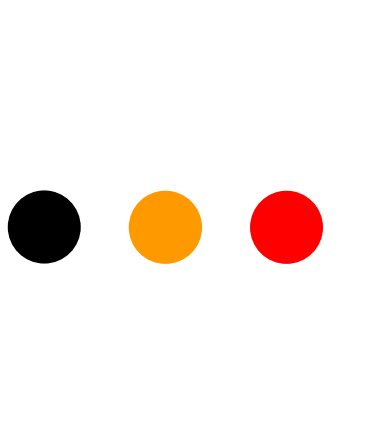

第13章

Verilog语句语法补遗

13.1 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。

13.1 Verilog文字规则

2. 实数

```
1.335,     88_670_551.453_909(=88670551.453909),     1.0,
44.99e-2(=0.4499),     0.1,     3E-4（=0.0003）
```

3. 字符串

```
"ERROR"  ,  "Both S and Q equal to 1" ,  "X"  ,  "BB$CC"
```

```
reg [8*5:1]  ALM; initial  begin ALM = "ERROR" ;  end
```

13.1 Verilog文字规则

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 *
```

13.2 数据类型

13.2.1 net网线类型

13.2.2 register寄存器类型

13.2.3 存储器类型

13.3 操作符

1. 逻辑操作符

例如设 A=4'b1001，B=4'b0001；则：

A && B=(1|0|0|1) & (0|0|0|1)=1&1=1'b1

1&z= 1'bz ;　　0&z= 1'b0 ;　1|z= 1'b1 ; 0|z= 1'bz;

- && 逻辑与
- || 逻辑或
- ! 逻辑非。例如 !A=0

2. 缩位操作符

&（与）、 **~&**（与非）、 **|**（或）、

~|（或非）、 **^**（异或）、 **^~，~^**（同或）。

13.4 常用语句补充

13.4.1 initial过程语句使用示例

【例 13-1】

```
`timescale 1ns/100ps  //声明仿真时间单位是 1ns，仿真精度也是 100ps
module test;          //定义 testbench 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

13.4 常用语句补充

13.4.2 forever循环语句

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

13.4 常用语句补充

13.4.3 编译指示语句

1. 文件包含语句`include

`include "文件名"

【例 13-2】

```
`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
```

13.4 常用语句补充

13.4.3 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	条件编译命令语句格式 2
`ifdef 宏名 语句块 `endif	`ifdef 宏名 语句块 1 `else 语句块 2 `endif

13.4 常用语句补充

【例 13-3】

```
`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
```

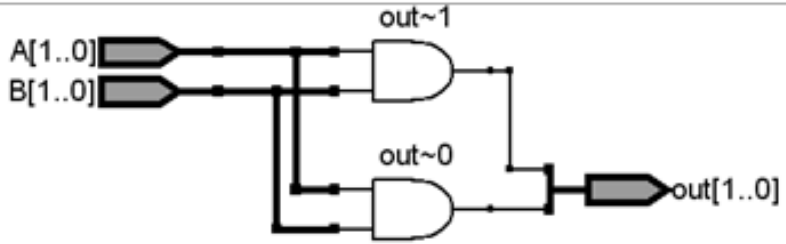

图 13-1 对应例 13-3 的 RTL 图

【例 13-4】

```
`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
```

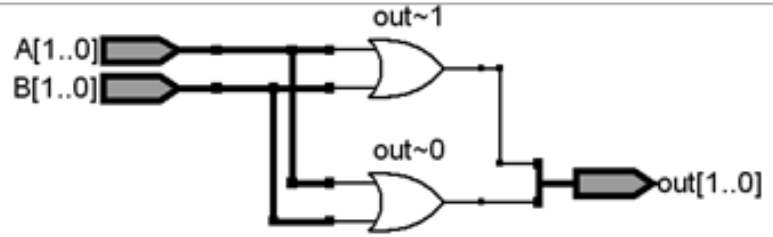

图 13-2 对应例 13-4 的 RTL 图

13.4 常用语句补充

13.4.4 任务和函数语句

1.任务（task）语句

任务（task）定义语句格式	任务调用格式
task <任务名>; 端口及数据类型声明语句 begin 过程语句； end endtask	<任务名>（端口1，端口2，…,端口N）;

