第7章

VHDL深入

7.1 进程中的信号赋值与变量赋值

表 7-1　信号与变量赋值语句功能的比较

比较对象	信号 SIGNAL	变量 VARIABLE
基本用法	用于作为电路中的信号连线	用于作为进程中局部数据存储单元
适用范围	在整个结构体内的任何地方都能适用	只能在所定义的进程中使用
行为特性	在进程的最后才对信号赋值，有延时	立即赋值，无延时
与 Verilog 对比	信号赋值类似于非阻塞式赋值	变量赋值类似于阻塞式赋值

7.1 进程中的信号赋值与变量赋值

【例 7-1】使用变量赋值的时序模块设计

```
ARCHITECTURE bhv OF DFF1 IS
  BEGIN
  PROCESS (CLK)
    VARIABLE Q1 : STD_LOGIC;
  BEGIN
    IF  CLK'EVENT AND CLK='1'
        THEN Q1 := D;  END IF;
    Q <= Q1;
  END PROCESS;
END;
```

【例 7-2】使用信号赋值的时序模块设计

```
ARCHITECTURE bhv OF DFF2 IS
    SIGNAL Q1 : STD_LOGIC;
BEGIN
  PROCESS (CLK)
  BEGIN
    IF  CLK'EVENT AND CLK='1'
        THEN  Q1 <= D;  END IF;
   END PROCESS;
        Q <= Q1;
END;
```

7.1 进程中的信号赋值与变量赋值

【例 7-3】

```
ARCHITECTURE bhv OF DFF1 IS
SIGNAL A,B : STD_LOGIC;
BEGIN
PROCESS (CLK) BEGIN
IF  CLK'EVENT AND CLK='1'  THEN
A <= D;     B <= A;
Q <= B;
END IF;
END PROCESS;
END;
```

【例 7-4】

```
ARCHITECTURE bhv OF DFF1 IS
BEGIN
PROCESS (CLK)
VARIABLE A,B : STD_LOGIC;
BEGIN
IF  CLK'EVENT AND CLK='1'  THEN
 A := D;       B := A;
 Q <= B;       END IF;
END PROCESS;
END;
```

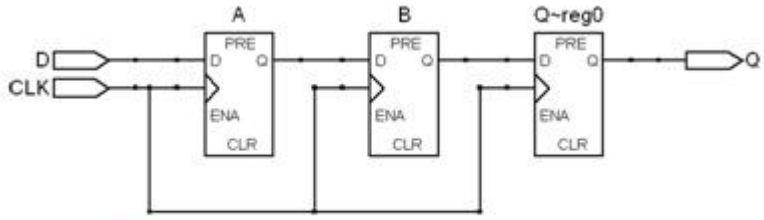

图 7-1　例 7-3 的 RTL 电路图

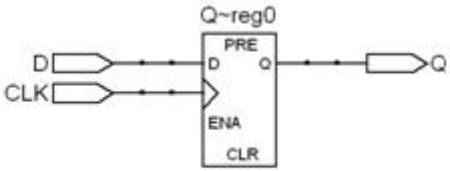

图 7-2　D 触发器电路

7.1 进程中的信号赋值与变量赋值

【例 7-5】

```
PROCESS(in1, in2, . . .)                    --此进程在5ns+δ时刻被启动
VARIABLE c1, . . . : STD_LOGIC_VECTOR(3 DOWNTO 0);
  BEGIN
...
        e1 <= "1010";                       --第 3 行
           ...
        c1 := "0011";                       --第 10行
           ...
END PROCESS;                                --在5ns+2δ时刻结束进程
```

7.1 进程中的信号赋值与变量赋值

【例 7-6】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
PORT (i0, i1, i2, i3, a, b : IN STD_LOGIC;  q : OUT STD_LOGIC);
END mux4;
ARCHITECTURE body_mux4 OF mux4 IS
signal muxval : integer range 7 downto 0;
BEGIN
process(i0,i1,i2,i3,a,b)  begin
  muxval <= 0;
if (a = '1') then   muxval <= muxval + 1; end if;
if (b = '1') then   muxval <= muxval + 2; end if;
case muxval is
    when 0 => q <= i0;
    when 1 => q <= i1;
    when 2 => q <= i2;
    when 3 => q <= i3;
    when others => null;
end case;
end process;
END body_mux4;
```

7.1 进程中的信号赋值与变量赋值

【例 7-7】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
PORT (i0, i1, i2, i3, a, b : IN STD_LOGIC;  q : OUT STD_LOGIC);
END mux4;
ARCHITECTURE body_mux4 OF mux4 IS
BEGIN
process(i0,i1,i2,i3,a,b)
variable muxval : integer range 7 downto 0;
begin
                    muxval := 0;
if (a = '1') then  muxval := muxval + 1;  end if;
if (b = '1') then  muxval := muxval + 2;  end if;
case muxval is
    when 0 => q <= i0;
    when 1 => q <= i1;
    when 2 => q <= i2;
    when 3 => q <= i3;
    when others => null;
end case;
end process;
END body_mux4;
```

7.1 进程中的信号赋值与变量赋值

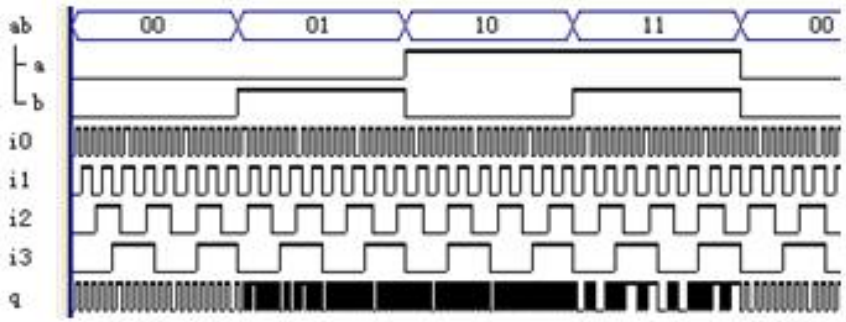

图 7-3　例 7-6 的错误工作时序

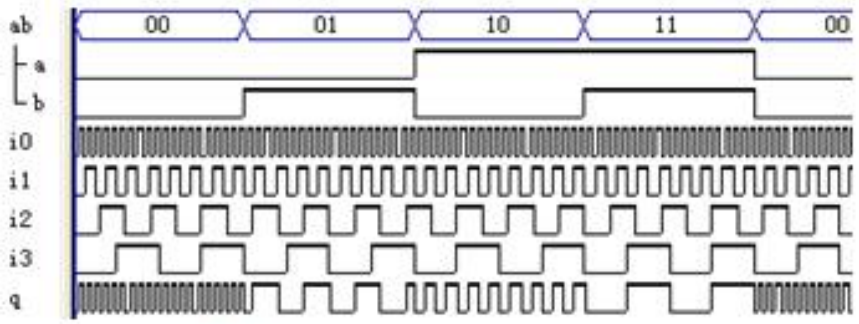

图 7-4　例 7-7 的正确工作时序

7.2 含高阻输出的电路设计

7.2.1 三态门设计

【例 7-8】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY tri_s IS
  port (enable : IN STD_LOGIC;
        datain :  IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        dataout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
END tri_s ;
ARCHITECTURE bhv OF tri_s IS
BEGIN
PROCESS(enable,datain)     BEGIN
   IF enable='1' THEN   dataout <= datain ;
     ELSE dataout <="ZZZZZZZZ" ;  END IF ;
END PROCESS;
END bhv;
```

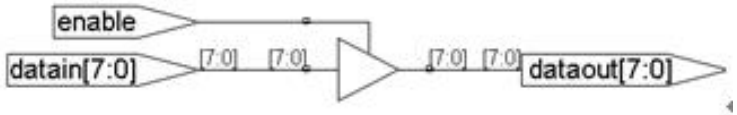

图 7-5　8 位三态控制门电路

7.2 含高阻输出的电路设计

7.2.2 双向端口的设计方法

【例 7-9】

