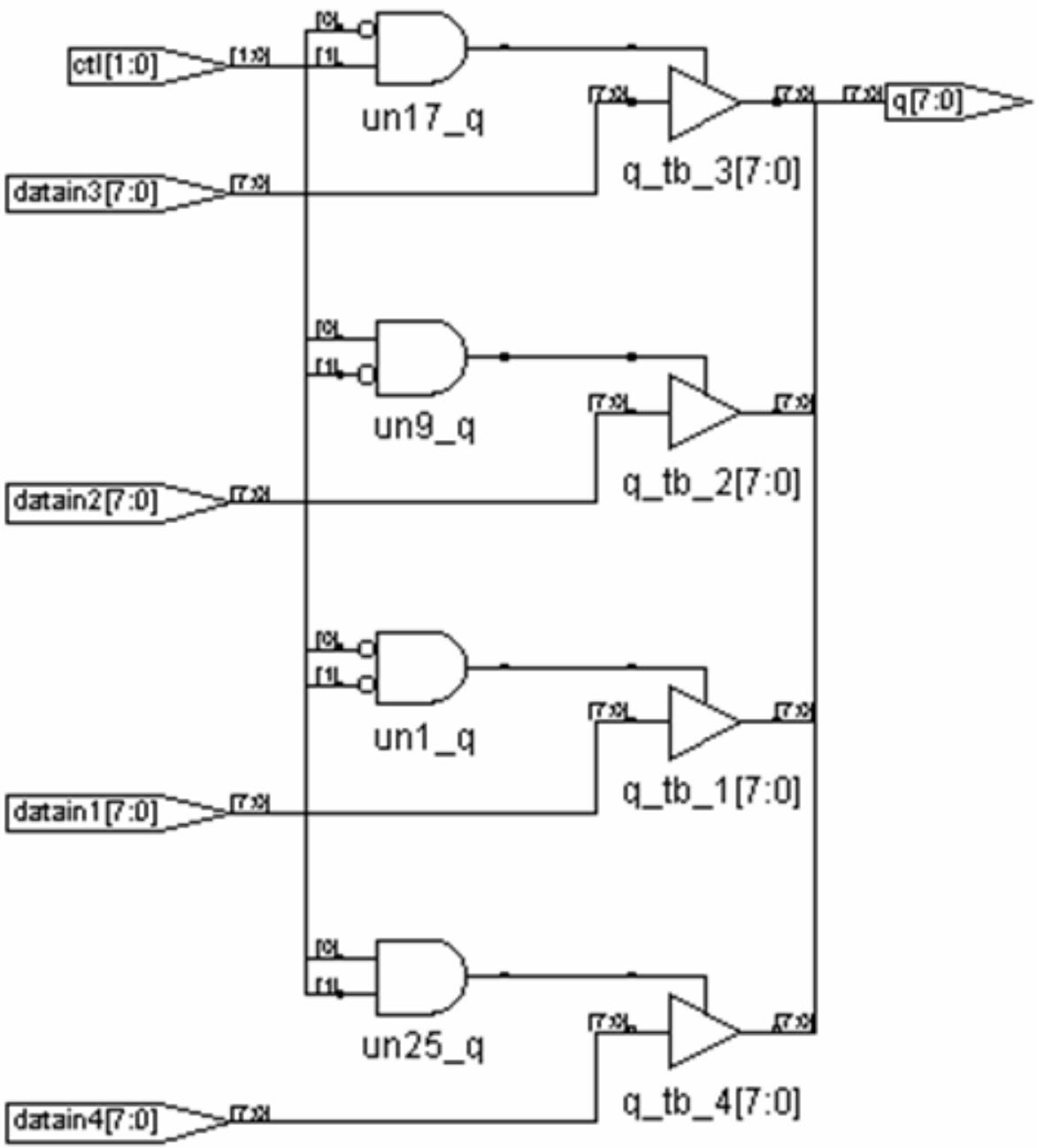


图 5-14 例 5-16 正确的综合结果

5.2 VHDL设计实例及其语法内涵

5.2.7 双边沿触发时序电路设计讨论

在同一进程中同一信号的双边沿操作

```
PROCESS (CLK)
BEGIN
    IF  RISING_EDGE(CLK)  THEN
 Q1 <= Q1 + 1 ;
    ELSIF        FALLING_EDGE(CLK)
THEN
 Q1 <= Q1 + 1 ;
    END IF;
END PROCESS ;
```

在不同进程中同一信号的双边沿操作

```
PROCESS (CLK)
 BEGIN
   IF  RISING_EDGE(CLK)  THEN
     Q1 <= Q1 + 1 ;   END IF;
END PROCESS ;
PROCESS (CLK)
 BEGIN
   IF  FALLING_EDGE(CLK)  THEN
     Q1 <= Q1 + 1 ;    END IF;
 END PROCESS ;
```

5.3 顺序语句归纳

5.3.1 进程语句格式

```
[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];
```

5.3 顺序语句归纳

5.3.2 进程结构组成

进程说明部分 { 定义一些局部量，可包括数据类型、常数、变量、属性、子程序等

顺序描述语句 {
- 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 —— 多数VHDL综合器要求敏感信号表必须列出本进程中所有输入信号名

