第10章

VHDL Test Bench仿真

10.1 VHDL行为仿真流程

图 10-1 HDL 系统设计描述层次

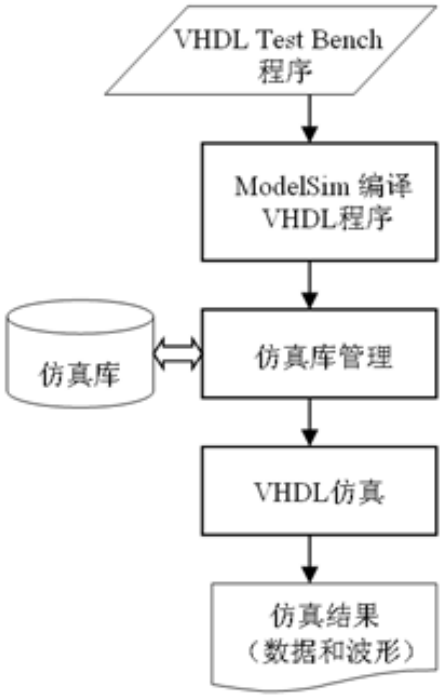

图 10-2 VHDL 仿真流程

10.2 VHDL测试基准实例

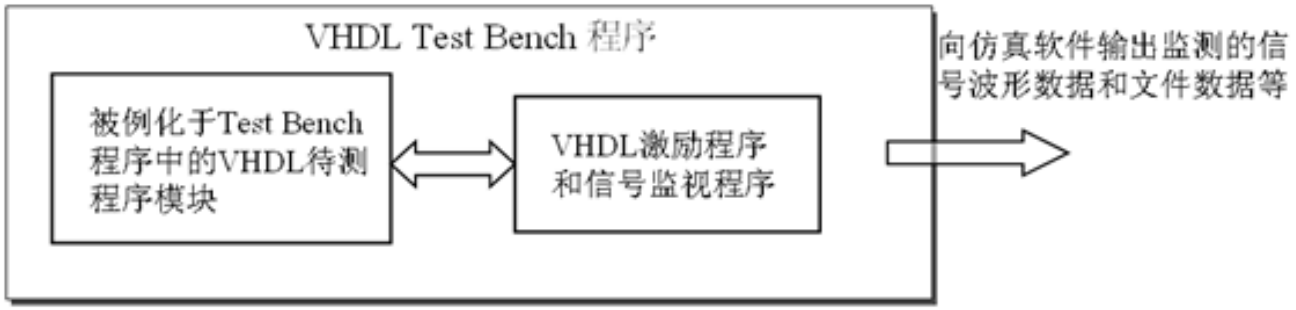

图 10-3　VHDL Test Bench 结构

【例 10-1】//Test Bench 文件名：CNT10_TB.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT10_TB IS
END CNT10_TB;
ARCHITECTURE ONE OF CNT10_TB IS
  COMPONENT CNT10
  PORT (CLK,RST,EN,LOAD : IN STD_LOGIC;
        DATA : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        DOUT : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        COUT : OUT STD_LOGIC);
  END COMPONENT;
SIGNAL  CLK  : STD_LOGIC :='0';  --定义向 CNT10 时钟端口输入的时钟信号
SIGNAL  RST  : STD_LOGIC :='1'; --定义向 CNT10 复位端口输入的复位信号
SIGNAL  EN   : STD_LOGIC :='0'; --定义向 CNT10 时钟使能端口输入的使能信号
