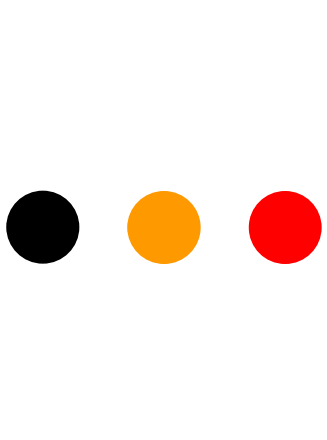

第12章

Verilog知识拾遗

12.1 Verilog文字规则

1. 整数

```
reg[3:0] A; reg[5:0] B; reg[31:0] C;

A<=6'B11_0110; //A 实际获得赋值 4'B0110，高 2 位被截去。进制符号 b 或 B 大小写都可
A<='o466;      //'o466='H136，A 实际获得低 4 位：4'B0110。高位被截去
A<=123;      //123=32'h0000_007B，转换为 32 位二进制数，A 实际获得赋值 4'B1011
A<=8'hAC;    //A 实际获得赋值 4'h1100，高 4 位被截去
C<=-5;       //-5=32'hFFFFFFFB，A 即获得赋值 32'hFFFFFFFB
B<=-7'd30;   //-7'd30=7'H62，A 实际获得赋值=6'H22，高 1 位被截去
```

12.1 Verilog文字规则

2．实数

3．字符串

4．标识符

```
Decoder_1,   FFT,   Sig_N,   Not_Ack,   State0,   _Decoder_,   REG

  2FFT                    // 起始为数字
  Sig_#N                  // 符号“#”不能成为标识符的构成
  Not-Ack                 // 符号“-”不能成为标识符的构成
  data_ _BUS              // 标识符中不能有双下划线
  Reg                     // 关键词
  ADDER*                  // 标识符中不允许包含字符*
```

12.2 数 据 类 型

12.2.1 net网线类型

12.2.2 register寄存器类型

12.2.3 存储器类型

12.3 操 作 符

1．逻辑操作符

- && 逻辑与
- || 逻辑或
- ! 逻辑非。例如：!A=0

2．缩位操作符

12.4 常用语句补充

12.4.1 initial过程语句使用示例

【例 12-1】

```
`timescale 1ns/100ps      //声明仿真时间单位是 1ns，仿真精度也是 100ps
module test;              //名为 test 的测试模块
reg A, B, C;
initial                   //定义 initial 过程语句结构
  begin
   A=0;B=1;C=0     //在过程中分别定义 A、B、C 在时刻 0 的初始值
  #50  A=1;B=0;    //经过 50ns 延时后，在仿真时刻 50ns 时 A 和 B 的输入值分别是 1,0
  #50  A=0;C=1;    //又经过 50ns 延时后，在时刻 100ns 时 A 和 C 的输入值分别是 0,1
  #50  B=1;        //再经过 50ns 延时后，在时刻 150ns 时 B 的输入值分别是 1
  #50  B=0;C=0     //再经过 50ns 延时后，在时刻 200ns 时 B 和 C 的输入值都是 0
  #50  $finish     //又经过 50ns 延时后，结束
  end
endmodule
```

`timescale 仿真时间单位/仿真精度

12.4 常用语句补充

12.4.2 forever循环语句

```
    forever   语句;
或  forever begin   语句;   end
```

12.4 常用语句补充

12.4.3 编译指示语句

1. 文件包含语句`include　　`include "文件名"

【例 12-2】

```
`include "h_adder.v"
`include "or2a.v"
   module f_adder(input ain,bin,cin,output cout,sum);
     wire  e,d,f ;
   h_adder  u1(ain, bin, e, d);
   h_adder  u2(.a(e), .so(sum), .b(cin),.co(f));
       or2a  u3(.a(d), .b(f),    .c(cout));
endmodule
```

12.4 常用语句补充

12.4.3 编译指示语句

2. 条件编译语句`ifdef、`else、`endif

条件编译命令语句格式 1

```
`ifdef 宏名
        语句块
    `endif
```

条件编译命令语句格式 2

```
`ifdef 宏名
        语句块 1
`else   语句块 2
`endif
```

12.4 常用语句补充

12.4.3 编译指示语句

2. 条件编译语句`ifdef、`else、`endif

【例 12-3】

```
`define AND
module andd (out,A,B);
   input[1:0] A,B;
   output[1:0] out;
   `ifdef AND
   assign out=A&B;
   `else assign out=A|B;
   `endif
endmodule
```

【例 12-4】

```
`define OR1
module andd (out,A,B);
   input[1:0] A,B;
   output[1:0] out;
   `ifdef AND
   assign out=A&B;
   `else assign out=A|B;
   `endif
endmodule
```

