

Lab01 ALU

PB15000102 王嵩超

实验内容与要求

设计一算数运算单元ALU

- 采用纯组合逻辑设计
- 32bit位宽
- 完成制定运算功能
- 模块接口需求

```
1 module ALU(  
2     input signed    [31:0]  alu_a,  
3     input signed    [31:0]  alu_b,  
4     input           [4:0]    alu_op,  
5     output          [31:0]  alu_out  
6 )
```

- 操作数与运算

实现以下7种操作：

parameter	A_NOP	= 5'h00;	空运算
parameter	A_ADD	= 5'h01;	符号加
parameter	A_SUB	= 5'h02;	符号减
parameter	A_AND	= 5'h03;	与
parameter	A_OR	= 5'h04;	或
parameter	A_XOR	= 5'h05;	异或
parameter	A_NOR	= 5'h06;	或非

- 完成以下运算

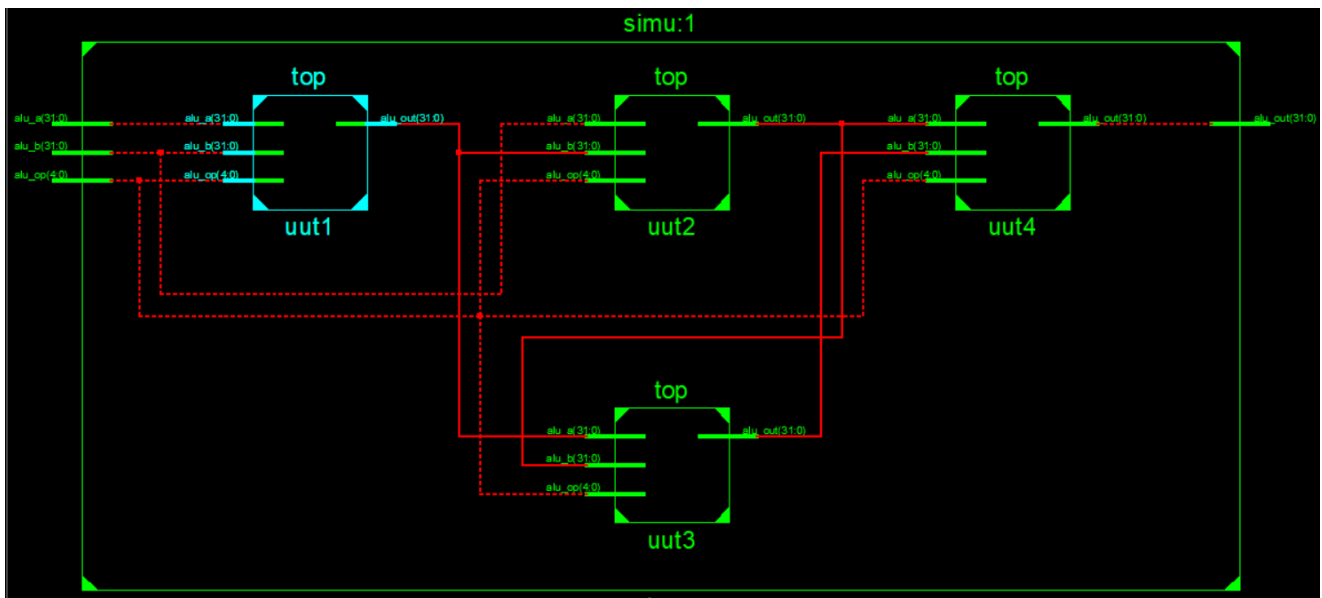
- 斐波拉契数列
2, 2, 4, 6, 10, 16...
- 输入为a, b, 其中a=2, b=2
- 调用ALU完成：
 - 输入为a=b=2, 输出为16
 - 需要定义一个顶层模块，模块内调用ALU模块N次