5.3 顺序语句归纳

5.3.3 进程要点

1. PROCESS为一无限循环语句

2. 进程中的顺序语句具有明显的顺序和并行双重性

```
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;
```

5.3 顺序语句归纳

5.3.3 进程要点

3. 进程语句本身是并行语句

【例 5-17】

```
   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;
```

5.3 顺序语句归纳

5.3.3 进程要点

4. 信号可以是多个进程间的通信线

5. 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

5.4 并行赋值语句讨论

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

【例 5-18】

```
SIGNAL seiect :  INTEGER RANGE 15 DOWNTO 0;
. . .
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 ;
```

5.5 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

5.5 IF语句概述

【例 5-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY control_stmts IS
  PORT (a, b, c : IN BOOLEAN;   --a、b、c 被定义为 BOOLEAN 数据类型
        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;
```

【例 5-20】

```
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
  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;
```

5.5 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。

5.6 半整数与奇数分频电路设计

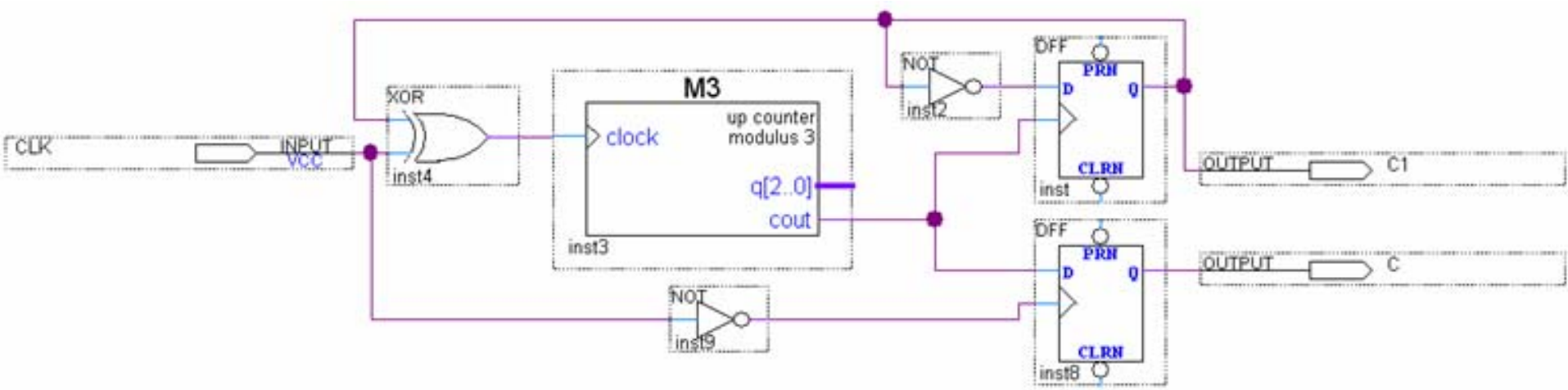


图 5-15 占空比为 50%的任意奇数次分频电路

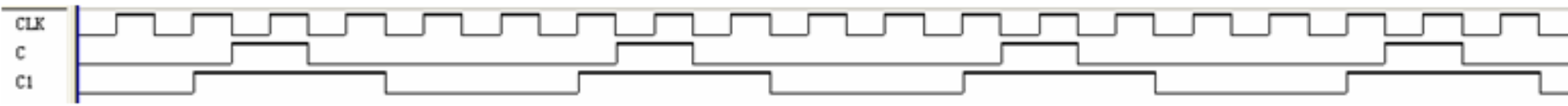


图 5-16 图 5-15 电路的仿真波形

5.6 半整数与奇数分频电路设计

【例 5-21】 占空比为 50%的任意奇数次 5 分频电路

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY CNT10 IS
PORT (CLK : IN STD_LOGIC ;
     K_OR,K1,K2 : OUT STD_LOGIC) ;
END ;
ARCHITECTURE bhv OF CNT10 IS
 SIGNAL C1,C2 : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
 SIGNAL M1,M2 : STD_LOGIC ;
 BEGIN
PROCESS(CLK,C1)
  BEGIN
  IF  RISING_EDGE(CLK) THEN
IF (C1="100") THEN   C1<="000";   ELSE C1<=C1+1; END IF;
IF (C1="001") THEN   M1<=NOT M1;  ELSIF (C1="011") THEN  M1<=NOT M1;
```

接下页

5.6 半整数与奇数分频电路设计

接上页

```
  END IF;    END IF;
END PROCESS;
PROCESS(CLK,C2)   BEGIN
IF  FALLING_EDGE(CLK) THEN
  IF (C2="100") THEN   C2<="000";   ELSE C2<=C2+1; END IF;
  IF (C2="001") THEN   M2<=NOT M2;  ELSIF (C2="011") THEN  M2<=NOT M2;
  END IF;  END IF;
END PROCESS;
  K1 <= M1;  K2 <= M2; K_OR <= M1 OR M2;
END bhv;
```