12.4 常用语句补充

12.4.3 编译指示语句

2. 条件编译语句`ifdef、`else、`endif

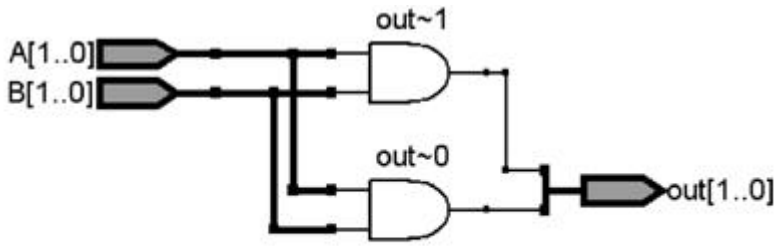

图 12-1 对应例 12-3 的 RTL 图

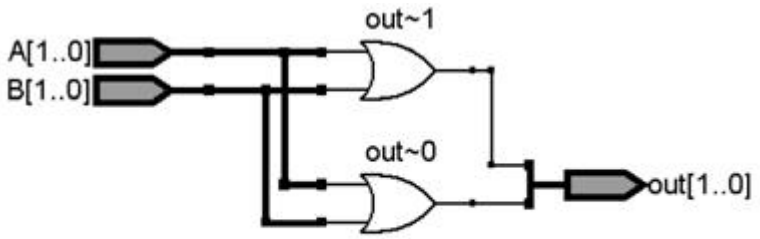

图 12-2 对应例 12-4 的 RTL 图

12.4 常用语句补充

12.4.4 任务和函数语句

1. 任务（task）语句

任务（task）定义语句格式

```
task <任务名>;
   端口及数据类型声明语句
   begin 过程语句； end
endtask
```

任务调用格式

```
<任务名>（端口1，端口2，...,端口N）;
```

12.4 常用语句补充

【例 12-5】

```
module TASKDEMO (S,D,C1,D1,C2,D2);   //主程序模块及端口定义
input S;   input[3:0] C1,D1,C2,D2;
output[3:0] D;                       //端口定义数目不受限制
reg[3:0] out1,out2;
task CMP;                          //任务定义，任务名 CMP，此行不能出现端口定义语句
 input[3:0] A,B;  output[3:0] DOUT;  //注意任务端口名的排序
 begin if (A>B) DOUT=A;            //任务过程语句描述一个比较电路
  else DOUT=B; end                 //在任务结构中可以调用其他任务或函数，甚至自身
 endtask                           //任务定义结束
 always @ (*) begin                //主程序过程开始
CMP(C1,D1,out1);                   //调用一次任务。任务调用语句只能出现在过程结构中
CMP(C2,D2,out2);   end             //第二次调用任务
assign D=S? out1:out2;
endmodule
```

