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

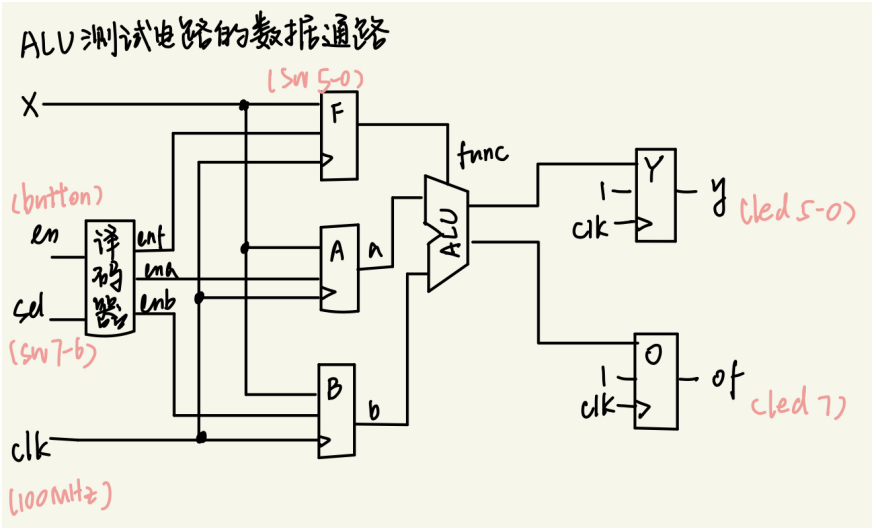
1. 算术逻辑单元 ALU

- 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 ^ b	0
1000	a >> b	0
1001	a << b	0
其他	0	0

- 数据通路

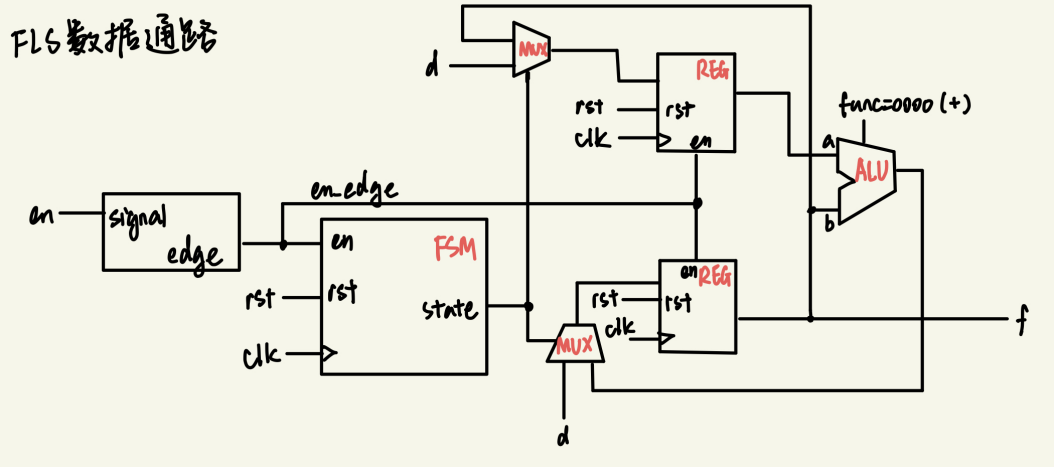


- 核心代码

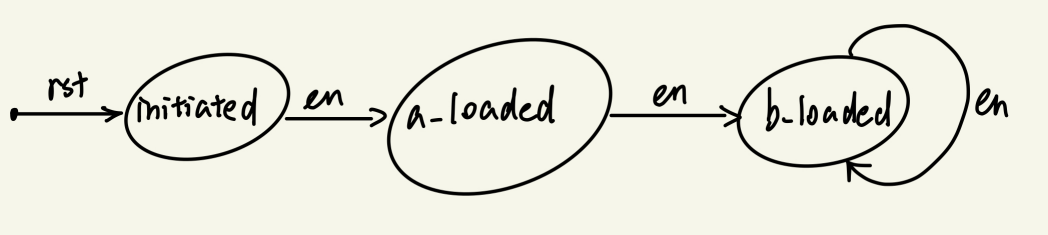
```
// ALU 模块 alu.v
module alu #(parameter WIDTH = 6) (
    input [WIDTH - 1:0] a, b,       // 操作数
