

第3章

VHDL数据类型与顺序语句

3.1 VHDL数据类型

3.1.1 BIT和BIT_VECTOR类型

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

```
TYPE BIT_VECTOR IS ARRAY(NATURAL RANGE<>) OF BIT; --属于非限制类数组类型
```

```
SIGNAL X, Y : BIT;
```

```
SIGNAL A, B : STD_LOGIC_VECTOR(3 DOWNTO 0)
```

```
...
```

```
X <= '1' ; -- 对 BIT 类型的信号 X 赋值 '1'
```

```
A <= "1101" ; --赋值后, A(3)、A(2)、A(1)、A(0)分别等于 '1'、'1'、'0'、'1'
```

```
B(2 DOWNTO 1) <= A(3 DOWNTO 2); --赋值后, B(2)=A(3), B(1)=A(2)
```

```
B(2 DOWNTO 0) <= X & Y & '1'; --赋值后, B(2)=x、B(1)=y, B(0)='1'
```

3.1 VHDL数据类型

3.1.2 STD_LOGIC和STD_LOGIC_VECTOR类型

```
TYPE STD_LOGIC IS ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'); --有九种取值
```

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

```
B : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); --元素排序从高到低
```

```
SIGNAL A : STD_LOGIC_VECTOR(1 TO 4);
```

```
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)
```

3.1 VHDL数据类型

3.1.3 整数类型INTEGER

```
SIGNAL Q : INTEGER RANGE 15 DOWNT0 0;
```

1	十进制整数
35	十进制整数
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 7200
2#11010010#	二进制整数，等于二进制数 11010010B

```
Q : BUFFER NATURAL RANGE 15 DOWNT0 0;
```

3.1 VHDL数据类型

3.1.4 布尔数据类型BOOLEAN

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

3.1.5 SIGNED和UNSIGNED类型

```
LIBRARY IEEE ;
```

```
USE IEEE.STD_LOGIC_ARITH.ALL ;
```

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

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

3.1 VHDL数据类型

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

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

SIGNED' ("0101") 代表 +5, 5

SIGNED' ("1011") 代表 -5

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

3.1 VHDL数据类型

【例 3-1】

```
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);
         A,B : IN SIGNED(3 DOWNTO 0);
         RCD : OUT UNSIGNED(3 DOWNTO 0);
         RAB : OUT SIGNED(3 DOWNTO 0);
         RM1 : OUT UNSIGNED(7 DOWNTO 0);
         RM2 : OUT SIGNED(7 DOWNTO 0);
         R1,R2 : OUT BOOLEAN );
END ENTITY COMP;
ARCHITECTURE ONE OF COMP IS
    BEGIN
        R1 <= (C>D); R2<=(A>B);
        RCD <= C+D; RAB <= A+B;
        RM1 <= C*D; RM2 <= A*B;
    END ARCHITECTURE ONE;
```

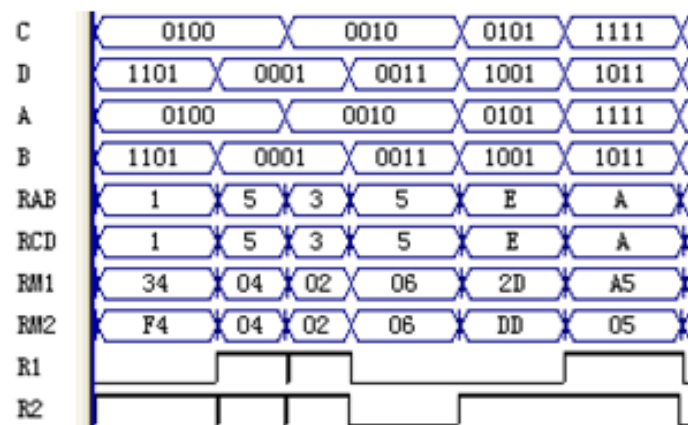


图 3-1 例 3-1 的仿真波形图

3.1 VHDL数据类型

3.1.6 其他预定义类型

1. 字符类型

2. 实数类型

1.0	十进制浮点数
0.0	十进制浮点数
65971.333333	十进制浮点数
65_971.333_3333	与上一行等价
8#43.6#e+4	八进制浮点数
43.6E-4	十进制浮点数

