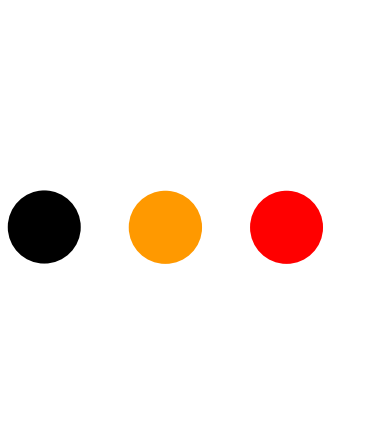


第8章

Verilog设计深入

8.1 过程中的两类赋值语句

8.1.1 未指定延时的阻塞式赋值语句

目标变量名 = 驱动表达式;

8.1.2 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 8-1】	【例 8-2】

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

8.1 过程中的两类赋值语句

8.1.3 未指定延时的非阻塞式赋值

目标变量名 <= 驱动表达式;

【例8-3】

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

【例8-4】

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

【例8-5】

```
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8-6】

```
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8-7】

```
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
```

8.1 过程中的两类赋值语句

8.1.4 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 8-8】	【例 8-9】	【例 8-10】
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

8.1 过程中的两类赋值语句

8.1.5 深入认识阻塞与非阻塞式赋值的特点

【例8-11】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
```

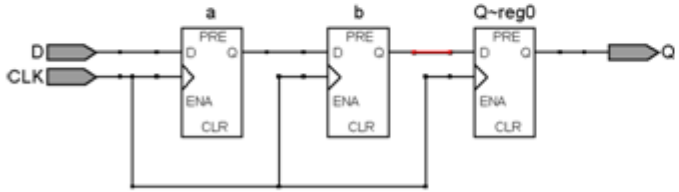


图8-1 例8-11综合后的RTL电路

【例8-12】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
```

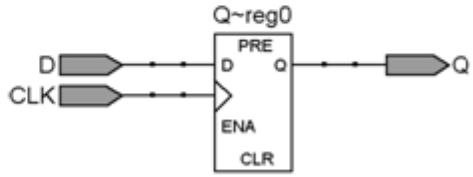


图8-2 例8-12综合后的RTL电路

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

8.1 过程中的两类赋值语句

8.1.6 不同的赋初值方式导致不同综合结果的示例

【例8-13】

```
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8-14】

```
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
```

8.1 过程中的两类赋值语句

8.1.6 不同的赋初值方式导致不同综合结果的示例

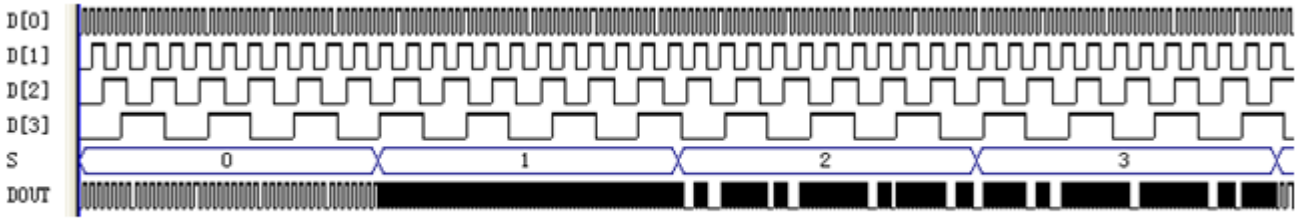


图 8-3 例 8-13 的错误工作时序

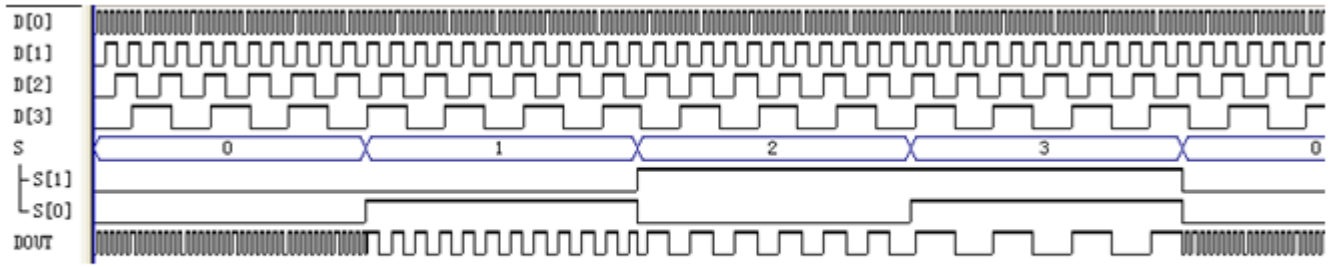


图 8-4 例 8-14 的正确工作时序

8.2 过程语句归纳

8.2.1 过程语句应用总结

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

8.2 过程语句归纳

8.2.2 深入认识不完整条件语句与时序电路的关系

【例8-15】

