同济大学计算机系

数字逻辑课程实验报告

学　　号 2252941

姓　　名 杨瑞灵

专　　业 计算机科学与技术

授课老师 张冬冬

一、实验内容

实验介绍

在本次实验中，我们将使用 Verilog HDL 语言实现行为级 ALU 的设计和仿真

实验目标

深入了解 ALU 的原理学习使用 Verilog HDL 语言进行行为级 ALU 的设计与仿真。 3.实验原理 ALU 是负责运算的电路。ALU 必须实现以下几个运算：加（ADD）、减（SUB）、与（AND）、 或（OR）、异或（XOR）、置高位立即数（LUI）、逻辑左移与算数左移（SLL）、逻辑右移（SRL） 以及算数右移（SRA）、 SLT、SLTU 等操作。输出 32 位计算结果、carry(借位进位标志 位)、zero(零标志位)、negative(负数标志位)和 overflow(溢出标志位)。 本实验实现 ALU 的基本思想是：在操作数输入之后将所有可能的结果都计算出来，通 过操作符 aluc 的输入来判别需要执行的操作来选择需要的结果进行输出。图 6.9.1 所示 为本实验的 ALU 参考原理图。表 6.9.1 所示为 aluc 的值所对应的运算。表 6.9.2 所示 为 addsub32 标志位规则（仅供参考）

