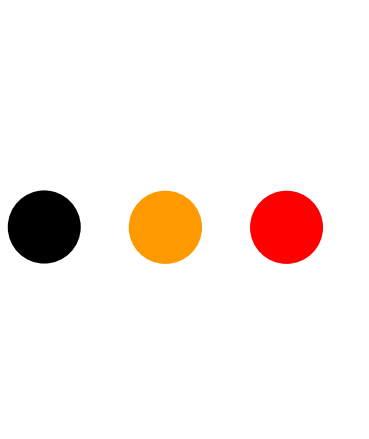

第9章

Verilog系统设计优化

9.1 资源优化

9.1.1 资源共享

【例 9-1】

```
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 9-2】

```
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
```

9.1 资源优化

9.1.1 资源共享

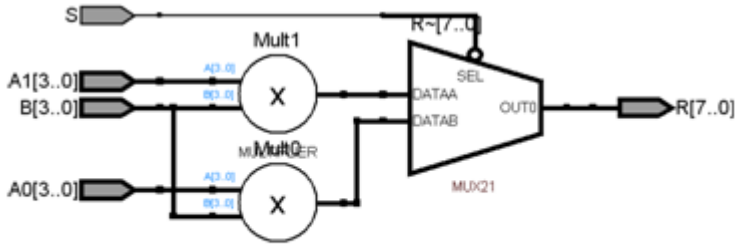

图 9-1 先乘后选择的设计方法 RTL 结构

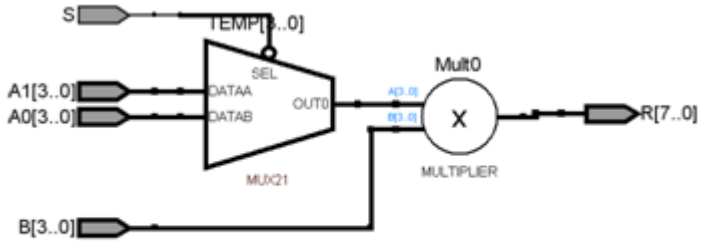

图 9-2 先选择后乘设计方法 RTL 结构

9.1 资源优化

9.1.1 资源共享

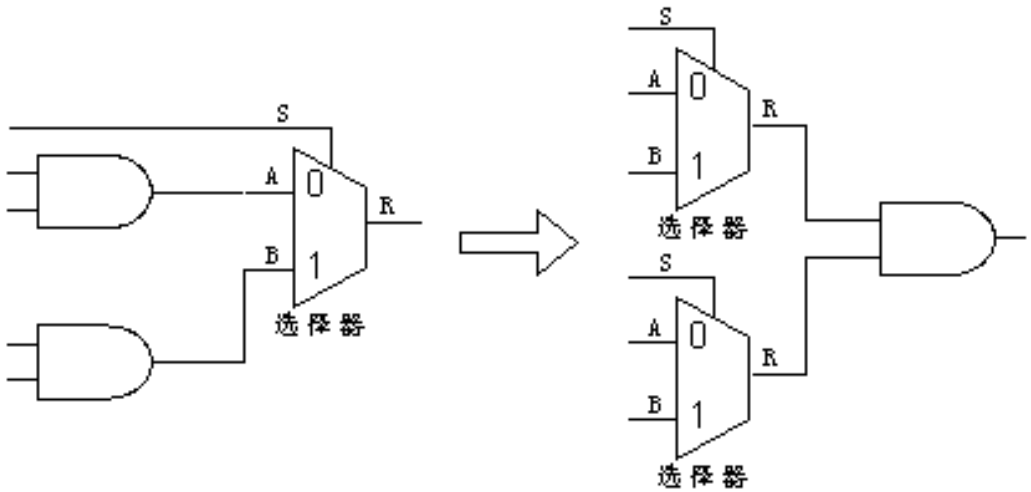

图 9-3　资源共享反例

9.1 资源优化

9.1.2 逻辑优化

【例 9-3】

```
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 9-4】

```
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
```

9.1 资源优化

9.1.3 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 9-5】

```
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
```

