


第9章

Verilog语句语法规则补充



9.1 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 ， C 即获得赋值 32'hFFFFFFFB。  
B<= -7'd30;     // -7'd30 = 7'H62，B 实际获得赋值=6'H22，高 1 位被截去。
```



9.1 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
```



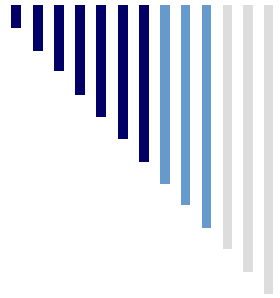
9.1 Verilog文字规则

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*

5. 关键词



9.2 数据类型

9.2.1 net网线类型

9.2.2 register寄存器类型

9.2.3 存储器类型



9.3 操作符

- 单目操作符 (unary operators)
 - 双目操作符 (binary operators)
 - 三目操作符 (ternary operators)
-



9.3 操作符

1. 逻辑操作符

例如设 $A=4'b1001$, $B=4'b0001$; 则:

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

- $\&\&$ 逻辑与
- $||$ 逻辑或
- $!$ 逻辑非。例如 $!A=0$

2. 缩位操作符

$\&$ (与)、 $\sim\&$ (与非)、
 $|$ (或)、 $\sim|$ (或非)、
 \wedge (异或)、 $\wedge\sim$, $\sim\wedge$ (同或)。

9.4 常用语句补充

9.4.1 initial过程语句使用示例

【例 9-1】 用 initial 过程语句对测试变量赋值

```
`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
```



9.4 常用语句补充

9.4.2 forever循环语句

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

9.4.3 编译指示语句

```
`define、`include、`ifdef、`else、`endif。
```

9.4 常用语句补充

1. 文件包含语句`include`

```
`include "文件名"
```

【例 9-2】

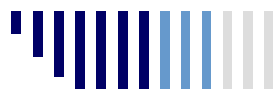
```
'include " h_adder.v "  
'include " or2a.v "  
module f_adder(ain,bin,cin,cout,sum); //例 4-9  
    output cout,sum ;  
    input ain,bin,cin ;  
    wire e,d,f ;  
    h_adder u1( ain, bin, e, d );  
    h_adder u2(.a(e), .so(sum), .b(cin),.co(f) );  
    or2a u3(.a(d), .b(f), .c(cout) );  
endmodule
```



9.4 常用语句补充

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
<pre>'ifdef 宏名 语句块 'endif</pre>	<pre>'ifdef 宏名 语句块 1 'else 语句块 2 'endif</pre>

**【例 9-3】**

```
`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
```

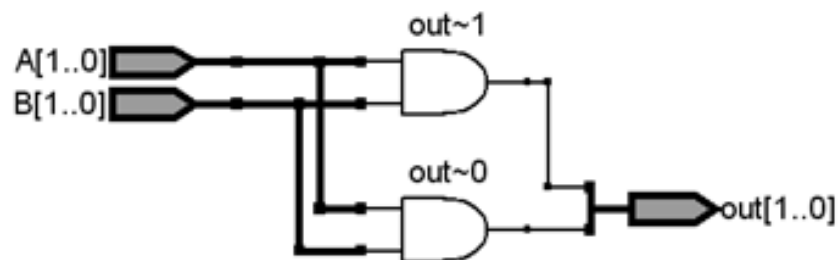


图 9-1 对应例 9-3 的 RTL 图

【例 9-4】

```
`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
```

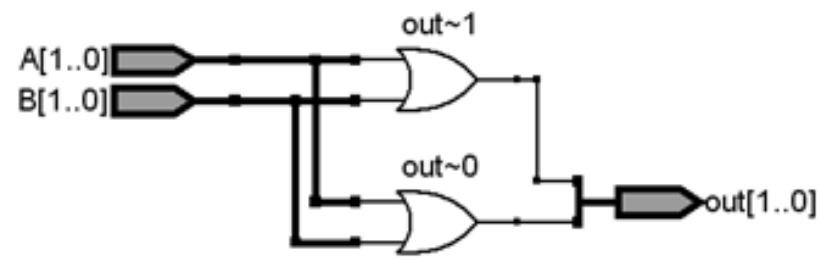


图 9-2 对应例 9-4 的 RTL 图



9.4 常用语句补充

9.4.4 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
<pre>task <任务名>; 端口及数据类型声明语句 begin 过程语句; end endtask</pre>	<pre><任务名> (端口 1, 端口 2, ..., 端口 N);</pre>