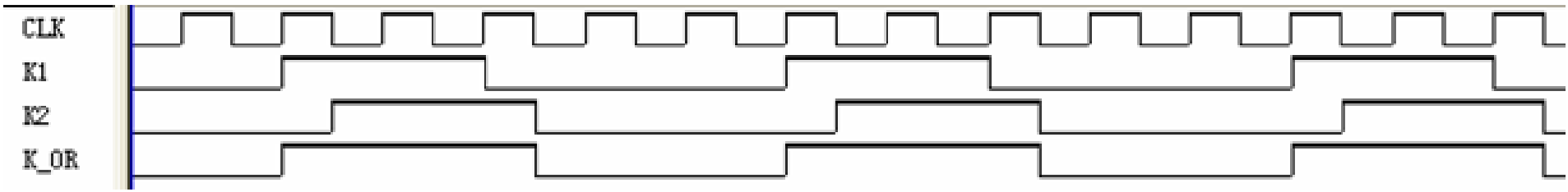


图 5-17 占空比为 50%的任意奇数次分频电路

5.6 半整数与奇数分频电路设计

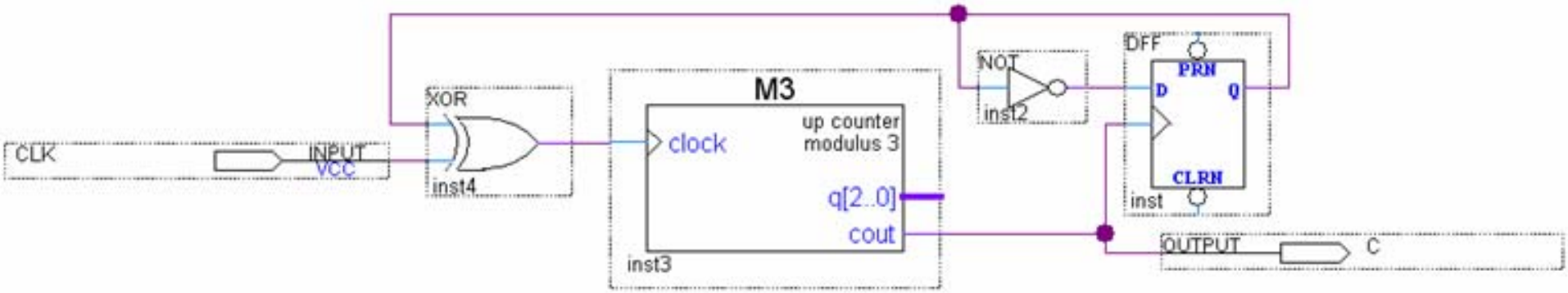


图 5-18 任意半整数分频电路

图 5-19 图 5-18 的电路仿真波形图

5.7 仿 真 延 时

5.7.1 固有延时

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

```
z <= x XOR y ;
```

```
B <= A AFTER 20ns ;--固有延时模型
```

5.7 仿真延时

5.7.2 传输延时

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

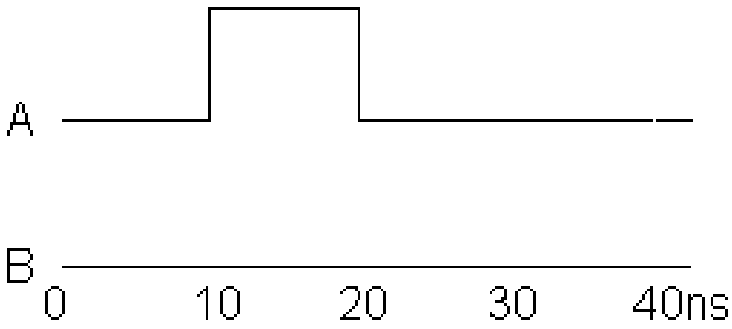


图 5-20 固有延时输入输出波形

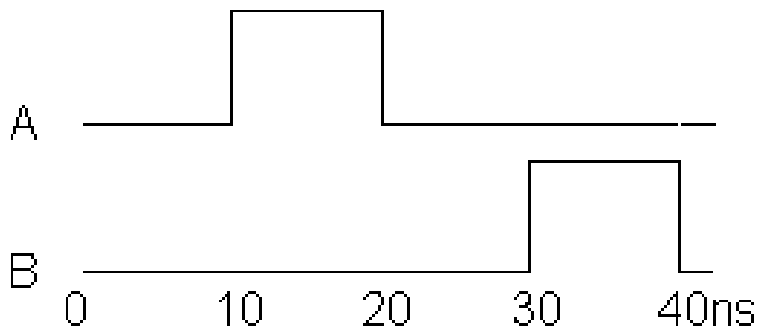


图 5-21 传输延时输入输出波形

5.7.3 仿真 δ

5.8 VHDL的RTL表述

5.8.1 行为描述

【例 5-22】

```
MODULE counter_up
    Clock ,reset,                PIN ;
    Counter7..counter0             PIN  ISTYPE  'COM' ;
    Cnt_ff7..cnt_ff0               NODE ISTYPE  'REG' ;
    Counter = [counter7..counter0];
    Cnt = [cnt_ff7..cnt_ff0];
   EQUATIONS
    Cnt.CLK = clock ;       Cnt.AR  = reset    ;
    Cnt :=  cnt.FB + 1 ;    Counter = cnt ;
   END counter_up
```

