第7章

Verilog设计深入

7.1 过程中的两类赋值语句

7.1.1 未指定延时的阻塞式赋值语句

目标变量名 = 驱动表达式;

7.1.2 指定了延时的阻塞式赋值

[延时] 目标变量名 = 驱动表达式;

目标变量名 = [延时] 驱动表达式;

【例 7-1】	【例 7-2】

.....//过程语句
Y1 = A^B;
#6 Y2 = A &B | C;
	..... //过程语句
	Y1 = A^D;
	Y2 = #6 A &E | C;

7.1 过程中的两类赋值语句

7.1.3 未指定延时的非阻塞式赋值

目标变量名 <= 驱动表达式;

【例7-3】

```
assign  Q1 = A|B;
assign  Q1 = B&C;
assign  Q1 = ~C;
```

【例7-4】

```
begin
Q1 = A|B;
Q1 = B&C;
Q1 = ~C ;    end
```

【例7-5】

```
begin
Q1 <= A|B;
Q1 <= B&C;
Q1 <= ~C ;    end
```

设进程启动后，A=2'b10，B=2'b01，C=2'b11

【例7-6】

```
always @(A,B) begin
M1 = A ;          //更新结果：M1=1
M2 = B&M1;        //更新结果：M2=1&1=1
Q=M1|M2; end      //更新结果：M2=1|1=1
```

【例7-7】

```
always @(A,B) begin
M1 <= A ;         //更新结果：M1=1
M2 <= B&M1;       //更新结果：M2=1&0=0
Q<=M1|M2; end //更新结果：Q=0|0=0
```

7.1 过程中的两类赋值语句

7.1.4 指定了延时的非阻塞式赋值

[延时] 目标变量名 <= 驱动表达式;

目标变量名 <= [延时] 驱动表达式;

【例 7-8】	【例 7-9】	【例 7-10】
begin Y1 = #6 A^B; Y2 = #4 A|B; Y3 = #7 A&B; end	begin Y1 <= #6 A^B; Y2 <= #4 A|B; Y3 <= #7 A&B; end	begin Y1 = #5 A^B; Y2 <= #3 A|B; Y3 <= #2 A&B; Y4 = #4 (~B); end

7.1 过程中的两类赋值语句

7.1.5 深入认识阻塞与非阻塞式赋值的特点

【例7-11】使用非阻塞赋值符的时序模块

```
module DDF3(CLK,D,Q);
input CLK,D;  output Q; reg a,b,Q;
  always @(posedge CLK) begin
            a <= D;
            b <= a;
            Q <= b;     end
endmodule
```

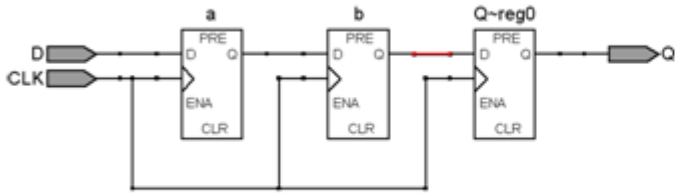


图7-1 例7-11综合后的RTL电路

【例7-12】使用阻塞赋值符的时序模块

```
module DFF3(CLK,D,Q);
input CLK,D; output Q;reg a,b,Q;
  always @(posedge CLK) begin
          a = D;
          b = a;
          Q = b;       end
endmodule
```

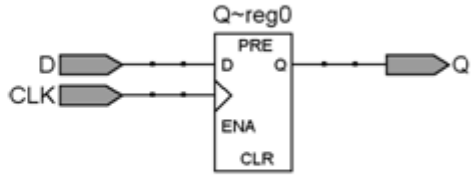


图7-2 例7-12综合后的RTL电路

```
begin Q = b;      b = a;     a = D;  end
```

7.1 过程中的两类赋值语句

7.1.6 不同的赋初值方式导致不同综合结果的示例

【例7-13】

```
module MUX41a(D,S,DOUT);
  output DOUT ;
  input [3:0] D; input [1:0] S;
  integer T;  reg DOUT;
  always @(D,S)
  begin   T <= 0;
    if (S[0]==1)   T<=T+1;
    if (S[1]==1)   T<=T+2;
     case (T)
        0 : DOUT = D[0] ;
        1 : DOUT = D[1] ;
        2 : DOUT = D[2] ;
        3 : DOUT = D[3] ;
     default : DOUT = D[0] ;
     endcase    end
endmodule
```

【例7-14】

```
module MUX41a(D,S,DOUT);
  output DOUT ;
  input [3:0] D; input [1:0] S;
  integer T; reg DOUT;
  always @(D,S)
  begin   T = 0;
    if (S[0]==1)   T=T+1;
    if (S[1]==1)   T=T+2;
     case (T)
        0 : DOUT = D[0] ;
        1 : DOUT = D[1] ;
        2 : DOUT = D[2] ;
        3 : DOUT = D[3] ;
     default : DOUT = D[0] ;
     endcase    end
endmodule
```

7.1 过程中的两类赋值语句

7.1.6 不同的赋初值方式导致不同综合结果的示例

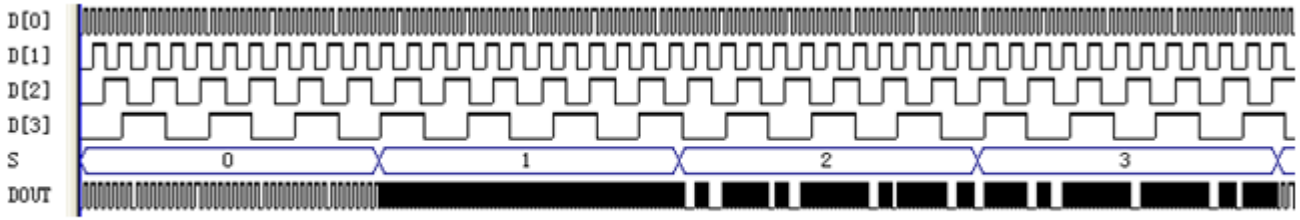


图 7-3 例 7-13 的错误工作时序

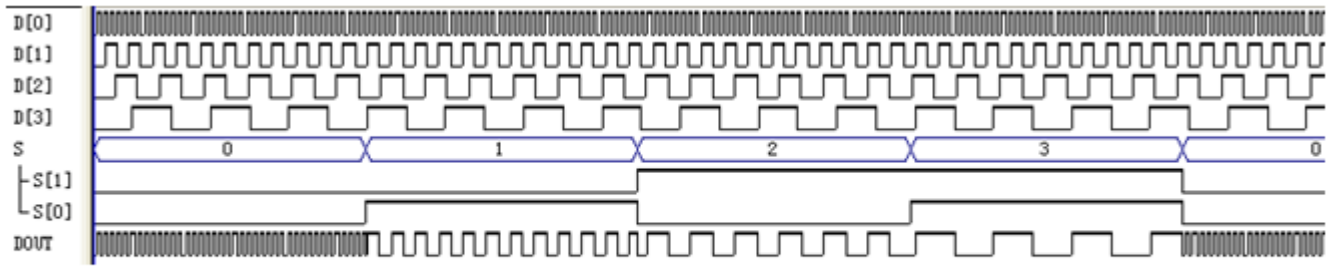


图 7-4 例 7-14 的正确工作时序

7.2 过程语句深入探讨

7.2.1 过程语句应用总结

```
assign DOUT = a & b;
```

7.2.2 深入认识不完整条件语句与时序电路的关系

【例7-15】

```
module mux2_1
  (CLK, D, Q, RST);
   output Q;
   input CLK,D,RST;
   reg Q;
 always @(D, CLK, RST)
   if(CLK)  Q <= D;
      else  Q <= RST;
endmodule
```

【例7-16】

```
module COMP(A,B,Q);
input[3:0] A,B;
output Q;  reg Q;
always @(A,B)
 begin
  if(A>B) Q=1'b1;
   else
  if(A<B)  Q=1'b0;
end   endmodule
```

【例7-17】

```
module COMP(A,B,Q);
input[3:0] A,B;
output Q;   reg Q;
always @(A,B)
 begin
   if(A>B) Q=1'b1;
else  if(A<B) Q=1'b0;
   else    Q=1'bz;
end    endmodule
```