    input [2:0] f,                  // 功能
    output reg [WIDTH - 1:0] y,     // 运算结果
    output z                        // 溢出标志of
);
    assign z = (y == {WIDTH{1'h0}}); // 如果有溢出，则为1，否则为0
    always @(*) 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
```

2. ALU 应用 - 计算 FLS

- 数据通路



- 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;
```

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

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

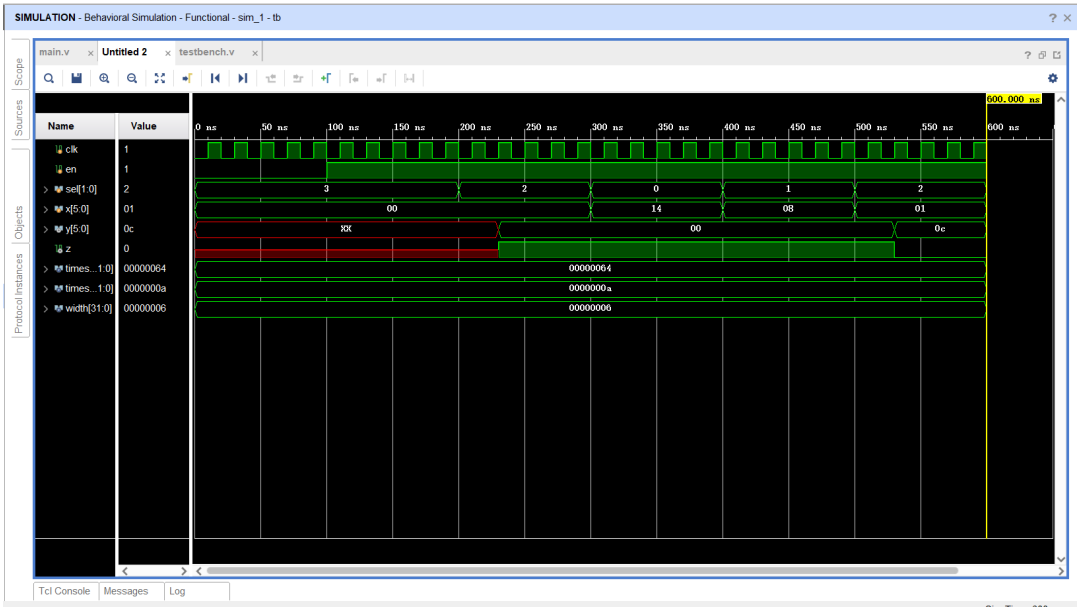
```
// 综合 fls.v
// 在 FSM 状态转移时刻 (en_edge 为高电平),
// ALU 输入处的寄存器被激活, 对应的信号被传入 ALU
module fls(
    input clk,
    input rst,
    input en,
    input [6:0] d,
    output reg [6:0] f
);
    // 布线
    // get edge
    wire en_edge;
    get_edge get_en_edge(
        .clk(clk),
        .rst(rst),