13.4 常用语句补充

【例 13-5】

```
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
```

13.4 常用语句补充

13.4.4 任务和函数语句

1.任务（task）语句

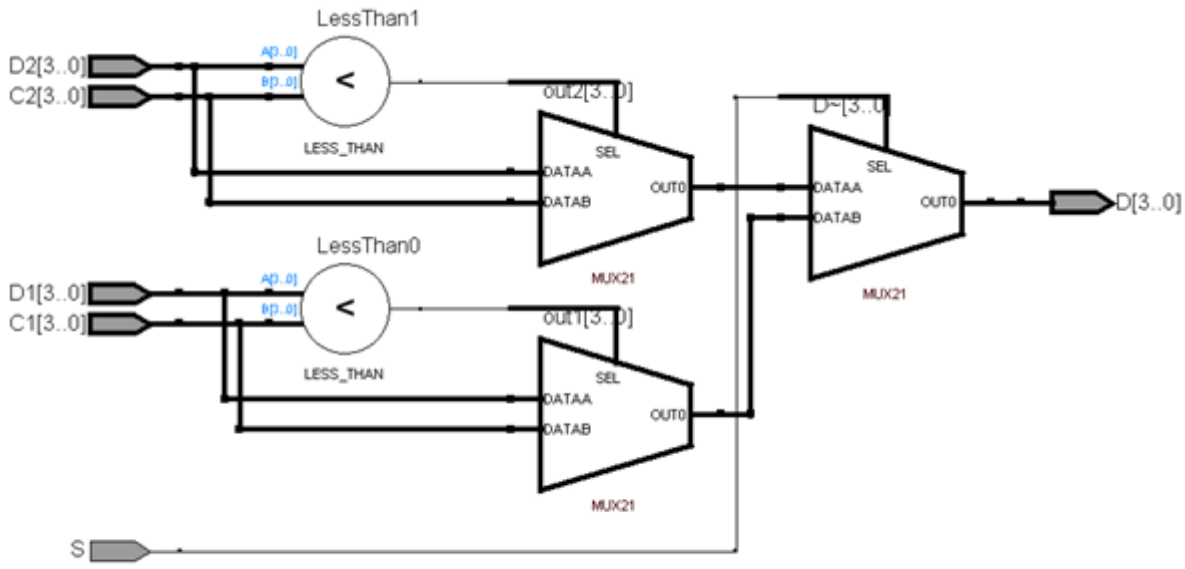

图 13-3 例 13-5 的 RTL 图

13.4 常用语句补充

13.4.4 任务和函数语句

2.函数（function）语句

函数定义语句格式	函数调用格式
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	<函数名>（输入参数 1，输入参数 2，…）

