第5章

运算符与结构描述语句

5.1 运算操作符

5.1.1 按位逻辑操作符

表 5-1 按位逻辑操作符功能说明与用法示例

逻辑操作符	逻辑功能	A、B 逻辑操作结果	C、D 逻辑操作结果	C、E 逻辑操作结果
~	逻辑取反	~A=1'b1	~C=4'b0011	~E=6'b101001
|	逻辑或	A | B=1'b1	C | D=4'b1111	C | E=6'b011110
&	逻辑与	A&B=1'b0	C&D=4'b1000	C&E=6'b000100
^	逻辑异或	A^B=1'b1	C^D=4'b0111	C^E=6'b011010
~^ 或 ^~	逻辑同或	A~^B=1'b0	C~^D=4'b1000	C~^E=6'b100101

设：A=1'b0、B=1'b1、C[3:0]=4'b1100、D[3:0]=4'b1011、E[5:0]=6'b010110

5.1 运算操作符

5.1.2 逻辑运算操作符

- 逻辑与：&&
- 逻辑或：||
- 逻辑非： !

例如 设A=4'b1001，B=4'b0001；则：

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

5.1 运算操作符

5.1.3 算术运算操作符

表 5-2 算术操作符功能及其示例

逻辑操作符	功能	说明	示例
+	加		S=A+B=8'b00011000
−	减		S=B−A=8'b11111110
*	乘		S=A*B=8'b10001111=2'H8F
/	除	结果：小数抛弃	S=A/3=8'b00000100
%	求余	除法求余数	S=A%3=8'b00000001

设：A[3:0]=4'b1101、B[3:0]=4'b1011、定义 S 为 S[7:0]

5.1 运算操作符

5.1.3 算术运算操作符

【例 5-1】

```
module test1 (A,B,C,D,RCD,RAB,RM1,RM2,S,C0,R1,R2);
input [3:0] C,D ;  input signed [3:0] A,B;
output [3:0] RCD;  output [3:0] RAB;
output [7:0] RM1;  output [7:0] RM2;
output [3:0] S;   output C0;  output R1,R2;
reg [3:0] S ;      reg C0;
reg [3:0] RCD ;    reg [7:0] RM1 ;
reg signed [3:0] RAB;  reg signed [7:0] RM2;
reg R1,R2;
always@(A,B,C,D) begin
  RCD <= C+D;  RAB <= A+B;
  RM1 <= C*D;  RM2 <= A*B;
  {C0,S} <= {1'b0,C} - {1'b0,D};//注意并位操作
   R1  <= (C>D); R2<=(A>B);
  end
endmodule
```

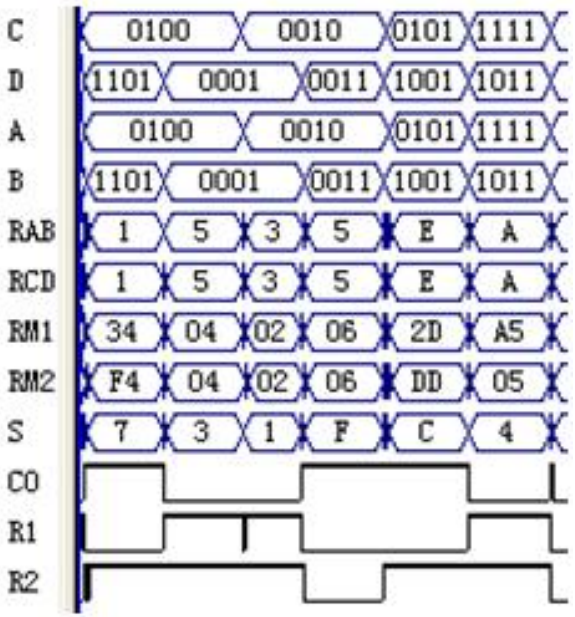

图 5-1 例 5-1 的仿真波形

5.1 运算操作符

5.1.4 关系运算操作符

表 5-3 等式操作符与示例

等式操作符	含义	等式操作示例
==	等于	(3==4)=0，(A==4'b1011)=1，(B==4'b1011)=0
!=	不等于	(D!=C)=1，(3!=4)=1
===	全等	(D===C)=1，(E===4'b0x10)=0
!==	不全等	(E!==4'b0x10)=1

设：A=5'b01011，B=4'b0010，C=4'b0z10，D=4'b0z10，E=3'bx10

表 5-4 不等式操作符与示例

等式操作符	含义	操作示例
>	大于	(A<B)=0，(A>B)=1 (A<20)=1，(A>12)=1 (A>=14)=0，(A<=13)=1
<	小于	
<=	小于或等于	
>=	大于或等于	

设：A=4'B1101，B=4'B0110

5.1 运算操作符

5.1.5 BCD码加法器设计示例

【例 5-2】

```
module BCD_ADDER (A,B,D) ;
  input [7:0] A,B;   output [8:0] D;
  wire [4:0] DT0, DT1 ; reg [8:0] D; reg S;
 always@ (DT0)
     begin  if (DT0[4:0] >= 5'b01010 )
    //如果低位 BCD 码的和大于等于 10,则使和加上 6，且有进位，使进位标志 S 等于 1。
              begin D[3:0] = (DT0[3:0]+4'b0110); S=1'b1; end
               else  begin D[3:0] = DT0[3:0] ; S=1'b0; end
     end //否则，将低位值赋予低位 BCD 码 D[3:0]输出，无进位，使进位标志 S 等于 0。
 always@ (DT1)   begin
     if (DT1[4:0]>=5'b01010)
    begin D[7:4] = (DT1[3:0]+4'b0110); D[8]=1'b1; end
       else begin D[7:4] = DT1[3:0] ; D[8]=1'b0;  end
     end
   assign DT0 = A[3:0] + B[3:0] ;    //设没有来自低位的进位。
   assign DT1 = A[7:4] + B[7:4] + S; //S 是来自低位 BCD 码相加的进位。
endmodule
```