7.2 过程语句深入探讨

7.2.2 深入认识不完整条件语句与时序电路的关系

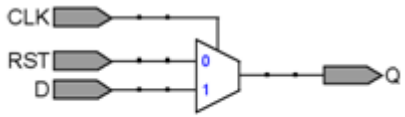


图 7-5 例 7-15 的 RTL 图

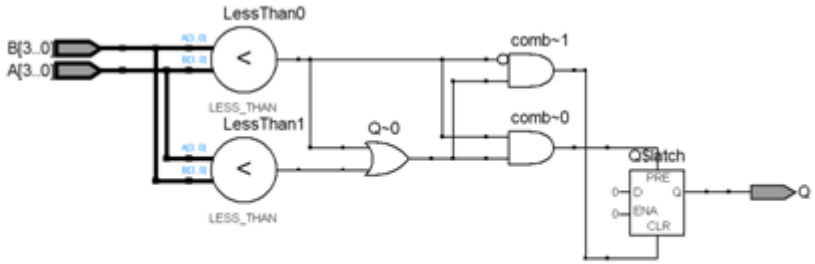


图 7-6 例 7-16 的 RTL 图，输出口被加上了锁存器

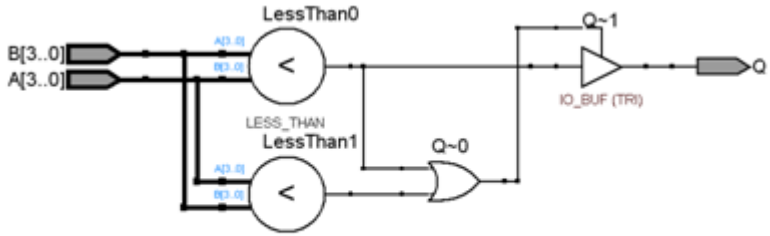


图 7-7 例 7-17 的电路，是纯组合电路

7.3 三态与双向端口设计

7.3.1 三态控制电路设计

【例7-18】

```
module tri4B (ENA,DIN,DOUT);
   input ENA;  input [3:0] DIN ;
   output [3:0] DOUT ;
   reg [3:0] DOUT;
   always @(DIN,ENA)
     if (ENA)  DOUT <= DIN ;
        else   DOUT <= 4'HZ;
endmodule
```

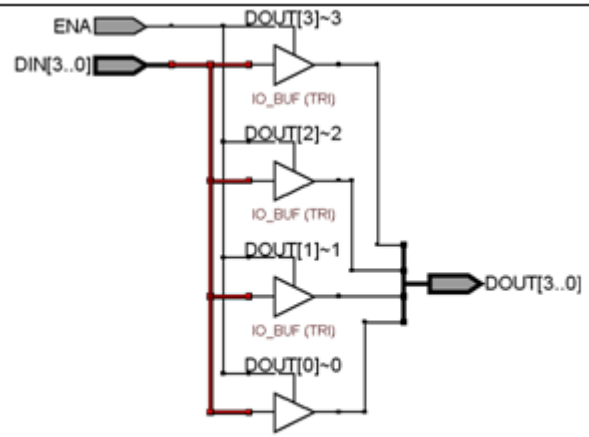


图7-8　4位三态控制门电路

7.3 三态与双向端口设计

7.3.2 双向端口设计

【例 7-19】

```
module bi4b(TRI_PORT,DOUT,DIN,ENA,CTRL);
   inout TRI_PORT;   input DIN,ENA,CTRL;   output DOUT ;
   assign TRI_PORT = ENA ? DIN : 1'bz;
   assign DOUT = TRI_PORT | CTRL;
endmodule
```

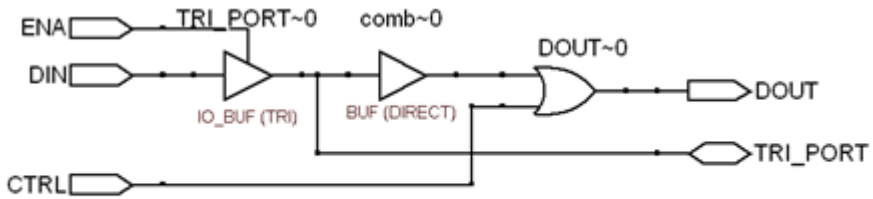


图 7-9 例 7-19 的 1 位双向端口电路设计之 RTL 图

7.3 三态与双向端口设计

7.3.2 双向端口设计

【例7-20】

```
module BI4B(CTRL,DIN,Q,DOUT);
  input CTRL;     input[3:0] DIN;
  inout[3:0]Q; output[3:0] DOUT;
  reg [3:0] DOUT,Q ;
  always @(Q,DIN,CTRL)
     if (!CTRL)  DOUT<=Q ;
       else
  begin Q<=DIN; DOUT<=4'HZ;  end
    endmodule
```

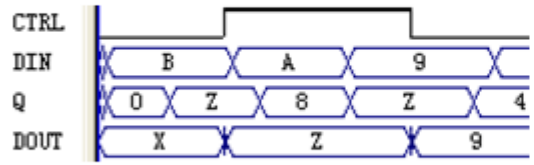


图7-10 本例的仿真波形图

【例7-21】

```
module BI4B(CTRL,DIN,Q,DOUT);
  input CTRL;     input[3:0] DIN;
  inout[3:0] Q; output[3:0] DOUT;
  reg [3:0] DOUT,Q ;
 always @(Q,DIN,CTRL)
 if (!CTRL)  begin DOUT<=Q;
       Q<=4'HZ; end  else
 begin Q<=DIN;  DOUT<=4'HZ;  end
 endmodule
```

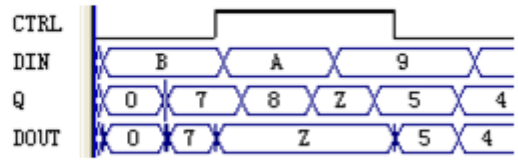


图7-11 本例的仿真波形图

7.3 三态与双向端口设计

7.3.2 双向端口设计

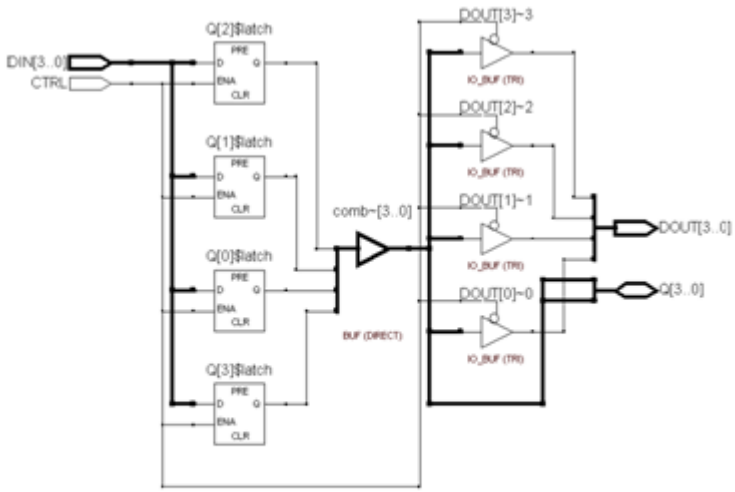


图 7-12 例 7-20 的 RTL 图

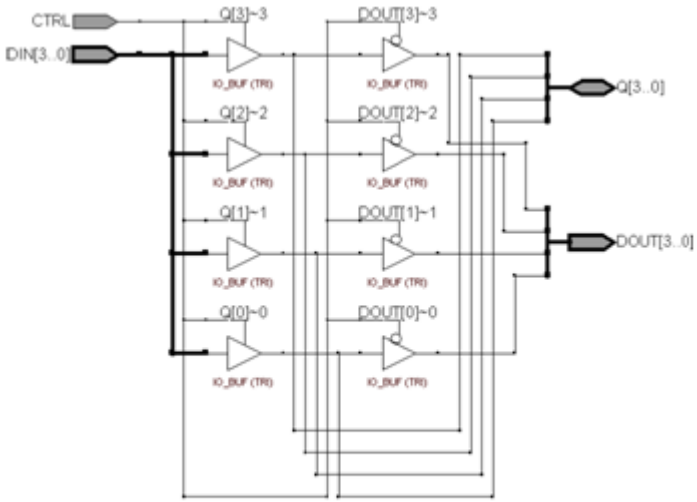


图 7-13 例 7-21 的 RLT 图

7.3 三态与双向端口设计

7.3.3 三态总线控制电路设计

【例7-22】

```
module triBUS4(
  IN3,IN2,IN1,IN0,ENA,DOUT);
  input[3:0] IN3,IN2,IN1,IN0 ;
  input[1:0]  ENA;
  output[3:0] DOUT;
  reg[3:0] DOUT;
  always @(ENA,IN3,IN2,IN1,IN0)
    begin
   if (ENA==0)  DOUT=IN3 ;
       else  DOUT=4'HZ;
   if (ENA==1)  DOUT=IN2 ;
       else  DOUT=4'HZ;
  if (ENA==2)  DOUT=IN1 ;
       else  DOUT=4'HZ;
  if (ENA==3)  DOUT=IN0 ;
       else  DOUT=4'HZ;
    end
endmodule
```

