

# lab 1 运算器及其运用

PB21081601 张芷苒

## 1. 实验目的

- 掌握算术逻辑单元 (ALU) 的功能
- 掌握数据通路和控制器的设计方法
- 掌握组合电路和时序电路，以及参数化和结构化的 Verilog 描述方法

## 2. 逻辑设计

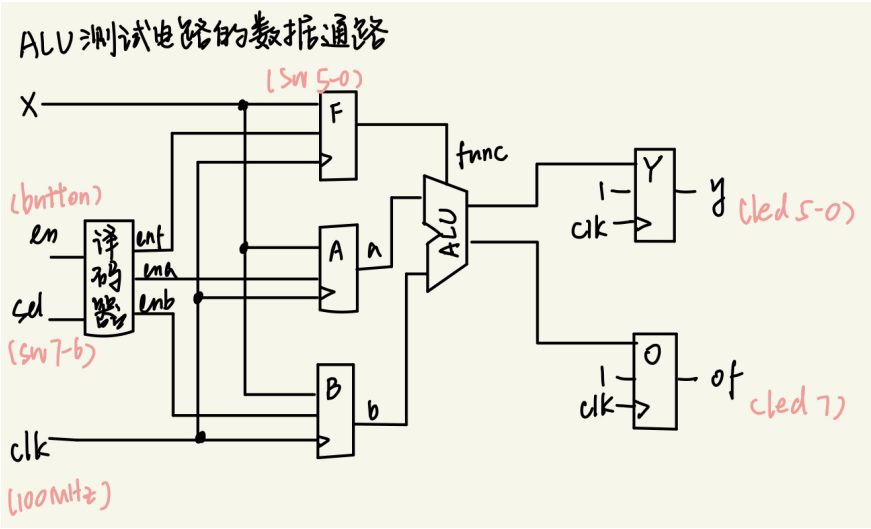
### 1. 算术逻辑单元 ALU

- 功能模块表

ALU模块功能表

func	y	of
0000	$a + b$	*
0001	$a - b$	*
0010	$a == b$	0
0011	$a <_u b$	0
0100	$a < b$	0
0101	$a \& b$	0
0110	$a   b$	0
0111	$a \wedge b$	0
1000	$a >> b$	0
1001	$a << b$	0
其他	0	0

- 数据通路



```
// ALU 模块 alu.v
module alu #(parameter WIDTH = 6) (
    input [WIDTH - 1:0] a, b,           // 操作数
    input [2:0] f,                     // 功能
    output reg [WIDTH - 1:0] y,        // 运算结果
    output z                            // 溢出标志of
);

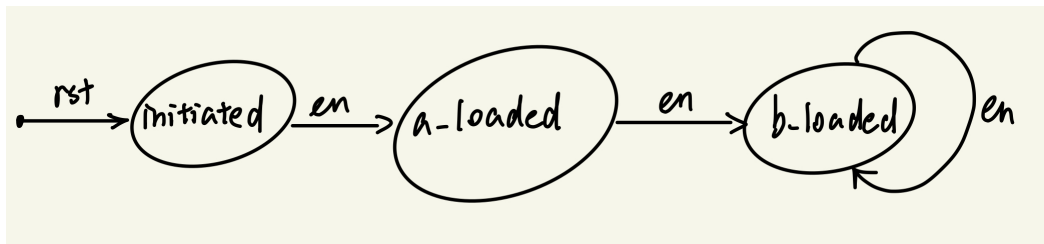
assign z = (y == {WIDTH{1'h0}}); // 如果有溢出，则为1，否则为0
always @(*) begin
    case (f)
        4'b0000:
            y = a + b;
        4'b0001:
            y = a - b;
        4'b0010:
            y = (a == b) ? 6'b000001 : 6'b0; // 若满足a等于b，则为1，否则为0
        4'b0011:
            y = (a < b) ? 6'b000001 : 6'b0; // 若满足a小于b，则为1，否则为0
        4'b0101:
            y = a & b;
        4'b0110:
            y = a | b;
        4'b0111:
            y = a ^ b;
        default:
            y = {WIDTH{1'h0}};           // 未定义功能
    endcase
end
endmodule
```

### FLG数据通路

The diagram illustrates the data path of the FLG (Finite Length Generator) block. It consists of the following components and connections:

- FSM (Finite State Machine):** Receives an external `en` (enable) signal through a `signal edge` block. It has `rst` (reset) and `clk` (clock) inputs. Its `state` output is connected to the `en` input of the top register and the `rst` input of the bottom register.
- Top Register (REG):** Labeled `REG` in red. It has `rst` and `clk` inputs. Its output is `a`. It is also connected to the `rst` input of the bottom register.
- Bottom Register (REG):** Labeled `REG` in red. It has `rst` and `clk` inputs. Its output is `b`. It is also connected to the `rst` input of the top register.
- MUX (Multiplexer):** Two MUX blocks are shown. The top MUX has inputs `d` and `a`, and its output is `a`. The bottom MUX has inputs `d` and `b`, and its output is `b`.
- ALU (Arithmetic Logic Unit):** Labeled `ALU` in red. It takes inputs `a` and `b` and performs a function `func=0000 (+)` to produce output `f`.
- Feedback:** The output `f` of the ALU is fed back to the `rst` input of the top register.

- 状态转移图



- 核心代码

```
//取信号边缘 get_edge.v
module get_edge(
    input clk,
    input rst,
    input signal,
    output signal_edge
);
    reg signal_r1, signal_r2;
    always @(posedge clk) signal_r1 <= ~rst & signal;
    always @(posedge clk) signal_r2 <= signal_r1;
    assign signal_edge = signal_r1 & ~signal_r2;
endmodule
```

```
//有限状态机 fsm.v
// 状态机有三个状态，分别为 initiated, a_loaded, b_loaded, 输出为当前状态
module fsm(
    input clk,
    input rst,
    input en,
    output [1:0] state
);
    reg [1:0] curr_state;
    reg [1:0] next_state;

    parameter initiated = 2'b00;
    parameter a_loaded = 2'b01;
    parameter b_loaded = 2'b10;

    // 1. 状态转换
    always @(posedge clk) begin
        if (rst) curr_state <= initiated;
        else if (en) curr_state <= next_state;
    end

    // 2. 下一状态 NS
    always @(curr_state) begin
        case (curr_state)
            initiated: next_state = a_loaded;
        endcase
    end
endmodule
```

```

        a_loaded: next_state = b_loaded;
        b_loaded: next_state = b_loaded;
        default: next_state = initiated;
    endcase
end
end

// 3. 输出逻辑
assign state = curr_state;
endmodule

```

```

// 综合 fls.v
// 在 FSM 状态转移时刻 (en_edge 为高电平),
// ALU 输入处的寄存器被激活, 对应的信号被传入 ALU
module fls(
    input clk,
    input rst,
    input en,
    input [6:0] d,
    output reg [6:0] f
);
    // 布线
    // get edge
    wire en_edge;
    get_edge get_en_edge(
        .clk(clk),
        .rst(rst),
        .signal(en),
        .signal_edge(en_edge)
    );
    // ALU
    reg [6:0] a;
    wire [6:0] alu_out;
    alu #(.WIDTH(7)) adder(
        .a(a),
        .b(f),
        .func(4'b0000),    // ALU 模式设定为加法
        .y(alu_out)
    );

    // FSM
    wire [1:0] sel;
    fsm fsm1(
        .clk(clk),
        .rst(rst),
        .en(en_edge),
        .state(sel)
    );

    // registers and MUXes
    always @(posedge clk) begin
        if (rst) a <= 7'h00;
    end
endmodule