5.8 VHDL的RTL表述

5.8.1 行为描述

【例 5-23】

```
module DFF1(CLK,D,Q);
   output Q ;   input  CLK, D;  reg Q;
   always @( posedge CLK )
      Q <= D;
endmodule
```

```
ELSIF (clock ='1' AND clock'EVENT) THEN
```

5.8 VHDL的RTL表述

5.8.2 数据流描述

5.8.3 结构描述

结构描述建模步骤如下：

- 元件说明：描述局部接口。
- 元件例化：相对于其他元件放置元件。
- 元件配置：指定元件所用的设计实体。

习　题

5-1 什么是固有延时？什么是惯性延时？

5-2 δ是什么？在**VHDL**中，δ有什么用处？

5-3 哪些情况下需要用到程序包**STD_LOGIC_UNSIGNED**？试举一例。

5-4 说明信号和变量的功能特点，以及应用上的异同点。

5-5 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？

5-6 在**VHDL**设计中，给时序电路清零（复位）有两种不同方法，它们是什么，如何实现？

5-7 用循环语句设计一个**7**人投票表决器，及一个**4**位**4**输入最大数值检测电路。

5-8 从不完整的条件语句产生时序模块的原理看，例**5-7**和例**5-10**从表面上看都包含不完整条件语句，试说明，为什么它们的综合结果都是组合电路。

5-9 设计一个求补码的程序，输入数据是一个有符号的**8**位二进制数。

习　题

5-10 设计一个比较电路，当输入的**8421BCD**码大于**5**时输出**1**，否则输出**0**。

5-11 用原理图或**VHDL**输入方式分别设计一个周期性产生二进制序列**01001011001**的序列发生器，用移位寄存器或用同步时序电路实现，并用时序仿真器验证其功能。

5-12 基于原理图输入方式，用**74194**、**74273**、**D**触发器等器件组成**8**位串入并出的转换电路，要求在转换过程中数据不变，只有当**8**位一组数据全部转换结束后，输出才变化一次。