```
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8-16】

```
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8-17】

```
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
```

8.2 过程语句归纳

8.2.2 深入认识不完整条件语句与时序电路的关系

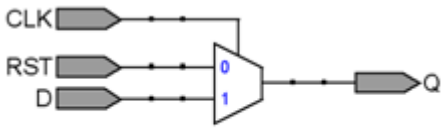


图 8-5 例 8-15 的 RTL 图

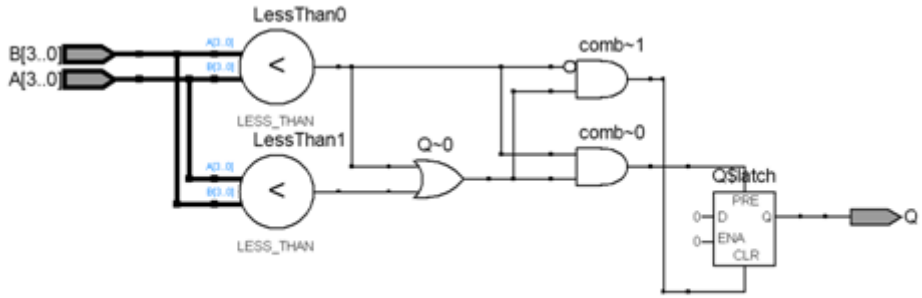


图 8-6 例 8-16 的 RTL 图，输出口被加上了锁存器

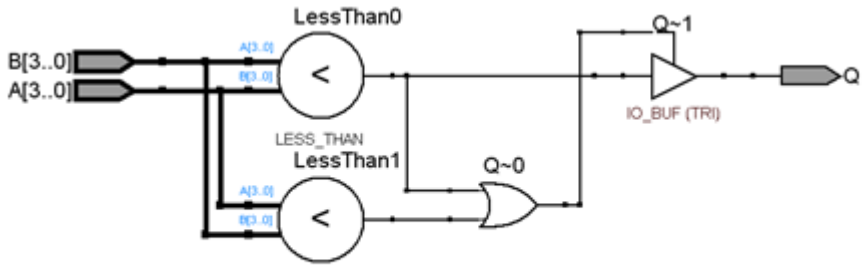


图 8-7 例 8-17 的 RLT 电路图，输出口没有锁存器，是纯组合电路

8.3 if语句归纳

8.3.1 if语句的一般表述形式

```
(1) if (条件表达式)   begin  语句块;   end
(2) if (条件表达式)   begin  语句块1;   end
                else  begin  语句块2;   end
(3) if (条件表达式1)                begin  语句块1;  end
        else if  (条件表达式2)  begin  语句块2;  end
            .
            .
        else if  (条件表达式n)  begin  语句块n;   end
                          else  begin  语句块n+1; end
```

8.3 if语句归纳

8.3.1 if语句的一般表述形式

【例8-18】

```
module CODER83 (DIN,DOUT);
  output[0:2]DOUT;  input[0:7]DIN;
  reg [0:2] DOUT;
  always @(DIN)  begin
    casez (DIN)
    8'b????????0 : DOUT<=3'b000;
    8'b??????01 : DOUT<=3'b100;
    8'b?????011 : DOUT<=3'b010;
    8'b????0111 : DOUT<=3'b110;
    8'b???01111 : DOUT<=3'b001;
    8'b??011111 : DOUT<=3'b101;
    8'b?0111111 : DOUT<=3'b011;
    8'b01111111 : DOUT<=3'b111;
    default : DOUT<=3'b000;
endcase     end    endmodule
```

【例8-19】