【例 9-5】

```
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
```

9.4 常用语句补充

9.4.4 任务和函数语句

1.任务 (task) 语句

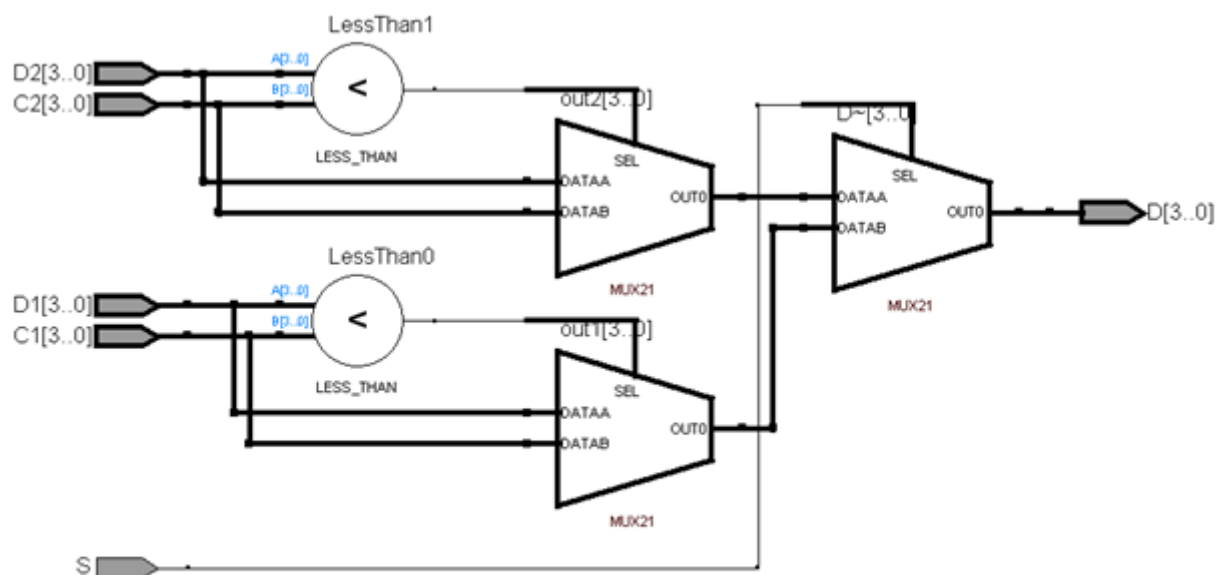


图 9-3 例 9-5 的 RTL 图



9.4 常用语句补充

9.4.4 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
<pre>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</pre>	<pre><函数名> (输入参数 1, 输入参数 2, ...)</pre>

【例 9-6】

```
module CN (A, OUT);  
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; //含 1 的位个数累加  
GP=CNT; end  
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
```

A	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
OUT	0	1	2	1	2	3	1	2	3	2	3	2	3	4		

图 9-4 例 9-6 的仿真图

9.5 用库元件实现结构描述

【例 9-7】

```
module LOGICGATE (A,B,C,S,OUT);  
  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
```

