# EDA技术实用教程

# 第3章

# 组合电路的VHDL设计

# 3.1多路选择器的VHDL描述

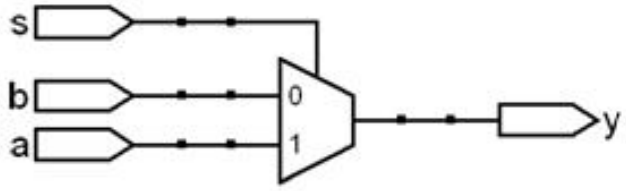



图 3-1　mux21a 实体

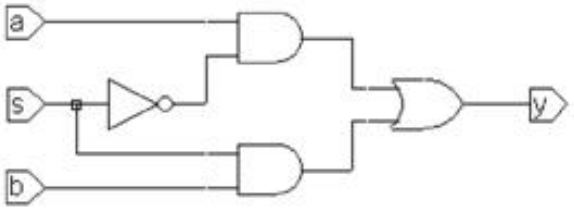



图 3-2　mux21a 结构体

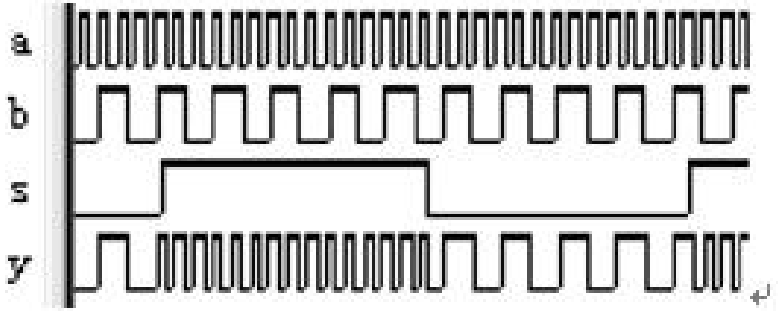



图 3-3　mux21a 电路的时序波形

# 3.1多路选择器的VHDL描述

【例 3-1】

```
ENTITY mux21a IS
    PORT ( a, b, s : IN  BIT;
                   y : OUT BIT  );
END ENTITY mux21a;
  ARCHITECTURE bhv OF mux21a IS
     BEGIN
       PROCESS (a,b,s)    --进程语句起始
       BEGIN
           IF (s='1')   THEN   y<=a ;   ELSE   y<=b;
           END IF;
       END PROCESS;
END ARCHITECTURE bhv ;
```