5.1 运算操作符

5.1.5 BCD码加法器设计示例

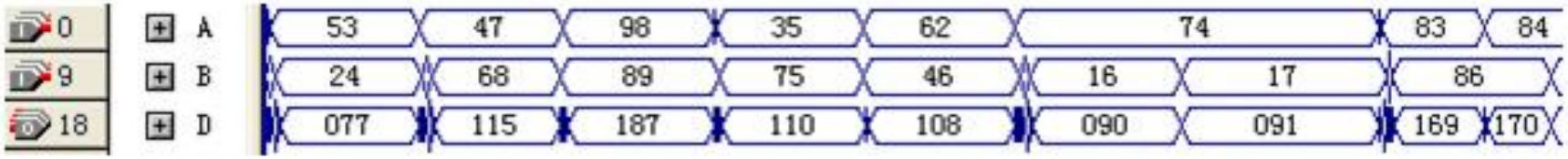

图 5-2 例 5-2 的仿真波形

5.1 运算操作符

5.1.6 缩位操作符

& （与） ~& （与非）

| （或） ~| （或非）

^ （异或） ^~，~^ （同或）

5.1.7 并位操作符

{a1, b1, 4{a2,b2}}={ a1, b1, {a2,b2},{a2,b2},{a2,b2},{a2,b2}}={a1,b1,a2,b2,a2,b2,a2,b2,a2,b2}

5.1 运算操作符

5.1.8 移位操作符用法

V >> n 或 V <<n

V >> 1 的值是 8'b01100100 ;

V << 3 的值是 8'b01001000

V >>> n 或 V <<< n

```
output signed[7:0] y;
 input signed[7:0] a;
 assign y = (a<<<2);
```

若 a=10101011,则输出 y=10101100

若 a=10001111,则输出 y=00111100

```
parameter  C=8'sb10101011;
parameter  D=8'sb01001110;
 output [7:0] Y1,Y2;
assign Y1=(C>>>2);//结果：Y1=11101010
assign Y2=(D>>>2);//结果：Y2=00010011
```

5.1 运算操作符

5.1.9 移位操作符用法示例

【例 5-3】

```
module MULT4B(R,A,B);
 parameter S=4;
 output[2*S:1] R ;
 input[S:1] A,B ;
  reg[2*S:1] R;
    integer i;
     always @(A or B)
      begin
      R = 0 ;
      for(i=1; i<=S; i=i+1)
      if(B[i]) R=R+(A<<(i-1));
      end
endmodule
```

【例 5-4】

```
module MULT4B (R,A,B);
 parameter S=4;
 output[2*S:1] R;
   input[S:1] A,B;
 reg[2*S:1] R,AT;
  reg[S:1] BT,CT;
 always @(A,B )
    begin
    R=0; AT = {{S{1'B0}},A};
    BT = B; CT = S;
    for(CT=S; CT>0; CT=CT-1)
      begin if(BT[1]) R=R+AT;
       AT = AT<<1; BT = BT>>1;
    end
 end endmodule
```

5.1 运算操作符

5.1.9 移位操作符用法示例

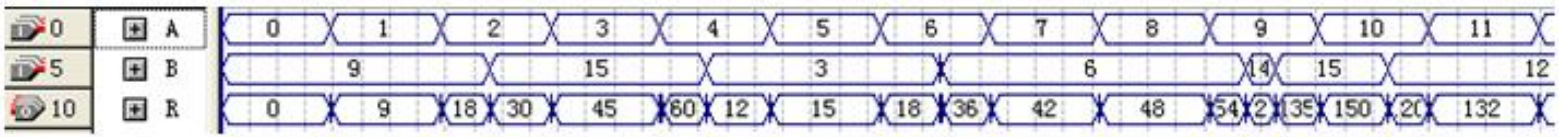

图 5-3　四位乘法器时序仿真图

【例 5-5】

```
module SHIF4(DIN,CLK,RST,DOUT);
 input CLK,DIN,RST;
 output DOUT;
 reg [3:0] SHFT;
 always@(posedge CLK or posedge RST)
     if(RST) SHFT<=4'B0;
     else begin   SHFT[3]<=DIN;
       SHFT[2:0] <= SHFT[3:1];
     end
     assign DOUT=SHFT[0];
endmodule
```

【例 5-6】

```
module SHIF5 (DIN,CLK,RST,DOUT);
 input CLK,DIN,RST;   output DOUT;
 reg [3:0] SHFT;
 always@(posedge CLK or posedge RST)
   if(RST) SHFT<=4'B0;
     else begin
       SHFT <= (SHFT >> 1);
       SHFT[3] <= DIN;
     end
   assign DOUT = SHFT[0];
endmodule
```

5.1 运算操作符

5.1.10 条件操作符

条件表达式 ? 表达式1 : 表达式2

【例 5-7】

```
module DFF2 (input CLK, input D, input RST , output reg Q );
    always @(posedge CLK ) ...
        Q <= RST ? 1'b0 : D;
endmodule
```

5.2 连续赋值语句

```
assign  目标变量名 = 驱动表达式;
```

```
assign [延时] 目标变量名 = 驱动表达式;
```

```
`timescale 10ns/100ps
assign #6 R1 = A & B;
```

【例 5-8】

```
module MUX41a (A,B,C,D,S1,S0,Y);
   input A,B,C,D,S1,S0;
   output Y;
   assign AT = S0 ? D : C ;
   assign BT = S0 ? B : A ;
   wire  Y = (S1 ? AT : BT);
endmodule
```

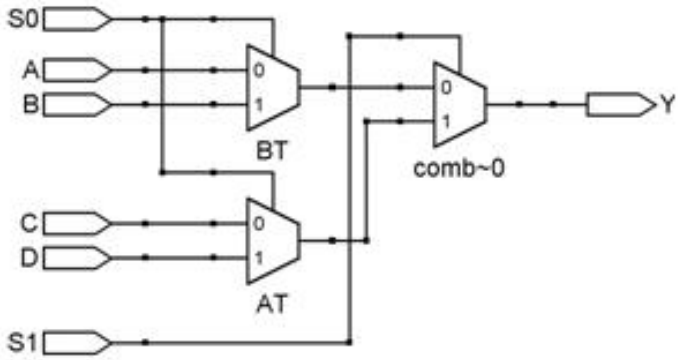

图 5-4　例 5-8 的 RTL 图

5.3 例化语句

5.3.1 半加器设计

【例 5-9】

```
module h_adder (A,B,SO,CO) ;
    input A,B;
    output SO,CO;
    assign SO = A ^ B;    //将变量 A 和 B 执行异或逻辑后将结果赋给输出信号 SO
    assign CO = A &B;     //将变量 A 和 B 执行与逻辑后将结果赋给输出信号 CO
endmodule
```

