实验三 层次结构设计方法及应用

**一、实验目的**

1. 进一步掌握verilog中的基本语法和语句。

**二、实验内容**

1. 掌握always语句的含义和用法。
2. 完成32位半加器、全加器模块的设计。
3. 设计一个基本的32位算术逻辑运算（ALU）模块。
4. 观察记录分析仿真波形。

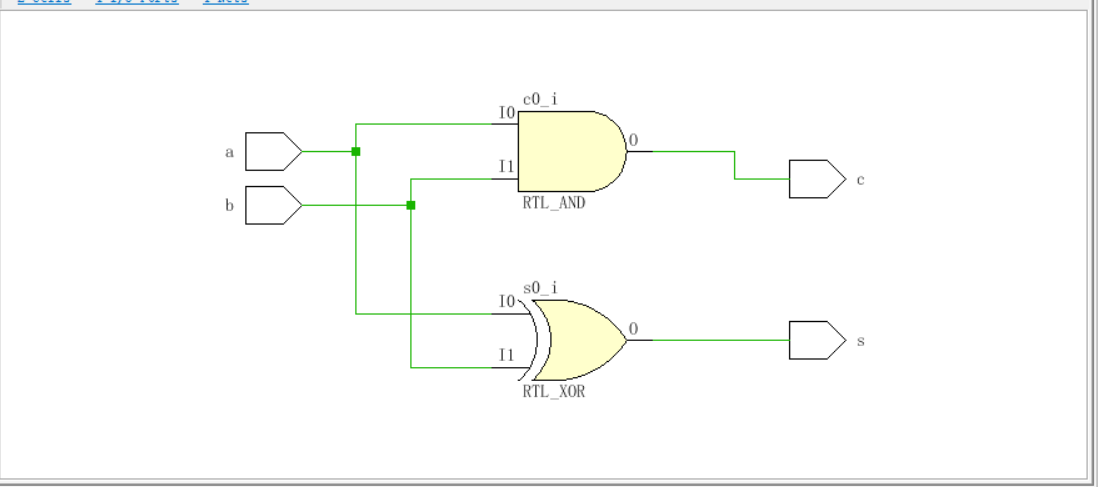
**三、实验要求**

1. 画出模块的电路图。
2. 分析电路的仿真波形，标出关键的数值。
3. 记录设计和调试过程。

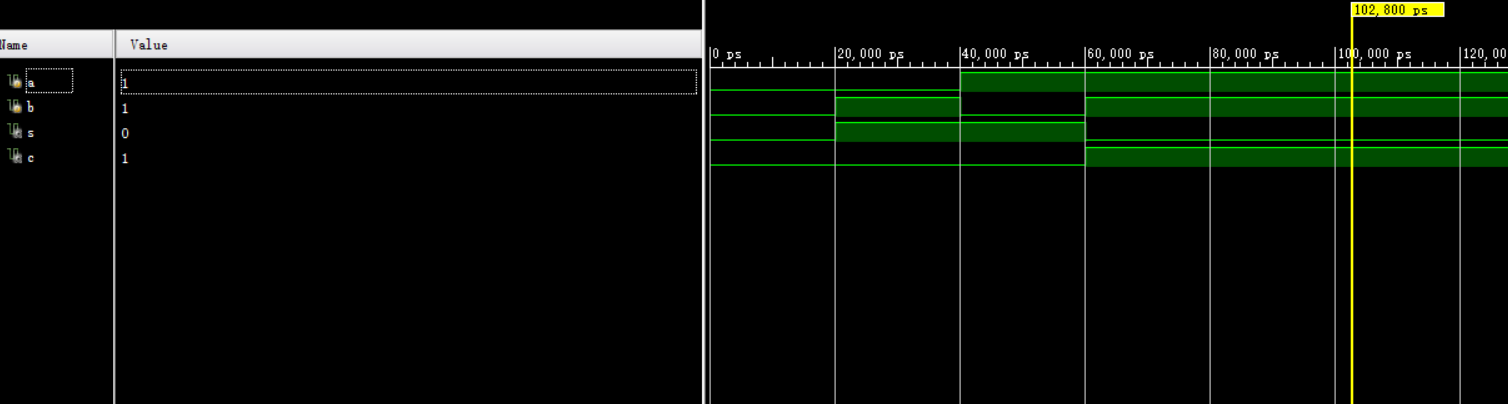
**四、实验代码及结果**

**1位半加器：**

**电路图：**



**仿真图：**



**器件源代码：**

module h\_a1(

s,c,a,b

);