9.1 资源优化

9.1.3 串行化

【例 9-6】

```
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
```

9.2 速 度 优 化

9.2.1 流水线设计

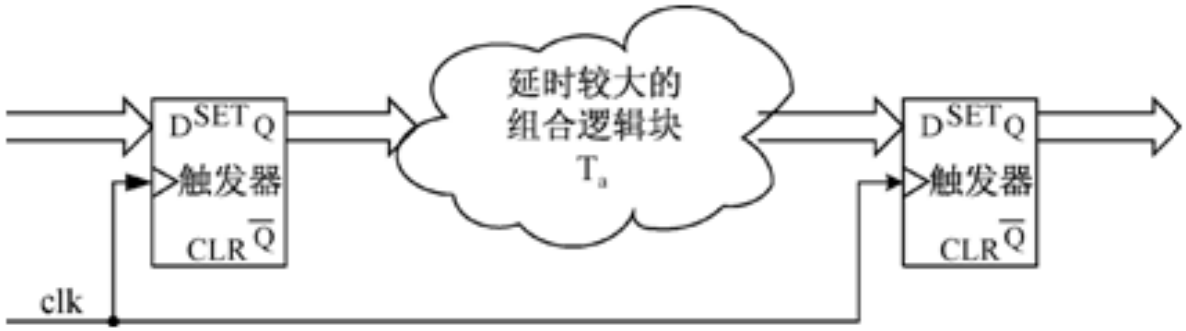

图 9-4 未使用流水线

9.2 速 度 优 化

9.2.1 流水线设计

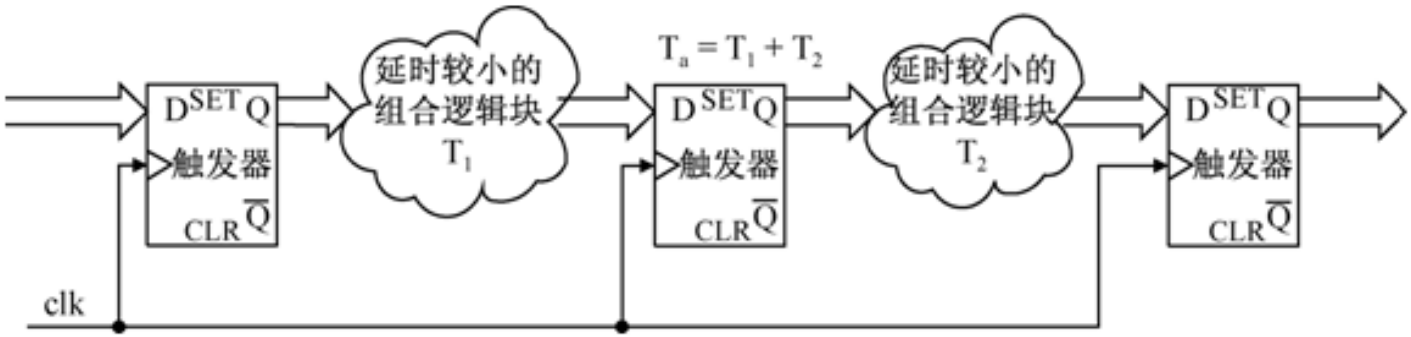

图 9-5 使用流水线结构

9.2 速 度 优 化

9.2.1 流水线设计

图 9-6 流水线工作图示

9.2 速度优化

9.2.1 流水线设计

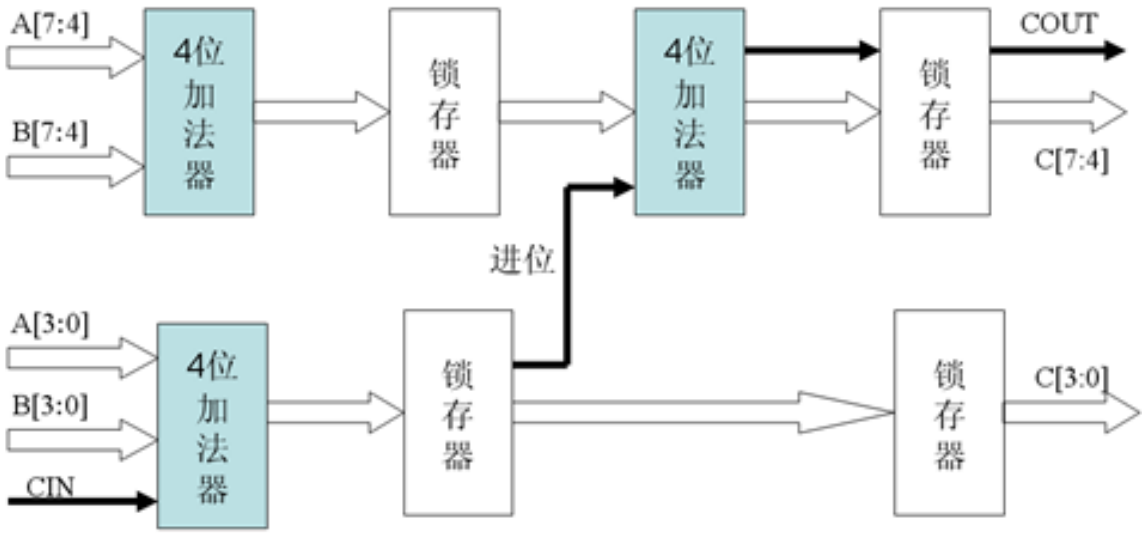

图 9-7　8 位加法器流水线工作图示

9.2 速 度 优 化

9.2.1 流水线设计

【例 9-7】普通加法器，EP3C5 综合结果：LCs=10,REG=0（纯组合逻辑）,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
```

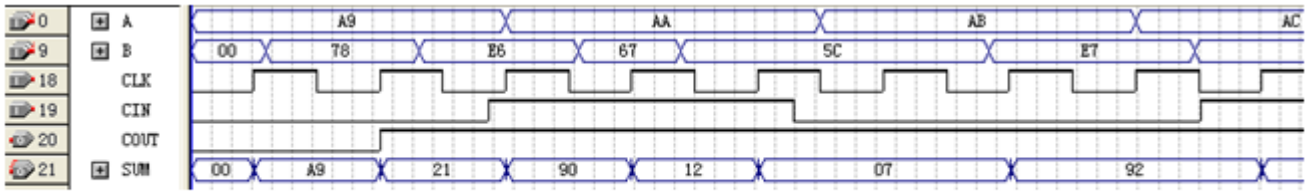

图 9-8　例 9-7 的时序仿真波形

9.2 速 度 优 化

9.2.1 流水线设计

【例 9-8】流水线加法器，EP3C5 综合结果：T=3.63ns，LCs=24，REG=22（时序逻辑）。

