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

10.1.2 IF语句

10.1.3 CASE语句

选择值 [|选择值]

10.1 顺序语句

10.1.3 CASE语句

【例10-1】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY EMP1 IS
PORT (          sel : IN INTEGER RANGE 0 TO 15 ;
      z4,z3, z2,z1 : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)  );
END EMP1 ;
ARCHITECTURE activ OF EMP1 IS
 BEGIN
PROCESS (sel )    BEGIN
 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 ;
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' ;                   -- 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;
```

【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

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

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 10-4】

```
...
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 10-5】

```
...
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 10-6】

```
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
```

【例 10-7】

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

10.1 顺序语句

10.1.7 WAIT语句

【例 10-8】

(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 10-9】

```
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 10-10】

```
    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 10-11】

```
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;
```

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}) ];
```

（1）将IN和INOUT模式的实参值赋给欲调用的过程中与它们对应的形参；

（2）执行这个过程。

（3）将过程中IN和INOUT模式的形参值返回给对应的实参。

【例 10-12】

```
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 ;
```

接下页

10.1 顺序语句

10.1.8 子程序调用语句

1. 过程调用

接上页

```
  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 10-13】

```
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

10.1.8 子程序调用语句

2. 函数调用

函数调用与过程调用十分相似，不同之处是，调用函数将返回一个指定数据类型的值，函数的参量只能是输入值。

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 10-14】

```
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 ;
```

10.1 顺序语句

10.1.9 RETURN语句

【例 10-15】

```
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 顺序语句

10.1.10 NULL语句

```
NULL;
```

```
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 VHDL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 VHDL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 VHDL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 ;
```

10.2 VHDL并行语句

10.2.1 并行信号赋值语句

2. 条件信号赋值语句

【例 10-16】

```
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 VHDL并行语句

10.2.1 并行信号赋值语句

2. 条件信号赋值语句

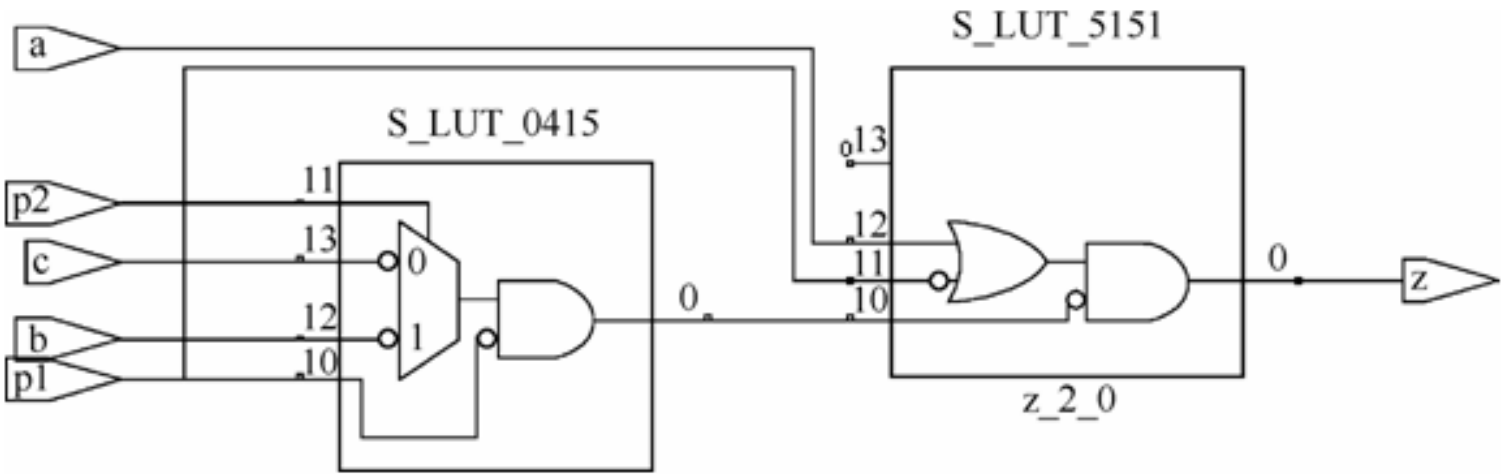

图 10-1　例 10-16 的 RTL 电路图(Symplify 综合)

10.2 VHDL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;

UNAFFECTED   WHEN OTHERS ;
```

【例 10-17】

```
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" ,
                 UNAFFECTED   WHEN OTHERS ;
END concunt ;
```

10.2 VHDL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 VHDL并行语句

10.2.2 块语句

```
块标号 : BLOCK [(块保护表达式)]
        接口说明
            类属说明
            BEGIN
        并行语句
END BLOCK 块标号 ;
```

【例 10-18】

```
  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 VHDL并行语句

10.2.2 块语句

【例 10-19】

```
  ...
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 10-20】

```
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;
```

接下页

10.2 VHDL并行语句

10.2.2 块语句

接上页

```
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 VHDL并行语句

10.2.2 块语句

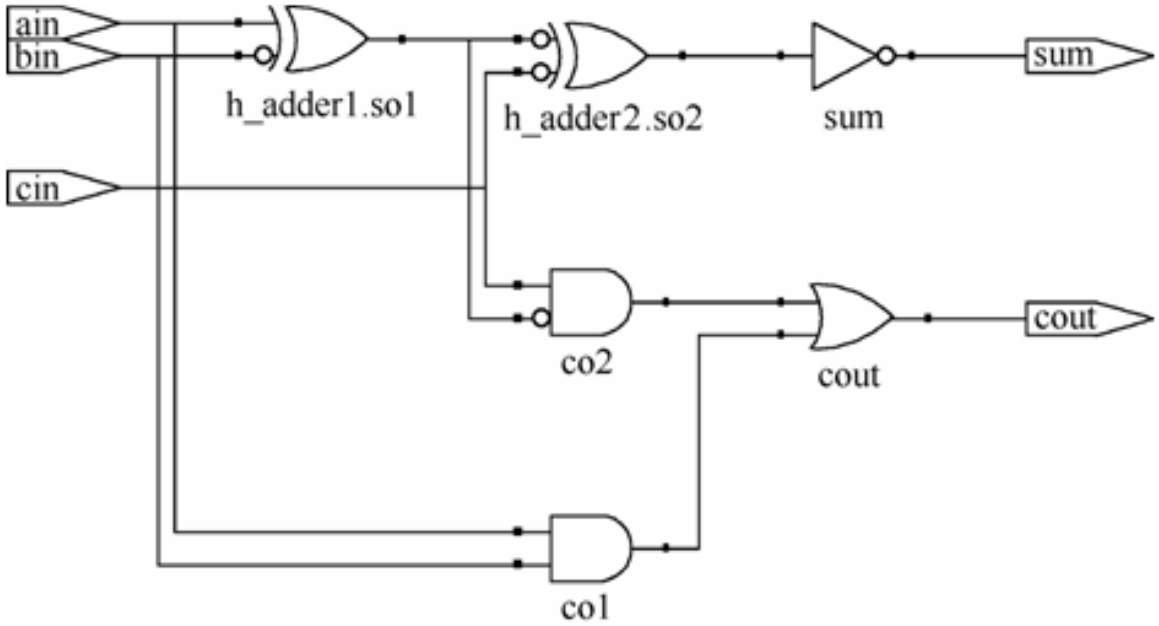

图 10-2　例 10-20 的 RTL 电路图（Symplify 综合）

10.2 VHDL并行语句

10.2.3 并行过程调用语句

【例 10-21】

```
    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 VHDL并行语句

10.2.3 并行过程调用语句

【例 10-22】

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

10.2 VHDL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 VHDL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 10-23】

