EDA技术实用教程

第8章

VHDL深入

8.1 数 据 对 象

8.1.1 常数

```
CONSTANT  常数名:数据类型 := 表达式 ;
```

```
CONSTANT  FBT : STD LOGIC VECTOR := "010110" ; --定义常数为标准位矢类型
CONSTANT  DATAIN : INTEGER := 15 ;              --定义常数为整数类型 15
```

8.1 数 据 对 象

8.1.2 变量

VARIABLE 变量名 : 数据类型 := 初始值 ;

```
VARIABLE  a : INTEGER RANGE 0 TO 15 ;  --变量a定义为整数，取值范围是0～15
VARIABLE  d : STD_LOGIC := '1';  --变量 d 定义为标准逻辑位数据类型，初始值是 1
```

目标变量名 := 表达式 ;

```
VARIABLE  x, y : INTEGER RANGE 15 DOWNTO 0; --分别定义变量x和y为整数类型
VARIABLE  a, b : STD LOGIC VECTOR(7 DOWNTO 0) ;
 x := 11 ;                                  --整数直接赋值
 y := 2 + x ;                               --运算表达式赋值，y也是整数变量
 a := b;                                    --b向a赋值
a(5 DOWNTO 0) :=  b(7 DOWNTO 2)  ;          --位矢量类型赋值
```

8.1 数 据 对 象

8.1.3 信号

```
SIGNAL 信号名： 数据类型 := 初始值 ;
```

```
目标信号名 <= 表达式 AFTER 时间量;          -- AFTER 是关键词
```

```
PROCESS (a, b, c)  BEGIN
    y <= a + b ;
    z <= c - a ;
    y <= b ;
END PROCESS ;
```

8.1 数 据 对 象

8.1.3 信号

赋值目标 <= v_0, v_1 AFTER T_1, v_2 AFTER T_2,..., v_n AFTER T_n ;

```
reset<= '1', '0' after reset_period ;
```

```
DATA <= "01", "10" AFTER 40 ns, "11" AFTER 60 ns, "00" AFTER 65 ns;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

表 8-1　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

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

【例 8-1】

```
... --以上部分与例5-1相同
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 8-2】

```
... --以上部分与例5-1相同
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 ;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

【例 8-3】

```
... --以上部分与例5-1相同
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 8-4】

```
... --以上部分与例5-1相同
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 ;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

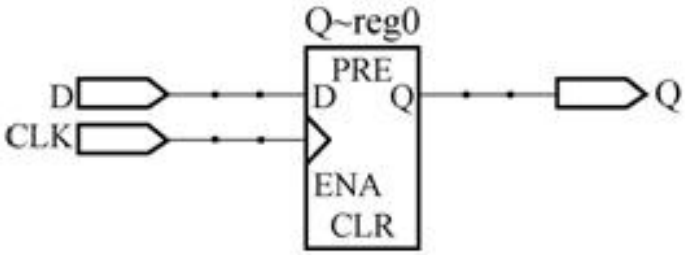

图 8-1　D 触发器电路

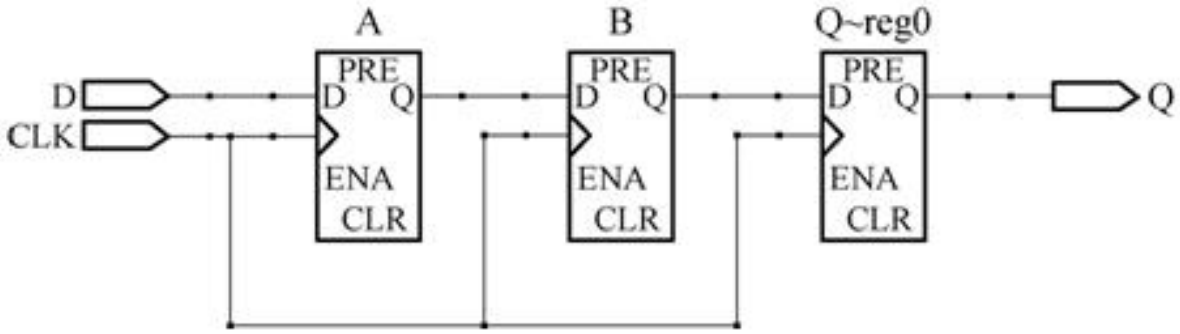

图 8-2　例 8-3 的 RTL 电路图

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

【例 8-5】

```
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
```

```
Q <= B ;
B := A ;
A := D ;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

【例 8-6】

```
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;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

【例 8-7】

```
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;
```

8.1 数 据 对 象

8.1.4 进程中的信号赋值与变量赋值

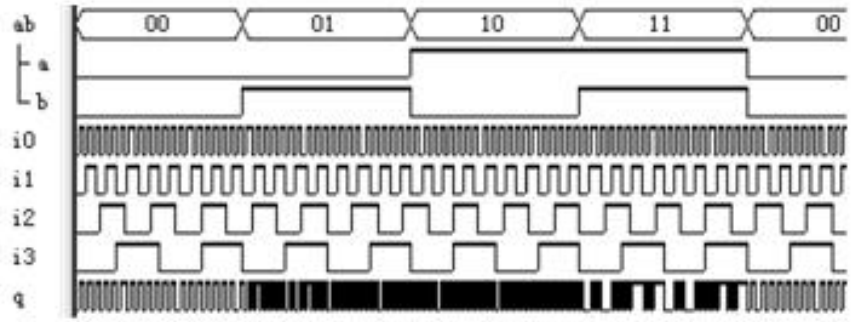

图 8-3　程序例 8-6 的错误工作时序

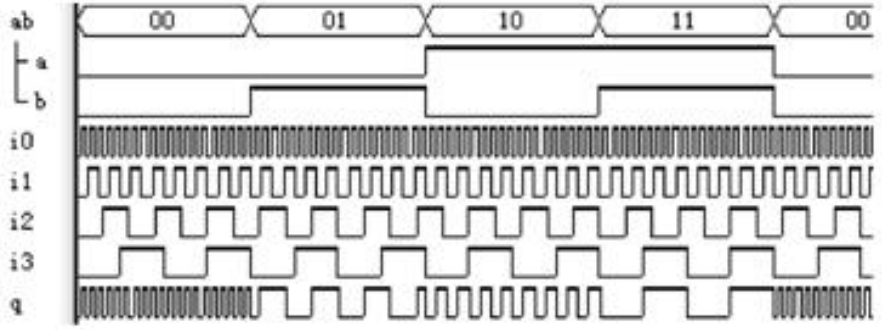

图 8-4　程序例 8-7 的正确工作时序

8.2 含高阻输出的电路设计

8.2.1 三态门设计

【例 8-8】

```
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;
```

8.2 含高阻输出的电路设计

8.2.1 三态门设计

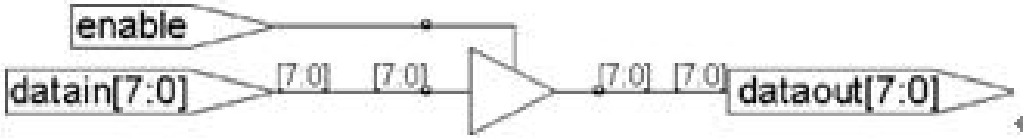

图 8-5　8 位三态控制门电路

8.2 含高阻输出的电路设计

8.2.2 双向端口的设计方法

【例 8-9】

```
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;
```

8.2 含高阻输出的电路设计

8.2.2 双向端口的设计方法

【例 8-10】

