Lab1_ALU

实验目的：

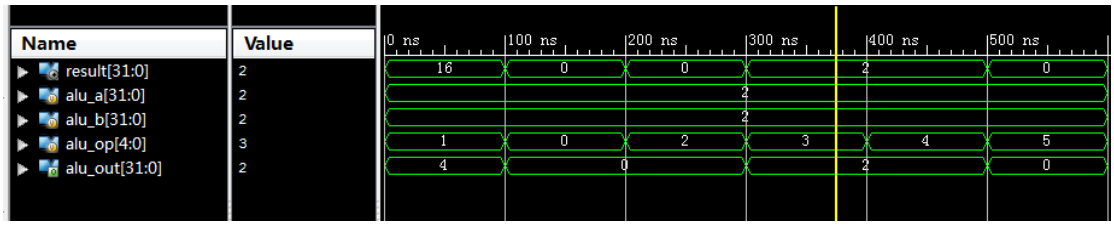
- 实现简单 ALU
- 同时，通过这个实验，复习 Verilog 语法，ISE 开发环境，仿真过程，为后续实验做准备。

实验内容：

- 设计一算数运算单元 ALU，按如下要求：
  - -采用纯组合逻辑设计
  - -32 位宽
  - -完成指定运算功能
- 使用模块调用完成一下运算
  - 调用 ALU 输出斐波那契数列中的 16
  - 调用 N 次

实验结果：

按要求完成。



仿真结果在一张图中给出。

第一段为输出斐波那契 16 部分；

后面依次为按 ppt 给出的运算顺序得出的结果：

操作数分别为 alu_a,alu_b，对应结果为 alu_out。

实验分析：

- ppt 已给出模块定义部分的代码，针对对应的操作符，只需要简单的位运算即得。
- 而 top 模块，只需要写出调用 4 次即可。

意见建议：

作为本学期第一次实验，个人感觉十分恰当！

附录：

- 代码：
  - ALU 部分：

```
`timescale 1ns / 1ps

module ALU(
input signed [31:0] alu_a,
input signed [31:0] alu_b,
input [4:0] alu_op,
output reg [31:0] alu_out
    );

parameter  A_NOP = 5'h00;
parameter  A_ADD = 5'h01;
parameter  A_SUB = 5'h02;
parameter  A_AND = 5'h03;
parameter  A_OR  = 5'h04;
parameter  A_XOR = 5'h05;
parameter  A_NOR = 5'h06;

always@(*)
begin
   case(alu_op)
     A_NOP:  alu_out = 0;
     A_ADD:  alu_out = alu_a + alu_b;
     A_SUB:  alu_out = alu_a - alu_b;
     A_AND:  alu_out = alu_a & alu_b;
     A_OR:   alu_out = alu_a | alu_b;
     A_XOR:  alu_out = (~alu_a & alu_b) | (alu_a & ~alu_b);
     A_NOR:  alu_out = ~(alu_a | alu_b);
     endcase
end
endmodule
```

Top 部分：

```
`timescale 1ns / 1ps

module top(
input [31:0] alu_a,
input [31:0] alu_b,
input [4:0] alu_op,
output [31:0] result
    );

wire [31:0] alu_out1,alu_out2,alu_out3;

ALU ALU1(alu_a,alu_b,alu_op,alu_out1);
ALU ALU2(alu_out1,alu_b,alu_op,alu_out2);
ALU ALU3(alu_out2,alu_out1,alu_op,alu_out3);
ALU ALU4(alu_out3,alu_out2,alu_op,result);

endmodule
```

仿真部分：

```
1  `timescale 1ns / 1ps
2
3
4
5  module lab_f;
6
7     // Inputs
8     reg [31:0] alu_a;
9     reg [31:0] alu_b;
10    reg [4:0] alu_op;
11
12    // Outputs
13    wire [31:0] result;
14
15    // Instantiate the Unit Under Test (UUT)
16    top uut (
17       .alu_a(alu_a),
18       .alu_b(alu_b),
19       .alu_op(alu_op),
20       .result(result)
21    );
22
23    initial begin
24       // Initialize Inputs
25       alu_a = 0;
26       alu_b = 0;
27       alu_op = 0;
28
29       // Wait 100 ns for global reset to finish
30       #100;
31
32       // Add stimulus here
33
34    end
35
36 endmodule
```