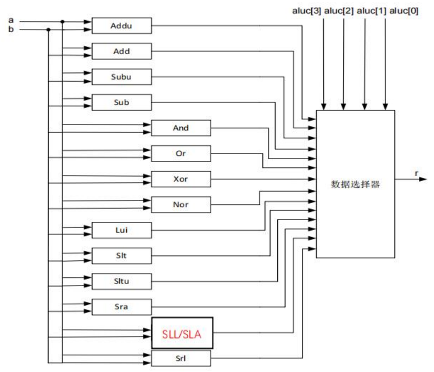
一、实验内容

使用 Verilog 实现行为级 ALU 的设计和仿真
算术逻辑单元(arithmetic and logic unit) 是能实现多组算术运算和逻辑运算的组合逻辑电路，简称 ALU。算术逻辑单元是中央处理器(CPU)的执行单元，是所有中央处理器的核心组成部分，由"And Gate"（与门） 和"Or Gate"（或门）构成的算术逻辑单元，主要功能是进行二位元的算术运算，如加减乘(不包括整数除法)。基本上，在所有现代 CPU 体系结构中，二进制都以补码的形式来表示。
模块连接图



Aluc 对应运算

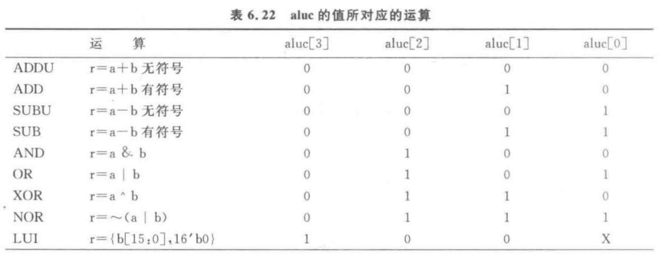
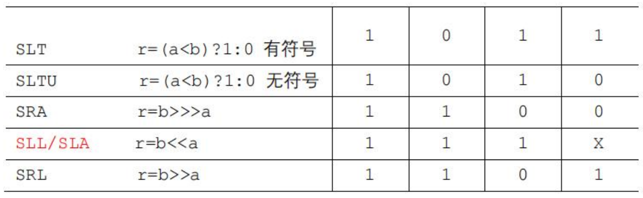


表 6.22 aluc 的值所对应的运算

	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/SLA	r=b<<a	1	1	1	X
SRL	r=b>>a	1	1	0	1

