EDA 技术实用教程

第 10 章
VHDL基本语句

10.1 顺序语句

10.1.1 赋值语句

信号赋值语句　　变量赋值语句

10.1.2 IF语句

10.1.3 CASE语句

单个普通数值，如6。
数值选择范围，如(2 TO 4)。
并列数值，如3|5。
混合方式，以上三种方式的混合。

【例10-1】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux41 IS
PORT (s4, s3, s2, s1 : IN STD_LOGIC;
      z4, z3, z2, z1 : OUT STD_LOGIC);
END mux41;
ARCHITECTURE activ OF mux41 IS
SIGNAL sel : INTEGER RANGE 0 TO 15;
BEGIN
PROCESS (sel , s4, s3, s2, s1 )
  BEGIN
       sel<= 0 ;                          -- 输入初始值
      IF (s1 ='1')  THEN sel <=  sel+1 ;
    ELSIF (s2 ='1')  THEN sel <= sel+2 ;
    ELSIF (s3 ='1')  THEN sel <= sel+4 ;
    ELSIF (s4 ='1')  THEN sel <= sel+8 ;
    ELSE NULL;                          -- 注意，这里使用了空操作语句
  END IF ;
      z1<='0' ; z2<='0'; z3<='0'; z4<='0';  --输入初始值
CASE sel IS
  WHEN  0   => z1<='1' ;                  -- 当sel=0时选中
  WHEN 1|3 => z2<='1' ;                   -- 当sel为1或3时选中
  WHEN 4 TO 7|2  => z3<='1';      -- 当sel为2、4、5、6或7时选中
  WHEN OTHERS => z4<='1' ;        -- 当sel为8~15中任一值时选中
 END CASE ;
END PROCESS ;
END activ ;
```

10.1 顺序语句

10.1.3 CASE语句

【例10-2】

```
SIGNAL value : INTEGER RANGE 0 TO 15;
SIGNAL  out1 : STD_LOGIC ;
   ...
  CASE value IS                          -- 缺少以WHEN引导的条件句
  END CASE;
    ...
 CASE value IS
    WHEN 0 => out1<= '1' ;              -- value2～15的值未包括进去
    WHEN 1 => out1<= '0' ;
      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;
```


【例10-3】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY alu IS
    PORT(   a, b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
            opcode: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
            result: OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
END alu;
ARCHITECTURE behave OF alu IS
    CONSTANT plus     : STD_LOGIC_VECTOR (1 DOWNTO 0) := b"00";
    CONSTANT minus    : STD_LOGIC_VECTOR (1 DOWNTO 0) := b"01";
    CONSTANT equal    : STD_LOGIC_VECTOR (1 DOWNTO 0) := b"10";
    CONSTANT not_equal: STD_LOGIC_VECTOR (1 DOWNTO 0) := b"11";
BEGIN
PROCESS (opcode,a,b)
BEGIN
    CASE opcode IS
        WHEN plus =>  result <= a + b;  -- a、b相加
        WHEN minus => result <= a - b;  -- a、b相减
        WHEN equal =>                   -- a、b相等
      IF (a = b) THEN  result <= x"01";
                  ELSE result <= x"00";
      END IF;
        WHEN not_equal =>               -- a、b不相等
            IF (a /= b) THEN result <= x"01";
                        ELSE    result <= x"00";
            END IF;
    END CASE;
END PROCESS;
END behave;
```

10.1 顺序语句

10.1.4 LOOP语句

(1) 单个LOOP语句，其语法格式如下：

```
[ 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;
...
```

10.1 顺序语句

10.1.4 LOOP语句

(2) FOR_LOOP语句，语法格式如下：

```
[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];
```


10.1 顺序语句

【例10-4】

```
 LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;
ENTITY p_check IS
 PORT (  a : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
         y : OUT STD_LOGIC );
END p_check;
ARCHITECTURE opt OF p_check IS
  SIGNAL tmp : STD_LOGIC ;
BEGIN
  PROCESS(a)
    BEGIN
     tmp <='0';
     FOR n IN 0 TO 7 LOOP
      tmp <= tmp XOR a(n);
     END LOOP ;
      y <= tmp;
     END PROCESS;
  END opt;
```

10.1 顺序语句

10.1.4 LOOP语句

【例10-5】

```
    SIGNAL a, b, c : STD_LOGIC_VECTOR (1 TO 3);
...
FOR n IN 1 To 3 LOOP
a(n) <= b(n) AND c(n);
END LOOP;
```

此段程序等效于顺序执行以下三个信号赋值操作：

```
a(1)<=b(1) AND c(1);
a(2)<=b(2) AND c(2);
a(3)<=b(3) AND c(3);
```

10.1 顺序语句

10.1.5 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10-6】

```
    ...
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;
```


10.1 顺序语句

10.1.5 NEXT语句

【例10-7】

```
        ...
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 ;
   …
```


10.1 顺序语句

10.1.6 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
```

10.1 顺序语句

10.1.6 EXIT语句

【例10-8】

```
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
```


10.1 顺序语句

10.1.7 WAIT语句

```
WAIT;                                -- 第一种语句格式
WAIT ON 信号表;                      -- 第二种语句格式
WAIT UNTIL 条件表达式;               -- 第三种语句格式
WAIT FOR 时间表达式;    -- 第四种语句格式，超时等待语句
```


10.1 顺序语句

10.1.7 WAIT语句

【例10-9】

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

【例10-10】

(a) WAIT_UNTIL结构

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

(b) WAIT_ON结构

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

10.1 顺序语句

10.1.7 WAIT语句

```
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)
```

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


10.1 顺序语句

10.1.7 WAIT语句

【例10-11】

```
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 ;
```


10.1 顺序语句

10.1.7 WAIT语句

【例10-12】

```
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10-13】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY shifter IS
    PORT ( data : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
           shift_left: IN STD_LOGIC;
           shift_right: IN STD_LOGIC;
           clk: IN STD_LOGIC;
           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;
```

10.1 顺序语句

10.1.8 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10-14】