5.3 例化语句

5.3.2 全加器设计

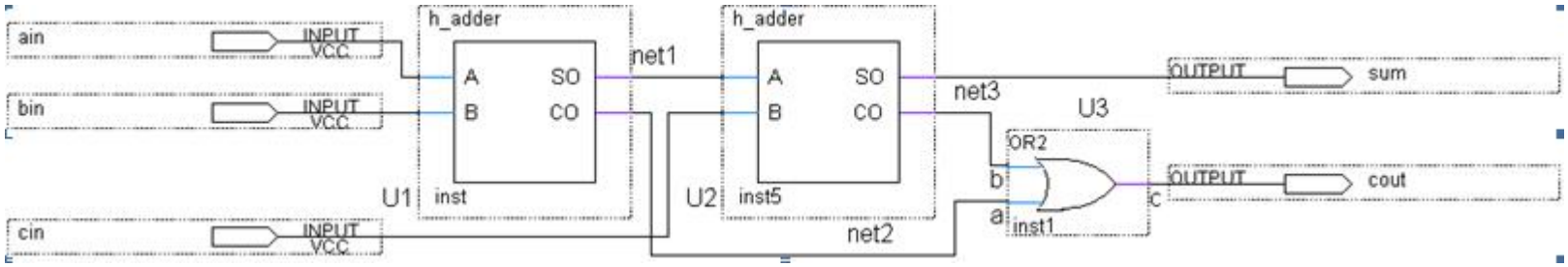

图4-32 全加器f_adder电路图

【例 5-10】

```
module f_adder(ain,bin,cin,cout,sum);
    output cout,sum;    input ain,bin,cin;
    wire  net1,net2,net3;
   h_adder  U1(  ain, bin, net1, net2);
   h_adder  U2(.A(net1), .SO(sum), .B(cin), .CO(net3) );
        or  U3(cout, net2, net3);
endmodule
```

5.3 例化语句

5.3.3 Verilog例化语句及其用法

1. 例化语句端口名关联法

```
<模块元件名>  <例化元件名> ( .例化元件端口（例化元件外接端口名）,...）;
```

```
h_adder  U2(.A(net1), .SO(sum), .B(cin),.CO(net3));
```

```
h_adder  U2(.B(cin), .CO(net3), .A(net1), .SO(sum));
```

2. 例化语句位置关联法

5.4 参数传递语句应用

【例 5-11】

```
module MULT4B
#(parameter S)(R,A,B);
 output[2*S:1] R ;
 input[S:1] A,B ;
  reg[2*S:1] R;   integer i;
... //以下与例 5-3 相同
```

【例 5-12】

```
module MULTB (RP,AP,BP);
 output[15:0] RP ;
 input[7:0] AP,BP ;
   MULT4B #(.S(8))
  U1(.R(RP), .A(AP), .B(BP) );
 endmodule
```

```
module SUB_E
  #(parameter S1=4, parameter S2=5, parameter S3=2)(A,B,C);
```

```
SUB_E  #(.S1(8), .S2(9), .S3(7)) U1(.C(CP), .A(AP), .B(BP) );
```

5.5 用库元件实现结构描述

【例 5-13】

```
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
```

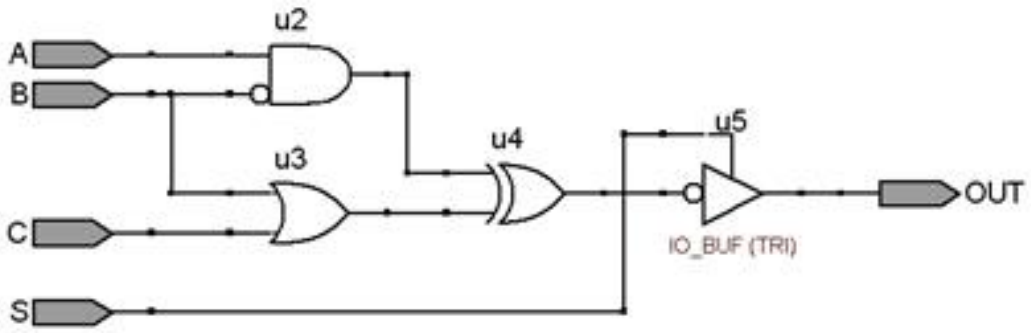

图 5-5 例 5-13 描述的逻辑电路

5.5 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
```

5.6 编译指示语句

5.6.1 宏定义命令语句

```
`define  宏名（标志符）  宏内容（字符串）
```

```
`define  s   A+B+C+D

assign   DOUT= `s + E;

assign   DOUT = A+B+C+D+E;
```

5.6 编译指示语句

5.6.2 文件包含语句`include

`include “文件名”

【例 5-14】

```
'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
```

5.6 编译指示语句

5.6.3 条件编译命令语句' 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

5.7 编译指示语句

5.6.3 条件编译命令语句'ifdef、'else、'endif

【例 5-15】

```
'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
```

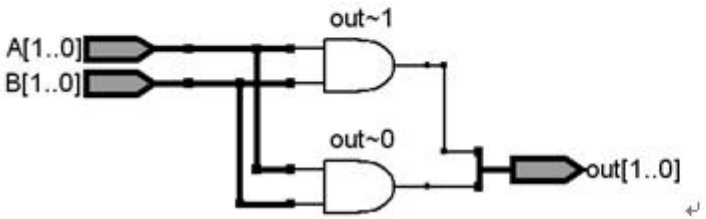

图 5-6 例 5-15 的 RTL 图

【例 5-16】