```
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 VHDL并行语句

10.2.5 生成语句

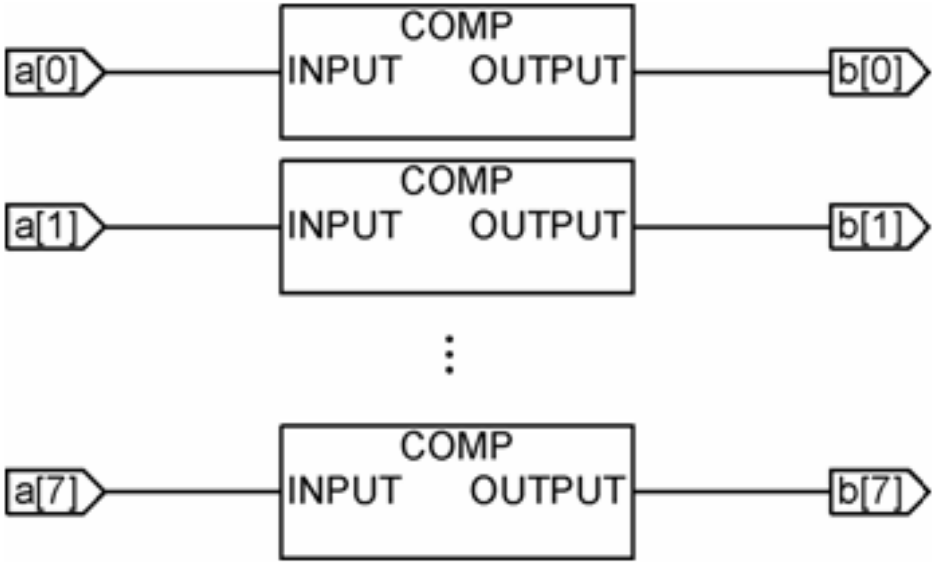

图 10-3　生成语句产生的 8 个相同的电路模块

10.2 VHDL并行语句

10.2.5 生成语句

【例 10-24】

```
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 10-25】

```
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 10-26】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY d_ff IS
PORT (  d, clk_s : IN STD_LOGIC ;
          q ,     nq : OUT STD_LOGIC  );
END ENTITY d_ff;
ARCHITECTURE a_rs_ff OF d_ff IS
BEGIN
bin_p_rs_ff : PROCESS(CLK_S)   BEGIN
     IF clk_s = '1' AND clk_s'EVENT
       THEN q <= d; nq <= NOT d;    END IF;
   END PROCESS;
END ARCHITECTURE a_rs_ff;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY cnt_bin_n is
GENERIC (n : INTEGER := 6);
PORT (q : OUT STD_LOGIC_VECTOR (0 TO n-1);
   in_1 : IN  STD_LOGIC );
END ENTITY cnt_bin_n;
```

接下页

10.2 VHDL并行语句

10.2.5 生成语句

接上页

```
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 VHDL并行语句

10.2.5 生成语句

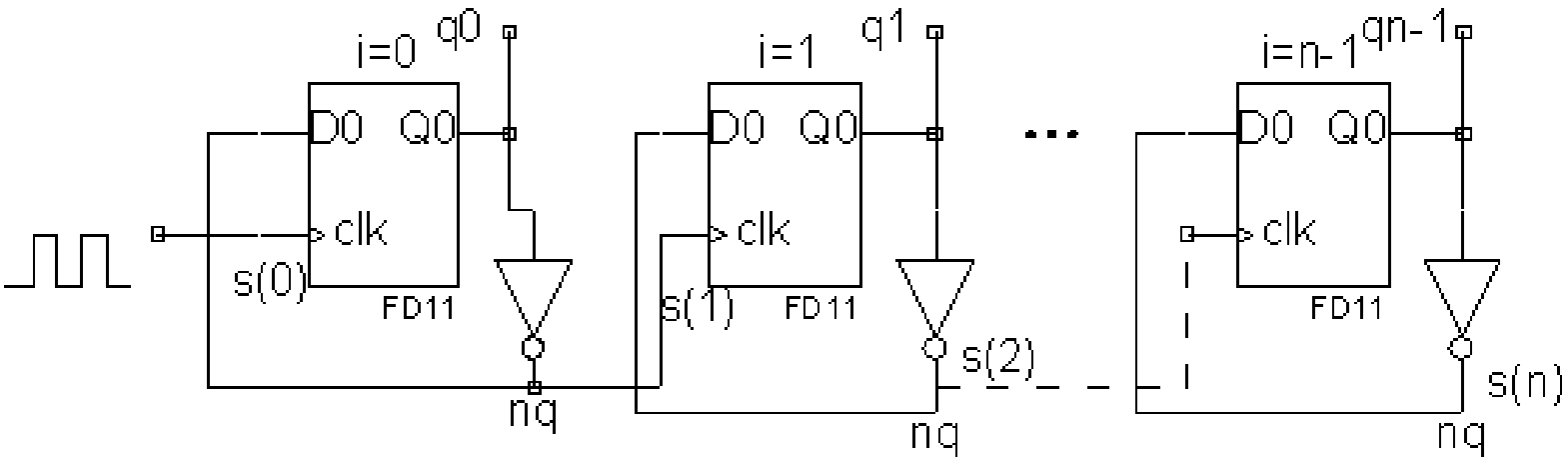

图 10-4　6 位二进制计数器原理图

10.2 VHDL并行语句

10.2.6 REPORT语句

【例 10-27】

```
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 VHDL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 VHDL并行语句

10.2.7 断言语句

1. 顺序断言语句

【例 10-28 】

```
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;
```

10.2 VHDL并行语句

10.2.7 断言语句

2. 并行断言语句

【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
  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

'RANGE [(n)]　　'REVERSE_RANGE[(n)]

```
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

‘LEFT、 ’RIGHT、 ‘HIGH 、 'LOW

```
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 10-30】

```
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 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;
```