```
... --（以上部分同例 8-9）
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;
```

8.2 含高阻输出的电路设计

8.2.2 双向端口的设计方法

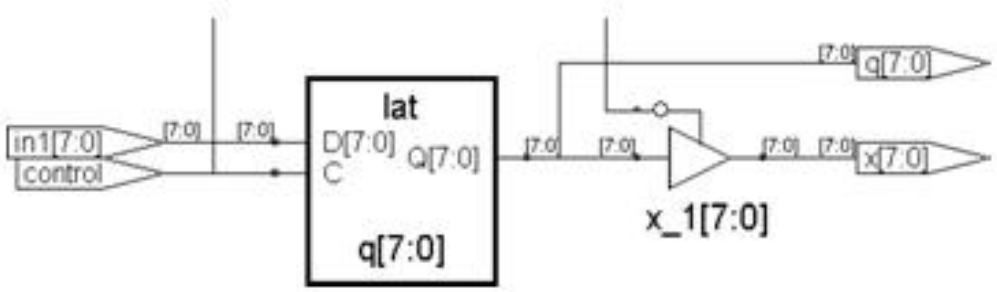

图 8-6　例 8-9 的综合结果

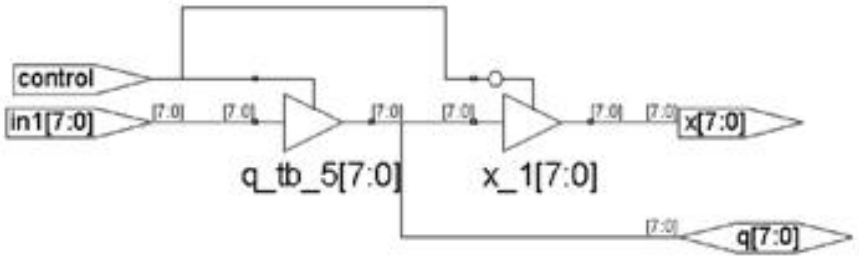

图 8-7　例 8-10 的综合结果

8.2 含高阻输出的电路设计

8.2.3 三态总线电路设计

【例 8-11】

```
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;
```

8.2 含高阻输出的电路设计

8.2.3 三态总线电路设计

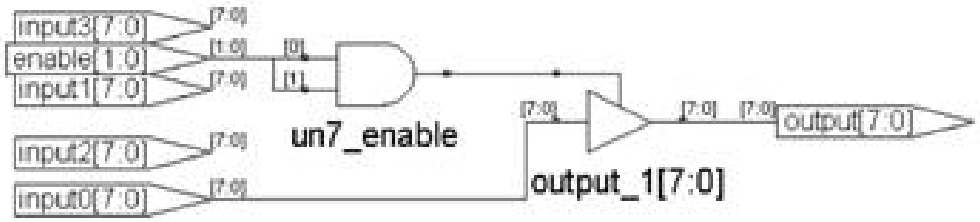

图 8-8　例 8-11 错误的综合结果

8.2 含高阻输出的电路设计

8.2.3 三态总线电路设计

【例 8-12】

```
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;
```

8.3 顺序语句归纳

8.3.1 进程语句格式

```
[进程标号: ] PROCESS [ (敏感信号参数表) ] [IS]
 [进程说明部分]
  BEGIN
      顺序描述语句
END PROCESS [进程标号];
```

8.3 顺序语句归纳

8.3.2 进程结构组成

信号赋值语句

变量赋值语句

进程启动语句

子程序调用语句

顺序描述语句

进程跳出语句

8.3 顺序语句归纳

8.3.3 进程要点

1．PROCESS为一无限循环语句

2．进程中的顺序语句具有明显的顺序和并行双重性

3．信号可以是多个进程间的通信线

4．一个进程中只允许描述对应于一个时钟信号的同步时序逻辑

8.4 并行赋值语句讨论

```
data1 <=  a AND b ;
data2 <=  c ;
```

【例 8-13】

```
SIGNAL select :  INTEGER  RANGE  15  DOWNTO  0;
...
Select <= 0 WHEN  s0='0'  AND  s1='0'  ELSE
          1 WHEN  s0='1'  AND  s1='0'  ELSE
          2 WHEN  s0='0'  AND  s1='1'  ELSE
          3 ;
     x <= a WHEN  select=0   ELSE
          b WHEN  select=1   ELSE
          c WHEN  select=2   ELSE
          d ;
```

8.5 IF语句概述

```
IF   条件句  Then      --类型（1）
     顺序语句
END IF ;
```

```
IF  条件句 Then         --类型（2）
    顺序语句
    ELSE
    顺序语句
END IF ;
```

```
IF  条件句 Then         --类型（3）
    IF  条件句 Then
    ...
    END IF
END IF
```

```
IF  条件句 Then         --类型（4）
    顺序语句
    ELSIF   条件句  Then
    顺序语句
     ...
    ELSE
    顺序语句
END IF
```

8.5 IF语句概述

【例 8-14】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY coder IS
  PORT (    din : IN STD_LOGIC_VECTOR(0 TO 7
         output : OUT STD_LOGIC_VECTOR(0 TO 2)  );
END coder;
ARCHITECTURE behav OF coder IS
 BEGIN
    PROCESS (din)   BEGIN
       IF (din(7)='0') THEN  output <= "000" ;
    ELSIF (din(6)='0') THEN  output <= "100" ;
    ELSIF (din(5)='0') THEN  output <= "010" ;
    ELSIF (din(4)='0') THEN  output <= "110" ;
    ELSIF (din(3)='0') THEN  output <= "001" ;
    ELSIF (din(2)='0') THEN  output <= "101" ;
    ELSIF (din(1)='0') THEN  output <= "011" ;
                       ELSE  output <= "111" ;     END IF ;
    END PROCESS ;
END behav;
```

8.5 IF语句概述

表 8-2　8-3 线优先编码器真值表（表中的“x”为任意，类似 VHDL 中的“—”值）

输　入								输　出		
din0	din1	din2	din3	din4	din5	din6	din7	output0	output1	output2
x	x	x	x	x	x	x	0	0	0	0
x	x	x	x	x	x	0	1	1	0	0
x	x	x	x	x	0	1	1	0	1	0
x	x	x	x	0	1	1	1	1	1	0
x	x	x	0	1	1	1	1	0	0	1
x	x	0	1	1	1	1	1	1	0	1
x	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	1

