习 题

11.1 分别用行为描述、数据流描述和结构描述三种方式描述七种逻辑门。

解答：

（1）行为描述方式

module basic\_gates(A,B,Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor);

input A,B;

output Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor;

reg Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor;

always @ (A,B)

begin

Yand = A & B;

Yor = A | B；

Ynot = ~A;

Ynand = ~(A & B);

Ynor = ~(A | B)；

Yxor = A ^ B；

Yxnor = ~(A ^ B)；

end

endmodule

1. 数据流描述方式

module basic\_gates(A,B,Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor);

input A,B;

output Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor;

assign Yand = A & B;

assign Yor = A | B；

assign Ynot = ~A;

assign Ynand = ~(A & B);

assign Ynor = ~(A | B)；

assign Yxor = A ^ B；

assign Yxnor = ~(A ^ B)；

endmodule

1. 结构描述方式

module basic\_gates(A,B,Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor);

input A,B;

output Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor;

and U1 (Yand,A,B);

or U2 (Yor,A,B);

not U3 (Ynot,A);

nand U4 (Ynand,A,B);

nor U5 (Ynor,A,B);

xor U6 (Ynxor,A,B);

xnor U7 (Yxnor,A,B);

endmodule

11.2 分别用行为描述、数据流描述和结构描述三种方式描述74HC138。

解答：

（1）行为描述方式

module HC138(s1,s2\_n,s3\_n,a,y\_n);

input s1,s2\_n,s3\_n;

input [2:0] a;

output [7:0] y\_n;

reg [7:0] y\_n;

wire en;

assign en = s1&(~s2\_n)&(~s3\_n);

always @(en,a)

if (en)

case (a)

3'b000: y\_n = 8'b11111110;

3'b001: y\_n = 8'b11111101;

3'b010: y\_n = 8'b11111011;

3'b011: y\_n = 8'b11110111;

3'b100: y\_n = 8'b11101111;

3'b101: y\_n = 8'b11011111;

3'b110: y\_n = 8'b10111111;

3'b111: y\_n = 8'b01111111;

default: y\_n = 8'b11111111;

endcase

else

y\_n = 8'b11111111;

endmodule

1. 数据流描述方式

module HC138(s1,s2\_n,s3\_n,a,y\_n);

input s1,s2\_n,s3\_n;

input [2:0] a;

output [7:0] y\_n;

reg [7:0] y\_n;

wire en;

assign en = s1&(~s2\_n)&(~s3\_n);

assign y\_n[0]=~((~s\_n)&(~a[2])&(~a[1])&(~a[0])&en);

assign y\_n[1]=~((~s\_n)&(~a[2])&(~a[1])&a[0]&en);

assign y\_n[2]=~((~s\_n)&(~a[2])&a[1]&(~a[0])&en);

assign y\_n[3]=~((~s\_n)&(~a[2])&a[1]&a[0]&en);

assign y\_n[4]=~((~s\_n)&a[2]&(~a[1])&(~a[0]&)en);

assign y\_n[5]=~((~s\_n)&a[2]&(~a[1])&a[0]&en);

assign y\_n[6]=~((~s\_n)&a[2]&a[1]&(~a[0])&en);

assign y\_n[7]=~((~s\_n)&a[2]&a[1]&a[0]&en);

endmodule

1. 结构描述方式

module HC138(s1,s2\_n,s3\_n,a,y\_n);

input s1,s2\_n,s3\_n;

input [2:0] a;

output [7:0] y\_n;

reg [7:0] y\_n;

wire s2,s3,en;

wire na2,na1,na0;

not (s2,s2\_n); not (s3,s3\_n); and (en,s1,s2,s3);

not (na2,~a[2]); not (na1,~a[1]); not (na0,~a[0]);

nand U0 (y\_n[0], na2,na1,na0,en);

nand U1 (y\_n[1], na2,na1, a[0],en);

nand U2 (y\_n[2], na2, a[1],na0,en);

nand U3 (y\_n[3], na2, a[1], a[0],en);

nand U4 (y\_n[4], a[2],na1,na0,en);

nand U5 (y\_n[5], a[2], na1, a[0],en);

nand U6 (y\_n[6], a[2], a[1],na0,en);

nand U7 (y\_n[7], a[2], a[1], a[0],en);

endmodule

11.3 用Verilog描述三态缓冲器74HC240。

解答：

module HC240(G1\_n,A1,y1,G2\_n,A2,y2);

input G1\_n,G2\_n; // 控制端

input [3:0] A1,A2; // 数据输入端

output [3:0] y1,y2;