```
'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
```

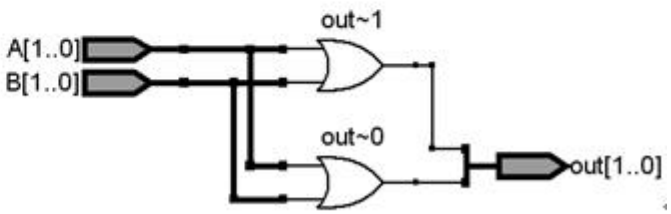

图 5-7 例 5-16 的 RTL 图

5.7 keep属性应用

【例 5-17】

```
module ff_adder(ain,bin,cin,cout,sum);
    output cout,sum ;    input ain,bin,cin ;
    (* synthesis, keep *) wire net3 ;
    wire net2,net1 ;
    ...  //以下与例5-10相同
```

(* synthesis, keep *) 或 (* synthesis, probe_port, keep *)

5.7 keep属性应用

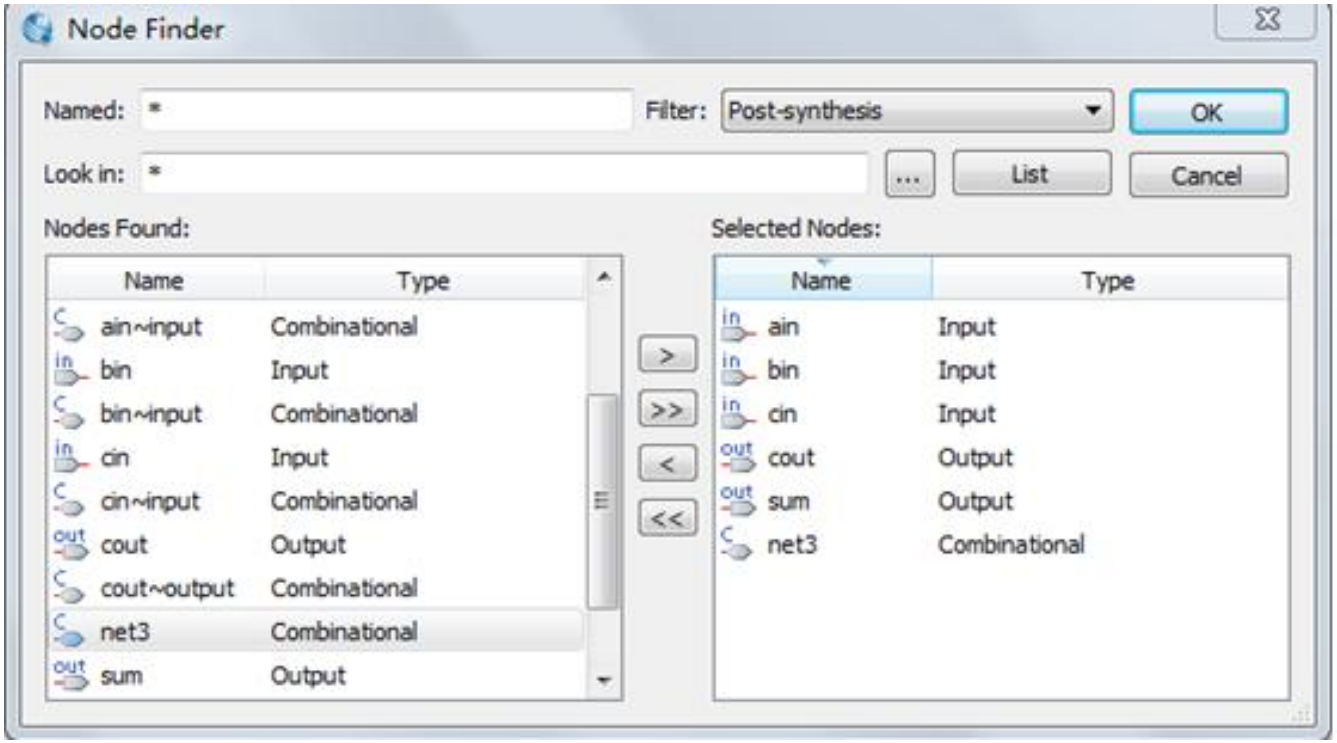

图5-8 加入仿真测试信号net3

5.7 keep属性应用

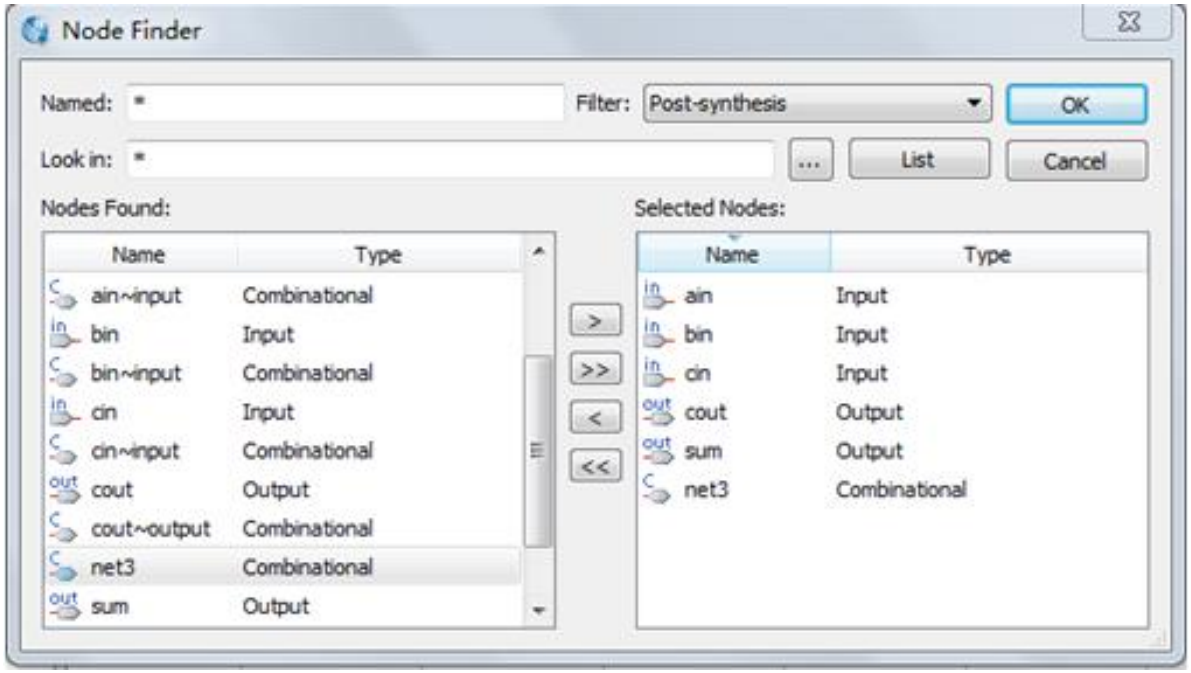

图5-9 例5-17的仿真波形

5.8 SignalProbe使用方法

