第5章

并行语句

5.1 并行信号赋值语句

5.1.1 简单信号赋值语句

赋值目标 <= 表达式

【例 5-1】

```
LIBRARY IEEE ;
USE IEEE.STD LOGIC 1164.ALL ;
ENTITY  h adder  IS
    PORT (A, B  : IN   STD LOGIC;  SO,CO : OUT  STD LOGIC );
END ENTITY  h adder ;
ARCHITECTURE  fh1 OF h adder IS
    BEGIN                        --XOR 是异或逻辑操作符，AND 是与逻辑操作符
     SO <= A XOR B ;    CO <= A AND B ;
END ARCHITECTURE fh1;
```

赋值目标 <= v_0, v_1 AFTER T_1, v_2 AFTER T_2,..., v_n AFTER T_n ;

```
reset<= '1', '0' after reset_period ;
```

```
DATA <= "01", "10" AFTER 40 ns, "11" AFTER 60 ns, "00" AFTER 65 ns;
```

5.1 并行信号赋值语句

5.1.2 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 ;
```

【例 5-2】

```
ENTITY  mux IS
  PORT(a,b,c : IN BIT;   p1,p2 : IN BIT ;  z  : OUT BIT );
END;
ARCHITECTURE behv OF mux IS
 BEGIN
   z  <= a WHEN p1 = '1' ELSE
         b WHEN p2 = '1' ELSE
         c  ;
END;
```

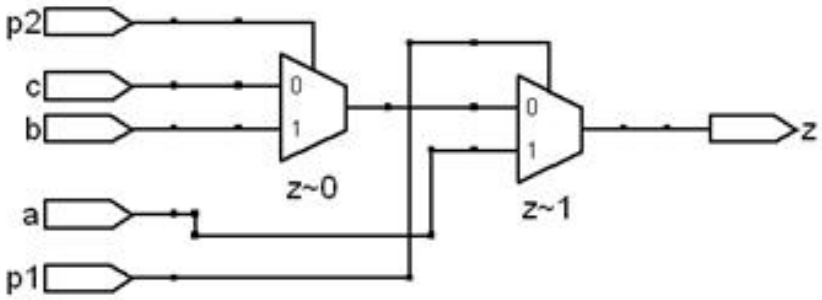

图5-1　例5-2的RTL电路图

5.1 并行信号赋值语句

5.1.3 选择信号赋值语句

```
WITH 选择表达式 SELECT
    赋值目标信号 <= 表达式 WHEN 选择值
                    表达式 WHEN 选择值
                    ...
                    表达式 WHEN 选择值;
```

```
WITH selt SELECT
muxout <= a   WHEN   0|1  ,     -- 0或1
          b   WHEN  2 TO 5 ,     -- 2或3，或4或5
          c   WHEN    6    ,
          d   WHEN    7    ,
        'Z'  WHEN OTHERS ;
```

5.1 并行信号赋值语句

5.1.4 块语句

```
块标号  : BLOCK [(块保护表达式)]
        接口说明
            类属说明
            BEGIN
        并行语句
END BLOCK 块标号 ;
```

5.1 并行信号赋值语句

5.1.5 元件例化语句

```
COMPONENT 元件名 IS
   GENERIC (类属表);                              -- 元件定义语句
   PORT  (端口名表);
END COMPONENT 元件名;
... -其它语句
例化名 : 元件名 PORT MAP(                         -- 元件例化语句
         [端口名 =>] 连接端口名, ...) ;
```

5.1 并行信号赋值语句

5.1.6 例化语句应用示例

【例 5-3】

```
LIBRARY  IEEE ;
 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 5-4】

```
LIBRARY  IEEE;   --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                 --调用或门元件声明语句
     PORT ( a, b : IN STD_LOGIC;   c : OUT STD_LOGIC);
  END COMPONENT;
SIGNAL net1,net2,net3  :  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=>net2,SO=>net1);
  u2 : h_adder PORT MAP(net1, cin, net3, sum);
  u3 :   or2a   PORT MAP(a=>net2,  b=>net3,   c=>cout);
END ARCHITECTURE fd1;
```

```
COMPONENT h_adder
    PORT (  c, d :  IN STD_LOGIC;  e, f :  OUT STD_LOGIC) ;
```

```
  PORT (A, B : IN STD_LOGIC;  CO,SO :  OUT  STD_LOGIC);
```

5.1 并行信号赋值语句

5.1.7 生成语句

```
[标号：] FOR 循环变量 IN 取值范围 GENERATE
      说明
    BEGIN
    并行语句
        END GENERATE [标号]；
```

```
[标号：] IF 条件 GENERATE
      说明
    Begin
    并行语句
        END GENERATE [标号]；
```

```
表达式   TO   表达式 ；          -- 递增方式，如 1 TO 5
表达式 DOWNTO 表达式 ；          -- 递减方式，如 5 DOWNTO 1
```

5.1 并行信号赋值语句

5.1.7 生成语句

【例 5-5】

```
COMPONENT comp
PORT (x : IN STD_LOGIC;
      y : OUT STD_LOGIC );
END COMPONENT ;
SIGNAL a :STD_LOGIC_VECTOR(0 TO 7);
SIGNAL b :STD_LOGIC_VECTOR(0 TO 7);
...
gen : FOR i IN a'RANGE GENERATE
 u1: comp PORT MA (x=>a(i), y=>b(i));
END GENERATE gen;
```

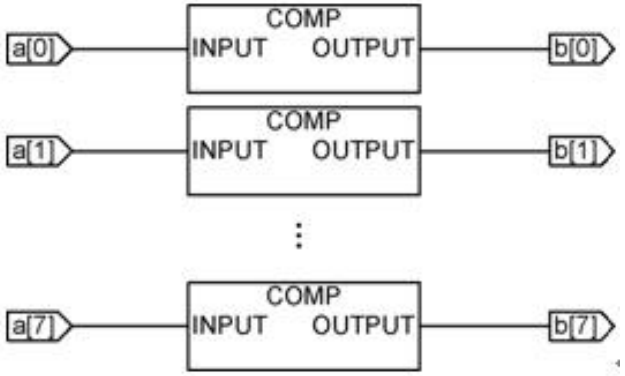

图 5-2 生成语句产生的 8 个相同的电路模块

5.1 并行信号赋值语句

5.1.7 生成语句

【例 5-6】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY d_ff IS
PORT ( d, clk_s : IN STD_LOGIC; q, nq : OUT STD_LOGIC);
END ENTITY d_ff;
ARCHITECTURE a_rs_ff OF d_ff IS
BEGIN
    PROCESS(CLK_S)   BEGIN
      IF clk_s='1' AND clk_s'EVENT  THEN q<=d; nq<=NOT d;  END IF;
    END PROCESS;
END ARCHITECTURE a_rs_ff;
```

5.1 并行信号赋值语句

5.1.7 生成语句

【例 5-7】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY cnt_bin_n is
GENERIC (n : INTEGER := 6);
PORT(q : OUT STD_LOGIC_VECTOR (0 TO n-1);  in_1 : IN  STD_LOGIC);
END ENTITY cnt_bin_n;
ARCHITECTURE behv OF cnt_bin_n IS
COMPONENT d_ff
  PORT(d, clk_s :  IN STD_LOGIC;  Q, NQ : OUT STD_LOGIC);
END COMPONENT d_ff;
SIGNAL s : STD_LOGIC_VECTOR(0 TO n);
BEGIN
  s(0) <= in_1;
   q_1 : FOR i IN 0 TO n-1 GENERATE
   dff : d_ff PORT MAP (s(i+1), s(I), q(i), s(i+1));
  END GENERATE;
END ARCHITECTURE behv;
```