```
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
```

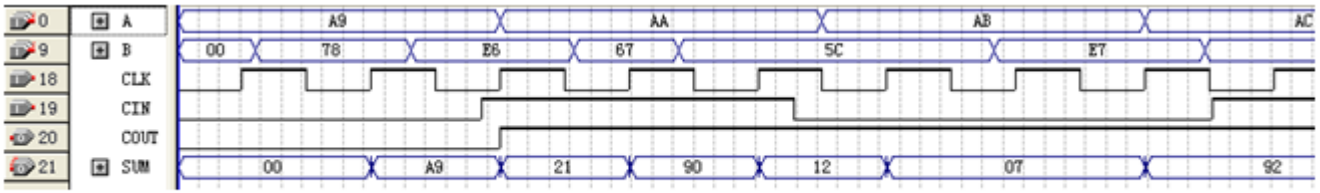

图 9-9　例 9-8 的时序仿真波形

9.2 速 度 优 化

9.2.2 寄存器配平

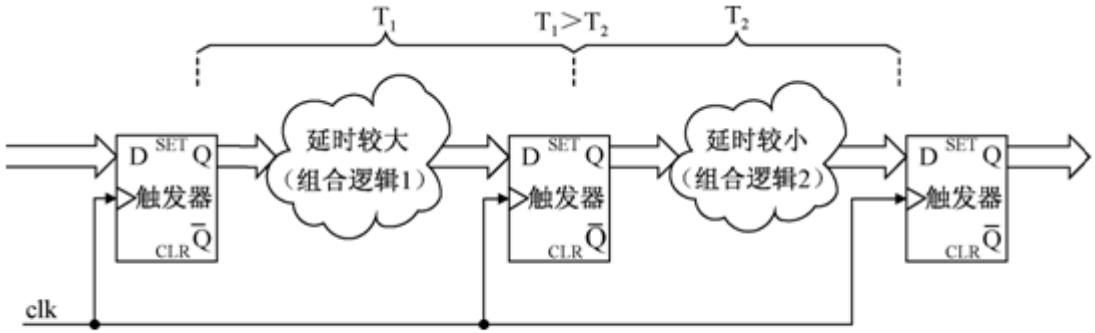

图 9-10 不合理的电路结构

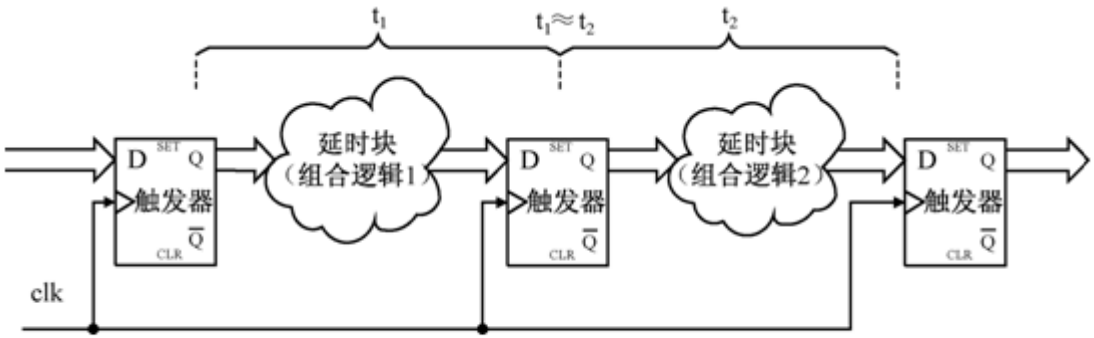

图 9-11 寄存器配平后的结构

9.2 速 度 优 化

9.2.3 关键路径法

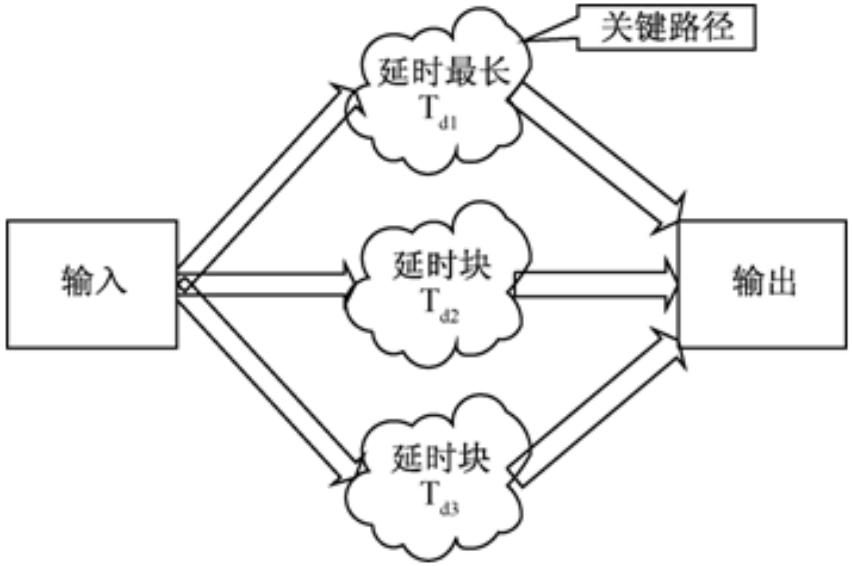

图 9-12 关键路径示意

9.2 速度优化

9.2.4 乒乓操作法

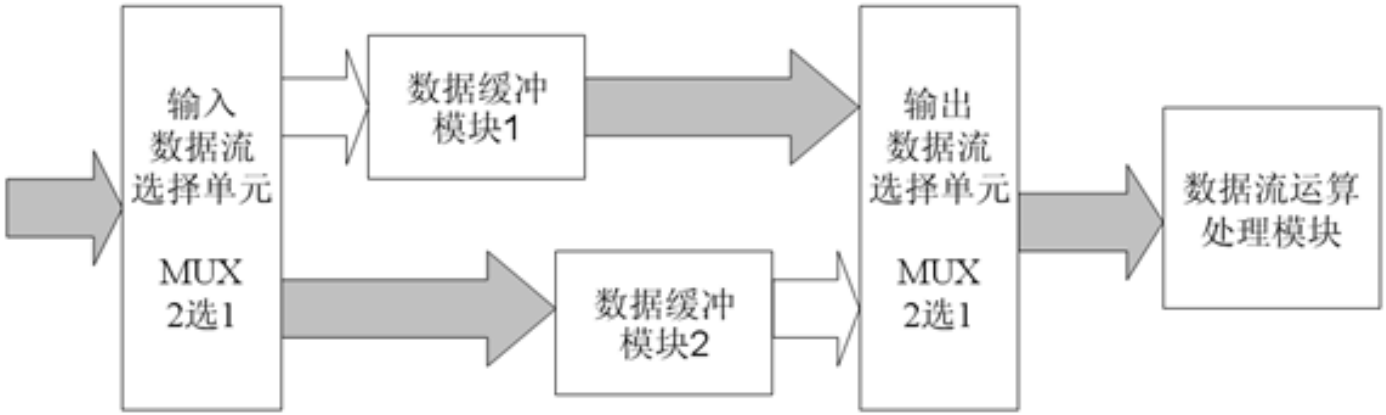

图 9-13　乒乓操作数据缓存结构示意图

9.2.5 加法树法

习　题