```
library ieee;
use ieee.std_logic_1164.all;
entity tri_state is
port (control : in std_logic;
      in1: in std_logic_vector(7 downto 0);
       q : inout std_logic_vector(7 downto 0);
       x : out std_logic_vector(7 downto 0) );
end tri_state;
architecture body_tri of tri_state is
begin
process(control,q,in1)  begin
if (control='0') then   x<=q; else  q<=in1;   x<="ZZZZZZZZ"; end if;
end process;
end body_tri;
```

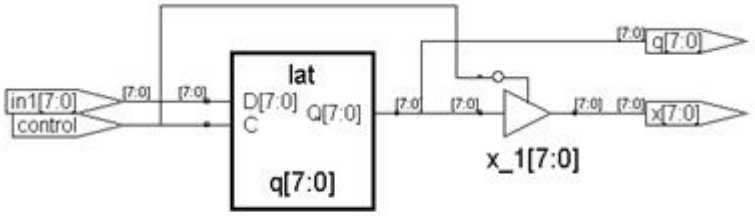

图 7-6　例 7-9 的综合结果

7.2 含高阻输出的电路设计

7.2.2 双向端口的设计方法

【例 7-10】

```
... --以上部分同上例
 process(control,q,in1)   begin
  if (control='0') then  x <= q ;   q <="ZZZZZZZZ";
                   else  q <= in1;  x <="ZZZZZZZZ";
  end if;
 end process;
end body_tri;
```

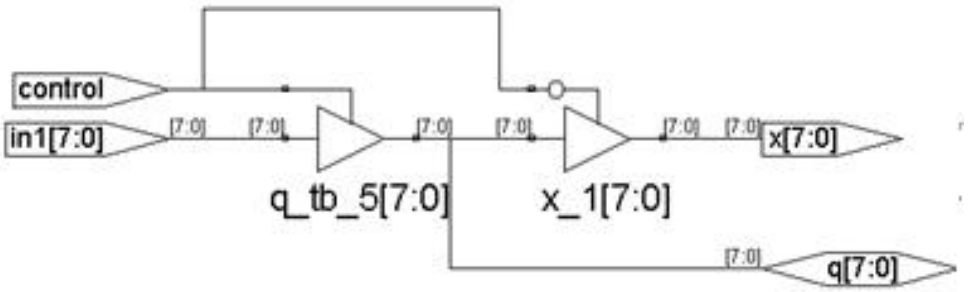

图 7-7　例 7-10 的综合结果

7.2 含高阻输出的电路设计

7.2.3 三态总线电路设计

【例 7-11】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY tristate2 IS
    port ( input3, input2, input1, input0 :
              IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    enable : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    output : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
END tristate2 ;
ARCHITECTURE multiple_drivers OF tristate2 IS
BEGIN
  PROCESS(enable,input3, input2, input1, input0 )   BEGIN
      IF enable = "00" THEN output <= input3 ;
        ELSE output <=(OTHERS =>'Z'); END IF ;
      IF enable = "01" THEN output <= input2 ;
        ELSE output <=(OTHERS =>'Z'); END IF ;
      IF enable = "10" THEN output <= input1 ;
        ELSE output <=(OTHERS =>'Z'); END IF ;
      IF enable = "11" THEN output <= input0 ;
        ELSE output <=(OTHERS =>'Z');  END IF ;
END PROCESS;
END multiple_drivers;
```

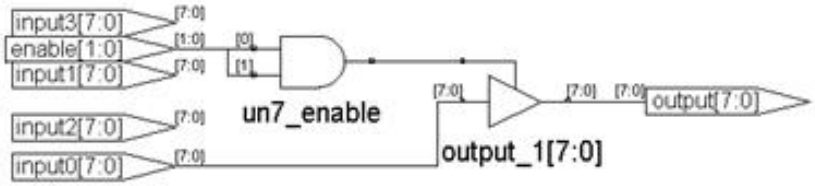

图 7-8　例 7-11 错误的综合结果

7.2 含高阻输出的电路设计

7.2.3 三态总线电路设计

【例 7-12】

```
library ieee;
use ieee.std_logic_1164.all;
entity tri2 is
port (ctl : in  std_logic_vector(1 downto 0);
        datain1, datain2, datain3, datain4 :
            in std_logic_vector(7 downto 0);
       q : out std_logic_vector(7 downto 0)   );
end tri2;
architecture body_tri of tri2 is
begin
 q <= datain1   when ctl="00" else  (others =>'Z') ;
 q <= datain2   when ctl="01" else  (others =>'Z') ;
 q <= datain3   when ctl="10" else  (others =>'Z') ;
 q <= datain4   when ctl="11" else  (others =>'Z') ;
end body_tri;
```

7.3 资源优化

7.3.1 资源共享

【例 7-13】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY multmux IS
    PORT (A0, A1, B  : IN  std_logic_vector(3 downto 0);
                  s  : IN  std_logic;
                  R  : OUT std_logic_vector(7 downto 0));
END multmux;
ARCHITECTURE rtl OF multmux IS
BEGIN
    process(A0,A1,B,s)    begin
       if(s='0') then   R<=A0 * B;  else  R<=A1 * B;  end if;
    end process;
END rtl;
```

7.3 资源优化

7.3.1 资源共享

【例 7-14】