【例7-23】

```
module triBUS4(
    IN3,IN2,IN1,IN0,ENA,DOUT);
  input[3:0] IN3,IN2,IN1,IN0 ;
  input[1:0]  ENA;
  output[3:0] DOUT; reg[3:0]DOUT;
  always @(ENA, IN0)
  if (ENA==2'b00)  DOUT=IN0;
     else  DOUT=4'hz;
  always @(ENA, IN1)
  if (ENA==2'b01)  DOUT=IN1;
     else  DOUT=4'hz;
  always @(ENA, IN2)
  if (ENA==2'b10)  DOUT=IN2;
     else  DOUT=4'hz;
  always @(ENA, IN3)
  if (ENA==2'b11)  DOUT=IN3;
     else  DOUT=4'hz;

endmodule
```

7.3 三态与双向端口设计

7.3.3 三态总线控制电路设计

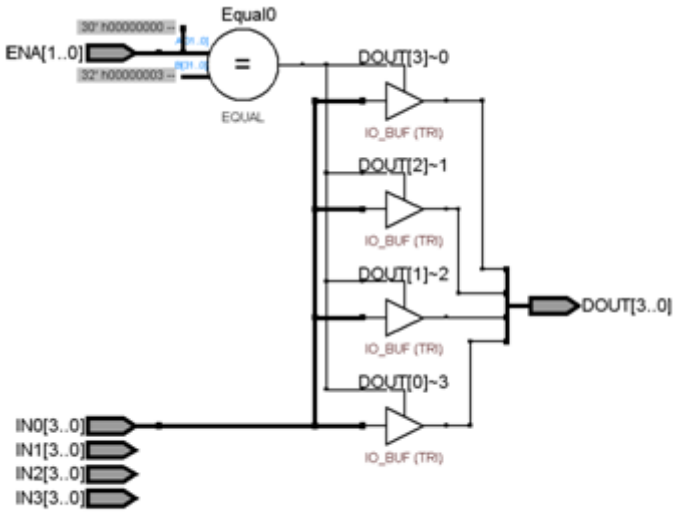


图7-14 例7-22的RTL图

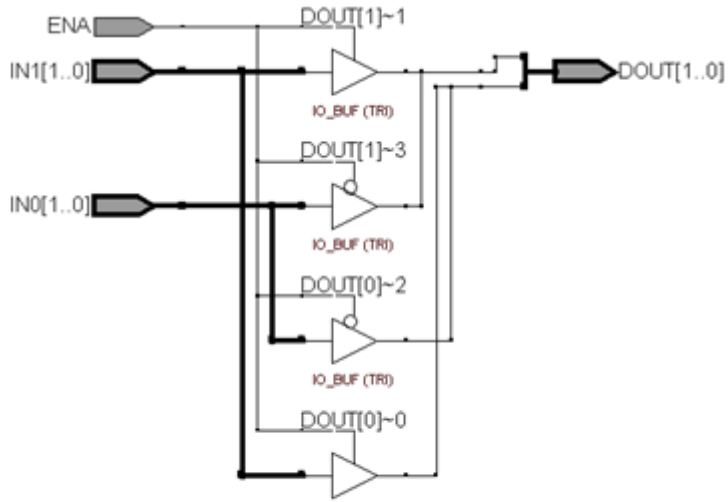


图7-15 例7-23的2位简化RTL图

7.4 资源优化

7.4.1 资源共享

【例 7-24】

```
module multmux (A0, A1, B, S, R);
  input[3:0] A0, A1, B;  input S;
  output[7:0] R;   reg[7:0] R;
  always @(A0 or A1 or B or S)
    begin
    if (S==1'b0)  R<=A0 * B;
      else  R<=A1 * B ;   end
endmodule
```

【例 7-25】

```
module multmux (A0, A1, B, S,R);
 input[3:0] A0, A1, B;  input S;
 output[7:0] R;   wire [7:0] R;
   reg [3:0] TEMP;
  always @(A0 or A1 or B or S)
  begin  if (S==1'b0)  TEMP<=A0;
   else  TEMP <= A1;   end
  assign R=TEMP * B;
endmodule
```

7.4 资源优化

7.4.1 资源共享

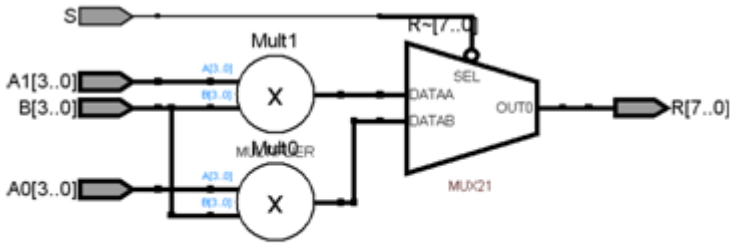


图 7-16 先乘后选择的设计方法 RTL 结构

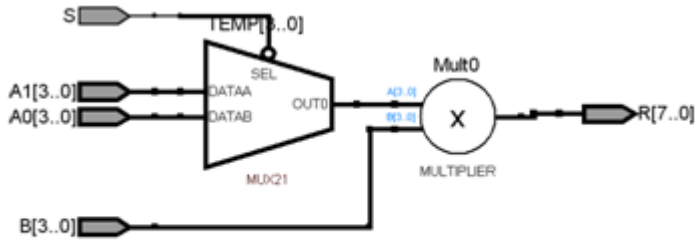


图 7-17 先选择后乘设计方法 RTL 结构

7.4 资源优化

7.4.2 逻辑优化

【例 7-26】

```
module mult1 (clk, ma, mc);
 input clk;  input[11:0] ma;
 output[23:0] mc;
 reg[23:0] mc; reg[11:0] ta,tb;
 always @(posedge clk)
 begin  ta<=ma; mc<=ta * tb;
   tb <= 12'b100110111001; end
endmodule
```

【例 7-27】

```
module mult2  (clk, ma, mc);
 input clk;  input[11:0] ma;
 output[23:0] mc;
 reg[23:0] mc;   reg[11:0] ta;
 parameter tb=12'b100110111001;
 always @(posedge clk)
 begin  ta<=ma ; mc<=ta * tb; end
endmodule
```

7.4 资源优化

7.4.3 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 7-28】

```
module pmultadd  (clk, a0, a1, a2, a3, b0, b1, b2, b3, yout);
   input clk;   input[7:0] a0, a1, a2, a3, b0, b1, b2, b3;
   output[15:0] yout;    reg[15:0] yout;
   always @(posedge clk)   begin
      yout <= ((a0 * b0)+(a1 * b1))+((a2 * b2)+(a3 * b3)) ;   end
endmodule
```

7.4 资源优化

7.4.3 串行化

【例 7-29】

```
module smultadd  (clk, start, a0, a1, a2, a3, b0, b1, b2, b3, yout);
    input clk, start;  input[7:0] a0, a1, a2, a3, b0, b1, b2, b3;
    output[15:0] yout;    reg[15:0] yout, ytmp;    reg[2:0] cnt;
    wire[7:0] tmpa, tmpb; wire[15:0] tmp;
 assign tmpa=(cnt==0)? a0:(cnt==1)? a1:(cnt==2)? a2:(cnt==3)? a3:a0;
 assign tmpb=(cnt==0)? b0:(cnt==1)? b1:(cnt==2)? b2:(cnt==3)? b3:b0;
  assign tmp = tmpa * tmpb ;
    always @(posedge clk)  begin
      if (start==1'b1)  begin  cnt<=3'b000 ; ytmp<={16{1'b0}} ;  end
       else if (cnt<4)    begin  cnt<=cnt+1  ; ytmp<=ytmp+tmp ;  end
       else if (cnt==4)   begin  yout<=ytmp ;   end    end
endmodule
```