3. 字符串类型

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


3.1 VHDL数据类型

4. 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 ;
```

3.1 VHDL数据类型

5. 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);
```

3.1 VHDL数据类型

3.1.7 数据类型转换函数

表 3-1 可综合的数据类型归纳

数据类型	可综合的取值范围
bit, bit_vector	'1', '0'
std_logic, std_logic_vector	"X", "0", "1", "Z"
boolean	true, false
natural	0~2 147 483 647
integer	-2 147 483 647~+2 147 483 647
signed	-2 147 483 647~+2 147 483 647
unsigned	0~2 147 483 647
用户自定义类型	用户自定义数组或元素
数组类型 (array)	可综合数据类型的组合
子类型 (subtype)	数据类型的子集

3.1 VHDL数据类型

3.1.7 数据类型转换函数

表 3-2 IEEE 库类型转换函数表

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

3.1 VHDL数据类型

3.1.7 数据类型转换函数

【例 3-2】

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL; --为使用转换函数 to_stdlogicvector(A)调用此程序包
ENTITY exg IS
    PORT (a,b : in bit_vector(3 downto 0); --注意定义 a、b 的数据类型
          q   : out std_logic_vector(3 downto 0));
end ;
architecture rtl of exg is
begin
    q<= to_stdlogicvector(a and b);--将位矢量数据类型转换成标准逻辑位矢量数据
end;
```

3.1 VHDL数据类型

conv_std_logic_vector 和 conv_integer.

【例 3-3】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL; --注意本例中的两个转换函数定义于此程序包
ENTITY axamp IS
    PORT(a,b,c : IN integer range 0 to 15 ;
         q : OUT std_logic_vector(3 downto 0) );
END;
ARCHITECTURE bhv OF axamp IS
    BEGIN
        q <= conv_std_logic_vector(a,4) when conv_integer(c)=8
            else conv_std_logic_vector(b,4) ;
    END;
```

3.2 VHDL最常用的顺序语句

3.2.1 赋值语句

信号赋值语句

变量赋值语句

赋值目标

赋值符号

赋值源

3.2 VHDL最常用的顺序语句

3.2.2 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]

【例 3-4】

```
sel : IN INTEGER RANGE 0 TO 15 ;
...
CASE sel IS
  WHEN 0      => z1 <= "010" ; -- 当 sel=0 时选中，并执行对 z1 的赋值
  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 ;
```


3.2 VHDL最常用的顺序语句

3.2.2 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;
```

3.2 VHDL最常用的顺序语句

3.2.3 PROCESS语句

```
[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];
```

3.2.4 并置操作符 &

```
SIGNAL a : STD_LOGIC_VECTOR (3 DOWNTO 0); --首先定义 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
```

3.2 VHDL最常用的顺序语句

3.2.5 IF语句

<pre>IF 条件句 Then --类型1语句 顺序语句 END IF ;</pre>	<pre>IF 条件句 Then --类型2语句 顺序语句 ELSE 顺序语句 END IF ;</pre>
<pre>IF 条件句 Then --类型3语句 IF 条件句 Then ... END IF END IF</pre>	<pre>IF 条件句 Then --类型4语句 顺序语句 ELSIF 条件句 Then 顺序语句 ... ELSE 顺序语句 END IF</pre>

3.3 IF语句使用示例

3.3.1 D触发器的VHDL描述

【例 3-6】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF1 IS
    PORT (CLK,D : IN STD_LOGIC ;
          Q : OUT STD_LOGIC );
END ;
ARCHITECTURE bhv OF DFF1 IS
    SIGNAL Q1 : STD_LOGIC ;
BEGIN
    PROCESS (CLK,Q1)    BEGIN
        IF CLK'EVENT AND CLK='1' THEN Q1<=D ; END IF ;
    END PROCESS ;
    Q <= Q1 ;
END bhv;
```