12.4 常用语句补充

12.4.4 任务和函数语句

1. 任务（task）语句

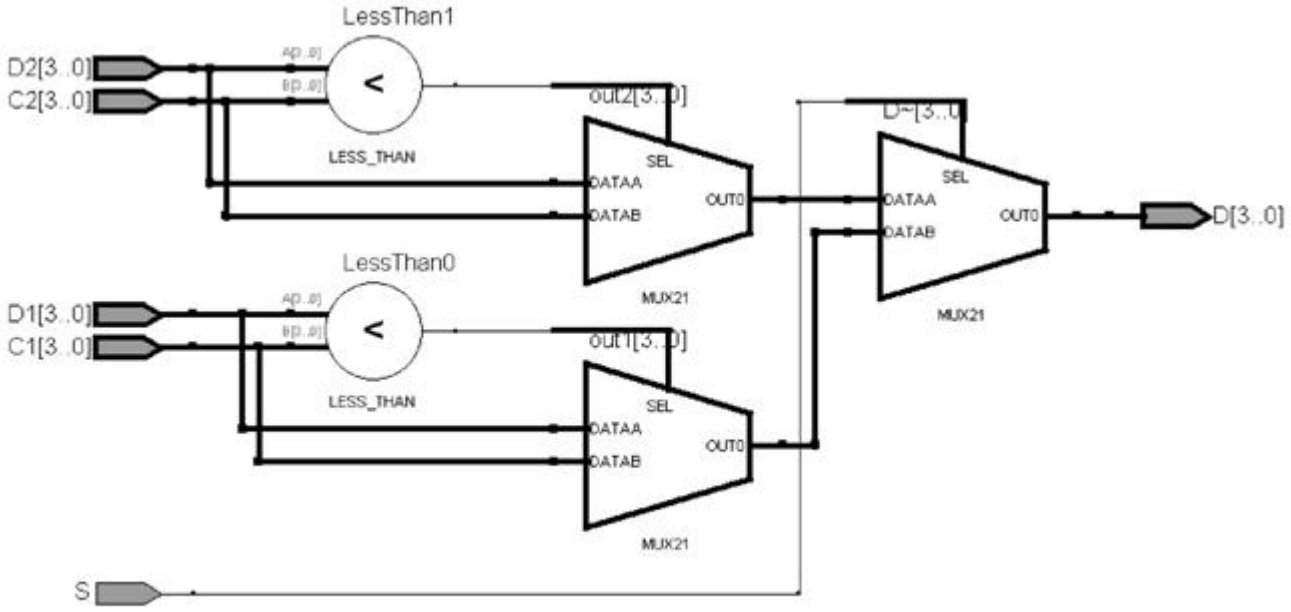

图 12-3　例 12-5 的 RTL 图

12.4 常用语句补充

12.4.4 任务和函数语句

2. 函数（function）语句

函数定义语句格式

```
function <位宽范围声明> 函数名;
        输入端口说明，其他类型变量定义;
        begin  过程语句;  end
endfunction
```

函数调用语句格式

```
<函数名>(输入参数1,输入参数2,…)
```

12.4 常用语句补充

12.4.4 任务和函数语句

2. 函数（function）语句

【例 12-6】

```
module CN (input[3:0] A, output[2:0] OUT);
function[2:0] GP;    //定义一个函数名为 GP 的函数，GP 同时作为位宽为 3 的输出参数
input[3:0] M;        //M 定义为此函数的输入值，位宽是 4
 reg[2:0] CNT,N;
 begin CNT=0;  for(N=0; N<=3; N=N+1)      //for 循环语句
if(M[N]==1)  CNT=CNT+1;  GP=CNT;  end      //含 1 的位个数累加
endfunction
assign OUT=(~|A) ? 0:GP(A); //主程序输入 A 或非缩位，若为 1 则输出函数计数结果
endmodule
```

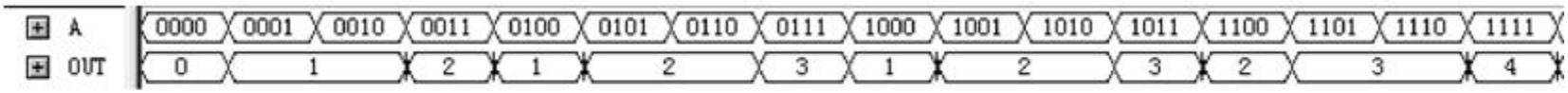

图 12-4　例 12-6 的仿真图

12.5 库元件和UDP用法介绍

12.5.1 Verilog原语库元件与用法

2. 函数（function）语句

【例 12-7】

```
module LOGICGATE (input A,B,C,S , output OUT);
wire a1,a2,a3,a4;
    not u1 (a1,B);
    and u2 (a2,A,a1);
    or  u3 (a3,C,B);
    xor u4 (a4,a3,a2);
 notif1 u5 (OUT,a4,S);
endmodule
```

12.5 库元件和UDP用法介绍

12.5.1 Verilog原语库元件与用法

2. 函数（function）语句

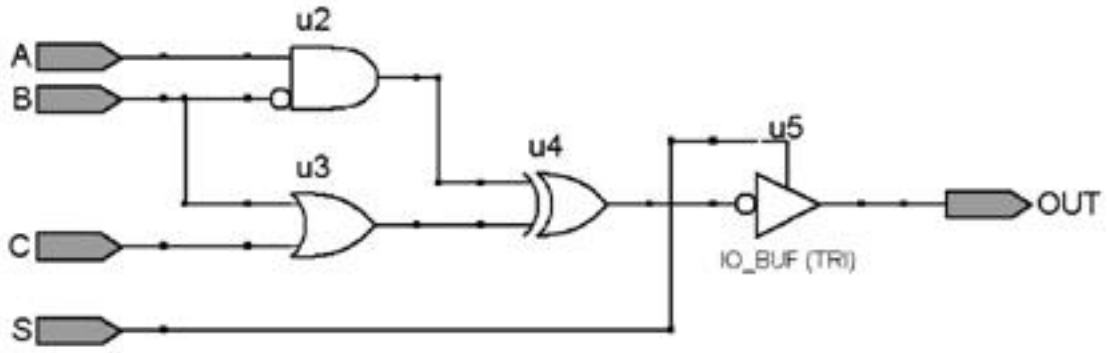

图 12-5 例 12-7 描述的逻辑电路

12.5 库元件和UDP用法介绍

12.5.1 Verilog原语库元件与用法

2. 函数（function）语句

基本门元件名 <门例化名> (<端口关联列表>)

