EDA技术实用教程

第12章

VHDL知识拾遗

12.1 VHDL库

12.1.1 库的种类

1. IEEE库

2. STD库

```
LIBRARY STD ;
USE STD.STANDARD.ALL ;
```

3. WORK库

4. VITAL库

12.1 VHDL库

12.1.2 库的用法

```
USE 库名.程序包名.项目名 ;
USE 库名.程序包名.ALL ;
```

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.STD_ULOGIC ;
USE IEEE.STD_LOGIC_1164.RISING_EDGE ;
```

```
USE WORK.STD_LOGIC_1164.ALL ;
```

12.2 VHDL程序包

```
PACKAGE   程序包名   IS                    --   程序包首
    程序包首说明部分
END   程序包名;
PACKAGE BODY   程序包名   IS               --   程序包体
    程序包体说明部分以及包体内容
END   程序包名;
```

12.2 VHDL程序包

【例 12-1】

```
PACKAGE pacl IS                                    -- 程序包首开始
  TYPE byte IS RANGE 0 TO 255 ;                    -- 定义数据类型byte
  SUBTYPE nibble IS byte RANGE 0 TO 15 ;           -- 定义子类型nibble
CONSTANT byte_ff  : byte := 255 ;                  -- 定义常数byte_ff
SIGNAL addend : nibble ;                           -- 定义信号addend
COMPONENT byte_adder                               -- 定义元件
PORT( a, b : IN byte;   c : OUT byte;  overflow : OUT BOOLEAN);
  END COMPONENT ;
  FUNCTION my_function (a : IN byte)Return byte ; -- 定义函数
END pacl ;                                         -- 程序包首结束
```

```
LIBRARY WORK ;
USE WORK.pacl.ALL ;
```

12.2 VHDL程序包

【例 12-2】

```
PACKAGE  seven IS
    SUBTYPE segments is BIT_VECTOR(0 TO 6);
    TYPE bcd IS RANGE 0 TO 9 ;
END seven ;
USE WORK.seven.ALL ;                                    -- WORK库默认是打开的
ENTITY decoder IS
    PORT (input: in bcd;   drive : out segments);
END decoder ;
ARCHITECTURE simple OF decoder IS
BEGIN
    WITH input SELECT
      drive <= "1111110"  WHEN 0 ,    "0110000"  WHEN 1 ,
               "1101101"  WHEN 2 ,    "1111001"  WHEN 3 ,
               "0110011"  WHEN 4 ,    "1011011"  WHEN 5 ,
               "1011111"  WHEN 6 ,    "1110000"  WHEN 7 ,
               "1111111"  WHEN 8 ,    "1111011"  WHEN 9 ,
               "0000000"  WHEN  OTHERS ;
END simple ;
```

12.3 VHDL文字规则补充说明

12.3.1 数字

整数　　5,　678,　0,　156E2(=15600),　45_234_287 (=45234287)

实数

```
1.335; 88_670_551.453_909(=88670551.453909); 1.0; 44.99E-2(=0.4499)
```

```
SIGNAL d1,d2,d3,d4,d5, : INTEGER RANGE 0 TO 255;--全部定义为整数类型
 d1 <= 10#170# ;              -- (十进制表示，等于 170)
 d2 <= 16#FE#    ;            -- (十六进制表示，等于 254)
 d3 <= 2#1111_1110#;          -- (二进制表示，等于 254)——也是整数类型!
 d4 <= 8#376#       ;         -- (八进制表示，等于 254)
 d5 <= 16#A#E3      ;         -- (十六进制表示，等于16#A000#)
```

```
G <= conv_std_logic_vector(16#A#E3, 16); --假设G的类型是std_logic_vector
```

物理量文字（**VHDL**综合器不接受此类文字）

```
60s (60秒),  100m (100米),  k (千欧姆),  177A (177安培)
```

12.3 VHDL文字规则补充说明

12.3.2 字符串

```
'R' , 'a' , '*' , 'Z' , 'U' , '0' , '11' ,'-' ,'L' …

 TYPE STD_ULOGIC IS ( 'U','X','0','1','W','L','H','-' )

"ERROR"  ,  "Both S and Q equal to 1" , "X"  , "BB$CC"
```

```
data1 <= B"1_1101_1110"          -- 二进制数数组，位矢数组长度是9
data2 <= O"15"                   -- 八进制数数组，位矢数组长度是6
data3 <= X"AD0"                  -- 十六进制数数组，位矢数组长度是12
data4 <= B"101_010_101_010"      -- 二进制数数组，位矢数组长度是12
data5 <= "101_010_101_010"       --表达错误，缺B
data6 <= "101010101010"          --表达正确。这里可以略去B，但不可加下划线
data6 <= "0AD0"                  --表达错误，缺X
```

12.3 VHDL文字规则补充说明

12.3.3 标识符及其表述规则