```
LIBRARY synplify;
USE synplicity.attributes.all;
```

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 详细讨论并用示例说明**WITH_SELECT_WHEN**语句和**CASE**语句的异同点。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？

10-5 用两种方法设计比较器，比较器的输入是两**8**位数**A[7..0]**和**B[7..0]**，输出是 **D**、**E**、**F**。当**A=B**时**D=1**；当**A>B**时**E=1**；当**A<B**时**F=1**。第一种设计方案是常规的比较器设计方法，即直接利用关系操作符进行编程设计；第二种设计方案是利用减法器来完成，通过减法运算后的符号和结果来判别两个被比较值的大小。对两种设计方案的资源耗用情况进行比较并给以解释。

10-6 设计**VHDL**程序，产生**0**至**100**间的随机数，其中小于**50**的数的比例是**70%**。

实验与设计

10-1. 直流电机综合测控系统设计

(1) 实验目的：

(2)）实验原理：

(3) 实验内容**1**：

(4) 实验内容**2**：

(5) 实验内容**3**：

基于**5E+**系统的示例演示文件：

/KX_7C5EE+/EXPERIMENTs/EXP28_DC_MOTO/PW1。

实验与设计

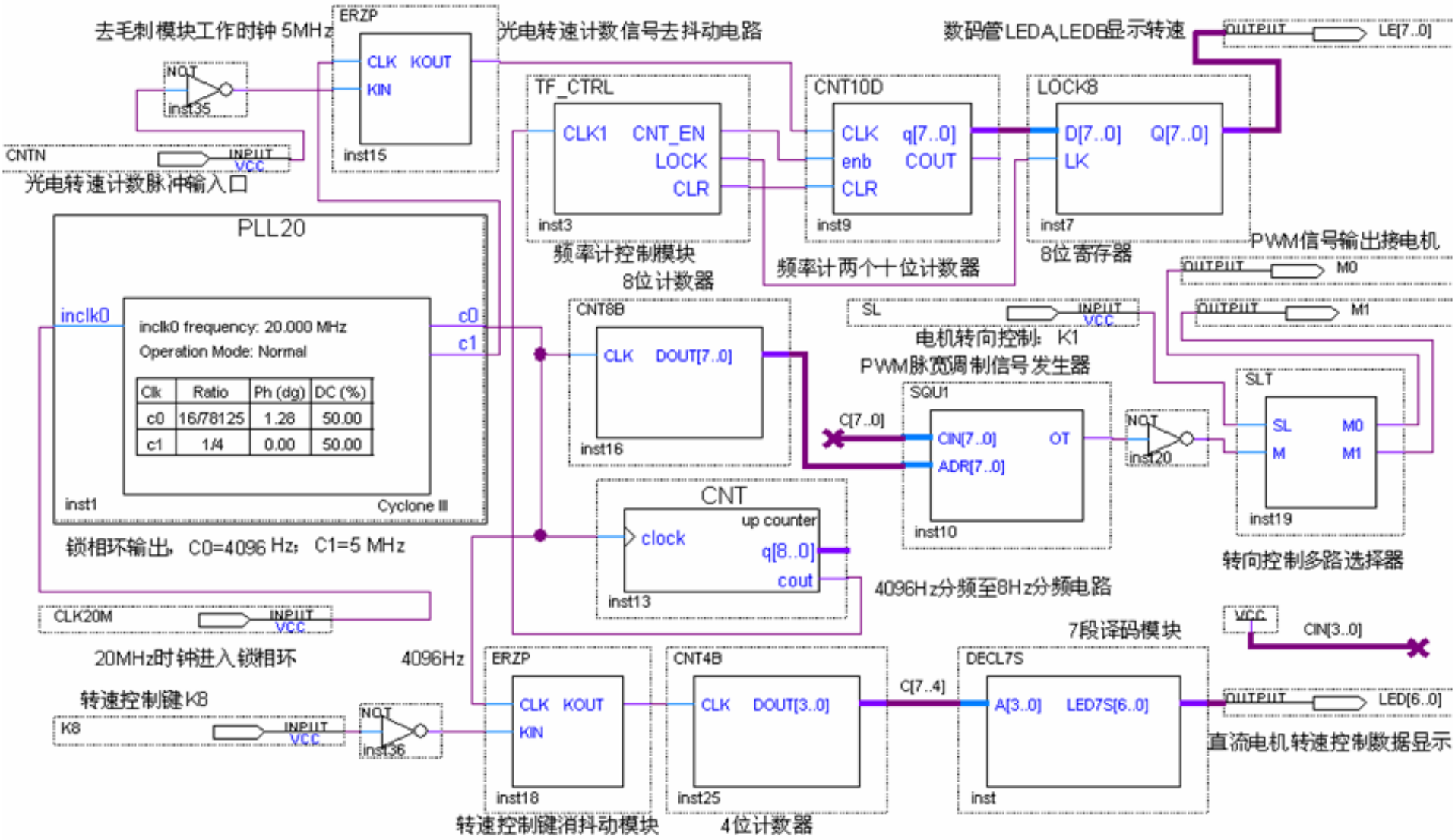

图 10-5 直流电机驱动控制电路顶层设计

实验与设计

【例 10-31】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SQU1 IS
   PORT ( CIN,ADR  :   IN STD_LOGIC_VECTOR(7 DOWNTO 0);
             OT   : OUT STD_LOGIC  );
END SQU1;
ARCHITECTURE BHV OF SQU1 IS
   BEGIN
 PROCESS(CIN)  BEGIN
    IF (ADR<CIN)  THEN OT<='0' ; ELSE   OT<='1' ;  END IF;
 END PROCESS;
END BHV ;
```