(输出, 输入1, 输入2, 输入3,...);

```
and U1 (out,in1,in2,in3);          //三输入与门，例化名是U1
and U2 (out,in1,in2);              //二输入与门，例化名是U2

  bufif1 U1(out,in,enable);        //高电平使能的三态门
  bufif2 U2(out,a,ctr1);           //低电平使能的三态门

not IC1 (out1,out2,in);            //1输入in，2输出out1，out2
buf IC2 (out1,out2, out3,in);      //1输入in，3输出out1，out2，out3
```

12.5 库元件和UDP用法介绍

12.5.2 用户自定义原语UDP及用法示例

【例 12-8】

```
primitive XOR2(DOUT,X1,X2);
  input X1,X2;  output DOUT;
    table // X1  X2  : DOUT
              0   0   :   0;
            0   1   :   1;
            1   0   :   1;
            1   1   :   0;
    endtable
endprimitive
```

【例 12-9】

```
module H_ADDER (A,B,SO,CO);
    input A,B;
    output SO,CO;
    XOR2 U1(SO,A,B); // 调用元件 XOR2
    and U2(CO,A,B); // 调用元件 and
endmodule
```

12.5 库元件和UDP用法介绍

12.5.3 利用UDP元件设计多路选择器

【例 12-10】

```
primitive
MUX41_UDP(Y,D3,D2,D1,D0,S1,S0);
 input D3,D2,D1,D0,S1,S0; output Y;
   table //D3  D2  D1  D0  S1  S0  :  Y
           ?   ?   ?   1   0   0   :  1;
           ?   ?   ?   0   0   0   :  0;
           ?   ?   1   ?   0   1   :  1;
           ?   ?   0   ?   0   1   :  0;
           ?   1   ?   ?   1   0   :  1;
           ?   0   ?   ?   1   0   :  0;
           1   ?   ?   ?   1   1   :  1;
           0   ?   ?   ?   1   1   :  0;
   endtable
endprimitive
```

【例 12-11】

```
module MUX41UDP (D,S,DOUT);
     input[3:0] D;
     input[1:0] S;
     output DOUT;
    MUX41_UDP (DOUT,D[3],D[2],
       D[1],D[0],S[1],S[0]);
endmodule
```

12.5 库元件和UDP用法介绍

12.5.4 用UDP表述D触发器

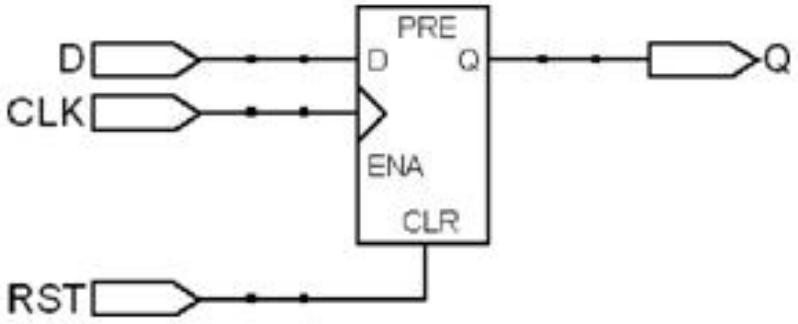

图 12-6 边沿 D 触发器

12.5 库元件和UDP用法介绍

12.5.4 用UDP表述D触发器

【例 12-12】

```
primitive EDGE_UDP(Q,D,CLK,RST);
 input D,CLK,RST; output Q; reg Q;
  table//D  CLK  RST  : Q  :  Q+
         0  (01)  0   : ?  :  0;
         1  (01)  0   : ?  :  1;
         ?  (1?)  0   : ?  :  -;
         ?  (?0)  0   : ?  :  -;
         1   0    1   : ?  :  0;
         1   1    1   : ?  :  0;
         0   0    1   : ?  :  0;
         0   1    1   : ?  :  0;
  endtable
endprimitive
```

【例 12-13】

```
module DFF_UDP (Q,D,CLK,RST);
  input D,CLK,RST;
  output Q;
  EDGE_UDP U1(Q,D,CLK,RST);
endmodule
```

12.6 其他仿真语句

12.6.1 fork_join块语句

【例 12-14】

```
module forkA(clk,a,b);
    input clk;
    output reg a, b;
    initial begin
      a=0;  b=0;  end
always @(posedge clk)
     fork
        #30 a = 1;
        #10 b = 1;
     join
endmodule
```

【例 12-15】

```
module forkB(clk,a,b);
    input clk;
    output reg a, b;
    initial begin
      a=0;  b=0;  end
always @(posedge clk)
     begin
        #30 a = 1;
        #10 b = a;
     end
endmodule
```