# 3.1多路选择器的VHDL描述

1．条件语句

```
IF a THEN ...
IF (s1='0')AND(s2='1')OR(c<b+1) THEN ...
```

2．数据类型

3．进程语句和顺序语句

# 3.1多路选择器的VHDL描述

4．端口语句和端口信号名

5．端口模式

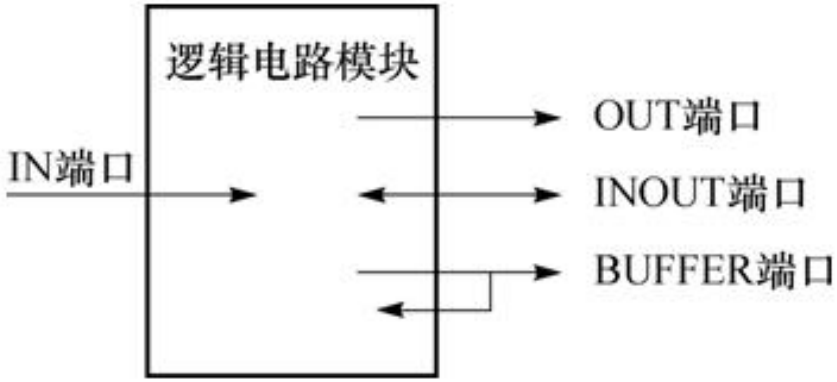



图 3-4　四种类型的端口模型图

6．关键字

7．标识符

# 3.2 半加器的VHDL描述

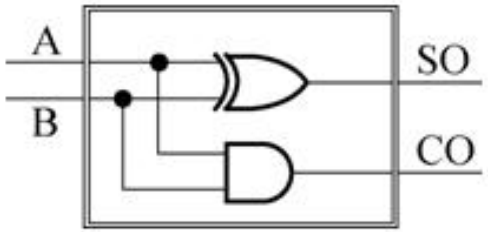



图 3-5　半加器的电路结构

| A | B | SO | CO |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

图 3-6　半加器的真值表

$$SO = A \oplus B ; \qquad CO = A \cdot B$$

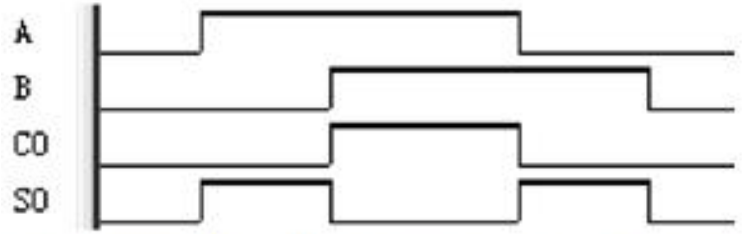



图 3-7　半加器的仿真功能波形图

# 3.2 半加器的VHDL描述

【例 3-2】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY  h_adder  IS
    PORT (
          A   : IN    STD_LOGIC;
          B   : IN    STD_LOGIC;
          SO  : OUT   STD_LOGIC;
          CO  : OUT   STD_LOGIC
           );
END ENTITY  h_adder ;
ARCHITECTURE  fh1 of h_adder IS
    BEGIN

        SO <= A XOR B ;

        CO <= A AND B ;

END ARCHITECTURE fh1;
```

设计库和程序包调用

电路模块端口说明和定义

VHDL实体描述部分

电路模块功能描述

VHDL结构体描述部分

VHDL表述的半加器完整电路模块程序代码

# 3.2 半加器的VHDL描述

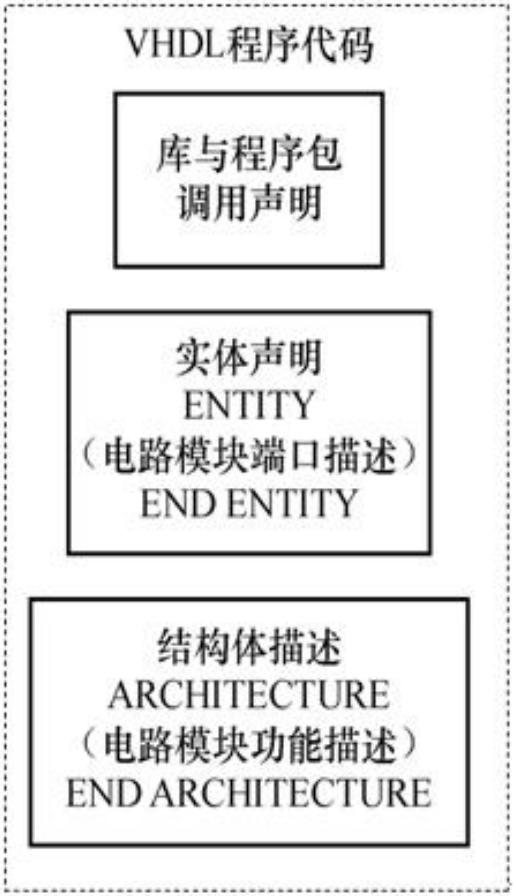



图 3-8 VHDL 程序结构

# 3.2 半加器的VHDL描述

1．实体表达与实体名

```
ENTITY  e_name  IS
    PORT ( p_name :  port_m   data_type;
           ...
           p_namei : port_mi   data_type );
END ENTITY e_name;
```

2．结构体表达

```
ARCHITECTURE arch_name OF e_name IS
  [说明语句]
BEGIN
  (功能描述语句)
END ARCHITECTURE arch_name ;
```

# 3.2 半加器的VHDL描述

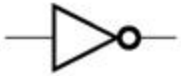
### 3．标准逻辑位数据类型STD_LOGIC

```
TYPE BIT IS('0','1');  --BIT类型只有两种取值
TYPE STD_LOGIC IS ('U','X','0','1','Z','W','L','H','-'); --有九种取值
```

### 4．赋值符号和逻辑操作符

目标变量名 <= 驱动表达式;

表 3-1　VHDL 逻辑操作符

| 逻辑操作符 | 逻辑图形 | 逻辑功能 | 逻辑操作符 | 逻辑图形 | 逻辑功能 |
|---|---|---|---|---|---|
| AND |  | 逻辑与操作 | NAND |  | 逻辑与非操作 |
| OR |  | 逻辑或操作 | NOR |  | 逻辑或非操作 |
| XOR |  | 逻辑异或操作 | XNOR |  | 逻辑同或操作 |
| | | | NOT |  | 逻辑取非操作 |

# 3.2 半加器的VHDL描述

5. 设计库和标准程序包

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