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

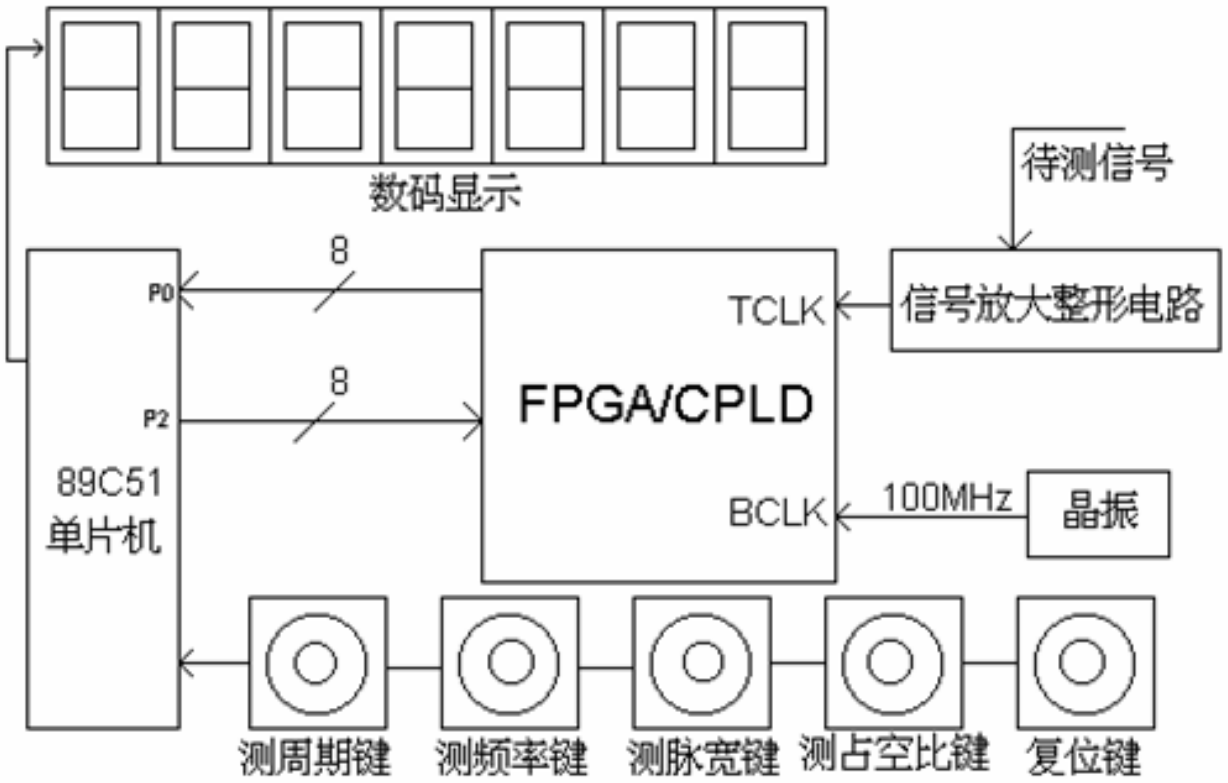

图 10-6 频率计主系统电路组成

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

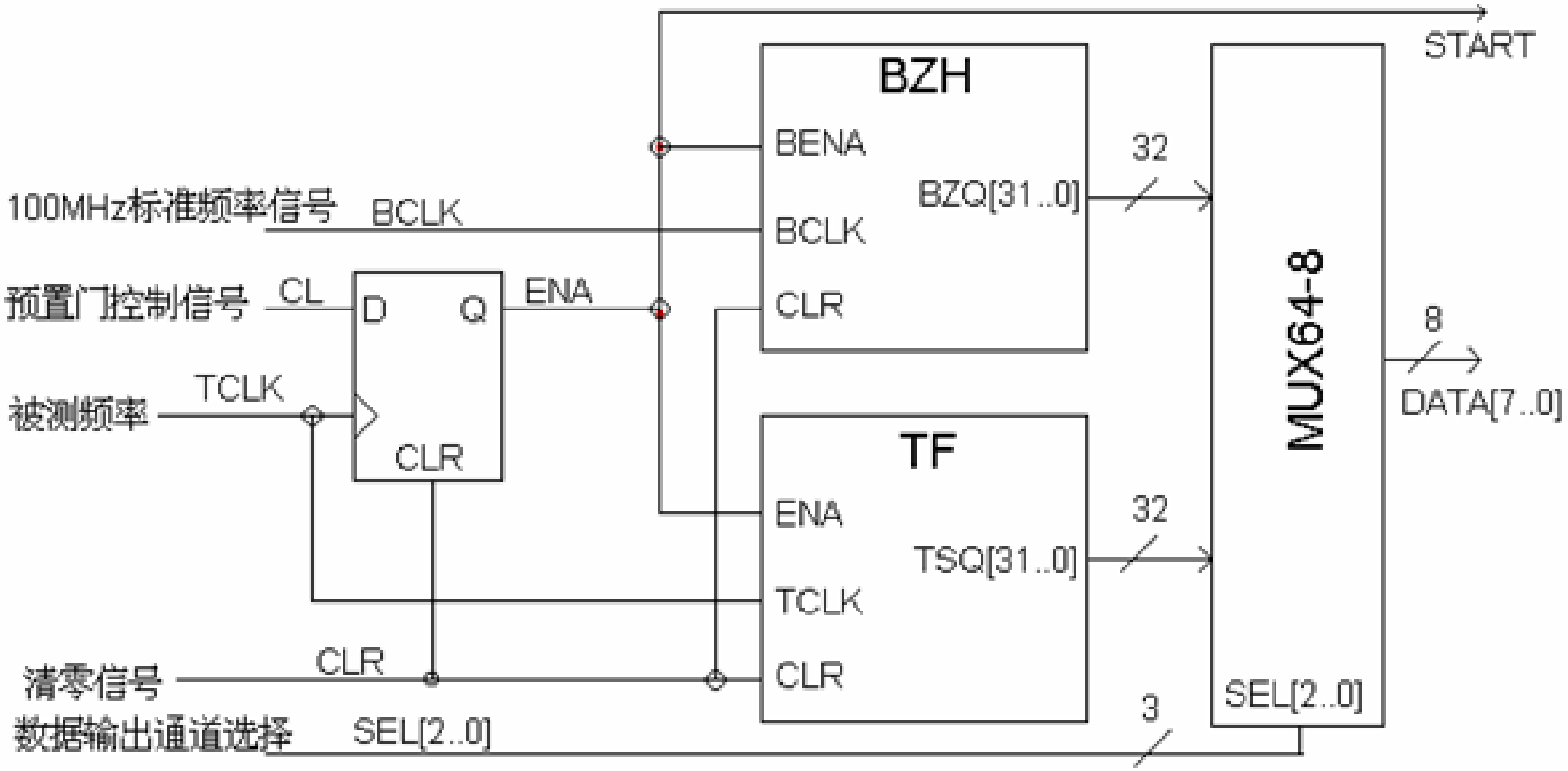

图 10-7 等精度频率计主控结构

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

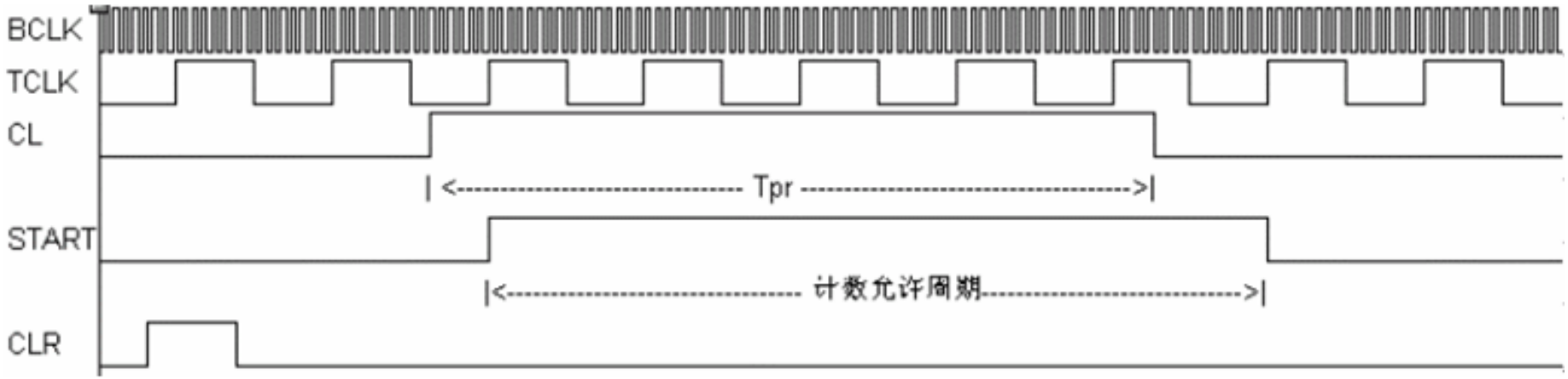

图 10-8 频率计测控时序

【例 10-32】

```
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 FTEST2 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
                   --的时间控制，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); --数据读出选择控制
     DATA : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)); --8 位数据读出
END FTEST2;
```

接下页

接上页

```
ARCHITECTURE behav OF FTEST2 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
       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) ;
```

接下页

```
BZH :  PROCESS(BCLK, CLR)  BEGIN     --标准频率测试计数器，标准计数器
```

接上页

```
      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)  BEGIN    --待测频率计数器，测频计数器
    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;
  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
```

接下页

接上页

```
      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
      '0' ;            --高电平时，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，SPUL 为 1 时测频率
      PUL WHEN SPUL='0' ELSE   --SPUL 为 0 时测脉宽和占空比
      PUL ;
END behav;
```