input a,b;

output s,c;

reg s,c;

always@(\*) s=a^b;

always@(\*) c=a&b;

endmodule

**仿真源代码：**

module sim\_ha1();

reg a,b;

wire s,c;

initial

begin

a=0;

b=0;

#20

a=0;

b=1;

#20

a=1;

b=0;

#20

a=1;

b=1;

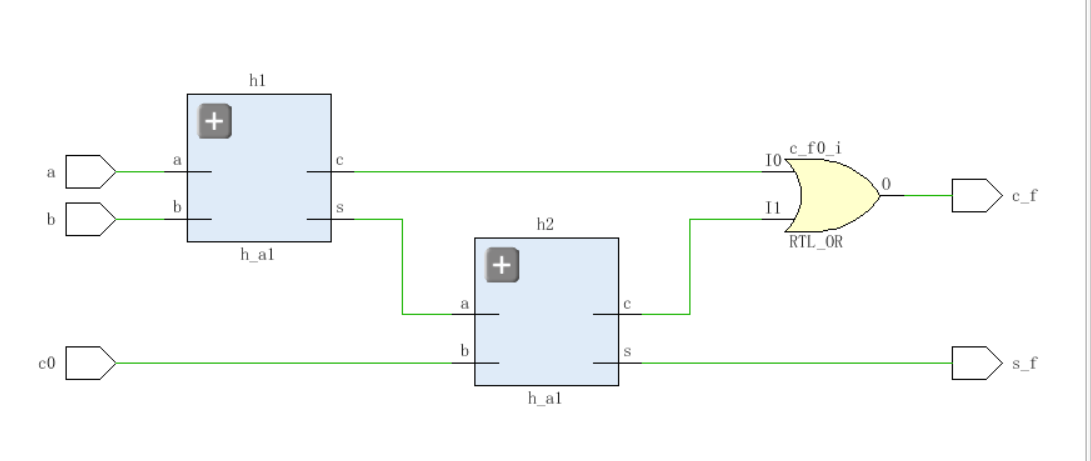
end

h\_a1 aaa(s,c,a,b);

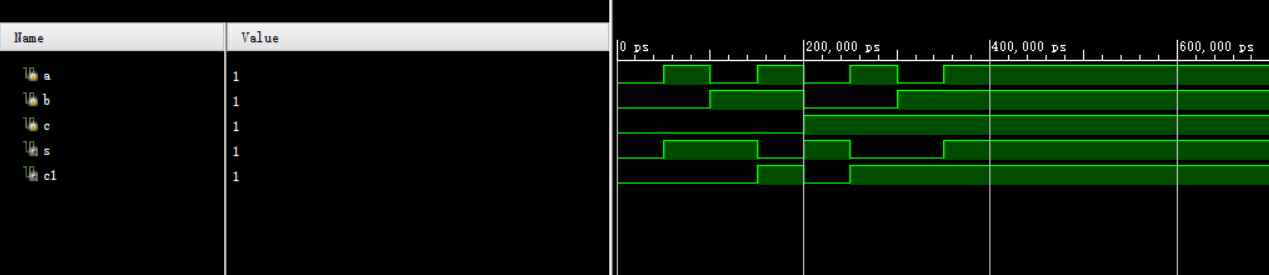
endmodule

**一位全加器：**

**电路图：**



**仿真图：**



**器件源代码：**

module a1(

s\_f,c\_f,a,b,c0

);

input a,b,c0;

output s\_f,c\_f;

reg c\_f;

wire c\_t1,c\_t2,s;

h\_a1 h1(s,c\_t1,a,b);

h\_a1 h2(s\_f,c\_t2,s,c0);

always@(\*) c\_f=c\_t1|c\_t2;

endmodule

**仿真源代码：**

module sim\_a1( );

reg a,b,c;

wire s,c1;

initial

begin

a=0;

b=0;

c=0;