```
ARCHITECTURE rtl OF muxmult IS  --以上部分与例 9-1 相同
    signal temp : std_logic_vector(3 downto 0);
BEGIN
    process(A0,A1,B,s)    begin
        if(s='0') then    temp<=A0; else  temp<=A1;   end if;
        R <= temp * B;
    end process;
END rtl;
```

7.3 资源优化

7.3.1 资源共享

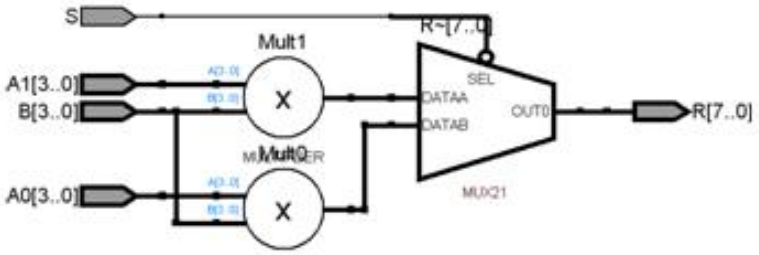

图 7-9 先乘后选择的设计方法 RTL 结构

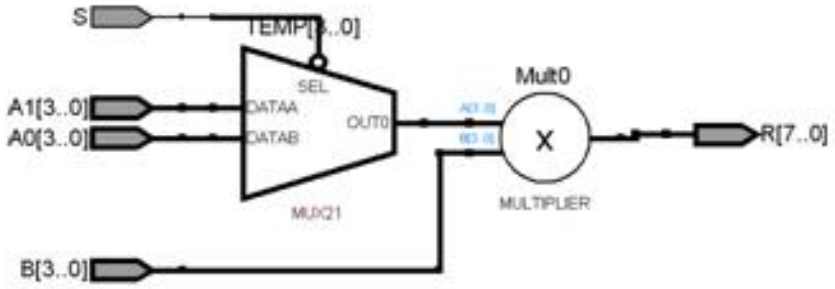

图 7-10 先选择后乘的设计方法 RTL 结构

7.3 资源优化

7.3.1 资源共享

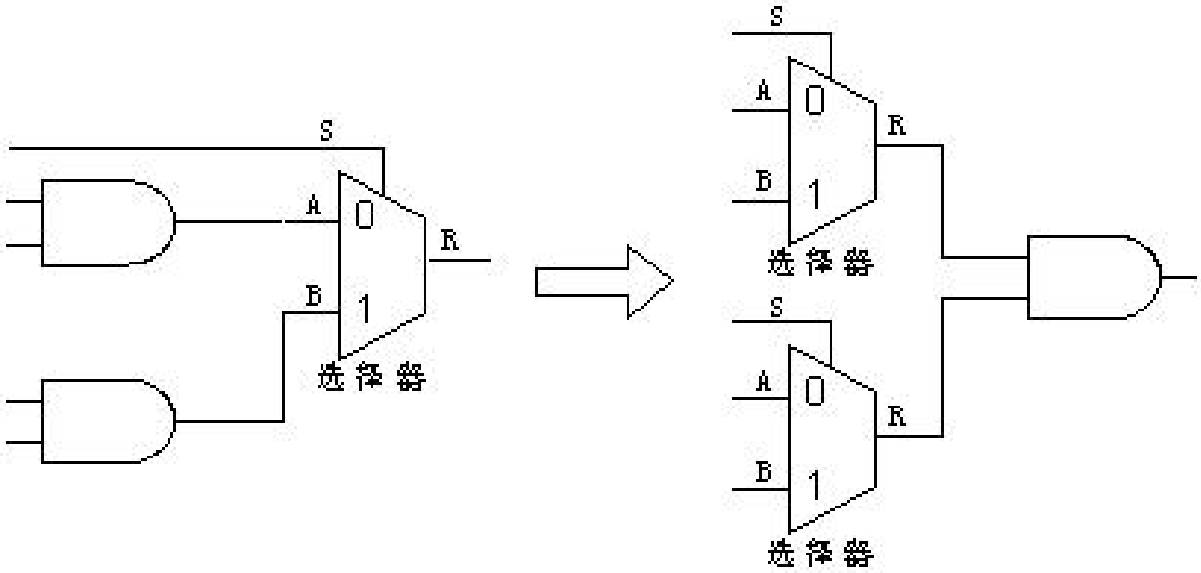

图 7-11　资源共享反例

7.3 资源优化

7.3.2 逻辑优化

【例 7-15】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY mult1 IS
    PORT(clk : in std_logic;
         ma : In std_logic_vector(11 downto 0);
         mc : out std_logic_vector(23 downto 0));
END mult1;
ARCHITECTURE rtl OF mult1 IS
    signal ta, tb : std_logic_vector(11 downto 0);
BEGIN
process(clk)  begin
    if(clk'event and clk='1') then
       ta<=ma;  tb<="100110111001";   mc<=ta * tb;   end if;
end process;
END rtl;
```

7.3 资源优化

7.3.2 逻辑优化

【例 7-16】

```
... --以上同例7-15
ARCHITECTURE rtl OF mult1 IS
    signal ta : std_logic_vector(11 downto 0);
    constant tb : std_logic_vector(11 downto 0):="100110111001";
BEGIN
process(clk) begin
    if(clk'event and clk='1') then  ta<=ma; mc<=ta*tb;  end if;
end process;
END rtl;
```

7.3 资源优化

7.3.3 串行化

$$yout = a_0 \times b_0 + a_1 \times b_1 + a_2 \times b_2 + a_3 \times b_3$$

【例 7-17】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY pmultadd IS
    PORT(a0,a1,a2,a3 : in std_logic_vector(7 downto 0);
         b0,b1,b2,b3 : in std_logic_vector(7 downto 0);
         yout : out std_logic_vector(15 downto 0);
         clk : in std_logic);
END pmultadd;
ARCHITECTURE p_arch OF pmultadd IS
BEGIN
process(clk) begin
    if(clk'event and clk = '1') then
       yout <= ((a0*b0)+(a1*b1))+((a2*b2)+(a3*b3));  end if;
end process;
END p_arch;
```

【例 7-18】