        .signal(en),
        .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;
```

```
        else if (en_edge) begin
            case (sel)
                2'b00: a <= d;
                2'b10: a <= f;
                default: a <= a;
            endcase
        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
```

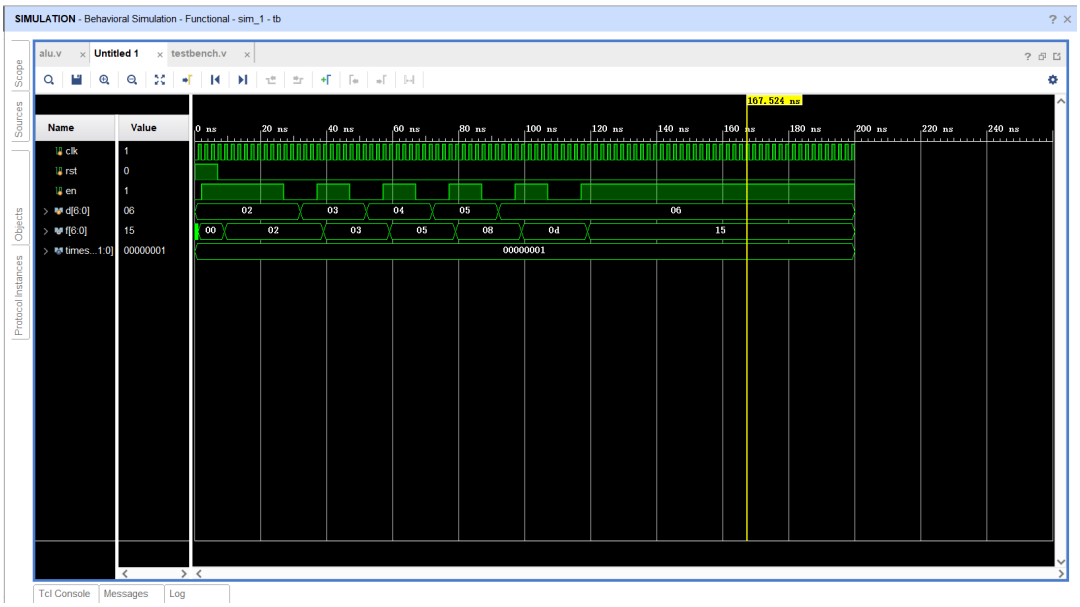
3. 仿真结果与分析

1. ALU 设计

由仿真文件生成的仿真波形如下：