#50

a=1;

b=0;

c=0;

#50

a=0;

b=1;

c=0;

#50

a=1;

b=1;

c=0;

#50

a=0;

b=0;

c=1;

#50

a=1;

b=0;

c=1;

#50

a=0;

b=1;

c=1;

#50

a=1;

b=1;

c=1;

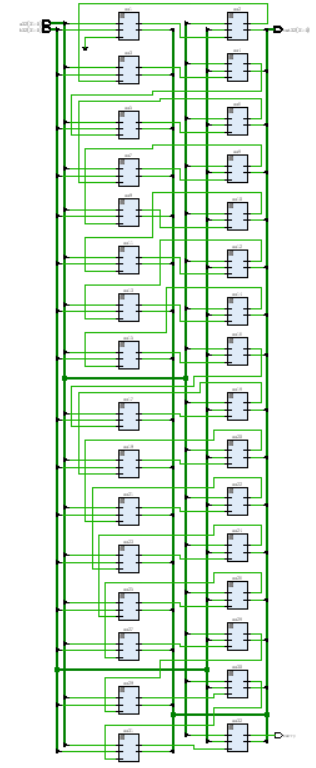
end

a1 aaaa(s,c1,a,b,c);

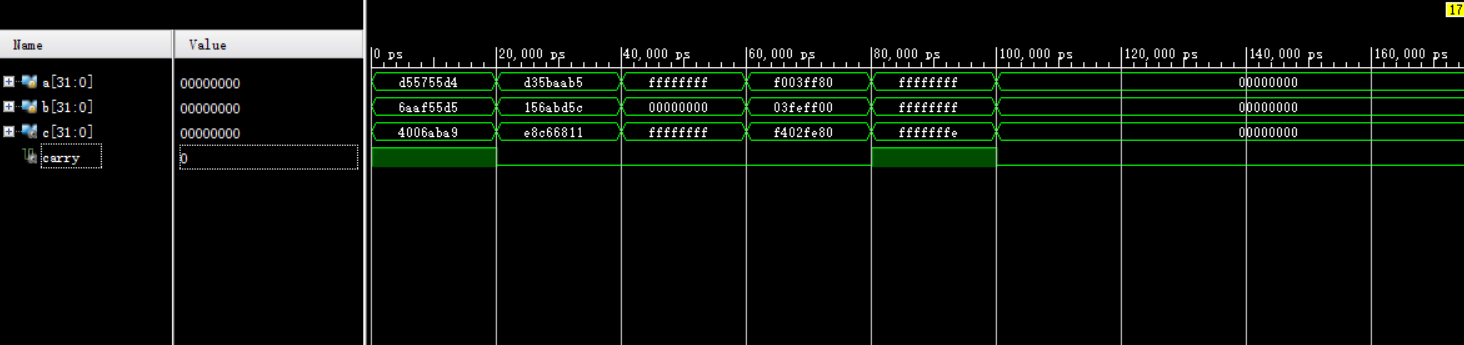
endmodule

**32位全加器：**

**电路图：**



**仿真图：**



**器件源代码：**

module a32(

out32,carry,a32,b32

);

input [31:0]a32,b32;

output [31:0]out32;

output reg carry;

wire [31:0]c;

wire c0=0;

a1 aa1(out32[0],c[0],a32[0],b32[0],c0);

a1 aa2(out32[1],c[1],a32[1],b32[1],c[0]);

a1 aa3(out32[2],c[2],a32[2],b32[2],c[1]);

a1 aa4(out32[3],c[3],a32[3],b32[3],c[2]);

a1 aa5(out32[4],c[4],a32[4],b32[4],c[3]);

a1 aa6(out32[5],c[5],a32[5],b32[5],c[4]);

a1 aa7(out32[6],c[6],a32[6],b32[6],c[5]);