5.1 并行信号赋值语句

5.1.7 生成语句

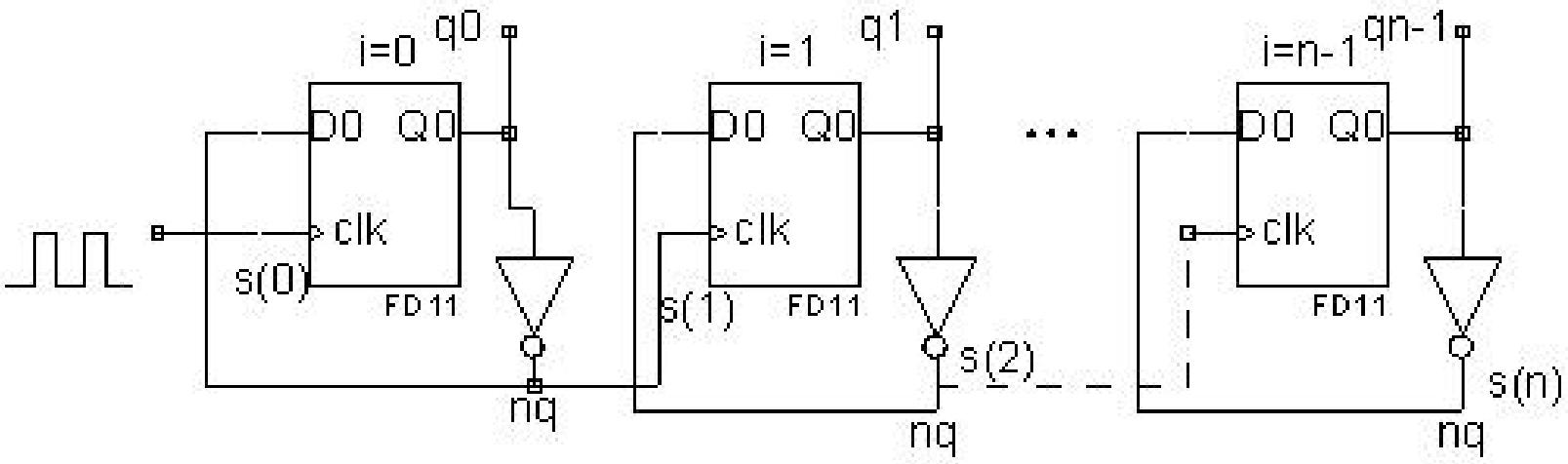

图 5-3　6 位二进制计数器原理图

5.1.8 GENERIC参数传递映射语句及其使用方法

【例 5-8】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
ENTITY MULT4B IS
GENERIC (   S  : INTEGER := 4);  --定义参数 S 为整数类型，且等于 4。
   PORT (   R  : OUT STD_LOGIC_VECTOR(2*S DOWNTO 1);
        A, B  : IN STD_LOGIC_VECTOR (S DOWNTO 1));
END ENTITY MULT4B;
ARCHITECTURE ONE OF MULT4B IS
   SIGNAL A0 :  STD_LOGIC_VECTOR(2*S DOWNTO 1);
BEGIN
   A0 <= CONV_STD_LOGIC_VECTOR(0,S) & A;--若 S=4，则此句是："0000"&A
 PROCESS (A, B)
  VARIABLE R1 : STD_LOGIC_VECTOR(2*S DOWNTO 1);
   BEGIN
      R1 := (others => '0');    --若 S=4，则此句等效于 R1:="00000000"
      FOR i IN 1 TO S LOOP
         IF (B(i) = '1') THEN
            R1 := R1 + TO_STDLOGICVECTOR(TO_BITVECTOR(A0) SLL (i-1));
          END IF;
      END LOOP;
      R <= R1;
   END PROCESS;
END ARCHITECTURE  ONE;
```

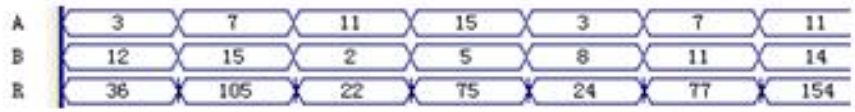

图 5-4 例 5-8 乘法器的仿真波形

5.1 并行信号赋值语句

5.1.8 GENERIC参数传递映射语句及其使用方法

例化名 ：元件名 GENERIC MAP(类属表)

【例 5-9】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MULT8B IS
   PORT(D1,D2 : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
              Q  : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
END;
   ARCHITECTURE BHV OF MULT8B IS
     COMPONENT MULT4B              --MULT4B 模块的调用声明
      GENERIC ( S  : integer);--照抄 MULT4B 实体中关于参数“端口”定义的语句
           PORT ( R  : OUT std_logic_vector(2*S DOWNTO 1);
                A, B  : IN std_logic_vector(S DOWNTO 1));
     END COMPONENT ;
  BEGIN
     u1: MULT4B GENERIC MAP(S=>8)
                PORT MAP (R =>Q, A=>D1, B=>D2);
END;
```

5.1 并行信号赋值语句

5.1.9 数据类型定义语句

1. 限定性数组型数据类型定义

```
TYPE 数组名 IS ARRAY(数组范围)OF 基本数据类型 ;

TYPE stb IS ARRAY (7 DOWNTO 0)of STD_LOGIC ;

TYPE TD IS ARRAY (7 DOWNTO 0, 3 TO 0)of STD_LOGIC ;
TYPE MATRIX IS ARRAY (127 DOWNTO 0)of STD_LOGIC_VECTOR(7 DOWNTO 0) ;
```

5.1 并行信号赋值语句

5.1.9 数据类型定义语句

2. 非限定性数组型数据类型定义

```
TYPE 数组名 IS ARRAY (数组下标名 RANGE <>) OF 数据类型 ;
```

```
Type bit is ('0','1');
Type bit_vector is array(natural rang<>) of bit;
```