```
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
```

（接下页）

```
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10-15】

```
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;
```


10.1 顺序语句

10.1.9 RETURN语句

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

【例10-16】

```
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 qual to '1'";
   RETURN ;
   ELSE
   q <= s AND nq AFTER 5 ns ;
  nq <= s AND  q AFTER 5 ns ;
  END IF ;
END PROCEDURE rs ;
```


10.1 顺序语句

【例10-17】

```
   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 ;
```

10.1.10 空操作语句

```
CASE Opcode IS
WHEN  "001" =>  tmp := rega AND regb ;
WHEN  "101" =>  tmp := rega OR regb ;
WHEN  "110" =>  tmp := NOT rega ;
WHEN OTHERS  =>  NULL ;
END CASE ;
```

10.2 并行语句

- 并行信号赋值语句(Concurrent Signal Assignments)。
- 进程语句(Process Statements)。
- 块语句(Block Statements)。
- 条件信号赋值语句(Selected Signal Assignments)。
- 元件例化语句(Component Instantiations)，其中包括类属配置语句。
- 生成语句(Generate Statements)。
- 并行过程调用语句(Concurrent Procedure Calls)。

```
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
```


10.2 并行语句

10.2.1 并行信号赋值语句

1. 简单信号赋值语句

赋值目标 <= 表达式

```
ARCHITECTURE curt OF bc1 IS
SIGNAL s1, e, f, g, h : STD_LOGIC ;
BEGIN
    output1 <= a AND b ;
    output2 <= c + d ;
    g <= e OR f ;
    h <= e XOR f ;
    s1 <= g ;
END ARCHITECTURE curt;
```


10.2 并行语句

10.2.1 并行信号赋值语句

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10-18】

```
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;
```

10.2 并行语句

10.2.1 并行信号赋值语句

2. 条件信号赋值语句

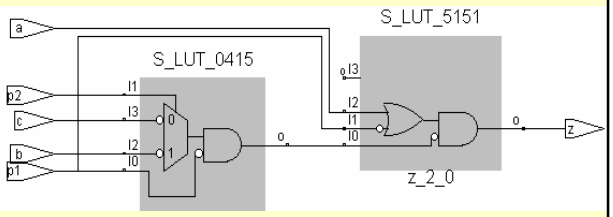


图10-1 例10-18的RTL电路图（Synplify综合）

10.2 并行语句

10.2.1 并行信号赋值语句

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;
```


【例10-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY decoder IS
   PORT (  a,  b,  c  :   IN STD_LOGIC;
         data1, data2 :   IN STD_LOGIC;
              dataout   : OUT STD_LOGIC  );
END decoder;
ARCHITECTURE concunt OF decoder IS
    SIGNAL instruction : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
    BEGIN
 instruction <= c & b & a ;
 WITH instruction SELECT
 dataout <= data1  AND data2  WHEN "000" ,
               data1  OR  data2  WHEN "001" ,
               data1 NAND data2  WHEN "010" ,
               data1 NOR  data2  WHEN "011" ,
               data1 XOR  data2  WHEN "100" ,
               data1 XNOR data2  WHEN "101" ,
                           'Z'   WHEN OTHERS ;
END concunt;
```

10.2 并行语句

10.2.1 并行信号赋值语句

3. 选择信号赋值语句

```
...
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 ;
  …
```


10.2 并行语句

10.2.2 块语句结构

BLOCK 语句的表达格式如下：

```
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 ;
```


【例10-20】

```
   ...
   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 ;
```

10.2 并行语句

10.2.2 块语句结构

【例10-21】

```
    ...
b1 :  BLOCK
      SIGNAL s1: BIT ;
      BEGIN
 S1 <= a AND b ;
b2 :  BLOCK
 SIGNAL s2: BIT ;
 BEGIN
   s2 <= c AND d ;
   b3 : BLOCK
    BEGIN
      Z <= s2 ;
    END BLOCK b3 ;
 END BLOCK b2 ;
   y <= s1 ;
   END BLOCK b1 ;
```

…

【例10-22】

```
LIBRARY IEEE;
 USE IEEE. std_logic_1164.ALL;
  ENTITY f_adder IS
    PORT ( ain, bin , cin : IN std_logic;
           sum, cout : OUT std_logic );
  END  f_adder;
  ARCHITECTURE e_ad OF f_adder IS
   SIGNAL so1, co1, co2 : std_logic;
  BEGIN
    h_adder1 : BLOCK          --半加器u1
      BEGIN
       PROCESS( ain,bin )
        BEGIN
        so1<=NOT(ain XOR (NOT bin)); co1<= ain AND bin;
       END PROCESS;
  END BLOCK   h_adder1;
h_adder2: BLOCK               --半加器u2
      SIGNAL so2 : std_logic;
      BEGIN
   so2 <= NOT(so1 XOR (NOT cin)) ; co2<=so1 and cin ; sum<=so2;
END BLOCK h_adder2;
  or2 : BLOCK                  --或门u3
         BEGIN
      PROCESS (co2, co1)
          BEGIN
        cout<= co2 OR co1;
END PROCESS;
END BLOCK or2;
      END e_ad;
```


10.2 并行语句

10.2.2 块语句结构

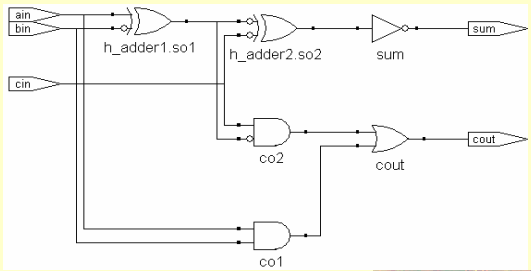


图10-2 例10-22的RTL电路图（Synplify综合）

10.2 并行语句

10.2.3 并行过程调用语句

过程名(关联参量名);