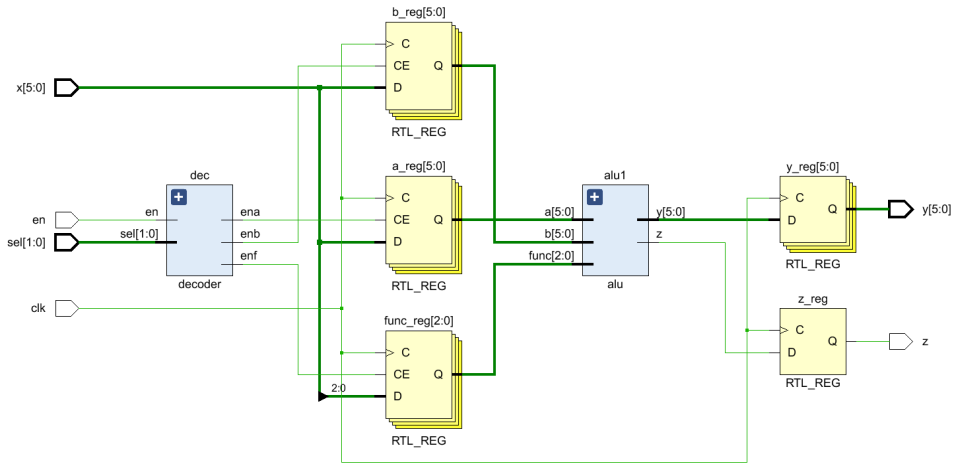
2. FLS 计算



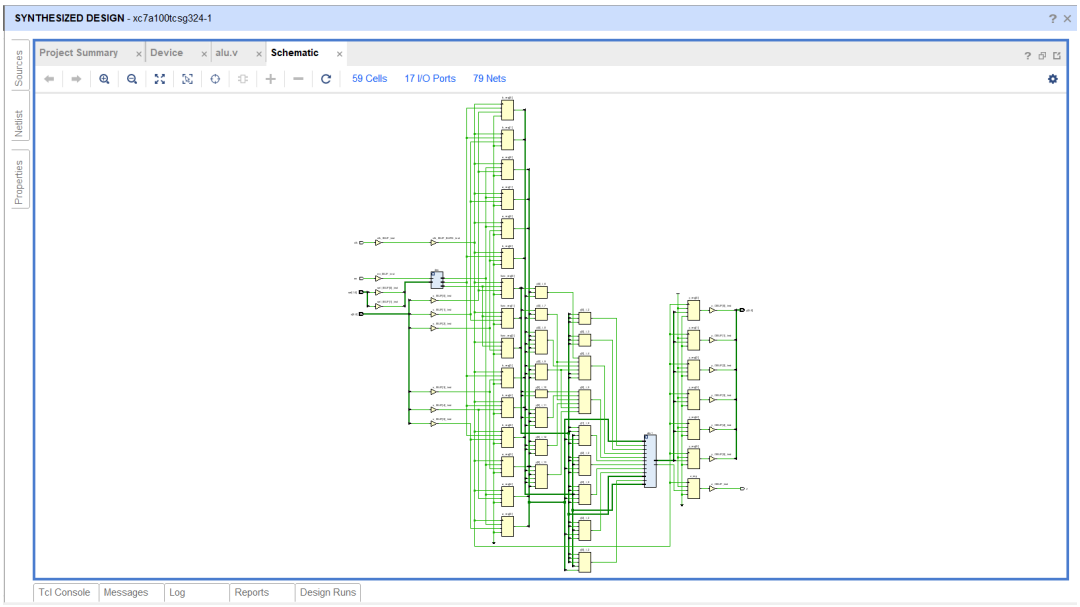
4. 电路设计与分析

1. ALU 设计

- RTL 电路图



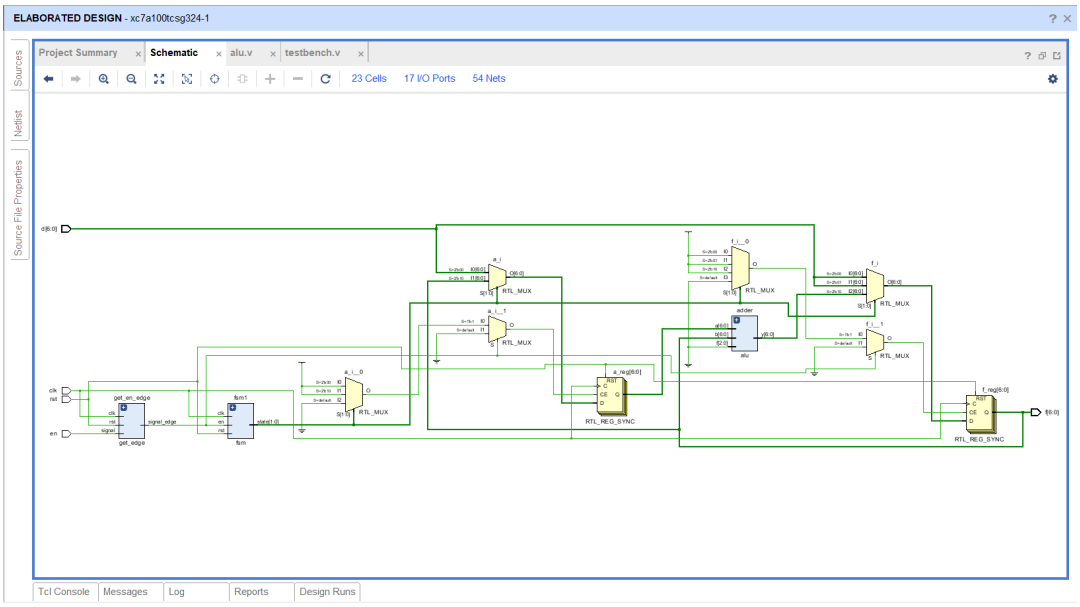
- 综合电路图



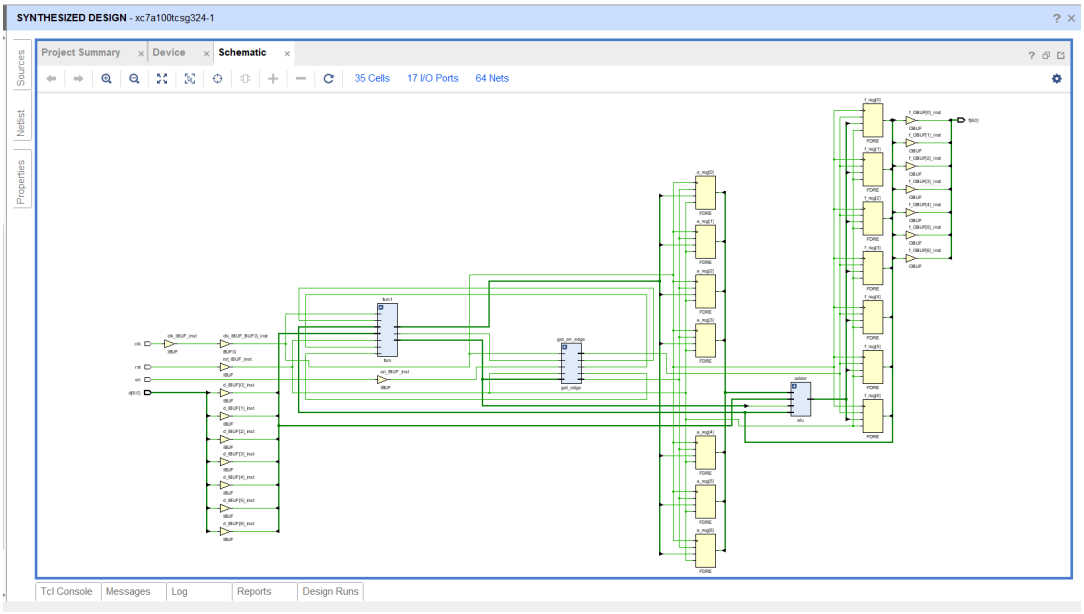
经对比，符合设计预期。

2. FLS 计算

- RTL 电路



- 综合电路



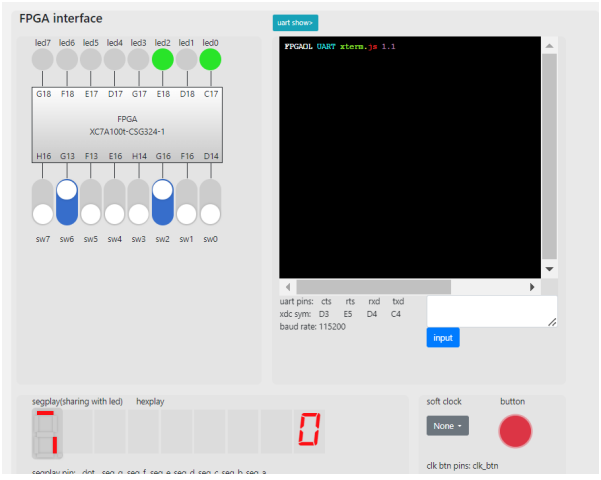
经对比，符合设计预期。

5. 测试结果与分析

1. ALU 设计

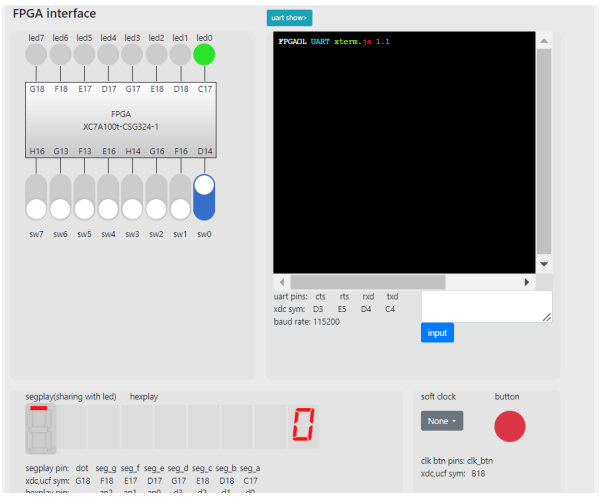
计算1 + 4 = 5.