1. 按常规流程完成设计仿真和硬件测试

2. 设置SignalProbe Pins

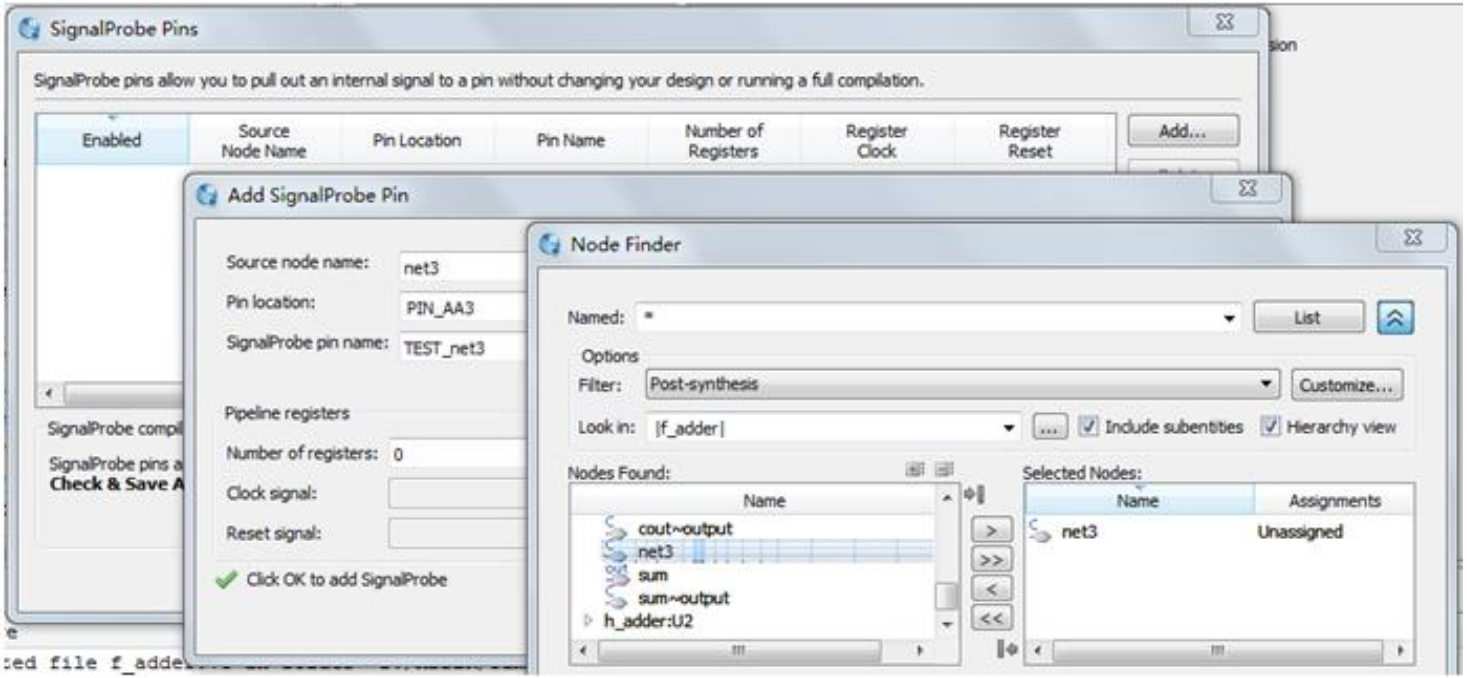

图5-10 在SignalProbe对话框设置探测信号net3

5.8 SignalProbe使用方法

3. 编译SignalProbe Pins测试信息并下载测试

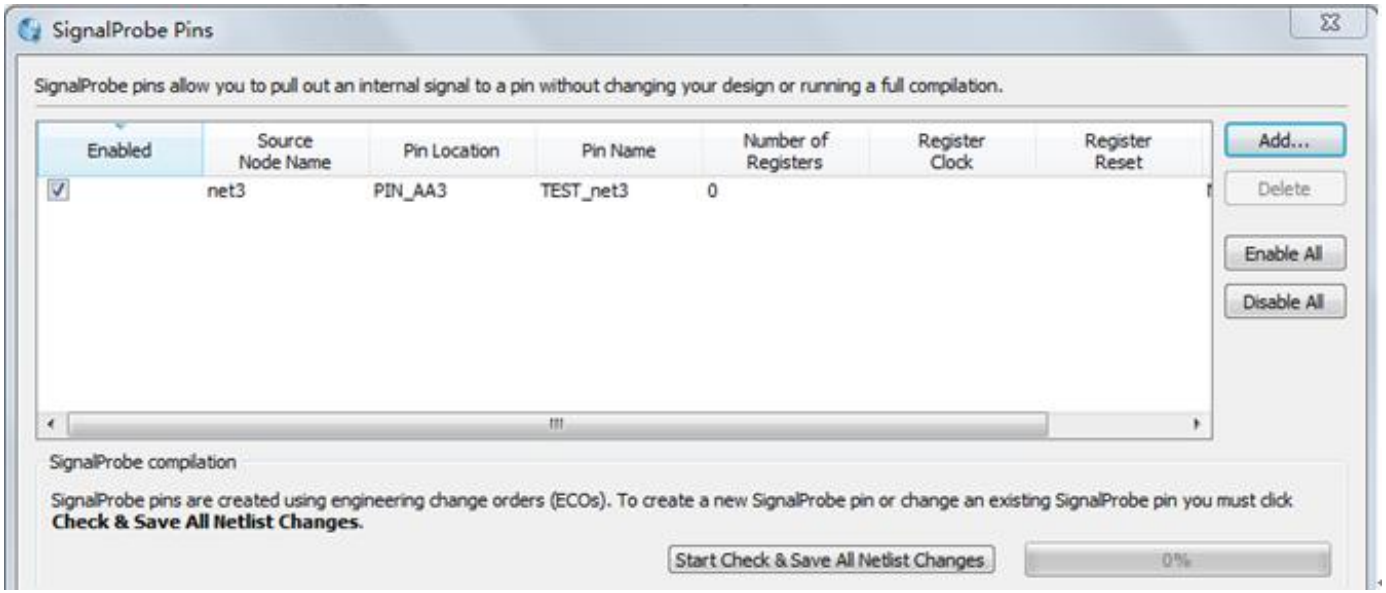

图5-11 SignalProbe Pins对话框设置情况

习　题

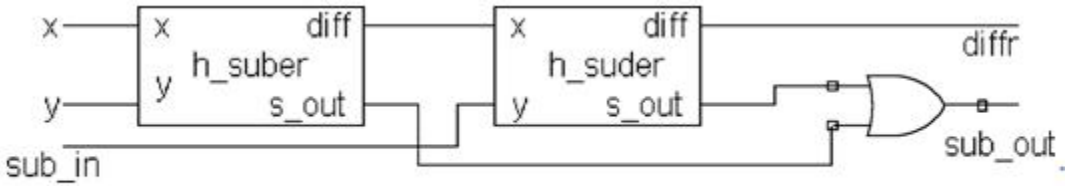

图 5-12　全减器模块图

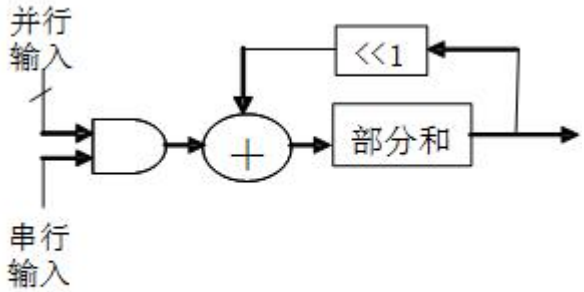

图 5-13　最基本的硬件乘法器

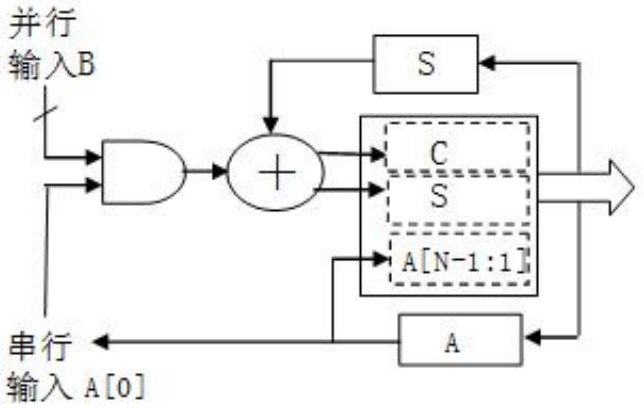

图 5-14　改进后的硬件乘法器

习　题

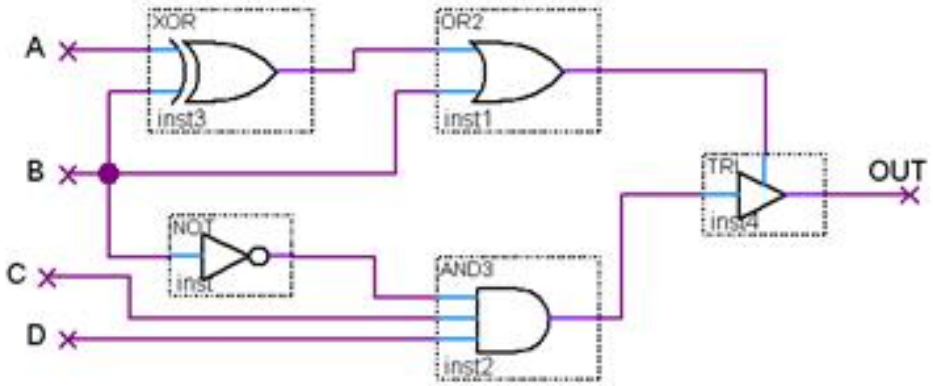

图 5-15 · · 习题 5-15 逻辑电路图

实验与设计

实验5-1 高速硬件除法器设计实验

【例 5-18】