【例10-23】

```
...
   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
```


10.2 并行语句

10.2.3 并行过程调用语句

【例10-24】

```
   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;
```


10.2 并行语句

10.2.3 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;
  …
```

10.2 并行语句

10.2.4 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 文件名;
例化名 : 元件名 PORT MAP(               -- 元件例化语句
    [端口名 =>] 连接端口名, …);
```


10.2 并行语句

10.2.5 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] ;
```


10.2 并行语句

10.2.5 生成语句

```
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
```

【例10-25】

```
  ...
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,
 …
```

10.2 并行语句

10.2.5 生成语句

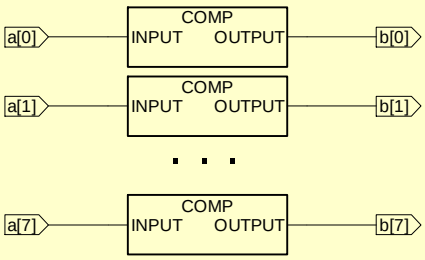


图10-3 生成语句产生的8个相同的电路模块

10.2 并行语句

10.2.5 生成语句

【例10-26】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY Latch IS
  PORT( D, ENA : IN STD_LOGIC;
        Q : OUT STD_LOGIC   );
END ENTITY Latch ;
  ARCHITECTURE one OF Latch IS
  SIGNAL sig_save : STD_LOGIC;
  BEGIN
   PROCESS (D, ENA)
     BEGIN
      IF  ENA = '1'  THEN  sig_save <= D ;  END IF ;
        Q <= sig_save ;
   END PROCESS ;
END ARCHITECTURE one;
```


【例10-27】

```
LIBRARY IEEE;
   USE IEEE.STD_LOGIC_1164.ALL;
   ENTITY SN74373 IS
   PORT (D : IN STD_LOGIC_VECTOR( 8 DOWNTO 1 );
     OEN ,G : IN  STD_LOGIC;
          Q : OUT STD_LOGIC_VECTOR(8 DOWNTO 1));
   END ENTITY SN74373;
ARCHITECTURE two OF SN74373 IS
     SIGNAL sigvec_save : STD_LOGIC_VECTOR(8 DOWNTO 1);
    BEGIN
    PROCESS(D, OEN, G , sigvec_save)
     BEGIN
      IF OEN = '0' THEN Q <= sigvec_save; ELSE Q <= "ZZZZZZZZ"; END IF;
      IF G = '1' THEN  Sigvec_save <= D;  END IF;
    END PROCESS;
  END ARCHITECTURE two;
 ARCHITECTURE one OF SN74373 IS
  COMPONENT Latch
  PORT (   D, ENA : IN STD_LOGIC;
           Q : OUT STD_LOGIC );
  END COMPONENT;
  SIGNAL sig_mid : STD_LOGIC_VECTOR( 8 DOWNTO 1 );
  BEGIN
   GeLatch : FOR iNum IN 1 TO 8 GENERATE
    Latchx : Latch PORT MAP(D(iNum),G,sig_mid(iNum));
  END GENERATE;
      Q <= sig_mid  WHEN OEN = '0' ELSE
           "ZZZZZZZZ";   --当OEN=1时，Q(8)~Q(1)输出状态呈高阻态
END ARCHITECTURE one;
```

【例10-28】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY d_ff IS
PORT (  d, clk_s : IN STD_LOGIC ;
             q : OUT STD_LOGIC ;
            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;
```

（接下页）

```
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;
```


10.2 并行语句

10.2.5 生成语句

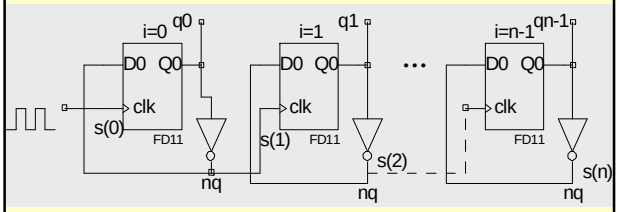


图10-4 6位二进制计数器原理图

10.2 并行语句

10.2.6 REPORT语句

【例10-29】

```
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;
```


10.2 并行语句

10.2.7 断言语句

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

表10-1 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

10.2 并行语句

10.2.7 断言语句

1. 顺序断言语句

【例10-30】

```
   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;
```

2. 并行断言语句

【例10-31】

```
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 ;
```

10.3 属性描述与定义语句

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
  ...
```


10.3 属性描述与定义语句

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'RIGNT ;
 ele2 <= obj'LEFT ;
 ele3 <= obj'HIGH ;
 ele4 <= obj'LOW  ;
```

【例10-32】

```
LIBRARY IEEE;--PARITY GENERATOR
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;
```


10.3 属性描述与定义语句

4. 数组属性'LENGTH

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


10.3 属性描述与定义语句

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. 用户定义属性

```
LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;
 ENTITY cntbuf IS
    PORT(  Dir: IN STD_LOGIC;
           Clk,Clr,OE: IN STD_LOGIC;
           A,B: INOUT STD_LOGIC_VECTOR (0 to 1);
           Q: INOUT STD_LOGIC_VECTOR (3 downto 0)  );
    ATTRIBUTE PINNUM : STRING;
    ATTRIBUTE PINNUM OF Clk: signal is "1";
    ATTRIBUTE PINNUM OF Clr: signal is "2";
    ATTRIBUTE PINNUM OF Dir: signal is "3";
    ATTRIBUTE PINNUM OF OE: signal is "11";
    ATTRIBUTE PINNUM OF Q: signal is "17,16,15,14";
END cntbuf;
```


习　题

10-1. 进程有哪几种主要类型？不完全组合进程是由什么原因引起的？有什么特点？如何避免？

10-2. 给触发器复位的方法有哪两种？如果时钟进程中用了敏感信号表，哪种复位方法要求把复位信号放在敏感信号表中？

10-3. 用WITH_SELECT_WHEN语句描述4个16位至1个16位输出的4选1多路选择器。

10-4. 为什么说一条并行赋值语句可以等效为一个进程？如果是这样的话，该语句怎样实现敏感信号的检测？

习　题

10-5. 下述VHDL代码的综合结果会有几个触发器或锁存器？

程序1：