13.4 常用语句补充

【例 13-6】

```
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
```

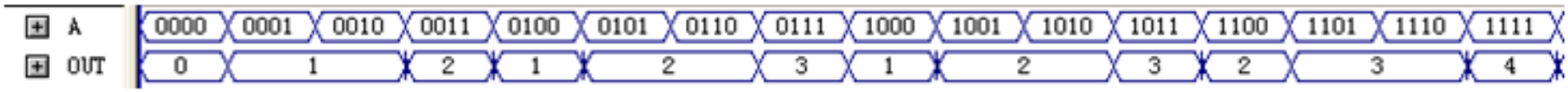

图 13-4 例 13-6 的仿真图

13.5 用库元件实现结构描述

【例 13-7】

```
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
```

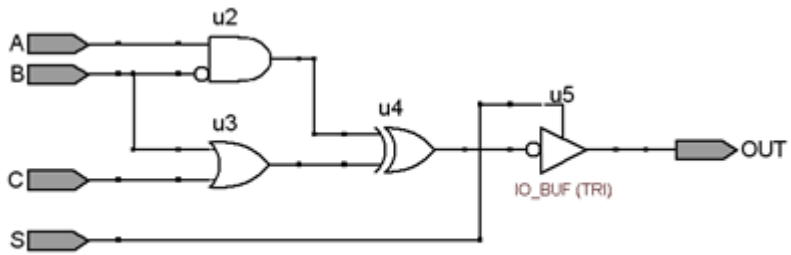

图 13-5 例 13-7 描述的逻辑电路

13.5 用库元件实现结构描述

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
```

习　　题

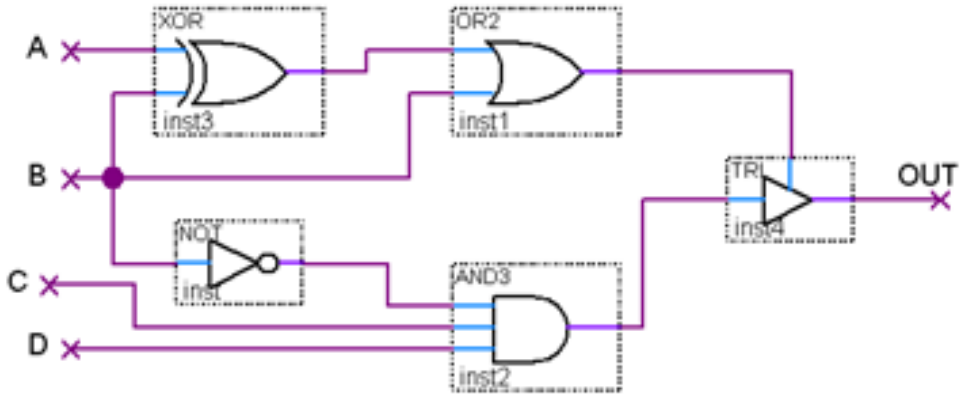

图 13-6　习题 13-3 逻辑电路图

实验与设计

13-1 SPWM脉宽调制控制系统设计

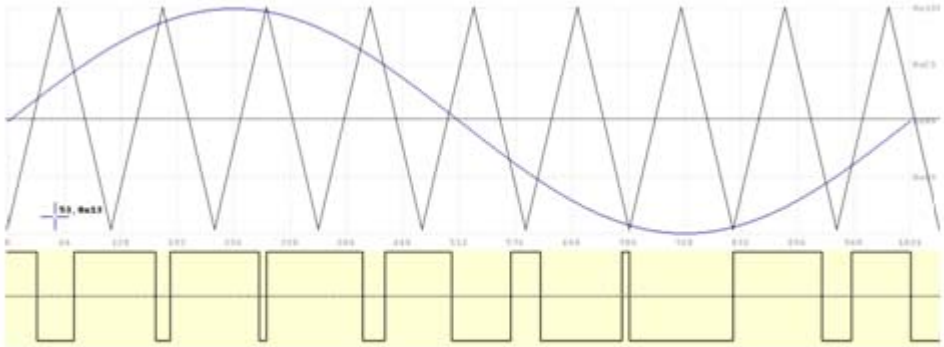

图 13-7 SPWM 波生成原理图

实验与设计

13-1 SPWM脉宽调制控制系统设计

【例 13-8】