```
LIBRARY  <设计库名>;
USE  < 设计库名>.<程序包名>.ALL ;
```

```
LIBRARY  IEEE ;
USE  IEEE.STD_LOGIC_1164.ALL ;
```

6. 文件取名和存盘

7. 规范的程序书写格式

# 3.3 4选1多路选择器的VHDL描述

## 3.3.1 基于CASE语句的4选1多路选择器表述

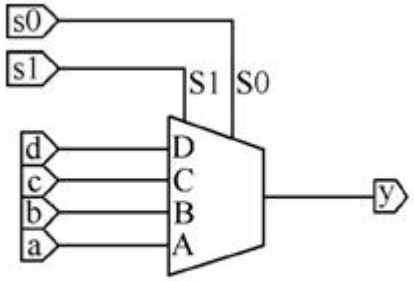



图 3-9　4 选 1 多路选择器

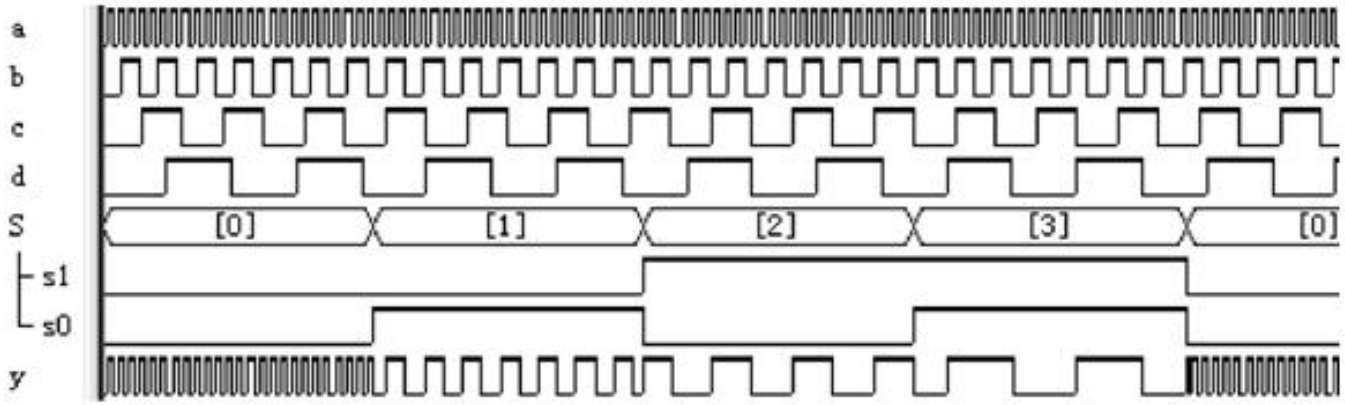



图 3-10　4 选 1 多路选择器 MUX41a 的时序波形

# 3.3 4选1多路选择器的VHDL描述

## 3.3.1 基于CASE语句的4选1多路选择器表述

【例 3-3】

```
LIBRARY  IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX41A IS
PORT (a, b,c,d,s0,s1 : IN  STD_LOGIC;     y  : OUT STD_LOGIC);
END ENTITY MUX41A;
ARCHITECTURE BHV OF MUX41A  is
SIGNAL S : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
S <= s1 & s0 ;
PROCESS(s1,s0)    BEGIN  --敏感信号表中可以放 s1、s0，也可直接放 S，如（S）
    CASE (S) IS
     WHEN "00" => y<=a ;
     WHEN "01" => y<=b ;
     WHEN "10" => y<=c ;
     WHEN "11" => y<=d ;
     WHEN OTHERS => NULL ;
    END CASE;
 END PROCESS;
END  BHV ;
```

# 3.3 4选1多路选择器的VHDL描述

## 3.3.2 CASE语句

```
CASE <表达式> IS
When <选择值或标识符> => <顺序语句>; ... ; <顺序语句> ;
When <选择值或标识符> => <顺序语句>; ... ; <顺序语句> ;
...
WHEN OTHERS => <顺序语句>;
END CASE ;
```

选择值 [ |选择值 ]

# 3.3 4选1多路选择器的VHDL描述

## 3.3.2 CASE语句

【例 3-4】

```
sel : IN INTEGER RANGE 0 TO 15 ;
  ...
 CASE sel IS
  WHEN   0      => z1 <= "010" ;    -- 当sel=0时选中
  WHEN  1|3     => z2 <= "110" ;    -- 当sel为1或3时选中
  WHEN 4 To 7|2 => z3 <="011";      -- 当sel为2、4、5、6或7时选中
  WHEN OTHERS => z4<= "111" ;       -- 当sel为8～15中任一值时选中
END CASE ;
```

# 3.3 4选1多路选择器的VHDL描述

## 3.3.2 CASE语句

【例 3-5】

```
SIGNAL value : INTEGER RANGE 0 TO 15;
SIGNAL  out1 : STD LOGIC ;
   ...
CASE value IS
   WHEN 0 => out1<= '1' ;            -- value2~15的值未包括进去，
   WHEN 1 => out1<= '0' ;            -- 除非加了WHEN OTHERS语句
END CASE
   ...
CASE value IS
   WHEN 0 TO 10 => out1<= '1';        -- 选择值中5~10的值有重叠
   WHEN 5 TO 15 => out1<= '0';
END CASE;
```

# 3.3 4选1多路选择器的VHDL描述

### 3.3.3 IEEE库预定义标准逻辑位与矢量