```
...--以上部分与例 7-17 相同
          clk,  start : in std_logic);
END pmultadd;
ARCHITECTURE s_arch OF pmultadd IS
   signal cnt : std_logic_vector(2 downto 0);
   signal tmpa, tmpb : std_logic_vector(7 downto 0);
   signal  tmp,  ytmp : std_logic_vector(15 downto 0);
  BEGIN
tmpa <= a0 when cnt = 0 else
        a1 when cnt = 1 else
        a2 when cnt = 2 else
        a3 when cnt = 3 else    a0;
tmpb <= b0 when cnt = 0 else
        b1 when cnt = 1 else
        b2 when cnt = 2 else
        b3 when cnt = 3 else    b0;
tmp <= tmpa * tmpb;
process(clk) begin
   if(clk'event and clk = '1') then
      if (start='1') then  cnt<="000"; ytmp<=(others=>'0');
      elsif (cnt<4) then  cnt<=cnt+1;  ytmp<=ytmp+tmp;
      elsif (cnt=4) then  yout<=ytmp;
      end if;    end if;
end process;
END s_arch;
```

7.4 速 度 优 化

7.4.1 流水线设计

图 7-12 未使用流水线

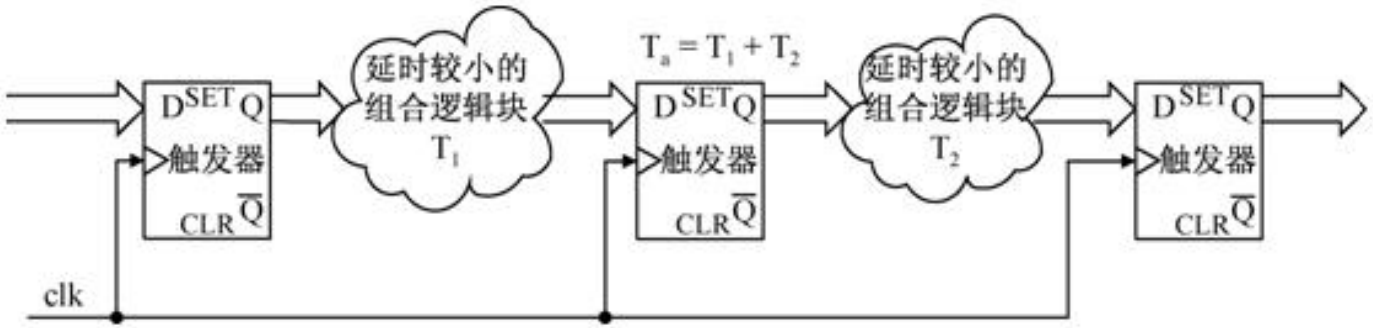

图 7-13 使用流水线结构

7.4 速度优化

7.4.1 流水线设计

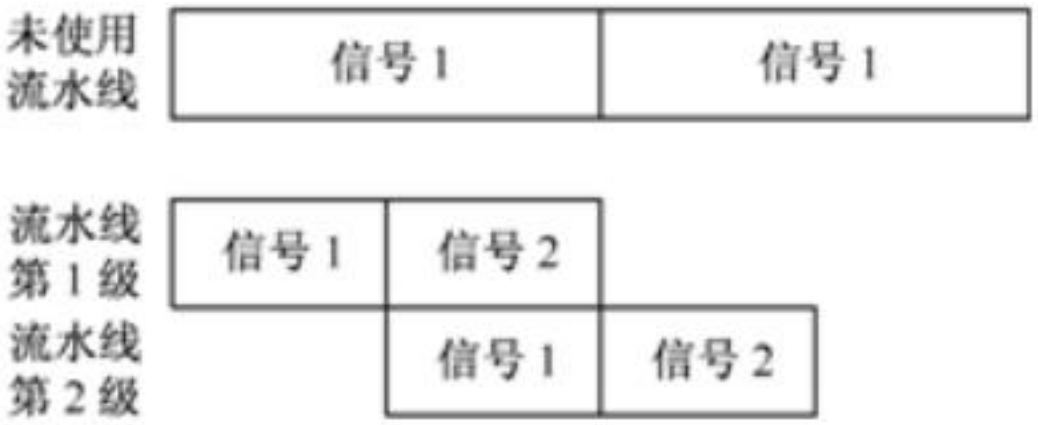

图 7-14　流水线工作图示

7.4 速度优化

7.4.1 流水线设计

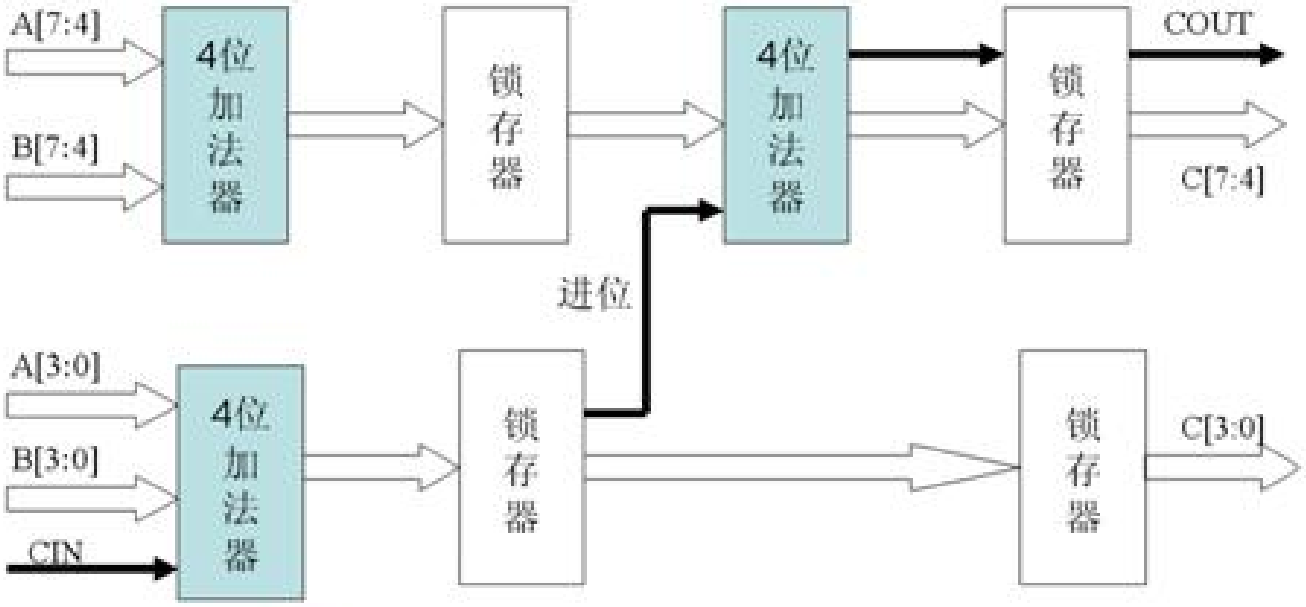

图 7-15　8 位加法器流水线工作图示

【例 7-19】普通加法器，EP3C10 FPGA 综合结果：LCs=10,REG=0,T=7.748ns.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY ADDER8 IS
   PORT (A, B : IN std_logic_vector(7 downto 0);
        CLK,CIN  : IN  std_logic;       COUT  : OUT  std_logic;
           SUM  : OUT std_logic_vector(7 downto 0));
END ADDER8 ;
ARCHITECTURE rtl OF ADDER8  IS
SIGNAL SUMC,A0,B0 : std_logic_vector(8 downto 0);
BEGIN
     A0<='0'& A ; B0<='0' & B ;
  process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN   SUMC <= A0+B0+CIN;  END IF;
  end process;
     COUT<=SUMC(8) ; SUM<=SUMC(7 downto 0);
END rtl;
```

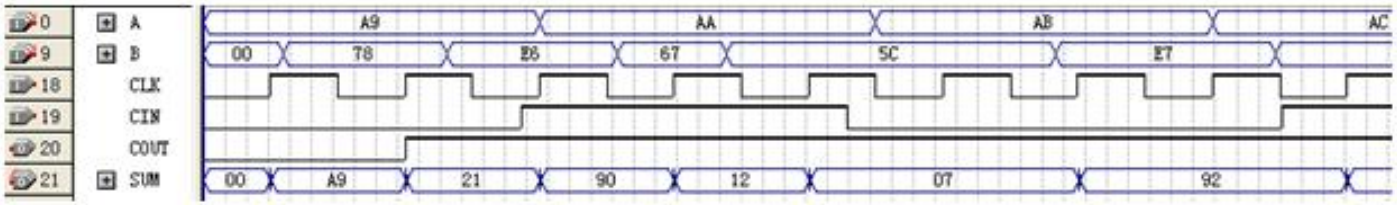

图 7-16　例 7-19 的时序仿真波形

【例 7-20】流水线加法器，EP3C10 综合结果：CLK=275MHz,T=3.63ns，LCs=24，REG=22。

```
...--以上部分与例 7-19 相同
ARCHITECTURE rtl OF ADDER8  IS
SIGNAL SUMC,A9,B9 : std_logic_vector(8 downto 0);
SIGNAL AB5,A5,B5,TA,TB,S : std_logic_vector(4 downto 0);
BEGIN
     A5<='0'& A(3 downto 0); B5<='0'& B(3 downto 0);
 process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN
       AB5<=A5+B5+CIN;  SUM(3 downto 0)<=AB5(3 downto 0); END IF;
   end process;
 process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN
     S<=('0'& A(7 downto 4))+('0'& B(7 downto 4))+ AB5(4); END IF;
   end process;
     COUT<=S(4) ; SUM(7 downto 4)<=S(3 downto 0);
END rtl;
```