```
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
```

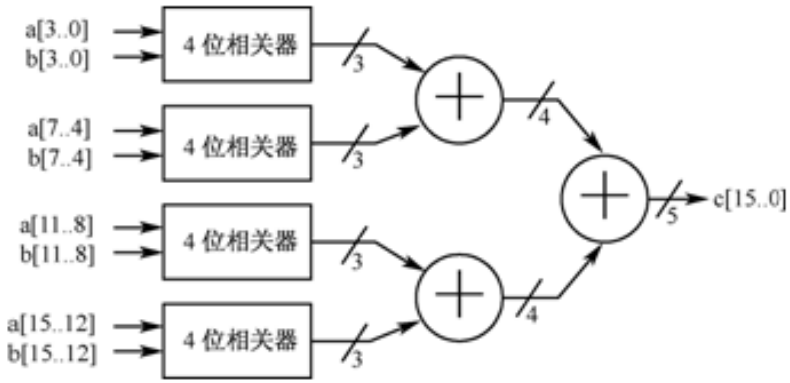

图 9-20　习题 9-9 图

实验与设计

9-1 采用流水线技术设计高速数字相关器

9-2 线性反馈移位寄存器设计

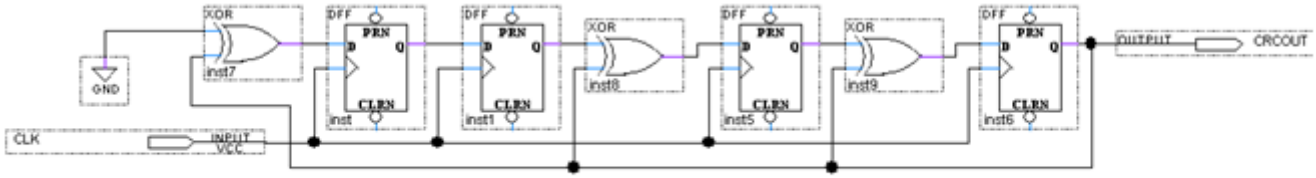

图 9-21 LFSR 举例

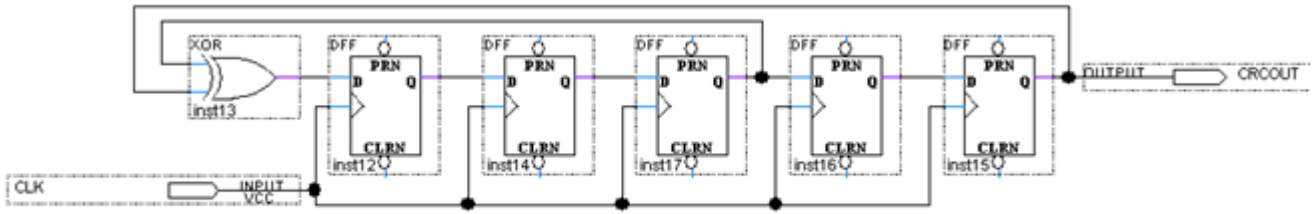

图 9-22 另一种 LFSR 结构

9-3 基于UART串口控制的模型电子琴设计

【例 9-9】