二、硬件逻辑图

三、模块建模

通过对每一个功能进行行为模拟，集合在同一模块里即可实现

```
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 ;

        wire [31:0] r_addu;              //无符号+
        assign r_addu = a + b;
        wire [31:0] r_add;               //有符号+
        assign r_add = a1 + b1;
        wire [31:0] r_subu;              //无符号-
        assign r_subu= a - b;
        wire [31:0] r_sub;              //有符号-
        assign r_sub = a1 - b1;
        wire [31:0] r_and;              //and
        assign r_and = a & b;
        wire [31:0] r_or;             //or
        assign r_or = a | b;
        wire [31:0] r_xor;            //xor
        assign r_xor = a ^ b;
        wire [31:0] r_nor;          //nor
        assign r_nor =~( a | b );
        wire [31:0] r_lui;        //高位置立即数
        assign r_lui = {b[15:0],16'b0};
        wire [31:0] r_slt;      //slt
        assign r_slt = (a1<b1)?1:0;
        wire [31:0] r_sltu;    //sltu
        assign r_sltu = (a<b)?1:0;
        wire [31:0] r_sra;    //算数右移
        reg [31:0] r_sra_t;
        assign r_sra = b>>>a;

        wire [31:0] r_sll;    //逻辑左移
        reg [31:0] r_sll_t;
        assign r_sll = b<<a;
```

```
wire [31:0] r_srl;    //逻辑右移
reg [31:0] r_srl_t;
assign r_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]};

     if (aluc == 4'b0000)
     begin
                    r <= r_addu;
                    if(r_addu == 0)
                       zero <= 1'b1;
                    else
                       zero <= 1'b0;
                    cn <= an + bn;
                    carry <= cn[32];
                    negative <= r_addu[31];
                    overflow <= 1'bz;
     end
   else    if (aluc == 4'b0010)
     begin     //有符号+
                    r <= r_add;
                    if(r_add == 0)
                       zero <= 1'b1;
                    else
                       zero <= 1'b0;
                    carry <= 1'bz;
                    negative <= r_add[31];

                    if( a1[31]== 1 && b1[31]==1 && r_add[31] == 0)
                         overflow <= 1'b1;
                    else if ( a1[31]== 0 && b1[31]==0 && r_add[31] == 1)
                         overflow <= 1'b1;
                    else
                         overflow <= 1'b0;
     end
   else if (aluc ==    4'b0001)
     begin     //无符号-
                    r <= r_subu;
                    if(r_subu == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                cn <= an - bn;
                carry <= cn[32];
                negative <= r_subu[31];
                overflow <= 1'bz;
  end
else if (aluc ==    4'b0011)
  begin     //有符号-
              r <= r_sub;
              if(r_sub == 0)
                 zero <= 1'b1;
              else
                 zero <= 1'b0;
              carry <= 1'bz;
              negative <= r_sub[31];

              if( a1[31]== 0 && b1[31]==1 && r_add[31] == 1)//正-负=负
                 overflow <= 1'b1;
              else if ( a1[31]== 1 && b1[31]==0 && r_add[31] == 0)//负-正=正
                 overflow <= 1'b1;
              else
                 overflow <= 1'b0;
  end
else if (aluc ==    4'b0100)
  begin    //and
              r <= r_and;
              if(r_and == 0)
                 zero <= 1'b1;
              else
                 zero <= 1'b0;
              carry <= 1'bz;
              negative <= r_and[31];
              overflow <= 1'bz;
  end
else if (aluc ==    4'b0101)
  begin    //or
              r <= r_or;
              if(r_or == 0)
                   zero <= 1'b1;
              else
                   zero <= 1'b0;
              carry <= 1'bz;
```

```
                negative <= r_or[31];
                overflow <= 1'bz;
  end
else if (aluc ==      4'b0110)
  begin    //xor
                r <= r_xor;
                if(r_xor == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                carry <= 1'bz;
                negative <= r_xor[31];
                overflow <= 1'bz;
  end
  else if (aluc ==    4'b0111)
  begin    //nor
                r <= r_nor;
                if(r_nor == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                carry <= 1'bz;
                negative <= r_nor[31];
                overflow <= 1'bz;
  end
  else if (aluc ==    4'b100x)
  begin    //lui
                r <= r_lui;
                if(r_lui == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                carry <= 1'bz;
                negative <= r_lui[31];
                overflow <= 1'bz;
  end
else if (aluc ==      4'b1011)
  begin    //slt
                r <= r_slt;
                if(r_slt == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                carry <= 1'bz;
                overflow <= r_slt[31];
                if (a1-b1<0)
                    negative <= 1'b1;
                else
                    negative <= 1'b0;
  end
else if (aluc ==      4'b1010)
  begin    //sltu
                r <= r_sltu;
                if(r_sltu == 0)
                    zero <= 1'b1;
                else
                    zero <= 1'b0;
                carry <= 1'bz;
                negative <= r_sltu[31];
                overflow <= 1'bz;
  end
else if (aluc ==      4'b1100)
  begin    //sra
                r <= r_sra;
                if(r_sra == 0)
                    zero <= 1'b1;
                else
                zero <= 1'b0;
                r_sra_t = b>>>(a-1);
                carry <= r_sra_t[0];
                negative <= r_sra[31];
                overflow <= 1'bz;
  end
  else if (aluc ==      4'b111x)
    begin    //sll/slr
                  r <= r_sll;
                  if(r_sll == 0)
                      zero <= 1'b1;
                  else
                  zero <= 1'b0;
                  r_sll_t = b<<(a-1);
                  carry <= r_sll_t[0];
                  negative <= r_sll[31];
                  overflow <= 1'bz;
    end
  else if (aluc ==        4'b1101)
      begin    //srl
                    r <= r_srl;
                    if(r_srl == 0)
                        zero <= 1'b1;
                    else
                    zero <= 1'b0;
                    r_srl_t = b>>(a-1);
                    carry <= r_srl_t[0];
                    negative <= r_srl[31];
                    overflow <= 1'bz;
            end

    end

endmodule
```