```
 module CODER83 (DIN,DOUT);
  output[0:2]DOUT; input[0:7]DIN;
  reg [0:2] DOUT;
  always @(DIN)
  begin
      if (DIN[7]==0)  DOUT=3'b000;
else if (DIN[6]==0)  DOUT=3'b100;
else if (DIN[5]==0)  DOUT=3'b010;
else if (DIN[4]==0)  DOUT=3'b110;
else if (DIN[3]==0)  DOUT=3'b001;
else if (DIN[2]==0)  DOUT=3'b101;
else if (DIN[1]==0)  DOUT=3'b011;
else                 DOUT=3'b111;
   end
endmodule
```

8.3 if语句归纳

8.3.1 if语句的一般表述形式

表 8-1　8 线-3 线优先编码器真值表

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

注：表中的“x”为任意

8.3 if语句归纳

8.3.1 if语句的一般表述形式

```
(DIN[7]==1)&(DIN[6]==1)&(DIN[5]==1)&(DIN[4]==1)&(DIN[3]==1)&
 (DIN[2]==1)&(DIN[1]==1)&(DIN[0]==0)          //这恰好与表6-1最后一行吻合。
```

图 8-8 例 8-18 和例 8-19 的时序仿真波形图

8.3 if语句归纳

8.3.2 关注if语句中的条件指示

【例8-20】

```
module andd(A,B,Q);
   output Q ;
   input A,B;
    reg Q;
   always @(A,B )
     if (A==0)
     if (B==0)  Q=0;
     else   Q=1;
endmodule
```

【例8-21】

```
module andd(A,B,Q);
  output Q ;
  input A,B; reg Q;
  always @(A,B )
   if (A==0)
   begin
   if (B==0)  Q=0;
  else  Q=1; end
endmodule
```

【例8-22】

```
module andd(A,B,Q);
 output Q; input A,B;
 reg Q;
 always @(A,B )
 if (A==0)
 begin if(B==0)  Q=0;
 end
 else   Q=1;
 endmodule
```

8.3 if语句归纳

8.3.2 关注if语句中的条件指示

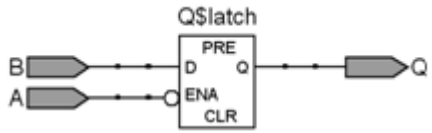


图 8-9 例 8-20/8-21 的 RTL 图

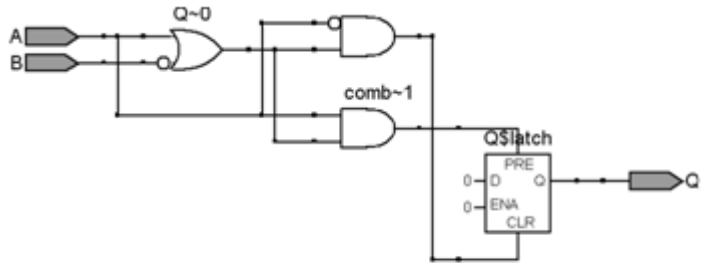


图 8-10 例 8-22 的 RTL 图

8.4 三态与双向端口设计

8.4.1 三态控制电路设计

【例8-23】

```
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
```

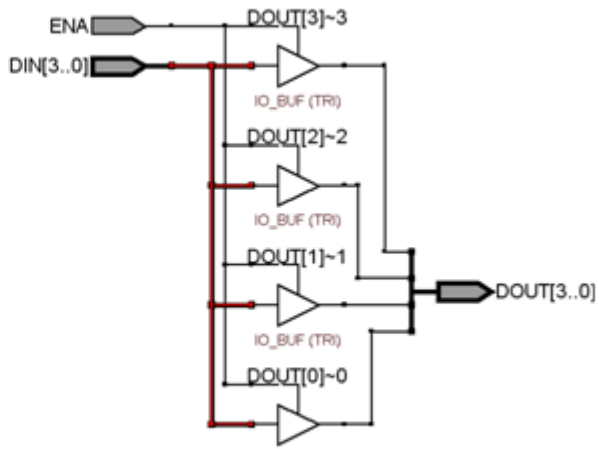


图8-11　4位三态控制门电路

8.4 三态与双向端口设计

8.4.2 双向端口设计

【例 8-24】