【例 12-16】

```
module forkC(clk,a,b);
    input clk;
    output reg a, b;
    initial begin
      a=0;   b=0;  end
always @(posedge clk)
     fork
        #30 a = 1;
        #10 b = a;
     join
Endmodule
```

12.6 其他仿真语句

12.6.1 fork_join块语句

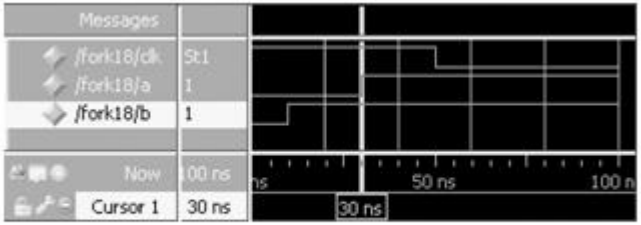

图 12-7 例 12-14 仿真波形

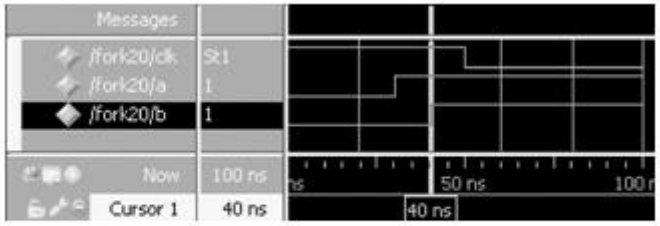

图 12-8 例 12-15 仿真波形

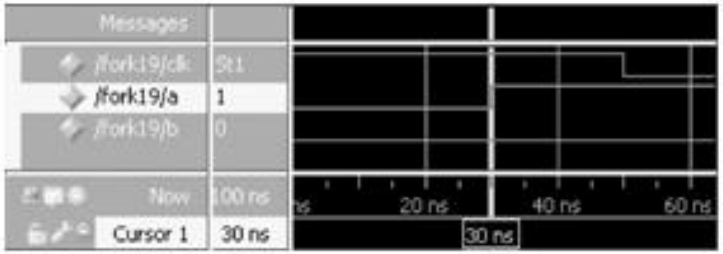

图 12-9 例 12-16 仿真波形

12.6 其他仿真语句

12.6.2 wait语句

```
wait (条件表达式) 语句;
```

```
forever  wait(start)  #10  go = ~go;
```

12.6 其他仿真语句

12.6.3 force语句和release语句

【例 12-17】

```
module testforce;                    //force 语句测试示例
   reg a, b, c, d;     wire e;                          #          0 d=0,e=0
   and and1 (e, a, b, c);                               #         10 d=1,e=1
   initial begin                     //监控 d、e 的变化
   $monitor("%d d=%b,e=%b", $stime, d, e);
   assign d = a & b & c;             //连续赋值 d
   a = 1;  b = 0;   c = 1;
   #10;                              //延迟 10 个时间单位
   force d = (a | b | c);            //强制赋值 d
   force e = (a | b | c);            //强制赋值 e
   #10 $stop;                        //暂停仿真                 #         20 d=0,e=0
   release d;                        //释放 d
   release e;                        //释放 e
   #10 $stop;                        //暂停仿真
   end
endmodule
```

12.6 其他仿真语句

12.6.4 deassign语句

```
always @(clear or preset)
  if (clear)
    assign q = 0;
  else if (preset)
    assign q = 1;
  else
    deassign q;
always @(posedge clock) q = d;
```