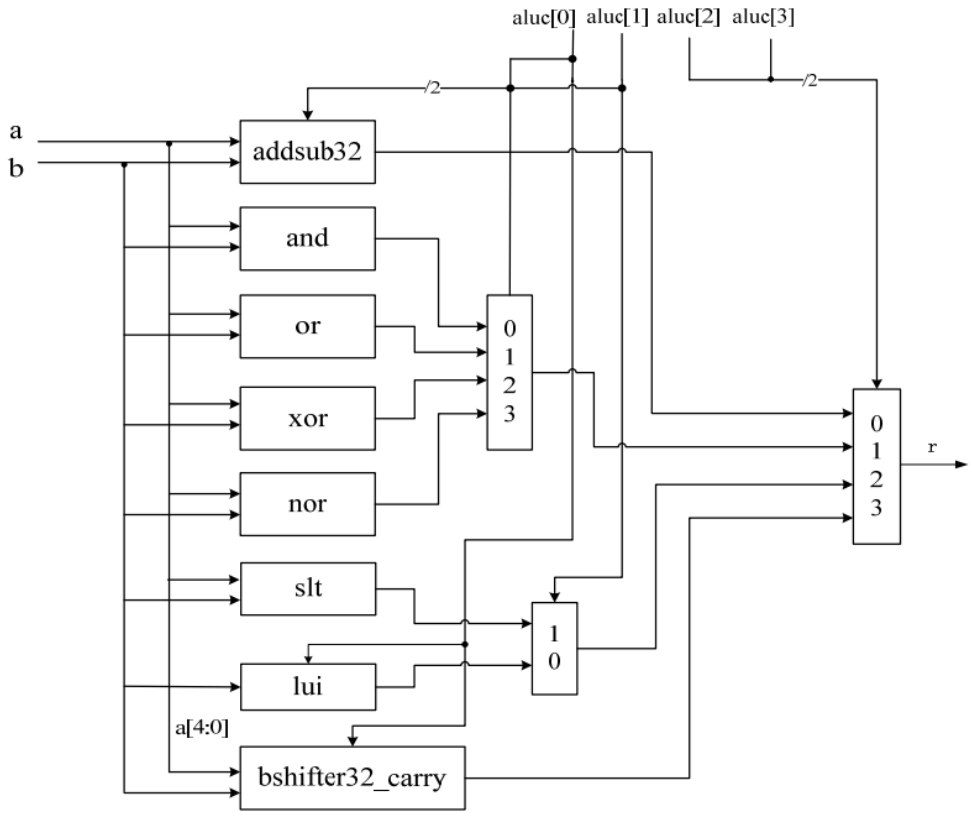


图 6.9.1 ALU 的原理图供参考

表 6.9.1　aluc 的值所对应的运算

	aluc[3]	aluc[2]	aluc[1]	aluc[0]
Addu　r=a+b 无符号	0	0	0	0
Add　r=a+b 有符号	0	0	1	0
Subu　r=a-b 无符号	0	0	0	1
Sub　r=a-b 有符号	0	0	1	1
And　r=a & b	0	1	0	0
Or　r=a | b	0	1	0	1
Xor　r=a ^ b	0	1	1	0
Nor　r=~（a | b）	0	1	1	1
Lui　r={b[15:0],16'b0}	1	0	0	X
Slt　r=(a<b)?1:0 有符号	1	0	1	1
Sltu　r=(a<b)?1:0 无符号	1	0	1	0
Sra　r=b>>>a	1	1	0	0
Sll/Slr　r=b<<a	1	1	1	X
Srl　r=b>>a	1	1	0	1

表 6.9.2　ALU 标志位规则

zero 标志位	1.Z＝1 表示运算结果是零，Z＝0 表示运算结果不是零。 2.对于 Slt 和 Sltu 运算，如 a-b=0，则 Z＝1，表示进行比较的两个数大小相等。 3.所有运算均影响此标志位。
carry 标志位	1.无符号数加法运算（Addu）发生上溢出，则该标志位为 1。 2.无符号数减法运算（Subu）发生下溢出，则该标志位为 1。 3.无符号数比较运算（Sltu），如 a-b<0，则该标志位为 1。 4.移位运算，该标志位为最后一次被移出的位的数值(在移位模块实现)。 5.其他运算不影响此标志位。
negative 标志位	1.有符号数运算 Add 和 Sub，操作数和运算结果均采用二进制补码的形式表示，N＝1 表示运算的结果为负数，N＝0 表示结果为正数或零。 2.有符号数比较运算（Slt），如果 a-b<0，则 N=1。 3.其他运算，运算最终结果的最高位 r[31]为 1，则 N=1。
overflow 标志位	1.对于有符号加减法运算（Add 和 Sub），操作数和运算结果均采用二进制补码的形式表示，有溢出时该标志位 o=1。 2.只有有符号加减法运算影响此标志位。