```
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
```

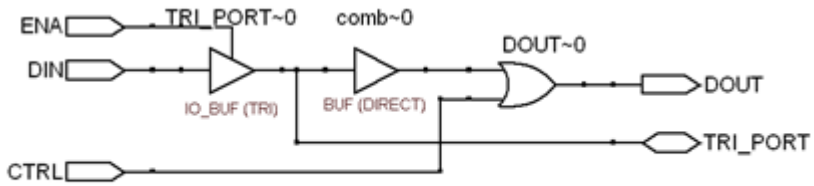


图 8-12 例 8-24 的 1 位双向端口电路设计之 RTL 图

8.4 三态与双向端口设计

8.4.2 双向端口设计

【例8-25】

```
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
```

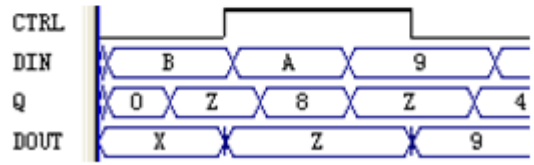


图8-13 本例的仿真波形图

【例8-26】

```
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
```

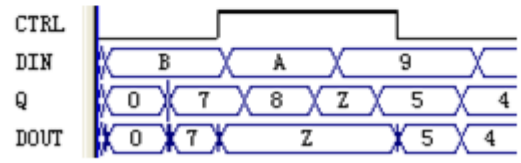


图8-14 本例的仿真波形图

8.4 三态与双向端口设计

8.4.2 双向端口设计

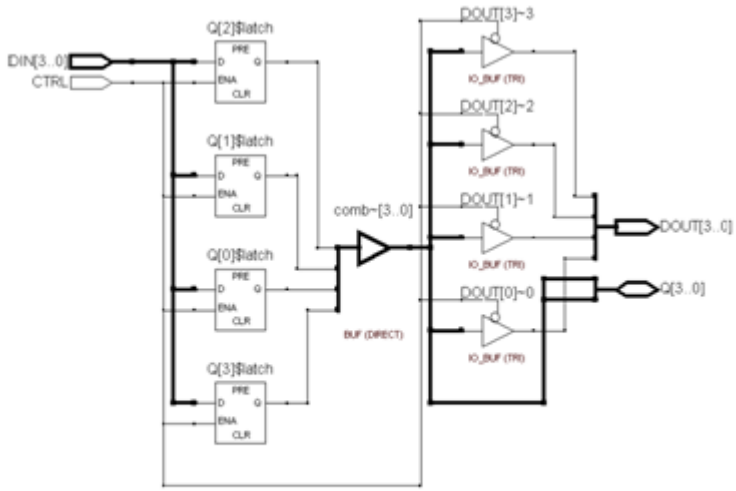


图 8-15 例 8-25 的 RTL 图

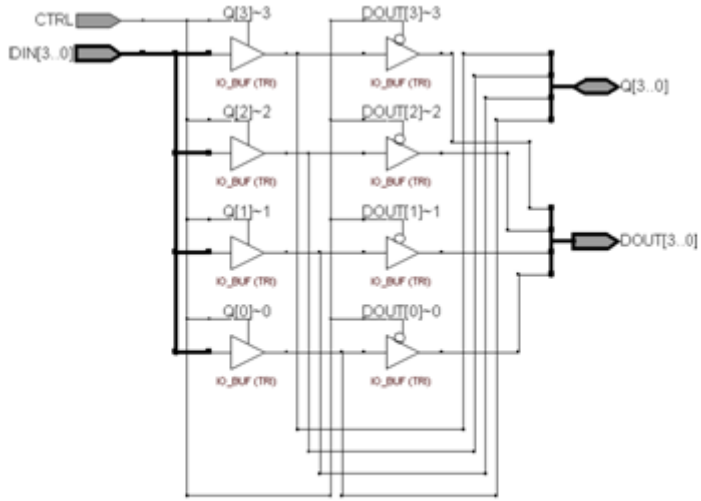


图 8-16 例 8-26 的 RLT 图

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

【例8-27】

```
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8-28】

```
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
```

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

【例 8-29】

```
module mux4_1(IN3,IN2,IN1,IN0,ENA,DOUT);
   input [3:0] IN3,IN2,IN1,IN0; input[1:0]  ENA;
    output[3:0] DOUT;
    assign DOUT = (ENA==2'B00) ? IN0 : 4'HZ;
    assign DOUT = (ENA==2'B01) ? IN1 : 4'HZ;
    assign DOUT = (ENA==2'B10) ? IN2 : 4'HZ;
    assign DOUT = (ENA==2'B11) ? IN3 : 4'HZ;
endmodule
```