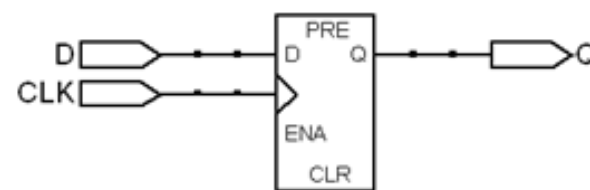


图 3-2 D 触发器模块图

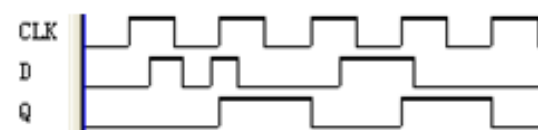


图 3-3 D 触发器时序波形

3.3 IF语句使用示例

1. 上升沿检测表达式和信号属性函数EVENT

<信号名>'EVENT

2. 不完整条件语句与时序电路

【例 3-7】

```
ENTITY COMP_BAD IS
    PORT( a, b : IN BIT;  q : OUT BIT  );
END ;
ARCHITECTURE one OF COMP_BAD IS
    BEGIN
CMP:  PROCESS (a,b) BEGIN          -- CMP 是当前进程的标号或名称, 不参与综合
        IF a>b THEN q<='1' ;
        ELSIF a<b THEN q<='0' ; END IF; -- 注意未提及当 a=b 时, q 作何操作
    END PROCESS ;
END ;
```

3.3 IF语句使用示例

1. 上升沿检测表达式和信号属性函数EVENT

<信号名>'EVENT

2. 不完整条件语句与时序电路

【例 3-7】

```
ENTITY COMP_BAD IS
  PORT( a, b : IN BIT;  q : OUT BIT
END ;
```

```
ARCHITECTURE one OF COMP_BAD IS
```

```
  BEGIN
```

```
  CMP: PROCESS (a,b) BEGIN      -- CMP 是当前进程的标号或名称, 不参与综合
    IF a>b THEN q<='1' ;
    ELSIF a<b THEN q<='0' ; END IF; -- 注意未提及当 a=b 时, q 作何操作
  END PROCESS ;
END ;
```