```
        B : OUT  STD LOGIC VECTOR(7 DOWNTO 0);     --定义端口信号
SIGNAL A : STD_LOGIC_VECTOR(1 TO 4);               --定义内部信号
```

```
B <= "01100010" ;              -- 可以对B赋值8位二进制数"01100010"
B（4 DOWNTO 1）<= "1101" ;     -- 赋值后，其中的B(4)为 '1'
B（7 DOWNTO 4）<= A ;          -- 其中 B(6)等于 A(2)，B(7)等于 A(1)
```

```
SIGNAL C : BIT_VECTOR(3 DOWNTO 0);
```

```
TYPE STD_LOGIC_VECTOR IS ARRAY ( NATURAL RANGE <> )OF STD_LOGIC;
```

# 3.3 4选1多路选择器的VHDL描述

### 3.3.4 其他预定义标准数据类型

```
LIBRARY IEEE ;
USE IEEE.STD_LOIGC_ARITH.ALL ;
```

（1）无符号数据类型。

```
UNSIGNED'("1000")
```

```
VARIABLE var : UNSIGNED(0 TO 10);
SIGNAL   sig : UNSIGNED(5 TO 0);
```

# 3.3 4选1多路选择器的VHDL描述

## 3.3.4 其他预定义标准数据类型

（2）有符号数据类型。

SIGNED'("0101")代表 +5，5
SIGNED'("1011")代表 -5

```
VARIABLE var : SIGNED(0 TO 10);
```

```
TYPE UNSIGNED IS ARRAY ( NATURAL RANGE <> )OF STD LOGIC;
TYPE   SIGNED IS ARRAY ( NATURAL RANGE <> )OF STD_LOGIC;
```

# 3.3 4选1多路选择器的VHDL描述

## 3.3.5 信号定义和数据对象

```
SIGNAL e : STD_LOGIC;
```

## 3.3.6 并置操作符&

```
SIGNAL a : STD_LOGIC_VECTOR（3 DOWNTO 0）;--定义a为4元素标准矢量
SIGNAL d : STD_LOGIC_VECTOR（1 DOWNTO 0）;--定义d为2元素标准矢量
    ...
    a  <= '1' & '0' & d（1）& '1'  ; --元素与数值并置，并置后的数组长度为4
    ...
   IF （a & d = "101011"）  THEN ... -- 在IF条件句中可以使用并置符
```

# 3.3 4选1多路选择器的VHDL描述

## 3.3.7 4选1多路选择器的VHDL不同描述方式

【例 3-6】

```
 S <= s1 & s0 ;
PROCESS(s1,s0)  BEGIN
   IF (S="00") THEN y<=a;
ELSIF (S="01") THEN y<=b;
ELSIF (S="10") THEN y<=c;
ELSE y<=d;   END IF;
END PROCESS;
END  BHV ;
```

【例 3-7】

```
S<=s1 & s0 ;
y<=a  WHEN S="00" ELSE
   b  WHEN S="01" ELSE
   c  WHEN S="10" ELSE
   d  ;
END  BHV
```

【例 3-8】

```
S <= s1 & s0 ;
WITH S SELECT
y<=a  WHEN "00",
   b  WHEN "01",
   c  WHEN "10",
   d  WHEN "11";
END  BHV ;
```

IF_THEN_ELSIF_ELSE_END IF

# 3.3 4选1多路选择器的VHDL描述

## 1. WHEN_ELSE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 ;
```

```
z  <= a WHEN p1 = '1' ELSE
      b WHEN p2 = '1' ELSE
      c  ;
```

# 3.3 4选1多路选择器的VHDL描述

2．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;

UNAFFECTED   WHEN OTHERS ;
```

# 3.4 全加器及其VHDL表述

## 3.4.1 全加器设计及例化语句应用

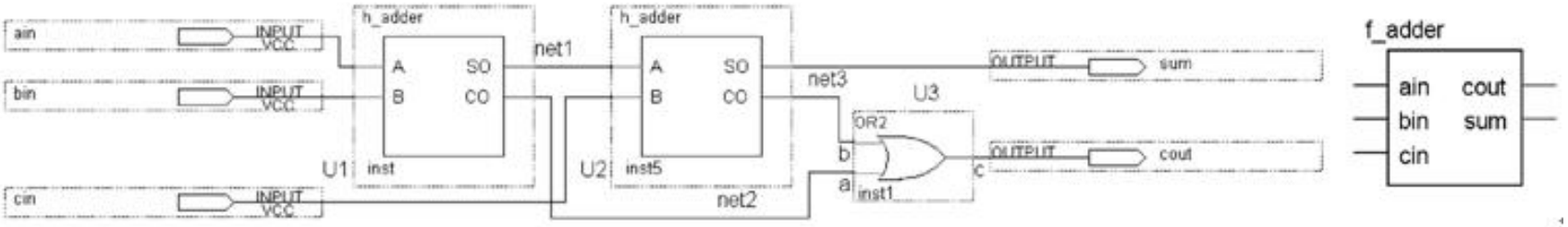



图 3-11　全加器 f_adder 电路图及其实体模块 f_adder

# 3.4 全加器及其VHDL表述

## 3.4.1 全加器设计及例化语句应用

【例 3-9】