习　　题

12-3 用基于基本库元件的结构描述方法给出图**12-10**的**Verilog**描述。

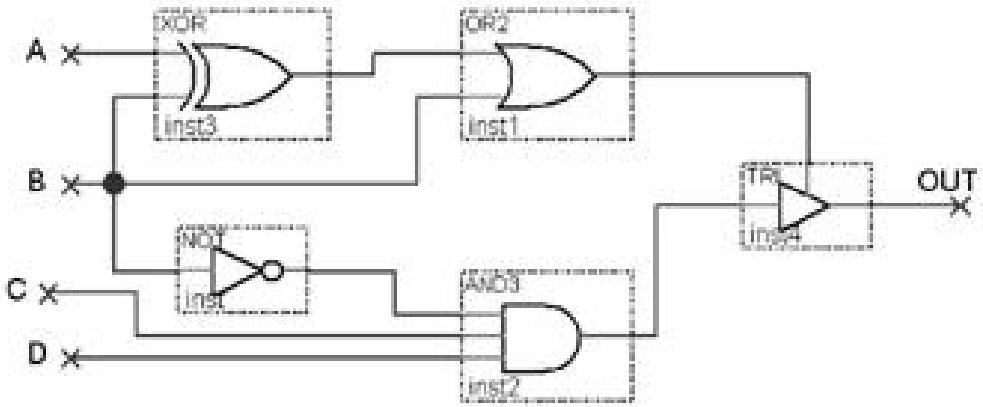

图 12-10　习题 12-3 逻辑电路图

实验与设计

12-1 SPWM脉宽调制控制系统设计

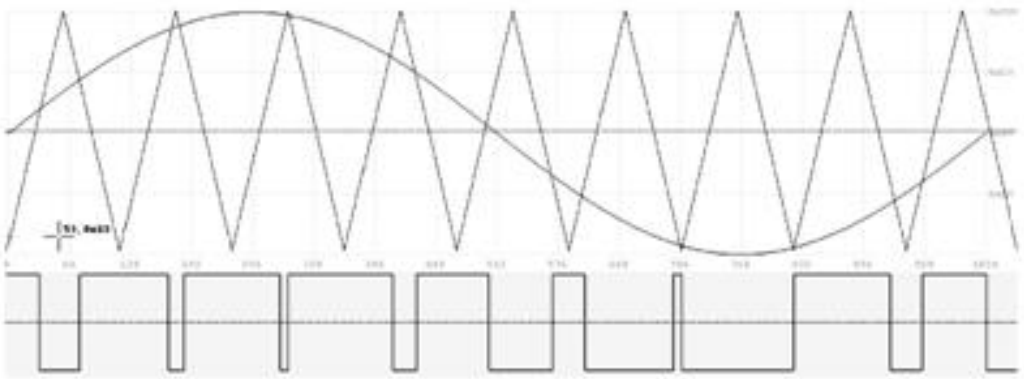

图 12-11　SPWM 波生成原理图

实验与设计

12-1 SPWM脉宽调制控制系统设计

【例 12-18】

```
module TRANG (input[9:0] ADR , output[9:0] OUTD);
   reg[9:0] OT1;    reg[10:0] CC;
   always @(ADR or CC)   begin
      if (ADR<10'H200)  begin  OT1[9:1]<=ADR[8:0]; OT1[0]<=1'b0;end
       else   begin   CC<=11'b10000000000 + (~ADR);
        OT1[9:1] <= CC[8:0];  OT1[0] <= 1'b0;      end    end
      assign OUTD = OT1;
endmodule
```