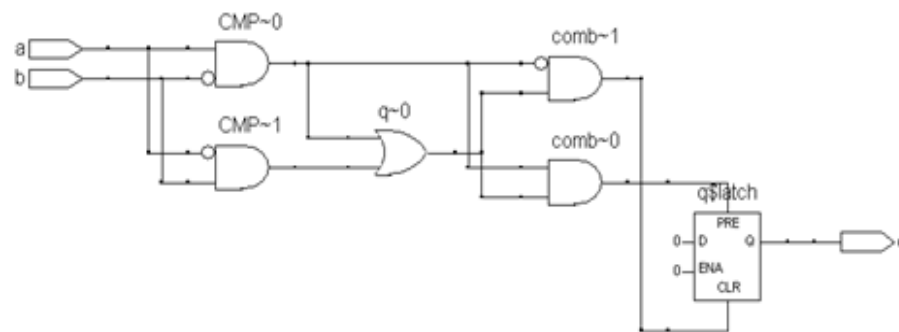


图 3-4 例 3-7 综合后的 RTL 电路图

3.3 IF语句使用示例

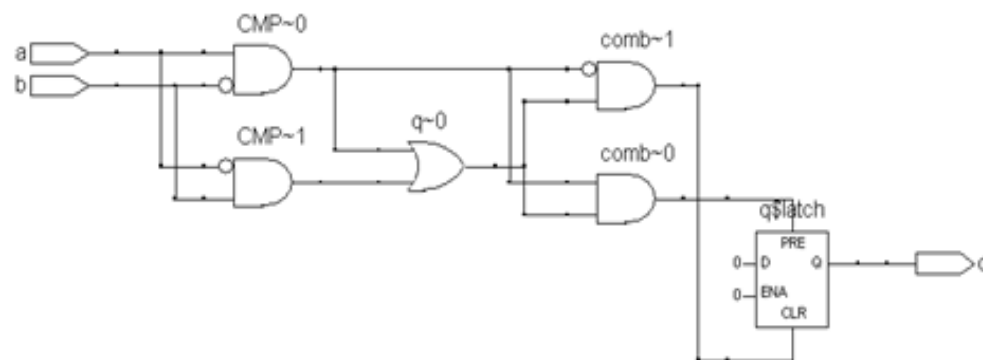


图 3-4 例 3-7 综合后的 RTL 电路图

【例 3-8】

```
IF a>b THEN q<='1' ; ELSE q<='0' ; END IF;
```



图 3-5 例 3-8 的 RTL 电路图

3.3 IF语句使用示例

3.3.2 含异步复位和时钟使能的D触发器的VHDL描述

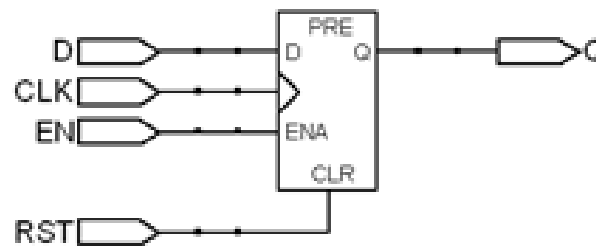


图 3-6 含使能和复位的 D 触发器

3.3 IF语句使用示例

【例 3-9】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF2 IS
    PORT (CLK,RST,EN,D : IN STD_LOGIC; Q : OUT STD_LOGIC );
    END ;
    ARCHITECTURE bhv OF DFF2 IS
        SIGNAL Q1 : STD_LOGIC ;
    BEGIN
        PROCESS (CLK,Q1,RST,EN) BEGIN
            IF RST='1' THEN Q1<='0';
                ELSIF CLK'EVENT AND CLK='1' THEN
                    IF EN='1' THEN Q1<=D; END IF;
                END IF;
            END PROCESS ;
            Q <= Q1 ;
        END bhv;
```

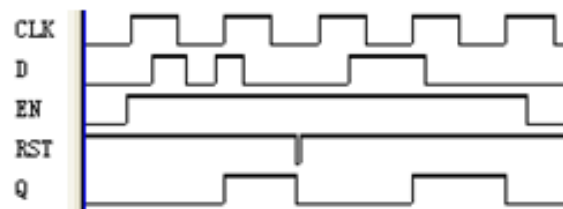


图 3-7 例 3-9 的时序图

3.3 IF语句使用示例

3.3.3 基本锁存器的VHDL描述

【例 3-10】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY LTCH2 IS
    PORT (CLK,D : IN STD_LOGIC;   Q : OUT STD_LOGIC);
END ;
ARCHITECTURE bhv OF LTCH2 IS
    BEGIN
        PROCESS (CLK, D)    BEGIN
            IF CLK='1' THEN Q <= D; END IF;
        END PROCESS ;
END bhv;
```