8.6 仿 真 延 时

8.6.1 固有延时

```
B <=  A AFTER 20 ns ;      --固有延时模型。注意，数字 20 与 ns 间应该有空格!
```

8.6 仿 真 延 时

8.6.2 传输延时

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

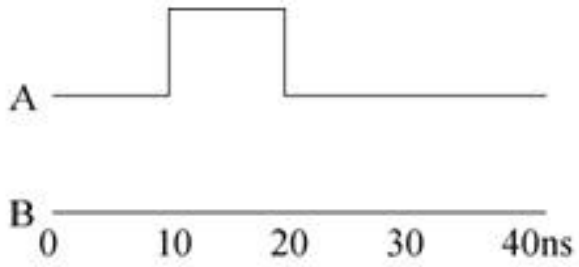

图 8-9 固有延时输入输出波形

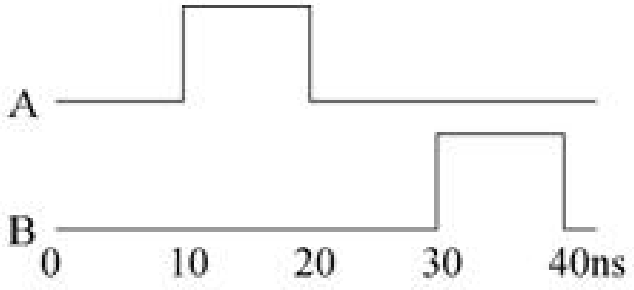

图 8-10 传输延时输入输出波形

8.6.3 仿真 δ

8.7 VHDL的描述风格

8.7.1 RTL描述

8.7.2 行为描述

8.7.3 数据流描述

8.7.4 结构描述

8.8 VHDL Test Bench仿真

8.8.1 VHDL仿真流程

图 8-11 HDL 系统设计描述层次

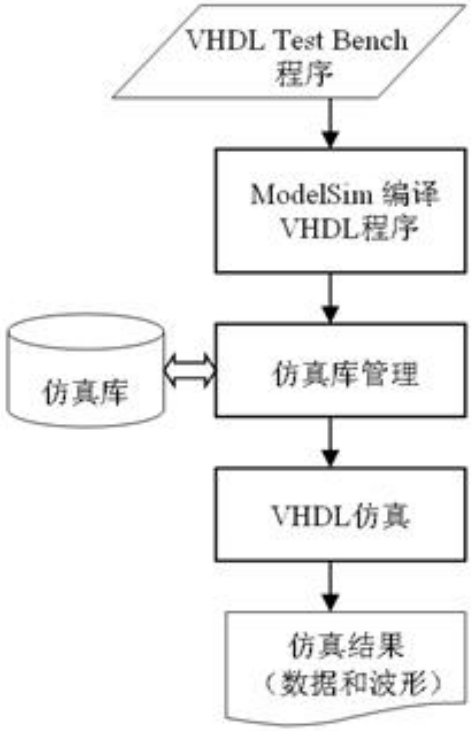

图 8-12 VHDL 仿真流程

8.8 VHDL Test Bench仿真

8.8.2 VHDL Test Bench仿真

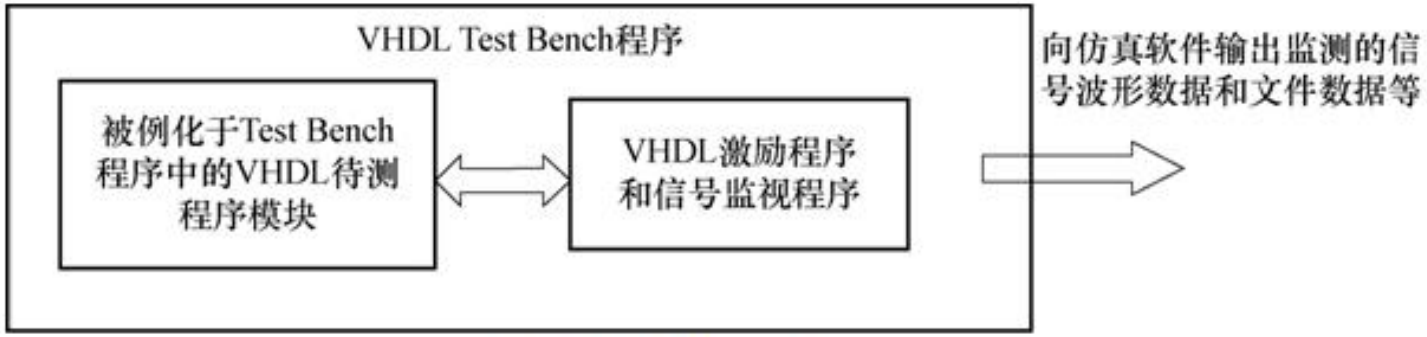

图 8-13　VHDL Test Bench 结构

【例8-15】Test Bench文件名：CNT10_TB.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT10_TB IS
END CNT10_TB;
ARCHITECTURE ONE OF CNT10_TB IS
  COMPONENT CNT10
  PORT (CLK,RST,EN,LOAD : IN STD_LOGIC;
        DATA : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        DOUT : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        COUT : OUT STD_LOGIC);
  END COMPONENT;
SIGNAL CLK1 : STD_LOGIC :='0';  --定义向CNT10时钟端口输入的时钟信号
SIGNAL RST1 : STD_LOGIC :='1';  --定义向CNT10复位端口输入的复位信号
SIGNAL EN1  : STD_LOGIC :='0';  --定义向CNT10时钟使能端口输入的使能信号
SIGNAL LOAD1 : STD_LOGIC :='1'; --定义控制CNT10加载的信号
SIGNAL DATA1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL DOUT1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL COUT1 : STD_LOGIC;