assign y1 = (!G1\_n)? ~A1 : 4'bz; // 第一组4位三态反相器

assign y2 = (!G2\_n)? ~A2 : 4'bz; // 第二组4位三态反相器

endmodule

11.4 用Verilog描述4位二进制同步计数器74HC161。

解答：

module HC161(CLK,Rd\_n,LD\_n,EP,ET,D,Q,CO);

input CLK;

input Rd\_n,LD\_n,EP,ET;

input [3:0] D;

output reg [3:0] Q;

output CO;

assign CO = ( (Q == 4'b1001) & ET ); // 进位逻辑

always @( posedge CLK or negedge Rd\_n ) // 计数逻辑

if (!Rd\_n)

Q <= 4'b0000;

else if (!LD\_n)

Q <= D;

else if (EP & ET)

Q <= Q + 1'b1;

endmodule

11.5 设计四位二进制计数器，按循环码的方式进行计数。

解答：

module cnt4b(clk,rst\_n,q);

input clk;

input rst\_n;

output [3:0] q;

always @(posedge clk or negedge rst\_n)

begin

if (!rst\_n)

q <= 4’b0000;

else

case (q)

4’b0000: q<= 4’b0001;

4’b0001: q<= 4’b0011;

4’b0011: q<= 4’b0010;

4’b0010: q<= 4’b0100;

4’b0100: q<= 4’b1001;

4’b0101: q<= 4’b0111;

4’b0111: q<= 4’b0110;

4’b0110: q<= 4’b1000;

4’b1000: q<= 4’b1001;

4’b1001: q<= 4’b1011;

4’b1011: q<= 4’b1010;

4’b1010: q<= 4’b1100;

4’b1100: q<= 4’b1101;

4’b1101: q<= 4’b1111;

4’b1111: q<= 4’b1110;

4’b1110: q<= 4’b0000;

default: q<= 4’b0000;

endcase

end

endmodule

11.6 设计十六进制显示译码器。对于输入四位二进制数“0000~1111”，在数码管显示0~9、A、B、C、D、E、F字符。

解答：

module Hex\_Disp\_Decoder(BIN,SEG7);

input [3:0] BIN;

output reg [6:0] SEG7;

always @(BIN)

case (BIN) // SEG7: gfedcba,低电平有效

4'b0000: SEG7 = 7'b1000000; // 显示 0

4'b0001: SEG7 = 7'b1111001; // 显示 1

4'b0010: SEG7 = 7'b0100100; // 显示 2

4'b0011: SEG7 = 7'b0110000; // 显示 3

4'b0100: SEG7 = 7'b0011001; // 显示 4

4'b0101: SEG7 = 7'b0010010; // 显示 5

4'b0110: SEG7 = 7'b0000010; // 显示 6

4'b0111: SEG7 = 7'b1111000; // 显示 7

4'b1000: SEG7 = 7'b0000000; // 显示 8

4'b1001: SEG7 = 7'b0010000; // 显示 9

4'b1010: SEG7 = 7'b0010000; // 显示 a

4'b1011: SEG7 = 7'b0010000; // 显示 b

4'b1100: SEG7 = 7'b0000100; // 显示 c

4'b1101: SEG7 = 7'b0100001; // 显示 d

4'b1110: SEG7 = 7'b0000110; // 显示 e

4'b1111: SEG7 = 7'b0001110; // 显示 f

default: SEG7 = 7'b1111111; // 不显示

endcase

endmodule

11.7 用Verilog描述8位加法器，能够实现二个8位二进制相加，输出加法和与进位信号。

解答：

module adder\_8bits(a,b,cin,sum,cout);

parameter Nbits = 8;

input [Nbits-1:0] a, b;

input cin;

output [Nbits-1:0] sum;

output cout;

assign {cout,sum} = a + b + cin;

endmodule

11.8 按表7-3所示真值表，用Verilog描述设计四位二进制乘法器。

解答：

module multiplier\_4x4b #（parameter Nbits = 4）(a,b,result);

input [Nbits-1:0] a, b;

output [2\*Nbits-1:0] result;

assign result = a \* b ;

endmodule

11.9 用Verilog描述8位二进制乘法器，能够实现两个8位无符号数相乘，输出16位乘法结果。

解答：

module multiplier\_8x8b #（parameter Nbits = 8）(a,b,result);

input [Nbits-1:0] a, b;