3.3 IF语句使用示例

3.3.3 基本锁存器的VHDL描述

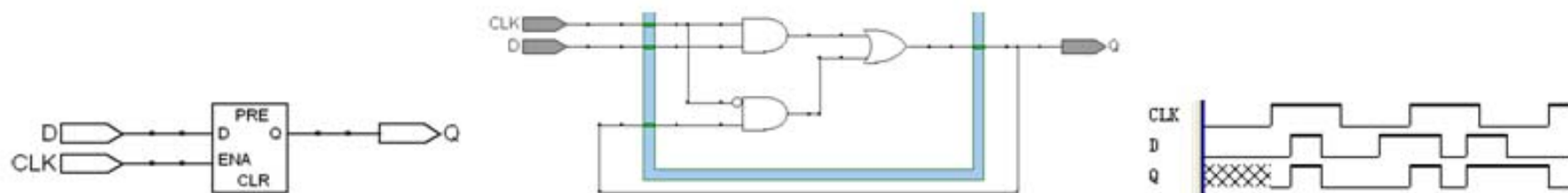


图 3-8 基本锁存器模块、内部电路结构以及锁存器时序波形图

【例 3-11】

```
PROCESS (CLK)      BEGIN
    IF CLK='1' THEN Q <= D; END IF;
END PROCESS ;
```

3.3 IF语句使用示例

3.3.4 含清0控制的锁存器的VHDL描述

【例 3-12】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY LTCH3 IS
    PORT (CLK,D,RST : IN STD_LOGIC;
          Q : OUT STD_LOGIC );
END ;
ARCHITECTURE bhv OF LTCH3 IS
    BEGIN
        PROCESS (CLK,D,RST)    BEGIN
            IF RST='1' THEN Q<='0';
                ELSIF CLK = '1' THEN Q <= D;
            END IF;
        END PROCESS ;
END bhv;
```

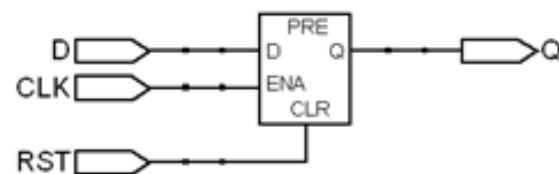


图 3-9 含异步清 0 的锁存器

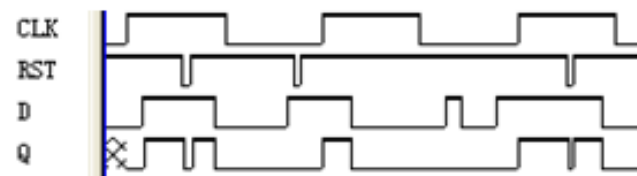


图 3-10 锁存器的仿真波

3.3 IF语句使用示例

3.3.5 VHDL实现时序电路的不同表述方式

【例 3-13】

```
IF (CLK'EVENT AND CLK='1') AND (CLK'LAST_VALUE='0')
    THEN Q <= D ;    --确保 CLK 的变化是一次上升沿的跳变
END IF;
```

【例 3-14】

```
IF CLK='1' AND CLK'LAST_VALUE ='0' THEN Q <= D; END IF;
```

【例 3-15】

```
IF rising_edge(CLK) -- 注意使用此函数必须打开 STD_LOGIC_1164 程序包
    THEN Q1 <= D ;
END IF;
```

3.3 IF语句使用示例

3.3.5 VHDL实现时序电路的不同表述方式

【例 3-16】

```
WREG: PROCESS      BEGIN
    wait until CLK = '1' ;    --利用 wait 语句
    Q <= D ;
END PROCESS;
```

```
G1 : BLOCK (CLK'EVENT AND clk='1') begin
    q<=GUARDED d;    END BLOCK G1;
```

3.3 IF语句使用示例

3.3.6 4位二进制加法计数器设计

【例 3-17】

```
ENTITY CNT4 IS
  PORT ( CLK : IN BIT; Q : BUFFER INTEGER RANGE 15 DOWNTO 0 );
  END ;
ARCHITECTURE bhv OF CNT4 IS
  BEGIN
    PROCESS (CLK) BEGIN
      IF CLK'EVENT AND CLK = '1' THEN Q<=Q+1 ; END IF;
    END PROCESS ;
  END bhv;
```

3.3 IF语句使用示例

3.3.7 计数器更常用的VHDL表达方式

【例 3-18】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY CNT4 IS
PORT (CLK : IN STD_LOGIC; Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END ;
ARCHITECTURE bhv OF CNT4 IS
    SIGNAL Q1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
    PROCESS(CLK) BEGIN
        IF CLK'EVENT AND CLK = '1' THEN Q1<=Q1+1; END IF;
    END PROCESS ;
    Q <= Q1 ;
END bhv;
```