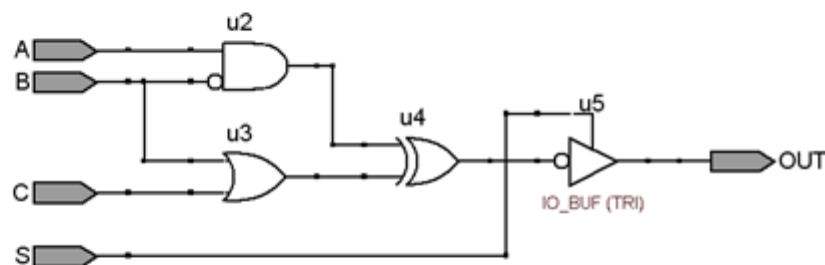


图 9-5 例 9-7 描述的逻辑电路

基本门元件名	<门例化名>	(<端口关联列表>)
--------	--------	------------

9.5 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>)

```
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,ctrl);         //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

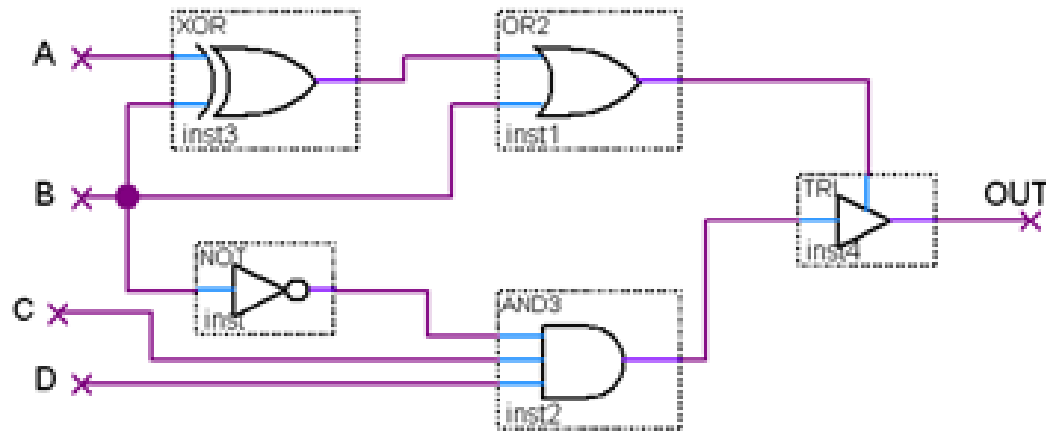


图 9-6 习题 9.3 逻辑电路图

EDA实验与创新实践

9-1 乐曲硬件演奏电路设计

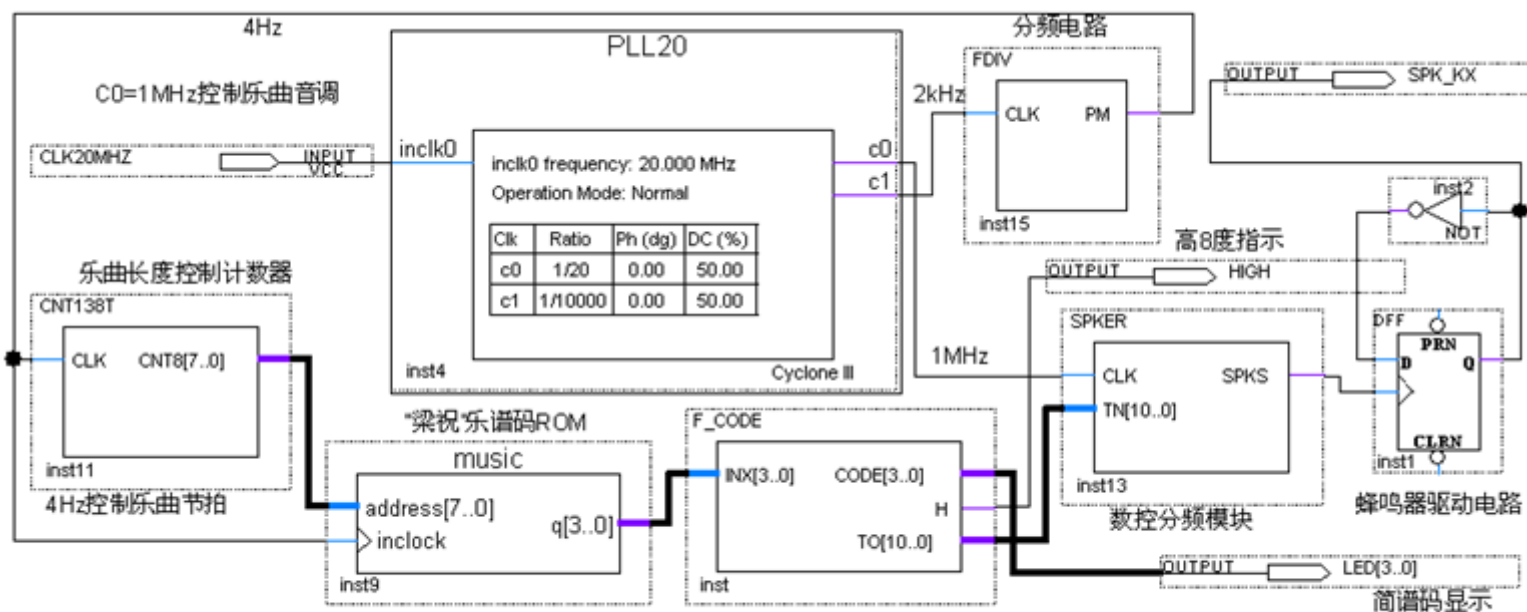


图 9-7 乐曲演奏电路顶层设计

EDA实验与创新实践

9-1 乐曲硬件演奏电路设计

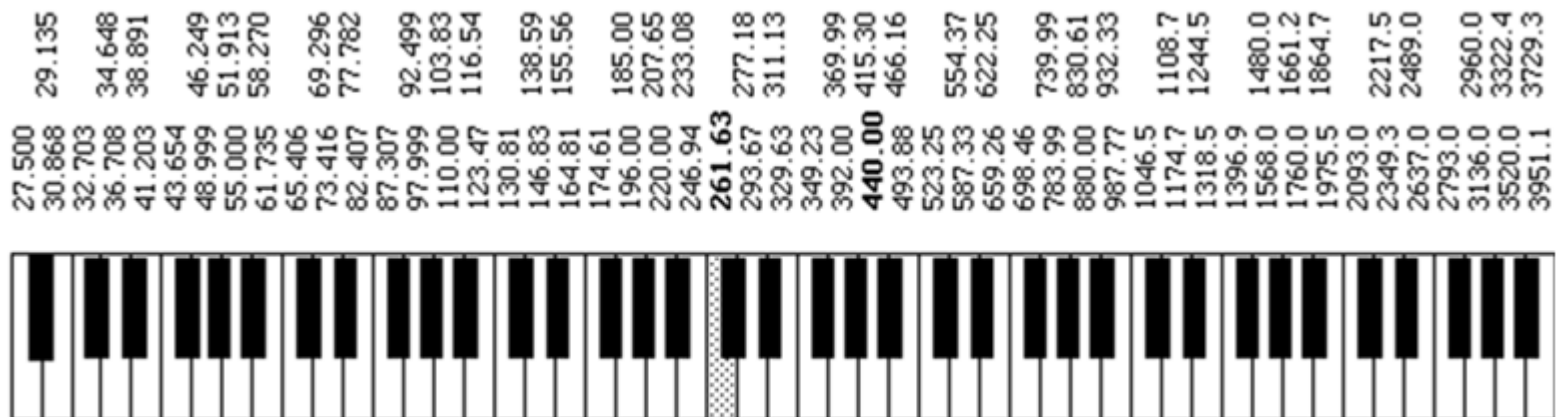


图 9-8 电子琴音阶基频对照图（单位 Hz）



EDA实验与创新实践

9-1 乐曲硬件演奏电路设计

【例 9-8】

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

【例 9-9】

```
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)
  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
endmodule
```



EDA实验与创新实践

9-1 乐曲硬件演奏电路设计

【例 9-10】 // 11 位可预置计数器

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

【例 9-11】

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

【例 9-12】