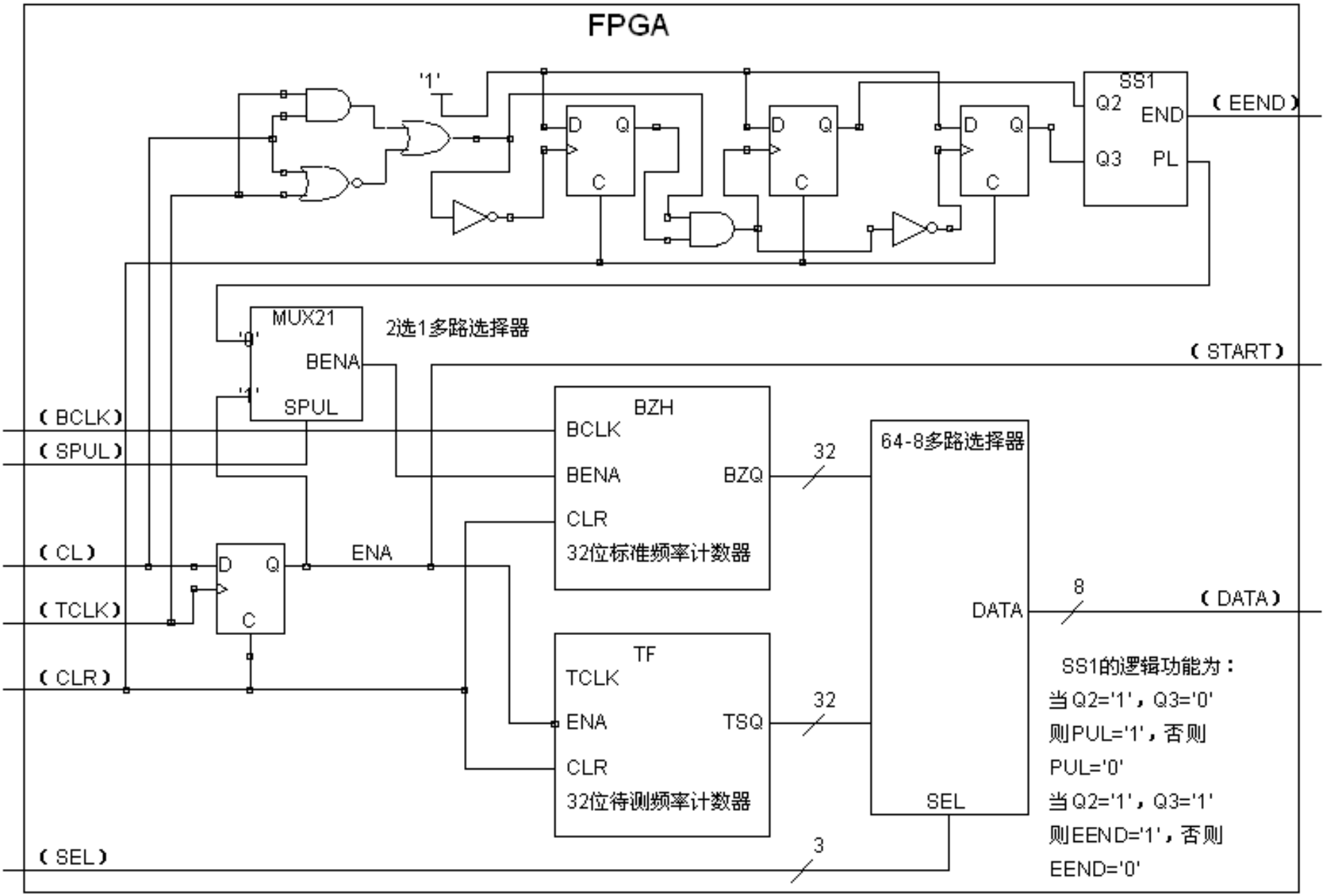

图 10-9 例 10-32 的逻辑图

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

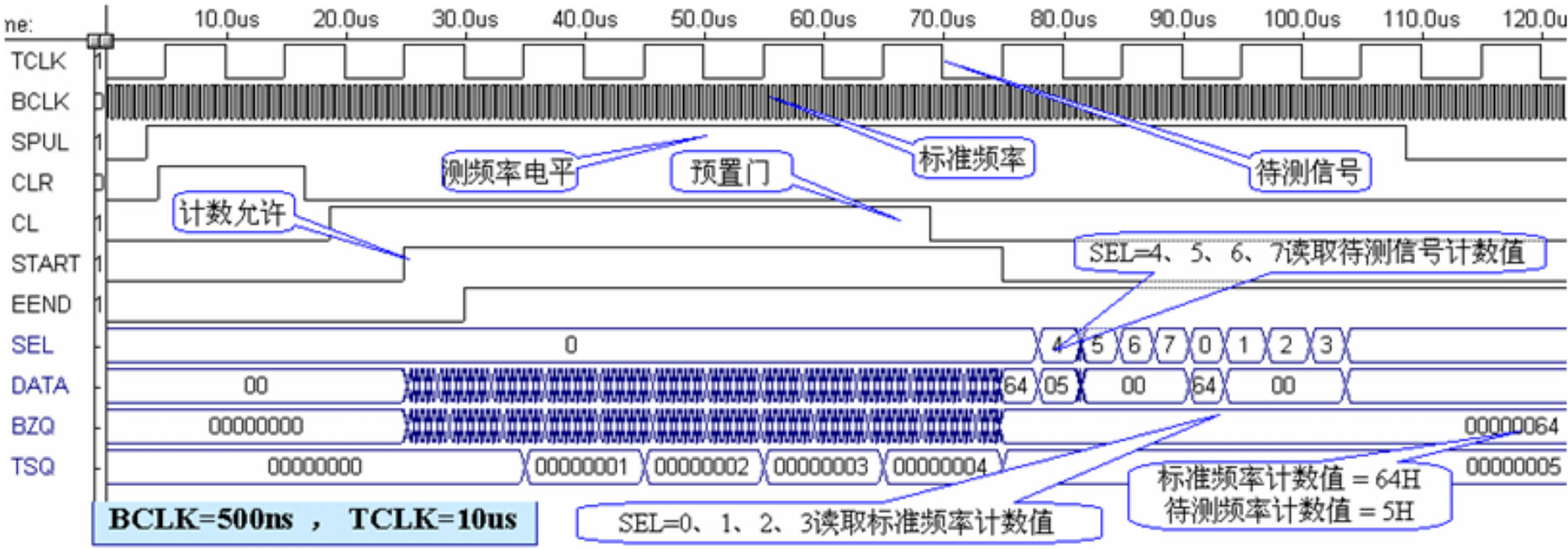

图 10-10 等精度频率计测频时序图

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

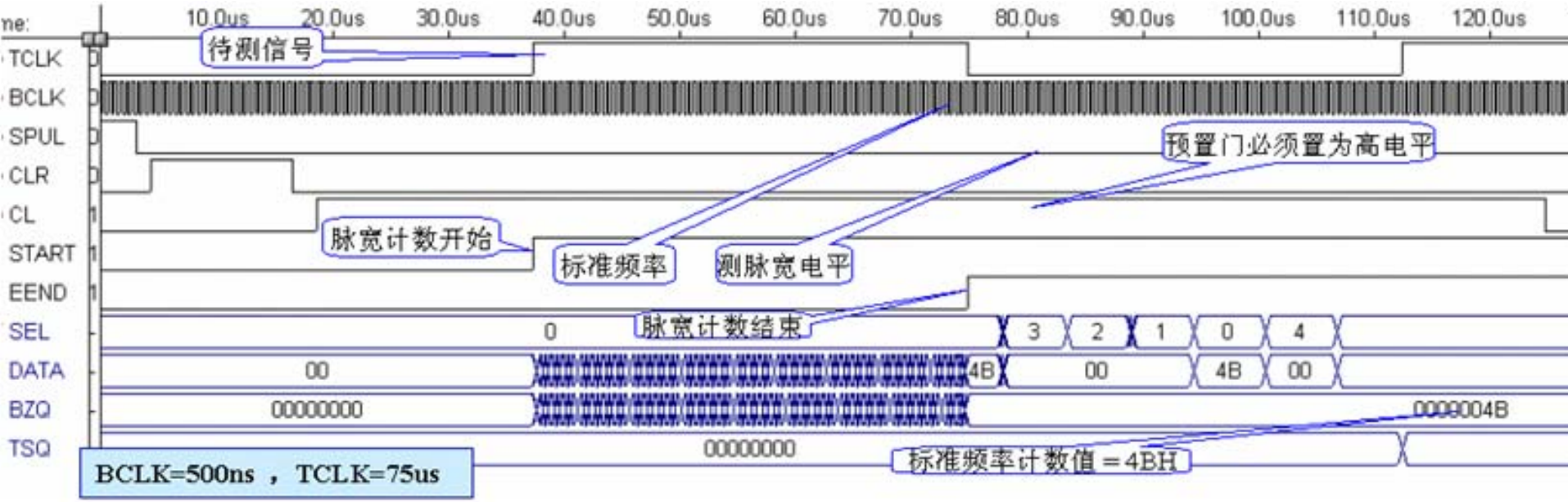

图 10-11 等精度频率计测脉宽时序图

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

$$占空比 = \frac{N1}{N1+N2} \times 100\% \qquad (10\text{-}1)$$

$$相位差 = (N1/(N1+N2)) \times 360^{\circ} \qquad (10\text{-}2)$$

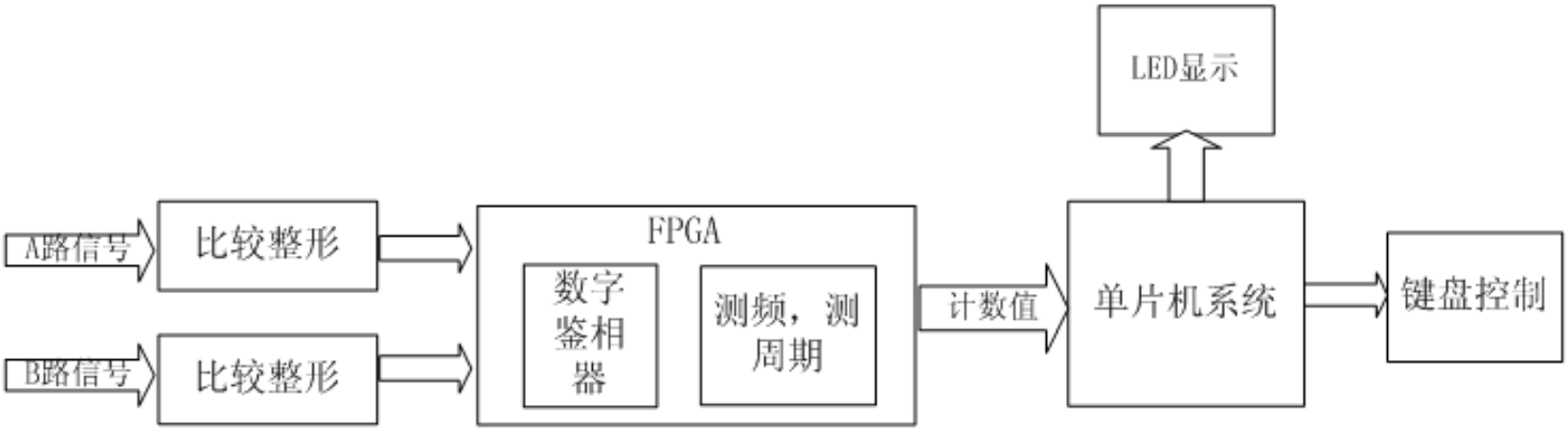

图 10-12 测相仪模型

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

（1）实验目的：

（2）实验原理：