a1 aa8(out32[7],c[7],a32[7],b32[7],c[6]);

a1 aa9(out32[8],c[8],a32[8],b32[8],c[7]);

a1 aa10(out32[9],c[9],a32[9],b32[9],c[8]);

a1 aa11(out32[10],c[10],a32[10],b32[10],c[9]);

a1 aa12(out32[11],c[11],a32[11],b32[11],c[10]);

a1 aa13(out32[12],c[12],a32[12],b32[12],c[11]);

a1 aa14(out32[13],c[13],a32[13],b32[13],c[12]);

a1 aa15(out32[14],c[14],a32[14],b32[14],c[13]);

a1 aa16(out32[15],c[15],a32[15],b32[15],c[14]);

a1 aa17(out32[16],c[16],a32[16],b32[16],c[15]);

a1 aa18(out32[17],c[17],a32[17],b32[17],c[16]);

a1 aa19(out32[18],c[18],a32[18],b32[18],c[17]);

a1 aa20(out32[19],c[19],a32[19],b32[19],c[18]);

a1 aa21(out32[20],c[20],a32[20],b32[20],c[19]);

a1 aa22(out32[21],c[21],a32[21],b32[21],c[20]);

a1 aa23(out32[22],c[22],a32[22],b32[22],c[21]);

a1 aa24(out32[23],c[23],a32[23],b32[23],c[22]);

a1 aa25(out32[24],c[24],a32[24],b32[24],c[23]);

a1 aa26(out32[25],c[25],a32[25],b32[25],c[24]);

a1 aa27(out32[26],c[26],a32[26],b32[26],c[25]);

a1 aa28(out32[27],c[27],a32[27],b32[27],c[26]);

a1 aa29(out32[28],c[28],a32[28],b32[28],c[27]);

a1 aa30(out32[29],c[29],a32[29],b32[29],c[28]);

a1 aa31(out32[30],c[30],a32[30],b32[30],c[29]);

a1 aa32(out32[31],c[31],a32[31],b32[31],c[30]);

always@(\*) carry=c[31];

endmodule

**仿真源代码：**

module sim\_a32();

reg[31:0] a,b;

wire[31:0]c;

wire carry;

initial

begin

a=32'b11010101010101110101010111010100;

b=32'b01101010101011110101010111010101;

#20

a=32'b11010011010110111010101010110101;

b=32'b00010101011010101011110101011100;

#20

a=32'b11111111111111111111111111111111;

b=32'b00000000000000000000000000000000;

#20

a=32'b11110000000000111111111110000000;

b=32'b00000011111111101111111100000000;

#20

a=32'b11111111111111111111111111111111;

b=32'b11111111111111111111111111111111;

#20

a=32'b00000000000000000000000000000000;

b=32'b00000000000000000000000000000000;