```
module DIV16 (input CLK,input [15:0] A,B, output reg [15:0] QU,RE);
    reg [15:0] AT,BT,P,Q; integer i;
    always @(posedge CLK) begin
        AT = A ; BT = B; P = 16'H0000; Q = 16'H0000 ;
        for (i=15; i>=0; i=i-1)
        begin P={P[14:0], AT[15]};
              AT={AT[14:0],1'B0} ; P=P-BT;
        if (P[15]==1) begin
            Q[i]=0; P = P+BT ;
        end
        else Q[i]=1 ; end
    end
  always @( * ) begin QU = Q; RE = P ; end
endmodule
```

实验与设计

实验5-2 不同类型的移位寄存器设计实验

实验5-3 基于Verilog代码的频率计设计

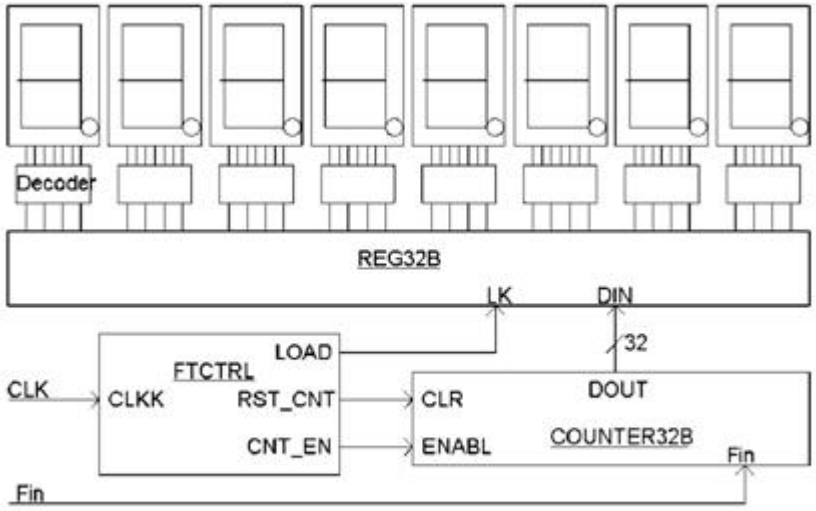

图 5-16 频率计电路图

实验与设计

实验5-3 基于Verilog代码的频率计设计

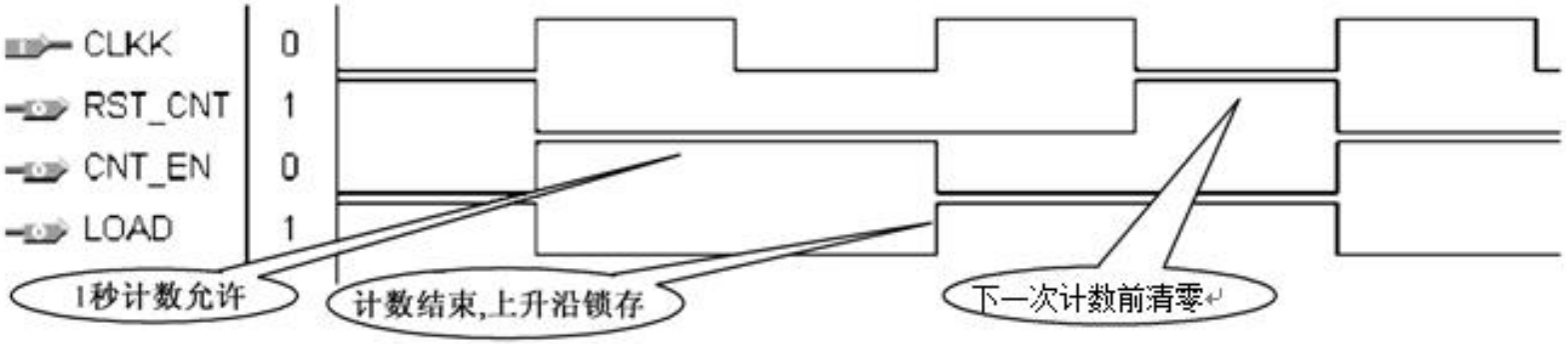

图 5-17 频率计测频控制器 FTCTRL 测控时序图

实验与设计

实验5-3 基于Verilog代码的频率计设计

【例 5-19】