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

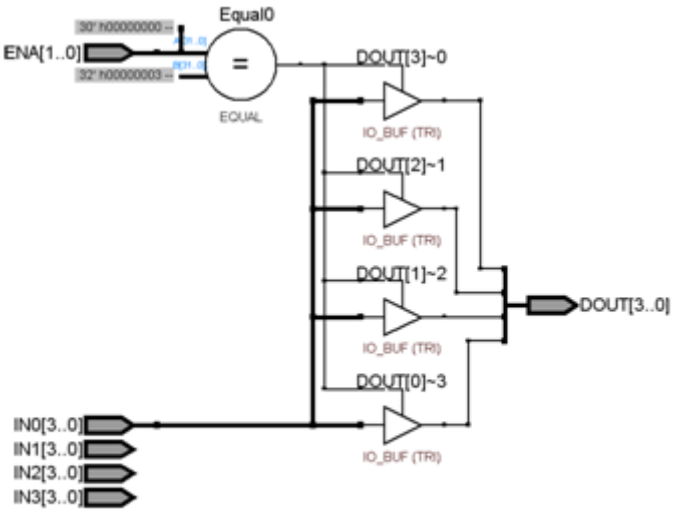


图8-17 例8-27的RTL图

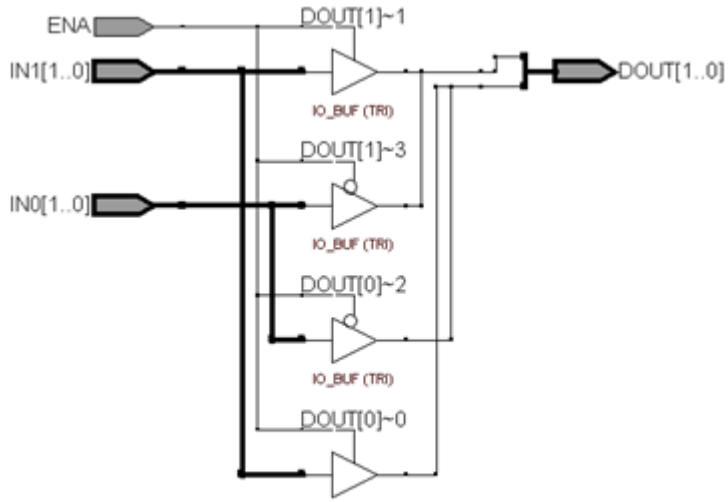
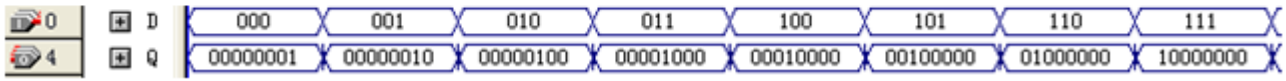


图8-18 例8-28的2位简化RTL图

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 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

8-1 硬件消抖动电路设计

【例8-30】

```
module ERZP  (CLK, KIN,KOUT);
   input   CLK, KIN;                    //工作时钟和输入信号
   output  KOUT;     reg KOUT;
   reg [3:0] KH,KL;                     //定义对高电平和低电平脉宽计数之寄存器。
   always @(posedge CLK)  if (!KIN)  KL<=KL+1;//对键输入的低电平脉宽计数
         else  KL<=4'b0000;             //若出现高电平，则计数器清 0
   always @(posedge CLK)  if (KIN)  KH<=KH+1;//同时对键输入的高电平脉宽计数
          else KH<=4'b0000;             //若出现高电平，则计数器清 0
   always @(posedge CLK)  begin
         if (KH>4'b1100) KOUT<=1'B1;//对高电平脉宽计数一旦大于 12，则输出 1
    else if(KL>4'b0111) KOUT<=1'B0;//对低电平脉宽计数若大于 7，则输出 0
    end
endmodule
```