```
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 ;
```

EDA实验与创新实践

9-1 乐曲硬件演奏电路设计

Index	Instance ID	Status	Width	Depth	Type	Mode
0	rom2	Not running	4	256	RAM/ROM	Read/Write

Index	Instance ID	Status	Width	Depth	Type	Mode
0	rom2	Not running	4	256	RAM/ROM	Read/Write

Address	Data
000000	3 3 3 3 5 5 5 6 8 8 8 9 6 8 5 5 C C C F D C A C 9 9 9 9 9 9 9 0 9 9 9 A 7 7 6 6 5 5 5 6 8 8 9 9 3 3 8 8 6 5
000036	6 8 5 5 5 5 5 5 A A C 7 7 9 9 6 8 5 5 5 5 3 5 3 3 5 6 7 9 6 6 6 6 6 5 6 8 8 8 9 C C C A 9 9 A 9
00006C	8 8 6 5 3 3 3 3 8 8 8 8 6 8 6 5 3 5 6 8 5 5 5 5 5 5 0
0000A2	0 0
0000D8	0 0

图 9-9 In-System Memory Content Editor 对 MUSIC 模块的数据读取

EDA实验与创新实践

9-2 PS2键盘控制模型电子琴电路设计

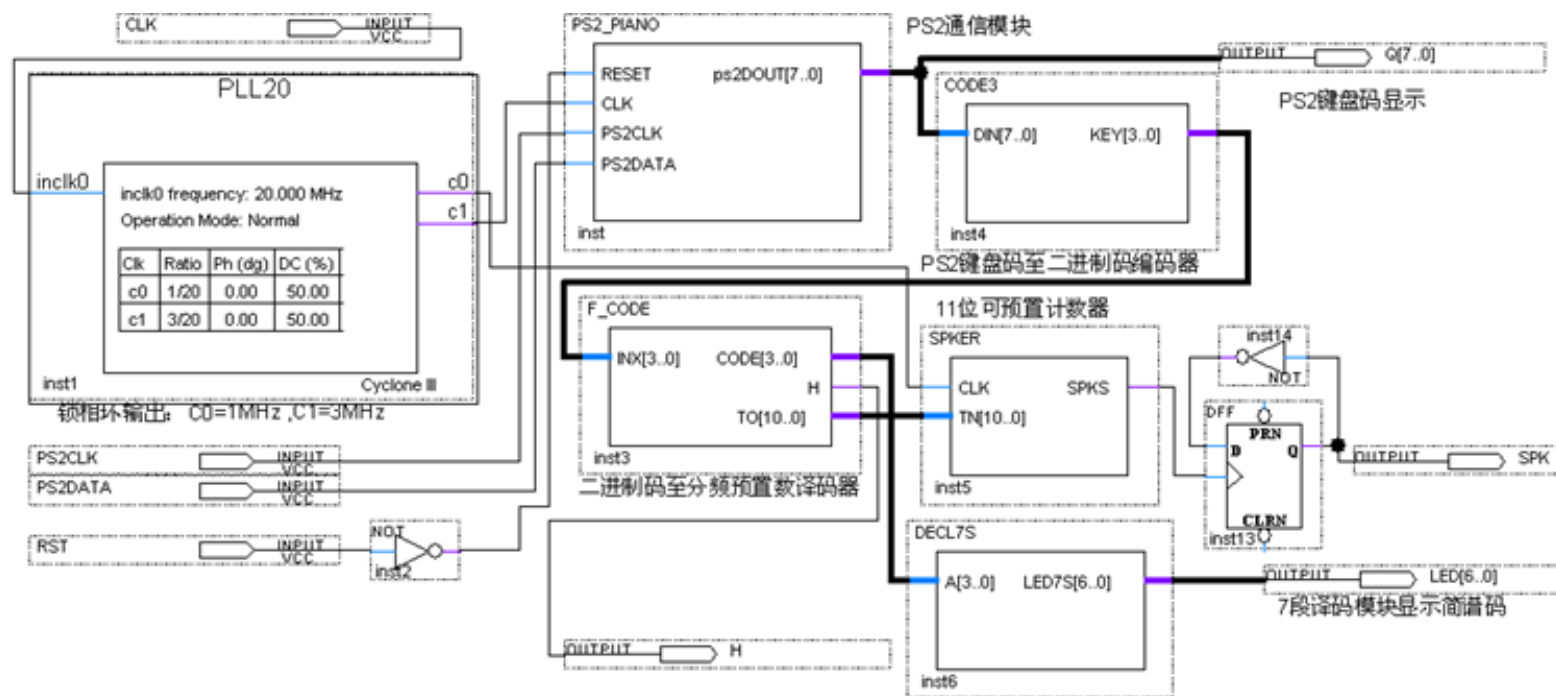


图 9-10 PS2 键盘控制模型电子琴电路顶层设计

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 9-13】

```
module PS2_PIANO (CLK, PS2CLK, PS2DATA, PS2DOUT);
    input CLK, PS2CLK, PS2DATA ; output[7:0] PS2DOUT;
    reg[7:0] PS2DOUT, shiftdata;
    wire[7:0] kbcodereg; reg[3:0] cnt;
    reg datacoming, kbclkfall, kbclkreg, parity, isfo;
    always @(posedge CLK)
    begin kbclkreg<=PS2CLK ; kbclkfall<=kbclkreg & (~PS2CLK); end
    always @(posedge CLK) begin
        if (kbclkfall == 1'b1 & datacoming == 1'b0 & PS2DATA == 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
                begin if (PS2DATA==1'b1) datacoming <= 1'b0 ; end
                    cnt <= cnt + 1 ; end
                else if (cnt==8) cnt<=cnt+1 ;
                    else begin shiftdata <= {PS2DATA, shiftdata[7:1]} ;
                        parity<=parity^PS2DATA ; cnt<=cnt+1 ; end
            end
        end
    always @(posedge CLK) begin
        if (cnt==10) begin if (shiftdata==8'b11110000) isfo<=1'b1 ;
            else PS2DOUT <= shiftdata ; end
    end
endmodule
```



EDA实验与创新实践

9-2 PS2键盘控制模型电子琴电路设计

【例 9-14】