```
module FTCTRL (CLKK, CNT_EN, RST_CNT, LOAD);
  input CLKK;        output CNT_EN, RST_CNT,LOAD;
  wire CNT_EN, LOAD;        reg RST_CNT,Div2CLK;
  always @(posedge CLKK)  Div2CLK <= ~Div2CLK ;
  always @(CLKK or Div2CLK) begin
    if (CLKK==1'b0 & Div2CLK==1'b0) RST_CNT <= 1'b1 ;
      else RST_CNT <= 1'b0 ;      end
  assign LOAD = ~Div2CLK ;        assign CNT_EN = Div2CLK ;
endmodule
```

实验与设计

实验5-4 8位加法器设计实验

实验5-5 VGA彩条信号显示控制电路设计

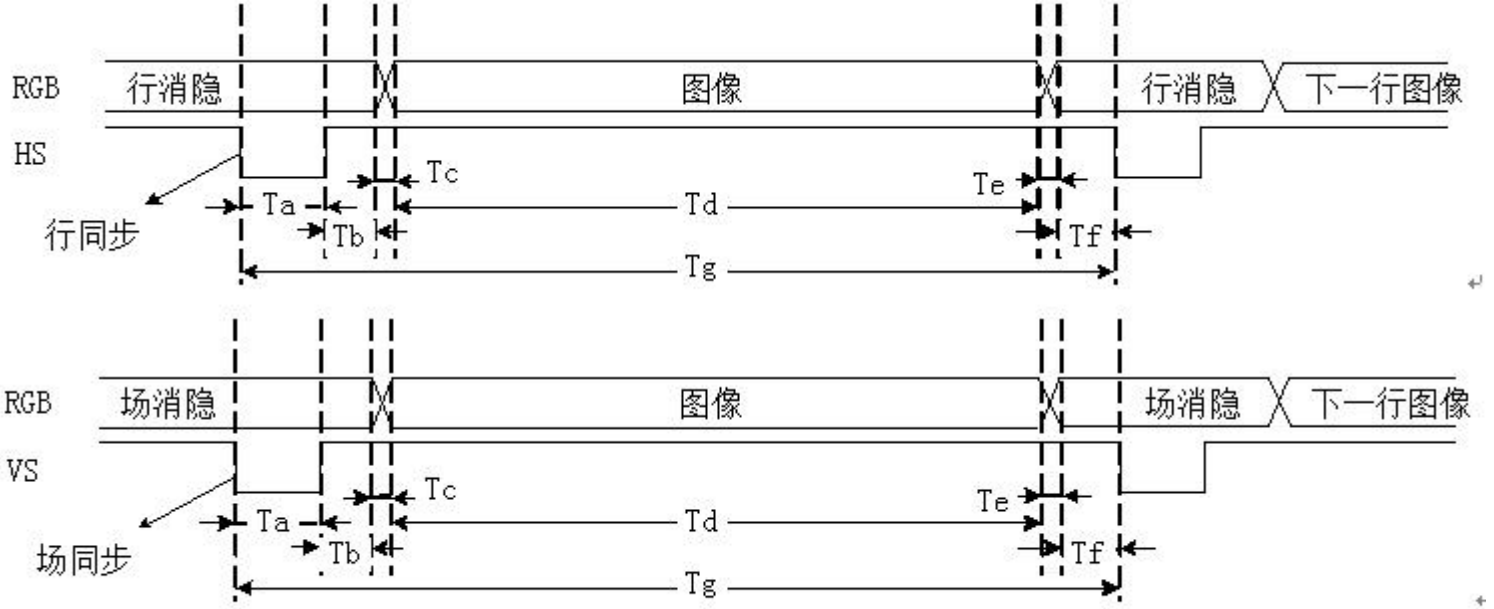

图 5-18 VGA 行扫描、场扫描时序示意图

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

表 5-6 行扫描时序要求（单位：像素，即输出一个像素 Pixel 的时间间隔）

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间（Pixels）	8	96	40	8	640	8	800

表 5-7 场扫描时序要求（单元：行，即输出一行 Line 的时间间隔）

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间（Lines）	2	2	25	8	480	8	525

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

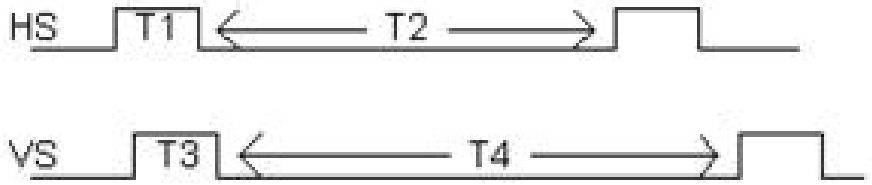

图 5-19 HS 和 VS 的时序图

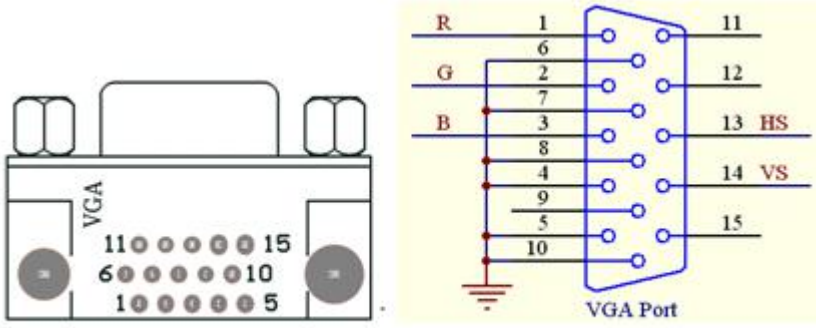

图 5-20 VGA 接口电路图，左接口从上往下看

实验与设计

实验5-5 VGA彩条信号显示控制电路设计

表 5-8 颜色编码

颜色	黑	蓝	红	品	绿	青	黄	白
R	0	0	0	0	1	1	1	1
G	0	0	1	1	0	0	1	1
B	0	1	0	1	0	1	0	1

表 5-9 彩条信号发生器 3 种显示模式

1	横彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
2	竖彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
3	棋盘格	1：棋盘格显示模式 1	2：棋盘格显示模式 2

【例5-20】