图 8-19 例 8-30 消抖动电路仿真波形

实验与设计

8-2 4X4阵列键盘键信号检测电路设计

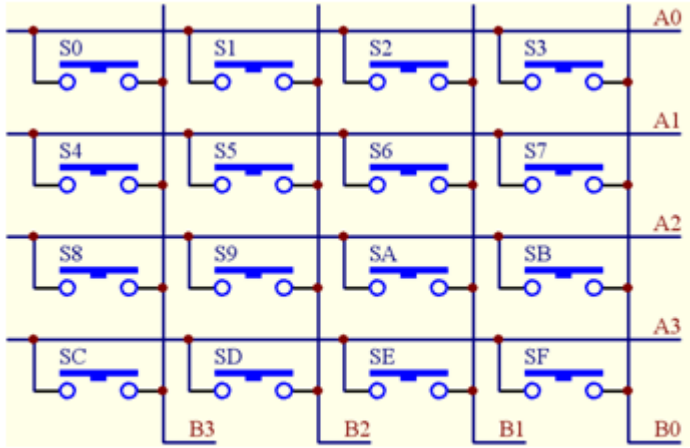


图 8-20　4X4 键盘电路

实验与设计

8-2 4X4阵列键盘键信号检测电路设计

【例 8-31】

```
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
```

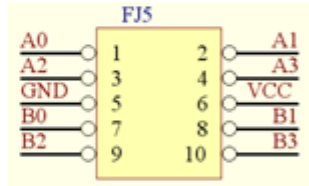


图 8-21 4X4 键盘的 10 芯接口

8-3 直流电机综合测控系统设计

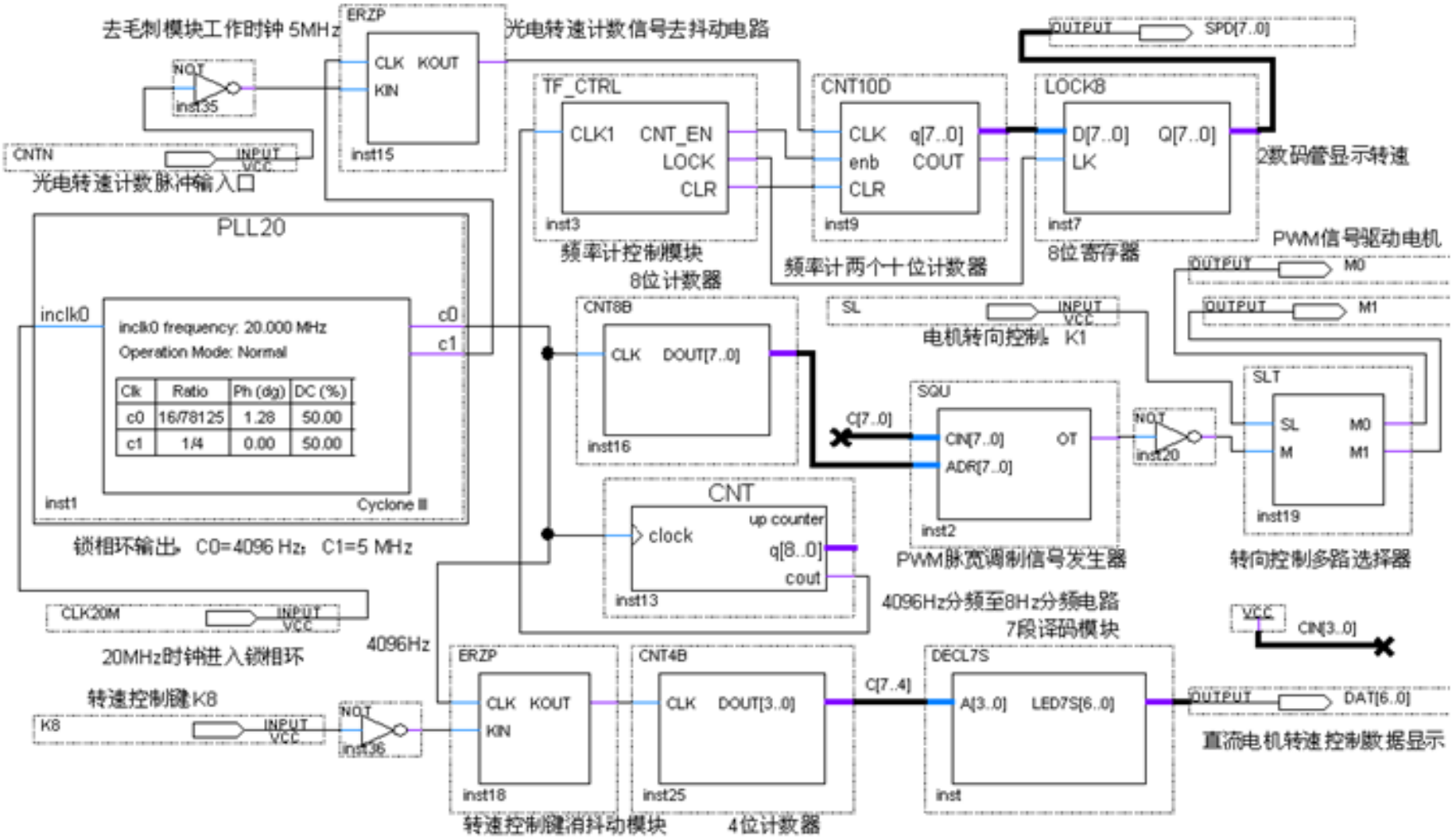


图 8-22 直流电机驱动控制电路顶层设计

实验与设计

8-3 直流电机综合测控系统设计

【例 8-32】

```
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