```
Decoder_1  ,  FFT  ,  Sig_N  ,  Not_Ack  ,  State0  ,  Idle
```

```
_Decoder_1          -- 起始为非英文字母
74LS164             -- 起始为数字
Sig_#N              -- 符号“#”不能成为标识符的构成
Not-Ack             -- 符号“-”不能成为标识符的构成
RyY_RST_            -- 标识符的最后不能是下划线“_”
data_ _BUS          -- 标识符中不能有双下划线
return              -- 关键词
```

12.3 VHDL文字规则补充说明

12.3.4 下标名

标识符(表达式)

```
SIGNAL  a, b : BIT_VECTOR (0 TO 3);
SIGNAL  m    : INTEGER  RANGE 0 TO 3 ;
SIGNAL  y, z : BIT ;
y <= a(m);                  --  不可计算型下标表示
z <= b(3);                  --  可计算型下标表示
```

12.4 子 程 序

12.4.1 函数

```
FUNCTION 函数名(参数表) RETURN  数据类型                --函数首
FUNCTION  函数名(参数表) RETURN  数据类型 IS             --函数体
    [ 说明部分 ]
    BEGIN
    顺序语句 ;
    END FUNCTION  函数名;
```

12.4 子 程 序

12.4.1 函数

【例 12-3】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
PACKAGE packexp IS                                    --定义程序包
  FUNCTION max( a,b : IN STD_LOGIC_VECTOR) --定义函数首
    RETURN STD_LOGIC_VECTOR ;
  FUNCTION func1 ( a,b,c : REAL )                     --定义函数首
    RETURN REAL ;
  FUNCTION "*" ( a ,b : INTEGER )                     --定义函数首
    RETURN INTEGER ;
  FUNCTION as2 (SIGNAL in1 ,in2 : REAL )  --定义函数首
    RETURN REAL ;
 END ;
PACKAGE BODY packexp IS
  FUNCTION max( a,b : IN STD_LOGIC_VECTOR)  --定义函数体
    RETURN STD_LOGIC_VECTOR IS
```

12.4 子 程 序

12.4.1 函数

```
  BEGIN
    IF a > b THEN RETURN  a;  ELSE  RETURN  b;   END IF;
  END FUNCTION max;          --结束FUNCTION语句
END;                         --结束PACKAGE  BODY语句
LIBRARY IEEE;                -- 函数应用实例
USE IEEE.STD_LOGIC_1164.ALL;
USE WORK.packexp.ALL ;
ENTITY  axamp IS
  PORT(dat1,dat2 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       dat3,dat4 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       out1,out2 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
 END;
 ARCHITECTURE bhv OF axamp IS
   BEGIN
    out1 <=  max(dat1,dat2); --用在赋值语句中的并行函数调用语句
   PROCESS(dat3,dat4)   BEGIN
    out2 <= max(dat3,dat4);      --顺序函数调用语句
   END PROCESS;
  END;
```

12.4 子 程 序

12.4.1 函数

【例 12-4】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY func IS
  PORT  (a :   IN STD_LOGIC_VECTOR (0 to 2);
         m :  OUT STD_LOGIC_VECTOR (0 to 2));
END ENTITY func ;
ARCHITECTURE demo OF func IS
FUNCTION sam(x ,y ,z : STD_LOGIC) RETURN STD_LOGIC IS
BEGIN
  RETURN (x AND y)OR y  ;
END FUNCTION sam ;
BEGIN
  PROCESS (a)    BEGIN
   m(0)<= sam(a(0), a(1), a(2));
   m(1)<= sam(a(2), a(0), a(1));
   m(2)<= sam(a(1), a(2), a(0));
  END PROCESS ;
END ARCHITECTURE demo ;
```

12.4 子 程 序

12.4.2 重载函数

【例 12-5】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
PACKAGE  packexp IS                                  --定义程序包
 FUNCTION  max( a,b : IN STD_LOGIC_VECTOR)   --定义函数首
   RETURN STD_LOGIC_VECTOR ;
 FUNCTION  max( a,b : IN BIT_VECTOR)         --定义函数首
   RETURN BIT_VECTOR ;
 FUNCTION  max( a,b : IN INTEGER )            --定义函数首
   RETURN INTEGER ;
 END;
PACKAGE BODY packexp IS
 FUNCTION  max( a,b : IN STD_LOGIC_VECTOR)   --定义函数体
   RETURN STD_LOGIC_VECTOR IS
 BEGIN
   IF a > b THEN RETURN  a;  ELSE   RETURN  b;  END IF;
  END FUNCTION max;                                 --结束FUNCTION语句
 FUNCTION  max( a,b : IN INTEGER)              --定义函数体
   RETURN INTEGER IS
 BEGIN
   IF a > b THEN RETURN  a;  ELSE   RETURN  b; END IF;
  END FUNCTION max;                                 --结束FUNCTION语句
  FUNCTION  max( a,b : IN BIT_VECTOR)          --定义函数体
   RETURN BIT_VECTOR IS
 BEGIN
```