5-13 设计**8**位左移移位寄存器，给出时序仿真波形。

5-14 将例**5-15**中的四个**IF**语句分别用四个并列进程语句表达出来。

实验与设计

5-1 半整数与奇数分频器设计

（1）实验目的：

（2）实验内容1：

（3）实验内容2：

（4）实验内容3：

（5）实验内容4 ：

实验与设计

5-2 简易分频器设计

（1）实验目的：

（2）实验内容1：

（3）实验内容2：

（4）实验内容3：

5E+系统演示示例：/KX_7C5EE+/EXPERIMENTs/EXP30_FDIV。

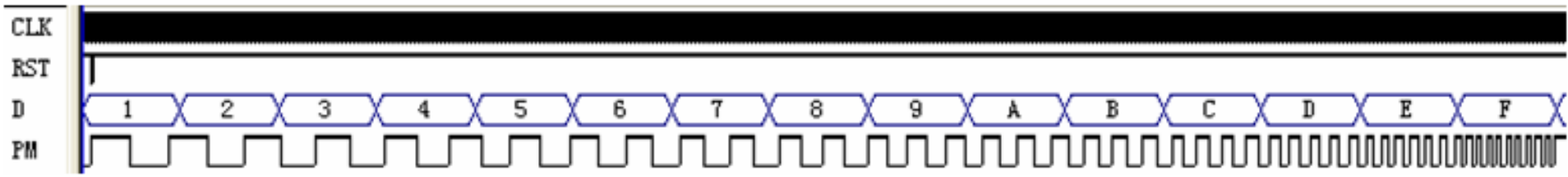


图 5-22 针对不同预置数，占空比均衡后的分频器输出

实验与设计

5-3 VGA彩条信号显示控制电路设计

（1）实验目的：

（2）实验原理：

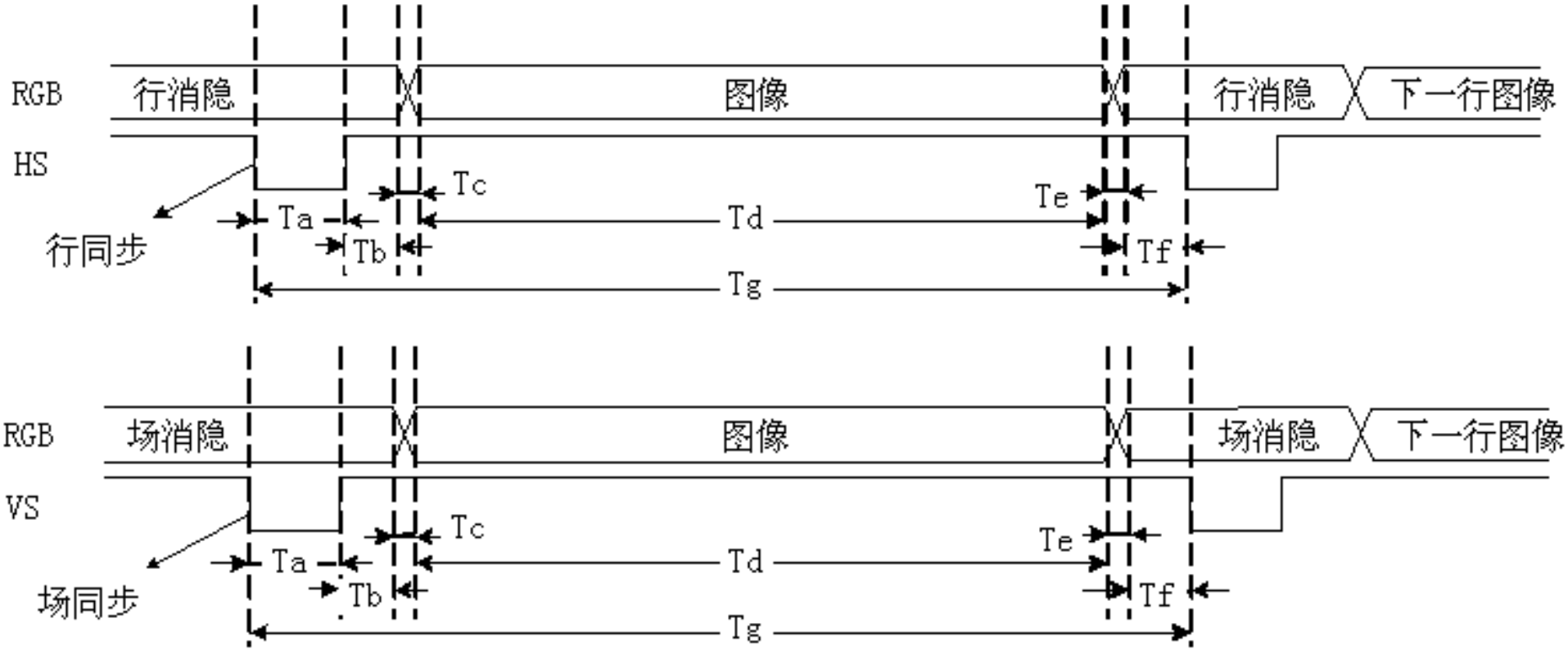


图 5-23 VGA 行扫描、场扫描时序示意图

实验与设计

表 5-3 行扫描时序要求：(单位：像素，即输出一个像素 Pixel 的时间间隔)

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间(Pixels)	8	96	40	8	640	8	800

表 5-4 场扫描时序要求：(单元：行，即输出一行 Line 的时间间隔)

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间(Lines)	2	2	25	8	480	8	525

实验与设计

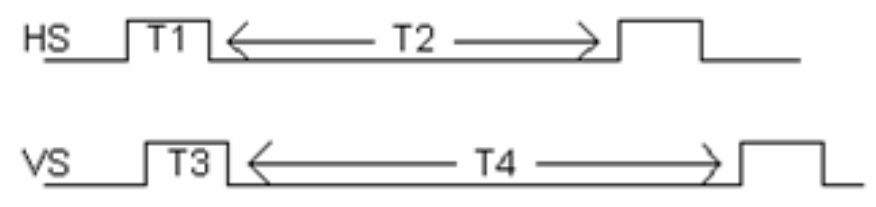
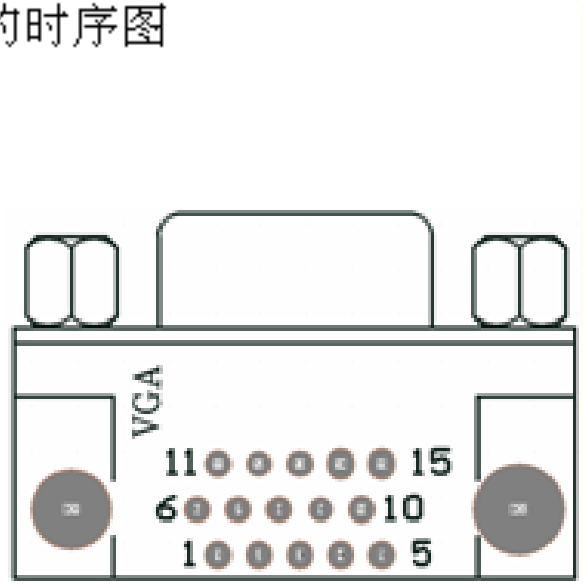