二、硬件逻辑图

三、模块建模

6.9

```
module alu(
    input [31:0] a,
    input [31:0] b,
    input [3:0] aluc,
    output reg [31:0] r,
    output reg zero,
    output reg carry,
    output reg negative,
    output reg overflow
    );
    wire signed [31:0]a1;
    wire signed [31:0]b1;
    assign a1 = a;
    assign b1 = b;
    //无符号加法
    wire [31:0] Addu;
    assign Addu = a + b;
    //有符号加法
    wire [31:0] Add;
    assign Add = a1 + b1;
    //无符号减法
    wire [31:0] Subu;
    assign Subu = a - b;
    //有符号减法
    wire [31:0] Sub;
    assign Sub = a1 - b1;
    //按位与
    wire [31:0] And;
    assign And = a & b;
    //按位或
    wire [31:0] Or;
    assign Or = a | b;
    //按位异或
    wire [31:0] Xor;
    assign Xor = a ^ b;
```

```
//按位或非
wire [31:0] Nor;
assign Nor = ~(a | b);
//置高位立即数（LUI）
wire [31:0] Lui;
assign Lui = (b & 32'h0000ffff) << 16;
//有符号的 a<b?    Slt
wire [31:0] Slt;
assign Slt = (a1 < b1)?1:0;
//无符号的 a<b?
wire [31:0] Sltu;
assign Sltu = (a < b)?1:0;
//算数右移（SRA）
wire [31:0] Sra;
assign Sra = b1 >>> a1;
//逻辑左移（SLA）/算数左移（SLL）
wire [31:0] Sll_Sla;
assign Sll_Sla = b << a;
//逻辑右移（SRL)
wire [31:0] Srl;
assign Srl = b >> a;


reg [32:0] an,bn,cn; //无符号数
always @(*)
begin
an={1'b0,a[31:0]};
bn={1'b0,b[31:0]};
//无符号加法
if (aluc == 4'b0000)
      begin
      r = Addu;
      cn = an + bn;
      carry = cn[32];
      end

else if (aluc == 4'b0010)
      begin
      r = Add;
      if( a1[31]== 1 && b1[31]==1 && Add[31] == 0)//负+负=正
          overflow <= 1'b1;
      else if ( a1[31]== 0 && b1[31]==0 && Add[31] == 1)//正+正=负
          overflow <= 1'b1;
      else
```

```
            overflow <= 1'b0;
        end
    else if (aluc == 4'b0001)
        begin
        r = Subu;
        cn <= an - bn;
        carry <= cn[32];
        end
    else if (aluc == 4'b0011)
        begin
        r = Sub;
        if( a1[31]== 0 && b1[31]==1 && Sub[31] == 1)//正-负=负
            overflow <= 1'b1;
        else if ( a1[31]== 1 && b1[31]==0 && Sub[31] == 0)//负-正=正
            overflow <= 1'b1;
        else
            overflow <= 1'b0;
        end
    else if (aluc == 4'b0100)
        r = And;
    else if (aluc == 4'b0101)
        r = Or;
    else if (aluc == 4'b0110)
        r = Xor;
    else if (aluc == 4'b0111)
        r = Nor;
    else if (aluc[3:1] == 3'b100)
        r = Lui;
    else if (aluc == 4'b1011)
        begin
        r = Slt;
        zero <= (a == b? 1:0);
        if (a1 - b1<0)
            negative <= 1'b1;
        else
            negative <= 1'b0;
        end
    else if (aluc == 4'b1010)
        begin
        r = Sltu;
        cn <= an - bn;
        zero <= (a == b? 1:0);
        carry <= cn[32];
        end
```

```
        else if (aluc == 4'b1100)
                begin
                r = Sra;
                carry <= (b1>>>(a-1)) & 32'b1;///
                end
        else if (aluc[3:1] == 3'b111)
                begin
                r = Sll_Sla;
                carry <= (b<<(a-1))>>31 & 32'b1;///
                end
        else if (aluc == 4'b1101)
                begin
                r = Srl;
                carry <= (b>>(a-1)) & 32'b1;///
                end
        if(aluc != 4'b1011 && aluc != 4'b1010)
                zero <= (r == 0 ? 1:0);
        if(aluc != 4'b1011)
                negative <= r[31];
        end
endmodule
```