12.4 子 程 序

12.4.2 重载函数

```
   IF a > b THEN RETURN  a;  ELSE  RETURN  b;  END IF;
  END FUNCTION max;                              --结束FUNCTION语句
END;                                             --结束PACKAGE BODY语句
LIBRARY IEEE ;          -- 以下是调用重载函数max的程序:
USE IEEE.STD_LOGIC_1164.ALL ;
 USE WORK.packexp.ALL;
ENTITY  axamp IS
  PORT(a1,b1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       a2,b2 : IN BIT_VECTOR(4 DOWNTO 0);
       a3,b3 : IN INTEGER RANGE 0 TO 15;
          c1 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          c2 : OUT BIT_VECTOR(4 DOWNTO 0);
          c3 : OUT INTEGER RANGE 0 TO 15);
 END;
ARCHITECTURE bhv OF axamp IS
  BEGIN
  c1 <=  max(a1,b1); --对函数max(a,b : IN STD_LOGIC_VECTOR)的调用
  c2 <=  max(a2,b2); --对函数max(a,b : IN BIT_VECTOR)的调用
  c3 <=  max(a3,b3); --对函数max(a,b : IN INTEGER)的调用
END;
```

12.4 子 程 序

12.4.2 重载函数

【例 12-6】

```
LIBRARY IEEE ;                                          -- 程序包首
USE IEEE.std_logic_1164.all ;
USE IEEE.std_logic_arith.all ;
PACKAGE STD_LOGIC_UNSIGNED is
function "+" (L : STD_LOGIC_VECTOR ; R : INTEGER)
                 return STD_LOGIC_VECTOR ;
function "+" (L : INTEGER;  R : STD_LOGIC_VECTOR)
                 return STD_LOGIC_VECTOR ;
function "+" (L : STD_LOGIC_VECTOR ; R : STD_LOGIC )
return STD_LOGIC_VECTOR ;
function SHR (ARG : STD_LOGIC_VECTOR ;
  COUNT : STD_LOGIC_VECTOR )return STD_LOGIC_VECTOR ;
...
end STD_LOGIC_UNSIGNED  ;
```

12.4 子 程 序

12.4.2 重载函数

```
-- 以下是程序包体
LIBRARY IEEE ;
use IEEE.std_logic_1164.all ;
use IEEE.std_logic_arith.all ;
package body STD_LOGIC_UNSIGNED is
function maximum (L, R : INTEGER)return INTEGER is
begin
   if L > R then return L; else return R; end if;
end;
function "+" (L : STD_LOGIC_VECTOR ; R : INTEGER)
return STD_LOGIC_VECTOR is
Variable result : STD_LOGIC_VECTOR (L'range);
Begin
 result := UNSIGNED(L)+ R ;
 return std_logic_vector(result);
end ;
...
end STD_LOGIC_UNSIGNED ;
```

12.4 子 程 序

12.4.3 决断函数

12.4.4 过程

```
PROCEDURE 过程名(参数表)                          -- 过程首

PROCEDURE 过程名(参数表)IS
    [说明部分]
    BIGIN                                         -- 过程体
      顺序语句;
     END PROCEDURE 过程名;
```

```
PROCEDURE  pro1  (VARIABLE  a, b :  INOUT REAL);
PROCEDURE  pro2 (CONSTANT  a1 :  IN INTEGER ;
                 VARIABLE  b1 :  OUT INTEGER );
PROCEDURE  pro3  (SIGNAL sig :  INOUT BIT);
```

12.4 子 程 序

12.4.4 过程

【例 12-7】

```
PROCEDURE  prg1(VARIABLE value :INOUT BIT_VECTOR(0 TO 7)) IS
BEGIN
 CASE value IS
  WHEN "0000" => value: "0101" ;
  WHEN "0101" => value: "0000" ;
  WHEN OTHERS => value: "1111" ;
 END CASE ;
END PROCEDURE  prg1 ;
```

12.4 子 程 序

12.4.4 过程

【例 12-8】

```
PROCEDURE  comp(a,r : IN REAL;  m : IN INTEGER;  v1,v2: OUT REAL) IS
  VARIABLE cnt : INTEGER ;
  BEGIN
  v1 := 1.6 * a ;    v2 := 1.0 ;          -- 赋初始值
  Q1 :  FOR  cnt  IN  1 TO m  LOOP
   v2 :=  v2 * v1 ;
EXIT  Q1  WHEN  v2 > v1;                   -- 当v2>v1，跳出循环LOOP
   END LOOP Q1
 ASSERT (v2 < v1 )
    REPORT  "OUT OF RANGE"                 -- 输出错误报告
       SEVERITY  ERROR ;
  END PROCEDURE  comp  ;
```