SIGNAL  LOAD : STD_LOGIC :='1'; --定义控制 CNT10 加载的信号
SIGNAL  DATA : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL  DOUT : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL  COUT : STD_LOGIC;
CONSTANT CLK_P : TIME := 30 ns ;
         -- 定义时间类型常数是 CLK_P=30 ns,注意 30 与 ns 间应该有空格!
BEGIN
  U1:  CNT10  PORT MAP(CLK=>CLK, RST=>RST, EN=>EN, LOAD=>LOAD,
              DATA=>DATA, DOUT=>DOUT, COUT=>COUT); --例化待测试模块
 PROCESS  BEGIN  --产生时钟信号的进程，这是个没有敏感信号的永久自动启动的进程
  CLK<='0';     WAIT FOR  CLK_P; --CLK 首先输出 0，,30ns 后 输出 1.
  CLK<='1';     WAIT FOR  CLK_P; --再过 30ns 后返回。
END PROCESS;
  RST  <= '1', '0' AFTER 110 ns, '1' AFTER 114 ns; --RST 的电平控制
  EN   <= '0', '1' AFTER 40 ns;                    --EN 电平控制
  LOAD <= '1', '0' AFTER 910 ns, '1' AFTER 940 ns;
  DATA <= "0100", "0110" AFTER 400 ns,             --加载数据输出
              "0111" AFTER 700 ns, "0100" AFTER 1000 ns;
END ONE;
```

10.3 VHDL Test Bench测试流程

1. 安装ModelSim

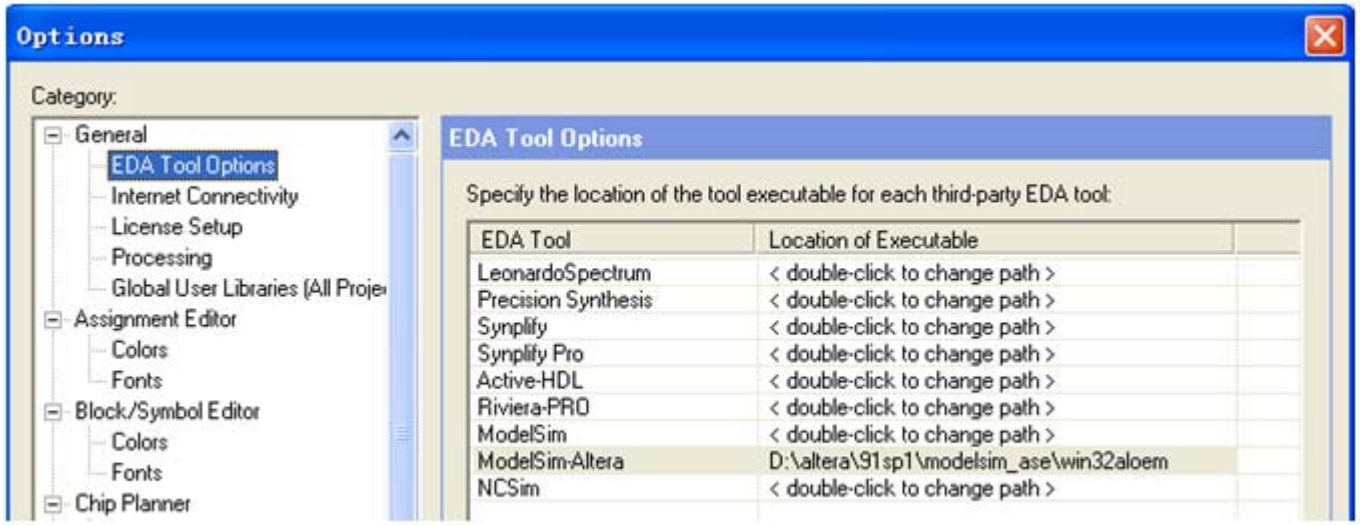

图 10-4　于 Quartus II 中设置接口 ModelSim-Altera 的路径

10.3 VHDL Test Bench测试流程

2. 为Test Bench仿真设置参数

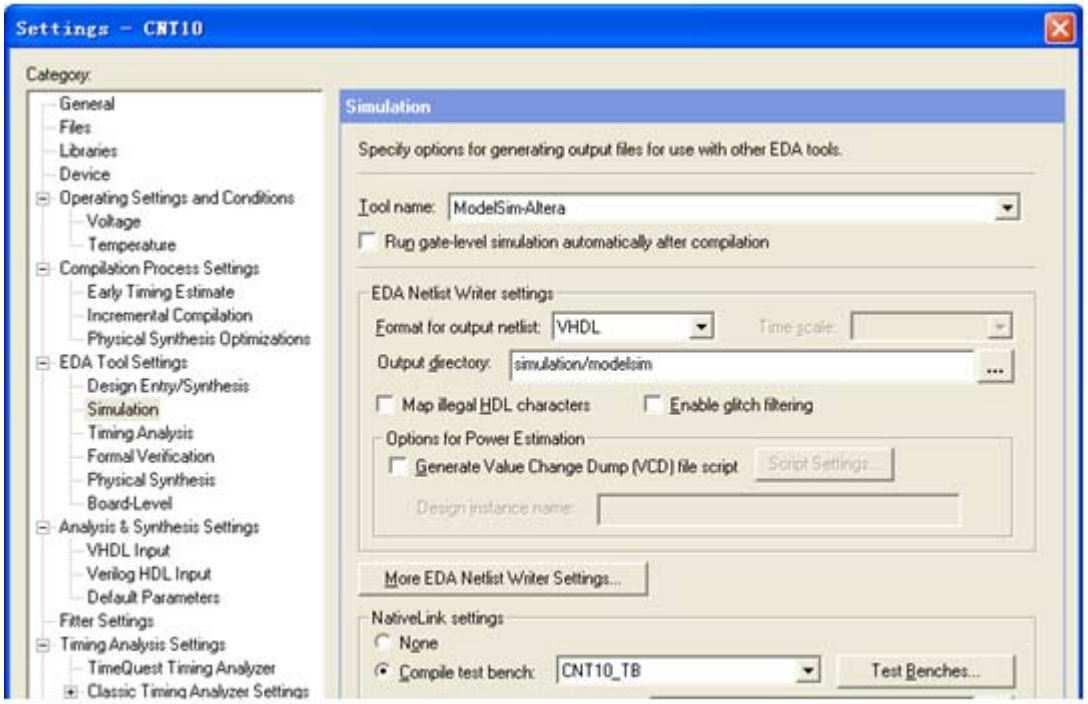

图 10-5　选择仿真工具名称和输出网表语言形式

10.3 VHDL Test Bench测试流程

2. 为Test Bench仿真设置参数

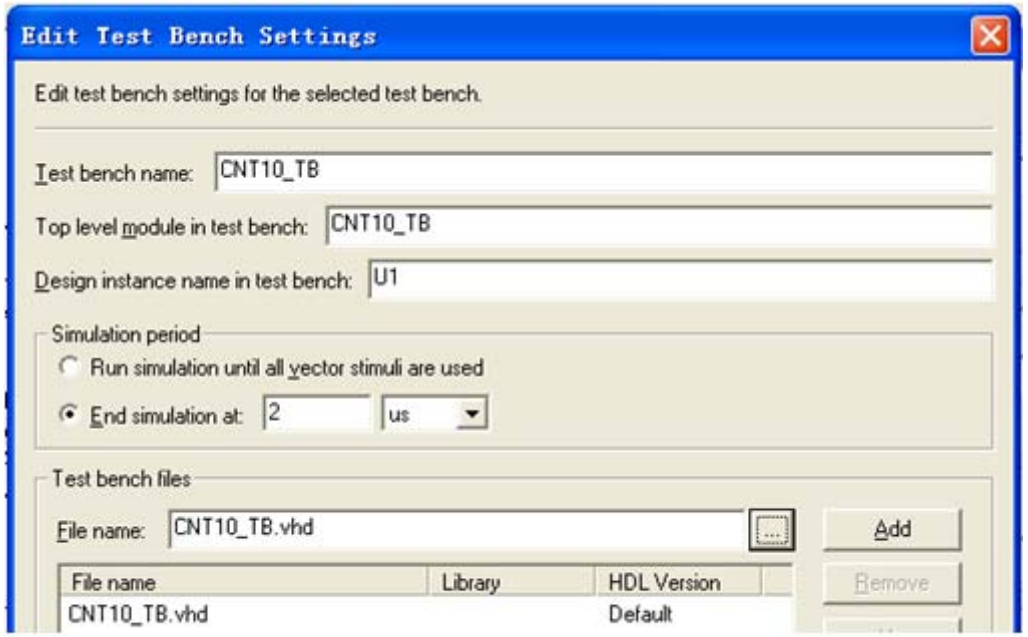

图 10-6　为 Test Bench 仿真设置参数

10.3 VHDL Test Bench测试流程

3. 启动**Test Bench**仿真

图 10-7　启动 RTL 级仿真

10.3 VHDL Test Bench测试流程

4. 分析Test Bench仿真结果

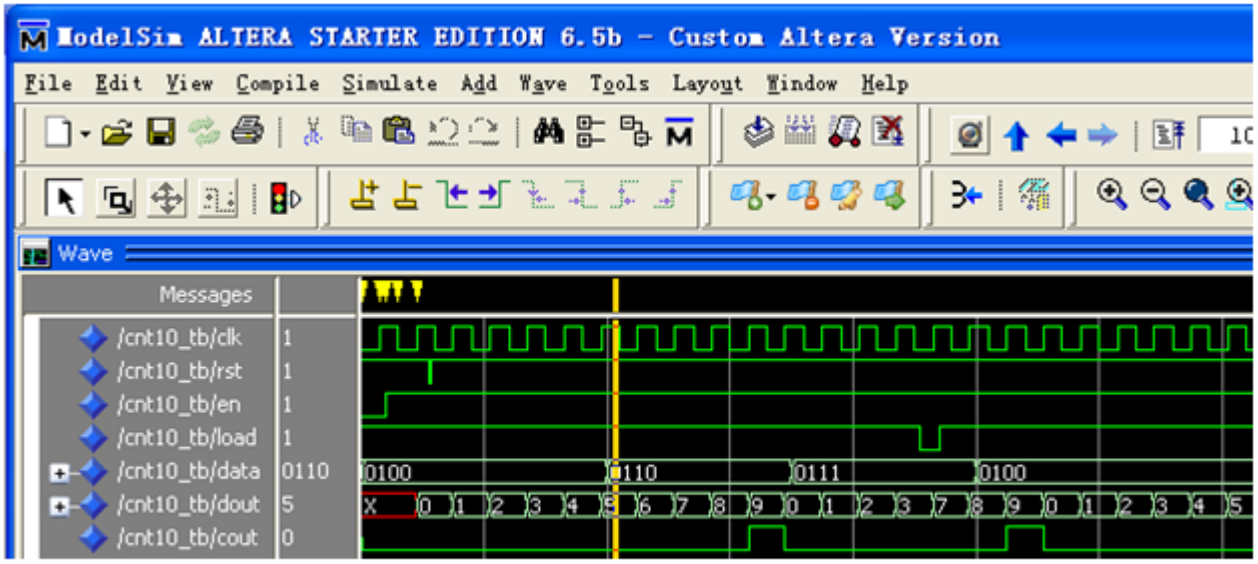

图 10-8　Test Bench 输出的仿真波形

10.4 VHDL子程序

10.4.1 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;
```