图 5-24 HS 和 VS 的时序图



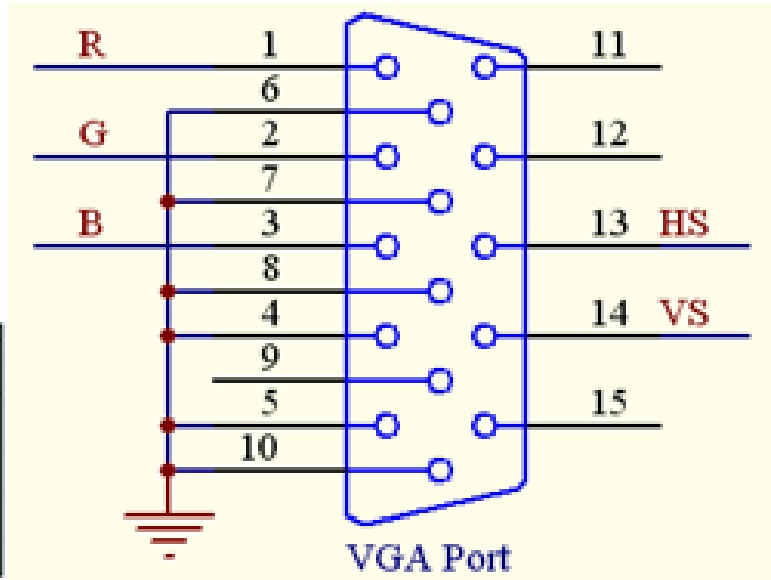


图 5-25 VGA 接口电路图，左接口从上往下看

【例 5-24】

```
LIBRARY IEEE;   -- VGA 显示器 彩条 发生器
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COLOR IS
    PORT (      CLK, MD : IN STD_LOGIC;
        HS, VS, R, G, B : OUT STD_LOGIC  ); -- 行场同步/红，绿，兰
END COLOR;
ARCHITECTURE behav OF COLOR IS
    SIGNAL HS1,VS1,FCLK,CCLK      : STD_LOGIC;
    SIGNAL MMD : STD_LOGIC_VECTOR(1 DOWNTO 0);-- 方式选择
    SIGNAL FS : STD_LOGIC_VECTOR (3 DOWNTO 0);
    SIGNAL CC : STD_LOGIC_VECTOR(4 DOWNTO 0); -- 行同步/横彩条生成
    SIGNAL LL : STD_LOGIC_VECTOR(8 DOWNTO 0); -- 场同步/竖彩条生成
    SIGNAL GRBX : STD_LOGIC_VECTOR(3 DOWNTO 1);-- X 横彩条
    SIGNAL GRBY : STD_LOGIC_VECTOR(3 DOWNTO 1);-- Y 竖彩条
    SIGNAL GRBP : STD_LOGIC_VECTOR(3 DOWNTO 1);
    SIGNAL GRB  : STD_LOGIC_VECTOR(3 DOWNTO 1);
```

接下页

接上页

```
BEGIN
    GRB(2) <= (GRBP(2) XOR MD) AND HS1 AND VS1;
    GRB(3) <= (GRBP(3) XOR MD) AND HS1 AND VS1;
    GRB(1) <= (GRBP(1) XOR MD) AND HS1 AND VS1;
    PROCESS( MD )    BEGIN
        IF MD'EVENT AND MD = '0' THEN
           IF MMD = "10" THEN  MMD <= "00";
            ELSE   MMD <= MMD + 1; END IF;        -- 三种模式
        END IF;
    END PROCESS;
    PROCESS( MMD )    BEGIN
        IF MMD = "00" THEN     GRBP <= GRBX;      -- 选择横彩条
        ELSIF MMD = "01" THEN  GRBP <= GRBY;      -- 选择竖彩条
        ELSIF MMD = "10" THEN  GRBP <= GRBX XOR GRBY; -- 产生棋盘格
         ELSE  GRBP <= "000";  END IF;
    END PROCESS;
```

接下页

接上页

```
PROCESS( CLK )    BEGIN
    IF CLK'EVENT AND CLK = '1' THEN -- 13MHz 13 分频
        IF FS = 13 THEN FS <= "0000";
        ELSE FS <= (FS + 1); END IF;
    END IF;
END PROCESS;
FCLK <= FS(3); CCLK <= CC(4);
PROCESS( FCLK )    BEGIN
    IF FCLK'EVENT AND FCLK = '1' THEN
        IF CC = 29 THEN  CC <= "00000";
        ELSE  CC <= CC + 1;   END IF;
    END IF;
END PROCESS;
PROCESS( CCLK )    BEGIN
    IF CCLK'EVENT AND CCLK = '0' THEN
        IF LL = 481 THEN  LL <= "000000000";
        ELSE  LL <= LL + 1; END IF;
    END IF;
```

接下页

接上页

```
END PROCESS;
PROCESS( CC,LL )    BEGIN
    IF CC > 23 THEN  HS1 <= '0';  --行同步
    ELSE HS1 <= '1';  END IF;
    IF LL > 479 THEN  VS1 <= '0'; --场同步
    ELSE  VS1 <= '1';   END IF;
END PROCESS;
PROCESS(CC, LL)      BEGIN
       IF CC < 3  THEN GRBX <= "111";  -- 横彩条
    ELSIF CC < 6  THEN GRBX <= "110";
    ELSIF CC < 9  THEN GRBX <= "101";
    ELSIF CC < 13 THEN GRBX <= "100";
    ELSIF CC < 15 THEN GRBX <= "011";
    ELSIF CC < 18 THEN GRBX <= "010";
    ELSIF CC < 21 THEN GRBX <= "001";
     ELSE GRBX <= "000";          END IF;
       IF LL <  60 THEN GRBY <= "111";  -- 竖彩条
```