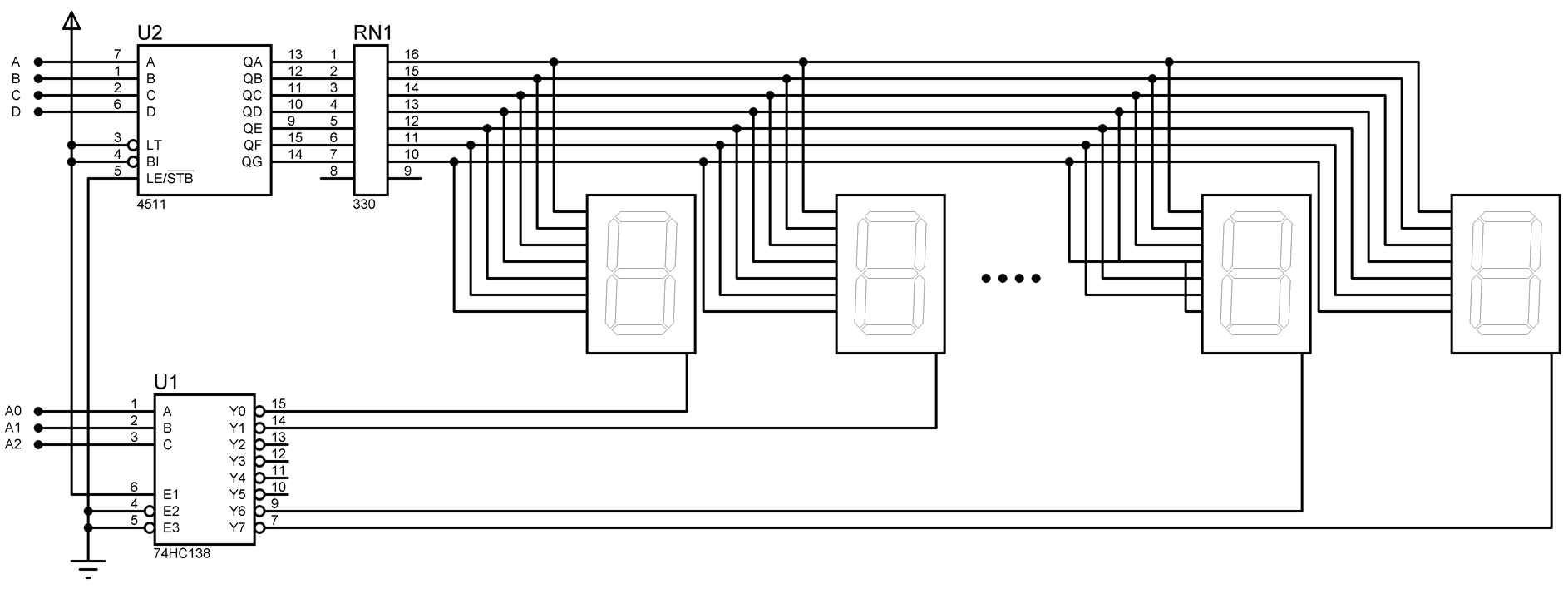
output [2\*Nbits-1:0] result;

assign result = a \* b ;

endmodule

11.10 按7.5.2节LED点阵驱动电路的设计原理，采用动态扫描方法设计8个数码管驱动电路，分别在8个数据管上显示由四位二进制输入*D*0、*D*1、...、*D*7定义的字符信息。

解答：采用动态扫描方法驱动8个数码管的参考电路设计方案如下图所示。在8进制计数器的作用下，分别选择从*D*0、*D*1、...、*D*7选择数据后由CD4511译码输出，由74HC138选通数码管显示。



能够驱动8个数码管分别显示由8个BCD码*D*0、*D*1、...、*D*7定义的数字信息的Verilog参考代码如下：

module Dynamic\_Scan\_Disp\_Decoder(clk,D0,D1,D2,D3,D4,D5,D6,D7,SEG,Y\_n)

input clk;

input [3:0] D0,D1,D2,D3,D4,D5,D6,D7;

output reg [6:0] SEG7;

output reg [0:7] y\_n;

reg [2:0] Qtmp;

reg [2:0] Disp\_DATA;

always @(posedge clk)

Qtmp <= Qtmp +1’b1;

always @(Qtmp)

case (Qtmp)