5.1 并行信号赋值语句

5.1.9 数据类型定义语句

3. 枚举型数据类型定义

```
TYPE BOOLEAN  IS (FALSE,TRUE) ;
```

```
TYPE my_logic IS  ( '1' ,'Z' ,'U' ,'0' ) ;
SIGNAL s1 : my_logic ;
s1 <=  'Z' ;
```

```
TYPE  数据类型名  IS  数据类型定义表述
```

```
TYPE week IS (sun,mon,tue,wed,thu,fri,sat) ;
```

```
TYPE x is (low, high);
TYPE data_bus IS ARRAY (0 TO 7, x)of BIT ;
```

```
TYPE m_state IS  ( st0,st1,st2,st3,st4,st5 ) ;
SIGNAL present_state,next_state  :  m_state  ;
```

5.1 并行信号赋值语句

5.1.9 数据类型定义语句

4. 枚举型子类型数据类型定义

```
SUBTYPE  子类型名 IS 基本数据类型 RANGE 约束范围;

SUBTYPE digits IS INTEGER RANGE 0 to 9 ;
```

5.1.10 VHDL的存储器描述

【例 5-10】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ; --此程序包包含转换函数 CONV_INTEGER(A)
USE IEEE.STD_LOGIC_UNSIGNED.ALL ; --此程序包包含算符重载函数
ENTITY RAM78 IS
PORT (CLK,WREN : IN STD_LOGIC ; --定义时钟和写允许控制
        A : IN STD_LOGIC_VECTOR(6 DOWNTO 0) ; --定义 RAM 的 7 位地址输入端口
      DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0) ; --定义 RAM 的 8 位数据输入端口
       Q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)); --定义 RAM 的 8 位数据输出端口
END ;
ARCHITECTURE bhv OF RAM78 IS
TYPE G_ARRAY IS ARRAY(0 TO 127) OF STD_LOGIC_VECTOR(7 DOWNTO 0) ;
SIGNAL MEM : G_ARRAY;  --定义信号 MEM 的数据类型是用户新定义的类型 G_ARRAY
BEGIN
  PROCESS (CLK)       BEGIN
     IF  RISING_EDGE(CLK)  THEN
--下句：如果时钟有上升沿出现，且写使能为高电平，则 RAM 数据口的数据被写入指定地址单元
        IF  WREN='1' THEN   MEM(CONV_INTEGER(A))<= DIN;  END IF;
        END IF;
 IF (FALLING_EDGE(CLK)) THEN  Q<=MEM(CONV_INTEGER(A));--读出存储器中的数据
      END IF;
   END PROCESS ;
END BHV;
```

5.1 并行信号赋值语句

5.1.10 VHDL的存储器描述

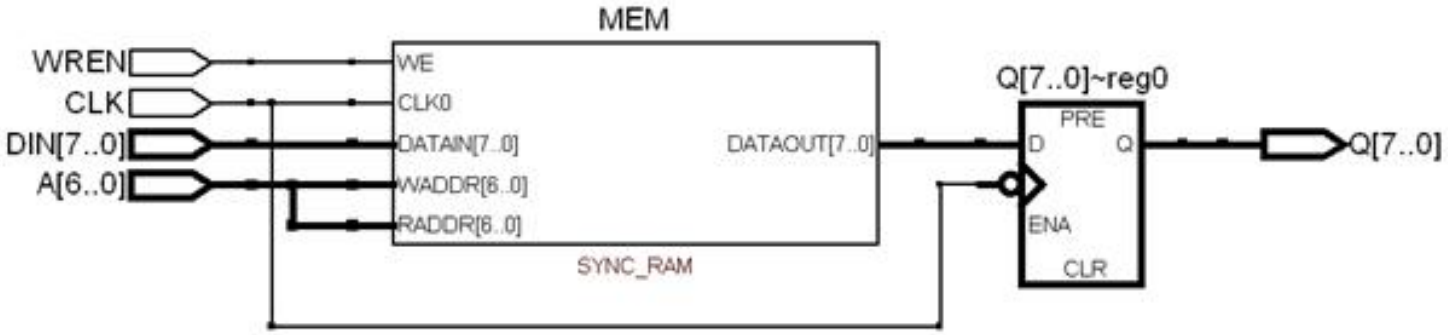

图 5-5　例 5-10 的 RAM78 的 RTL 图

5.1 并行信号赋值语句

5.1.11 信号属性及属性函数

属性测试项目名'属性标识符

1. 信号类属性

```
NOT(clock'STABLE AND clock ='1')
(clock'EVENT AND clock ='1')
```

2. 数据区间类属性

```
SIGNAL range1 : IN STD_LOGIC_VECTOR (0 TO 7) ;
...
FOR i IN range1'RANGE LOOP
...
```

5.1 并行信号赋值语句

5.1.11 信号属性及属性函数

3. 数值类属性

```
PROCESS (clock, a, b);
TYPE obj IS ARRAY (0 TO 15) OF BIT ;
SIGNAL ele1, ele2, ele3, ele4    : INTEGER ;
BEGIN
 ele1 <= obj'RIGHT ; --测得数据类型 obj 的最右侧位是第 15 位
 ele2 <= obj'LEFT ;  --测得数据类型 obj 的最左侧位是第 0 位
 ele3 <= obj'HIGH ;  --测得数据类型 obj 的最高位是第 15 位
 ele4 <= obj'LOW  ;  --测得数据类型 obj 的最低位是第 0 位
...
```

5.1 并行信号赋值语句

【例 5-11】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY parity IS
   GENERIC (bus_size : INTEGER := 8 );
   PORT (input_bus : IN STD_LOGIC_VECTOR(bus_size-1 DOWNTO 0);
         even_numbits, odd_numbits : OUT STD_LOGIC ) ;
END parity ;
ARCHITECTURE behave OF parity IS
BEGIN
PROCESS (input_bus)
   VARIABLE temp: STD_LOGIC;
BEGIN
   temp := '0';
   FOR i IN input_bus'LOW TO input_bus'HIGH LOOP
temp := temp XOR input_bus(i) ;
   END LOOP ;
   odd_numbits <= temp ;    even_numbits <= NOT temp;
END PROCESS;
END behave;
```

5.1 并行信号赋值语句

5.1.11 信号属性及属性函数

4. 数组类属性'LENGTH

```
TYPE arry1 ARRAY (0 TO 7) OF BIT ;
VARIABLE wth1 : INTEGER;
...
wth1: =arry1'LENGTH; -- wth1 = 8
```

5. 用户定义属性

```
ATTRIBUTE 属性名 : 数据类型;
ATTRIBUTE 属性名 OF 对象名 : 对象类型 IS 值;
```

```
LIBRARY synplify;
USE synplicity.attributes.all;
```