```

```

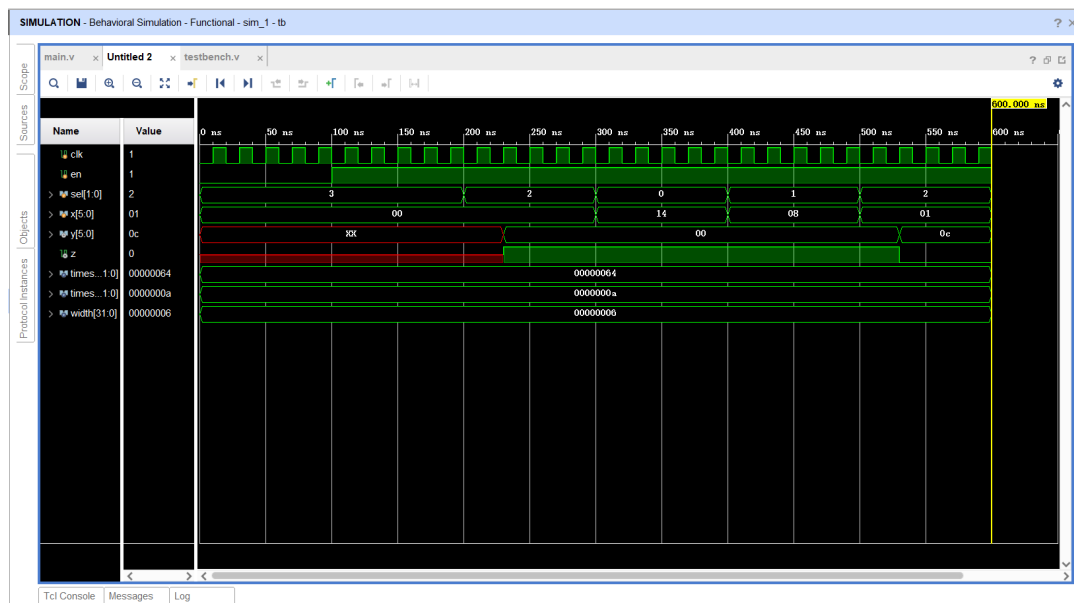
        else if (en_edge) begin
            case (sel)
                2'b00: a <= d;
                2'b10: a <= f;
                default: a <= a;
            endcase
        end
    end
end
always @(posedge clk) begin
    if (rst) f <= 7'h00;
    else if (en_edge) begin
        case (sel)
            2'b00: f <= d;
            2'b01: f <= d;
            2'b10: f <= alu_out;
            default: f <= f;
        endcase
    end
end
endmodule