实验与设计

12-1 SPWM脉宽调制控制系统设计

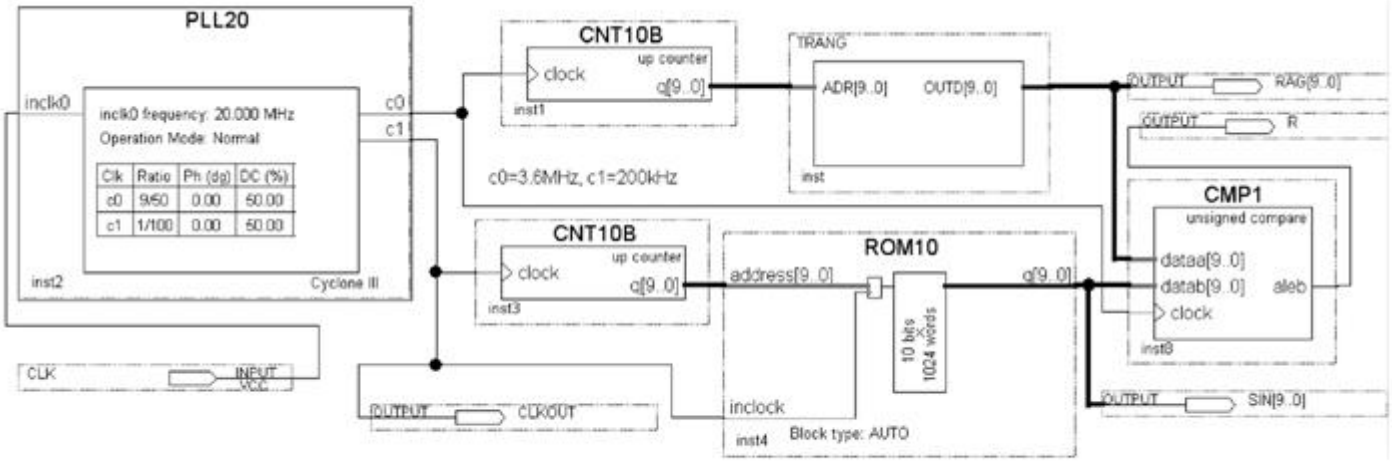

图 12-12　SPWM 波发生器基本电路图

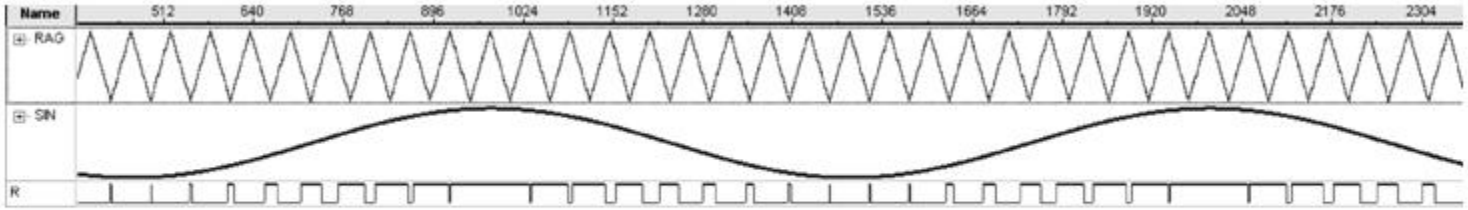

图 12-13　图 12-12 电路的 SignalTap II 实测波形

实验与设计

12-1 SPWM脉宽调制控制系统设计

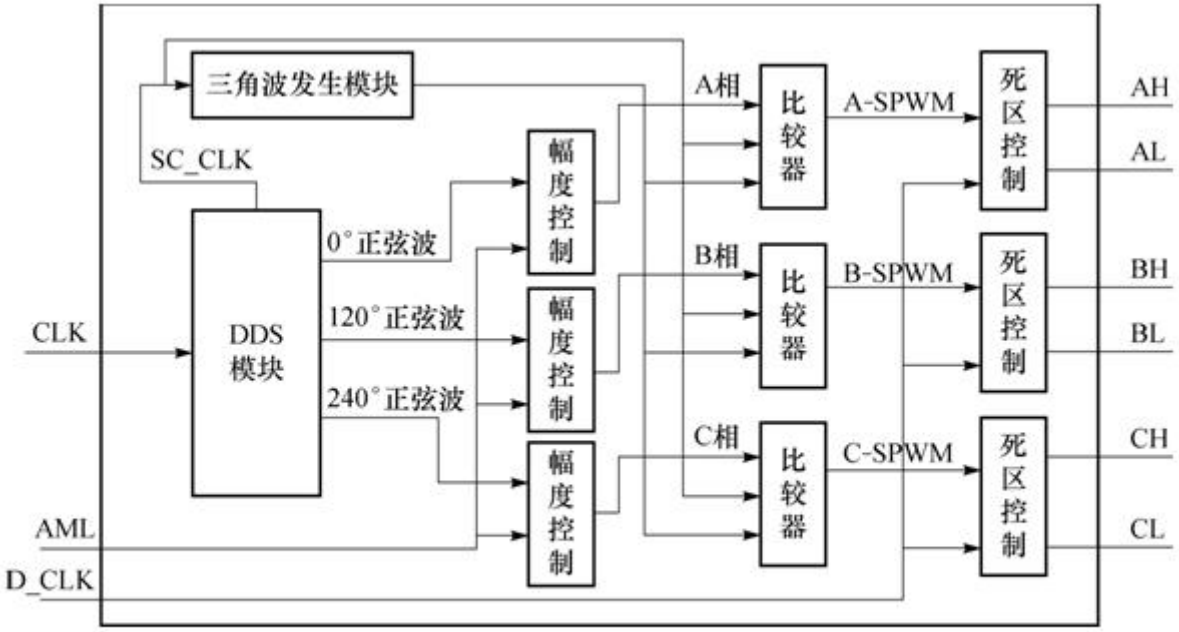

图 12-14　三相 SPWM 控制器电路模块图

实验与设计

12-4 AM幅度调制信号发生器设计

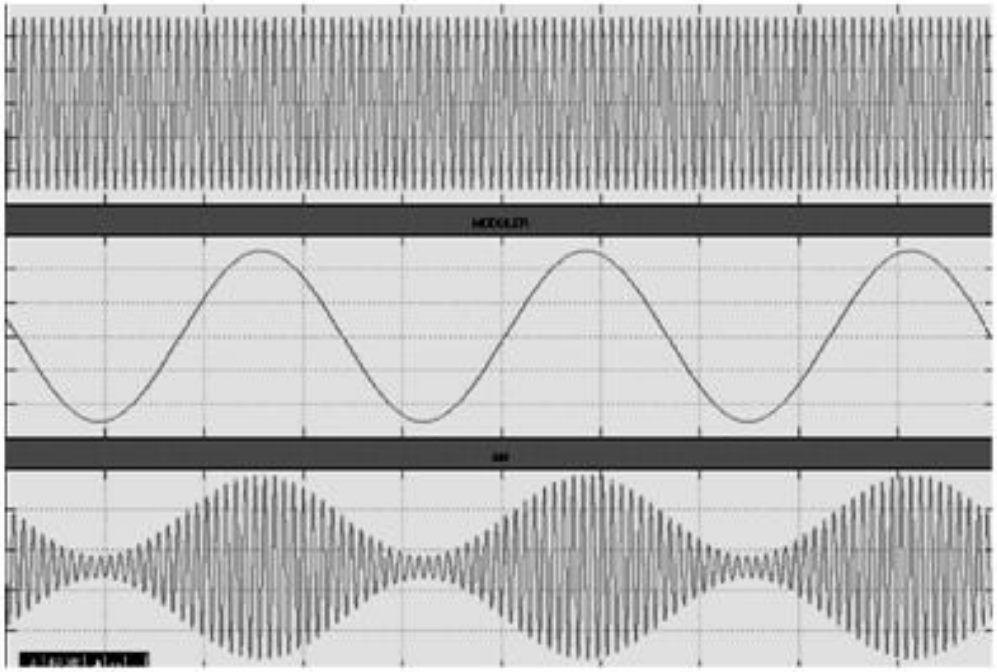

图 12-15　AM 模型仿真波形

实验与设计

12-5 VGA简单图像显示控制模块设计

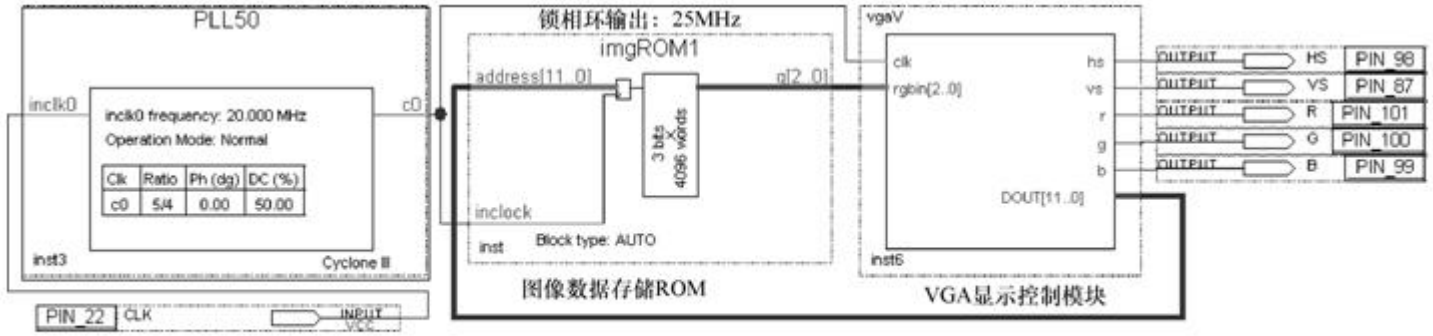

图 12-16　VGA 图像显示控制模块原理图

【例 12-19】

```
module vgaV (clk, hs, vs, r, g, b, rgbin, DOUT);
  input clk;           //工作时钟 25MHz
  output hs,vs;        //场同步、行同步信号
  output r,g,b;        // 红、绿、蓝信号
  input[2:0] rgbin;    //像素数据
  output[11:0] DOUT;  //图像数据 ROM 的地址信号
  reg[9:0] hcnt, vcnt;      reg r,g,b;      reg hs,vs;
  assign DOUT = {vcnt[5:0], hcnt[5:0]};
  always @(posedge clk) begin  //水平扫描计数器
    if (hcnt<800)   hcnt<=hcnt+1;
    else            hcnt<={10{1'b0}};
    end
  always @(posedge clk)   begin  //垂直扫描计数器
    if (hcnt==640+8)    begin
      if (vcnt<525)  vcnt<=vcnt+1;
      else    vcnt<={10{1'b0}};  end   end
  always @(posedge clk)   begin //场同步信号发生
    if ((hcnt>=640+8+8) & (hcnt<640+8+8+96))
     hs<=1'b0; else  hs<=1'b1;    end
  always @(vcnt)   begin        //行同步信号发生
    if ((vcnt>=480+8+2) & (vcnt<480+8+2+2))
      vs<=1'b0;  else  vs<=1'b1;  end
  always @(posedge clk)   begin
    if (hcnt<640 & vcnt<480)   //扫描终止
    begin  r<=rgbin[2];  g<=rgbin[1];  b<=rgbin[0];   end
    else begin  r<=1'b0; g<=1'b0;  b<=1'b0;  end
  end
endmodule
```