5.2 VHDL运算操作符

5.2.1 逻辑操作符

表 5-1 VHDL 操作符列表

类　　型	操　作　符	功　　能	操作数数据类型
算术操作符	+	加	整数
	−	减	整数
	&	并置	一维数组
	*	乘	整数和实数（包括浮点数）
	/	除	整数和实数（包括浮点数）
	MOD	取模	整数
	REM	取余	整数
	SLL	逻辑左移	BIT 或布尔型一维数组
	SRL	逻辑右移	BIT 或布尔型一维数组
	SLA	算术左移	BIT 或布尔型一维数组
	SRA	算术右移	BIT 或布尔型一维数组
	ROL	逻辑循环左移	BIT 或布尔型一维数组
	ROR	逻辑循环右移	BIT 或布尔型一维数组
	**	乘方	整数
	ABS	取绝对值	整数

5.2 VHDL运算操作符

5.2.1 逻辑操作符

续表

类　　型	操　作　符	功　　能	操作数数据类型
关系操作符	=	等于	任何数据类型
	/=	不等于	任何数据类型
	<	小于	枚举与整数类型，及对应的一维数组
	>	大于	枚举与整数类型，及对应的一维数组
	<=	小于等于	枚举与整数类型，及对应的一维数组
	>=	大于等于	枚举与整数类型，及对应的一维数组
逻辑操作符	AND	与	BIT，BOOLEAN，STD_LOGIC
	OR	或	BIT，BOOLEAN，STD_LOGIC
	NAND	与非	BIT，BOOLEAN，STD_LOGIC
	NOR	或非	BIT，BOOLEAN，STD_LOGIC
	XOR	异或	BIT，BOOLEAN，STD_LOGIC
	XNOR	异或非	BIT，BOOLEAN，STD_LOGIC
	NOT	非	BIT，BOOLEAN，STD_LOGIC
符号操作符	+	正	整数
	–	负	整数

5.2 VHDL运算操作符

5.2.1 逻辑操作符

表 5-2 VHDL 操作符优先级

运 算 符	优 先 级
NOT，ABS，**	最高优先级
*，/，MOD，REM	
+（正号），－（负号）	
+，－，&	
SLL，SLA，SRL，SRA，ROL，ROR	
=，/=，<，<=，>，>=	
AND，OR，NAND，NOR，XOR，XNOR	最低优先级

5.2 VHDL运算操作符

5.2.1 逻辑操作符

【例 5-12】

```
SIGNAL a , b, c : STD_LOGIC_VECTOR (3 DOWNTO 0);
SIGNAL d, e, f, g : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL h, I, j, k : STD_LOGIC ;
SIGNAL l, m, n, o, p : BOOLEAN ;
...
a<=b AND c;     --b、c 相与后向 a 赋值，a、b、c 的数据类型同属 4 位长的位矢量
d<=e OR f OR g ;              -- 两个操作符 OR 相同，不需括号
h<=(i NAND j)NAND k ;         -- NAND 不属上述三种算符中的一种，必须加括号
l<=(m XOR n)AND(o XOR p); -- 操作符不同，必须加括号
h<=i AND j OR k ;             -- 两个操作符不同，未加括号，表达错误
a<=b AND e ;                  -- 操作数 b 与 e 的位矢长度不一致，表达错误
h<=i OR l ;  --i 的数据类型是 STD_LOGIC，而 l 的数据类型是布尔量，表达错误
```

5.2 VHDL运算操作符

5.2.2 关系操作符

“ = ”	(等于)
“/=”	(不等于)
“ >”	(大于)
“< ”	(小于)
“>=”	(大于等于)
“<=”	(小于等于)

```
'1' = '1';    "101" = "101";    "1" > "011";    "101" < "110";
```

5.2 VHDL运算操作符

5.2.3 算术操作符

表 5-3 算术操作符分类表

	类　　别	算术操作符分类
1	求和操作符（Adding Operator）	+（加），－（减），&（并置）
2	求积操作符（Multiplying Operator）	* ， / ，MOD ，REM
3	符号操作符（Sign Operator）	+（正），－（负）
4	混合操作符（Miscellaneous Operator）	** ，ABS
5	移位操作符（Shift Operator）	SLL，SRL，SLA，SRA，ROL，ROR

5.2 VHDL运算操作符

1. 求和操作符

【例 5-13】

```
PACKAGE example_arithmetic IS
   TYPE small_INt IS RANGE 0 TO 7 ;
END example_arithmetic ;
USE WORK.example_arithmetic.ALL ;
ENTITY arithmetic IS
   PORT (a, b : IN SMALL_INT ;   c : OUT SMALL_INT);
END arithmetic ;
ARCHITECTURE example OF arithmetic IS
BEGIN
   c <= a + b ;
END example ;
```

5.2 VHDL运算操作符

1. 求和操作符

【例 5-14】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;--此程序包中包含算术操作符的重载函数
ENTITY ADDER8B  IS
    PORT (A, B  :   IN  STD_LOGIC_VECTOR(7 DOWNTO 0) ;
          CIN :   IN  STD_LOGIC;      COUT :  OUT  STD_LOGIC;
          DOUT :  OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) );
END ENTITY  ADDER8B ;
ARCHITECTURE  BHV OF  ADDER8B  IS
SIGNAL DATA :  STD_LOGIC_VECTOR(8 DOWNTO 0) ;
    BEGIN
 DATA <= ('0'& A) + ('0'& B) + ("00000000"&CIN);
 COUT <= DATA(8);     DOUT <= DATA(7 DOWNTO 0) ;
END ARCHITECTURE BHV;
```

5.2 VHDL运算操作符

1. 求和操作符

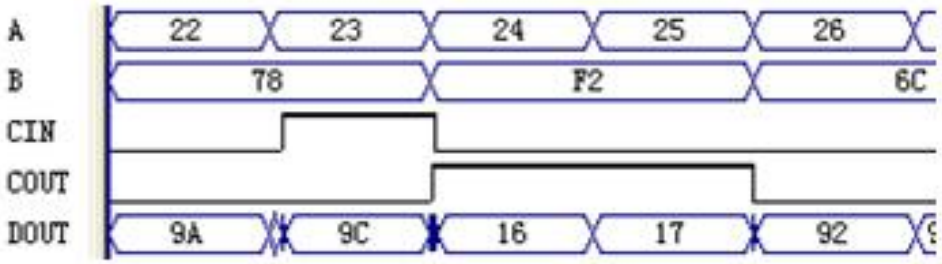

图 5-6　8 位加法器仿真波形

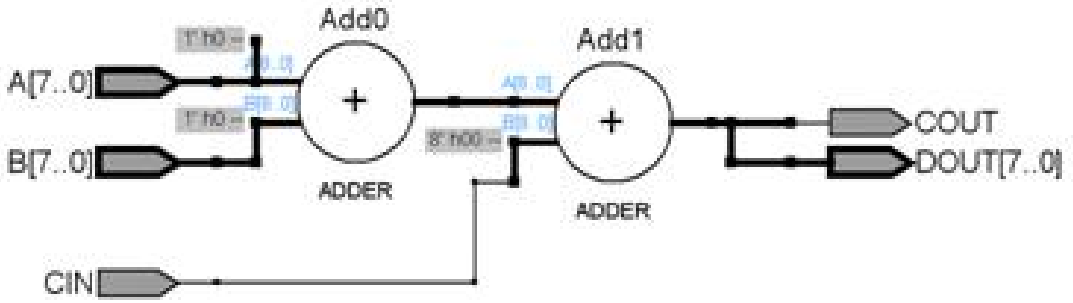

图 5-7　8 位加法器 Quartus II 综合之 RTL 电路

5.2 VHDL运算操作符

2. 求积操作符

*　　(乘)

/　　(除)

MOD (取模)

RED　(取余)

3. 符号操作符

4. 混合操作符

5.2 VHDL运算操作符

5. 移位操作符

【例 5-15】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY DC3to8 IS
    port (   DIN  : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
             DOUT : OUT BIT_VECTOR (7 DOWNTO 0));
END DC3to8;
ARCHITECTURE behave OF DC3to8 IS
BEGIN
DOUT <=  "00000001" SLL CONV_INTEGER(DIN);   --被移位部分是常数!
END behave;
```