```

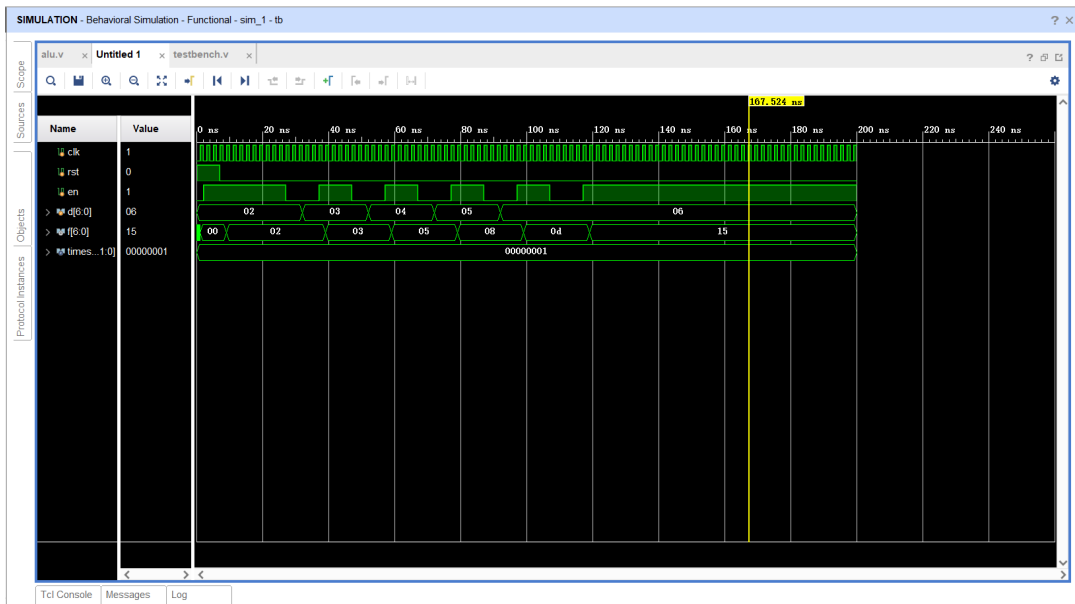
## 3. 仿真结果与分析

### 1. ALU 设计

由仿真文件生成的仿真波形如下：



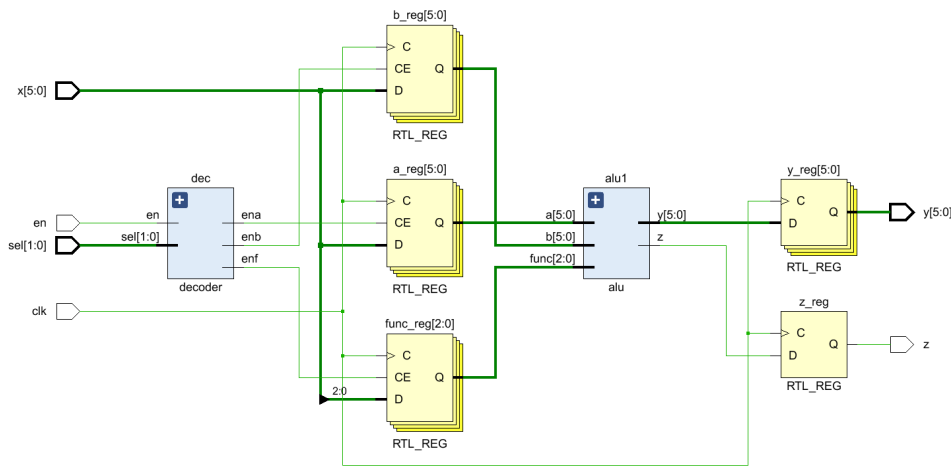
## 2. FLS 计算



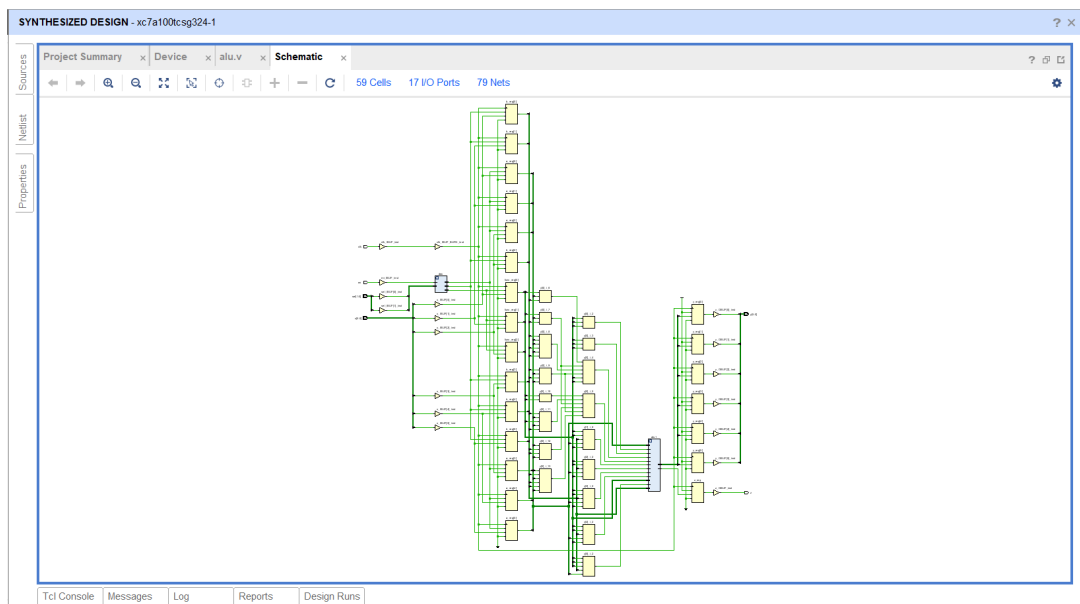
## 4. 电路设计与分析

### 1. ALU 设计

- RTL 电路图



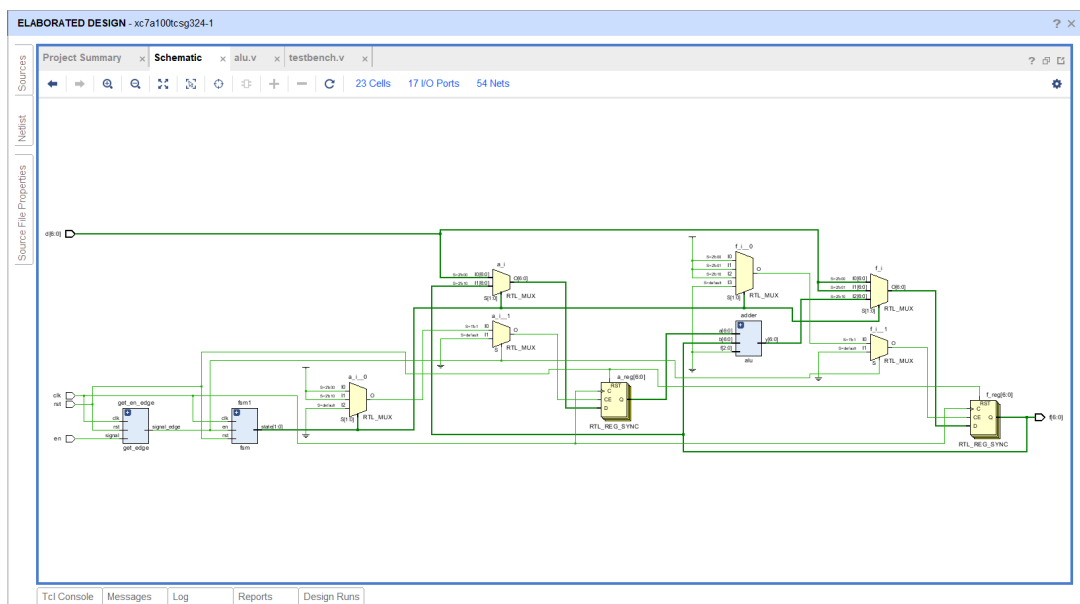
- 综合电路图



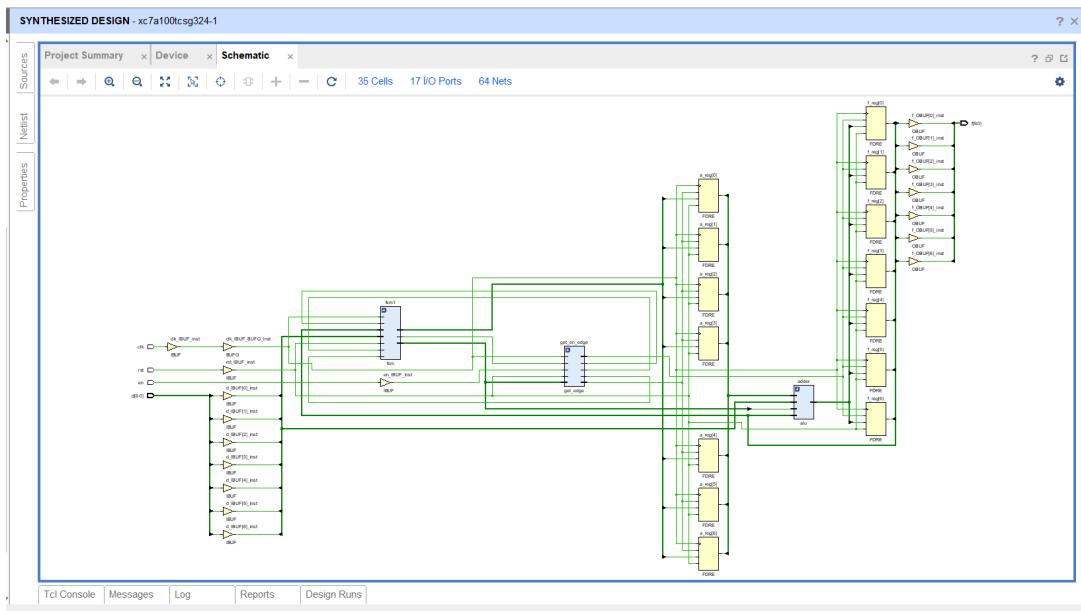
经对比，符合设计预期。

## 2. FLS 计算

- RTL 电路



- 综合电路



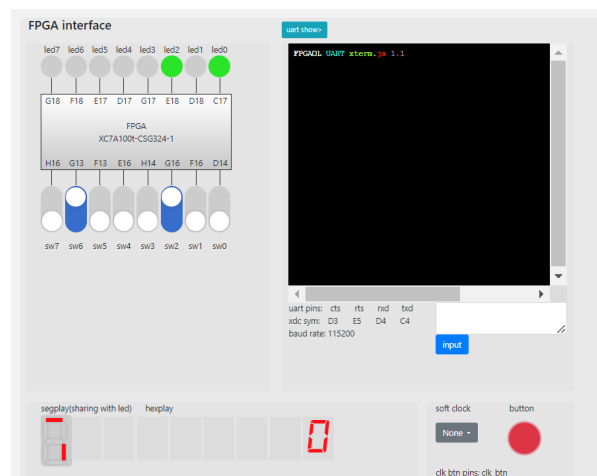
经对比，符合设计预期。

## 5. 测试结果与分析

### 1. ALU 设计

计算 $1 + 4 = 5$ 。