```
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

13-1 SPWM脉宽调制控制系统设计

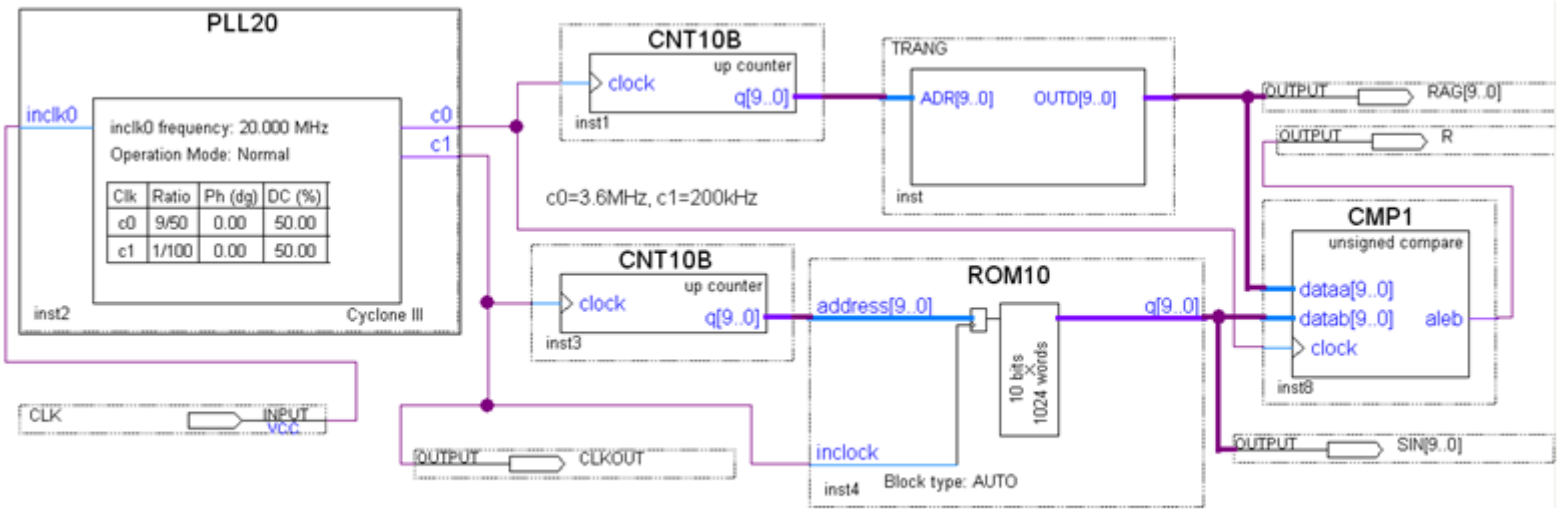

图 13-8 SPWM 波发生器基本电路图

实验与设计

13-1 SPWM脉宽调制控制系统设计

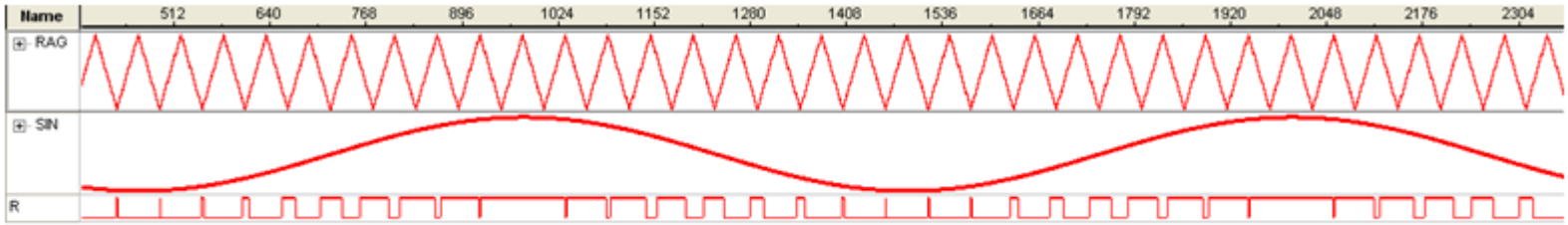

图 13-9 图 13-8 电路的 SignalTap II 实测波形

实验与设计

13-1 SPWM脉宽调制控制系统设计

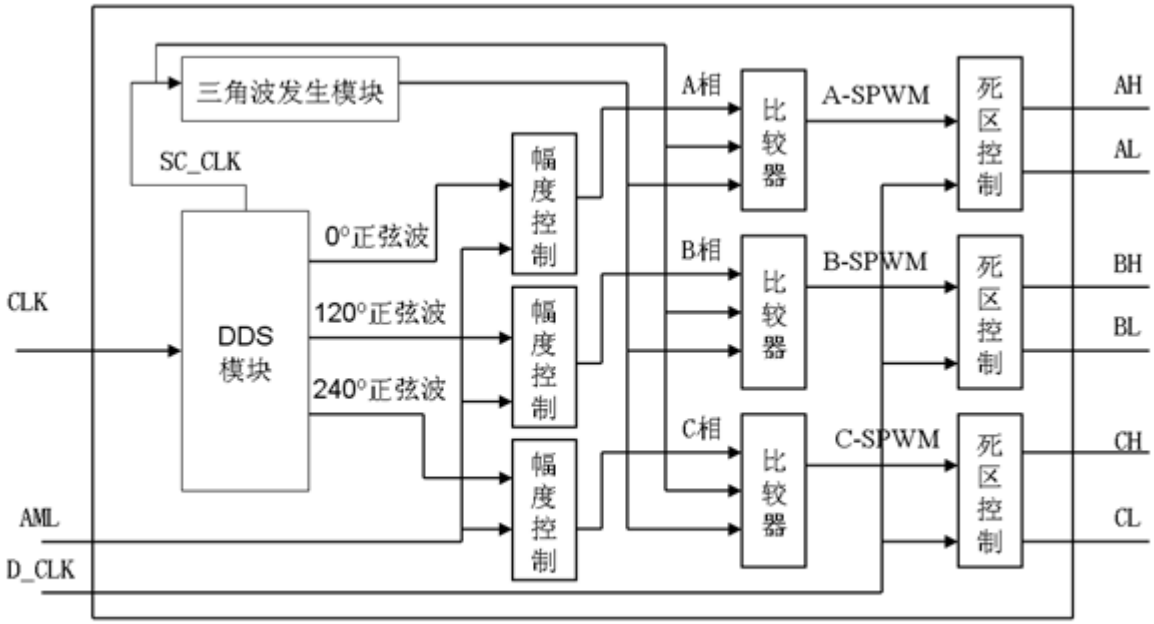

图 13-10 三相 SPWM 控制器电路模块图

实验与设计

13-2 点阵型与字符型液晶显示器驱动控制电路设计

13-3 数字彩色液晶显示控制电路设计

13-4 串行**ADC/DAC**控制电路设计