5.2 VHDL运算操作符

5.2.4 省略赋值操作符

```
SIGNAL    d1  : STD_LOGIC_VECTOR(4 DOWNTO 0);
VARIABLE  a1  : STD_LOGIC_VECTOR(15 DOWNTO 0);
...
d1 <= (OTHERS=>'1');   a1 := (OTHERS=>'0') ;
```

```
d1 <= (1=>e(3),3=>e(5), OTHERS=>e(1) );
```

```
d1 <= e(1) & e(5) & e(1) & e(3) & e(1) ;
```

【例 5-16 】

```
... --其它部分同例 5-15
   PROCESS (DIN)      BEGIN
      DOUT<=(OTHERS =>'0');   DOUT(CONV_INTEGER(DIN))<='1';
   END PROCESS;
END behave;
```

5.3 keep属性应用

【例 5-17】

```
ARCHITECTURE fd1 OF f_adder IS
  . . .
 SIGNAL net1,net2,net3  :  STD_LOGIC;
    attribute keep : boolean;  --由于“true”的类型是布尔类型 boolean
    attribute keep  of net1 : signal is true;
BEGIN
```

5.3 keep属性应用

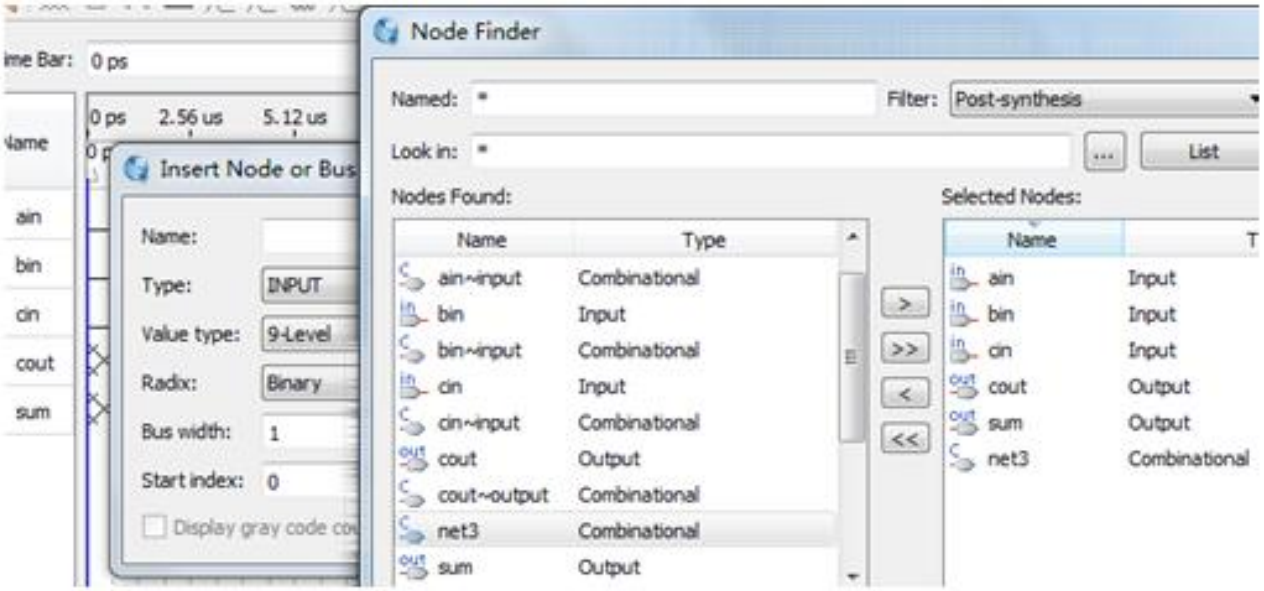