12.4 子 程 序

12.4.4 过程

【例 12-9】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
PACKAGE axamp IS                                          --过程首定义
 PROCEDURE nand4a (SIGNAL a,b,c,d : IN STD_LOGIC ;
                   SIGNAL  y : OUT  STD_LOGIC );
 END axamp;
PACKAGE BODY axamp IS                                     --过程体定义
 PROCEDURE nand4a  (SIGNAL a,b,c,d : IN STD_LOGIC ;
                    SIGNAL  y : OUT  STD_LOGIC ) IS
 BEGIN
   y<= NOT(a AND b AND c AND d);
 RETURN;
END nand4a;
END axamp;
LIBRARY IEEE;                                             --主程序
USE IEEE.STD_LOGIC_1164.ALL;
USE WORK.axamp.ALL;
ENTITY EX IS
  PORT( e,f,g,h : IN STD_LOGIC ;  x : OUT  STD_LOGIC );
 END;
ARCHITECTURE bhv OF EX IS
    BEGIN
     nand4a(e,f,g,h,x) ;                                  --并行调用过程
END;
```

12.4 子 程 序

12.4.5 重载过程

【例 12-10】

```
PROCEDURE calcu (v1, v2 : IN REAL;  SIGNAL out1 : INOUT INTEGER);
PROCEDURE calcu (v1, v2 : IN INTEGER; SIGNAL out1 : INOUT REAL);
...
calcu (20.15, 1.42, signl);      -- 调用第一个重载过程 calcu
calcu (23,  320, sign2 );        -- 调用第二个重载过程 calcu
...
```

12.5 数据类型

（1）布尔类型

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

（2）位数据类型

```
TYPE BIT IS ('0', '1');
```

（3）位矢量类型

```
TYPE BIT_VECTOR IS ARRAY (Natural Range <> )OF  BIT ;

SIGNAL a : BIT_VECTOR(7 TO 0);
```

12.5 数据类型

（4）字符类型

（5）整数类型

（6）实数类型

1.0	十进制浮点数	65_971.333_3333	十进制浮点数
0.0	十进制浮点数	8#43.6#e+4	八进制浮点数
65971.333333	十进制浮点数	43.6E－4	十进制浮点数

12.5 数据类型

（7）字符串类型

```
VARIABLE string_var : STRING (1 TO 7 );
 string_var := "a b c d" ;
```

（8）时间类型

```
TYPE time IS RANGE -2147483647 TO 2147483647
   units
      fs ;                    -- 飞秒，VHDL中的最小时间单位
      ps = 1000 fs ;          -- 皮秒
      ns = 1000 ps ;          -- 纳秒
      us = 1000 ns ;          -- 微秒
      ms = 1000 us ;          -- 毫秒
      sec = 1000 ms ;         -- 秒
      min = 60 sec ;          -- 分
      hr = 60 min ;           -- 时
end units ;
```

12.5 数据类型

（9）文件类型

```
PROCEDUER Readline (F: IN TEXT; L: OUT LINE);
PROCEDUER Writeline (F: OUT TEXT; L: IN LINE);
PROCEDUER Read (   L: INOUT LINE; Value: OUT std_logic;
                Good: OUT BOOLEAN);
PROCEDUER Read (L: INOUT LINE; Value: OUT std_logic);
PROCEDUER Read (L: INOUT LINE; Value: OUT std_logic_ vector;
             Good: OUT BOOLEAN);
PROCEDUER Read (L: INOUT LINE; Value: OUT std_logic_ vector;
PROCEDUER Write (L: INOUT LINE; Value: IN std_logic;
               Justiaied: IN SIDE :=Right;field; IN WIDTH :=0);
PROCEDUER Write (L: INOUT LINE; Value: IN std_logic _ vector,
            Justiaied: IN SIDE :=Right;field; IN WIDTH :=0);
```

12.6 VHDL操作符补充说明

12.6.1 逻辑操作符

12.6.2 关系操作符

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

12.6.3 算术操作符

表 12-1 算术操作符分类表

序 号	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

12.6 VHDL操作符补充说明

1. 求和操作符

【例 12-11】

```
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 ;
```

12.6 VHDL操作符补充说明

2．求积操作符

3．符号操作符

4．混合操作符

12.7 VHDL基本语句补充说明

12.7.1 NEXT语句