CONSTANT CLK_P : TIME := 250 ns;
           --定义时间类型常数是CLK_P=250 ns,注意250与ns间应该有空格!
BEGIN
  U1:  CNT10  PORT MAP(CLK=>CLK1, RST=>RST1, EN=>EN1, LOAD=>LOAD1,
           DATA=>DATA1, DOUT=>DOUT1, COUT=>COUT1); --例化待测试模块
 PROCESS  BEGIN  --产生时钟信号的进程.这是个没有敏感信号的永久自动启动的进程
  CLK1<='0';    WAIT FOR  CLK_P; --CLK1首先输出0. 250ns后输出1
  CLK1<='1';    WAIT FOR  CLK_P; --再过250ns后返回.
END PROCESS;
  RST1  <= '1', '0' AFTER 610 ns, '1' AFTER 880 ns; --RST1的电平控制
  EN1   <= '0', '1' AFTER 300 ns;                   --EN1电平控制
  LOAD1 <= '1', '0' AFTER 3000 ns, '1' AFTER 3300 ns,'0' AFTER 8000 ns,
           '1' AFTER 8360 ns ;
  DATA1 <= "0100", "0111" AFTER 860 ns,"1000" AFTER 5500 ns,
           "0100" AFTER 9000 ns;
END ONE;
```

8.8 VHDL Test Bench仿真

8.8.3 VHDL Test Bench仿真实例

1．安装ModelSim

2．为Test Bench仿真设置参数

8.8 VHDL Test Bench仿真

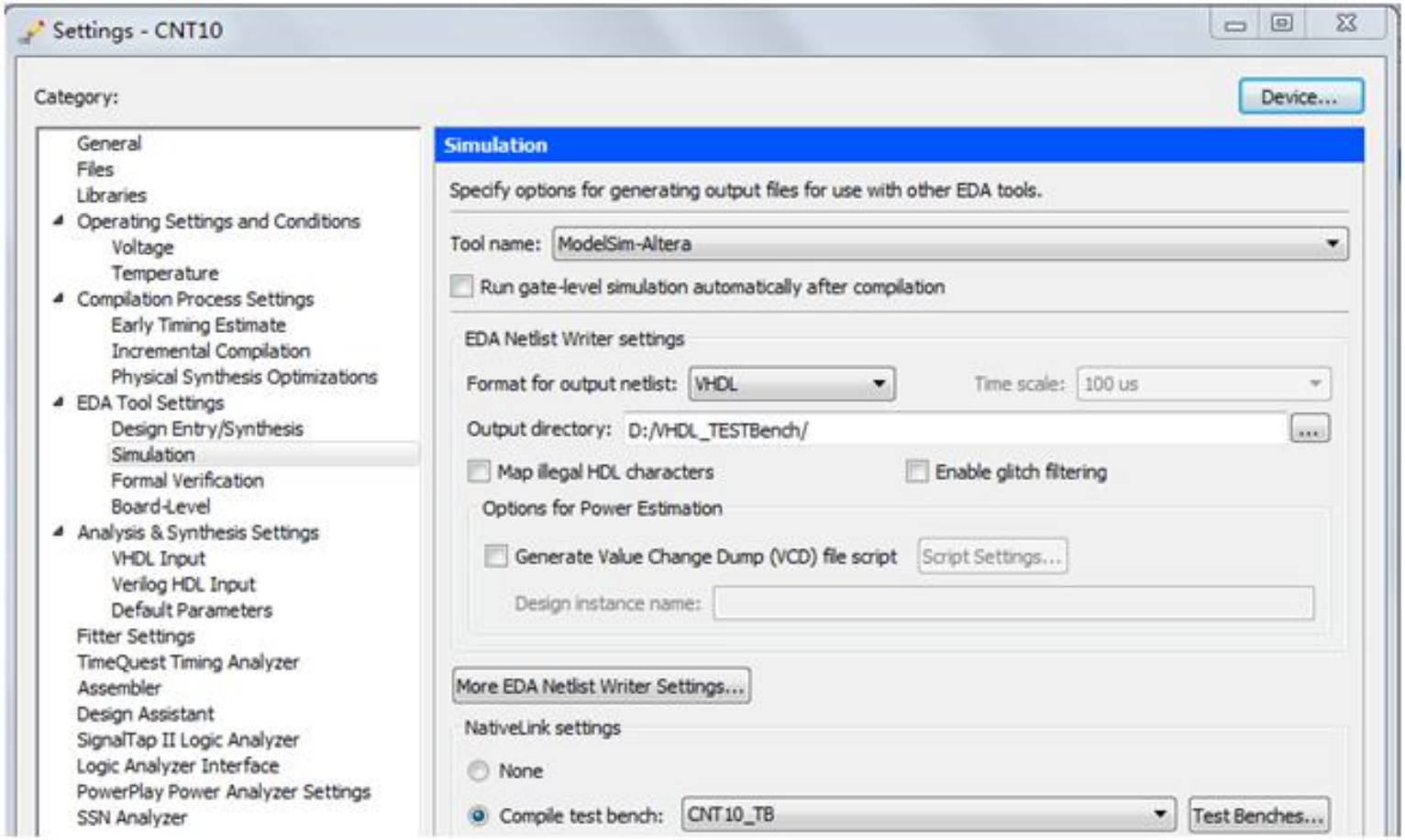

图 8-14 选择仿真工具名称和输出网表语言形式

8.8 VHDL Test Bench仿真

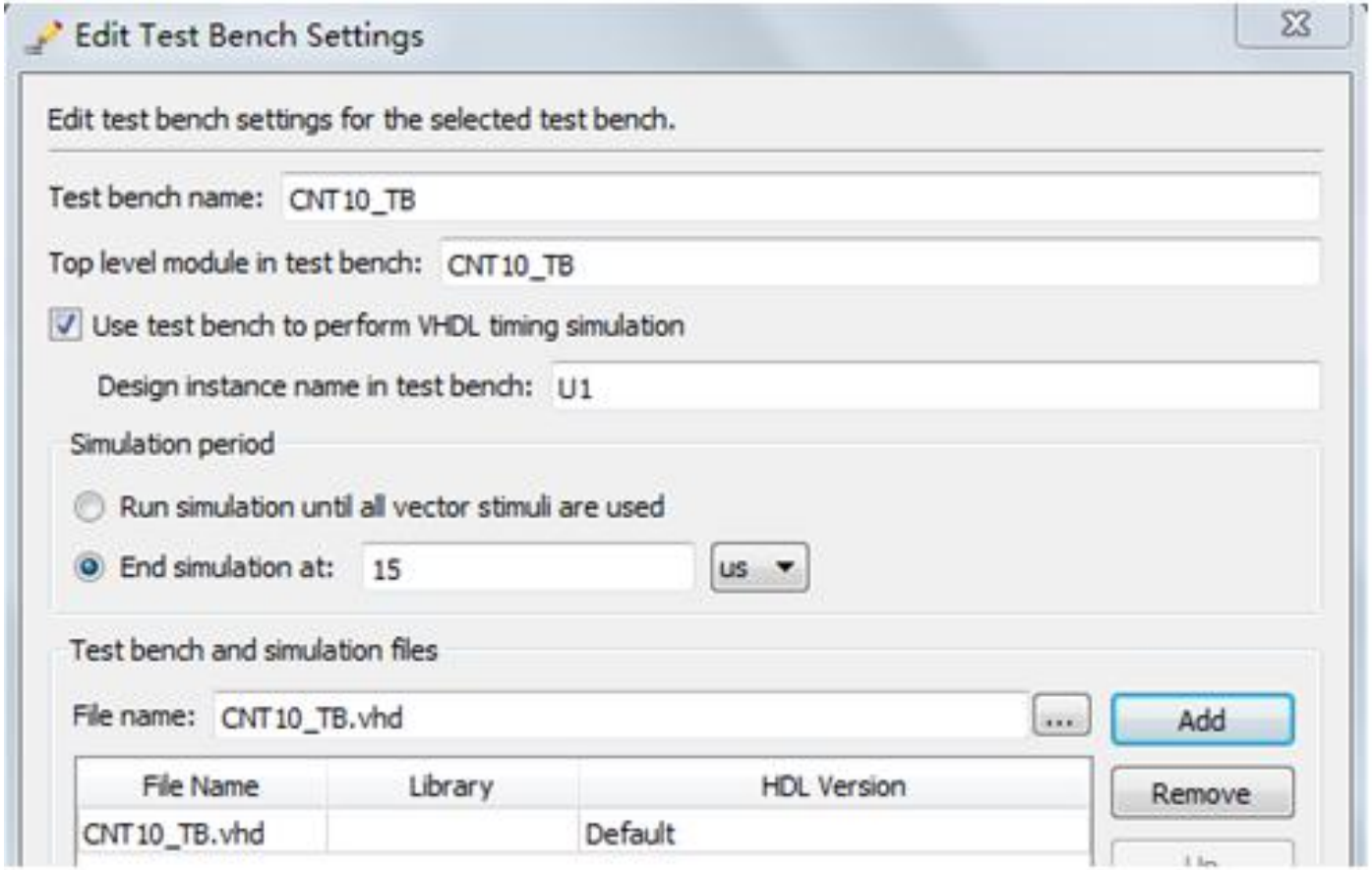

图 8-15　为 Test Bench 仿真设置参数

3．启动Test Bench仿真

8.8 VHDL Test Bench仿真

8.8.3 VHDL Test Bench仿真实例

4. 分析Test Bench仿真结果

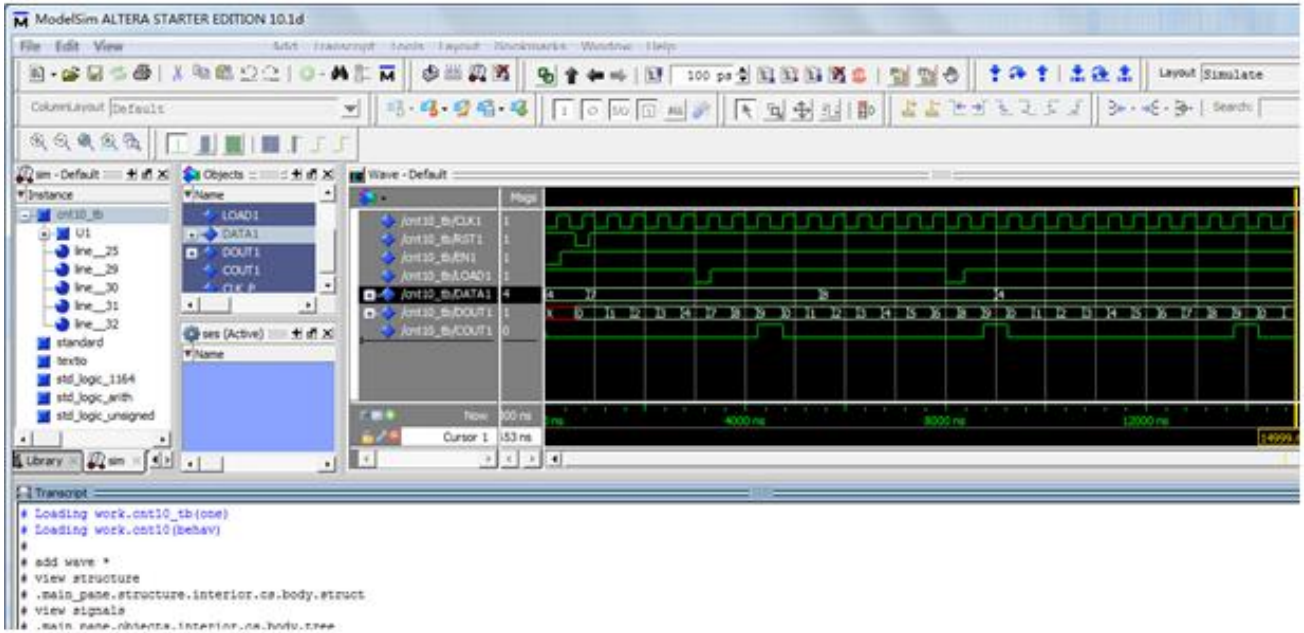

图 8-16 启动仿真后弹出的 ModelSim 界面（时序波形清晰画面如图 8-17 所示）

8.8 VHDL Test Bench仿真

8.8.3 VHDL Test Bench仿真实例

4. 分析Test Bench仿真结果

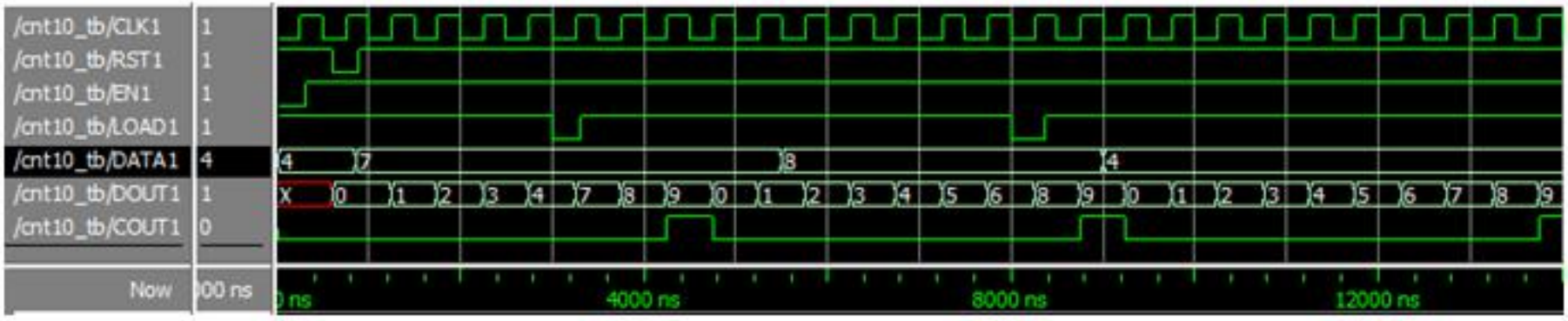

图 8-17 Test Bench 输出的仿真波形详图

【例 8-16】