```
module CODE3 (input [7:0] DIN, output reg[3:0] KEY) ;
  always @(DIN)
    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
endmodule
```

EDA实验与创新实践

9-3. 简易逻辑分析仪设计

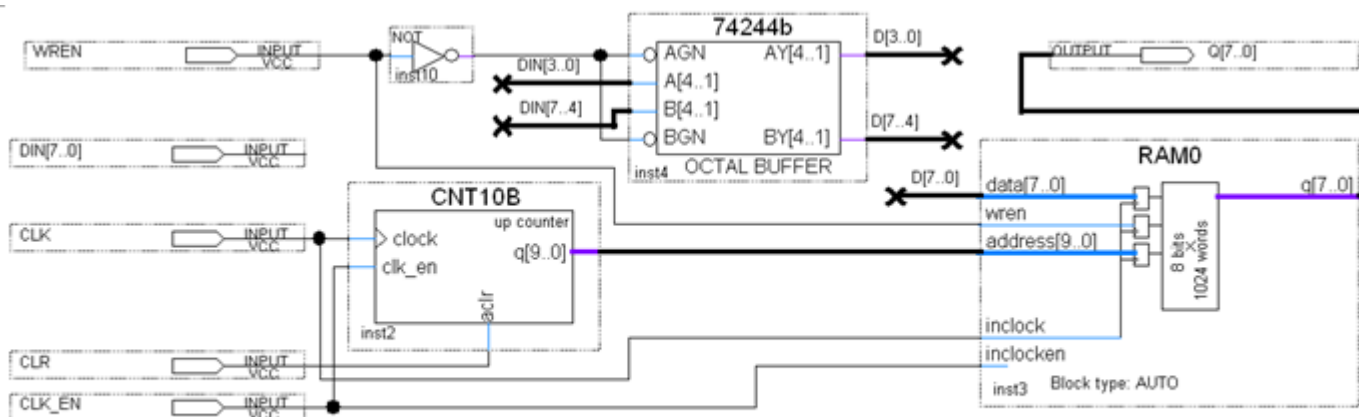


图 9-11 逻辑数据采样电路顶层设计

EDA实验与创新实践

9-3. 简易逻辑分析仪设计

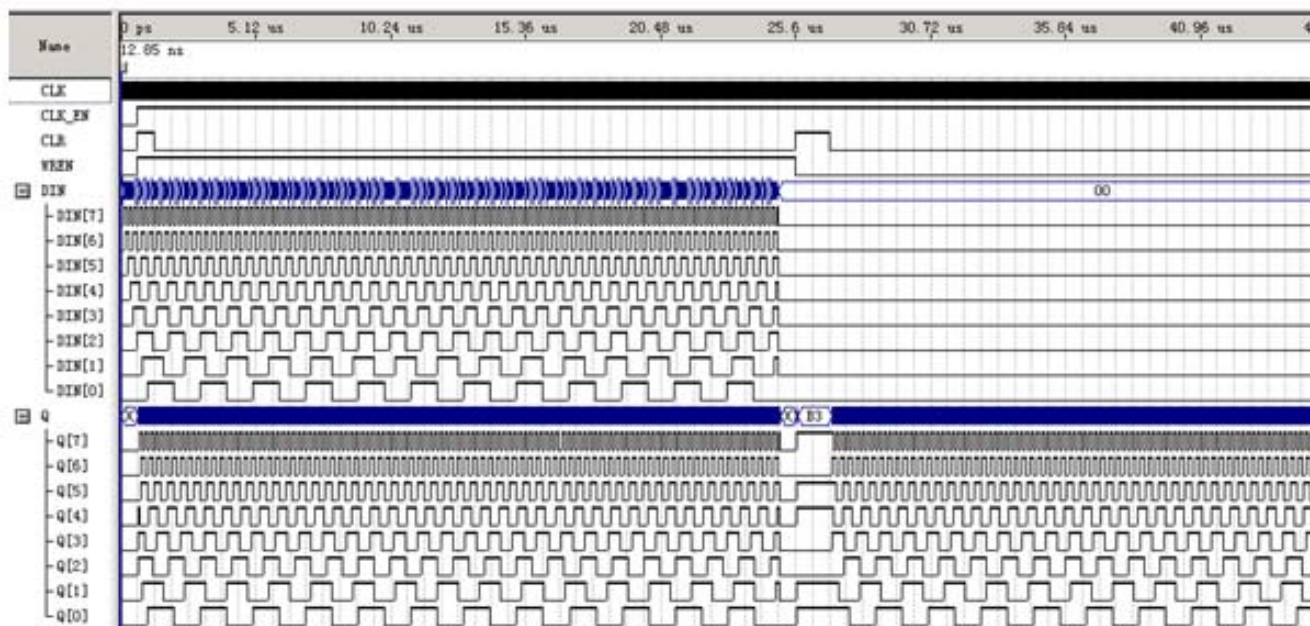


图 9-12 逻辑数据采样电路时序仿真波形

EDA实验与创新实践

9-4 SPWM脉宽调制控制系统设计

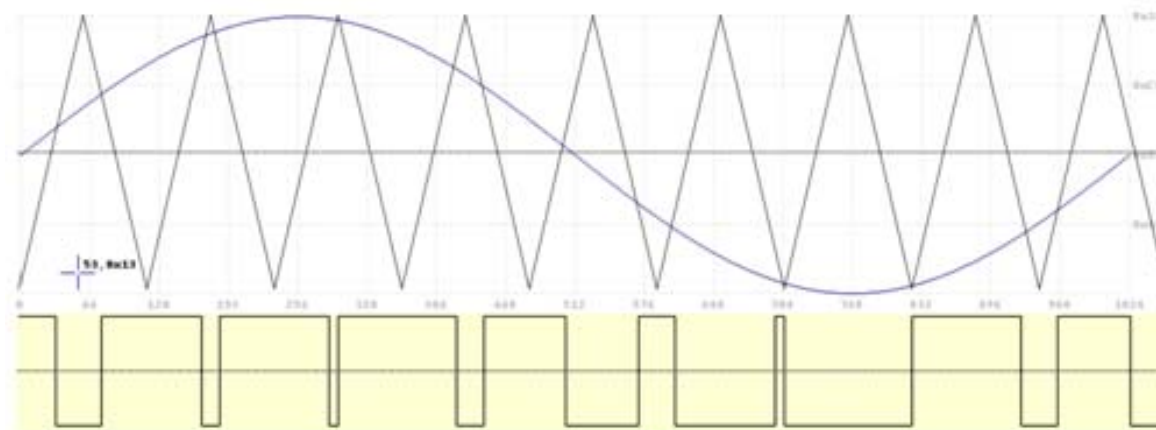


图 9-13 SPWM 波生成原理图



EDA实验与创新实践

9-4 SPWM脉宽调制控制系统设计

【例 9-15】三角波发生模块