详细过程及其他功能在检查时展示。

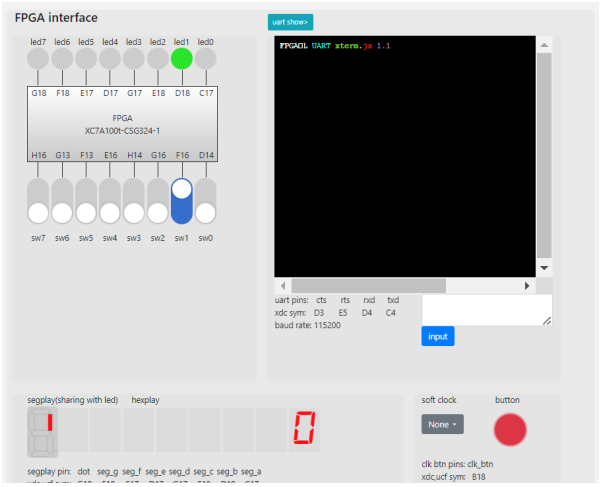


2. FLS 计算

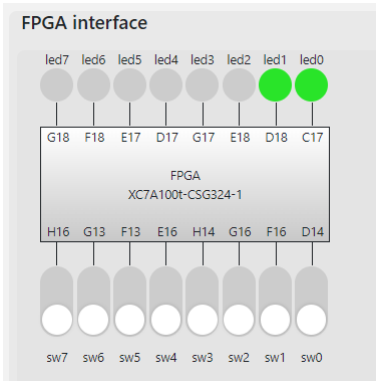
依次输入1，

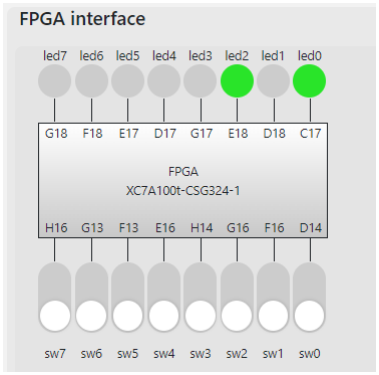

FPGA interface
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
uart show>
FPGADL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200
input
segplay(sharing with led)
hexplay
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18

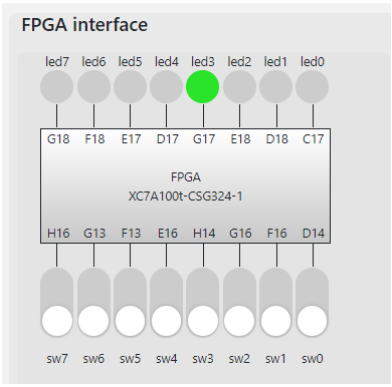
和2：


FPGA interface
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
uart show>
FPGADL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200
input
segplay(sharing with led)
hexplay
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18

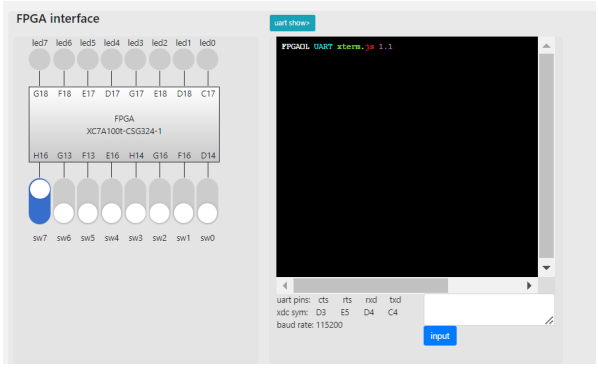
得到如下序列：


FPGA interface
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0





复位：



6. 总结

在本次实验中，我利用 verilog 语言和 vivado 软件完成了一个算术逻辑单元的设计，并利用它计算了斐波那契-卢卡斯数列。除了巩固 verilog 语言并强化了对仿真文件、有限状态机三段式的书写规范之外，本次实验也让我对 ALU 有了更深刻的理解。不过，在写仿真文件方面，由于不够熟练，我仍然遇到了不少困难。

在此想对助教和老师提出一个建议，就是希望以后在实验验收截止期后像公布作业答案一样公布一下一个标准的实验范例，因为根据以往学期的经验有些东西自己写不出来，提交之后拿完基础分就不了了之。如果担心有迟交的同学会抄答案，也希望助教们能在实验截止后做一份总结/反馈，私聊告诉有需求的同学。

最后，感谢助教阅读我的实验报告和建议！