```
LIBRARY IEEE;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.all;
ENTITY CNT4B IS
PORT(CLR, EN, CLK, LOAD, UD  : IN STD_LOGIC ;
 DATA    : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
 COUT    : OUT STD_LOGIC ;
 DOUT    : OUT STD_LOGIC_VECTOR (3 DOWNTO 0) );
END CNT4B;
ARCHITECTURE SYN OF CNT4B IS
 SIGNAL sub_wire0  : STD_LOGIC ;
 SIGNAL sub_wire1  : STD_LOGIC_VECTOR (3 DOWNTO 0);
 COMPONENT lpm_counter
GENERIC(lpm_direction,lpm_port_updown ,lpm_type : STRING;
     lpm_modulus, lpm_width : NATURAL );
PORT (sload, clk_en, aclr, clock, updown   : IN STD_LOGIC ;
      cout : OUT STD_LOGIC ;
         q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
     data : IN STD_LOGIC_VECTOR (3 DOWNTO 0) );
END COMPONENT;
BEGIN
COUT <= sub_wire0;  DOUT <= sub_wire1(3 DOWNTO 0);
  U1 : lpm_counter GENERIC MAP (
      lpm_direction => "UNUSED", lpm_modulus => 12,
      lpm_port_updown => "PORT_USED", lpm_type => "LPM_COUNTER",
      lpm_width => 4)
PORT MAP (clk_en=>EN, aclr=>CLR,  clock=>CLK , sload=>LOAD,
       data=>DATA, updown=>UD, cout=>sub_wire0,q => sub_wire1);
END SYN;
```

【例8-17】