```
module VGA_COLOR_LINE (CLK, MD, HS, VS, R, G, B);//VGA显示器 彩条 发生器
  input CLK, input MD;       output HS, VS, R,  G, B;
    wire R,G,B,VS,HS;                  //红，绿，蓝信号，和场同步，行同步信号
    wire FCLK, CCLK;    reg  HS1, VS1;   reg[1:0] MMD;   reg[4:0] FS;
  reg[4:0] CC;                        //行同步，横彩条生成
  reg[8:0] LL;                        //场同步，竖彩条生成
  reg[3:1] GRBX,GRBY,GRBP;            // X横彩条，Y竖彩条
  wire[3:1] GRB;
  assign GRB[2] = (GRBP[2] ^ MD) & HS1 & VS1 ;
  assign GRB[3] = (GRBP[3] ^ MD) & HS1 & VS1 ;
  assign GRB[1] = (GRBP[1] ^ MD) & HS1 & VS1 ;
  always @(posedge MD)  begin
        if (MMD==2'b10) MMD<=2'b00; else MMD<=MMD+1 ;  end //3种模式
  always @(MMD)   begin
         if (MMD == 2'b00)  GRBP <= GRBX ; // 选择横彩条
    else if (MMD == 2'b01)  GRBP <= GRBY ; // 选择竖彩条
    else if (MMD == 2'b10)  GRBP <= GRBX ^ GRBY ; //产生棋盘格
    else GRBP <= 3'b000 ;    end
  always @(posedge CLK ) begin                    // 20MHz 21分频
        if (FS==20)  FS<=0;  else  FS<=(FS+1) ;  end
  always @(posedge FCLK)   begin
        if (CC==29)  CC<=0;  else  CC<=CC+1 ;    end
```

接下页

```
   always @(posedge CCLK)   begin
         if (LL==481)   LL<=0; else  LL<=LL+1 ;    end
   always @(CC or LL)   begin
         if (CC > 23)  HS1<=1'b0; else HS1<=1'b1 ; //行同步
         if (LL > 479) VS1<=1'b0; else VS1<=1'b1 ;     end //场同步
   always @(CC or LL)   begin
     if (CC < 3)  GRBX <= 3'b111 ;                  // 横彩条
     else if (CC < 6)  GRBX <= 3'b110 ;
     else if (CC < 9)  GRBX <= 3'b101 ;
     else if (CC < 12) GRBX <= 3'b100 ;
     else if (CC < 15) GRBX <= 3'b011 ;
     else if (CC < 18) GRBX <= 3'b010 ;
     else if (CC < 21) GRBX <= 3'b001 ;
     else  GRBX <= 3'b000 ;
     if (LL < 60)  GRBY <= 3'b111 ; // 竖彩条
     else if (LL < 120)  GRBY <= 3'b110 ;
     else if (LL < 180)  GRBY <= 3'b101 ;
     else if (LL < 240)  GRBY <= 3'b100 ;
     else if (LL < 300)  GRBY <= 3'b011 ;
     else if (LL < 360)  GRBY <= 3'b010 ;
     else if (LL < 420)  GRBY <= 3'b001 ;
     else    GRBY <= 0 ;    end
   assign HS = HS1 ;    assign FCLK = FS[3] ;
   assign HS = HS1 ;    assign VS = VS1   ;  assign R = GRB[2] ;
   assign G = GRB[3] ;  assign B = GRB[1] ;  assign CCLK = CC[4] ;
endmodule
```