```
module TRANG3 (input[9:0] ADR, output[9:0] OUTD );
    reg[9:0] OT1;    reg[10:0] CC;
    always @(ADR or CC)
    if (ADR<10'b1000000000) 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
    assign OUTD = OT1;    endmodule
```

EDA实验与创新实践

9-4 SPWM脉宽调制控制系统设计

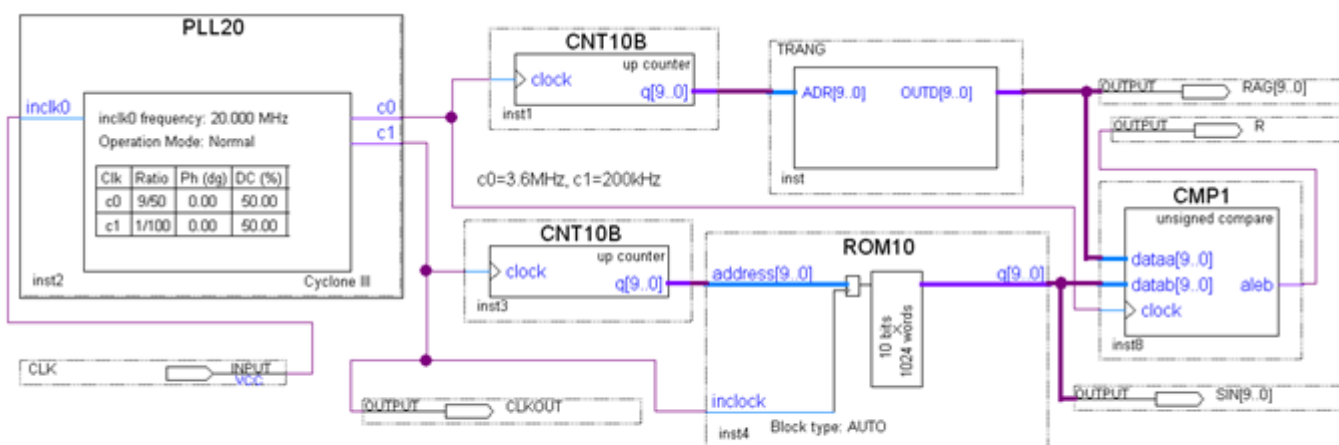


图 9-14 SPWM 波发生器基本电路图

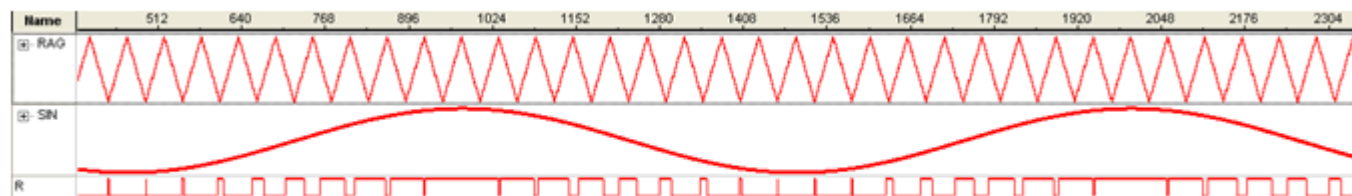


图 9-15 图 9-14 电路的 SignalTap II 实测波形

EDA实验与创新实践

9-4 SPWM脉宽调制控制系统设计

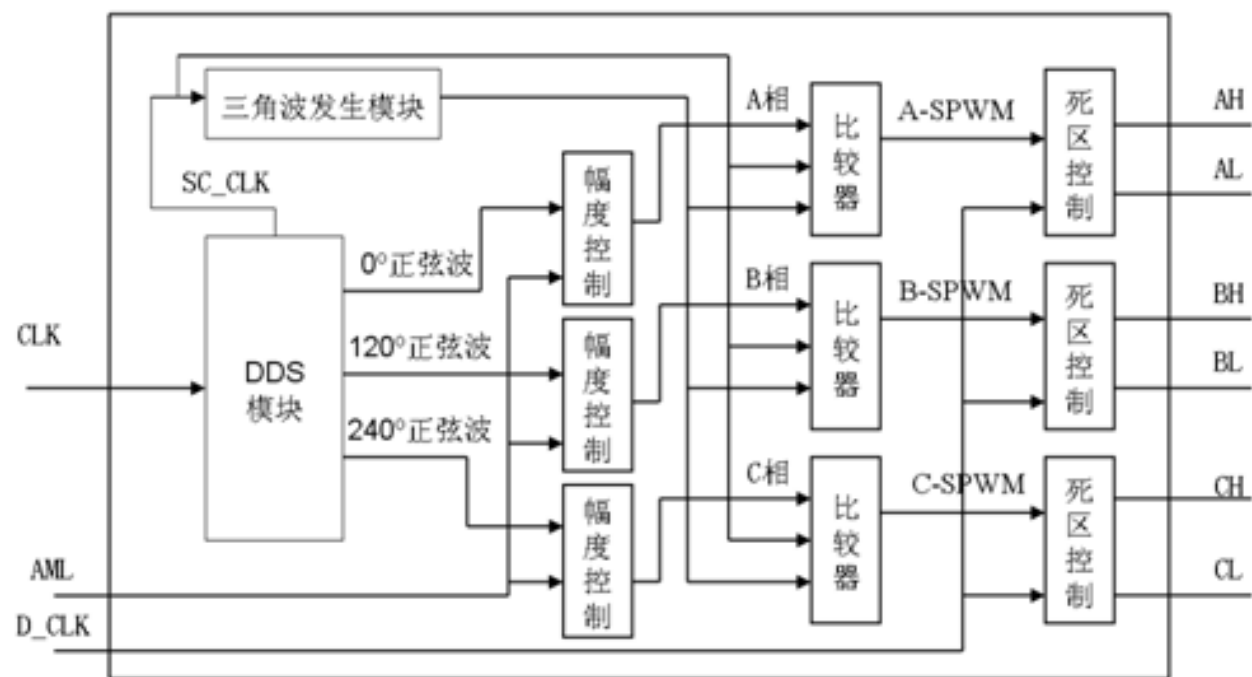


图 9-16 三相 SPWM 控制器电路模块图

EDA实验与创新实践