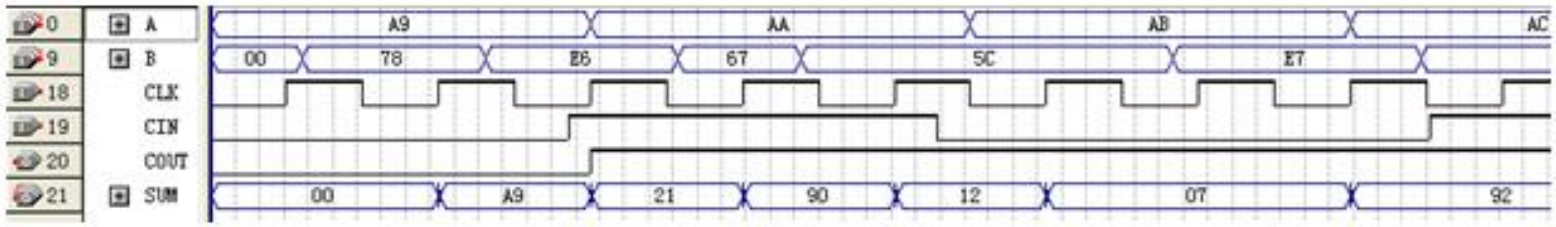

图 7-17　例 7-20 的时序仿真波形

7.4 速 度 优 化

7.4.2 关键路径法

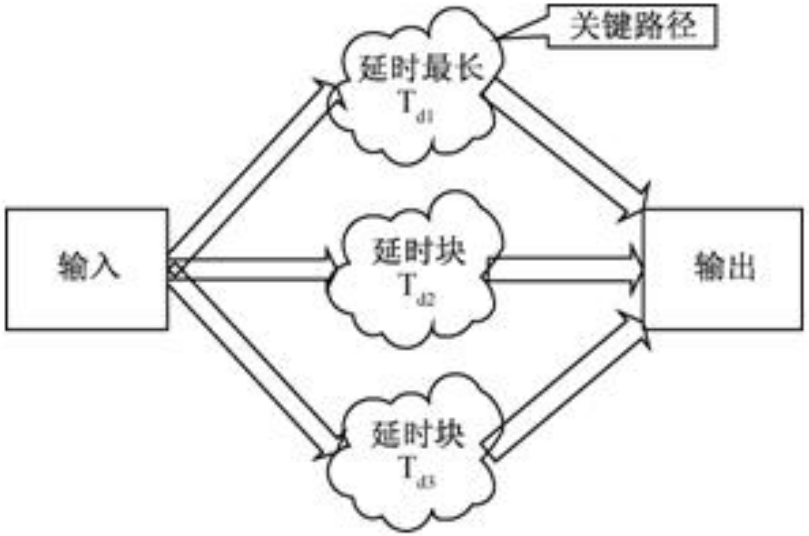

图 7-18 关键路径示意

7.5 仿 真 延 时

7.5.1 固有延时

```
B <= A AFTER 20ns;   --固有延时模型
```

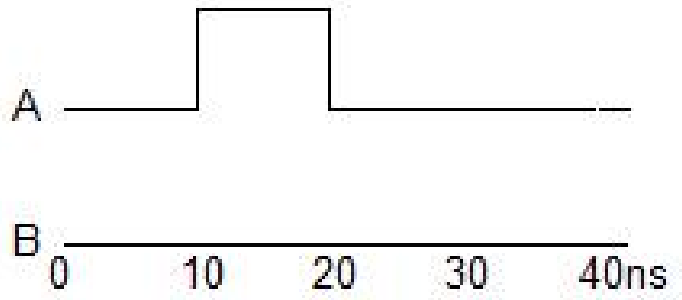

图 7-19 固有延时输入输出波形

7.5 仿 真 延 时

7.5.2 传输延时

B <= TRANSPORT A AFTER 20 ns;　　--传输延时模型

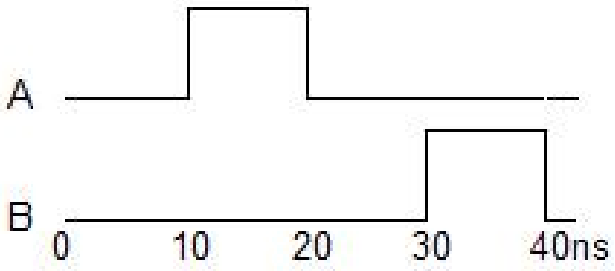

图 7-20　传输延时输入输出波形

7.5.3 仿真δ

习　题

【例 7-21】

```
LIBRARY ieee;
USE ieee.std logic 1164.all;
USE ieee.std logic unsigned.all;
USE ieee.std logic arith.all;
ENTITY addmux IS
   PORT (R : OUT std_logic vector(7 downto 0);  sel : IN  std logic;
         A,B,C,D  : IN  std logic vector(7 downto 0) );
END addmux;
ARCHITECTURE rtl OF addmux IS
BEGIN
    process(A,B,C,D,sel)    begin
       if(sel='0') then  R<=A+B;  else   R<=C+D;  end if;
    end process;
END rtl;
```