```
   architecture rtl of ex is
       signal a, b: std_logic_vector(3 downto 0);
   begin
        process(clk)
        begin
            if clk = '1' and clk'event then
                  if q(3) /= '1' then  q <= a + b;
                   end if;
              end if;
        end process;
  end rtl;
```

习　题

10-5. 下述VHDL代码的综合结果会有几个触发器或锁存器？

程序2：

```
architecture rtl of ex is
    signal a, b: std_logic_vector(3 downto 0);
begin
    process(clk)
        variable int: std_logic_vector(3 downto 0);
    begin
        if clk ='1' and clk'event then
            if int(3) /= '1' then  int := a + b ; q <= int;
             end if;
        end if;
    end process;
end rtl;
```


习　题

10-5. 下述VHDL代码的综合结果会有几个触发器或锁存器？

程序3：

```
architecture rtl of ex is
    signal a, b，c, d, e: std_logic_vector(3 downto 0);
begin
    process(c, d, e, en)
    begin
      if en ='1'  then  a <= c ;  b <= d;
        else    a <= e;
      end if;
    end process;
end rtl;
```


习　题

10-6. 比较CASE语句与WITH_SELECT语句，叙述它们的异同点。

10-7. 将以下程序段转换为WHEN_ELSE语句：

```
PROCESS (a, b, c, d)
  BEGIN
  IF a= '0' AND b='1' THEN  next1 <= "1101" ;
      ELSIF  a='0' THEN  next1 <= d ;
   ELSIF  b='1' THEN  next1 <= c ;
        ELSE
        Next1 <= "1011" ;
  END  IF;
END PROCESS;
```

习　题

10-8. 说明以下两程序有何不同，哪一电路更合理？试画出它们的电路。

程序1：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY EXAP IS  PORT ( clk,a,b  : IN STD_LOGIC;
                       y : OUT STD_LOGIC );
END EXAP ;
 ARCHITECTURE behav OF EXAP IS
 SIGNAL x : STD_LOGIC;
BEGIN
 PROCESS
 BEGIN
  WAIT UNTIL CLK ='1' ;
       x <= '0';    y <= '0';
  IF a = b THEN    x <= '1';
  END IF;
  IF x='1' THEN    y <= '1' ;
  END IF ;
 END PROCESS  ;
END behav;
```


习　题

10-8. 说明以下两程序有何不同，哪一电路更合理？试画出它们的电路。

程序2：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY EXAP IS  PORT ( clk,a,b  : IN STD_LOGIC;
                       y : OUT STD_LOGIC );
END EXAP ;
 ARCHITECTURE behav OF EXAP IS
 BEGIN
 PROCESS
  VARIABLE x : STD_LOGIC;
 BEGIN
  WAIT UNTIL CLK ='1' ;
      x := '0';    y <= '0';
   IF a = b THEN  x := '1';
  END IF;
  IF x='1' THEN   y <= '1' ;
  END IF ;
 END PROCESS  ;
END behav;
```


实 验 与 设 计

10-1 移位相加硬件乘法器设计

(1) 实验目的：学习应用移位相加原理设计8位乘法器。

(2) 实验原理：该乘法器是由8位加法器构成的以时序方式设计的8位乘法器。原理是：乘法通过逐项移位相加来实现相乘。从被乘数的最低位开始，若为1，则乘数左移后与上一次的和相加；若为0，左移后以全零相加，直至被乘数的最高位。从图10-5的逻辑图及其乘法操作时序图图10-6(示例中的相乘数为9FH和FDH)上可以清楚地看出此乘法器的工作原理。

【例10-33】

```
LIBRARY IEEE;       -- 8位右移寄存器
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SREG8B IS
    PORT (  CLK, LOAD : IN STD_LOGIC;
             DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
              QB : OUT STD_LOGIC  );
END SREG8B;
ARCHITECTURE behav OF SREG8B 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;
             ELSE  REG8(6 DOWNTO 0) <= REG8(7 DOWNTO 1);
           END IF;
        END IF;
    END PROCESS;
    QB <= REG8(0);                          -- 输出最低位
END behav;
```

【例10-34】

```
LIBRARY IEEE;     --8位加法器
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ADDER8B IS
    PORT ( CIN : IN STD_LOGIC;
        A , B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        S : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        COUT : OUT STD_LOGIC );
END ADDER8B;
ARCHITECTURE behav OF ADDER8B IS
    SIGNAL SINT, AA,BB : STD_LOGIC_VECTOR(8 DOWNTO 0);
BEGIN
AA<='0'&A;BB<='0'&B; SINT<=AA+BB+CIN;S<=SINT(7 DOWNTO 0);
COUT<=SINT(8);
END behav;
```


【例10-35】

```
LIBRARY IEEE; --1位乘法器
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ANDARITH IS                          -- 选通与门模块
    PORT ( ABIN : IN STD_LOGIC;
         DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        DOUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
END ANDARITH;
ARCHITECTURE behav OF ANDARITH IS
BEGIN
    PROCESS(ABIN, DIN)
    BEGIN
        FOR I IN 0 TO 7 LOOP       -- 循环，完成8位与1位运算
            DOUT(I) <= DIN(I) AND ABIN;
        END LOOP;
    END PROCESS;
END behav;
```

【例10-36】