9-5 AM幅度调制信号发生器设计

$$F = F_{dr} \cdot (1 + F_{am} \cdot m) \quad (9-1)$$

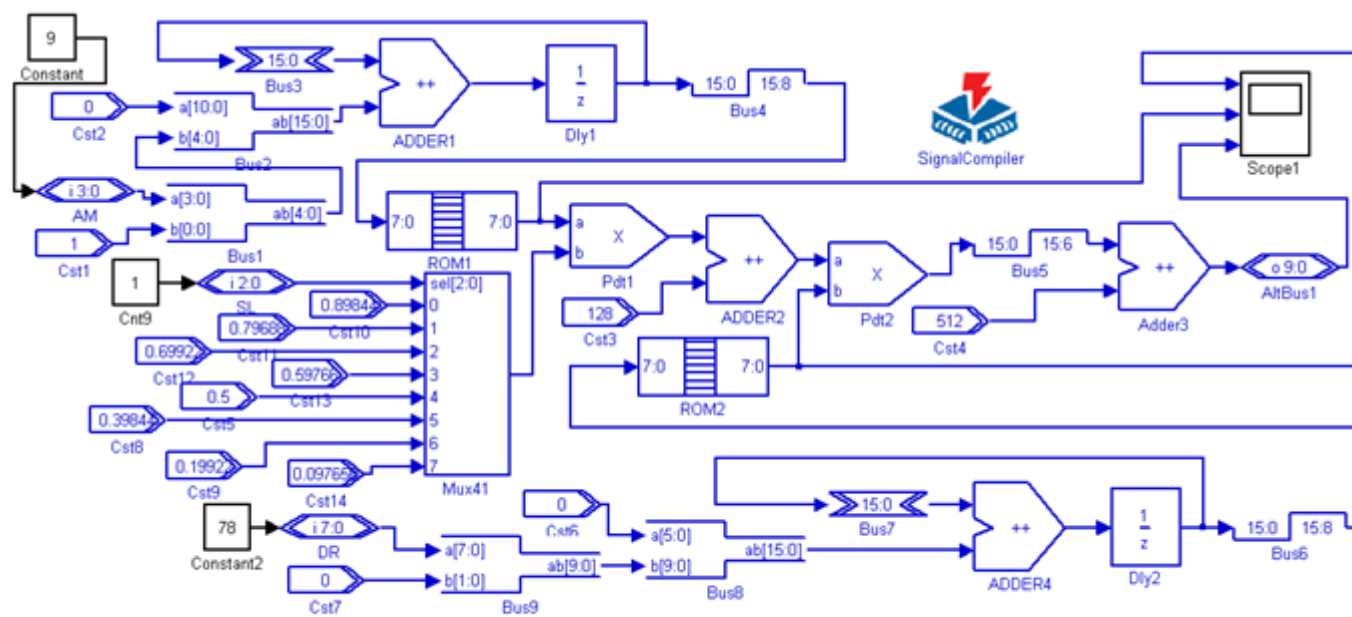


图 9-17 AM 信号发生器 DSP Builder/MATLAB Simulink 模型

EDA实验与创新实践

9-5 AM幅度调制信号发生器设计

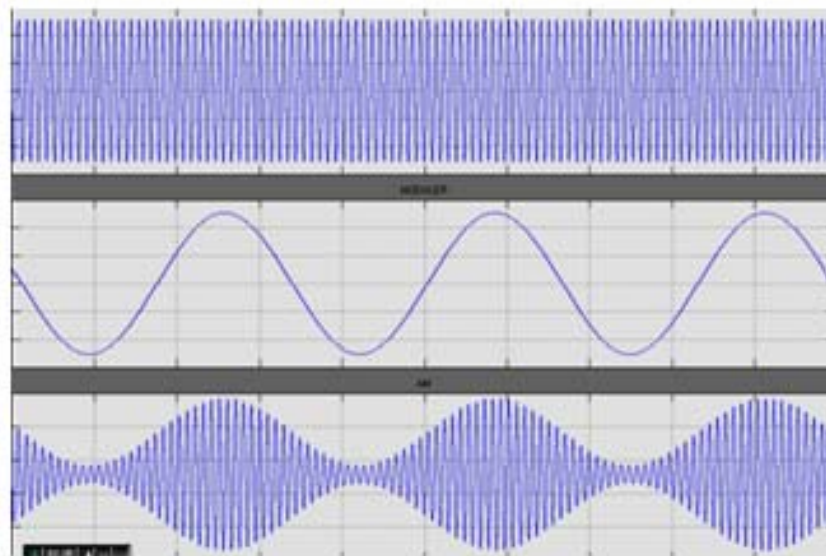


图 9-18 AM 模型中 Scope1 输出的仿真波形

EDA实验与创新实践

9-6 正交幅度调制与解调系统实现

$$I(t) = a(t) \cos \omega_0 t$$

$$Q(t) = b(t) \sin \omega_0 t$$

$$X(t) = I(t) + Q(t) = a(t) \cos \omega_0 t + b(t) \sin \omega_0 t$$

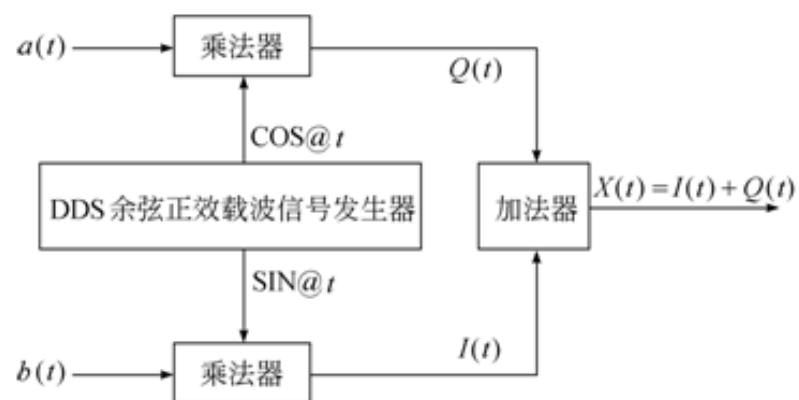


图 9-19 正交幅度调制原理图

EDA实验与创新实践

9-6 正交幅度调制与解调系统实现