$$y(n) = x(n)h(0) + x(n-1) + x(n-2)h(2) + x(n-3)h(3)$$

习　题

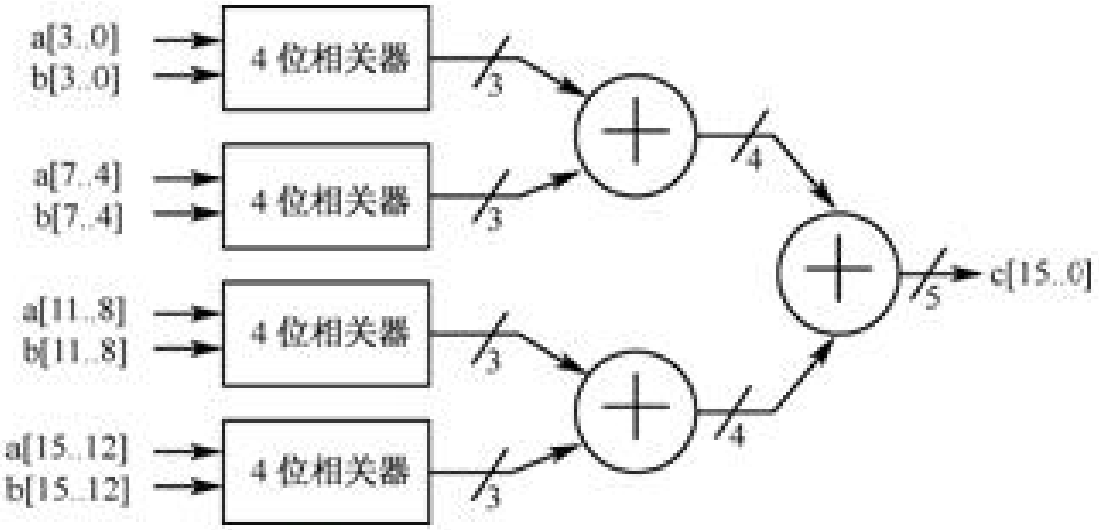

图 7-21　习题 7-16 图

实验与设计

实验7-1 4X4阵列键盘键信号检测电路设计

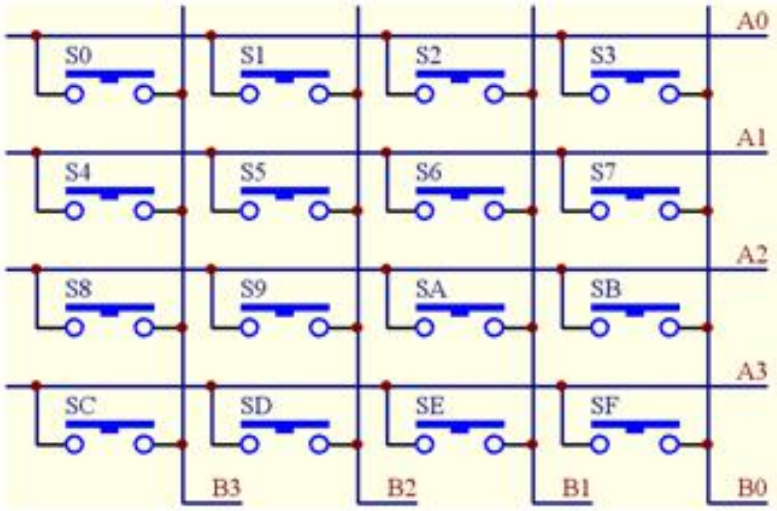

图 7-22 4×4 键盘电路

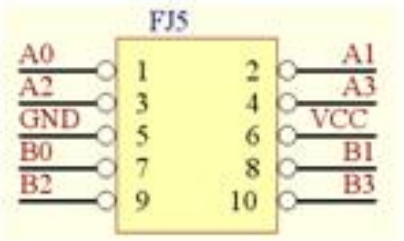

图 7-23 10 芯接口原理图

实验与设计

实验7-2 乐曲硬件演奏电路设计

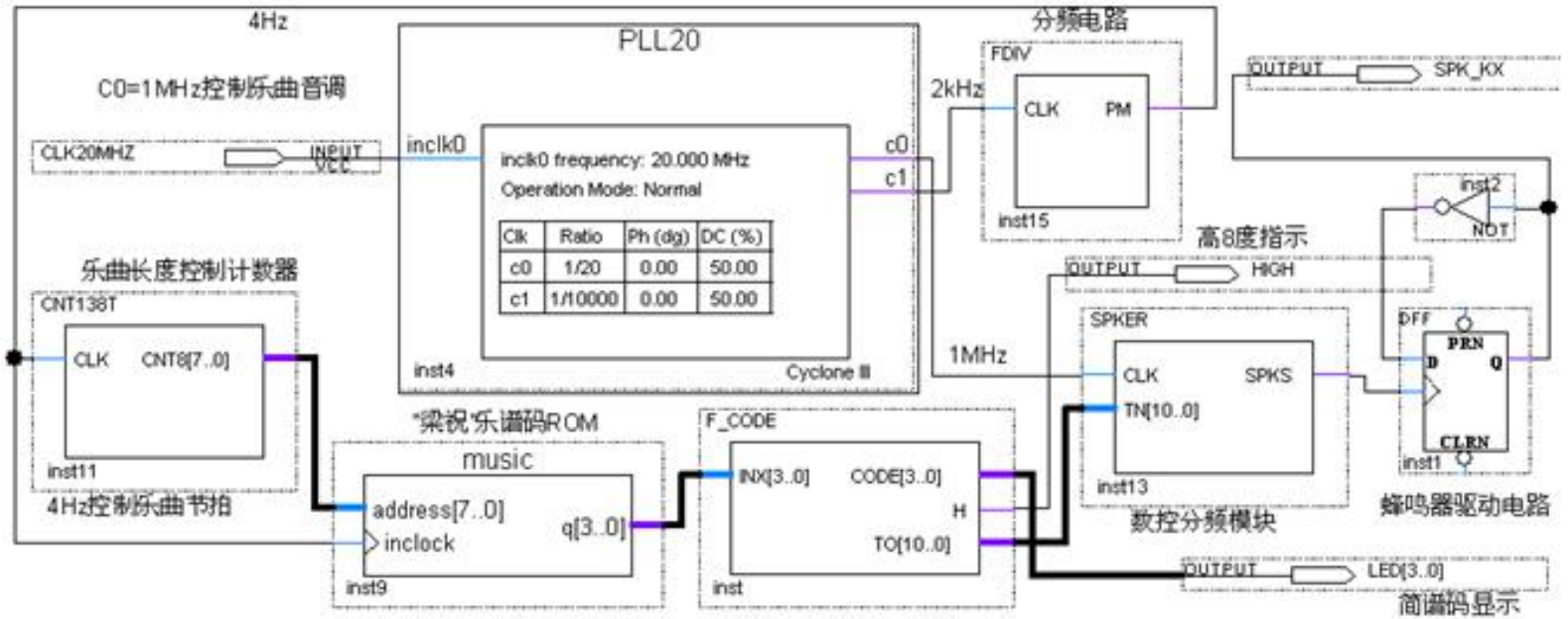

图 7-24 乐曲演奏电路顶层设计

实验与设计

实验7-2 乐曲硬件演奏电路设计

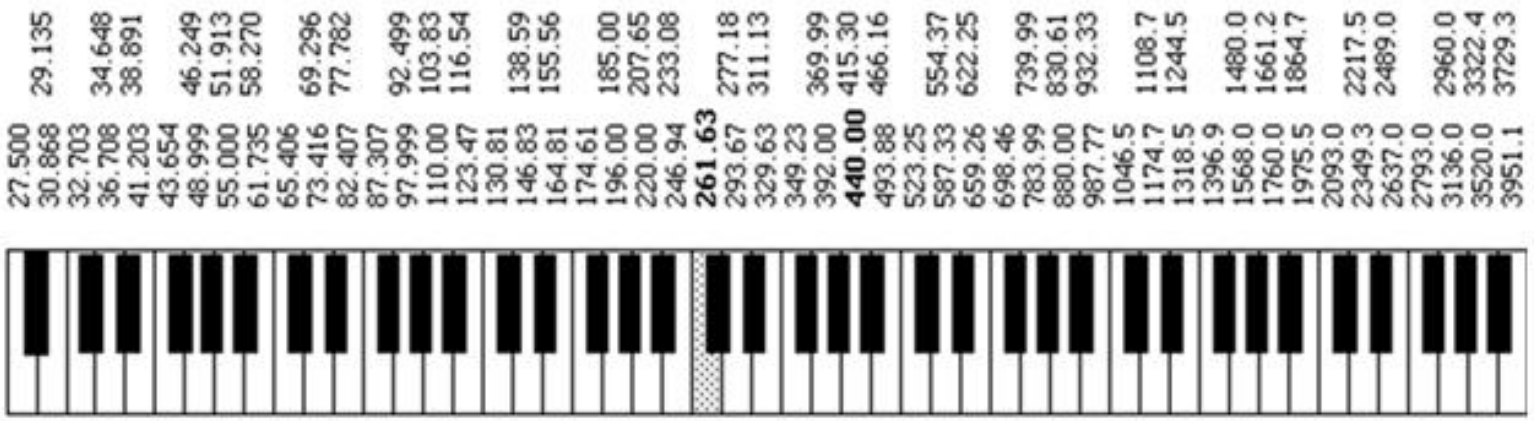

图 7-25 电子琴音阶基频对照图（单位：Hz）

实验与设计

实验7-2 乐曲硬件演奏电路设计

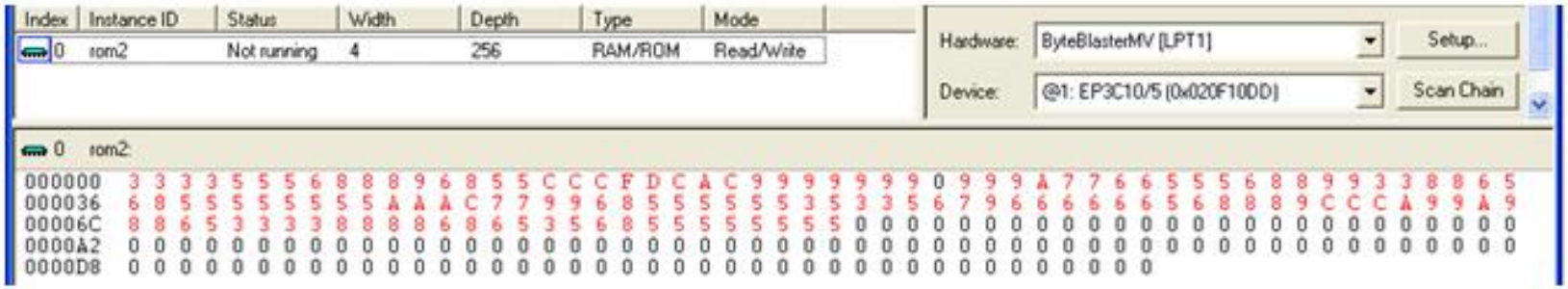

图 7-26 In-System Memory Content Editor 对 MUSIC 模块的数据读取

实验与设计

实验7-2 乐曲硬件演奏电路设计

【例 7-22】

```
    WIDTH = 4;                  --“梁祝”乐曲演奏数据
    DEPTH = 256;                --实际深度139
    ADDRESS_RADIX = DEC;        --地址数据类是十进制
    DATA_RADIX = DEC;           --输出数据的类型也是十进制
    CONTENT  BEGIN              --注意实用文件中要展开以下数据，每一组占一行
00: 3 ; 01: 3 ; 02: 3 ; 03: 3; 04: 5;  05: 5; 06: 5; 07: 6; 08: 8; 09: 8;
10: 8 ; 11: 9 ; 12: 6 ; 13: 8; 14: 5;  15: 5; 16:12; 17: 12;18: 12;19:15;
20:13 ; 21:12 ; 22:10 ; 23:12; 24: 9;  25: 9; 26: 9; 27: 9; 28: 9; 29: 9;
30: 9 ; 31: 0 ; 32: 9 ; 33: 9; 34: 9;  35:10; 36: 7; 37: 7; 38: 6; 39: 6;
40: 5 ; 41: 5 ; 42: 5 ; 43: 6; 44: 8;  45: 8; 46: 9; 47: 9; 48: 3; 49: 3;
50: 8 ; 51: 8 ; 52: 6 ; 53: 5; 54: 6;  55: 8; 56: 5; 57: 5; 58: 5; 59: 5;
60: 5 ; 61: 5 ; 62: 5 ; 63: 5; 64:10;  65:10; 66:10; 67:12; 68: 7; 69: 7;
70: 9 ; 71: 9 ; 72: 6 ; 73: 8; 74: 5;  75: 5; 76: 5; 77: 5; 78: 5; 79: 5;
80: 3 ; 81: 5 ; 82: 3 ; 83: 3; 84: 5;  85: 6; 86: 7; 87: 9; 88: 6; 89: 6;
90: 6 ; 91: 6 ; 92: 6 ; 93: 6; 94: 5;  95: 6; 96: 8; 97: 8; 98: 8; 99: 9;
100:12;101:12 ;102:12 ;103:10;104: 9;  105: 9;106:10;107: 9;108: 8;109: 8;
110: 6;111: 5 ;112: 3 ;113: 3;114: 3;  115: 3;116: 8;117: 8;118: 8;119: 8;
120: 6;121: 8 ;122: 6 ;123: 5;124: 3;  125: 5;126: 6;127: 8;128: 5;129: 5;
130: 5;131: 5 ;132: 5 ;133: 5;134: 5;  135: 5;136: 0;137: 0;138: 0;
END;
```

【例 7-23】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY F_CODE IS
    PORT ( INX :  IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
           CODE  : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0);
              H  : OUT STD_LOGIC;
             TO  : OUT  STD_LOGIC_VECTOR (10 DOWNTO 0));
END;
ARCHITECTURE one OF F_CODE IS
BEGIN
    Search : PROCESS(INX)
    BEGIN
        CASE INX IS     -- 译码电路，查表方式，控制音调的预置数
   WHEN "0000" => TO<="11111111111" ; CODE<="0000"; H<='0';-- 2047
   WHEN "0001" => TO<="01100000101" ; CODE<="0001"; H<='0';-- 773;
   WHEN "0010" => TO<="01110010000" ; CODE<="0010"; H<='0';-- 912;
   WHEN "0011" => TO<="10000001100" ; CODE<="0011"; H<='0';--1036;
   WHEN "0101" => TO<="10010101101" ; CODE<="0101"; H<='0';--1197;
   WHEN "0110" => TO<="10100001010" ; CODE<="0110"; H<='0';--1290;
   WHEN "0111" => TO<="10101011100" ; CODE<="0111"; H<='0';--1372;
   WHEN "1000" => TO<="10110000010" ; CODE<="0001"; H<='1';--1410;
   WHEN "1001" => TO<="10111001000" ; CODE<="0010"; H<='1';--1480;
   WHEN "1010" => TO<="11000000110" ; CODE<="0011"; H<='1';--1542;
   WHEN "1100" => TO<="11001010110" ; CODE<="0101"; H<='1';--1622;