```
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);--例化语句
  u2 : h_adder PORT MAP(net1, cin, net3, sum);
  u3 :   or2a   PORT MAP(a=>net2,  b=>net3,   c=>cout);
END ARCHITECTURE fd1;
```

# 3.4 全加器及其VHDL表述

## 3.4.1 全加器设计及例化语句应用

【例 3-10】

```
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 ;
```

# 3.4 全加器及其VHDL表述

## 3.4.2 VHDL例化语句

```
COMPONENT 元件名 IS
        PORT （端口名表）;
END COMPONENT 文件名;
```

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

```
例化名 : 元件名 PORT MAP（ [端口名 =>] 连接端口名,...）;
```

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

# 3.4 全加器及其VHDL表述

## 3.4.3 8位加法器设计及算术操作符应用

【例 3-11】

```
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;
```

# 3.4 全加器及其VHDL表述

## 3.4.3 8位加法器设计及算术操作符应用

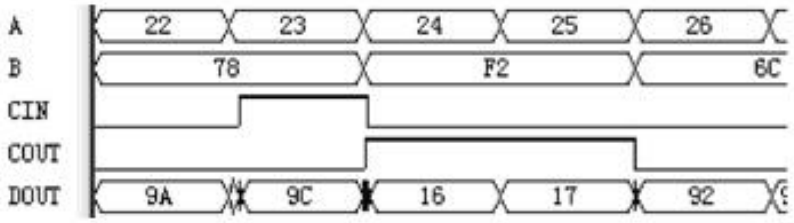



图 3-12　8 位加法器仿真波形

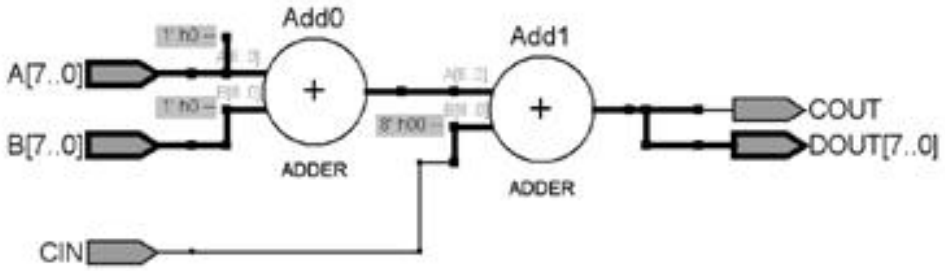



图 3-13　8 位加法器 Quartus II 综合之 RTL 电路

# 3.5 乘法器的VHDL表述

## 3.5.1 统计位矢中含 '1' 个数的电路模块设计

【例 3-12】

```
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;
```

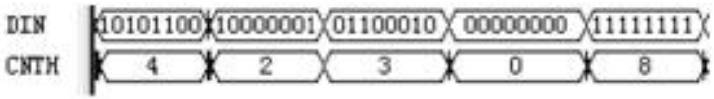



图 3-14　例 3-12 的仿真波形

# 3.5 乘法器的VHDL表述

## 3.5.2 FOR_LOOP循环语句用法

（1）单个LOOP语句

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

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

# 3.5 乘法器的VHDL表述

## 3.5.2 FOR_LOOP循环语句用法

### （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];
```

# 3.5 乘法器的VHDL表述

## 3.5.3 移位相加型乘法器的VHDL表述方法

【例 3-13】

```
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;
 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 3-15　例 3-13 乘法器的仿真波形

# 3.5 乘法器的VHDL表述

## 3.5.4 GENERIC参数定义语句

```
GENERIC( 常数名 : 数据类型 [ : 设定值 ]
 { ;常数名 : 数据类型 [ : 设定值 ] } ) ;
```

# 3.5 乘法器的VHDL表述

## 3.5.5 整数数据类型