```
NEXT;                                    -- 语句格式1
NEXT LOOP标号;                           -- 语句格式2
NEXT LOOP标号 WHEN 条件表达式;           -- 语句格式3
```

【例 12-12】

```
...
L1 : FOR cnt_value IN 1 TO 8 LOOP
s1 : a(cnt_value) := '0';
      NEXT WHEN (b=c);
s2 : a(cnt_value + 8 ):= '0';
END LOOP L1;
```

12.7 VHDL基本语句补充说明

12.7.1 NEXT语句

【例 12-13】

```
...
L_x  : FOR cnt_value IN 1 TO 8 LOOP
 s1  :  a(cnt_value):= '0';
                  k := 0;
L_y  : LOOP
  s2 : b(k) := '0';
       NEXT L_x WHEN (e>f);
  s3 : b(k+8) := '0';
            k := k+1;
       NEXT LOOP L_y ;
       NEXT LOOP L_x ;
 ...
```

12.7 VHDL基本语句补充说明

12.7.2 EXIT语句

```
EXIT;                                  -- 语句格式1
EXIT LOOP标号;                         -- 语句格式2
EXIT LOOP标号 WHEN 条件表达式;         -- 语句格式3
```

【例 12-14】

```
SIGNAL a, b : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL a_less_then_b : Boolean;
...
 a_less_then_b <= FALSE ;                                    -- 设初始值
 FOR i IN 1 DOWNTO 0 LOOP
 IF (a(i)='1' AND b(i)='0') THEN  a_less_then_b<=FALSE; -- a > b
 EXIT ;
 ELSIF (a(i)='0' AND b(i)='1') THEN  a_less_then_b<=TRUE; -- a < b
    EXIT;
    ELSE    NULL;
    END IF;
 END LOOP;                          -- 当 i=1时返回LOOP语句继续比较
```

12.7 VHDL基本语句补充说明

12.7.3 WAIT语句

```
WAIT;                              -- 语句格式1
WAIT ON 信号表;                    -- 语句格式2
WAIT UNTIL 条件表达式;             -- 语句格式3
WAIT FOR 时间表达式;               -- 语句格式4，超时等待语句
```

【例 12-15】

```
SIGNAL s1,s2 : STD_LOGIC;
...
PROCESS  BEGIN
...
WAIT ON s1,s2 ;
END PROCESS ;
```

12.7 VHDL基本语句补充说明

12.7.3 WAIT语句

```
WAIT UNTIL clock ='1';
WAIT UNTIL rising_edge(clock);
WAIT UNTIL NOT clock'STABLE AND clock ='1';
WAIT UNTIL clock ='1' AND clock'EVENT;
```

【例 12-17】

```
PROCESS  BEGIN
WAIT UNTIL clk ='1';   ave <= a;
WAIT UNTIL clk ='1';   ave <= ave + a;
WAIT UNTIL clk ='1';   ave <= ave + a;
WAIT UNTIL clk ='1';   ave <= (ave + a)/4 ;
END PROCESS ;
```

12.7 VHDL基本语句补充说明

12.7.3 WAIT语句

【例 12-18】

```
 PROCESS  BEGIN
  rst_loop :  LOOP
  WAIT UNTIL clock ='1' AND clock'EVENT;   -- 等待时钟信号
  NEXT rst_loop WHEN (rst='1');            -- 检测复位信号rst
  x <= a ;                                 -- 无复位信号，执行赋值操作
  WAIT UNTIL clock ='1' AND clock'EVENT;   -- 等待时钟信号
  NEXT rst_loop When (rst='1');            -- 检测复位信号rst
  y <= b ;                                 -- 无复位信号，执行赋值操作
  END LOOP rst_loop ;
END PROCESS;
```

12.7 VHDL基本语句补充说明

12.7.3 WAIT语句

【例 12-19】

```
ROCESS  BEGIN
WAIT UNTIL (RISING_EDGE(clk) );                    --等待时钟上升沿
 IF (reset='1') THEN   qout<="00000000"; ELSE
  CASE mode IS
   WHEN "01" => qout<=shift_right & qout(7 DOWNTO 1); --右移
   WHEN "10" => qout<=qout(6 DOWNTO 0) & shift_left;  --左移
   WHEN "11" => qout <= data;                         -- 并行加载
   WHEN OTHERS => NULL;
   END CASE;
 END IF;
END PROCESS;
```

```
PROCESS  BEGIN
  CLK<='0';
  WAIT FOR 10 ns ;
  CLK<='1';
  WAIT FOR 10 ns ;
END PROCESS;
```

12.7 VHDL基本语句补充说明

12.7.4 子程序调用语句