   WHEN "1101" => TO<="11010000100" ; CODE<="0110"; H<='1';--1668;
   WHEN "1111" => TO<="11011000000" ; CODE<="0001"; H<='1';--1728;
   WHEN OTHERS => TO<="11111111111" ; CODE<="0000"; H<='0';-- 2047;
   END CASE;
   END PROCESS;
END;
```

实验与设计

实验7-3 PS2键盘控制模型电子琴电路设计

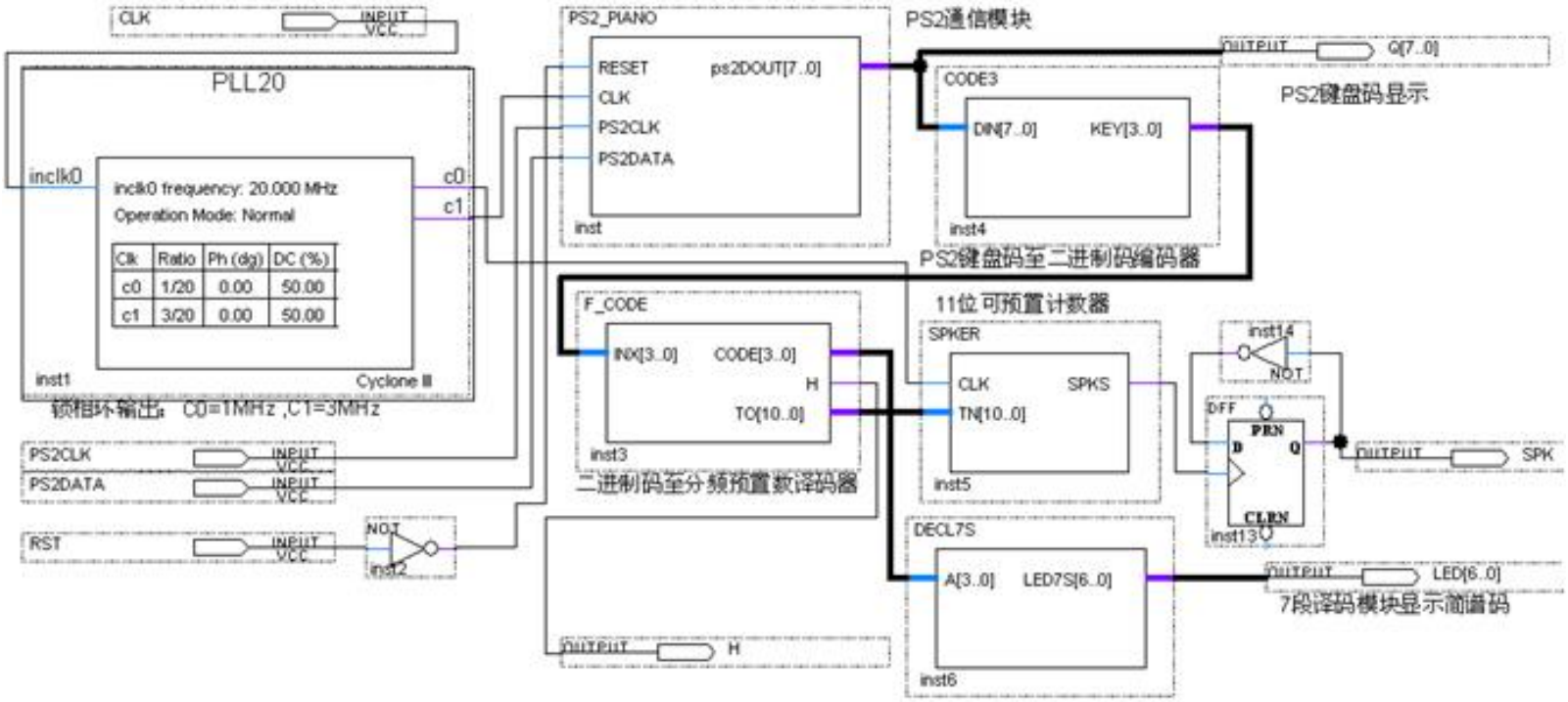

图 7-27 PS2 键盘控制模型电子琴电路顶层设计

实验与设计

实验7-3 PS2键盘控制模型电子琴电路设计

Key	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Data	1C	32	21	23	24	2B	34	33	43	3B	42	4B	3A	31	44
Key	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3
Data	4D	15	2D	1B	2C	3C	2A	1D	22	35	1A	45	16	1E	26
Key	4	5	6	7	8	9	`	-	=	\	]	;	'	,	.
Data	25	2E	36	3D	3E	46	0E	4E	55	5D	5B	4C	52	41	49
Key	/	[	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	KP0
Data	4A	54	05	06	04	0C	03	0B	83	0A	01	09	78	07	70
Key	KP1	KP2	KP3	KP4	KP5	KP6	KP7	KP8	KP9	KP.	KP-	KP+	KP/	KP*	END
Data	69	72	7A	6B	73	74	6C	75	7D	71	7B	79	4A	7C	69

Key	BKSP	SPACE	TAB	CAPS	L SHFT	L CTRL	L CUI	L ALT	R SHFT	R CTRL	R CUI
Data	66	29	0D	58	12	14	1F	11	59	14	27

Key	R ALT	APPS	ENTER	ESC	INSERT	HOME	PG UP	DELETE	PG DN	NUM
Data	11	2F	5A	76	70	6C	7D	71	7A	77

Key	U ARROW	L ARROW	D ARROW	R ARROW	KP EN	SCROLL	PRNT SCRN	PAUSE
Data	75	6B	72	74	5A	7E	12 7C	14

图 7-28 PS2 键盘键控与输出码对照表

【例 7-24】

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity ps2_PIANO is
port(clk,ps2clk,ps2data : in std_logic;
                keycode : out std_logic_vector(7 downto 0);
    keydown, keyup, dataerror : out std_logic);
end ps2_PIANO;
architecture behave of ps2_PIANO is
    signal shiftdata, kbcodereg: std_logic_vector(7 downto 0);
    signal datacoming,kbclkfall, kbclkreg, parity, isfo : std_logic
    signal cnt : std_logic_vector(3 downto 0);
 begin
   process(clk)  begin
     if rising_edge(clk) then  kbclkreg<=ps2clk;
         kbclkfall<=kbclkreg and (not ps2clk);end if;
   end process;
   process(clk)  begin
      if rising_edge(clk) then
          if kbclkfall='1' and datacoming='0' and ps2data='0' then
              datacoming<='1';    cnt<="0000";    parity<='0';
           elsif kbclkfall='1' and datacoming='1' then
```

```
            if cnt=9 then
                if ps2data='1' then datacoming<='0'; dataerror<='0';
                    else dataerror<='1';   end if;
                        cnt<=cnt+1;
            elsif cnt=8 then
                if ps2data=parity  then  dataerror<='0';
                else dataerror<='1';   end if;
                    cnt<=cnt+1;
            else shiftdata<=ps2data & shiftdata(7 downto 1);
              parity<=parity xor ps2data;  cnt<=cnt+1;  end if;
        end if;
     end if;
  end process;
 process(clk)  begin
  if rising_edge(clk) then
      if cnt=10 then
          if shiftdata="11110000"  then  isfo<='1';
            elsif shiftdata /= "11100000" then
               if isfo='1' then  keyup<='1'; keycode<=shiftdata;
                  else keydown<='1'; keycode<=shiftdata;  end if;
              end if;
          else keyup<='0';   keydown<='0';   end if;
    end if;
  end process;
end behave;
```