```
SIGNAL Q : INTEGER RANGE 15 DOWNTO 0;
```

| | |
|---|---|
| 1, 35 | 十进制整数1和35 |
| 10E3 | 十进制整数，等于十进制整数1000 |
| 16#D9# | 十六进制整数，等于十六进制数D9H |
| 8#720# | 八进制整数，等于八进制数720O |
| 2#11010010# | 二进制整数，等于二进制数 11010010B |

```
Q : BUFFER NATURAL RANGE 15 DOWNTO 0;
```

# 3.5 乘法器的VHDL表述

## 3.5.6 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)  ;
```

# 3.5 乘法器的VHDL表述

## 3.5.7 移位操作符

标识符　移位操作符　移位位数 ；　　--如 "10110001" SRL 3，结果是"00010110"

【例 3-14】

```
LIBRARY IEEE;
USE IEEE.STD LOGIC 1164.ALL;
USE IEEE.STD LOGIC UNSIGNED.ALL; --为使用类型转换函数，打开此程序包。
ENTITY decoder3to8 IS
   port (DIN  : IN STD LOGIC VECTOR (2 DOWNTO 0);
         DOUT : OUT BIT VECTOR (7 DOWNTO 0));
END decoder3to8;
ARCHITECTURE behave OF decoder3to8 IS
BEGIN
    DOUT<="00000001" SLL CONV INTEGER(DIN);    --被移位部分是常数
END behave;
```

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

【例 3-15】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY COMP IS
 PORT(C,D : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
      RA  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      RM  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        R : OUT BOOLEAN );
END ENTITY  COMP;
ARCHITECTURE ONE OF COMP IS
 BEGIN
   R<= (C>D); RA <=C+D; RM<= C*D;
END ARCHITECTURE  ONE;
```

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

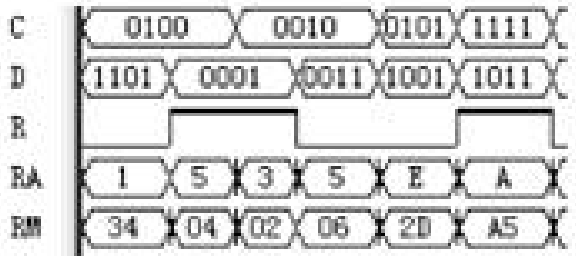



图 3-16　例 3-15 的仿真波形

【例 3-16】

```
...  --上下其余部分与例 3-15 相同
USE IEEE.STD_LOGIC_SIGNED.ALL ;
```

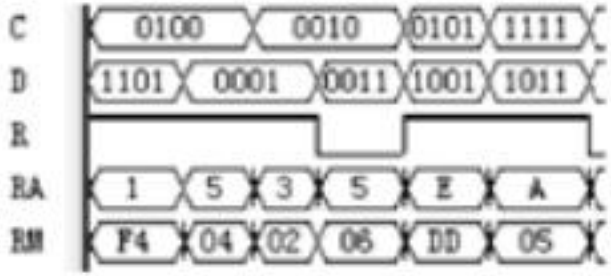



图 3-17　例 3-16 的仿真波形

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

【例 3-17】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
ENTITY COMP IS
   PORT (  C,D : IN  UNSIGNED(3 DOWNTO 0);
           RA  : OUT UNSIGNED(3 DOWNTO 0);
           RM  : OUT UNSIGNED(7 DOWNTO 0);
            R : OUT BOOLEAN );
END ENTITY  COMP;
ARCHITECTURE ONE OF COMP IS
 BEGIN
    R<= (C>D); RA <=C+D; RM<= C*D;
END ARCHITECTURE  ONE;
```

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

【例 3-18】

```
...    --上下其余部分与例3-17相同
PORT (  C,D : IN  SIGNED(3 DOWNTO 0);
        RA  : OUT SIGNED(3 DOWNTO 0);
        RM  : OUT SIGNED(7 DOWNTO 0);
         R  : OUT BOOLEAN );
```

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

表 3-2　VHDL 操作符列表给出了不同数据类型定义和用法

| 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 | 554x323 | 整数 |
| | SLL | 逻辑左移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | SRL | 逻辑右移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | SLA | 算术左移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | SRA | 算术右移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | ROL | 逻辑循环左移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | ROR | 逻辑循环右移 | BIT、BIT_VECTOR 或布尔型一维数组 |
| | ** | 乘方 | 整数 |
| | ABS | 取绝对值 | 整数 |

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

| | | | |
|---|---|---|---|
| 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 |

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

表 3-3　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 |

# 3.5 乘法器的VHDL表述

## 3.5.8 各类运算操作对数据类型的要求

【例 3-19】

```
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 ;
...
d<=e OR f OR g ;              -- 两个操作符OR相同，不需括号
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，因而不能相互作用，表达错误
```