四、测试模块建模

6.9

```
module ALU_tb();
        reg [31:0] a;
        reg [31:0] b;
        reg [3:0] aluc;
        wire [31:0] r;
        wire zero;
        wire carry;
        wire negative;
        wire overflow;
        alu
ALU(.a(a), .b(b), .aluc(aluc), .r(r), .zero(zero), .carry(carry), .negative(negative), .overflow(overfl
ow));
        initial
```

```
begin
        a <= 32'hffffffff;
        b <= 32'h0000ffff;
        aluc <= 4'b0011;
        #40
        a <= 32'h00000000;
        b <= 32'hffff1234;
        aluc <= 4'b1000;
        #40
        a <= 32'h00000008;
        b <= 32'hffffffff;
        aluc <= 4'b1100;
        #40
        a <= 32'h00000010;
        b <= 32'h80000000;
        aluc <= 4'b1100;
        #40
        a <= 32'h00000001;
        b <= 32'hffffffff;
        aluc <= 4'b1110;
        #40
        a <= 32'h00000008;
        b <= 32'hffffffff;
        aluc <= 4'b1110;
        #40
        a <= 32'h0000001f;
        b <= 32'h0000ffff;
        aluc <= 4'b1110;
end
        initial
                begin
                        //ADDU
                        aluc = 4'b0000;
                        a = 15;
                        b = 16;
                        #40
                        a = 32'b0;
                        b = 32'b0;
                        #40
                        a = 32'b10000000000000000000000000000000;
                        b = 32'b10000000000000000000000000000000;
                        //ADD
                        #40
                        aluc = 4'b0010;
```

```
a = -5;
b = 15;
#40
a = 10;
b = 15;
#40
a = -15;
b = -17;
#40
a = 32'b0;
b = 32'b0;
#40
a = 32'b01000000000000000000000000000000;
b = 32'b01000000000000000000000000000000;
#40
a = 32'b10000000000000000000000000000000;
b = 32'b11000000000000000000000000000000;
//SUBU
#40
aluc = 4'b0001;
a = 20;
b = 15;
#40
a = 100;
b = 99;
#40
a = 5;
b = 10;
//SUB
#40
aluc = 4'b0011;
a = 20;
b = 15;
#40
a = -5;
b = -10;
#40
a = -10;
b = -5;
#40
a = 32'b1;
b = 32'b00000000000000000000000111111111;
//AND OR XOR NOR
#40
```

```
        aluc = 4'b0100;
        a = 32'b10001000100010001000100010001000;
        b = 32'b01001000110010011000100010001000;
        #40 aluc = 4'b0101;
        #40 aluc = 4'b0110;
        #40 aluc = 4'b0111;
        //LUI
        #40
        aluc = 4'b1001;
        a = 32'b1;
        b = 32'b1;
        aluc = 4'b1000;
        a = 32'b1;
        b = 32'b1;
        //SLT, SLTU
        #40
        aluc = 4'b1011;
        a = 20;
        b = 15;
        #40
        a = 5;
        b = 10;
        aluc = 4'b1011;
        a = 20;
        b = 15;
        #40
        a = 5;
        b = 10;
        #40
        a = -5;
        b = -10;
        #40
        a = -5;
        b = 10;
        //SRA SLL/SLR SRL
        b = 32'b11111111000000000;
        a = 32'b101;
        //aluc = 4'b1110;
        #40
        aluc = 4'b1111;
        #40
        aluc = 4'b1101;
    end
endmodule
```

五、实验结果

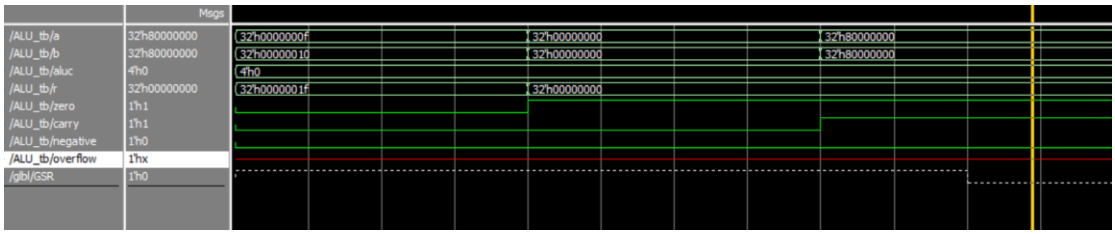
● 接口定义：

```
module alu(
    input [31:0] a,    //32 位输入，操作数 1
    input [31:0] b,    //32 位输入，操作数 2
    input [3:0] aluc, //4 位输入，控制 alu 的操作
    output [31:0] r,   //32 位输出，由 a、b 经过 aluc 指定的操作生成
    output zero,       //0 标志位
    output carry,      // 进位标志位
    output negative,   // 负数标志位
    output overflow   // 溢出标志位
);
```