接下页

实验与设计

接上页

```
        ELSIF LL < 130 THEN GRBY <= "110";
        ELSIF LL < 180 THEN GRBY <= "101";
        ELSIF LL < 240 THEN GRBY <= "100";
        ELSIF LL < 300 THEN GRBY <= "011";
        ELSIF LL < 360 THEN GRBY <= "010";
        ELSIF LL < 420 THEN GRBY <= "001";
        ELSE GRBY <= "000";          END IF;
    END PROCESS;
    HS<=HS1 ; VS<=VS1 ;R<=GRB(2) ;G<=GRB(3) ; B<=GRB(1);
END behav;
```

实验与设计

(3) 实验内容**1**：演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP11_VGA_COLOR_SQUR/**，和**/EXP11_VGA_COLOR_LINE/**。

(4) 实验内容**2**：

(5) 实验内容**3**：

(6) 实验内容**4**：

表 5-5 颜色编码：

颜色	黑	蓝	红	品	绿	青	黄	白
R	0	0	0	0	1	1	1	1
G	0	0	1	1	0	0	1	1
B	0	1	0	1	0	1	0	1

表 5-6 彩条信号发生器 3 种显示模式，

1	横彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
2	竖彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
3	棋盘格	1：棋盘格显示模式 1	2：棋盘格显示模式 2