```
LIBRARY IEEE;
USE IEEE.STD LOGIC 1164.ALL;
USE IEEE.STD LOGIC UNSIGNED.ALL;
ENTITY CNT4B TB IS
END CNT4B TB;
ARCHITECTURE ONE OF CNT10 TB IS
  COMPONENT CNT4B
  PORT (CLK,CLR,EN,LOAD,UD : IN STD LOGIC;
        DATA : IN STD LOGIC VECTOR(3 DOWNTO 0);
        DOUT : OUT STD LOGIC VECTOR(3 DOWNTO 0);
        COUT : OUT STD LOGIC);
  END COMPONENT;
SIGNAL clk1  : STD LOGIC :='0';
SIGNAL clr1  : STD LOGIC :='0';
SIGNAL en1   : STD LOGIC :='0';
SIGNAL ud1   : STD LOGIC :='1';
SIGNAL load1 : STD LOGIC :='0';
SIGNAL data1 : STD LOGIC VECTOR(3 DOWNTO 0);
SIGNAL dout1 : STD LOGIC VECTOR(3 DOWNTO 0);
SIGNAL cout1 : STD LOGIC;
CONSTANT CLK P : TIME := 410 ns;
BEGIN
  U1:  CNT4B  PORT MAP(CLK=>clk1, CLR=>clr1, EN=>en1, LOAD=>load1,
                       UD=>ud1, DATA=>data1, DOUT=>dout1, COUT=>cout1);
 PROCESS BEGIN
  clk1<='0';    WAIT FOR  CLK P;
  clk1<='1';    WAIT FOR  CLK P;
END PROCESS;
  clr1  <= '0', '1' AFTER 2000 ns, '0' AFTER 2500 ns;
  ud1   <= '1', '0' AFTER 15000 ns ;
  en1   <= '0', '1' AFTER 1800 ns;
  load1 <= '0', '1' AFTER 5900 ns, '0' AFTER 6600 ns,
           '1' AFTER 12000 ns,'0' AFTER 12400 ns,
           '1' AFTER 23000 ns,'0' AFTER 23400 ns ;
  data1 <= "1000", "0111" AFTER 8000 ns,"0100" AFTER 16000 ns,
                   "1001" AFTER 25000 ns;
END ONE;
```

8.8 VHDL Test Bench仿真

8.8.3 VHDL Test Bench仿真实例

4. 分析Test Bench仿真结果

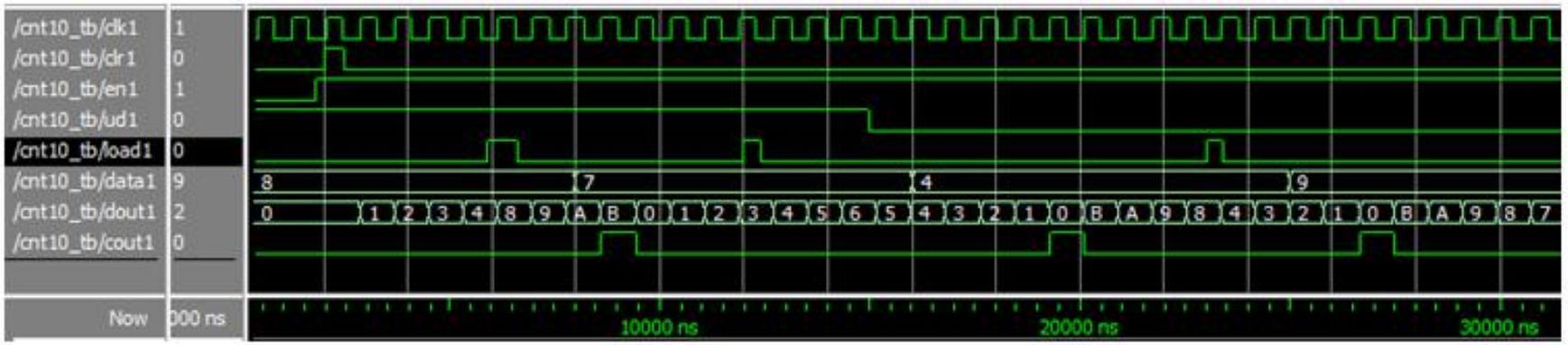

图 8-18　Test Bench 输出的仿真波形

实验与设计

8-1 4×4阵列键盘键信号检测电路设计

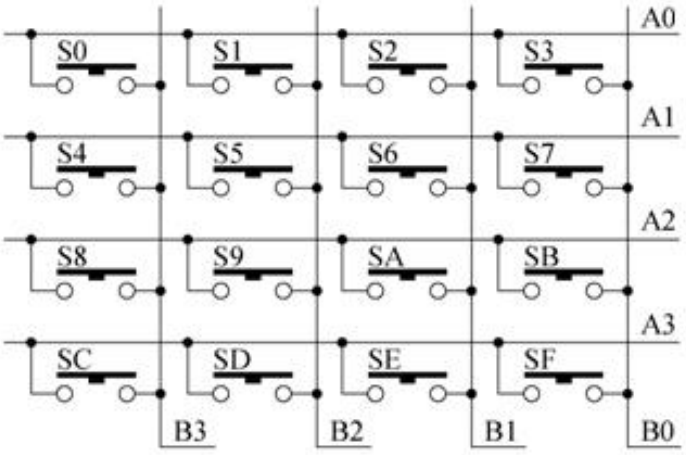

图 8-19 4×4 键盘电路

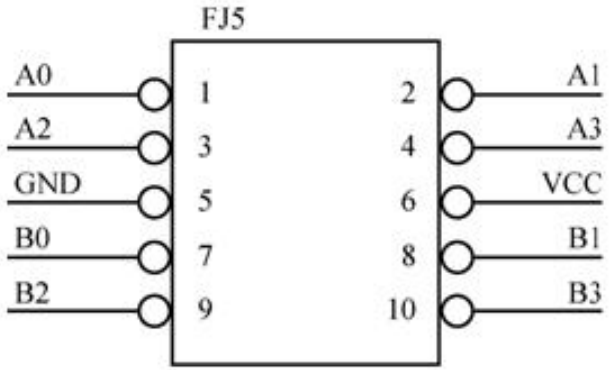

图 8-20 10 芯接口

实验与设计

8-2 乐曲硬件演奏电路设计

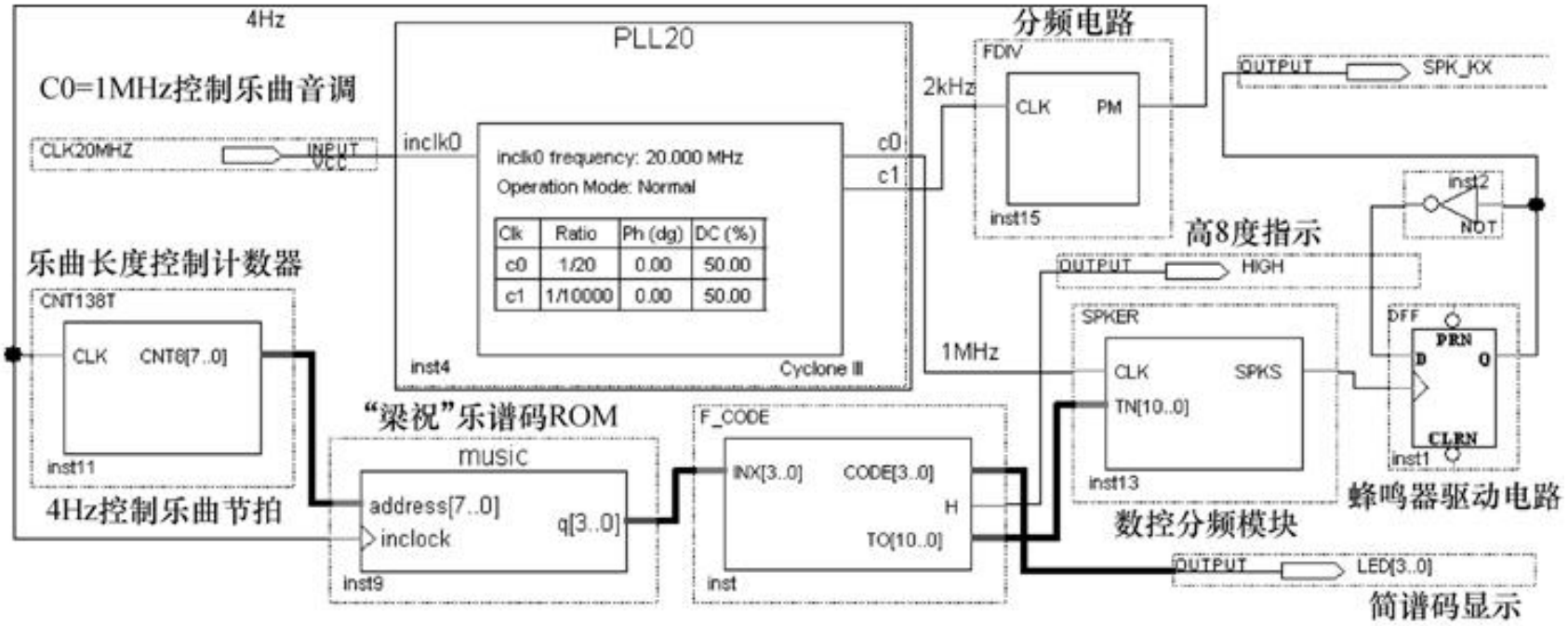

图 8-21　乐曲演奏电路顶层设计

实验与设计

8-2 乐曲硬件演奏电路设计

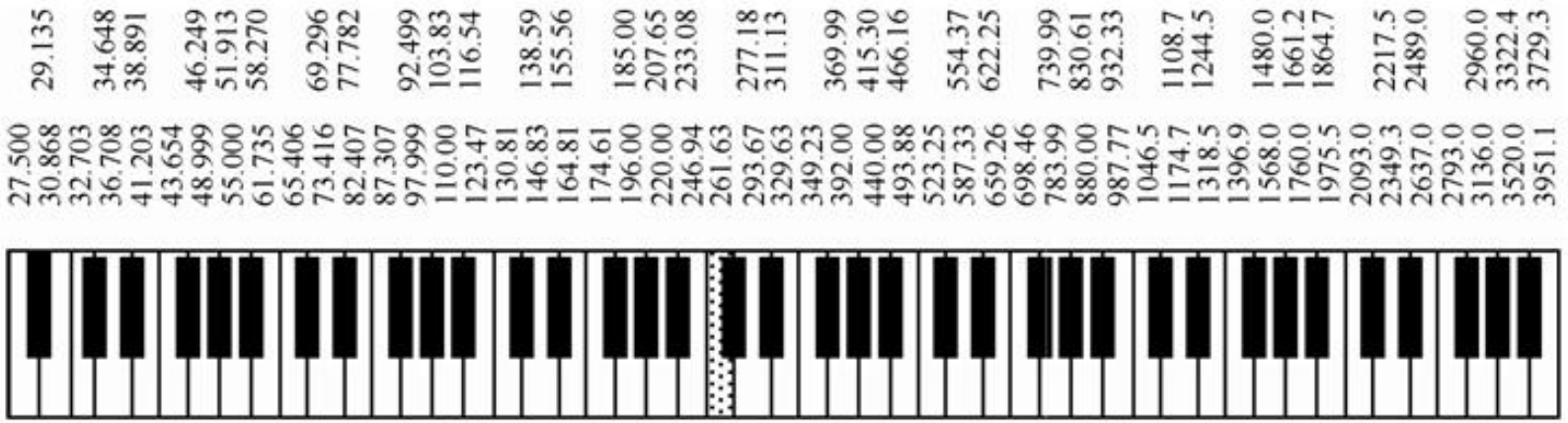

图 8-22　电子琴音阶基频对照图（单位 Hz）

实验与设计

【例 8-18】