6.9

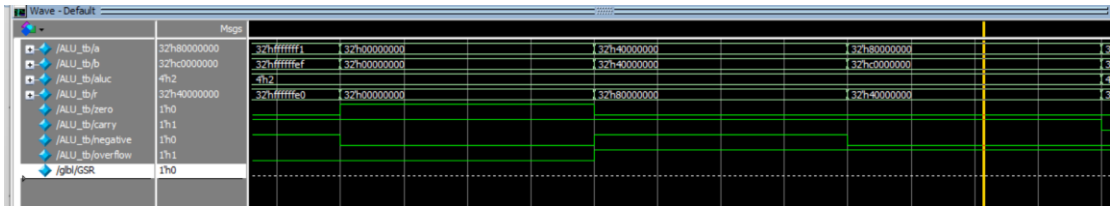
1.Addu 无符号加法

上溢出：carry=1



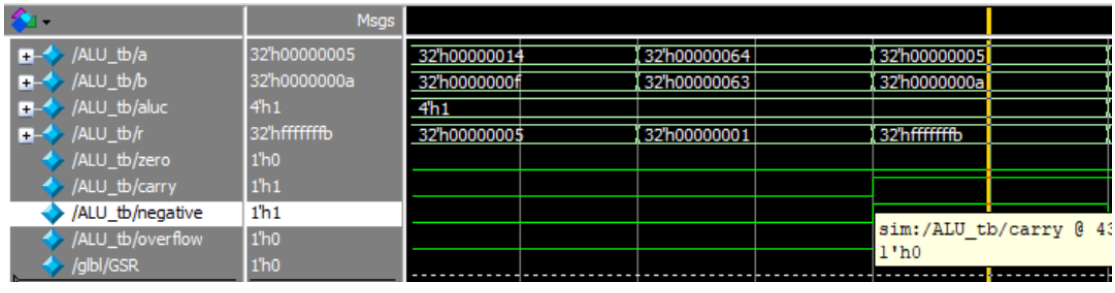
2.Add 有符号加法

上溢出：overflow=1



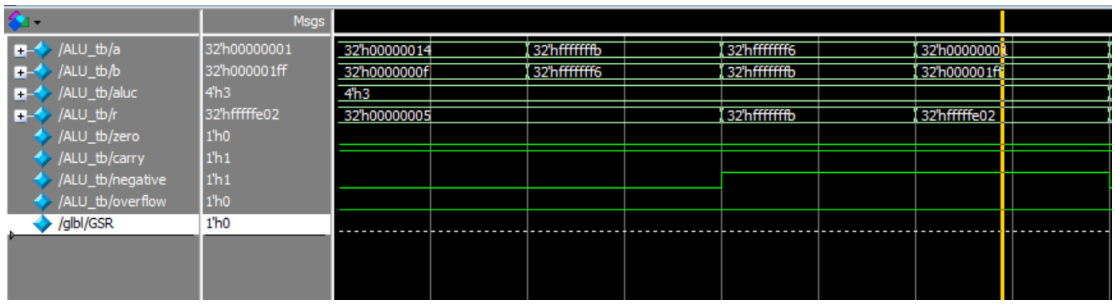
3.Subu 无符号减法

下溢出：carry=1

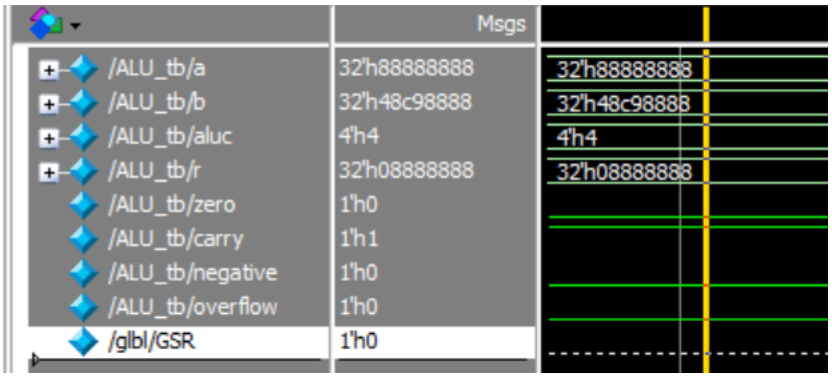


4.Add 有符号减法