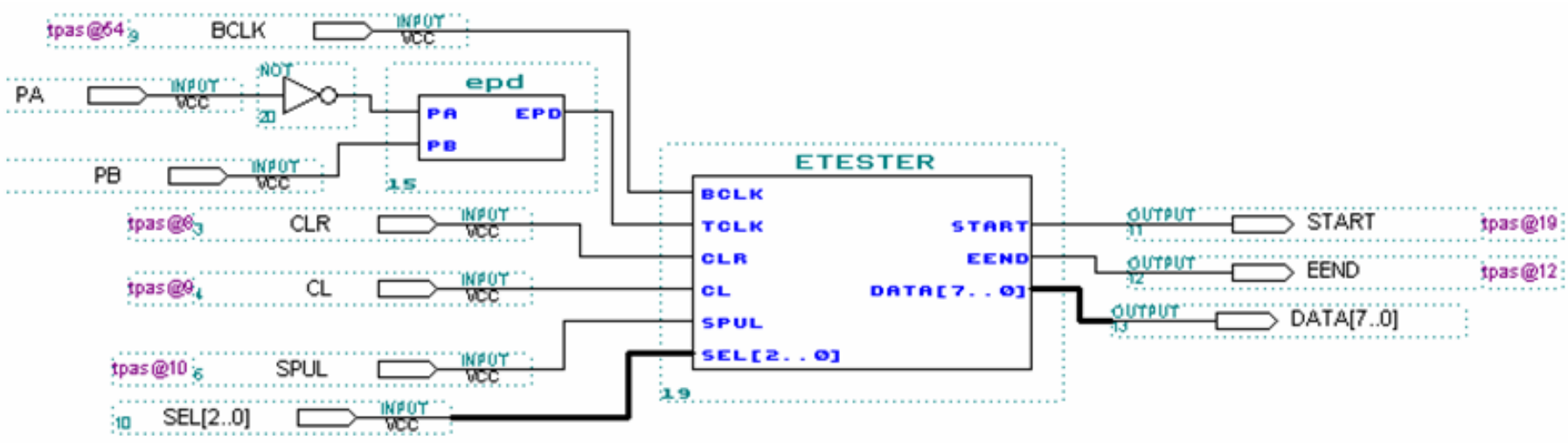

图 10-13 测相仪电路原理图

实验与设计

10-2 等精度频率/脉宽/占空比/相位多功能测试仪设计

(3) 实验内容**1**：。

(4) 实验内容**2**：本项设计的另一同等示例是：
/KX_7C5EE+/EXPERIMENTs/EXP18_KX8051_FTEST_K4X4/。

(5) 实验内容**3**：

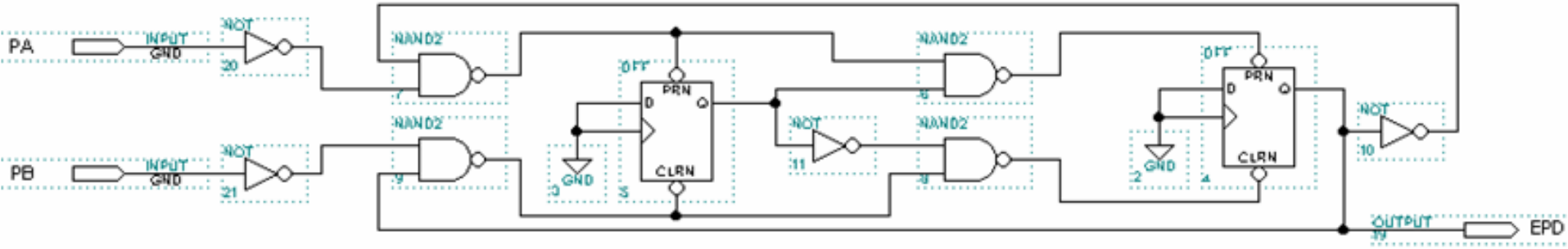

图 10-14 相位检测原理图 epd

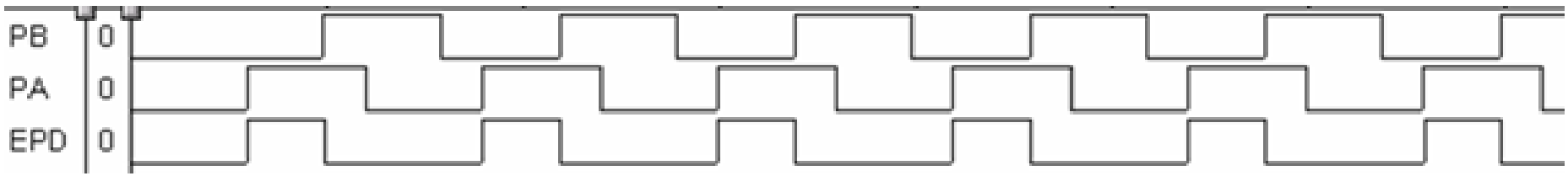

图 10-15 鉴相器 EPD 的仿真波形

实验与设计

10-3 PC机键盘经UART串口控制模型电子琴电路设计

（1）实验原理：

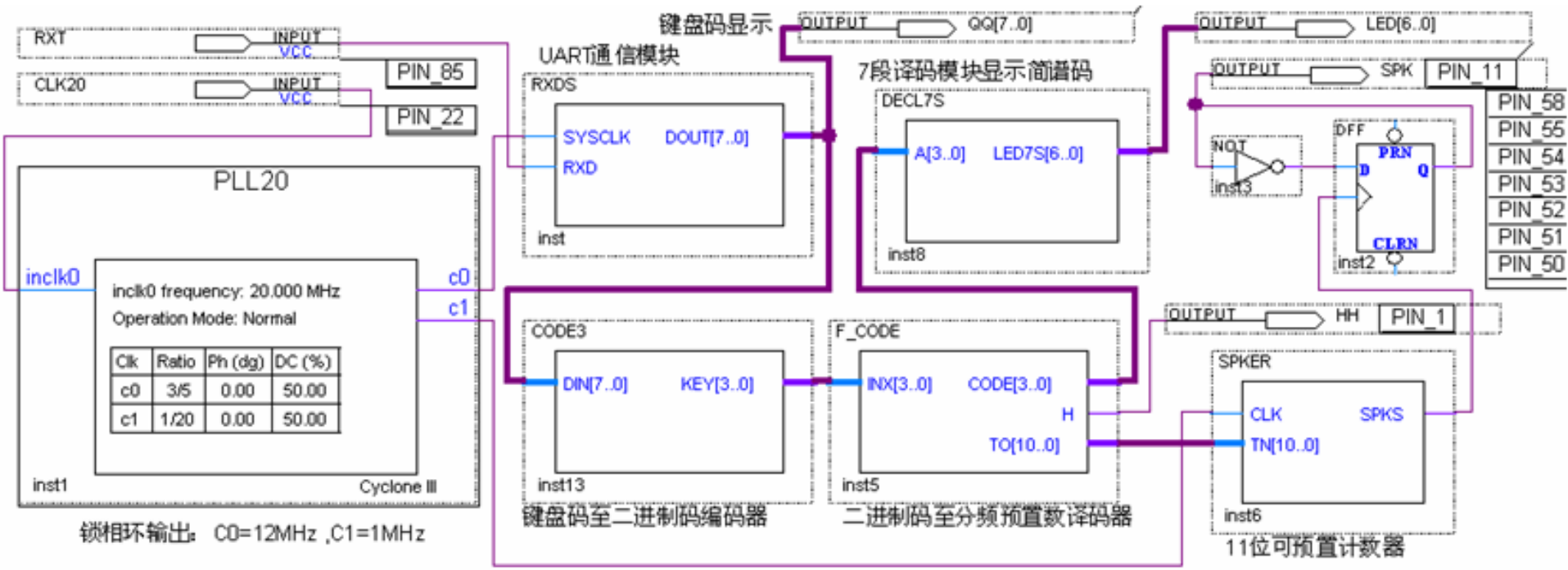

图 10-16 UART 串口控制模型电子琴电路顶层设计

实验与设计

10-3 PC机键盘经UART串口控制模型电子琴电路设计

（2）实验内容1：本项实验基于5E+系统的示例演示文件：
/KX_7C5EE+/EXPERIMENTs/EXP42_RS232_PIANO/ECHD。

（3）实验内容2：

（4）实验内容3：

（5）实验内容4：

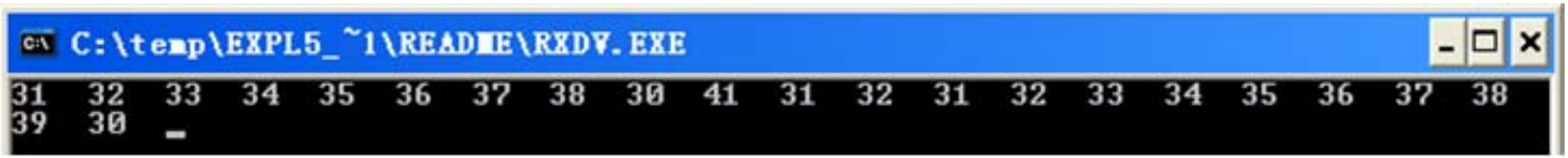