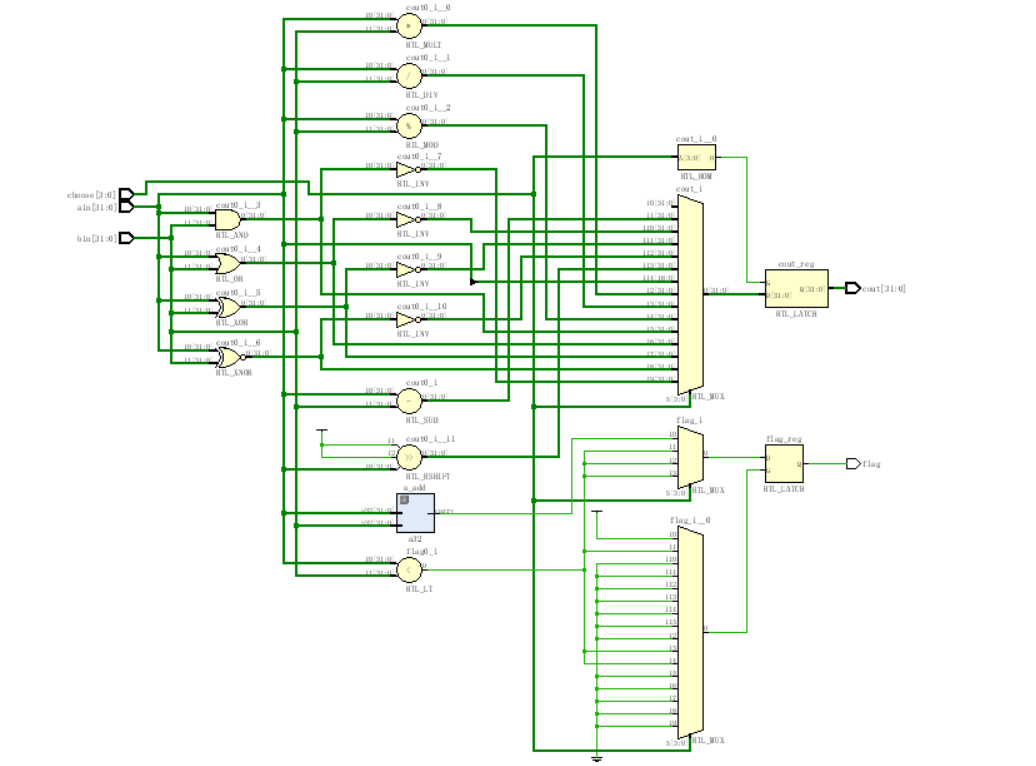
end

a32 a1(c,carry,a,b);

endmodule

**alu:**

**电路图：**



**源代码：**

module alu\_32(

ain,bin,cout,choose,flag

);

input [31:0] ain,bin;

input [3:0]choose;

output reg [31:0] cout;

output reg flag;

initial flag=0;

wire [31:0]invari;

wire overflow;

a32 a\_add(invari,overflow,ain,bin);

always@(\*)

begin

case(choose)

4'b0000:

begin

cout=invari;

flag=overflow;

end

4'b0001:

begin

cout=ain-bin;

if(ain<bin)

flag=1;

end

4'b0010:

begin

cout=ain\*bin;

if((ain\*bin)>32'b11111111111111111111111111111111)

flag=1;

end

4'b0011:

begin

cout=ain/bin;

if(bin==32'b0)

flag=1;

end

4'b0100:

begin

cout=ain%bin;

if(ain<bin)

flag=1;

end

4'b0110:cout=ain&bin;

4'b0111:cout=ain|bin;

4'b1000:cout=ain^bin;

4'b1001:cout=ain^~bin;

4'b1010:cout=~(ain&bin);

4'b1011:cout=~(ain|bin);

4'b1100:cout=~(ain^bin);

4'b1101:cout=~(ain^~bin);

4'b1110:cout=ain>>2;

4'b1111:cout={ain[15:0],ain[16]};

endcase

end

endmodule

**仿真图：**



**仿真源代码：**

module sim\_alu32();

reg [31:0]ain;

reg [31:0]bin;

reg [3:0]choose;

wire flag;

wire [31:0]cout;

reg i;

initial

begin

ain=32'b00000000000000000000000000001001;

bin=32'b00000000000000000000000000000011;

choose=4'b0000;

for(i=0;i<=15;i=i+1)

begin

#20

choose=choose+4'b1;

end

end

alu\_32 alu1(ain,bin,cout,choose,flag);