```
module RXDS (SYSCLK, RXD, DOUT);
   input SYSCLK, RXD;   output[7:0] DOUT;
   wire[7:0] DOUT;  wire FRXD,  GATE;
   reg[9:0] B;  reg[3:0] R;  reg[15:0] J;
   reg GT, GTCLR, CCLK;
   always @(posedge SYSCLK or negedge GT)  begin : S1
      if (GT==1'b0)     J<=16'H0000;
       else  begin  if (J==16'H0068)  J<=16'H0000 ;
                     else        J<=J + 1 ;  end   end
   always @(J)   begin : S2
      if (J==16'H0039)   CCLK<=1'b1; else  CCLK<=1'b0 ;  end
   always @(posedge GATE or posedge GTCLR)  begin : S3
      if (GTCLR==1'b1)   R<=4'b0000 ;  else  R<=R+1 ;   end
   always @(GATE or R)   begin : S4
      if (R==4'b1010)  GTCLR<=~GATE;   else  GTCLR<=1'b0;  end
   always @(posedge GATE)         begin : S5
      B[9:0] <= {B[8:0], RXD} ;   end
   always @(posedge FRXD or posedge GTCLR)  begin : S6
      if (GTCLR==1'b1)   GT<=1'b0;  else  GT<=1'b1 ;  end
   assign DOUT = {1'b0, B[8:2]} ;
   assign GATE = GT & CCLK ;         assign FRXD = ~RXD ;
endmodule
```

实验与设计

9-3 基于UART串口控制的模型电子琴设计

【例 9-10】