4'b0000: begin Disp\_DATA = D0; y\_n = 8’b01111111; end

4'b0000: begin Disp\_DATA = D1; y\_n = 8’b10111111; end

4'b0000: begin Disp\_DATA = D2; y\_n = 8’b11011111; end

4'b0000: begin Disp\_DATA = D3; y\_n = 8’b11101111; end

4'b0000: begin Disp\_DATA = D4; y\_n = 8’b11110111; end

4'b0000: begin Disp\_DATA = D5; y\_n = 8’b11111011; end

4'b0000: begin Disp\_DATA = D6; y\_n = 8’b11111101; end

4'b0000: begin Disp\_DATA = D7; y\_n = 8’b11111110; end

default: begin Disp\_DATA = D0; y\_n = 8’b11111111; end

endcase

always @(Disp\_DATA)

case(Disp\_DATA)

4'b0000: SEG7 = 7'b1000000; // 显示 0

4'b0001: SEG7 = 7'b1111001; // 显示 1

4'b0010: SEG7 = 7'b0100100; // 显示 2

4'b0011: SEG7 = 7'b0110000; // 显示 3

4'b0100: SEG7 = 7'b0011001; // 显示 4

4'b0101: SEG7 = 7'b0010010; // 显示 5

4'b0110: SEG7 = 7'b0000010; // 显示 6

4'b0111: SEG7 = 7'b1111000; // 显示 7

4'b1000: SEG7 = 7'b0000000; // 显示 8

4'b1001: SEG7 = 7'b0010000; // 显示 9

default: SEG7 = 7'b1111111; // 不显示

endcase

endmodule

11.11\* 按7.5.2节LED点阵驱动电路的设计原理，采用动态扫描方法设计16\*16点阵驱动电路，显示不少于8个16×16字符或图像。

解答：以下代码只显示一个16×16字符或图像，具体显示的字符或图像由D0,D1,...,D15确定，参考代码如下：

module Dynamic\_Scan\_Dot\_Driver(clk,Lin\_DATA,Y)

input clk;

output reg [15:0] Lin\_DATA;

output reg [0:15] Y\_n;

reg [3:0] Qtmp;

always @(posedge clk)

Qtmp <= Qtmp +1’b1;

always @(Qtmp)

case (Qtmp)

4'b0000: begin Lin\_DATA = D0; y\_n =16’h7fff; end

4'b0001: begin Lin\_DATA = D1; y\_n =16’hbfff; end

4'b0010: begin Lin\_DATA = D2; y\_n =16’hdfff; end

4'b0011: begin Lin\_DATA = D3; y\_n =16’hefff; end

4'b0100: begin Lin\_DATA = D4; y\_n =16’hf7ff; end

4'b0101: begin Lin\_DATA = D5; y\_n =16’hfbff; end

4'b0110: begin Lin\_DATA = D6; y\_n =16’hfdff; end

4'b0111: begin Lin\_DATA = D7; y\_n =16’hfeff; end

4'b1000: begin Lin\_DATA = D8; y\_n =16’hff7f; end

4'b1001: begin Lin\_DATA = D9; y\_n =16’hffbf; end

4'b1010: begin Lin\_DATA = D10; y\_n =16’hfdff; end

4'b1011: begin Lin\_DATA = D11; y\_n =16’hffef; end

4'b1100: begin Lin\_DATA = D12; y\_n =16’hfff7; end

4'b1101: begin Lin\_DATA = D13; y\_n =16’hfffb; end

4'b1110: begin Lin\_DATA = D14; y\_n =16’hfffd; end

4'b1111: begin Lin\_DATA = D15; y\_n =16’hfffe; end

default: begin Lin\_DATA = D0; y\_n =16’hffff; end

endcase

endmodule

需要显示8个字符或图像时，可添加3位输入信号A2、A1和A0，由A2、A1和A0切换存储D0,D1,...,D15的ROM数据区。具体方法由读者自行扩展。

11.12 用Verilog描述能够产生“1101000101”序列信号的序列信号产生器。

解答：

由于序列信号的长度为10，因此应用10进制计数器驱动10选一数据选择器实现。

Verilog HDL参考代码如下：

module serial\_gen(clk,y);

input clk;

output reg y;

reg [3:0] Qtmp;

always @(posedge clk)

if (q==4’b1001)

Qtmp <= 4’b0000;

else

Qtmp <= Qtmp + 1’b1;

always @(Qtmp)

case (Qtmp)

4’b0000: y = 1’b1;

4’b0001: y = 1’b1;

4’b0010: y = 1’b0;

4’b0011: y = 1’b1;

4’b0100: y = 1’b0;

4’b0101: y = 1’b0;

4’b0110: y = 1’b0;

4’b0111: y = 1’b1;

4’b1000: y = 1’b0;

4’b1001: y = 1’b1;

default: y = 1’b0;

endcase

endmodule