四、测试模块建模

分别通过对每一个模块的测试来观察逻辑是否正确，可将模块分为加减运算、逻辑运算以及移位运算三大部分

```
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));


        //ADDU
        /*initial
            begin
                //ADDU
                aluc = 4'b0000;
                a = 15;
                b = 16;
        #40
        a = 5;
        b = 7;

        #40
        a = 32'b0;
        b = 32'b0;

        #40
        a = 32'b10000000000000000000000000000000;
        b = 32'b10000000000000000000000000000000;


    end*/

/*
  initial
  begin
    //ADD
    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;
```

```
end*/

/*
initial
        begin
                //SUBU
                aluc = 4'b0001;
                a = 20;
                b = 15;

                #40
                a = 100;
                b = 99;

                #40
                a = 5;
                b = 10;


        end
*/
/*
initial
begin
        //SUB
        aluc = 4'b0011;
        a = 20;
        b = 15;

        #40
        a = -5;
        b = -10;

        #40
        a = -10;
        b = -5;

      #40
        a = 32'b1;
        b = 32'b00000000000000000000000111111111;


end
*/
```

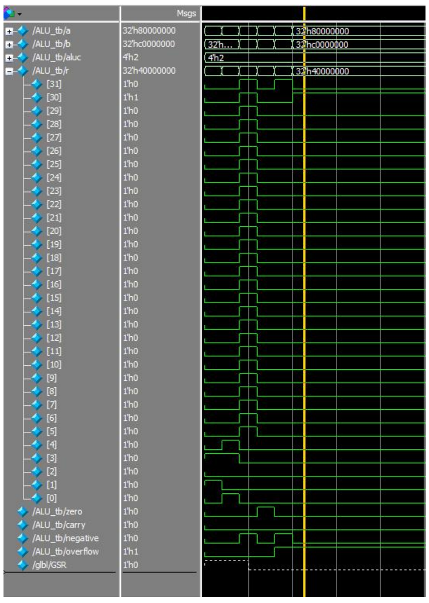
```
    /*
        initial
        begin
            //AND OR XOR NOR
            a = 32'b10001000100010001000100010001000;
            b = 32'b01001000110010011000100010001000;

            aluc = 4'b0100;
            #40 aluc = 4'b0101;
            #40 aluc = 4'b0110;
            #40 aluc = 4'b0111;

        end
*/
     /*
        initial
        begin
            //LUI
            aluc = 4'b1001;
            a = 32'b1;
            b = 32'b1;

            aluc = 4'b1000;
            a = 32'b1;
            b = 32'b1;


        end
       */

     /*
        initial
        begin
            //SLT, SLTU
            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;

            end

        */

        initial
        begin

              //SRA SLL/SLR SRL
                b = 32'b000000000000000011111111000000000;
              a = 32'b101;
              //aluc = 4'b1110;



              #40

              aluc = 4'b1111;

              #40

              aluc = 4'b1101;

        end


endmodule
```