7.5 速度优化

7.5.1 流水线设计

图 7-18 未使用流水线

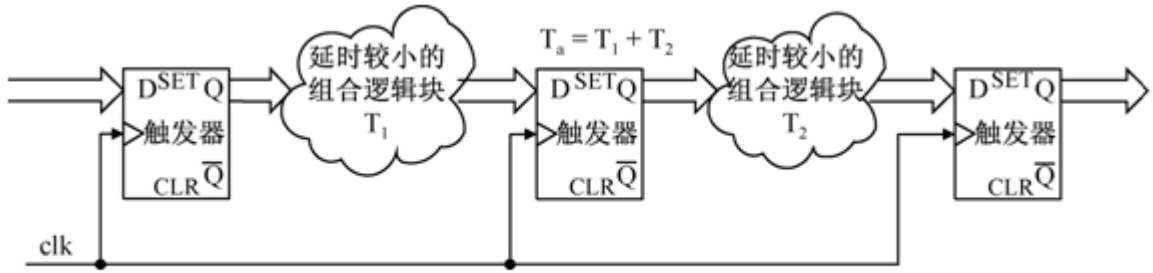


图 7-19 使用流水线结构

7.5 速度优化

7.5.1 流水线设计

图 7-20 流水线工作图示

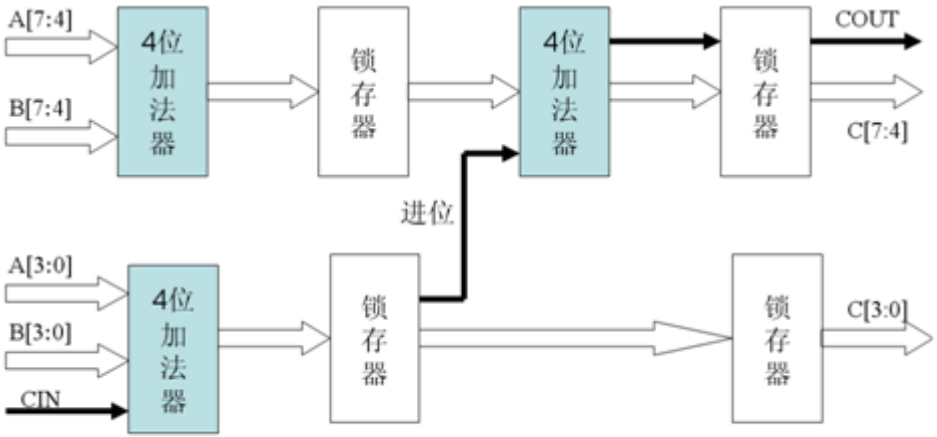


图 7-21 8 位加法器流水线工作图示

7.5 速 度 优 化

7.5.1 流水线设计

【例 7-30】普通加法器，EP3C5 综合结果：LCs=10,REG=0,T=7.748ns.

```
module ADDER8(CLK,SUM,A,B,COUT,CIN);
   input [7:0] A,B;  input CLK,CIN;  output COUT;  output [7:0] SUM;
   reg COUT;   reg [7:0] SUM;
    always @(posedge CLK)    {COUT,SUM[7:0]} <= A+B+CIN;
  endmodule
```

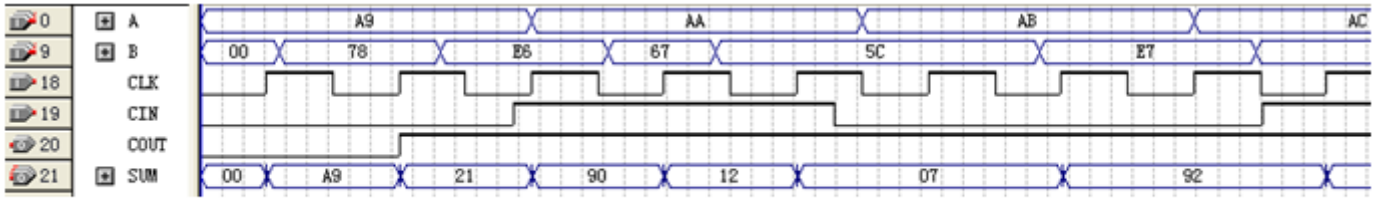


图 7-22 例 7-30 的时序仿真波形

7.5 速度优化

7.5.1 流水线设计

【例 7-31】流水线加法器，EP3C5 综合结果：T=3.63ns，LCs=24，REG=22。

```
module ADDER8(CLK,SUM,A,B,COUT,CIN);
   input [7:0] A,B;  input CLK,CIN;  output COUT;   output[7:0] SUM;
    reg TC,COUT;   reg[3:0] TS,TA,TB;    reg[7:0] SUM;
     always @(posedge CLK)  begin
        {TC,TS} <= A[3:0]+B[3:0]+CIN ;   SUM[3:0]<=TS;    end
     always @(posedge CLK)  begin
        TA <= A[7:4];   TB <= B[7:4];
      {COUT,SUM[7:4]} <= TA+TB+TC;     end
endmodule
```

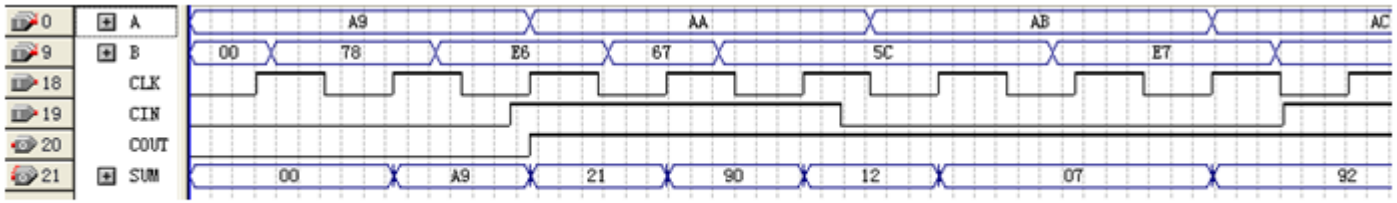


图 7-23 例 7-31 的时序仿真波形

7.5 速 度 优 化

7.5.2 关键路径法

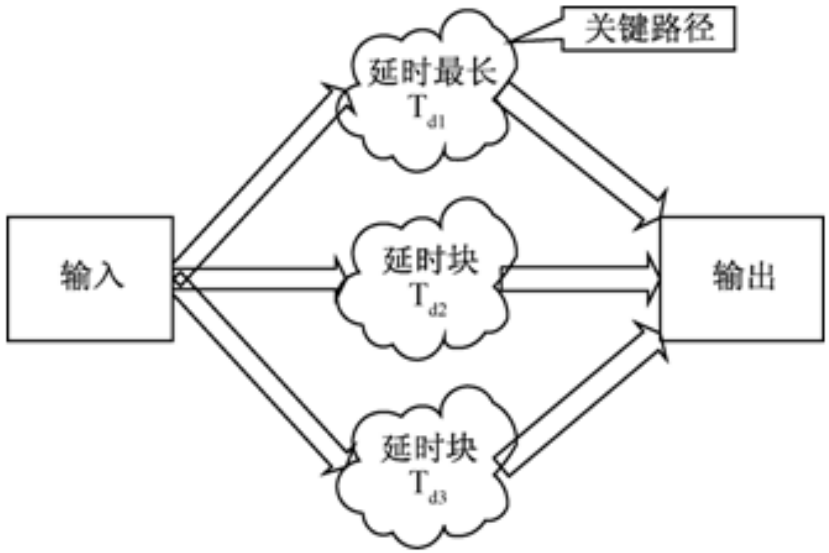


图 7-24 关键路径示意

7.5 速度优化

7.5.3 乒乓操作法

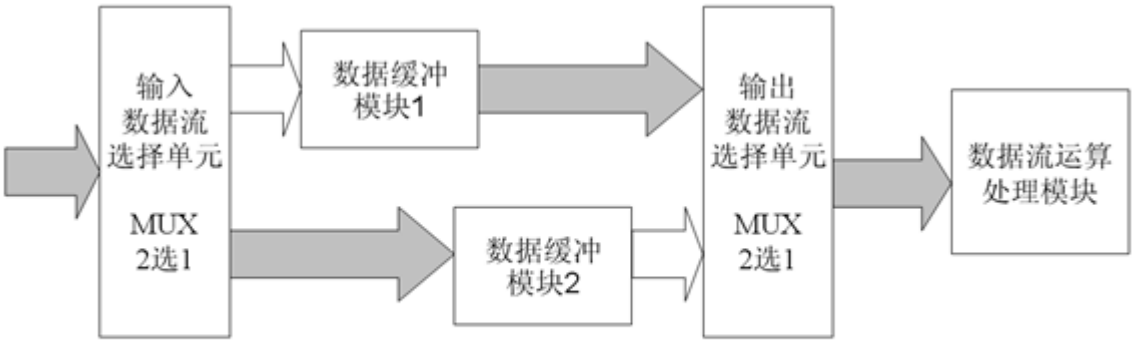


图 7-25　乒乓操作数据缓存结构示意图

7.5.4 加法树法

习　题

示例 1	示例 2
module test1 (X1,X2,A,B,C,D,CLK) ; input CLK,X1,X2; output A,B,C,D; reg A,B,C,D; always@(posedge CLK) begin A=X1; D=X2; B<=D; C<=A; end endmodule	module test1 (X1,X2,A,B,C,D,CLK) ; input CLK,X1,X2; output A,B,C,D; reg A,B,C,D; always@(posedge CLK) begin B<=D; C<=A; A=X1; D=X2; end endmodule

```
module addmux (A, B, C, D, sel, Result);
   input[7:0] A, B, C, D;   input sel;
   output[7:0] Result;    reg[7:0] Result;
   always @(A or B or C or D or sel)  begin
      if (sel==1'b0)  Result <= A+B;  else  Result <= C+D ; end
endmodule
```