实验与设计

5-4 基于时序电路的移位相加型8位硬件乘法器设计

（1）实验原理：

（2）实验任务1：

（3）实验任务2：

（4）实验任务4：

演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP32_MULTI8X8/MLTL8X8。**

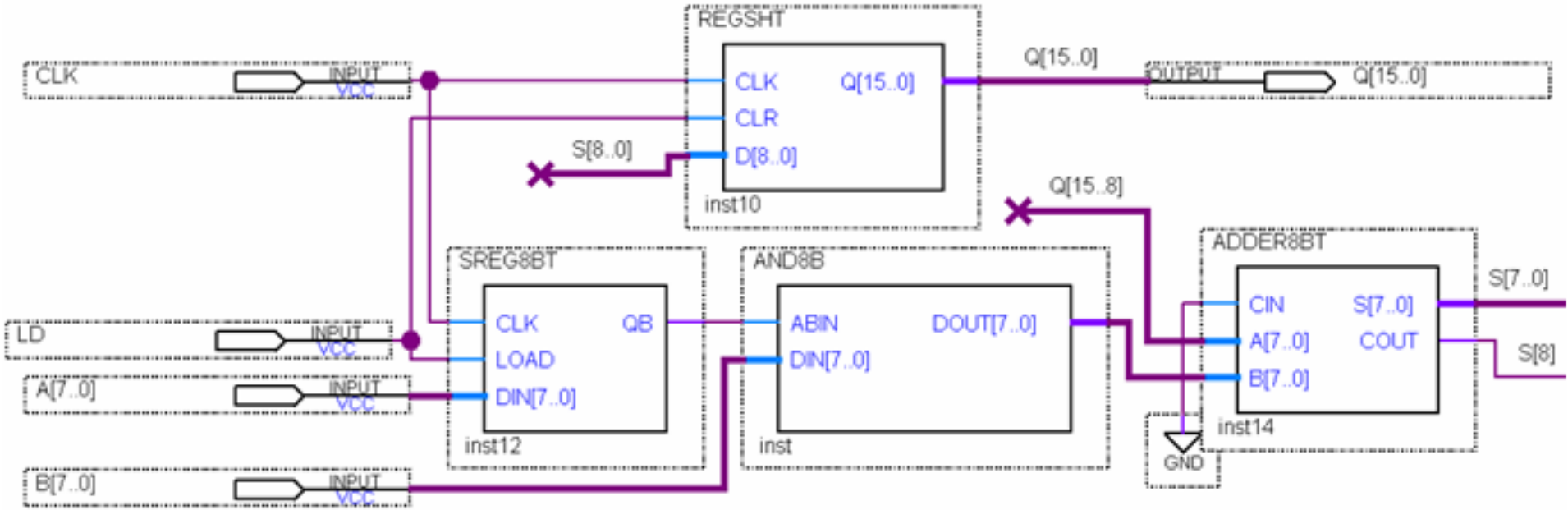


图 5-26　8 位乘法器逻辑原理图

【例 5-25】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REGSHT IS                                    -- 16 位锁存器
    PORT (  CLK，  CLR: IN STD_LOGIC;
            D : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
        Q : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)    );
END REGSHT;
ARCHITECTURE behav OF REGSHT IS
    SIGNAL R16S : STD_LOGIC_VECTOR(15 DOWNTO 0);
BEGIN
    PROCESS(CLK, CLR)    BEGIN
     IF CLR = '1' THEN      R16S <= "0000000000000000";
        ELSIF RISING_EDGE(CLK) THEN
            R16S(6 DOWNTO 0)<=R16S(7 DOWNTO 1); -- 右移低 8 位
            R16S(15 DOWNTO 7)<=D;   END IF;        -- 将输入锁到高 8 位
    END PROCESS;
    Q <= R16S;
END behav;
```

实验与设计

5-4 基于时序电路的移位相加型8位硬件乘法器设计

（1）实验原理：

（2）实验任务1：

（3）实验任务2：

（4）实验任务4：

演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP32_MULTI8X8/MLTL8X8**。

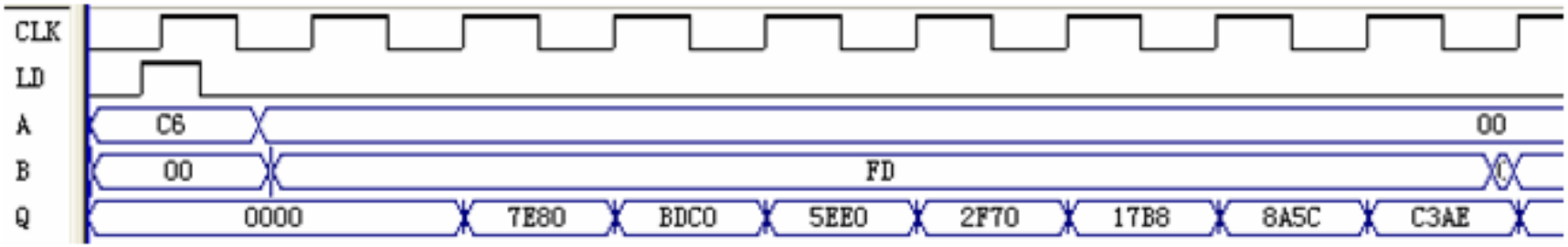


图 5-27　8 位移位相加乘法器运算逻辑波形图

实验与设计

5-5 移位寄存器设计

演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP39_SHIFTER/** 。

5-6 串/并转换数码静态显示控制电路设计

（1）实验原理：　**（2）**实验任务**1**：　**（3）**实验任务**2**：

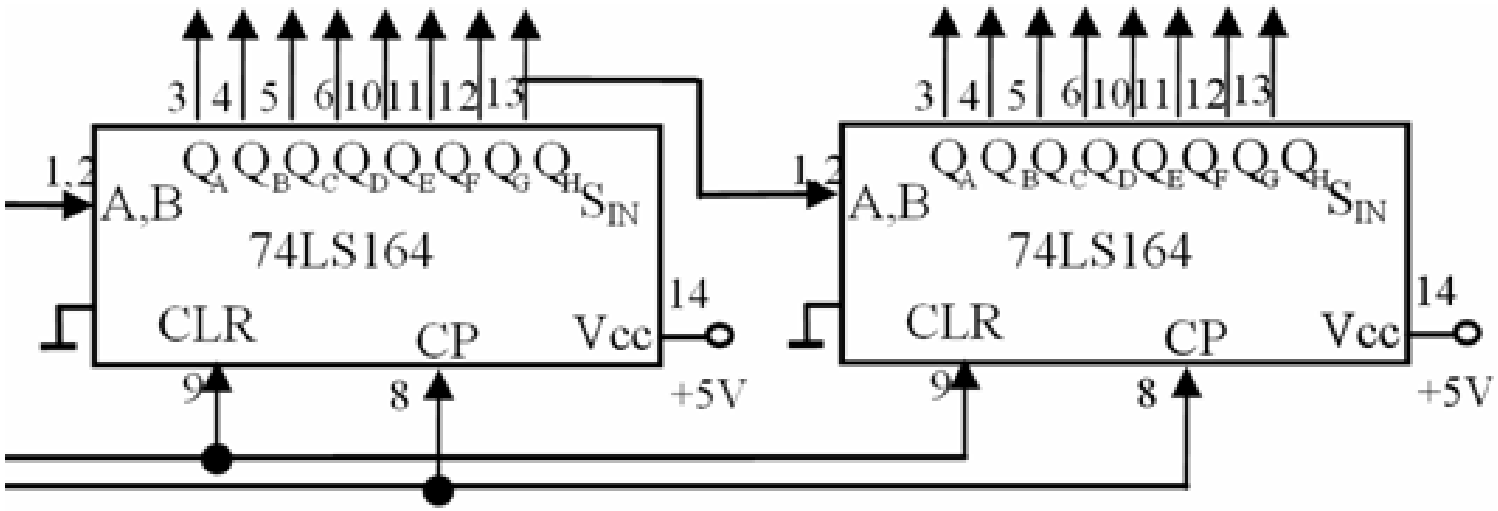


图 5-28　串/并转换数码管静态显示电路

实验与设计

5-7 并/串转换扩展输入口电路设计

实验任务：仅使用FPGA的2到3个I/O口，通过数个74LS165或4021扩展输入口。

此类电路在单片机开发中也常用，但是由于单片机本身的速度不高，再加上并/串转换，每一个通过转换后的输入口的实际速度已经非常低，适用范围较小。如无法用于读取较高速度ADC的数据等。如果用FPGA来控制，则仍能获得较高速度，从而提高了此类扩展电路的实用价值。

通常，74LS165的速度约25MHz左右，如果只扩展一片，对于FPGA的控制电路，可用足其上限频率，每一个位的速度可以达到3MHz。