【例 10-2】

```
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
   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;
```

10.4 VHDL子程序

10.4.1 函数

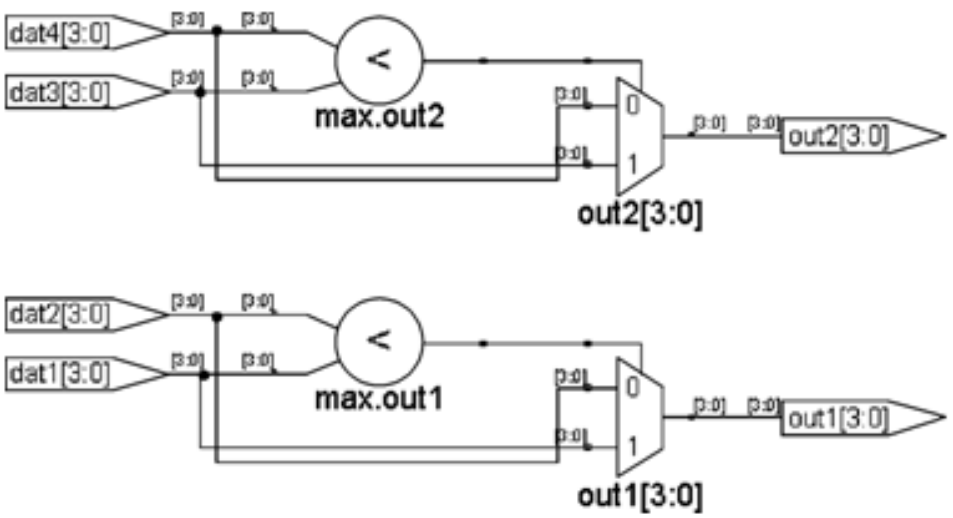

图 10-9　例 10-2 的逻辑电路图

10.4 VHDL子程序

10.4.1 函数

【例 10-3】

```
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 ;
```

10.4 VHDL子程序

10.4.2 重载函数

【例 10-4】

```
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
```

接下页

```
 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
   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
   -- 以下是调用重载函数 max 的程序:
LIBRARY IEEE ;
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;
```

【例 10-5】

```
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  ;
--  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 ;
```

10.4 VHDL子程序

10.4.3 决断函数

10.4.4 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);
```

10.4 VHDL子程序

10.4.4 过程

【例 10-6】

```
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 ;
```

10.4 VHDL子程序

10.4.4 过程

【例 10-7】

```
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  ;
```

【例 10-8】

```
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;
```

10.4 VHDL子程序

10.4.5 重载过程

【例 10-9】

```
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
...
```

10.4 VHDL子程序

10.4.6 子程序调用语句

1. 过程调用

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

【例 10-10】

```
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;
-- swap 的形参名为 data、low、high
     low, high : IN INTEGER) IS
    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 ;
```

【例 10-11】

```
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;
```

10.4 VHDL子程序

10.4.6 子程序调用语句

2. 函数调用

10.4 VHDL子程序

10.4.7 RETURN语句

```
RETURN;                    -- 第一种语句格式
RETURN 表达式;             -- 第二种语句格式
```

【例 10-12】

```
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 ;
```

10.4 VHDL子程序

10.4.7 RETURN语句

【例 10-13】

```
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 ;
```

10.4 VHDL子程序

10.4.8 并行过程调用语句

【例 10-14】

```
 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) ;                                        -- 进程语句执行
   BEGIN
   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
```

10.4 VHDL子程序

10.4.8 并行过程调用语句

【例 10-15】

```
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;
```

10.4 VHDL子程序

10.4.8 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;
```

10.5 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;
```

10.5 VHDL程序包

【例 10-16】

```
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
```

10.5 VHDL程序包

【例 10-17】

```
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: 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 ;
```