```
LIBRARY IEEE;                                  --16位锁存器/右移寄存器
USE IEEE.STD_LOGIC_1164.ALL;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG16B IS                               -- 16位锁存器
   PORT (CLK, CLR : IN STD_LOGIC;
      D : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
      Q : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
END REG16B;
ARCHITECTURE behav OF REG16B IS
   SIGNAL R16S : STD_LOGIC_VECTOR(15 DOWNTO 0);
BEGIN
   PROCESS(CLK, CLR)
   BEGIN
IF CLR='1' THEN R16S<="0000000000000000";--时钟到来时，锁存输入值，并右移低8位
ELSIF CLK'EVENT AND CLK='1' THEN
 R16S(6 DOWNTO 0) <=R16S(7 DOWNTO 1);       --右移低8位
        R16S(15 DOWNTO 7) <= D;                    -- 将输入锁到高8位
      END IF;
   END PROCESS;
   Q <= R16S;
END behav;
```

【例10-37】

```
END behav;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ARICTL IS
    PORT (CLK, START : IN STD_LOGIC;
        CLKOUT,RSTALL : OUT STD_LOGIC );
END ARICTL;
ARCHITECTURE behav OF ARICTL IS
    SIGNAL CNT4B : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
    PROCESS(CLK, START)
    BEGIN
        RSTALL <= START;
        IF START = '1' THEN  CNT4B <= "0000";
        ELSIF CLK'EVENT AND CLK ='1' THEN
         IF CNT4B < 8 THEN CNT4B <= CNT4B + 1; END IF;
        END IF;
    END PROCESS;
    PROCESS(CLK, CNT4B, START)
    BEGIN
        IF START = '0' THEN
            IF CNT4B < 8 THEN  CLKOUT <= CLK;
            ELSE CLKOUT <= '0';  END IF;
        ELSE  CLKOUT <= CLK; END IF;
    END PROCESS;
END behav;
```

【例10-38】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
ENTITY MULTI8X8 IS                             -- 8位乘法器顶层设计
    PORT ( CLKK,START  : IN STD_LOGIC;
                A,   B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                 DOUT : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
END MULTI8X8;
ARCHITECTURE struc OF MULTI8X8 IS
COMPONENT ARICTL
    PORT (  CLK, START : IN STD_LOGIC;
         CLKOUT, RSTALL : OUT STD_LOGIC  );
END COMPONENT;
COMPONENT ANDARITH
    PORT (  ABIN : IN STD_LOGIC;
              DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
             DOUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)  );
END COMPONENT;
COMPONENT ADDER8B
    PORT (CIN : IN STD_LOGIC;
            A, B : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            S : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
         COUT : OUT STD_LOGIC );
END COMPONENT;
COMPONENT SREG8B
    PORT (  CLK, LOAD : IN STD_LOGIC;
```

（接下页）

```
  DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
             QB : OUT STD_LOGIC  );
  END COMPONENT;
  COMPONENT REG16B
      PORT (   CLK, CLR : IN STD_LOGIC;
          D : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
          Q : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)  );
  END COMPONENT;
      SIGNAL GNDINT, INTCLK, RSTALL, NEWSTART, QB : STD_LOGIC;
      SIGNAL ANDSD : STD_LOGIC_VECTOR(7 DOWNTO 0);
       SIGNAL DTBIN : STD_LOGIC_VECTOR(8 DOWNTO 0);
      SIGNAL DTBOUT : STD_LOGIC_VECTOR(15 DOWNTO 0);
  BEGIN
      DOUT <= DTBOUT;   GNDINT <= '0';
  PROCESS(CLKK,START)
  BEGIN
       IF START='1' THEN NEWSTART<='1';
       ELSIF CLKK='0' THEN   NEWSTART<='0'; END IF;
  END PROCESS;
  U1  :  ARICTL    PORT  MAP(CLK=>CLKK,START=>NEWSTART,  CLKOUT=>INTCLK,
RSTALL=>RSTALL);
  U2 : SREG8B  PORT MAP(CLK=>INTCLK, LOAD=>RSTALL, DIN=>B, QB=>QB  );
  U3 : ANDARITH PORT MAP(ABIN => QB, DIN => A,DOUT => ANDSD);
  U4 : ADDER8B  PORT MAP(CIN => GNDINT, A=>DTBOUT(15 DOWNTO 8), B=>ANDSD,
                S => DTBIN(7 DOWNTO 0), COUT => DTBIN(8) );
  U5 : REG16B   PORT MAP(CLK=>INTCLK, CLR=>RSTALL, D=>DTBIN, Q=>DTBOUT );
   END struc;
```

实 验 与 设 计

10-1 移位相加硬件乘法器设计

(3) 实验内容1：根据给出的乘法器逻辑原理图及其各模块的VHDL描述，在QuartusII上完成全部设计，包括编辑、编译、综合和仿真操作等。以87H乘以F5H为例，进行仿真，对仿真波形作出详细解释，包括对8个工作时钟节拍中，每一节拍乘法操作的方式和结果，对照波形图给以详细说明，根据顶层设计例10-38，结合图10-5，画出乘法器的详细电路原理框图。

(4) 实验内容2：编程下载，进行实验验证。实验电路选择No.1，8位乘数用键2、键1输入；8位被乘数用键4和键3输入；16位乘积可由4个数码管（数码管8、7、6、5）显示；用键8输入CLK，键7输入START（注意，START由高到低是清0，由低到高电平是允许乘法计算）。详细观察每一时钟节拍的运算结果，并与仿真结果进行比较。

实 验 与 设 计

10-1 移位相加硬件乘法器设计

(5) 实验内容3：乘法时钟连接实验系统上的连续脉冲，如clock0，设计一个此乘法器的控制模块，接受实验系统上的连续脉冲，如clock0，当给定启动/清0信号后，能自动发出CLK信号驱动乘法运算，当8个脉冲后自动停止（参考程序：例10-37）。

(6) 实验内容4：设计一个纯组合电路的8X8等于16位的乘法器和一个LPM乘法器（选择不同的流水线方式），具体说明并比较这几种乘法器的逻辑资源占用情况和运行速度情况。