endmodule

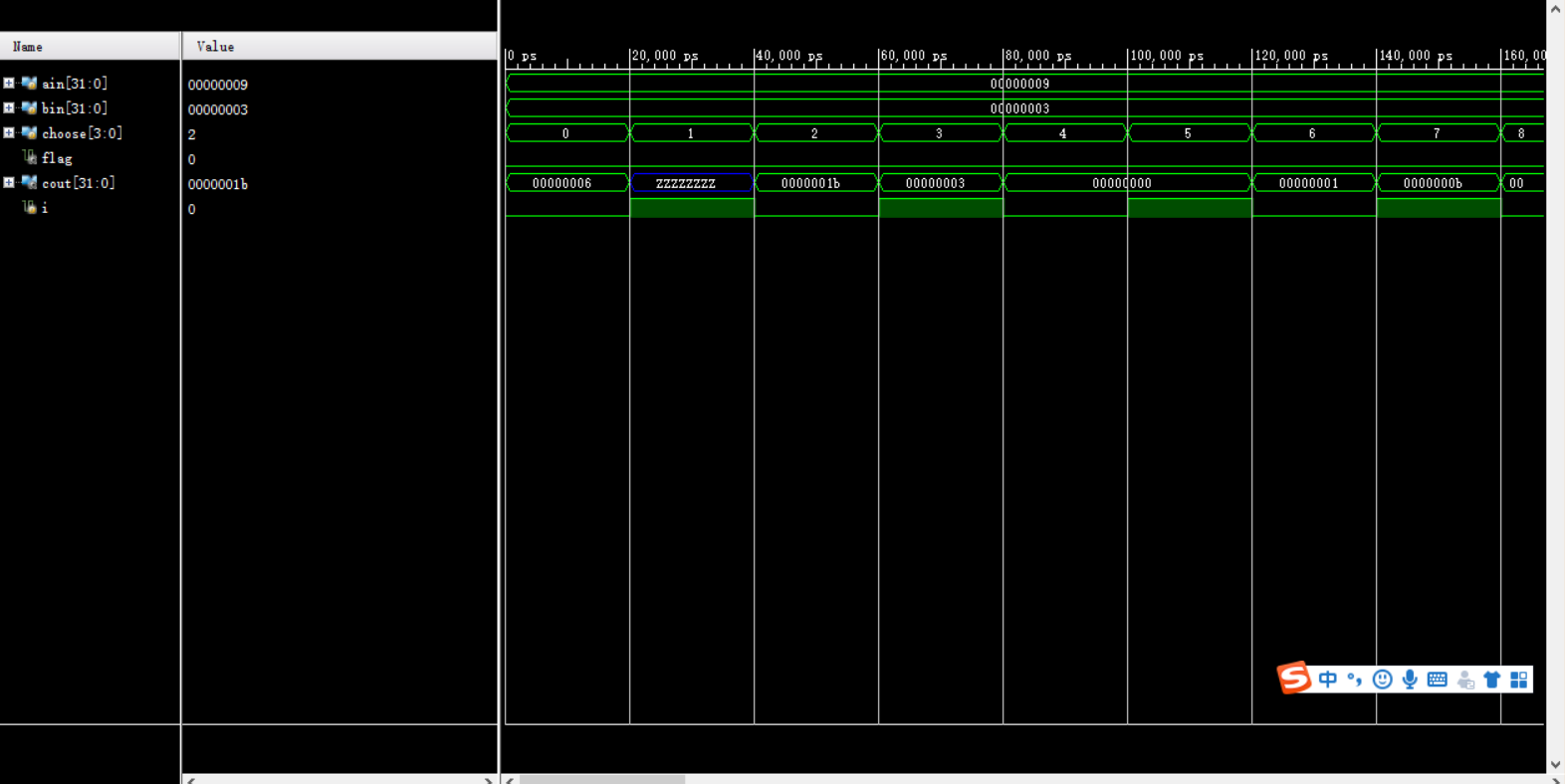
**五、调试和心得体会**

做最后一个alu的时候，我发现一个问题，即第一个状态的加法器输出为x，后面的都

正常。我最开始以为是代码问题，在机房调试了许久，但第一个加法器输出仍然为未知，且我已经单独验证的加法器模块，最后连两者实验用例都调成了相同，可在alu中还是无法正常显示结果。