```

实验与设计

8-4 VGA简单图像显示控制模块设计

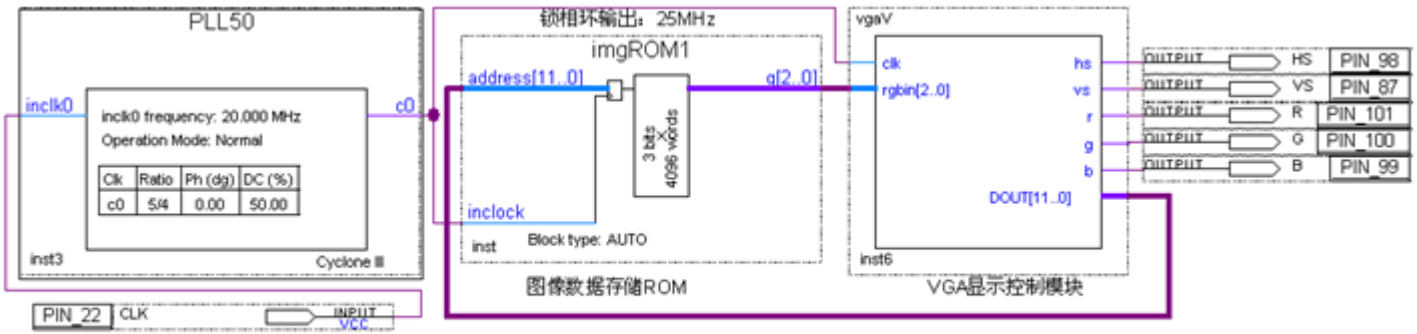


图 8-23 VGA 图像显示控制模块原理图

【例 8-33】

```
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
```

实验与设计

8-5 乐曲硬件演奏电路设计

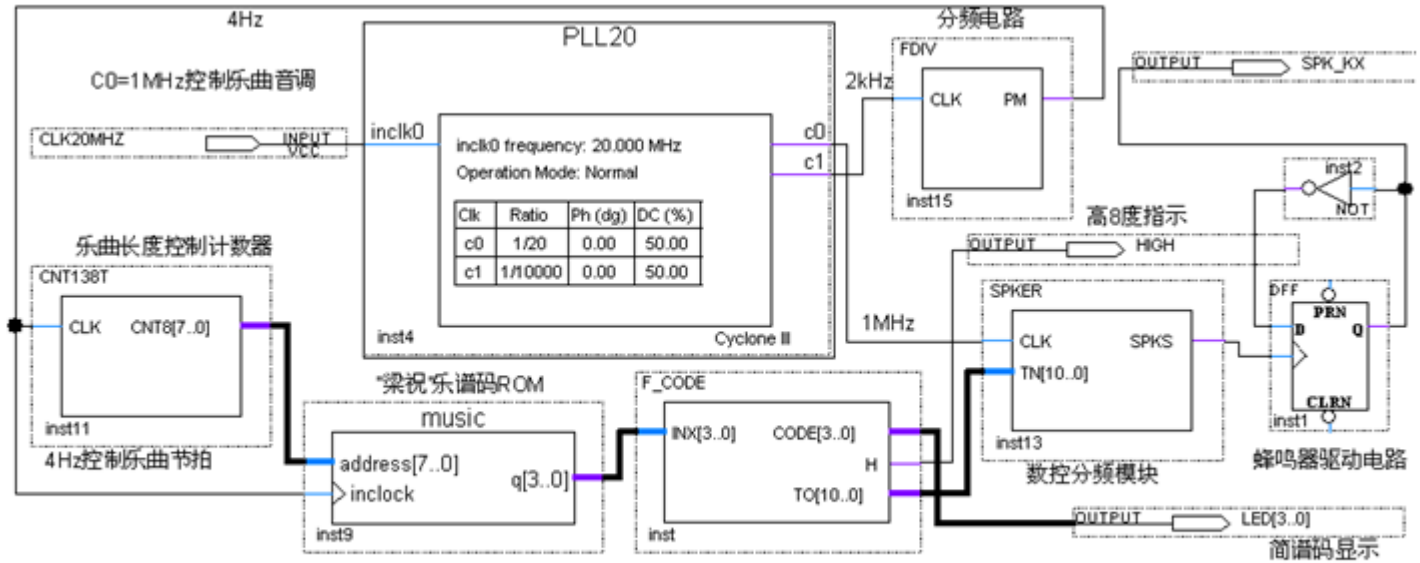


图 8-24 乐曲演奏电路顶层设计

实验与设计

8-5 乐曲硬件演奏电路设计

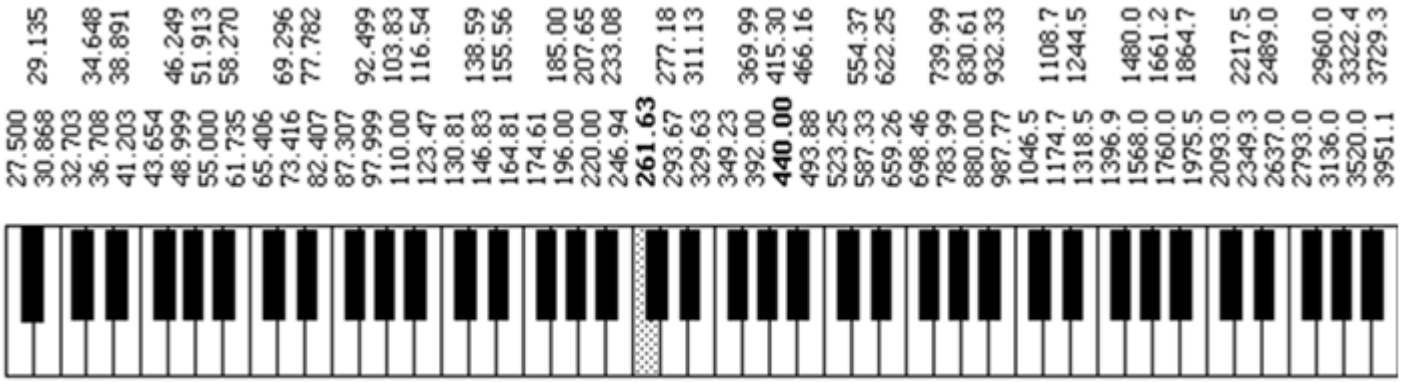


图 8-25 电子琴音阶基频对照图（单位 Hz）

实验与设计

8-5 乐曲硬件演奏电路设计

【例 8-34】

```