习　题

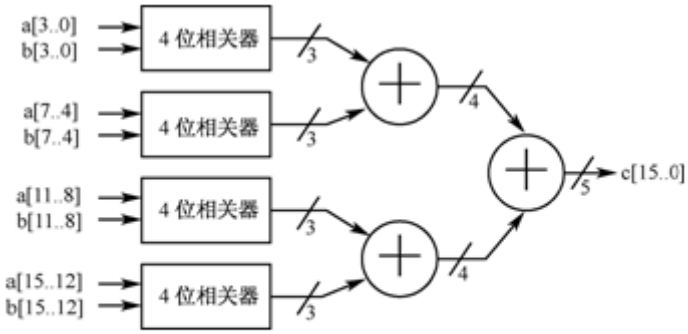
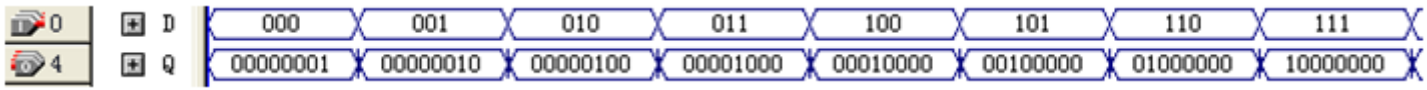


图 7-26 习题 7-16 图

```
module DCD3_8  (output reg [7:0]Q, input[2:0]D);
  always @(D )
    begin  Q<= 8'b00000000;  Q[D]<=1;  end
endmodule
```



实验与设计

7-1 4X4阵列键盘键信号检测电路设计

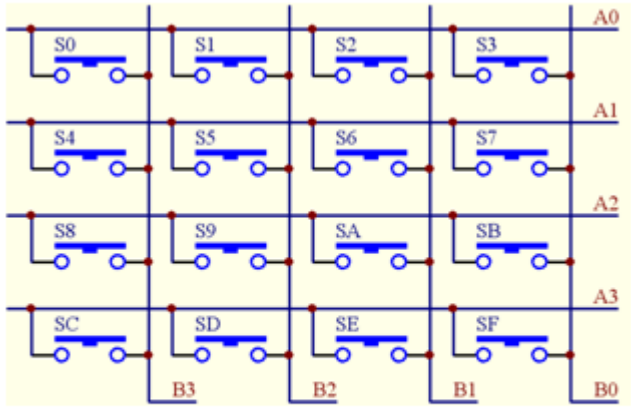


图 7-27　4X4 键盘电路

实验与设计

7-1 4X4阵列键盘键信号检测电路设计

【例 7-32】

```
module KEY4X4 (input CLK, input [3:0]A, output reg[3:0]B,R);
    reg [1:0] C;
   always @(posedge CLK)    begin     C<=C+1;
     case(C)
     0: B=4'B0111; 1: B=4'B1011; 2: B=4'B1101; 3: B=4'B1110;
     endcase
     case({B,A} )
     8'B0111_1110: R=4'H0; 8'B0111_1101 : R=4'H1;
     8'B0111_1011: R=4'H2; 8'B0111_0111 : R=4'H3;
     8'B1011_1110: R=4'H4; 8'B1011_1101 : R=4'H5;
     8'B1011_1011: R=4'H6; 8'B1011_0111 : R=4'H7;
     8'B1101_1110: R=4'H8; 8'B1101_1101 : R=4'H9;
     8'B1101_1011: R=4'HA; 8'B1101_0111 : R=4'HB;
     8'B1110_1110: R=4'HC; 8'B1110_1101 : R=4'HD;
     8'B1110_1011: R=4'HE; 8'B1110_0111 : R=4'HF;
     endcase    end
endmodule
```

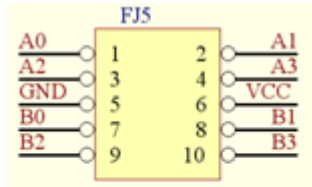


图 7-28 4X4 键盘的 10 芯接口

实验与设计

7-2 直流电机综合测控系统设计

【例 7-33】

```
module SQU (input[7:0] CIN, input[7:0] ADR, output reg OT) ;
  always @(CIN)  if (ADR<CIN)  OT<=1'b0;  else  OT<=1'b1 ;
endmodule
```