图 5-8 向仿真激励信号波形编辑窗调入信号 net1

5.3 keep属性应用

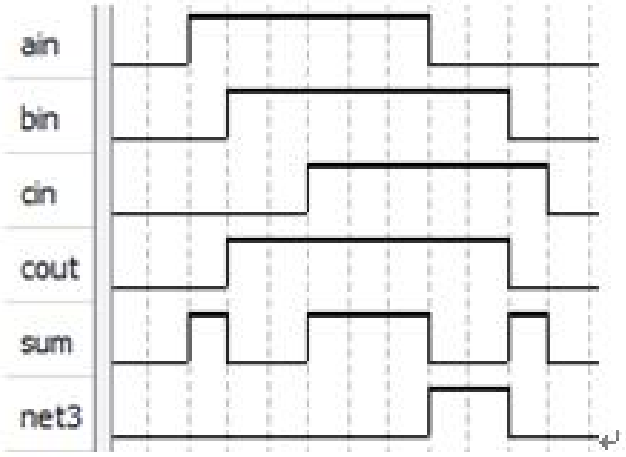

图 5-9 例 5-4 生成的仿真波形

5.4 SignalProbe使用方法

1. 按常规流程完成设计仿真和硬件测试

2. 设置SignalProbe Pins

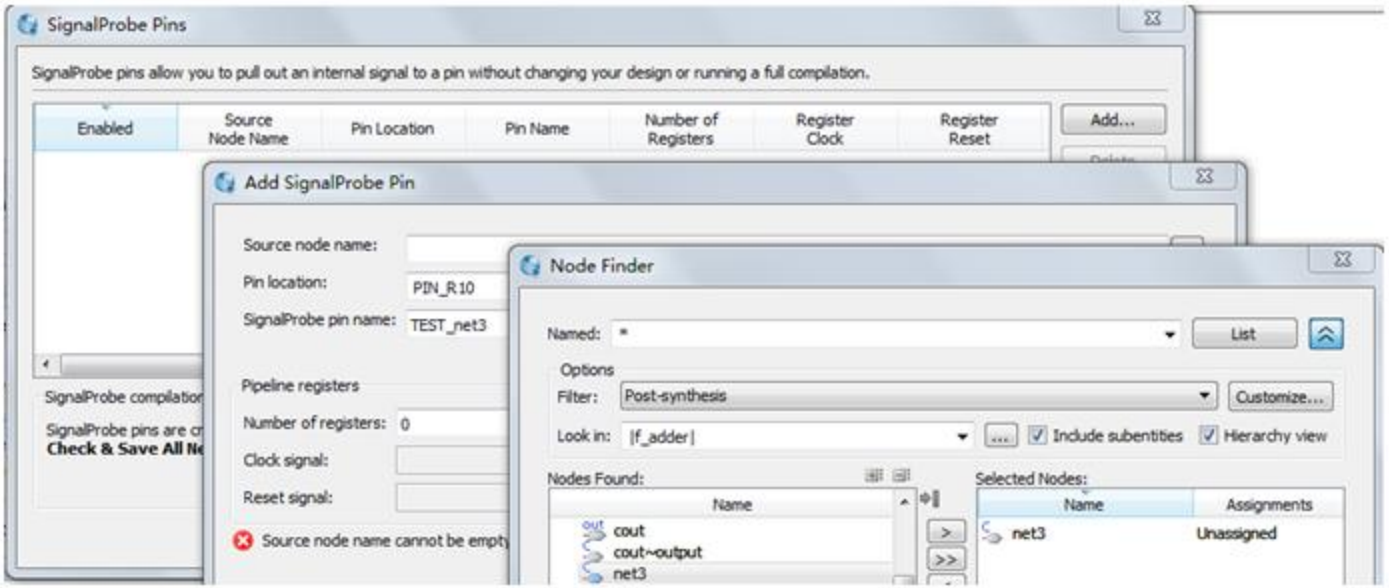

图 5-10　在 SignalProbe 对话框设置探测信号 net3

5.4 SignalProbe使用方法

2. 设置SignalProbe Pins

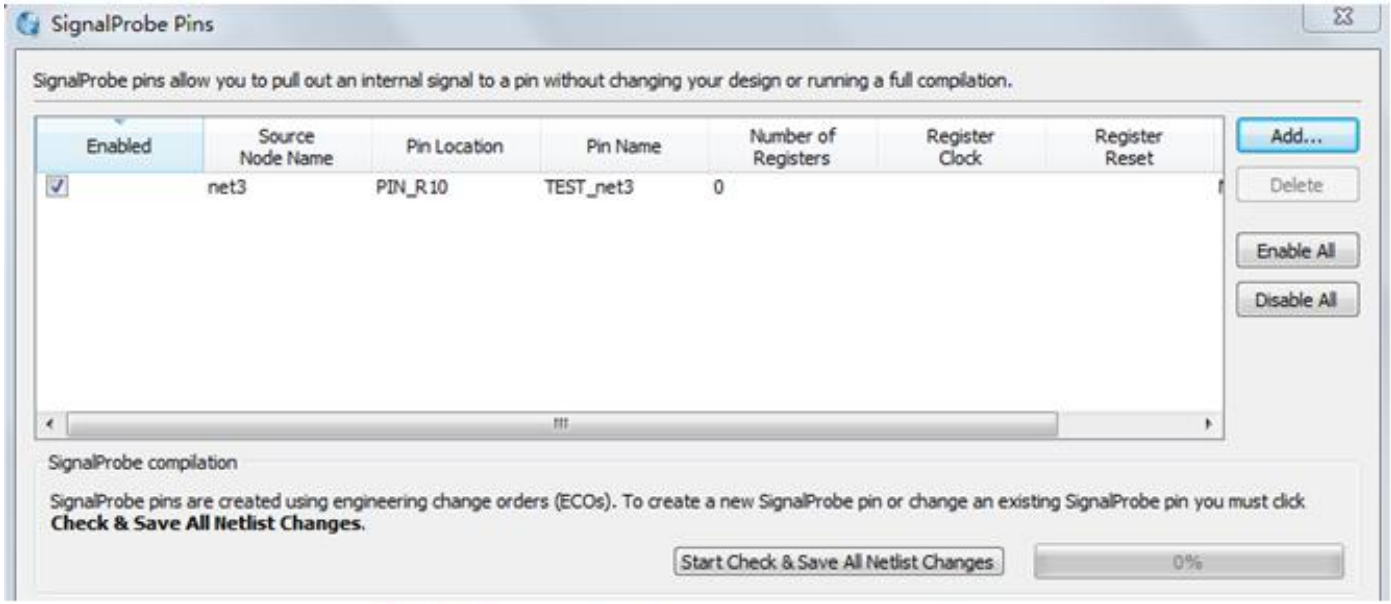

图 5-11 SignalProbe Pins 对话框设置情况

3. 编译SignalProbe Pins测试信息并下载测试

5.4 SignalProbe使用方法

1. 按常规流程完成设计仿真和硬件测试

2. 设置SignalProbe Pins

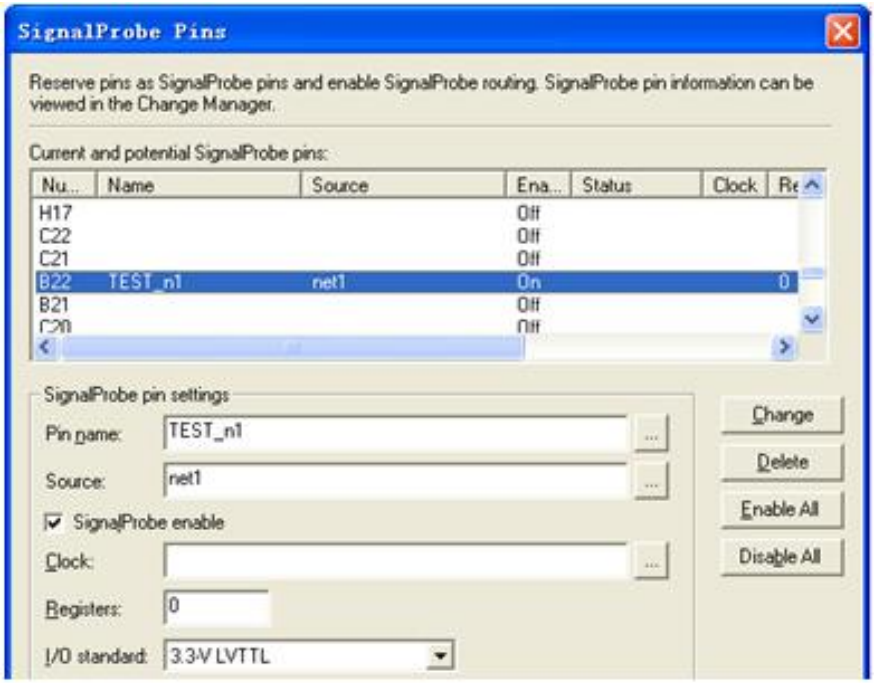

图 5-10 在 SignalProbe 对话框设置探测信号 net1

3. 编译SignalProbe Pins测试信息并下载测试

图 5-11 ECO 文件编译成功

习题

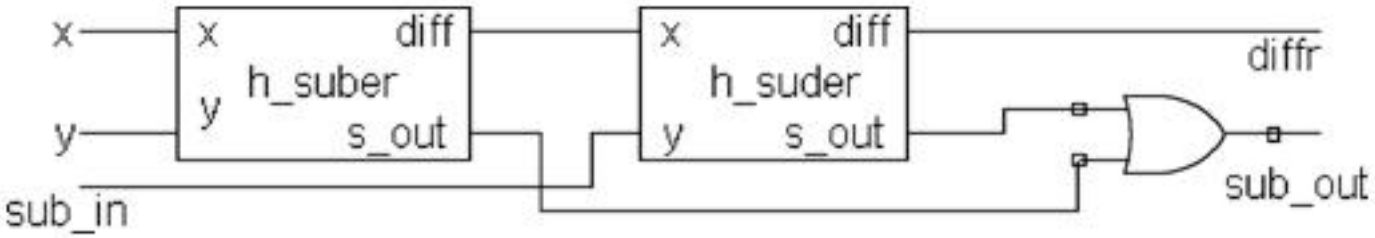
x
y
sub_in
x
y
h_suber
diff
s_out
x
y
h_suder
diff
s_out
diffr
sub_out