```
module CODE3 (DIN, KEY);
  input[7:0] DIN;   output[3:0] KEY;   reg[3:0] KEY;
  always @(DIN)   begin
    case (DIN)
      8'b01000110 : KEY<=4'b0001; 8'b00100110 : KEY<=4'b0010 ;
      8'b01100110 : KEY<=4'b0011; 8'b00010110 : KEY<=4'b0100 ;
      8'b01010110 : KEY<=4'b0101; 8'b00110110 : KEY<=4'b0110 ;
      8'b01110110 : KEY<=4'b0111; 8'b00001110 : KEY<=4'b1000 ;
      8'b01001110 : KEY<=4'b1001; 8'b00000110 : KEY<=4'b0000 ;
          default : KEY<=4'b0000 ;
    endcase  end
endmodule
```

实验与设计

9-3 基于UART串口控制的模型电子琴设计

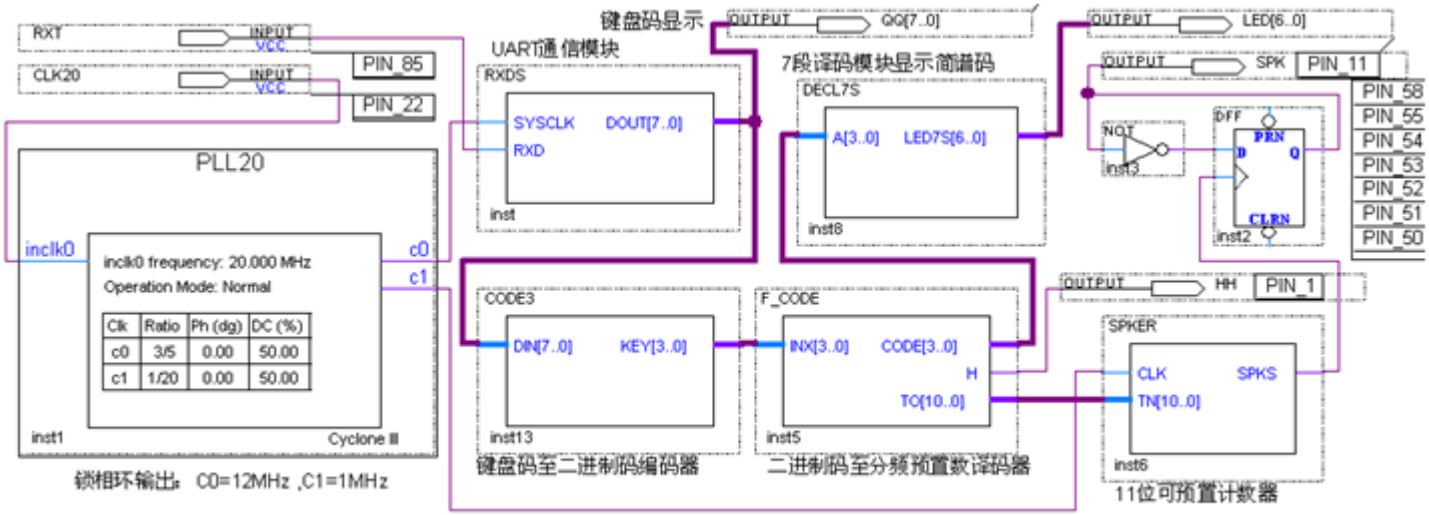

图 9-23 UART 串口控制模型电子琴电路顶层设计

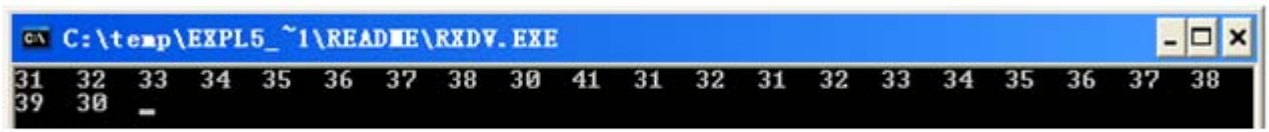

图 9-24 串口通信及键盘 ASCII 码显示程序窗口

实验与设计

9-4 PS2键盘控制模型电子琴电路设计

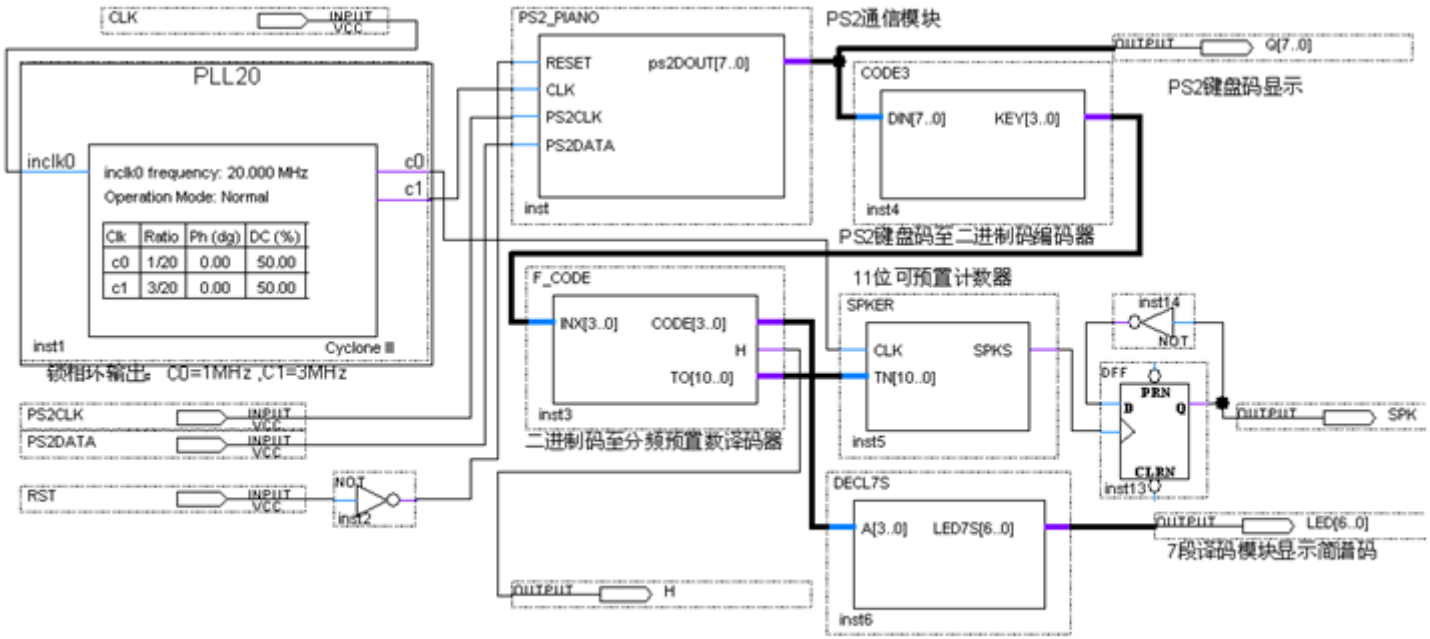

图 9-25　PS2 键盘控制模型电子琴电路顶层设计

实验与设计

9-4 PS2键盘控制模型电子琴电路设计

表 9-1 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

【例 9-11】

```
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
```

接下页

实验与设计

9-4 PS2键盘控制模型电子琴电路设计