(7) 实验报告：根据例10-33至例10-38，详细分析各模块的逻辑功能，及其他们工作原理，详细记录并分析实验2和实验3的过程和结果，完成实验报告。

实 验 与 设 计

10-1 移位相加硬件乘法器设计

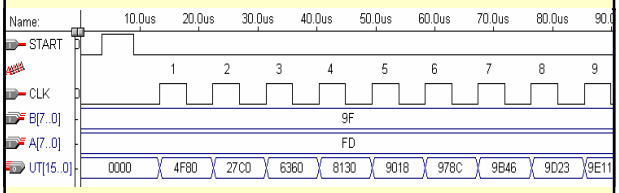


图10-5 8位移位相加乘法器运算逻辑波形图

实 验 与 设 计

10-2 等精度频率计/相位计设计

(1) 实验原理：基于传统测频原理的频率计的测量精度将随被测信号频率的下降而降低，即测量精度随被测信号的频率的变化而变化，在实用中有较大的局限性，而等精度频率计不但具有较高的测量精度，且在整个频率区域能保持恒定的测试精度。设计项目可达到的指标为：

频率测试功能：测频范围0.1Hz～100MHz。测频精度：测频全域相对误差恒为百万分之一。

脉宽测试功能：测试范围0.1μs～1s，测试精度0.01μs。

占空比测试功能：测试（显示）精度1%～99%。

相位测试功能：测试范围0～，测试精度0.。

实 验 与 设 计

1、主系统组成

等精度频率计的主系统如图10-6所示，主要由6个部分构成：

(1) 信号整形电路。用于对待测信号进行放大和整形，以作PLD器件的输入信号。

(2) 测频电路。是测频的核心电路模块，可以由FPGA器件担任。

(3) 100MHz的标准频率信号源(可通过PLL倍频所得)接入FPGA。

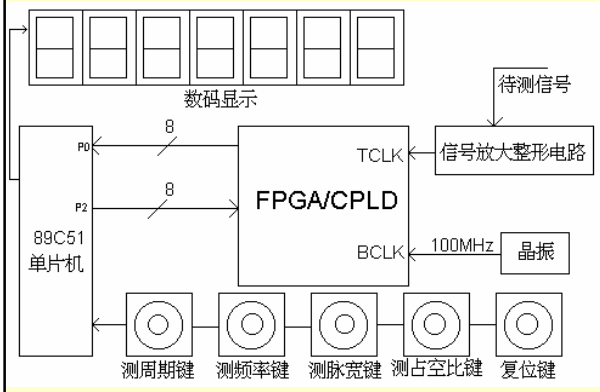
(4) 单片机电路模块。用于控制FPGA的测频操作和读取测频数据，并作出相应数据处理。安排单片机的P0口读取测试数据，P2口向FPGA发控制命令。

(5) 键盘模块。可以用5个键执行测试控制，一个是复位键，其余是命令键。

(6) 数码显示模块。可以用7个数码管显示测试结果，最高可表示百万分之一的精度。

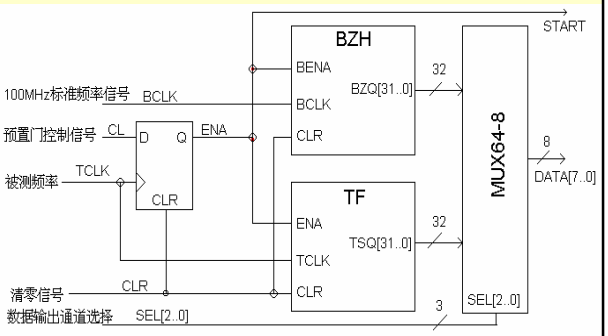
考虑到提高单片机IO口的利用率，降低编程复杂性，提高单片机的计算速度以及降低数码显示器对主系统的干扰，可以采用串行静态显示方式或液晶显示。