```
过程名[([形参名=> ]实参表达式
        { , [形参名=> ]实参表达式 }) ];
```

【例 12-20】

```
PACKAGE data_types IS                                          -- 定义程序包
SUBTYPE data_element IS INTEGER RANGE 0 TO 3 ;                 -- 定义数据类型
TYPE data_array IS ARRAY (1 TO 3) OF data_element;
END data_types;
USE WORK.data_types.ALL;  --打开以上建立在当前工作库的程序包data_types
ENTITY sort IS
   PORT ( in_array : IN  data_array;  out_array : OUT data_array);
END sort;
 ARCHITECTURE exmp OF sort IS
 BEGIN
 PROCESS (in_array)                    -- 进程开始，设data_types为敏感信号
  PROCEDURE swap(data : INOUT data_array; low, high : IN INTEGER) IS
                                        -- swap的形参名为data、low、high
    VARIABLE temp : data_element;
   BEGIN                                -- 开始描述本过程的逻辑功能
     IF (data(low) > data(high)) THEN  -- 检测数据
          temp := data(low) ;   data(low) := data(high);
          data(high) := temp ;       END IF ;
  END swap ;                              -- 过程swap定义结束
  VARIABLE my_array : data_array ;       -- 在本进程中定义变量my_array
  BEGIN                                 -- 进程开始
   my_array := in_array ;                -- 将输入值读入变量
   swap(my_array,1,2);-- my_array、1、2是对应于data、low、high的实参
   swap(my_array, 2, 3);        -- 位置关联法调用，第2、第3元素交换
   swap(my_array, 1, 2);        -- 位置关联法调用，第1、第2元素再次交换
    out_array <= my_array ;
END PROCESS;
END exmp ;
```

12.7 VHDL基本语句补充说明

12.7.4 子程序调用语句

【例 12-21】

```
ENTITY sort4 is
GENERIC (top : INTEGER :=3);
   PORT (a, b, c, d : IN BIT_VECTOR (0 TO top);
     ra, rb, rc, rd : OUT BIT_VECTOR (0 TO top));
END sort4;
ARCHITECTURE muxes OF sort4 IS
PROCEDURE sort2(x, y : INOUT BIT_VECTOR (0 TO top)) is
   VARIABLE tmp : BIT_VECTOR (0 TO top);
BEGIN
   IF x>y THEN  tmp := x;  x := y; y := tmp;    END IF;
END sort2;
BEGIN
 PROCESS (a, b, c, d)↵
   VARIABLE va, vb, vc, vd : BIT_VECTOR(0 TO top);
BEGIN
   va := a;  vb := b; vc := c;  vd := d;
   sort2(va, vc);
   sort2(vb, vd);
   sort2(va, vb);
   sort2(vc, vd);
   sort2(vb, vc);
   ra <= va; rb <= vb;  rc <= vc;  rd <= vd;
   END PROCESS;
END muxes;
```

12.7 VHDL基本语句补充说明

12.7.5 RETURN语句

```
RETURN;                          -- 语句格式1
RETURN 表达式;                   -- 语句格式2
```

【例 12-22】

```
PROCEDURE rs (SIGNAL s ,  r :  IN   STD_LOGIC ;
              SIGNAL q , nq : INOUT STD_LOGIC) IS
BEGIN
 IF ( s ='1' AND r ='1') THEN
   REPORT "Forbidden state : s and r are quual to '1'";
   RETURN ;
   ELSE   q<= (s AND nq) AFTER 5 ns;  nq<= (s AND q) AFTER 5 ns;
  END IF ;
END PROCEDURE rs ;
```

12.7 VHDL基本语句补充说明

12.7.5 RETURN语句

【例 12-23】

```
FUNCTION opt (a, b, opr :STD_LOGIC)  RETURN  STD_LOGIC IS
BEGIN
IF (opr ='1') THEN   RETURN (a AND b);
              ELSE  RETURN (a OR b) ;   END IF ;
END FUNCTION opt ;
```

12.8 VHDL并行语句补充说明

```
ARCHITECTURE 结构体名 OF 实体名 IS
    说明语句
    BEGIN
    并行语句
END ARCHITECTURE 结构体名
```

12.8.1 并行信号赋值语句

12.8.2 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 ;
```

12.8 VHDL并行语句补充说明

【例 12-24】

```
ENTITY gat IS
GENERIC(l_time : TIME ;  s_time : TIME ) ; -- （参数传递）类属说明
  PORT (b1, b2, b3 : INOUT BIT) ;           -- 结构体全局端口定义
  END ENTITY gat  ;
  ARCHITECTURE func OF gat IS
  SIGNAL  a1 : BIT ;                        -- 结构体全局信号a1定义
  BEGIN
  Blk1 : BLOCK                              -- 块定义，块标号名是blk1
  GENERIC (gb1, gb2 : Time) ;               -- 定义块中的局部类属参量
  GENERIC MAP (gb1 => l_time,gb2 => s_time);   -- 局部端口参量设定
  PORT (pb : IN BIT; pb2 : INOUT BIT );     -- 块结构中局部端口定义
  PORT MAP (pb1 => b1, pb2 => a1 ) ;        -- 块结构端口连接说明
  CONSTANT delay : Time := 1 ms ;           -- 局部常数定义
  SIGNAL s1 : BIT ;                         -- 局部信号定义
  BEGIN
  s1 <= pb1 AFTER delay ;
  pb2 <= s1 AFTER gb1, b1 AFTER gb2 ;
  END BLOCK blk1 ;
END ARCHITECTURE func ;
```