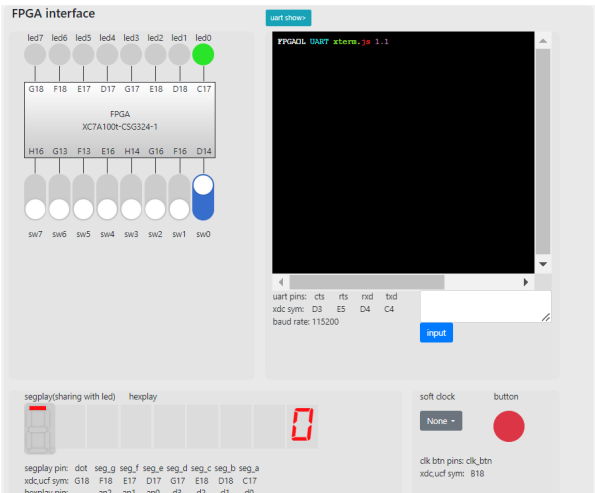
详细过程及其他功能在检查时展示。



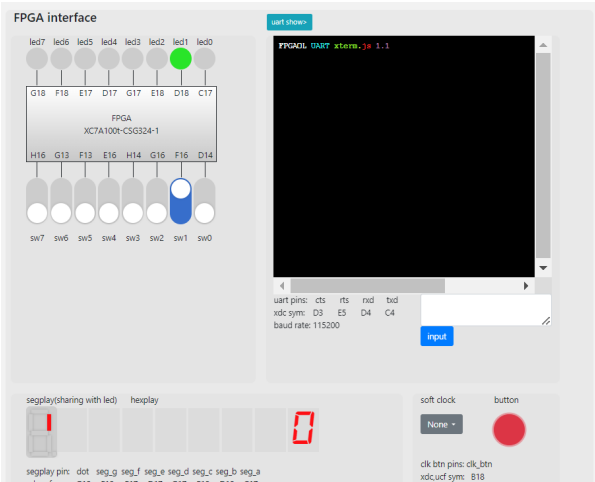
### 2. FLS 计算

依次输入1,

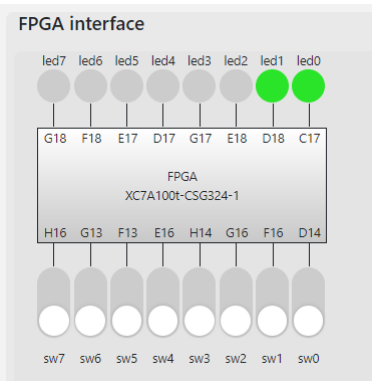


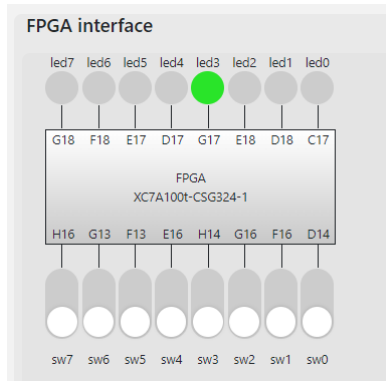


和2:



得到如下序列:





The screenshot shows the 'FPGA interface' window. On the left, a logic diagram is displayed with two rows of logic elements. The top row contains eight logic elements labeled 'led7' through 'led0'. The bottom row contains eight logic elements labeled 'H16' through 'H14'. Below these are eight switches labeled 'sw7' through 'sw0'. A blue circle is shown next to 'sw7', indicating it is turned on. In the center, the text 'FPGA' and 'XC7A100T-CSG324-1' are displayed. On the right, a terminal window titled 'uart show-' shows the command 'FPGA::UART xterm.js 1.1'. Below the terminal, the UART pins are listed: 'uart pins: cts rts nrd txd', 'xdc sym: D3 E5 D4 C4', and 'baud rate: 115200'. A blue button labeled 'input' is located at the bottom right.

最后，感谢助教阅读我的实验报告和建议！