# 3.5 乘法器的VHDL表述

## 3.5.9 数据类型转换函数

表 3-4 IEEE 库数据类型转换函数表

| 函 数 名 | 功 能 |
|---|---|
| 所在程序包： STD_LOGIC_1164 | |
| TO_STDLOGICVECTOR(A) | 由 BIT_VECTOR 类型转换为 STD_LOGIC_VECTOR |
| TO_BITVECTOR(A) | 由 STD_LOGIC_VECTOR 转换为 BIT_VECTOR |
| TO_STDLOGIC (A) | 由 BIT 转换成 STD_LOGIC |
| TO_BIT(A) | 由 STD_LOGIC 类型转换成 BIT 类型 |
| 所在程序包： STD_LOGIC_ARITH | |
| CONV_STD_LOGIC_VECTOR(A, 位长) | 将 INTEGER 转换成 STD_LOGIC_VECTOR 类型，A 是整数 |
| CONV_INTEGER(A) | 将 STD_LOGIC_VECTOR 转换成 INTEGER |
| CONV_UNSIGNED(A, 位长) | 将 UNSIGNED，SIGNED，INTEGER 类型转换为指定位长的 UNSIGNED 类型 |
| CONV_SIGNED(A, 位长) | 将 UNSIGNED，SIGNED，INTEGER 类型转换为指定位长的 SIGNED 类型 |
| 所在程序包： STD_LOGIC_UNSIGNED | |
| CONV_INTEGER(A) | 由 STD_LOGIC_VECTOR 转换成 INTEGER |

# 3.5 乘法器的VHDL表述

## 3.5.9 数据类型转换函数

```
FUNCTION TO_STDLOGICVECTOR( s : BIT_VECTOR)
RETURN STD_LOGIC_VECTOR;
```

【例 3-20】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY  amp IS
   PORT (    a1，a2 : IN BIT_VECTOR(3 DOWNTO 0);
          c1，c2,c3 : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          b1，b2，b3 : INTEGER RANGE 0 TO 15;
        d1,d2,d3,d4 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) );
END amp;
 d1 <= TO_STDLOGICVECTOR(a1 AND a2);                          --(1)
 d2 <= CONV_STD_LOGIC_VECTOR(b1,4) WHEN CONV_INTEGER(b2)=9
       else   CONV_STD_LOGIC_VECTOR(b3,4);                   --(2)
 d3 <= c1 WHEN CONV_INTEGER(c2)= 8 ELSE c3;                  --(3)
 d4 <= c1 WHEN  c2 = 8 else c3;                              --(4)
```

# 3.5 乘法器的VHDL表述

## 3.5.9 数据类型转换函数

【例 3-21】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY decoder3to8 IS
 PORT (  input: IN STD_LOGIC_VECTOR (2 DOWNTO 0);
         output: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END decoder3to8;
ARCHITECTURE behave OF decoder3to8 IS
  BEGIN
    PROCESS (input)
    BEGIN
        Output <= (OTHERS =>'0');
        output(CONV_INTEGER(input)) <= '1';
    END PROCESS;
END behave;
```

# 3.5 乘法器的VHDL表述

## 3.5.9 数据类型转换函数

```
FUNCTION To bitvector ( s : std logic vector;
                        xmap : BIT := '0' )
                        RETURN BIT VECTOR IS
  ALIAS sv : std logic vector(s'LENGTH-1 DOWNTO 0 )IS s ;
  VARIABLE result : BIT VECTOR(s'LENGTH-1 DOWNTO 0 );
BEGIN
  FOR i IN result'RANGE LOOP
     CASE sv(i)IS
        WHEN '0'|'L' => result(i):= '0';
        WHEN '1'|'H' => result(i):= '1';
        WHEN OTHERS => result(i):= xmap;
     END CASE ;
  END LOOP ;
  RETURN result ;
END ;
```

# 3.5 乘法器的VHDL表述

## 3.5.9 数据类型转换函数

【例 3-23】

```
LIBRARY IEEE;                          --主程序，用户定义转换函数应用实例
USE IEEE.STD_LOGIC_1164.ALL;
USE WORK.n_pack.ALL ;
ENTITY  axamp IS
   PORT(dat : IN nat;                  --注意数据类型的定义
        ou  : OUT Bit8);               --注意数据类型的定义
  END;
ARCHITECTURE bhv OF axamp IS
    BEGIN
     ou <=  nat_to_Bit8(dat);
END;
```