12.8 VHDL并行语句补充说明

12.8.3 并行过程调用语句

过程名(关联参量名);

【例 12-25】

```
PROCEDURE adder(SIGNAL a, b :IN STD_LOGIC ;        --过程名为adder
                SIGNAL sum : OUT STD_LOGIC );
 ...
 adder(a1, b1, sum1) ;                                  -- 并行过程调用
 ...              -- 在此，a1、b1、sum1即为分别对应于a、b、sum的关联参量名
 PROCESS( c1, c2)   BEGIN                               -- 进程语句执行
    Adder(c1, c2, s1) ;          -- 顺序过程调用，在此c1、c2、s1即为分别对
 END PROCESS ;                    -- 应于a、b、sum的关联参量名
```

12.8 VHDL并行语句补充说明

12.8.3 并行过程调用语句

【例 12-26】

```
PROCEDURE check(SIGNAL a : IN  STD_LOGIC_VECTOR;    -- 在调用时
            SIGNAL error : OUT BOOLEAN ) IS          -- 再定位宽
VARIABLE found_one : BOOLEAN := FALSE ;              -- 设初始值
BEGIN
FOR i IN a'RANGE LOOP      -- 对位矢量a的所有的位元素进行循环检测
IF a(i) = '1' THEN         -- 发现a中有 '1'
IF found_one THEN     -- 若found_one为TRUE，则表明发现了一个以上的'1'
  ERROR <= TRUE;      -- 发现了一个以上的'1'，令found_one为TRUE
    RETURN;           -- 结束过程
END IF;
Found_one := TRUE;    -- 在a中已发现了一个'1'
End IF;
End LOOP;             -- 再测a中的其他位
error <= NOT found_one; -- 如果没有任何'1' 被发现，error 将被置TRUE
END PROCEDURE  check;
```

12.8 VHDL并行语句补充说明

12.8.3 并行过程调用语句

```
CHBLK: BLOCK
SIGNAL s1: STD LOGIC VECTOR (0 TO 0);   -- 过程调用前设定位矢尺寸
SIGNAL s2: STD LOGIC VECTOR (0 TO 1);
SIGNAL s3: STD LOGIC VECTOR (0 TO 2);
SIGNAL s4: STD LOGIC VECTOR (0 TO 3);
SIGNAL e1, e2, e3, e4: Boolean;
BEGIN
 Check (s1, e1);              -- 并行过程调用，关联参数名为s1、e1
 Check (s2, e2);              -- 并行过程调用，关联参数名为s2、e2
 Check (s3, e3);              -- 并行过程调用，关联参数名为s3、e3
 Check (s4, e4);              -- 并行过程调用，关联参数名为s4、e4
  END  BLOCK;
```

12.8 VHDL并行语句补充说明

12.8.4 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
```

12.8 VHDL并行语句补充说明

12.8.4 生成语句

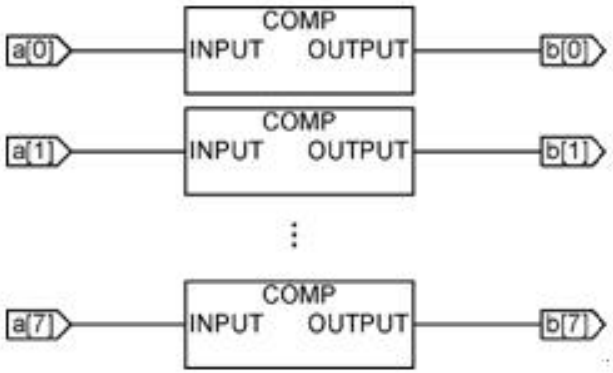

图 12-1 生成语句产生的相同的电路模块

12.8 VHDL并行语句补充说明

12.8.4 生成语句

【例 12-27】

```
COMPONENT comp
PORT (x : IN STD_LOGIC;  y : OUT STD_LOGIC );
END COMPONENT ;
SIGNAL a, 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;
```

【例 12-28】