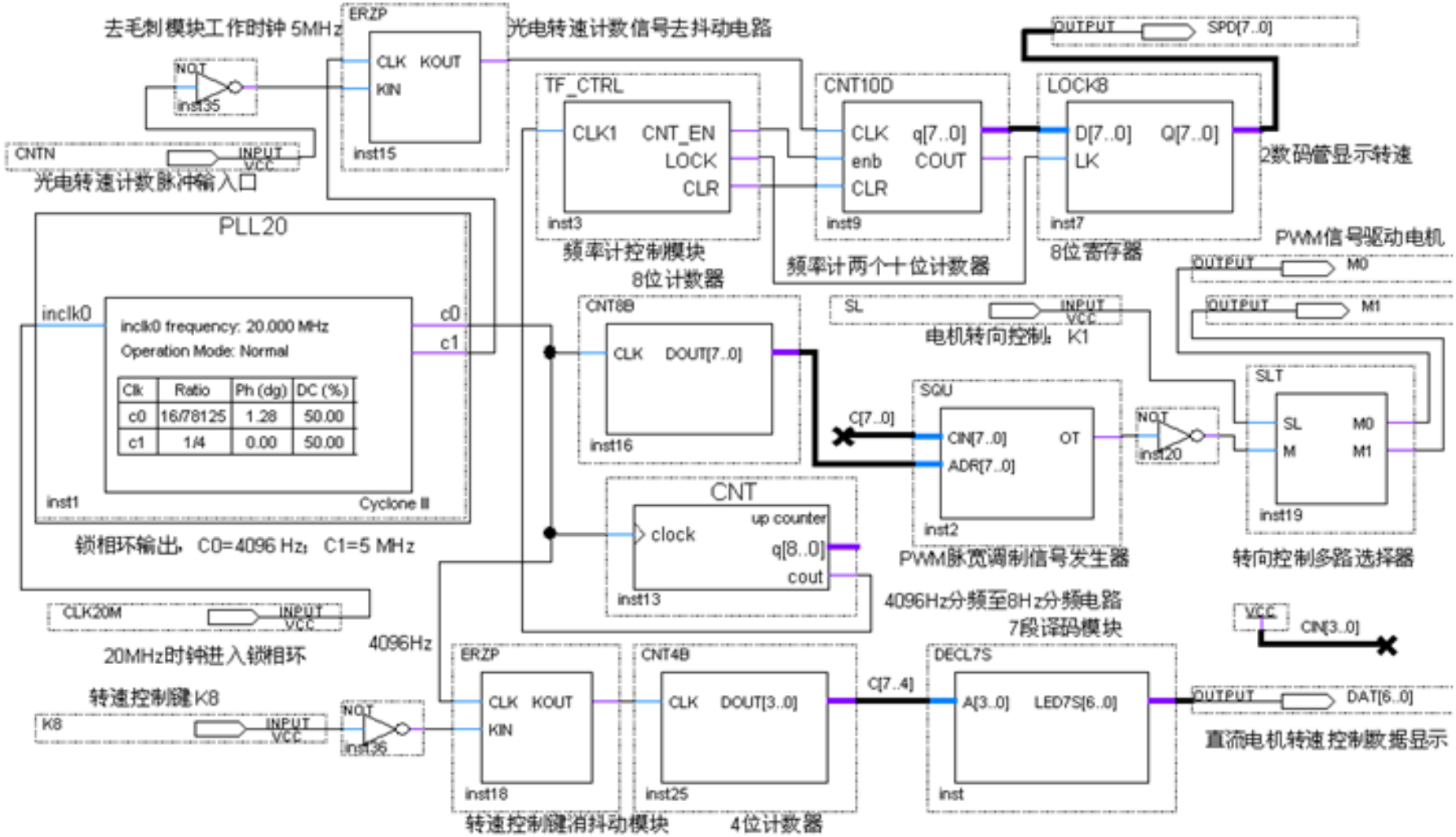


图 7-29 直流电机驱动控制电路顶层设计

实验与设计

7-3 VGA简单图像显示控制模块设计

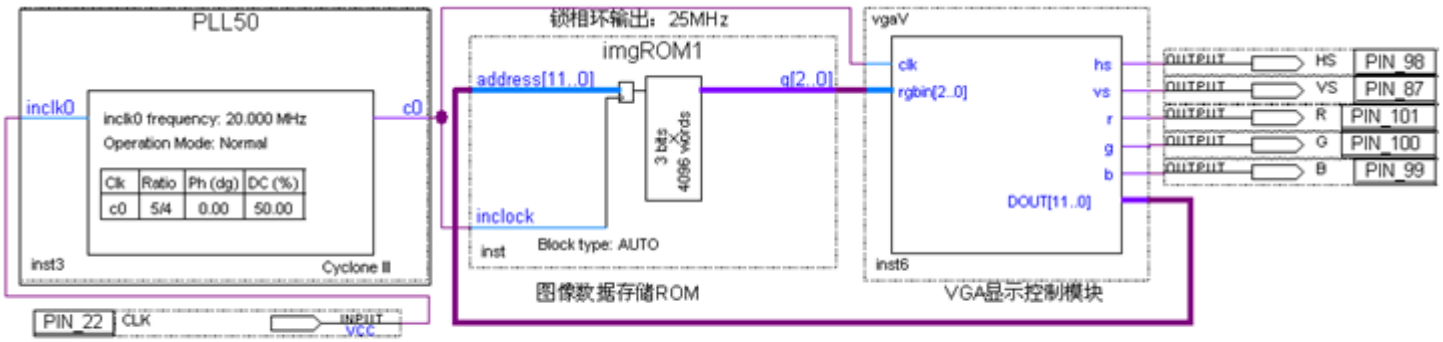


图 7-30 VGA 图像显示控制模块原理图

【例 7-34】

```
module vgaV (clk, hs, vs, r, g, b, rgbin, DOUT);
   input clk ;     //工作时钟 25MHz
    output hs,vs;  output r,g,b;  //场同步，行同步信号，以及红、绿、蓝控制信号
   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
```

实验与设计

7-4 乐曲硬件演奏电路设计

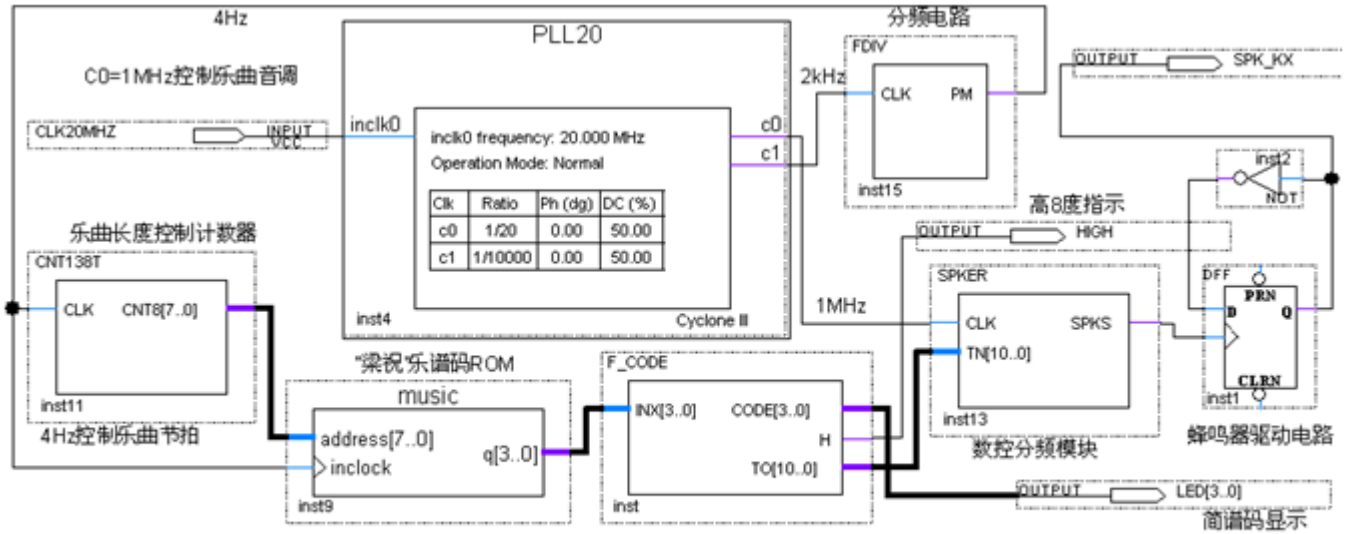


图 7-31 乐曲演奏电路顶层设计

实验与设计

7-4 乐曲硬件演奏电路设计

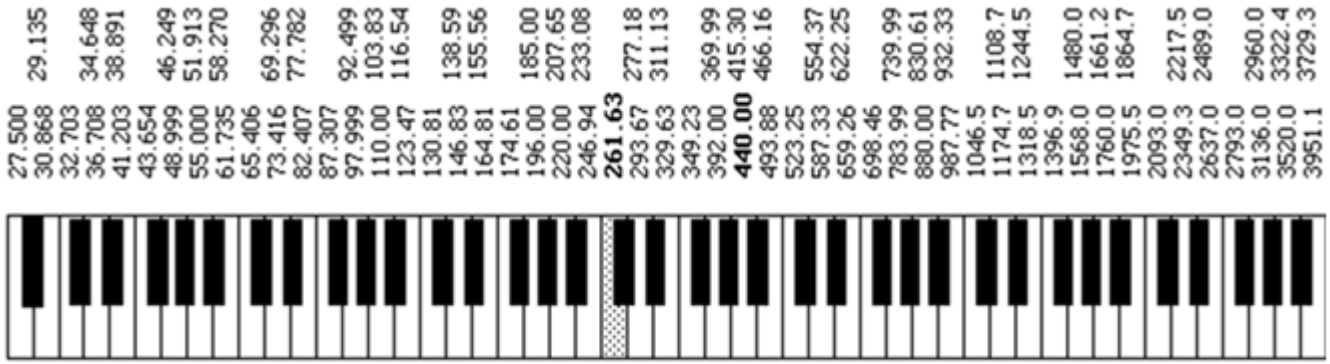


图 7-32 电子琴音阶基频对照图（单位 Hz）

【例 7-35】

```
module CNT138T (CLK, CNT8);
    input CLK;    output[7:0] CNT8 ;   reg[7:0] CNT;   wire LD;
    always @(posedge CLK or posedge LD )   begin
      if (LD) CNT <= 8'b00000000 ;  else    CNT<=CNT+1;   end
    assign CNT8=CNT;   assign LD=(CNT==138) ;
endmodule
```