# 3.5 乘法器的VHDL表述

【例 3-24】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
PACKAGE n_pack IS
  SUBTYPE nat IS Integer range 0 to 255;  --定义一个Integer的子类型
    TYPE Bit8 IS array (7 downto 0) OF std_logic; --定义一个数据类型
  FUNCTION nat_to_Bit8 (s: nat) RETURN Bit8;
END n_pack;
  PACKAGE BODY n_pack IS
  FUNCTION nat_to_Bit8 (s: nat) RETURN Bit8 IS
    VARIABLE Din: Integer range 255 downto 0;
    VARIABLE Rut: Bit8;
    VARIABLE Rig: Integer :=2**7;
 BEGIN
    Din := s;
  FOR I in 7 downto 0 LOOP
    IF Din/Rig > 0 THEN
        Rut(i) := '1';
          Din := Din-Rig;
      ELSE Rut (i):= '0';
    END IF;
       Rig := Rig / 2;
  END LOOP;
  RETURN Rut;
 END nat_to_Bit8;
END n_pack;
```

# 3.5 乘法器的VHDL表述

## 3.5.10 GENERIC参数传递映射语句

【例 3-25】

```
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;
```

例化名 : 元件名  GENERIC MAP(类属表)

# 习　题

**3-2** 画出与以下实体描述对应的原理图符号元件：

```
ENTITY buf3s IS                              -- 实体1:  三态缓冲器
    PORT （input  : IN STD LOGIC ;           -- 输入端
          enable  : IN STD LOGIC ;           -- 使能端
          output  : OUT STD LOGIC ）;        -- 输出端
END buf3x ;
ENTITY mux21 IS                              --实体2:   2选1多路选择器
    PORT （in0,  in1, sel :  IN STD LOGIC;
           output  : OUT STD_LOGIC）;
```

# 习　题

**3-4** 给出全减器的**VHDL**描述。要求：

（**1**）首先设计半减器，然后用例化语句将它们连接起来，图**3-18**中**h_suber**是半减器，**diff**是输出差，**s_out**是借位输出，**sub_in**是借位输入。

（**2**）根据图**3-18**设计全减器。以全减器为基本硬件，构成串行借位的**8**位减法器，要求用例化语句来完成此项设计（减法运算是 **x – y - sun_in = diffr**）。

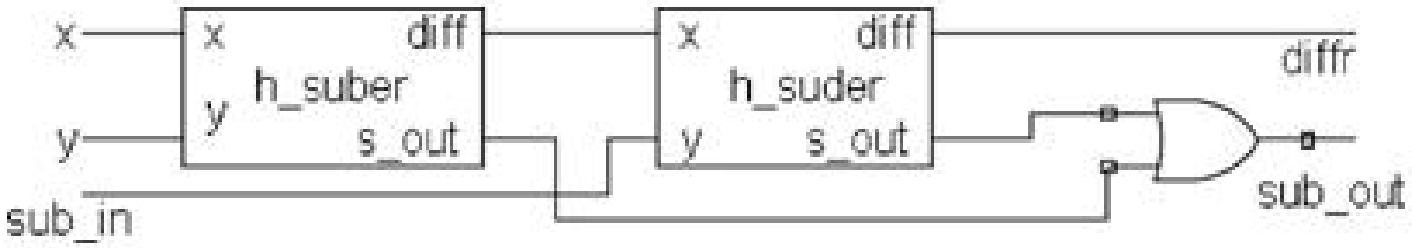



图 3-18　全减器结构图

# 习　题

**3-23** 根据图**3-19**，用两种不同描述方式设计一**4**选**1**多路选择器。在设计中需要体现此电路由三个**2**选**1**多路选择器构成。

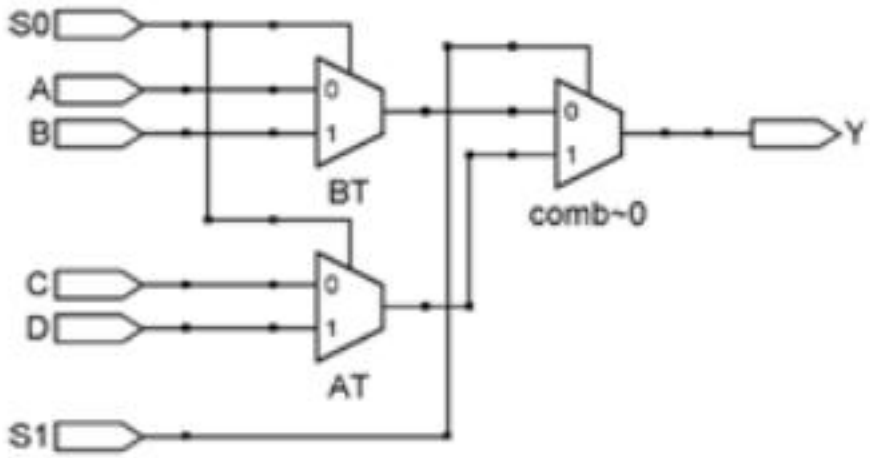



图 3-19　4 选 1 多路选择器 RTL 图