注意：要求中提到，运算单元采用纯组合逻辑设计。故在顶层模块应实例化4个ALU模块。在平时用时序逻辑设计中没必要用如此大的开销。

实验设计

- 首先设计**ALU**模块，该模块是运算单元。
仅用**always**模块，**if**语句和各输入变量的逻辑运算即可完成。
- 再编写**top**模块，该模块例化了4个**ALU**模块，并将运算结果依次串接，实现斐波拉契数列的计算。
- 再编写**simu**仿真模块，该模块提供输入用来仿真。

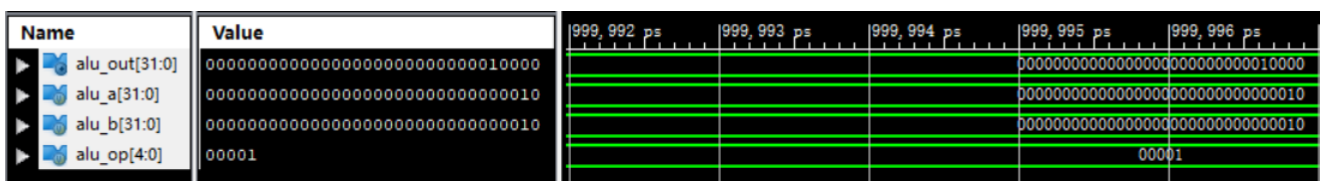
RTL级电路框图（模块名称稍有不同）



仿真结果

斐波拉契数列计算

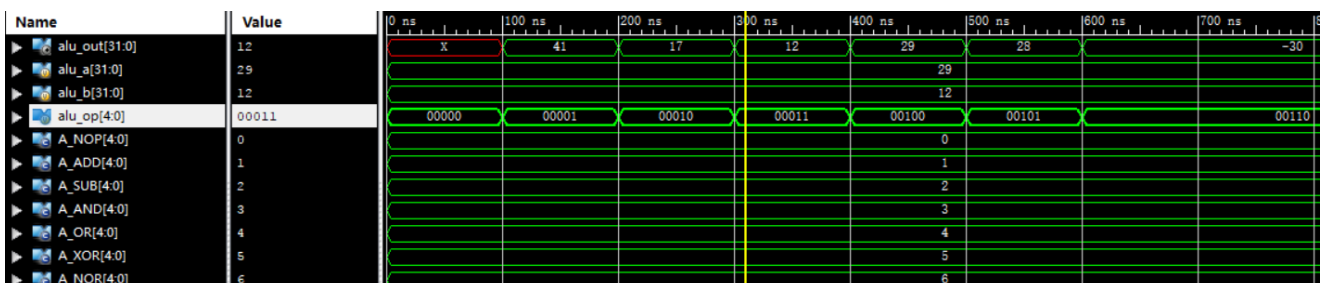
新建一个Verilog Test Fixture源文件，对top模块仿真：



输出结果为16(10000)。

各操作符运算仿真

新建一个Verilog Test Fixture源文件，对alu模块仿真：



操作符每隔100ns切换一次，刚开始的空操作使alu_out置为高阻态。往后依次是符号加、符号减、与运算、或运算、异或运算、或非运算。

源代码

ALU模块：

```

1  `timescale 1ns / 1ps
2
3  module alu(
4      input signed [31:0] alu_a,
5      input signed [31:0] alu_b,
6      input [4:0] alu_op,
7      output reg [31:0] alu_out
8  );
9  parameter A_NOP = 5'h00; //空运算
10 parameter A_ADD = 5'h01; //符号加
11 parameter A_SUB = 5'h02; //符号减
12 parameter A_AND = 5'h03; //与
13 parameter A_OR = 5'h04; //或
14 parameter A_XOR = 5'h05; //异或
15 parameter A_NOR = 5'h06; //或非
16
17 always@(*)
18 begin
19     case(alu_op)
20         A_NOP:
21             begin
22
23             end
24         A_ADD:
25             begin
26                 alu_out <= alu_a + alu_b;
27             end
28         A_SUB:
29             begin
30                 alu_out <= alu_a - alu_b;
31             end
32         A_AND:
33             begin
34                 alu_out <= alu_a & alu_b;
35             end
36         A_OR:
37             begin
38                 alu_out <= alu_a | alu_b;
39             end
40         A_XOR:
41             begin
42                 alu_out <= alu_a ^ alu_b;
43             end
44         A_NOR:
45             begin
46                 alu_out <= ~(alu_a | alu_b);
47             end
48     endcase
49 end
50 endmodule