3.3 IF语句使用示例

3.3.7 计数器更常用的VHDL表达方式

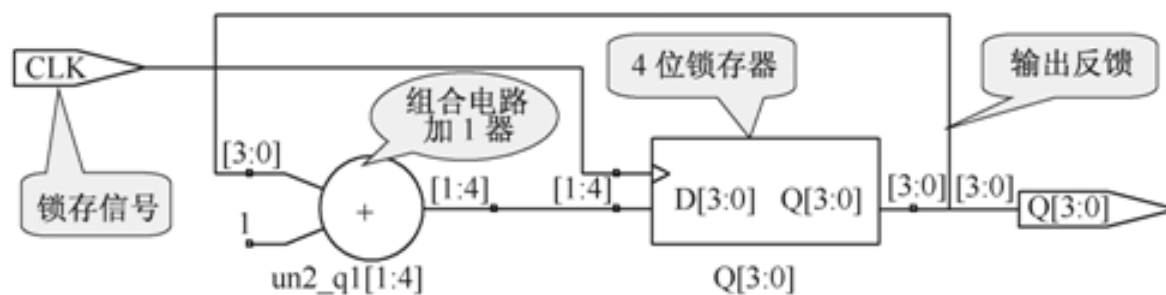


图 3-11 4 位加法计数器 RTL 电路

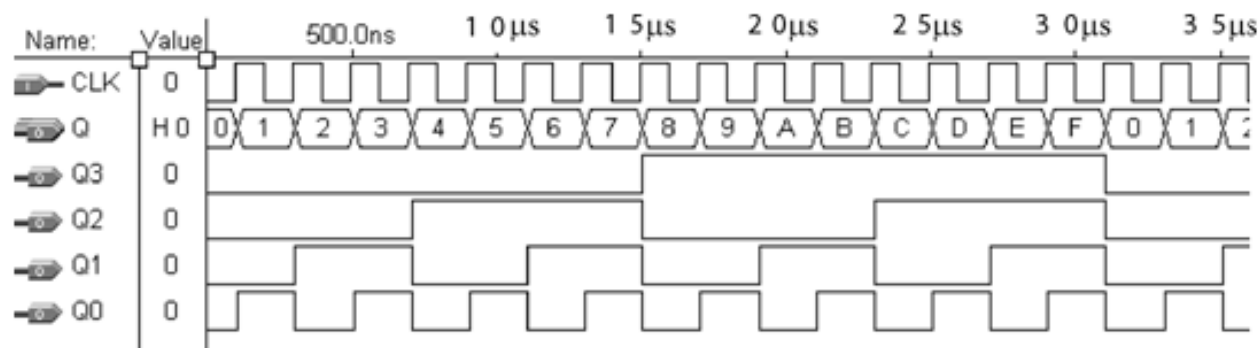


图 3-12 4 位加法计数器工作时序

3.3.8 实用计数器的VHDL设计

【例 3-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT10 IS
    PORT (CLK,RST,EN,LOAD : IN STD_LOGIC;
          DATA : IN STD_LOGIC_VECTOR(3 DOWNTO 0); --4 位预置数
          DOUT : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); --计数值输出
          COUT : OUT STD_LOGIC); --计数进位输出
END CNT10;
ARCHITECTURE behav OF CNT10 IS
BEGIN
    PROCESS (CLK, RST, EN, LOAD)
        VARIABLE Q : STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
        IF RST='0' THEN Q := (OTHERS=>'0'); --复位低电平时，计数寄存器清 0
        ELSIF CLK'EVENT AND CLK='1' THEN --测试时钟上升沿
            IF EN='1' THEN --计数使能高电平，允许计数
                IF (LOAD='0') THEN Q := DATA; ELSE --预置控制低电平，允许加载
                    IF Q<9 THEN Q := Q + 1; --计数小于 9，继续累加
                    ELSE Q := (OTHERS=>'0'); END IF; --否则计数清 0
                END IF;
            END IF;
        END IF;
    END IF;
    DOUT <= Q; --计数寄存器的值输出端口
END PROCESS;
END behav;
```

3.3 IF语句使用示例

3.3.8 实用计数器的VHDL设计

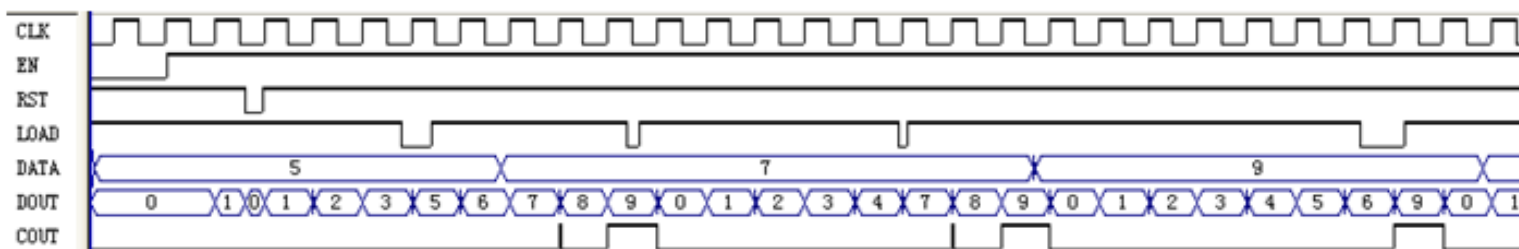


图 3-13 例 3-19 的工作时序 (目标器件是 Cyclone 系列 EP1C3)

3.3 IF语句使用示例

3.3.8 实用计数器的VHDL设计

【例 3-20】