【例 7-36】

```
module F_CODE (INX, CODE, H, TO);
  input[3:0] INX;  output[3:0] CODE;   output H;  output[10:0] TO;
  reg[10:0] TO;   reg[3:0] CODE;   reg H;
  always @(INX)   begin
  case (INX)          // 译码电路，查表方式，控制音调的预置？
   0 : begin TO <= 11'H7FF; CODE<=0; H<=0; end
   1 : begin TO <= 11'H305; CODE<=1; H<=0; end
   2 : begin TO <= 11'H390; CODE<=2; H<=0; end
   3 : begin TO <= 11'H40C; CODE<=3; H<=0; end
   4 : begin TO <= 11'H45C; CODE<=4; H<=0; end
   5 : begin TO <= 11'H4AD; CODE<=5; H<=0; end
   6 : begin TO <= 11'H50A; CODE<=6; H<=0; end
   7 : begin TO <= 11'H55C; CODE<=7; H<=0; end
   8 : begin TO <= 11'H582; CODE<=1; H<=1; end
   9 : begin TO <= 11'H5C8; CODE<=2; H<=1; end
  10 : begin TO <= 11'H606; CODE<=3; H<=1; end
  11 : begin TO <= 11'H640; CODE<=4; H<=1; end
  12 : begin TO <= 11'H656; CODE<=5; H<=1; end
  13 : begin TO <= 11'H684; CODE<=6; H<=1; end
  14 : begin TO <= 11'H69A; CODE<=7; H<=1; end
  15 : begin TO <= 11'H6C0; CODE<=1; H<=1; end
   default : begin TO <= 11'H6C0; CODE<=1; H<=1;
    end
  endcase    end
endmodule
```

实验与设计

7-4 乐曲硬件演奏电路设计

【例 7-37】

```
module SPKER (CLK, TN, SPKS);
 input CLK;  input[10:0] TN;  output SPKS;
 reg SPKS;  reg[10:0] CNT11;
 always @(posedge CLK)  begin  : CNT11B_LOAD// 11位可预置计数器
     if (CNT11==11'h7FF) begin CNT11=TN;  SPKS<=1'b1;  end
      else begin  CNT11=CNT11+1;  SPKS<=1'b0 ;   end
 end
endmodule
```

【例 7-38】

```
module FDIV  (CLK,PM );
 input CLK ;  output PM ;  reg [8:0] Q1;  reg FULL;  wire RST ;
  always @(posedge CLK or posedge RST) begin
     if (RST) begin Q1<=0; FULL<=1; end
       else  begin Q1 <= Q1+1; FULL<=0 ;  end   end
  assign RST = ( Q1==499 ) ; assign PM = FULL ;
  assign DOUT = Q1 ;
endmodule
```

实验与设计

【例 7-39】

```
  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

7-4 乐曲硬件演奏电路设计

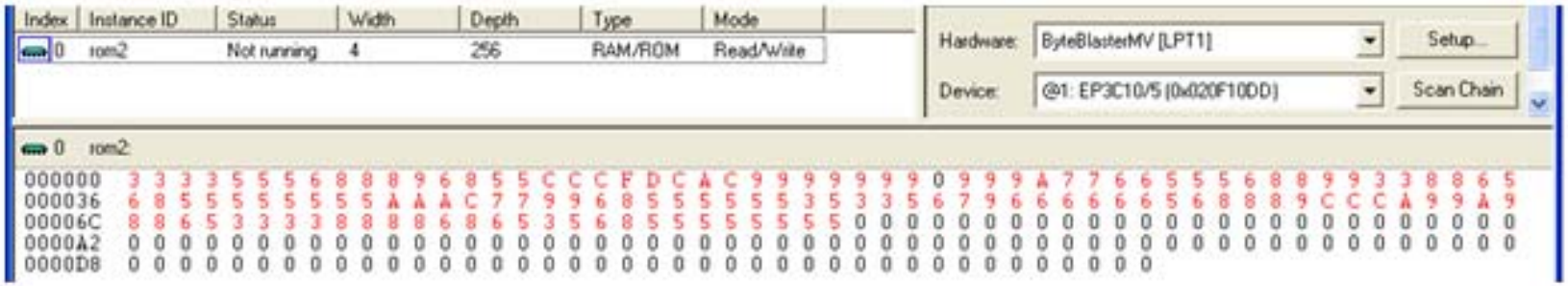


图 7-33 In-System Memory Content Editor 对 MUSIC 模块的数据读取

实验与设计

7-5 PS2键盘控制模型电子琴电路设计

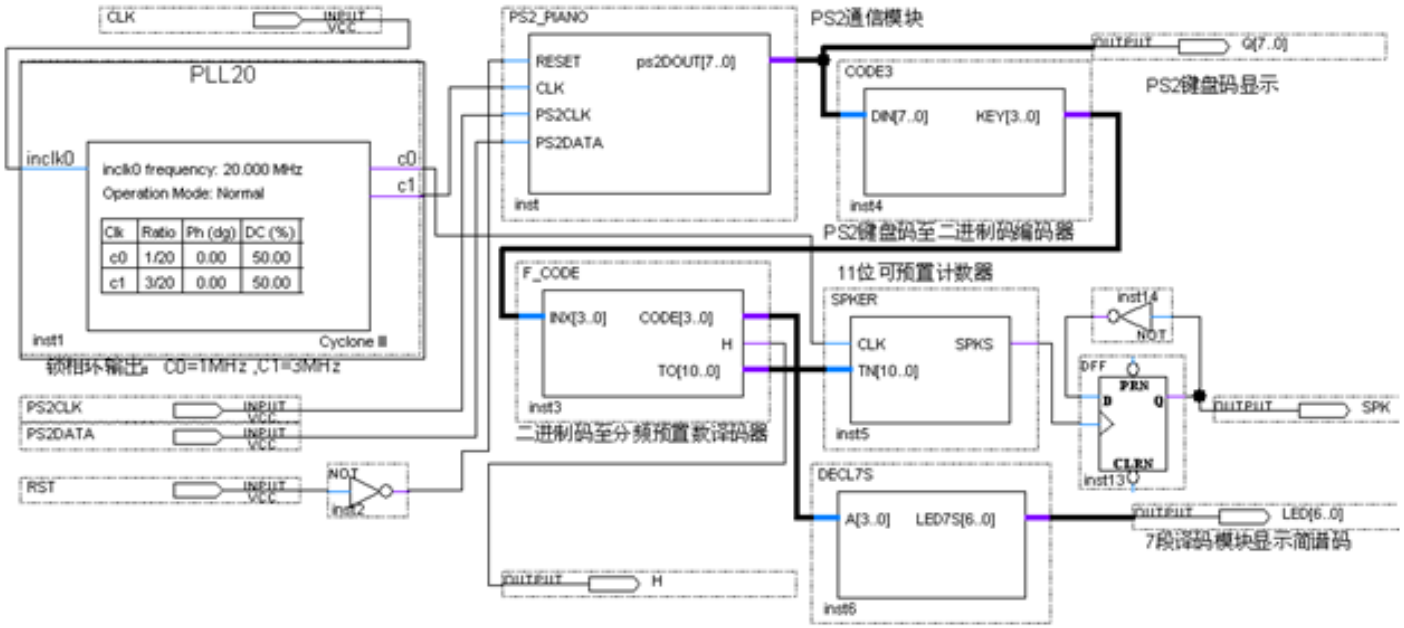


图 7-34　PS2 键盘控制模型电子琴电路顶层设计

【例 7-40】

```
module PS2_PIANO(clk,kb_clk,kb_data,keycode,keydown,keyup,dataerror);
   input clk, kb_clk, kb_data;  output keydown, keyup, dataerror;
    output[7:0] keycode;
    reg[7:0] keycode, shiftdata;  reg keydown, keyup, dataerror;
    wire[7:0] kbcodereg;  reg[3:0] cnt;
    reg datacoming, kbclkfall, kbclkreg, parity, isfo;
    always @(posedge clk)   begin
      kbclkreg <= kb_clk ;
      kbclkfall <= kbclkreg & (~kb_clk) ;    end
    always @(posedge clk)  begin
      if (kbclkfall == 1'b1 & datacoming == 1'b0 & kb_data == 1'b0)
      begin
        datacoming<=1'b1; cnt<=4'b0000;  parity<=1'b0;   end
      else if (kbclkfall == 1'b1 & datacoming == 1'b1)
      begin  if (cnt == 9)
        begin
          if (kb_data == 1'b1)
           begin   datacoming<=1'b0;  dataerror<=1'b0; end
          else  begin  dataerror<=1'b1;  end
          cnt <= cnt + 1 ;  end
        else if (cnt == 8)  begin if (kb_data == parity)
          begin  dataerror <= 1'b0 ;  end
          else  begin  dataerror<=1'b1;  end
```

接下页

实验与设计

7-5 PS2键盘控制模型电子琴电路设计