图 5-12　全减器

实验与设计

实验5-1 8位加法器设计实验

实验5-2 高速硬件除法器设计

【例5-18】

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DIV16 is
port(CLK : in std_logic;  A,B : in std_logic_vector(15 downto 0);
   QU,RE  : out std_logic_vector(15 downto 0));
end DIV16;
architecture rtl of DIV16 is
begin
process(CLK)
variable AT, BT,P,Q: std_logic_vector(15 downto 0);
begin
if  rising_edge(CLK)  then   AT:=A;  BT:=B;
   P:="0000000000000000";   Q:="0000000000000000";
for i in  QU'range  loop
p := P(14 downto 0) & AT(15); AT:=AT(14 downto 0)&'0'; P:=P-BT;
if P(15)='1' then Q(i):='0'; P:=P+BT ; else Q(i):='1';  end if;
end loop;   end if ;
QU <= Q;  RE <= P ;
end process;
end rtl ;
```

实验与设计

实验5-3 移位相加型8位硬件乘法器设计

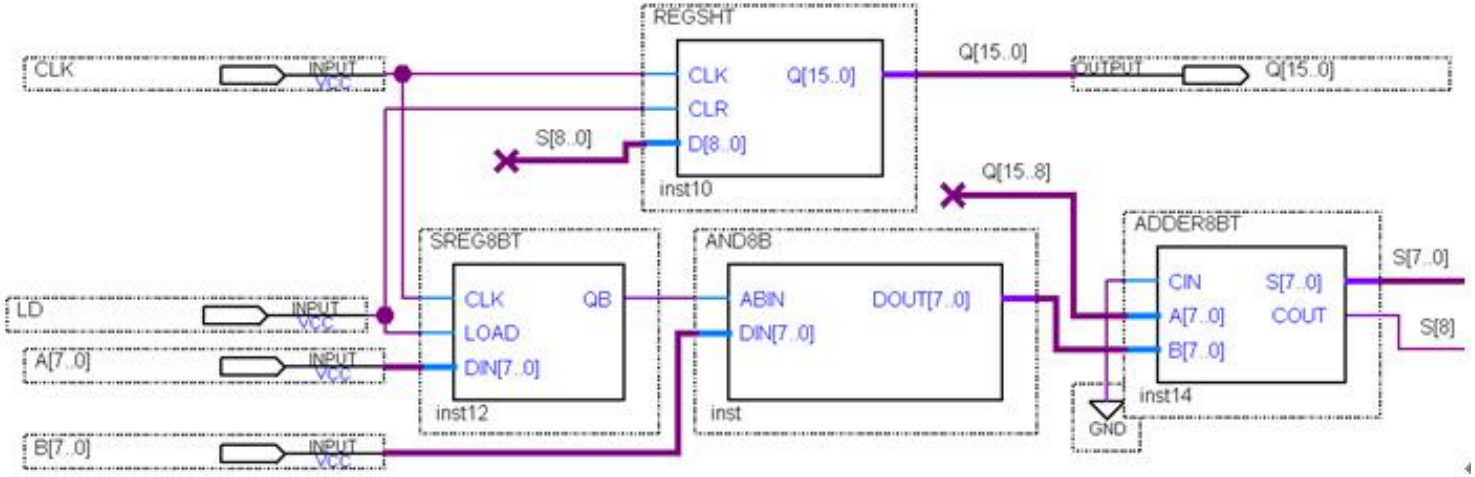

图 5-13 8 位乘法器逻辑原理图

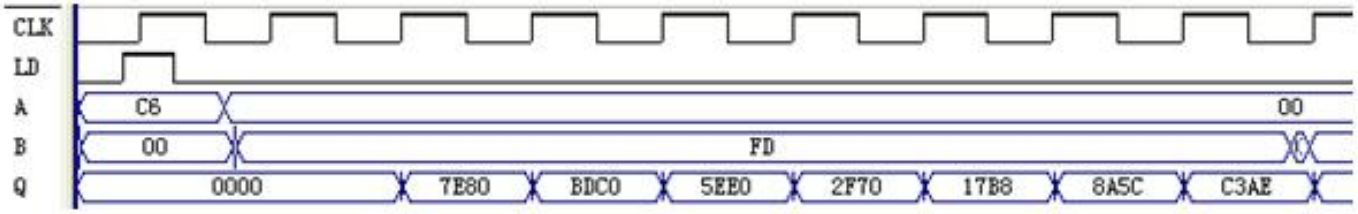

图 5-14 8 位移位相加乘法器运算逻辑波形图

实验与设计

实验5-4 基于VHDL代码的频率计设计

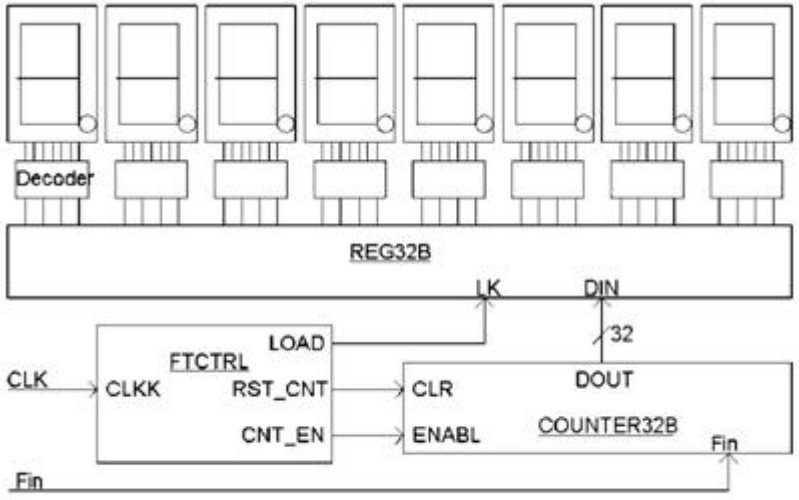

图 5-15 频率计电路框图

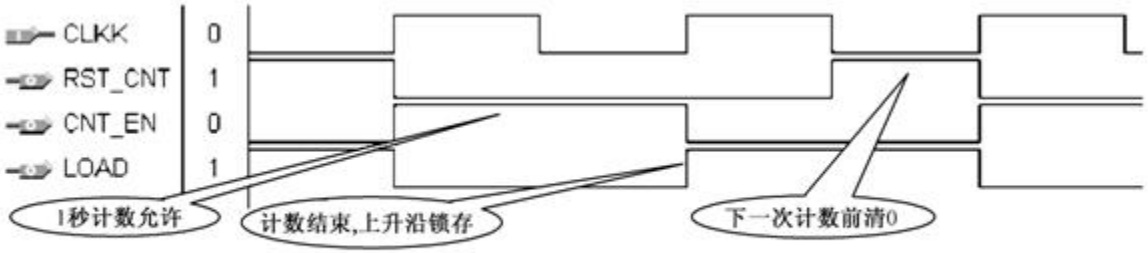

图 5-16 频率计测频控制器 FTCTRL 测控时序图

【例 5-19】

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