```

斐波拉契数列模块:

```

1  `timescale 1ns / 1ps
2  module top(
3      input signed [31:0] alu_a,
4      input signed [31:0] alu_b,
5      input [4:0] alu_op,
6      output [31:0] alu_out
7  );
8
9      // Temps
10     wire [31:0] sum1;
11     wire [31:0] sum2;
12     wire [31:0] sum3;
13
14     // Outputs
15
16     // Instantiate the Unit Under Test (UUT)
17     alu uut1 (
18         .alu_a(alu_a),
19         .alu_b(alu_b),
20         .alu_op(alu_op),
21         .alu_out(sum1)
22     );
23     alu uut2 (
24         .alu_a(alu_b),
25         .alu_b(sum1),
26         .alu_op(alu_op),
27         .alu_out(sum2)
28     );
29     alu uut3 (
30         .alu_a(sum1),
31         .alu_b(sum2),
32         .alu_op(alu_op),
33         .alu_out(sum3)
34     );
35     alu uut4 (
36         .alu_a(sum2),
37         .alu_b(sum3),
38         .alu_op(alu_op),
39         .alu_out(alu_out)
40     );
41
42 endmodule

```

斐波拉契数列仿真:

```
1 `timescale 1ns / 1ps
2 module simu0;
3
4     // Inputs
5     reg [31:0] alu_a;
6     reg [31:0] alu_b;
7     reg [4:0] alu_op;
8
9     // Outputs
10    wire [31:0] alu_out;
11
12    // Instantiate the Unit Under Test (UUT)
13    simu uut (
14        .alu_a(alu_a),
15        .alu_b(alu_b),
16        .alu_op(alu_op),
17        .alu_out(alu_out)
18    );
19
20    initial begin
21        // Initialize Inputs
22        // Add stimulus here
23        alu_a = 2;
24        alu_b = 2;
25        alu_op = 5'h01;
26    end
27 endmodule
```

各操作符仿真模块:

```

1  `timescale 1ns / 1ps
2
3  module simueach;
4
5      // Inputs
6      reg [31:0] alu_a;
7      reg [31:0] alu_b;
8      reg [4:0] alu_op;
9
10     // Outputs
11     wire [31:0] alu_out;
12
13     parameter A_NOP = 5'h00; //空运算
14     parameter A_ADD = 5'h01; //符号加
15     parameter A_SUB = 5'h02; //符号减
16     parameter A_AND = 5'h03; //与
17     parameter A_OR  = 5'h04; //或
18     parameter A_XOR = 5'h05; //异或
19     parameter A_NOR = 5'h06; //或非
20     // Instantiate the Unit Under Test (UUT)
21     top uut (
22         .alu_a(alu_a),
23         .alu_b(alu_b),
24         .alu_op(alu_op),
25         .alu_out(alu_out)
26     );
27     initial begin
28         // Initialize Inputs
29         alu_a = 29;
30         alu_b = 12;
31         alu_op = A_NOP;
32
33         // Wait 100 ns for global reset to finish
34         #100;
35         alu_op = A_ADD;
36         #100;
37         alu_op = A_SUB;
38         #100;
39         alu_op = A_AND;
40         #100;
41         alu_op = A_OR;
42         #100;
43         alu_op = A_XOR;
44         #100;
45         alu_op = A_NOR;
46         // Add stimulus here
47     end
48 endmodule

```