```
          cnt <= cnt + 1 ;    end
        else    begin   shiftdata <= {kb_data, shiftdata[7:1]} ;
          parity <= parity ^ kb_data;  cnt <= cnt + 1 ;   end
    end     end
  always @(posedge clk)   begin
    if (cnt == 10)    begin   if (shiftdata==8'b11110000)
      begin   isfo<=1'b1 ;    end
      else if (shiftdata!=8'b11100000)    begin   if (isfo==1'b1)
        begin   keyup<=1'b1;   keycode<=shiftdata;    end
        else  begin   keydown<=1'b1;   keycode<=shiftdata;   end
      end       end
    else   begin   keyup<=1'b0;   keydown<=1'b0;   end    end
endmodule
```

实验与设计

7-5 PS2键盘控制模型电子琴电路设计

【例 7-41】

```
module CODE3 (input[7:0] DIN,  output reg [3:0] KEY ) ;
  always @(DIN)   begin
    case (DIN)
     8'b00010110 : KEY<=4'b0001;       8'b00011110 : KEY<=4'b0010 ;
     8'b00100110 : KEY<=4'b0011;       8'b00100101 : KEY<=4'b0100 ;
     8'b00101110 : KEY<=4'b0101;       8'b00110110 : KEY<=4'b0110 ;
     8'b00111101 : KEY<=4'b0111;       8'b00111110 : KEY<=4'b1000 ;
     8'b01000101 : KEY<=4'b1001;
    default : KEY<=4'b0000 ;
    endcase  end
endmodule
```

实验与设计

7-5 PS2键盘控制模型电子琴电路设计

表 7-1 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	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

实验与设计

7-6 SPWM脉宽调制控制系统设计

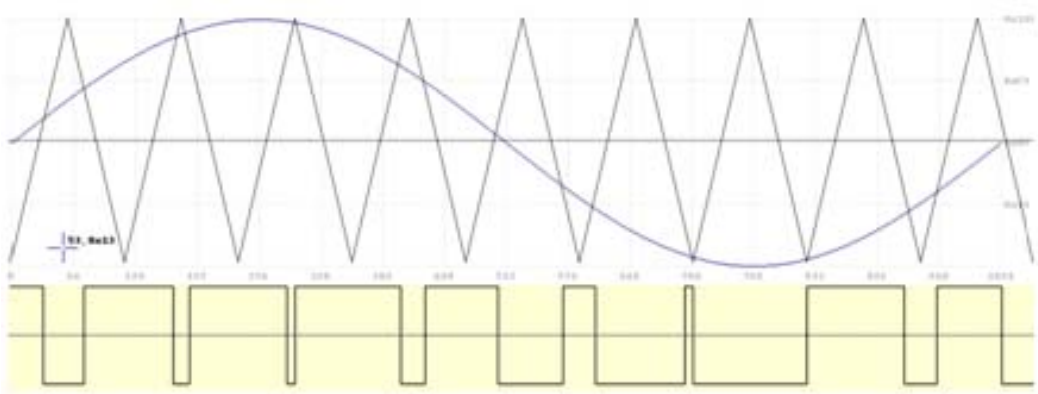


图 7-35　SPWM 波生成原理图

实验与设计

7-6 SPWM脉宽调制控制系统设计

【例 7-42】

```
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

7-6 SPWM脉宽调制控制系统设计

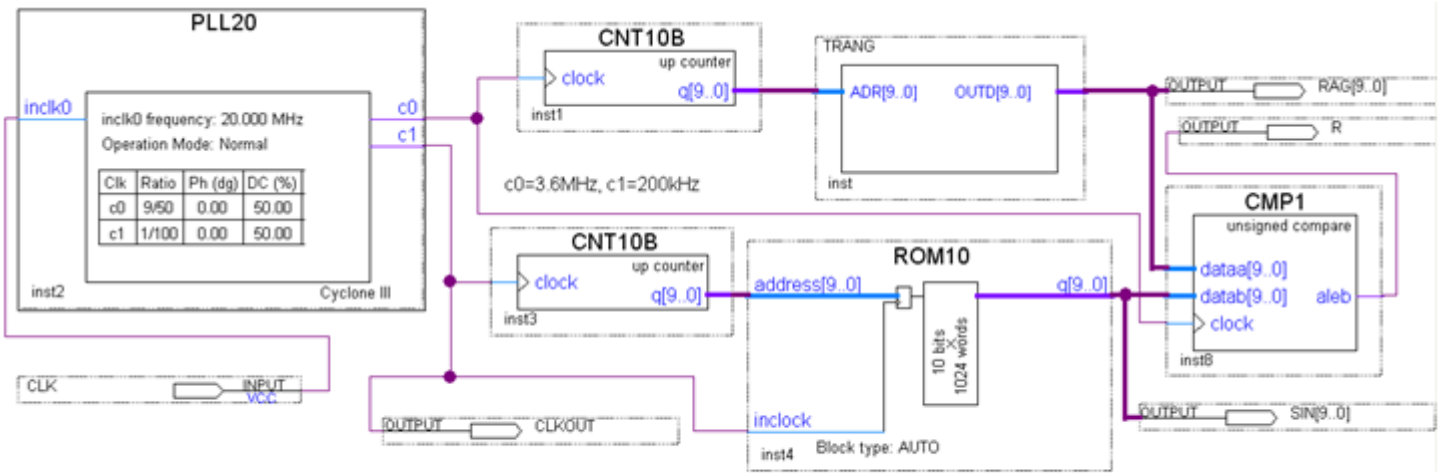


图 7-36　SPWM 波发生器基本电路图

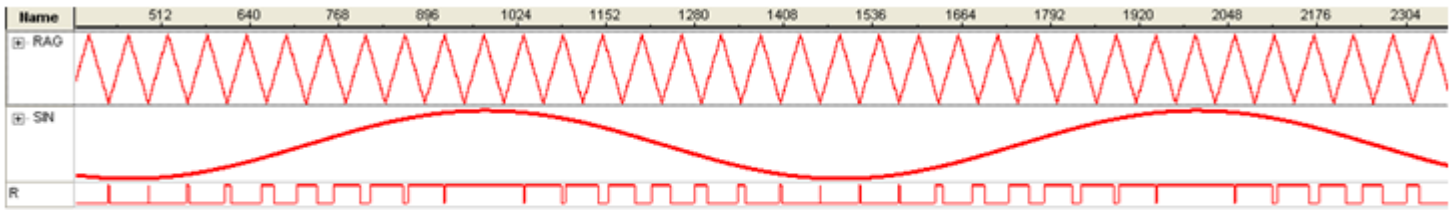


图 7-37　图 7-36 电路的 SignalTap II 实测波形

实验与设计

7-6 SPWM脉宽调制控制系统设计

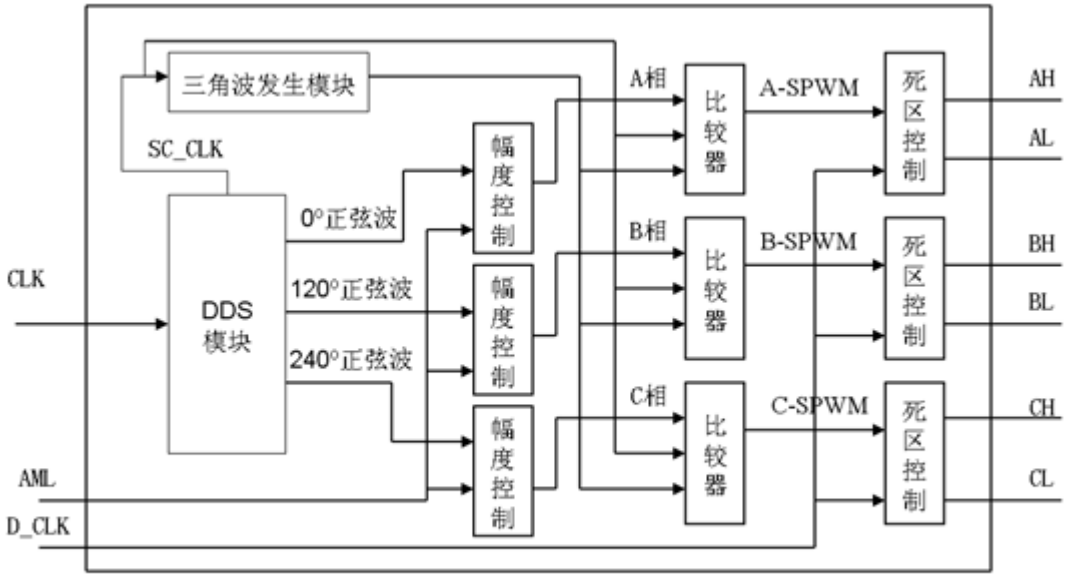


图 7-38 三相 SPWM 控制器电路模块图