```
            cnt <= cnt + 1 ;   end
          else if (cnt == 8)   begin if (kb_data == parity)
            begin  dataerror <= 1'b0 ;  end
            else  begin  dataerror<=1'b1;  end
            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

9-4 PS2键盘控制模型电子琴电路设计

【例 9-12】

```
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

9-5 AM幅度调制信号发生器设计

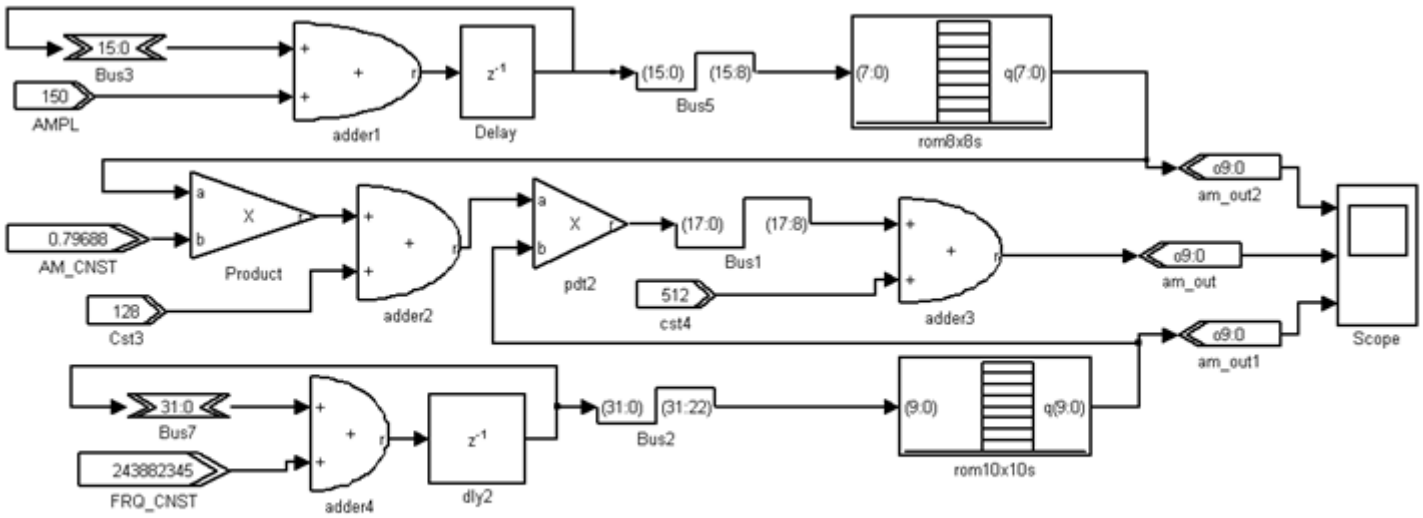

图 9-26　AM 信号发生器 DSP Builder/MATLAB Simulink 模型

实验与设计

9-5 AM幅度调制信号发生器设计

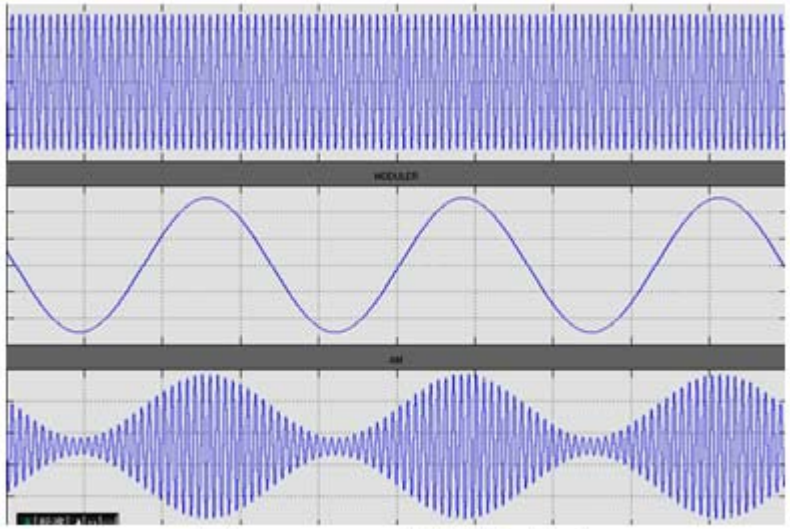

图 9-27　AM 模型仿真波形