五、实验结果

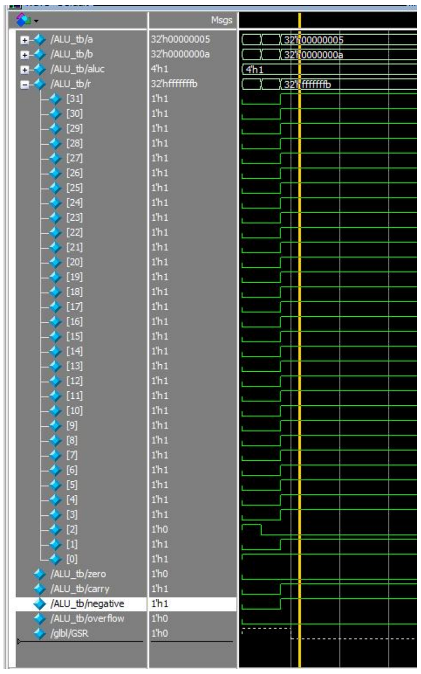
1. ModelSim 波形仿真

ADD/ADDU 功能



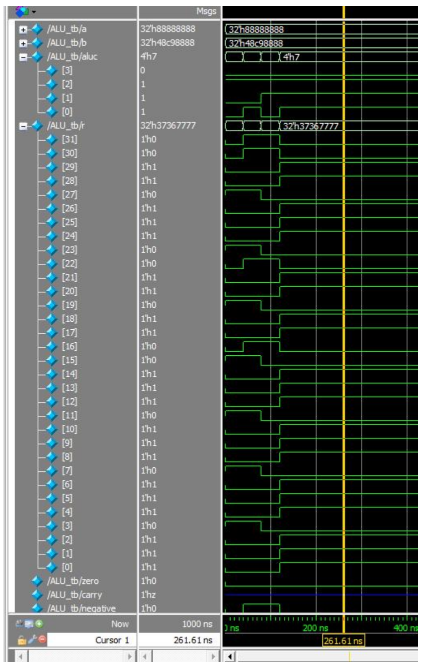
```
                     Msgs
/ALU_tb/a            32'h80000000
/ALU_tb/b            32'hc0000000
/ALU_tb/aluc         4'h2
/ALU_tb/r            32'h40000000
  [31]               1'h0
  [30]               1'h1
  [29]               1'h0
  [28]               1'h0
  [27]               1'h0
  [26]               1'h0
  [25]               1'h0
  [24]               1'h0
  [23]               1'h0
  [22]               1'h0
  [21]               1'h0
  [20]               1'h0
  [19]               1'h0
  [18]               1'h0
  [17]               1'h0
  [16]               1'h0
  [15]               1'h0
  [14]               1'h0
  [13]               1'h0
  [12]               1'h0
  [11]               1'h0
  [10]               1'h0
  [9]                1'h0
  [8]                1'h0
  [7]                1'h0
  [6]                1'h0
  [5]                1'h0
  [4]                1'h0
  [3]                1'h0
  [2]                1'h0
  [1]                1'h0
  [0]                1'h0
/ALU_tb/zero         1'h0
/ALU_tb/carry        1'h0
/ALU_tb/negative     1'h0
/ALU_tb/overflow     1'h1
/glbl/GSR            1'h0
```

SUB/SUBU 功能



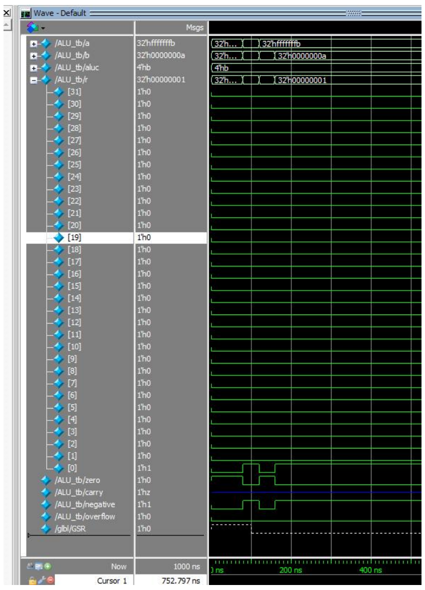
```
                     Msgs
/ALU_tb/a            32'h00000005
/ALU_tb/b            32'h0000000a
/ALU_tb/aluc         4'h1
/ALU_tb/r            32'hfffffffb
  [31]               1'h1
  [30]               1'h1
  [29]               1'h1
  [28]               1'h1
  [27]               1'h1
  [26]               1'h1
  [25]               1'h1
  [24]               1'h1
  [23]               1'h1
  [22]               1'h1
  [21]               1'h1
  [20]               1'h1
  [19]               1'h1
  [18]               1'h1
  [17]               1'h1
  [16]               1'h1
  [15]               1'h1
  [14]               1'h1
  [13]               1'h1
  [12]               1'h1
  [11]               1'h1
  [10]               1'h1
  [9]                1'h1
  [8]                1'h1
  [7]                1'h1
  [6]                1'h1
  [5]                1'h1
  [4]                1'h1
  [3]                1'h1
  [2]                1'h0
  [1]                1'h1
  [0]                1'h1
/ALU_tb/zero         1'h0
/ALU_tb/carry        1'h1
/ALU_tb/negative     1'h1
/ALU_tb/overflow     1'h0
/glbl/GSR            1'h0
```