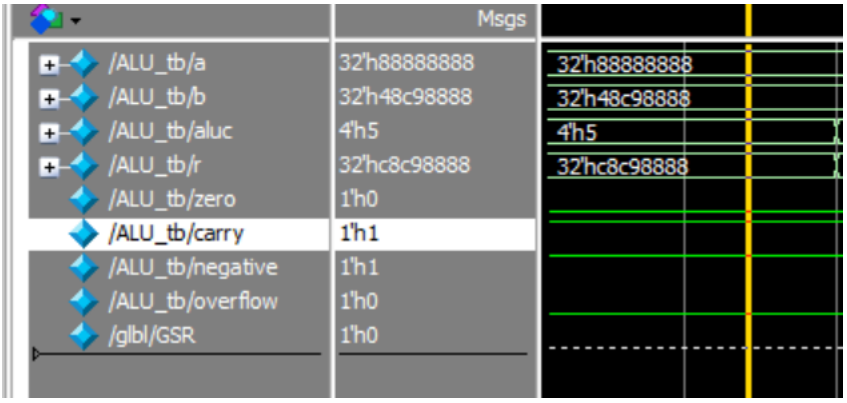
下溢出：overflow=1



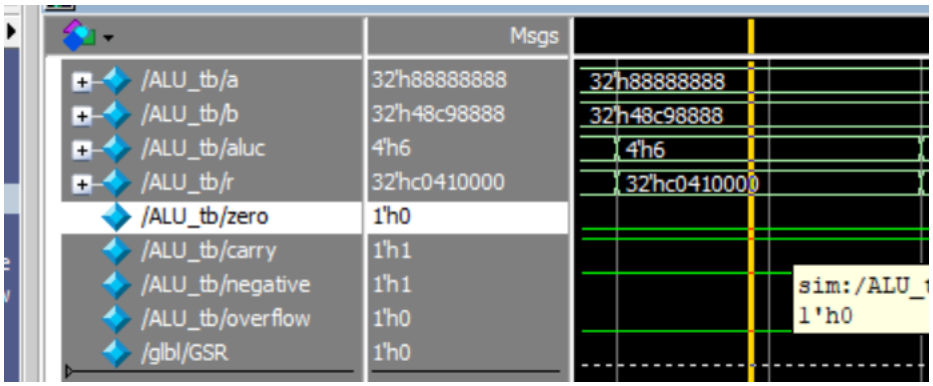
5.按位与



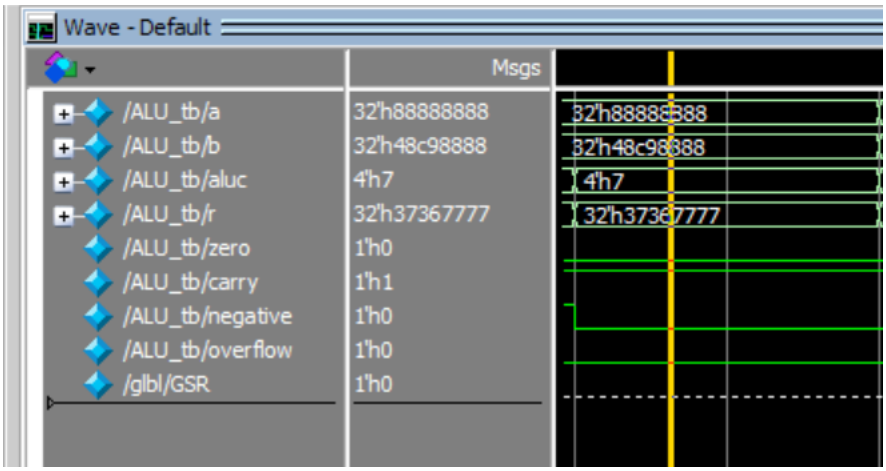
6.按位或



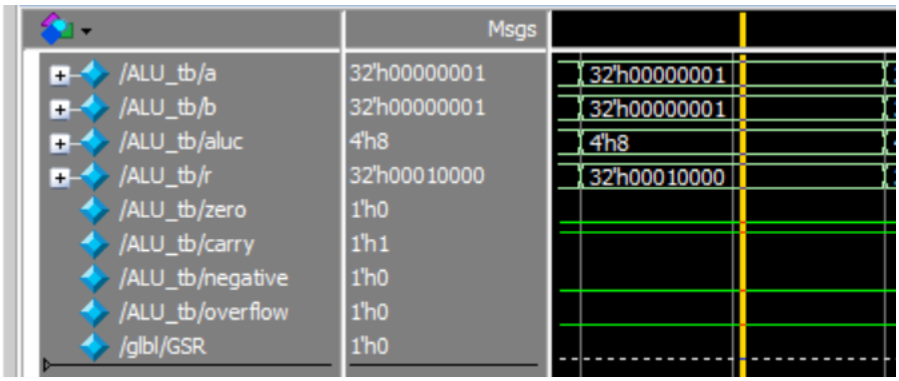
7.按位异或



8.按位与非



9.立即数



10.有符号比较（a<b）