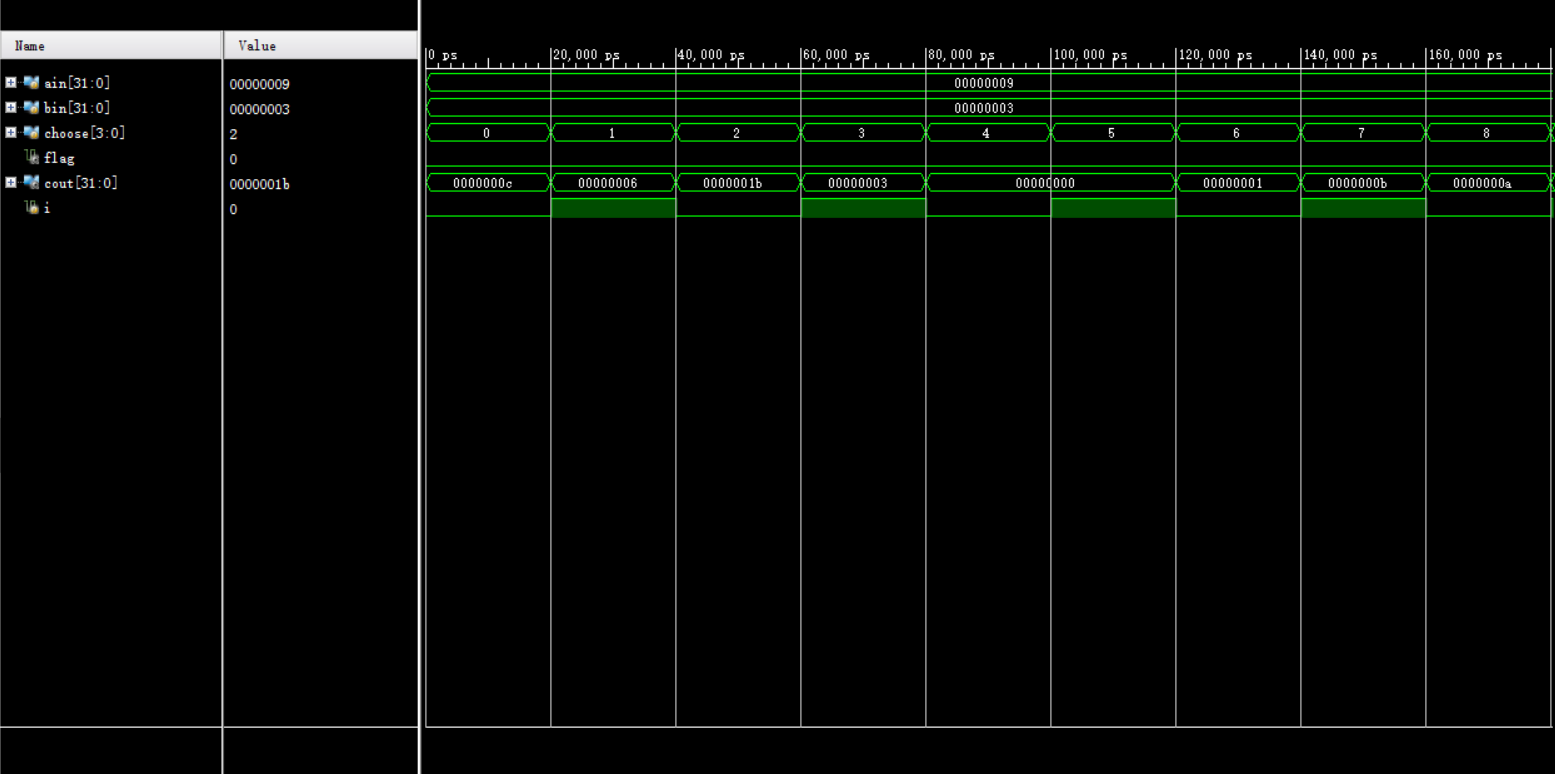
为了正常解决这个问题，我将alu的代码中的状态对应下的运算顺序进行交换；

上面仿真图0000状态为加法，0001为减法。我将0000改为减法，0001改为加法。仿真后得到的结果如下图所示：



可以看到，减法仍然正常，加法仍然出现问题。很明确是实例化adder出了问题。

尝试了很多种改法，突然，加法变正常了；



经过检查，发现居然实例化拼错了。创建了一个32位的wire值，命名为invari，然而，实例化加法器引用为输出时却打成了invart……

这次让我深刻意识到，小错误是多么致命，自己调bug的能力还是太差，耽误了很多时间，这种低等错误下次不会在发生了。