图 10-17 串口通信及键盘 ASCII 码显示程序窗口

实验与设计

10-4 AM幅度调制信号发生器设计

（1）实验原理：

$$F = F_{dr} \cdot (1 + F_{am} \cdot m)$$

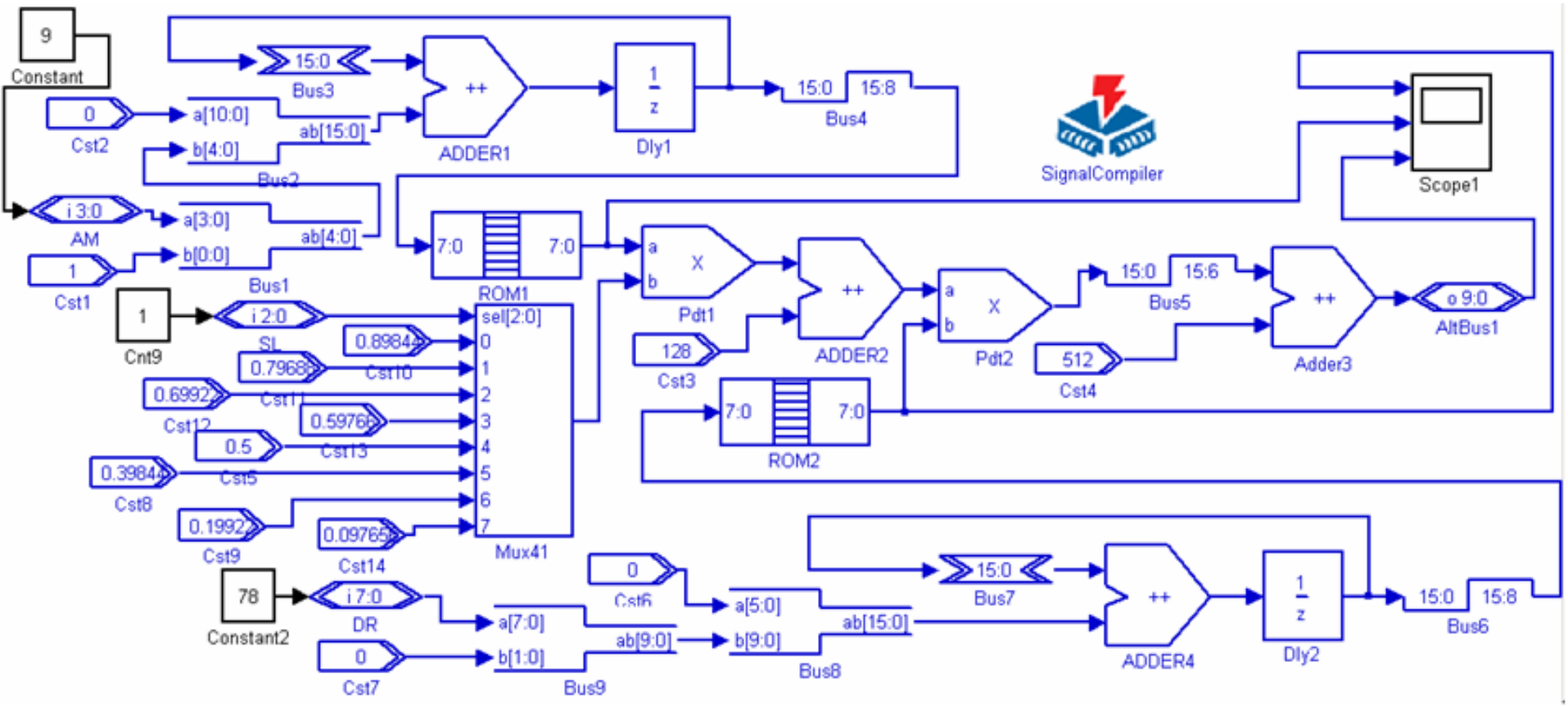

图 10-18 AM 信号发生器 DSP Builder/MATLAB Simulink 模型

实验与设计

10-4 AM幅度调制信号发生器设计

（2）实验任务1：基于5E+系统的演示示例设计是：
/KX_7C5EE+/DEMOs/EXPL10_DDS_Core_DAC0832/；或
/EXPL11_DDS_Core_HSpeed/

（3）实验任务2：

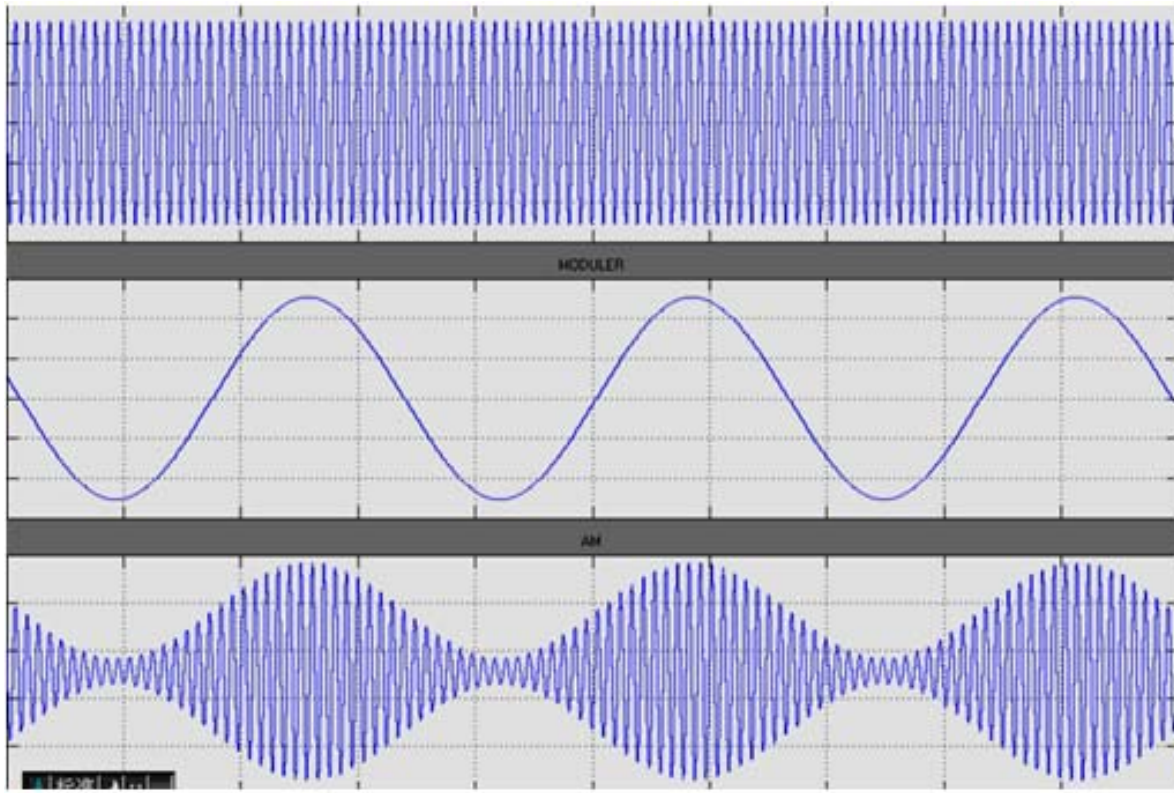

图 10-19 AM 模型仿真波形

实验与设计

10-5 单片全数字型DDS函数信号发生器综合设计实验

基本功能包括：

1、PSK、MSK、AM、FM、FSK；PSK信号发生器（调制信号、载波信号频率可设）；

2、移相信号发生器(两路频率同步可数控，相差可数控)；

3、里萨如图信号发生器(两路频率独立可数控，相差可数控)；

4、扫频信号源，要求最大单程扫频0.1Hz至10MHz，在此范围内，扫频频幅可设（最小0.05Hz）、扫速可设、扫频起始终止点可设、线性/对数扫频方式可选、内外扫频时钟可选；