1、主系统组成



79 图10-6 频率计主系统电路组成 Kx康芯科技

2、主系统组成测频原理



80 图10-7 等精度频率计主控结构 Kx康芯科技

实 验 与 设 计

2、主系统组成测频原理

图10-8 频率计测控时序

81 Kx康芯科技

【例10-39】

```
LIBRARY IEEE;  --等精度频率计FPGA设计部分
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY etester IS
  PORT (BCLK : IN STD_LOGIC; --标准频率时钟信号clock2，50MHZ
        TCLK : IN STD_LOGIC; --待测频率时钟信号
        CLR : IN STD_LOGIC; --清零和初始化信号
CL : IN STD_LOGIC; --当SPUL为高电平时，CL为预置门控信号，用于测频计数
--时间控制当SPUL为低电平时，CL为测脉宽控制信号，
--CL高电平时测高电平脉宽而当CL为低电平时，测低电平脉宽。
     SPUL : IN STD_LOGIC; --测频或测脉宽控制
    START : OUT STD_LOGIC;--起始计数标志信号
        EEND : OUT STD_LOGIC; --由低电平变到高电平时指示脉宽计数结束，
        SEL : IN STD_LOGIC_VECTOR(2 DOWNTO 0); --数据读出选同控制
     DATA : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)); --8位数据读出
END etester;
ARCHITECTURE behav OF etester IS
SIGNAL BZQ  : STD_LOGIC_VECTOR(31 DOWNTO 0);  --标准计数器
SIGNAL TSQ  : STD_LOGIC_VECTOR(31 DOWNTO 0);  --测频计数器
SIGNAL ENA  : STD_LOGIC;                      --计数使能
SIGNAL  MA, CLK1, CLK2, CLK3 : STD_LOGIC;
SIGNAL   Q1, Q2, Q3,  BENA, PUL : STD_LOGIC;
SIGNAL   SS : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
START <= ENA ;
DATA <= BZQ(7  DOWNTO  0) WHEN SEL="000" ELSE -- 标准频率计数低8位输出  （接下页）
```

```
BZQ(15 DOWNTO  8) WHEN SEL="001" ELSE
        BZQ(23 DOWNTO 16) WHEN SEL="010" ELSE
       BZQ(31 DOWNTO 24) WHEN SEL="011" ELSE -- 标准频率计数最高8位输出
       TSQ(7 DOWNTO   0) WHEN SEL="100" ELSE --待测频率计数值最低8位输出
       TSQ(15 DOWNTO  8) WHEN SEL="101" ELSE
       TSQ(23 DOWNTO 16) WHEN SEL="110" ELSE
       TSQ(31 DOWNTO 24) WHEN SEL="111" ELSE --待测频率计数值最高8位输出
       TSQ(31 DOWNTO 24) ;
BZH :  PROCESS(BCLK, CLR)                --标准频率测试计数器，标准计数器
    BEGIN
     IF CLR = '1' THEN   BZQ <= ( OTHERS=>'0' ) ;
     ELSIF BCLK'EVENT AND BCLK = '1' THEN
      IF BENA = '1' THEN   BZQ <= BZQ + 1;   END IF;
     END IF;
   END PROCESS;
TF : PROCESS(TCLK, CLR, ENA)                  --待测频率计数器，测频计数器
   BEGIN
   IF CLR = '1' THEN   TSQ <= ( OTHERS=>'0' );
   ELSIF TCLK'EVENT AND TCLK = '1' THEN
    IF ENA = '1' THEN   TSQ <= TSQ + 1;  END IF;
   END IF;
END PROCESS;
PROCESS(TCLK,CLR)
 BEGIN
  IF CLR = '1' THEN   ENA <= '0' ;
  ELSIF  TCLK'EVENT AND TCLK='1' THEN ENA <= CL ;  END IF;
END PROCESS;                                                    （接下页）
```

```
MA<=(TCLK AND CL) OR NOT(TCLK OR CL)  ; --测脉宽逻辑
  CLK1<=NOT MA ; CLK2<=MA AND Q1 ; CLK3<=NOT CLK2; SS<=Q2 & Q3 ;
 DD1:  PROCESS(CLK1,CLR)
    BEGIN
     IF CLR = '1' THEN   Q1 <= '0' ;
      ELSIF CLK1'EVENT AND CLK1 = '1' THEN  Q1 <= '1' ; END IF;
     END PROCESS;
DD2:  PROCESS(CLK2,CLR)
  BEGIN
    IF CLR = '1' THEN   Q2 <= '0' ;
    ELSIF CLK2'EVENT AND CLK2 = '1' THEN  Q2 <= '1' ;  END IF;
  END PROCESS;
DD3:  PROCESS(CLK3,CLR)
   BEGIN
    IF CLR = '1' THEN   Q3 <= '0' ;
    ELSIF CLK3'EVENT AND CLK3 = '1' THEN  Q3 <= '1' ;  END IF;
 END PROCESS;
PUL<='1' WHEN SS="10" ELSE --当SS=“10”时，PUL高电平，允许标准计数器计数，
     '0' ;                 --禁止计数
EEND<='1' WHEN SS="11" ELSE --EEND为低电平时，表示正在计数，由低电平变到高电平
      '0' ;                --时，表示计数结束，可以从标准计数器中读数据了
BENA<=ENA WHEN SPUL='1' ELSE--标准计数器时钟使能控制信号，当SPUL为1时，测频率
      PUL WHEN SPUL='0' ELSE--当SPUL为0时，测脉宽和占空比
      PUL ;
END behav;
```

3、VHDL测试程序设计

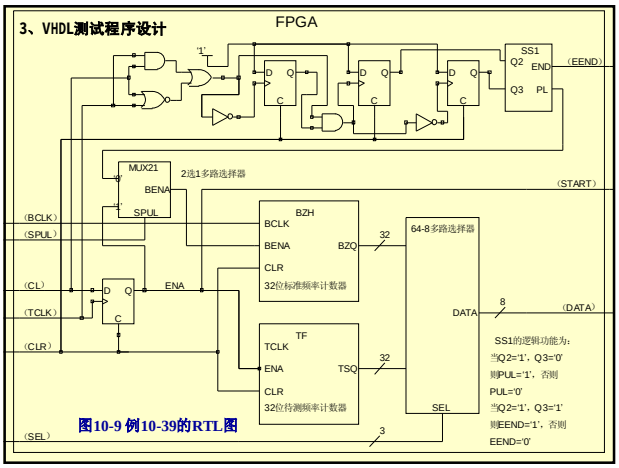
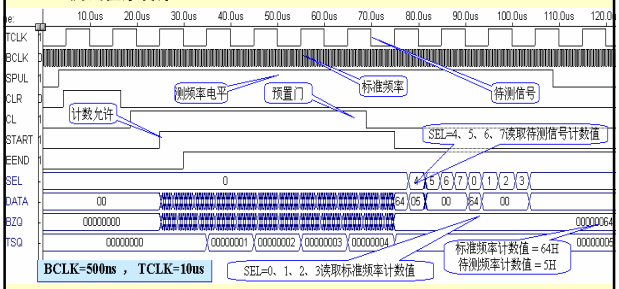


图10-9 例10-39的RTL图

实 验 与 设 计

3、VHDL测试程序设计



86 图10-10 等精度频率计测频时序图 Kx康芯科技

实 验 与 设 计

3、VHDL测试程序设计

87 图10-11 等精度频率计测脉宽时序图 Kx康芯科技

实 验 与 设 计

4、主系统组成测试与实现

5、相位测试

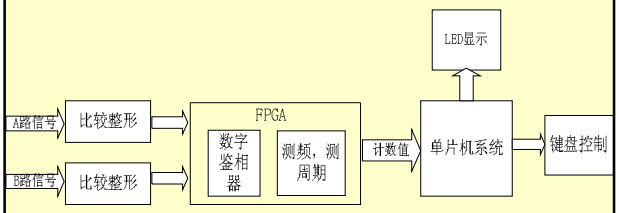


图10-12 测相仪模型

实 验 与 设 计

10-2 等精度频率计/相位计设计

（2）实验内容1：根据以上给出步骤首先完成等精度频率计FPGA的设计，按照图10-11和图10-12的时序，在GW48系统上硬件验证例10-39的各项功能：等精度测频率、测脉宽、测占空比。与GW48系统上给出的标准待测频率，计算误差，并与理论误差值比较。