AND/OR/XOR/NOR 功能



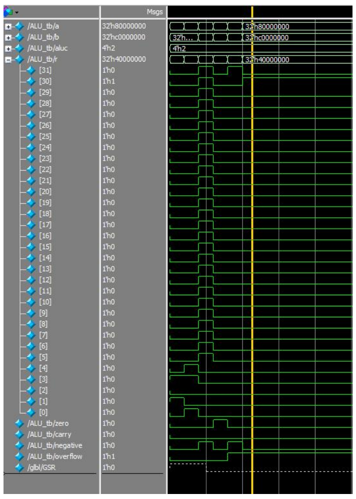
```
                          Msgs
/ALU_tb/a                 32'h88888888
/ALU_tb/b                 32'h48c98888
/ALU_tb/aluc              4'h7
  [3]                     0
  [2]                     1
  [1]                     1
  [0]                     1
/ALU_tb/r                 32'h37367777
  [31]                    1'h0
  [30]                    1'h0
  [29]                    1'h1
  [28]                    1'h1
  [27]                    1'h0
  [26]                    1'h1
  [25]                    1'h1
  [24]                    1'h1
  [23]                    1'h0
  [22]                    1'h0
  [21]                    1'h1
  [20]                    1'h1
  [19]                    1'h0
  [18]                    1'h1
  [17]                    1'h1
  [16]                    1'h0
  [15]                    1'h0
  [14]                    1'h1
  [13]                    1'h1
  [12]                    1'h1
  [11]                    1'h0
  [10]                    1'h1
  [9]                     1'h1
  [8]                     1'h1
  [7]                     1'h0
  [6]                     1'h1
  [5]                     1'h1
  [4]                     1'h1
  [3]                     1'h0
  [2]                     1'h1
  [1]                     1'h1
  [0]                     1'h1
/ALU_tb/zero              1'h0
/ALU_tb/carry             1'hz
/ALU_tb/negative          1'h0
Now                       1000 ns
Cursor 1                  261.61 ns
```