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 8-35】

```
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

8-5 乐曲硬件演奏电路设计

【例 8-36】

```
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 8-37】

```
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
```

【例 8-38】

```
  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

8-5 乐曲硬件演奏电路设计

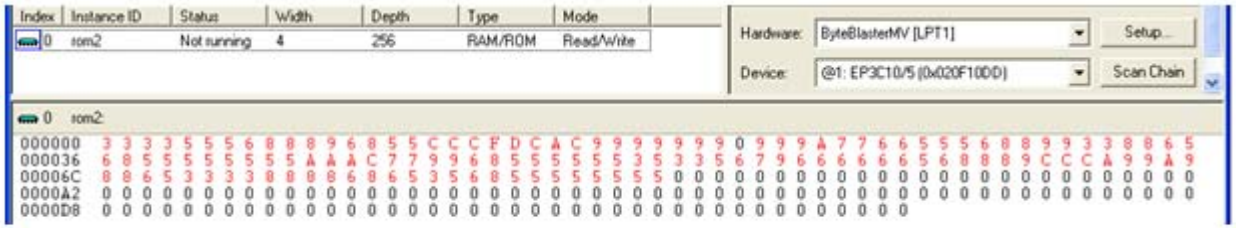


图 8-26 In-System Memory Content Editor 对 MUSIC 模块的数据读取