（3）实验内容2：根据图10-11和图10-12和式10-2、10-3设计单片机程序，完成单片机与FPGA的接口程序、控制程序和计算显示程序的设计。完成等精度频率计独立系统的设计，控制键可以参考图10-6的电路，每一个键控制一种功能。

（4）实验内容3：根据图10-14、10-13、10-12和10-10，修改原设计，增加测相位功能，并在系统上增加一个键，控制测相差和显示。被测信号可以用前面设计的移相信号发生器产生。

（5）实验内容4：用嵌入式锁相环PLL的LPM模块对实验系统的50MHz或20MHz时钟源倍频，达到100MHz，PLL的输出信号作为频率计的标准频率源。注意PLL的输入时钟必须是器件的专用时钟输入脚CLK0 à pin16（对于EP1C3TC144）。

实 验 与 设 计

10-2 等精度频率计/相位计设计

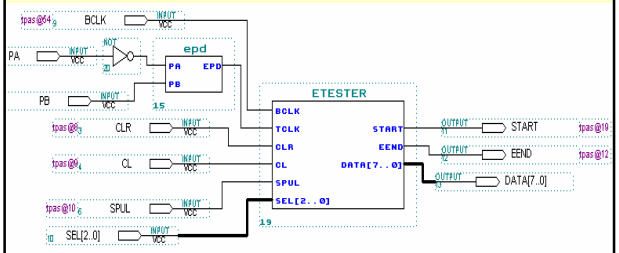


图10-13 测相仪电路原理图（TPAS.bdf工程）

实 验 与 设 计

10-2 等精度频率计/相位计设计

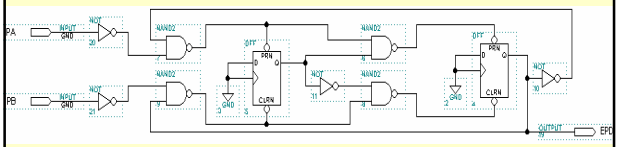


图10-14 相位检测原理图epd

实 验 与 设 计

10-2 等精度频率计/相位计设计

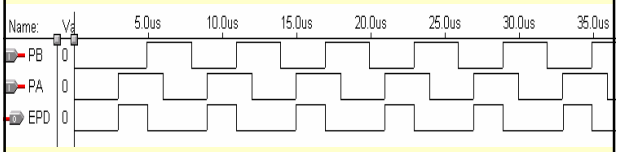


图10-15 鉴相器EPD的仿真波形

实 验 与 设 计

10-3 基于8051单片机IP核的等精度频率计单片系统设计（LCD显示）

(1) 实验原理：如果利用8051单片机核，则能将图10-6主要元件集成在单片FPGA中。图10-16是一个含有等精度频率计测试模块的8051单片机系统，其中的单片机核已在第7章中作了介绍，图中的ETESTER模块的VHDL源程序是例10-39。该模块与单片机的接口方式与前面完全一样 ，因此，除了显示部分，单片机中的汇编程序也完全相同。

(2) 实验内容1：对图10-16所示的工程完成验证性实验。设定实验系统为模式3。用Quartusll打开工程文件：/D8051_COPY/GW48PK_LCD/MCU8951，编译后下载，按复位键（键9 0->1->0），用一短线将适配板上方P208与clock0相接，作为等精度频率计信号输入口。

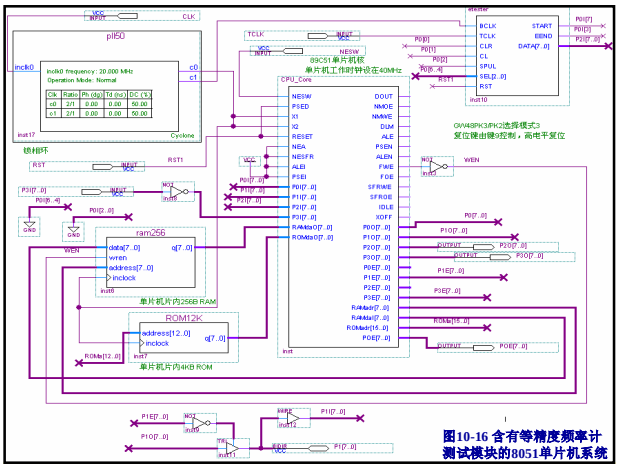


图10-16 含有等精度频率计测试模块的8051单片机系统

实 验 与 设 计

10-3 基于8051单片机IP核的等精度频率计单片系统设计（LCD显示）

（3）实验内容2：修改单片机示例程序，增加一些测试功能，编译、下载，调试软件。

（4）实验内容3：基于图10-16，修改硬件平台，比如ROM大小、模块etester的功能，增加其他接口模块，完成进一步软硬件调试实验。注意，使用单片机核的双向口时必须设置FPGA的端口为上拉！

（5）实验内容4：根据图10-13，加入一个鉴相器模块，增加测相功能。

10-4 基于8051单片机IP核的等精度频率计单片系统设计（LED显示）

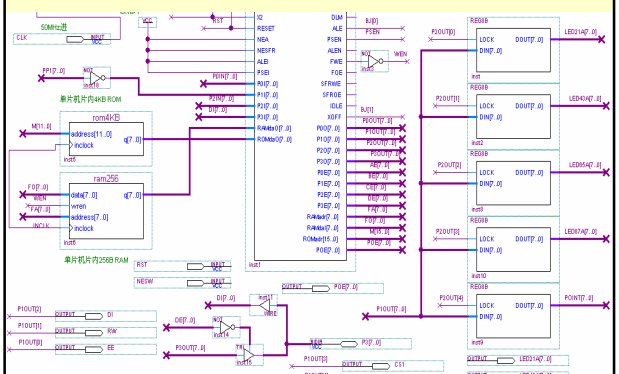


 图10-17 含有数码管显示控制电路的等精度频率计单片测试系统