SLT/SLTU 功能



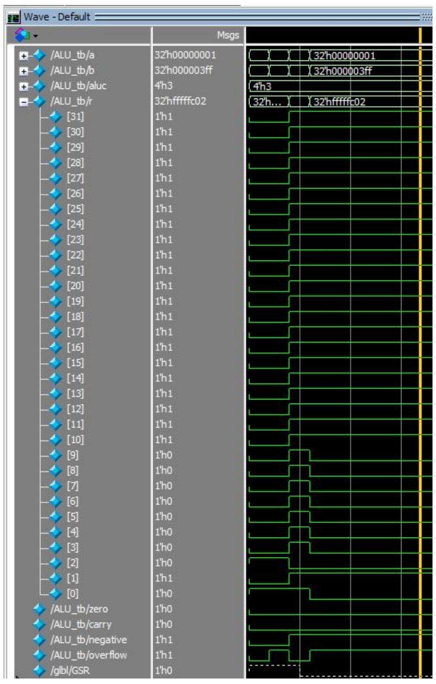
```
Wave - Default
                          Msgs
/ALU_tb/a                 32'hfffffffb
/ALU_tb/b                 32'h0000000a
/ALU_tb/aluc              4'hb
/ALU_tb/r                 32'h00000001
  [31]                    1'h0
  [30]                    1'h0
  [29]                    1'h0
  [28]                    1'h0
  [27]                    1'h0
  [26]                    1'h0
  [25]                    1'h0
  [24]                    1'h0
  [23]                    1'h0
  [22]                    1'h0
  [21]                    1'h0
  [20]                    1'h0
  [19]                    1'h0
  [18]                    1'h0
  [17]                    1'h0
  [16]                    1'h0
  [15]                    1'h0
  [14]                    1'h0
  [13]                    1'h0
  [12]                    1'h0
  [11]                    1'h0
  [10]                    1'h0
  [9]                     1'h0
  [8]                     1'h0
  [7]                     1'h0
  [6]                     1'h0
  [5]                     1'h0
  [4]                     1'h0
  [3]                     1'h0
  [2]                     1'h0
  [1]                     1'h0
  [0]                     1'h1
/ALU_tb/zero              1'h0
/ALU_tb/carry             1'hz
/ALU_tb/negative          1'h1
/ALU_tb/overflow          1'h0
/glbl/GSR                 1'h0
Now                       1000 ns
Cursor 1                  752.797 ns
```

SRA SLL/SLR SRL 功能



```
                          Msgs
/ALU_tb/a                 32'h80000000
/ALU_tb/b                 32'hc0000000
/ALU_tb/aluc              4'h2
/ALU_tb/r                 32'h40000000
  [31]                    1'h0
  [30]                    1'h1
  [29]                    1'h0
  [28]                    1'h0
  [27]                    1'h0
  [26]                    1'h0
  [25]                    1'h0
  [24]                    1'h0
  [23]                    1'h0
  [22]                    1'h0
  [21]                    1'h0
  [20]                    1'h0
  [19]                    1'h0
  [18]                    1'h0
  [17]                    1'h0
  [16]                    1'h0
  [15]                    1'h0
  [14]                    1'h0
  [13]                    1'h0
  [12]                    1'h0
  [11]                    1'h0
  [10]                    1'h0
  [9]                     1'h0
  [8]                     1'h0
  [7]                     1'h0
  [6]                     1'h0
  [5]                     1'h0
  [4]                     1'h0
  [3]                     1'h0
  [2]                     1'h0
  [1]                     1'h0
  [0]                     1'h0
/ALU_tb/zero              1'h0
/ALU_tb/carry             1'h0
/ALU_tb/negative          1'h0
/ALU_tb/overflow          1'h1
/glbl/GSR                 1'h0
```

LUI 功能



```
Wave - Default
                     Msgs
/ALU_tb/a            32'h00000001
/ALU_tb/b            32'h000003ff
/ALU_tb/aluc         4'h3
/ALU_tb/r            32'hfffffc02
  [31]               1'h1
  [30]               1'h1
  [29]               1'h1
  [28]               1'h1
  [27]               1'h1
  [26]               1'h1
  [25]               1'h1
  [24]               1'h1
  [23]               1'h1
  [22]               1'h1
  [21]               1'h1
  [20]               1'h1
  [19]               1'h1
  [18]               1'h1
  [17]               1'h1
  [16]               1'h1
  [15]               1'h1
  [14]               1'h1
  [13]               1'h1
  [12]               1'h1
  [11]               1'h1
  [10]               1'h1
  [9]                1'h0
  [8]                1'h0
  [7]                1'h0
  [6]                1'h0
  [5]                1'h0
  [4]                1'h0
  [3]                1'h0
  [2]                1'h0
  [1]                1'h1
  [0]                1'h0
/ALU_tb/zero         1'h0
/ALU_tb/carry        1'h0
/ALU_tb/negative     1'h1
/ALU_tb/overflow     1'h1
/glbl/GSR            1'h0
```

可以看到所有输出据符合预期，逻辑验证正确