```
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
bin_p_rs_ff : 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;
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;
```

12.8 VHDL并行语句补充说明

12.8.4 生成语句

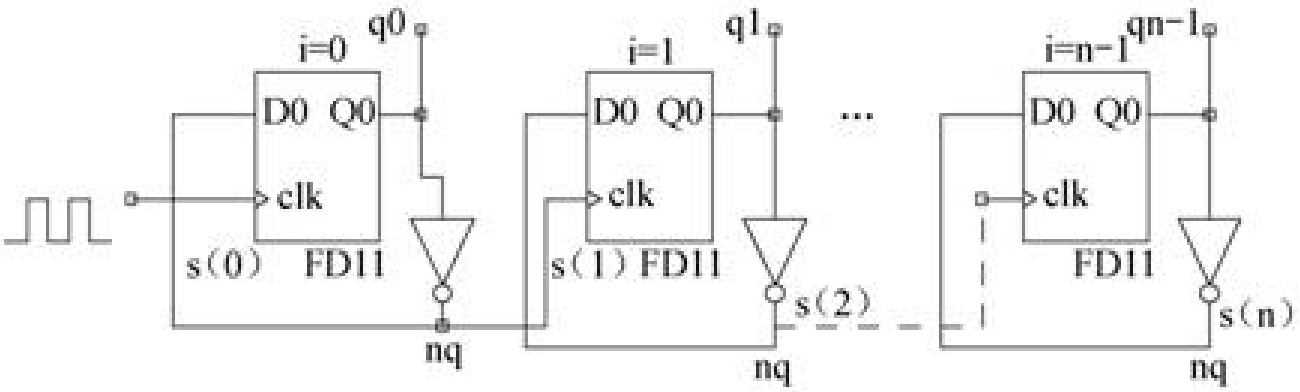

图 13-2　6 位二进制计数器逻辑原理图

12.8 VHDL并行语句补充说明

12.8.5 REPORT语句

```
REPORT <字符串> ;
```

【例 12-29】

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY RSFF2 IS
   PORT ( S, R : IN std_logic; Q, QF : OUT std_logic );
 END RSFF2;
 ARCHITECTURE BHV OF RSFF2 IS
 BEGIN
  P1: PROCESS(S,R)
    VARIABLE D : std_logic;
 BEGIN
   IF (R='1' and S='1') THEN
     REPORT " BOTH R AND S IS '1'"; --报告出错信息
   ELSIF (R='1' and S='0') THEN D := '0';
   ELSIF (R='0' and S='1') THEN D := '1' ;    END IF;
     Q  <= D;  QF <= NOT D;
   END PROCESS;
 END BHV;
```

12.8 VHDL并行语句补充说明

12.8.6 断言语句

```
ASSERT<条件表达式>
REPORT<出错信息>
SEVERITY<错误级别>；
```

表 12-2 预定义错误等级

Note（通报）	报告出错信息，可以通过编译
Warning（警告）	报告出错信息，可以通过编译
Error（错误）	报告出错信息，暂停编译
Failure（失败）	报告出错信息，暂停编译

12.8 VHDL并行语句补充说明

12.8.6 断言语句

1. 顺序断言语句

【例 12-30】

```
P1: PROCESS(S,R)
   VARIABLE D : std_logic;
BEGIN
 ASSERT not (R='1'and S='1')
      REPORT "both R and S equal to '1'"
      SEVERITY Error;
  IF  R = '1' and S = '0' THEN  D := '0';
     ELSIF (R='0' and S='1') THEN  D := '1' ;   END IF;
   Q  <= D;  QF <= NOT D;
 END PROCESS;
```

12.8 VHDL并行语句补充说明

12.8.6 断言语句

2. 并行断言语句

【例 12-31】

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY RSFF2 IS
 PORT(S, R : IN std_logic;   Q, QF : OUT std_logic);
 END  RSFF2;
  ARCHITECTURE BHV OF RSFF2 IS
    BEGIN
 PROCESS(R,S)   BEGIN
      ASSERT not (R='1'and S='1')
      REPORT "both R and S equal to ' 1 '"
      SEVERITY Error;
END PROCESS;
PROCESS(R,S)
  VARIABLE D : std_logic := '0';
   BEGIN
   IF (R='1' and S='0')  THEN D :='0';
     ELSIF (R='0' and S='1') THEN D :='1';  END IF;
      Q <= D ; QF <= NOT D ;
END PROCESS;
END ;
```

习　题

12-5 运算符重载函数通常要调用转换函数，以便能够利用已有的数据类型。下面给出一个新的数据类型**AGE**，并且下面的转换函数已经实现：

```
function CONV_INTEGER(ARG: AGE) return INTEGER;
```

仿照本章中的示例，利用此函数编写一个“**+**”运算符重载函数，支持下面的运算：

```
SIGNAL a, c : AGE;
...
C <= a + 20;
```