5、三角波、锯齿波、方波（频率/占空比可数控）；

6、频率、脉宽、占空比、相位可测。基于5E+系统的演示示例：/KX_7C5EE+/DEMOs/EXP10_DDS_Core_DAC0832/。操作方法参考所附资料。

实验与设计

10-6 正交幅度调制与解调系统实现

（1）实验目的：

（2）实验原理：

$$I(t)=a(t)\cos\omega_0 t \qquad Q(t)=b(t)\sin\omega_0 t$$

$$X(t)=I(t)+Q(t)=a(t)\cos\omega_0 t+b(t)\sin\omega_0 t$$

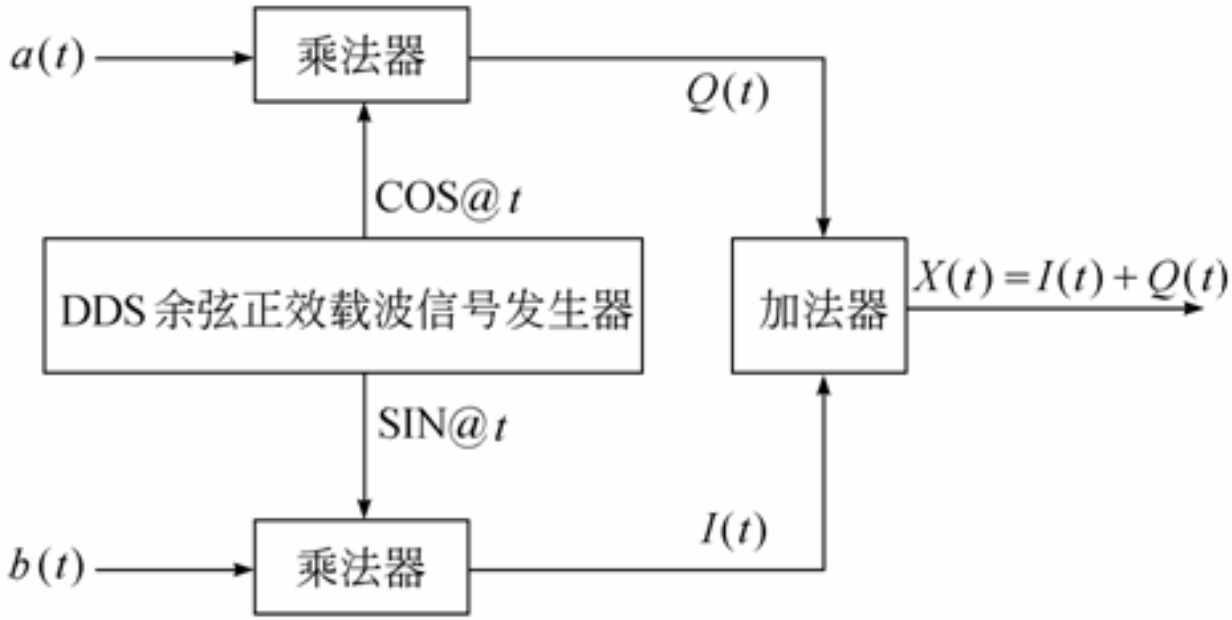

图 10-20 正交幅度调制原理图

实验与设计

10-6 正交幅度调制与解调系统实现

（1）实验目的：

（2）实验原理：

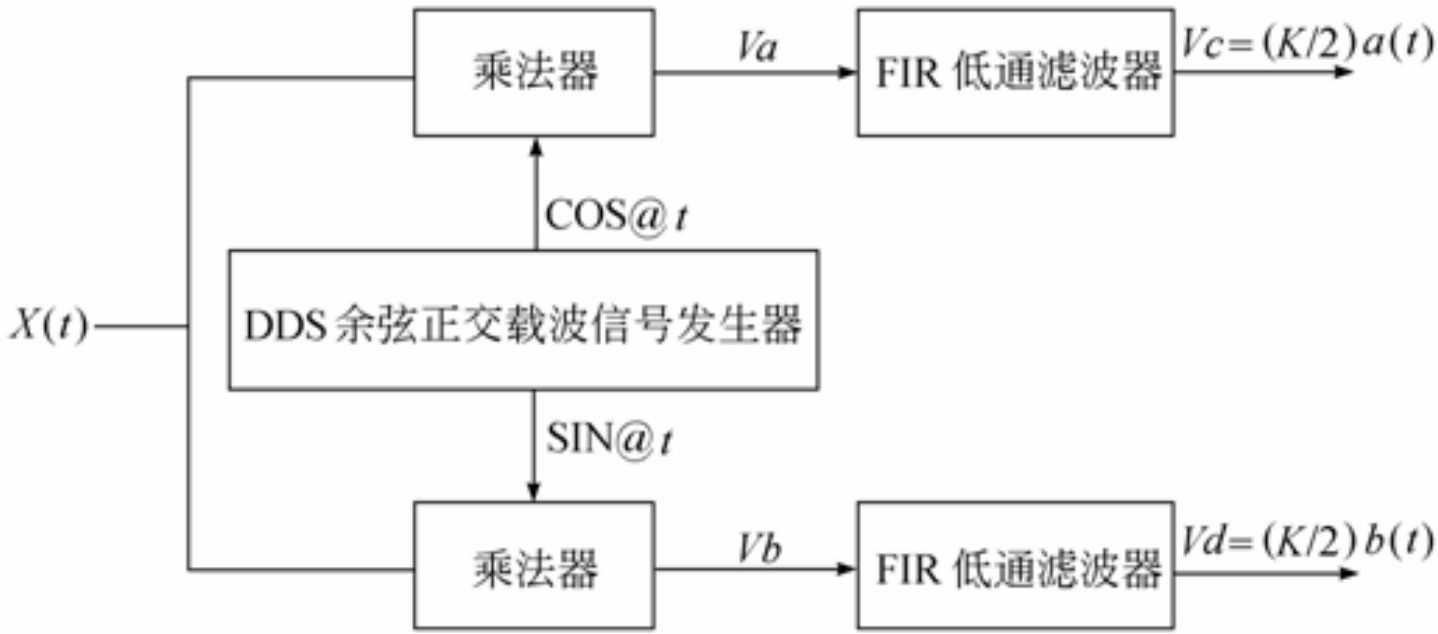

图 10-21 正交幅度信号解调原理图

实验与设计

10-6 正交幅度调制与解调系统实现

（3）实验内容1：

（4）实验内容2：

$$V_a = X(t)\cos\omega_0 t = a(t)\cos^2\omega_0 t + b(t)\sin\omega_0 t\cos\omega_0 t = \frac{1}{2}a(t) + \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}b(t)\sin 2\omega_0 t$$

$$V_b = X(t)\sin\omega_0 t = b(t)\sin^2\omega_0 t + a(t)\sin\omega_0 t\cos\omega_0 t = \frac{1}{2}b(t) - \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}a(t)\sin 2\omega_0 t$$

$$V_c = \frac{K}{2}a(t) \qquad V_d = \frac{K}{2}b(t)$$