$$V_a = X(t)\cos\omega_0 t = a(t)\cos^2\omega_0 t + b(t)\sin\omega_0 t\cos\omega_0 t = \frac{1}{2}a(t) + \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}b(t)\sin 2\omega_0 t$$

$$V_b = X(t)\sin\omega_0 t = b(t)\sin^2\omega_0 t + a(t)\sin\omega_0 t\cos\omega_0 t = \frac{1}{2}b(t) - \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}a(t)\sin 2\omega_0 t$$

$$V_c = \frac{K}{2}a(t)$$

$$V_d = \frac{K}{2}b(t)$$

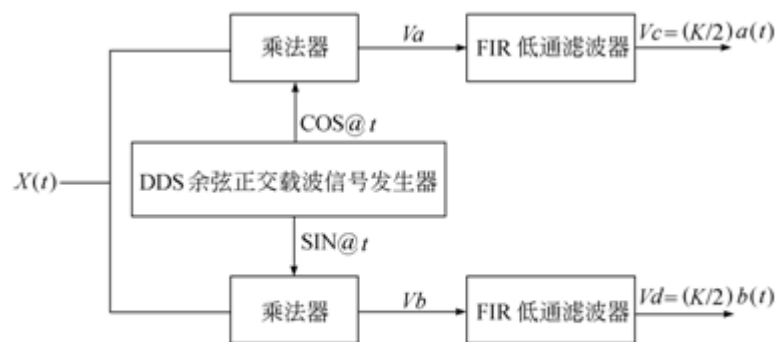


图 9-20 正交幅度信号解调原理图

EDA实验与创新实践

9-7 基于DES数据加密标准的加解密系统设计



图 9-21 DES 加解密模块示意