实验与设计

实验7-4 直流电机综合测控系统设计

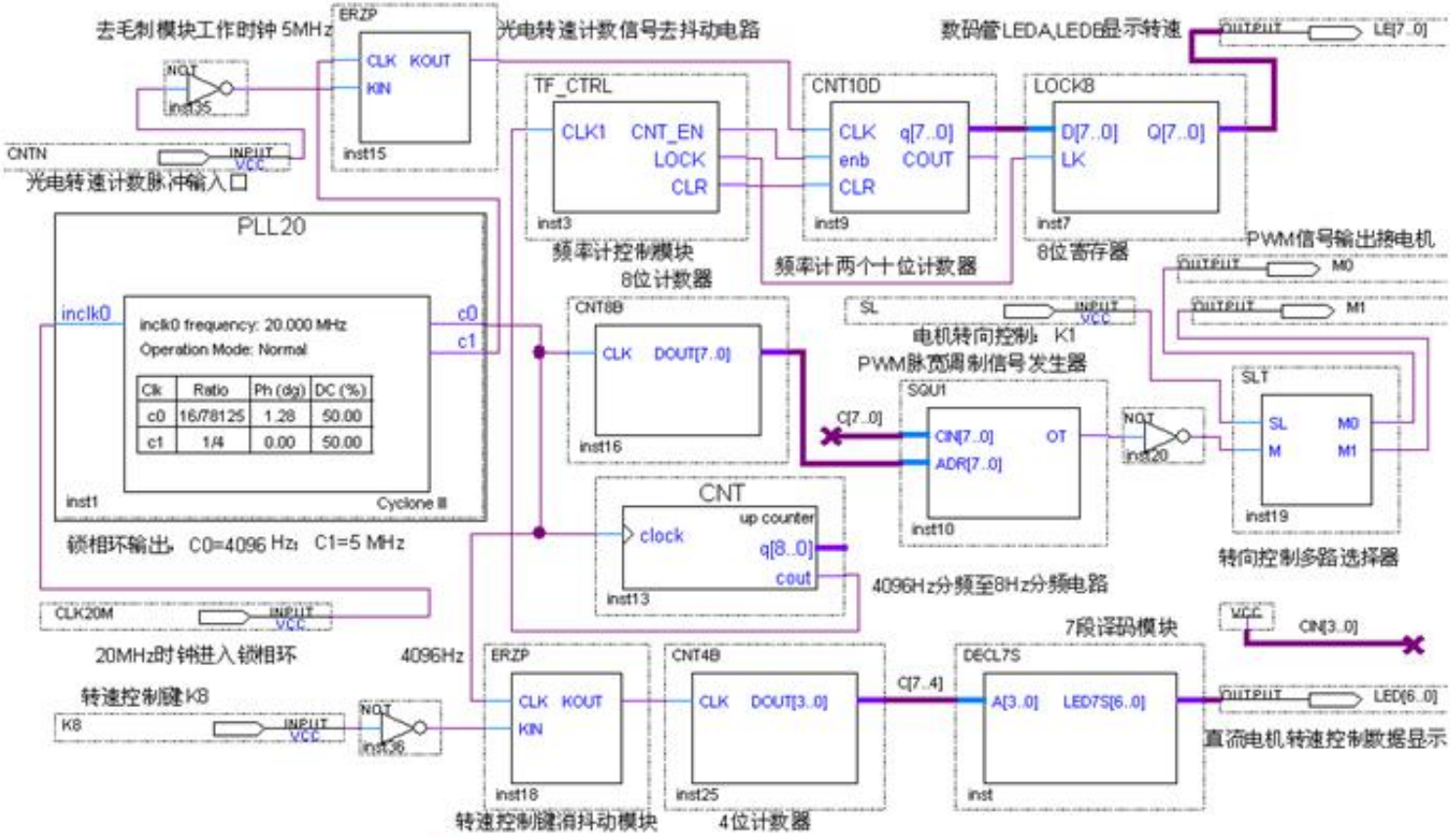

图 7-29 直流电机驱动控制电路顶层设计

实验与设计

实验7-4 直流电机综合测控系统设计

【例 7-25】

```
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

实验7-5 VGA动画图像显示控制电路设计