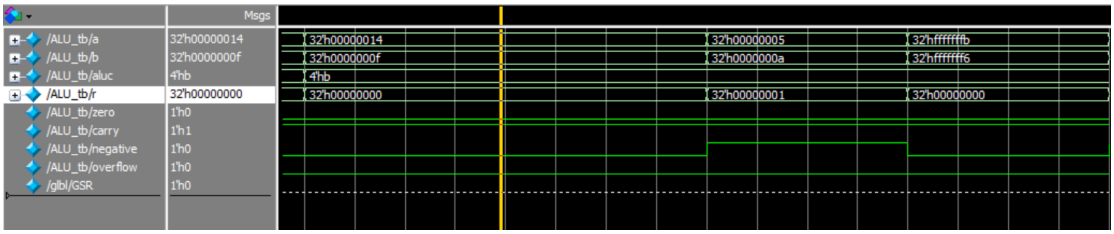
a-b=0：zero=1

a-b<0：negative=1

11.无符号比较（a<b）

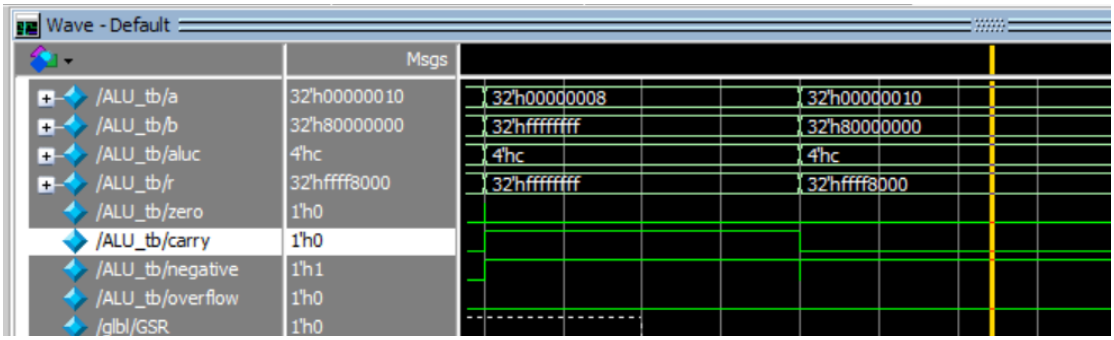
a-b=0：zero=1

a-b<0：carry=1



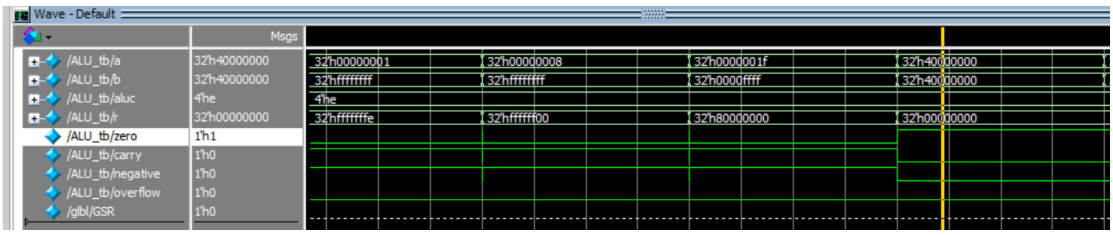
12.算术右移

最后一次移的数：carry



13.算术左移/算术右移

最后一次移的数：carry



14.逻辑右移

最后一次移的数：carry