```
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

8-2 乐曲硬件演奏电路设计

【例 8-19】

```
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 ;
```

实验与设计

8-3 PS2键盘控制模型电子琴电路设计

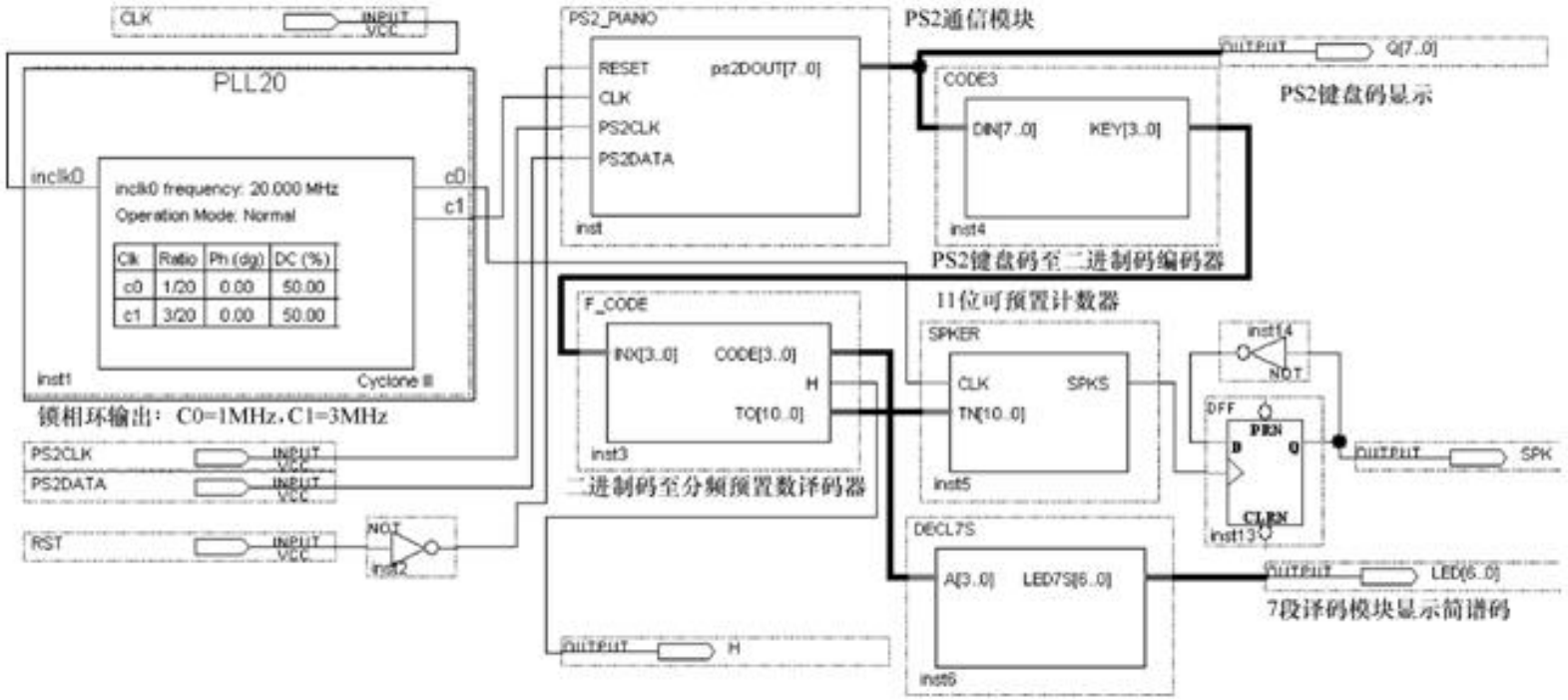

图 8-23　PS2 键盘控制模型电子琴电路顶层设计

【例 8-20】

```
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

8-3 PS2键盘控制模型电子琴电路设计

表 8-3 PS2 键盘键控与输出码对照表

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	LSHFT	LCTRL	LCUI	LALT	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	UARROW	LARROW	D ARROW	R ARROW	KP EN	SCROLL	PRNT SCRN	PAUSE
Data	75	6B	72	74	5A	7E	12 7C	14

实验与设计

8-4 直流电机综合测控系统设计

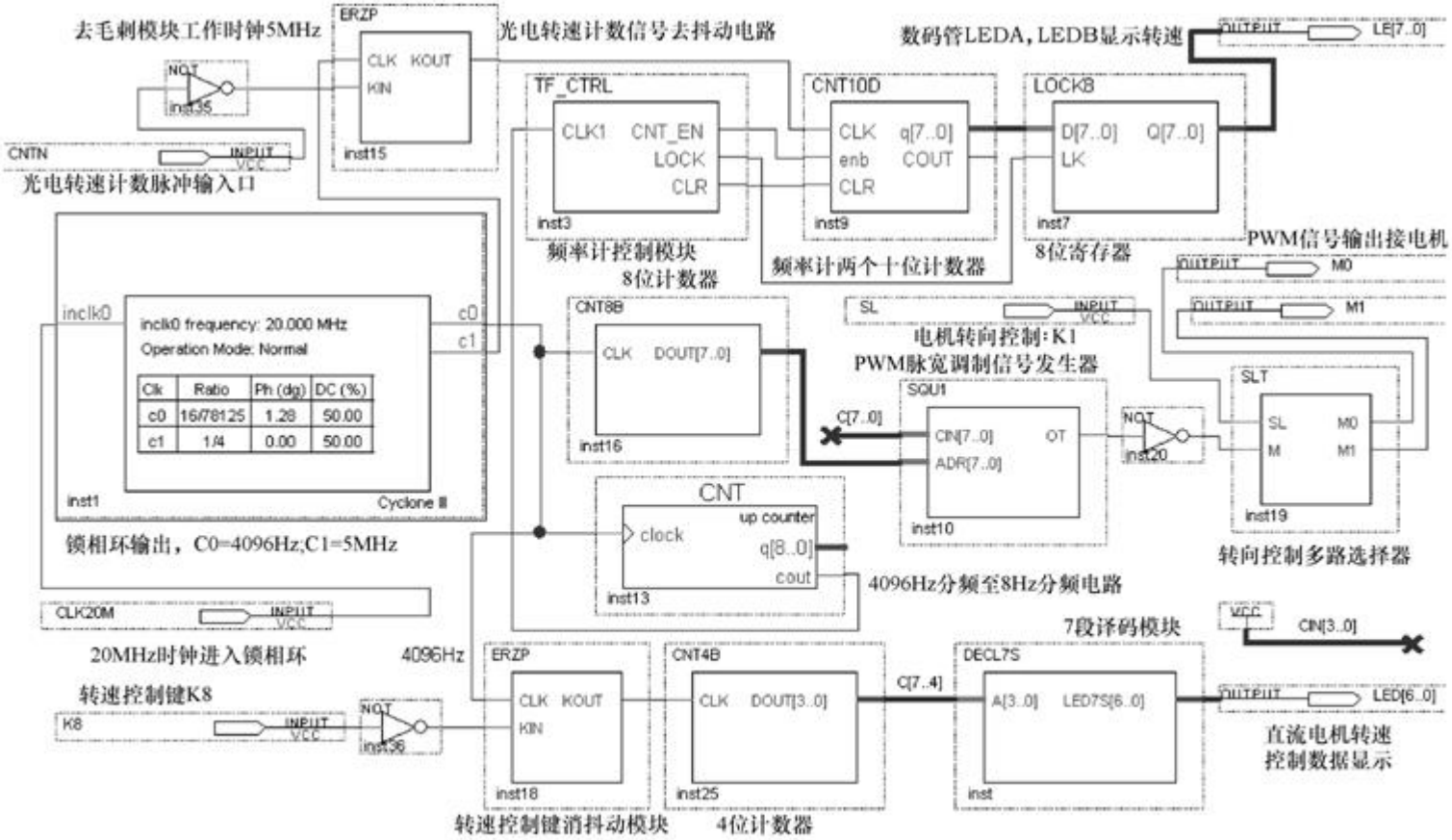

图 8-24　直流电机驱动控制电路顶层设计

实验与设计

8-4 直流电机综合测控系统设计

【例 8-21】