```
SIGNAL Q : STD_LOGIC_VECTOR(3 DOWNTO 0);  
.  
.  
.  
REG: PROCESS (CLK, RST, EN, Q, LOAD) BEGIN  
    IF RST='0' THEN Q <= (OTHERS=>'0') ;  
    ELSIF CLK'EVENT AND CLK='1' THEN  
        IF EN='1' THEN  
            IF (LOAD='0') THEN Q<=DATA; ELSE  
                IF Q<9 THEN Q<=Q+1; ELSE Q<=(OTHERS=>'0'); END IF;  
            END IF;  
        END IF;  
    END IF;  
END PROCESS;  
DOUT <= Q;  
COM: PROCESS (Q) BEGIN  
    IF Q="1001" THEN COUT<='1'; ELSE COUT<='0'; END IF;  
END PROCESS;
```

3.3 IF语句使用示例

3.3.9 含同步并行预置功能的8位移位寄存器设计

【例 3-21】

```
LIBRARY IEEE;
USE IEEE.STD LOGIC 1164.ALL;
ENTITY SHFT IS
    PORT (CLK, LOAD : IN STD LOGIC;  QB : OUT STD LOGIC
          DIN  : IN STD LOGIC VECTOR(7 DOWNTO 0);
          DOUT : OUT STD LOGIC VECTOR(7 DOWNTO 0) );
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;
```

3.3 IF语句使用示例

3.3.9 含同步并行预置功能的8位移位寄存器设计



图 3-14 例 3-21 的工作时序

3.3 IF语句使用示例

3.3.10 优先编码器设计

【例 3-22】

```
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;
```

3.3 IF语句使用示例

3.3.10 优先编码器设计

表 3-3 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。

7.3.5 数据类型定义语句

3.4 VHDL其它顺序语句

3.4.1 LOOP循环语句

(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;  
...
```

3.4 VHDL其它顺序语句

3.4.1 LOOP循环语句

(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];
```

```
WHILE 条件 LOOP  
    顺序语句  
END LOOP;
```

3.4 VHDL其它顺序语句

3.4.2 NEXT语句

```
NEXT;                                -- 第一种语句格式
NEXT LOOP 标号;                       -- 第二种语句格式
NEXT LOOP 标号 WHEN 条件表达式;      -- 第三种语句格式
```

【例 3-23】

```
...
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;
```

3.4 VHDL其它顺序语句

3.4.2 NEXT语句

【例 3-24】

```
...
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 ;
...

```

3.4 VHDL其它顺序语句

3.4.3 EXIT语句

```
EXIT; -- 第一种语句格式
EXIT LOOP 标号; -- 第二种语句格式
EXIT LOOP 标号 WHEN 条件表达式; -- 第三种语句格式
```

【例 3-25】