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

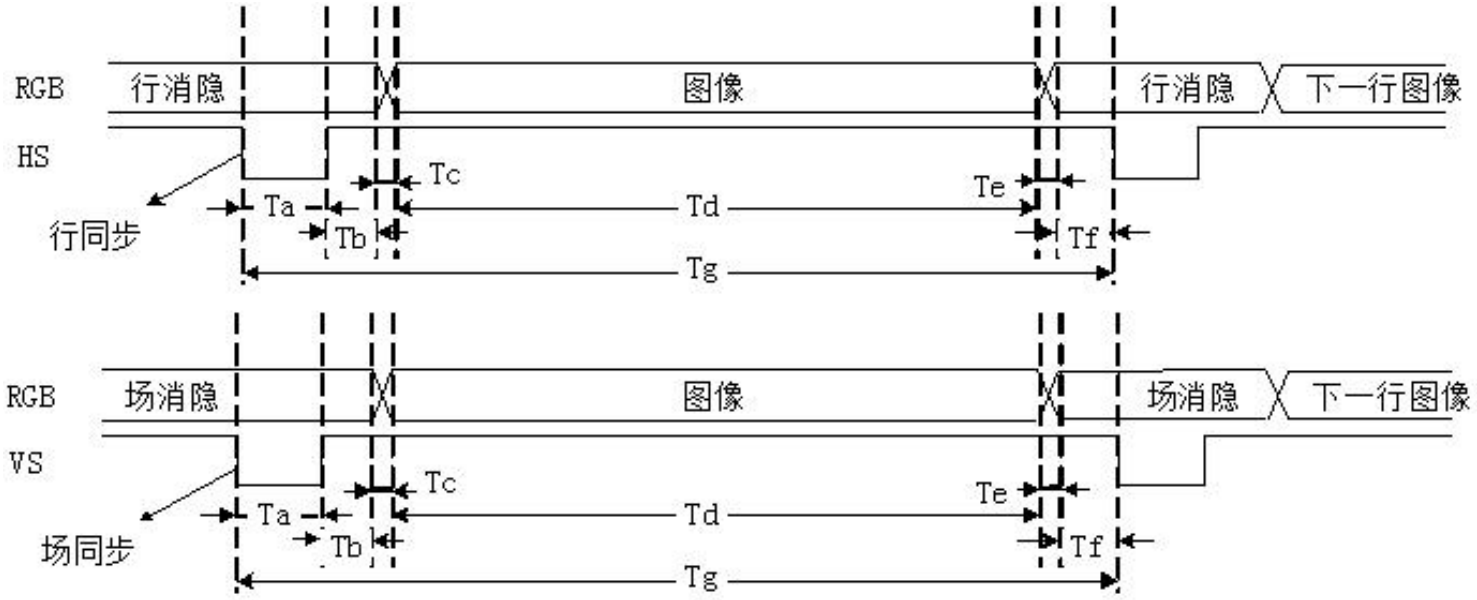

图 5-17 VGA 行扫描、场扫描时序示意图

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

表 5-4 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 5-5 场扫描时序要求 （单位：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

实验5-5 VGA彩条信号显示控制电路设计

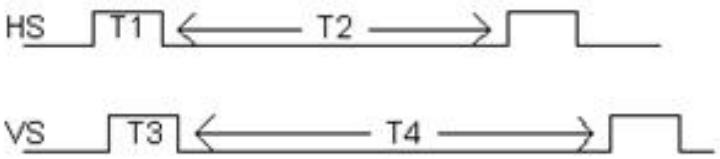

图 5-18 HS 和 VS 的时序图

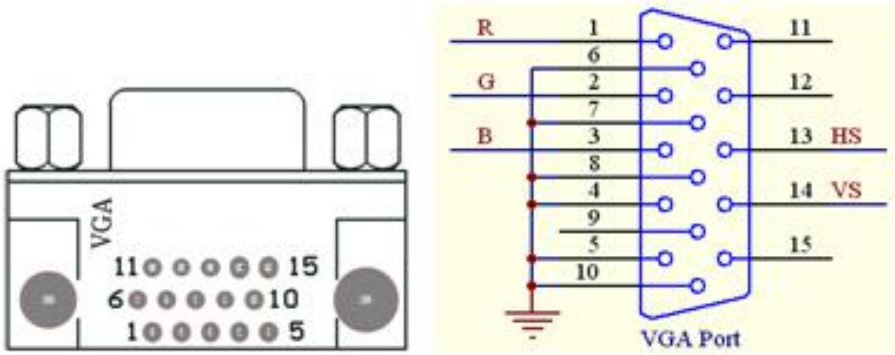

图 5-19 VGA 接口电路图，左接口从上往下看

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

表 5-6 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 5-7 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

【例 5-20】

```
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  ); -- 行场同步及红，绿，兰控制
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);
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;
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;
```

```
  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
     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

实验5-6 不同类型的移位寄存器设计实验