```
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

8-5 AM幅度调制信号发生器设计

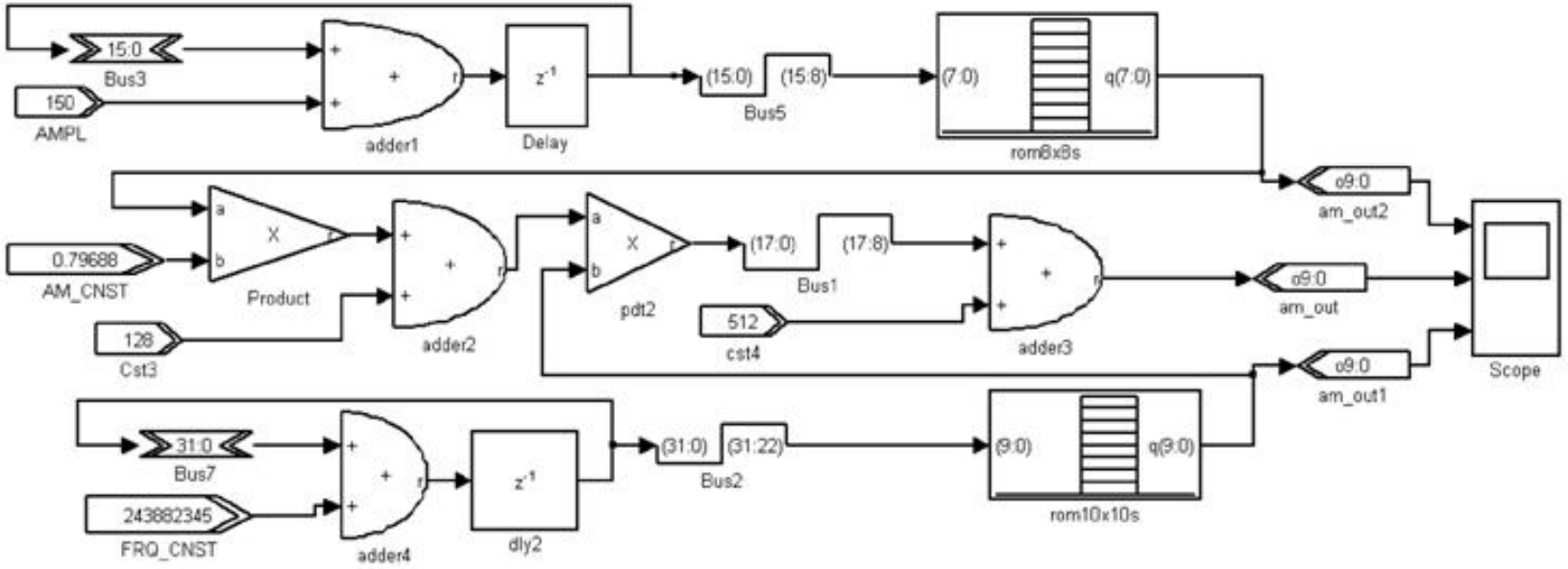

图 8-25 AM 信号发生器 DSP Builder/MATLAB Simulink 模型

实验与设计

8-6 在ModelSim上对VHDL Test Bench进行仿真

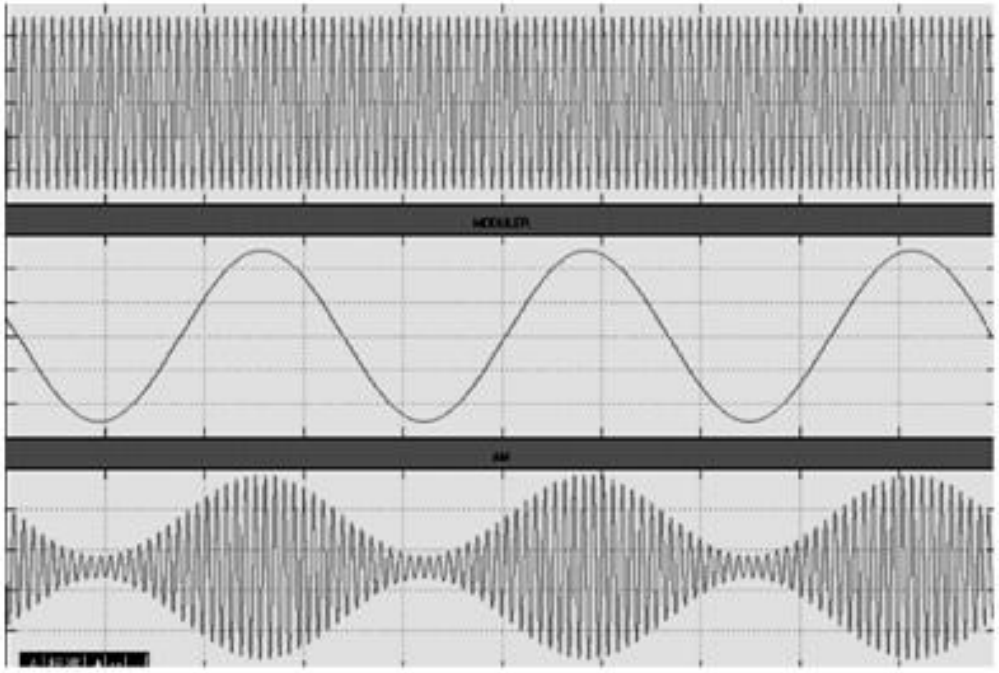

图 8-26　AM 模型仿真波形