```
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; --NULL 的加入仅仅是为了 ELSE 语句的应用, 没有别的语法功能
END IF;
END LOOP; -- 当 i=1 时返回 LOOP 语句继续比较
```

3.4 VHDL其它顺序语句

3.4.4 WAIT语句

```
WAIT ON 信号表;                -- 第一种语句格式
WAIT UNTIL 条件表达式;        -- 第二种语句格式
WAIT FOR 时间表达式;          -- 第三种语句格式, 超时等待语句
```

【例 3-26】

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

3.4 VHDL其它顺序语句

3.4.4 WAIT语句

【例 3-27】

(a) WAIT_UNTIL 结构

```
...  
wait until enable = '1';  
...
```

(b) WAIT_ON 结构

```
LOOP  
    wait on enable;  
EXIT WHEN enable = '1';  
END LOOP;
```

WAIT UNTIL 信号=Value ; -- (1)

WAIT UNTIL 信号' EVENT AND 信号=Value; -- (2)

WAIT UNTIL NOT 信号' STABLE AND 信号=Value; -- (3)

3.4 VHDL其它顺序语句

3.4.4 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 3-28】

```
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 ;
```


3.4 VHDL其它顺序语句

3.4.4 WAIT语句

【例 3-29】

```
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;
```

【例 3-30】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY shifter IS
    PORT ( data : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
          shift_left, shift_right: IN STD_LOGIC;
          clk, reset : IN STD_LOGIC;
          mode : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
          qout : BUFFER STD_LOGIC_VECTOR (7 DOWNTO 0) );
END shifter;
ARCHITECTURE behave OF shifter IS
    SIGNAL enable: STD_LOGIC;
    BEGIN
    PROCESS    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;
END behave;
```

3.4 VHDL其它顺序语句

3.4.4 WAIT语句

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

3.4 VHDL其它顺序语句

3.4.5 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 ] } ) ;
```

3.4 VHDL其它顺序语句

3.4.6 REPORT语句

REPORT <字符串> ;

【例 3-31】

```
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
    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;
```

3.4 VHDL其它顺序语句

3.4.7 断言语句

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

表 3-4 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

3.4 VHDL其它顺序语句

1. 顺序断言语句

【例 3-32】

```
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;
```

3.4 VHDL其它顺序语句

2. 并行断言语句 【例 3-33】

```
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 ;
```


3.4 VHDL其它顺序语句

3.4.8 端口数据含1个数统计电路模块设计

【例 3-34】

```
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;
```

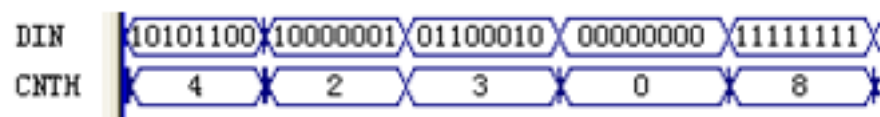


图 3-15 例 3-34 的仿真波形

习题

3-2

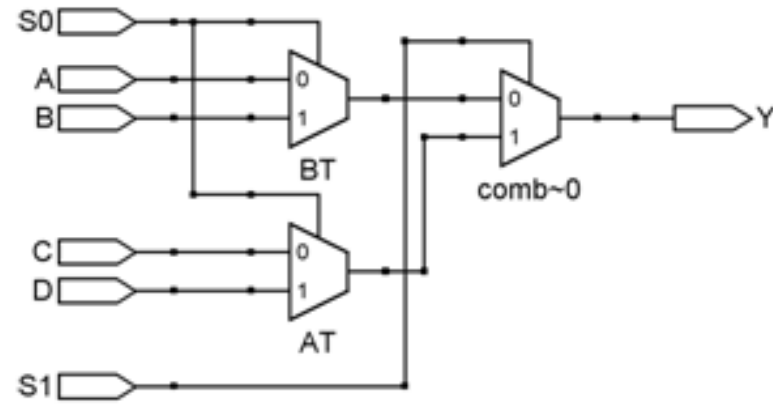


图 3-16 4 选 1 多路选择器 RTL 图