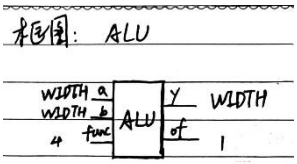
Lab 1 report

实验目的与内容

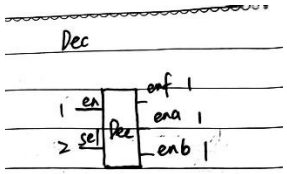
1. 本次实验主要完成了 alu 运算器模块的设计，并例化该模块使用译码器分时输入数据测试了该模块。使用该模块的加法功能完成了对斐波那契数列的计算。均在 vivado 中完成了仿真，最后上板实验。本次实验的目的为复习 verilog 的基础语法和例化方法，使用 vivado 进行仿真实验，使用 vlab 平台在线 fpga。

逻辑设计

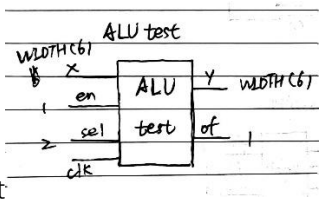
1. 框图和数据通路如下



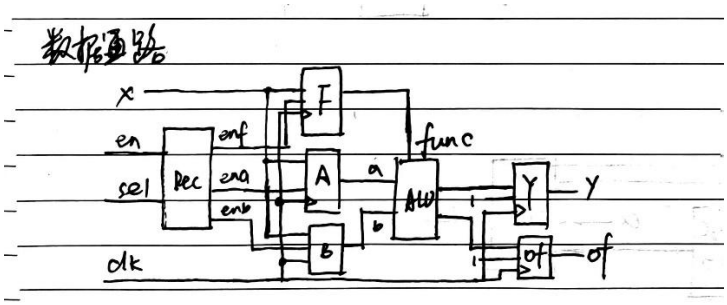
alu:



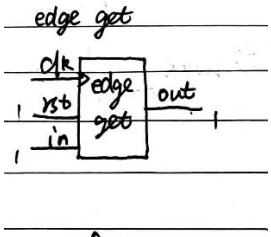
dec



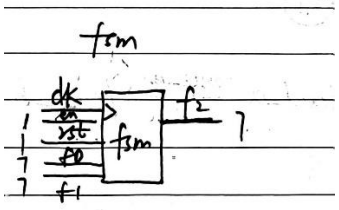
alutest



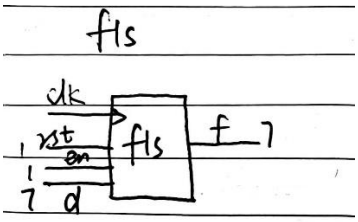
数据通路:



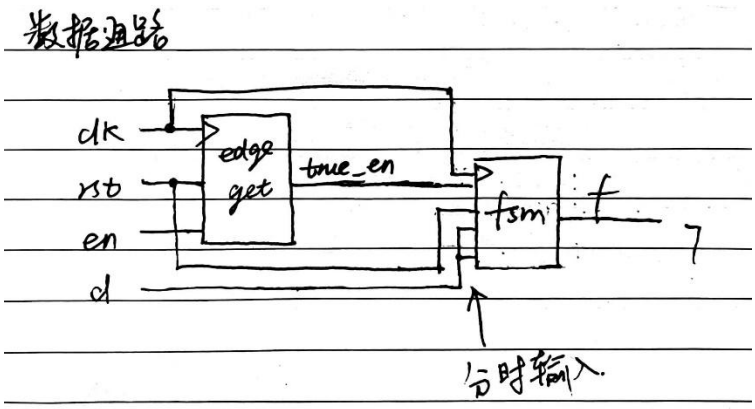
edgeget



fsm

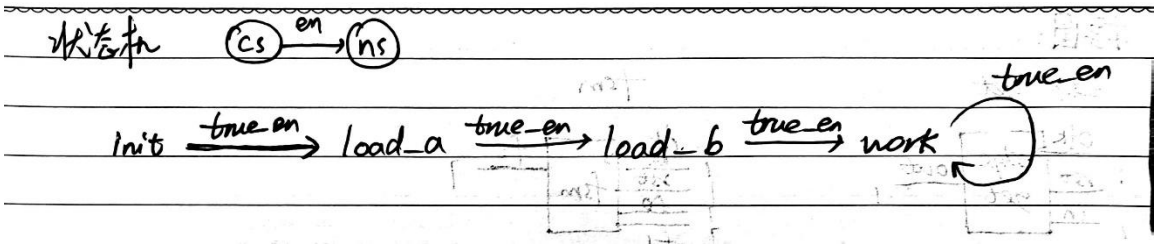


fls



数据通路：

2. 状态机在 fls 中有 fsm 为状态机：如图



3. 核心设计代码如下：

```
case(func)
    4'b0000: begin
        y = (a + b);
        if({a_s, b_s, y[WIDTH - 1]} == 3'b001)
            of = 1;
        if({a_s, b_s, y[WIDTH - 1]} == 3'b110)
            of = 1;
    end
    4'b0001: begin
        y = (a - b);
        if({a_s, b_s, y[WIDTH - 1]} == 3'b011)
            of = 1;
        if({a_s, b_s, y[WIDTH - 1]} == 3'b100)
            of = 1;
    end
    4'b0010:
        y = (a == b);
    4'b0011:
        y = (a < b);
    4'b0100:
        y = ($signed(a) < $signed(b));
    4'b0101:
        y = (a & b);
    4'b0110:
        y = (a | b);
    4'b0111:
        y = (a ^ b);
    4'b1000:
        y = (a >> b);
    4'b1001:
        y = (a << b);
    default:
        y = 0;
endcase
```

对于溢出标志 of，采用取 a,b,y 的最高位，通过罗列溢出的各种可能性，用==判断语句描述溢出。

译码器如下：

```
module decoder (            // 译码器
    input en,
    input [1:0] sel,
    output enf, ena, enb
);
    assign ena = en & (sel == 2'b00);
    assign enb = en & (sel == 2'b01);
    assign enf = en & (sel == 2'b10);
endmodule
```

简单的 assign 赋值。

取边沿代码沿用之前的设计：

```
module edgeget (        // 取边沿模块
    input clk, rst,
    input in,
    output out
);
    reg out1, out2;
    always @(posedge clk) begin
        out1 <= ~rst & in;
    end
    always @(posedge clk) begin
        out2 <= out1;
    end
    assign out = out1 & ~out2;
endmodule
```

保证每次生效均在时钟上升沿。

三段式如下：

```
//描述cs
always @(posedge clk, posedge rst) begin    // 当前状态的状态转换
    if(rst)
        cs <= init;
    else
        cs <= ns;
end

always @(posedge clk) begin                 // 当前状态的赋值
    if (true_en)
        case(cs)
            init: begin
                f0 <= d;
            end
            load_a: begin
                f1 <= d;
            end
            load_b: begin
            end
            work: begin
                f0 <= f1;
                f1 <= f2;
            end
        endcase
end
```

对当前状态进行赋值

```
//描述NS
always @(*) begin                                   // 次态的状态转换
    ns = cs;
    case(cs)
        init: begin
            if(true_en) begin
                ns = load_a;
            end
        end
        load_a: begin
            if(true_en) begin
                ns = load_b;
            end
        end
        load_b: begin
            if(true_en) begin
                ns = work;
            end
        end
    endcase
end
```

状态机的状态转换

```
alu #(7) alu1(                                      // 调用alu加法模块
    .a(f1_w),
    .b(f0_w),
    .func(4'b0000),
    .y(f2),
    .of(of)
);

//描述输出
assign f = (cs == work) ? f2 : ((cs == load_b) ? f1 :(cs == load_a ? f0  : 0));
```

对当前态处于哪种状态进行判断得到输出。

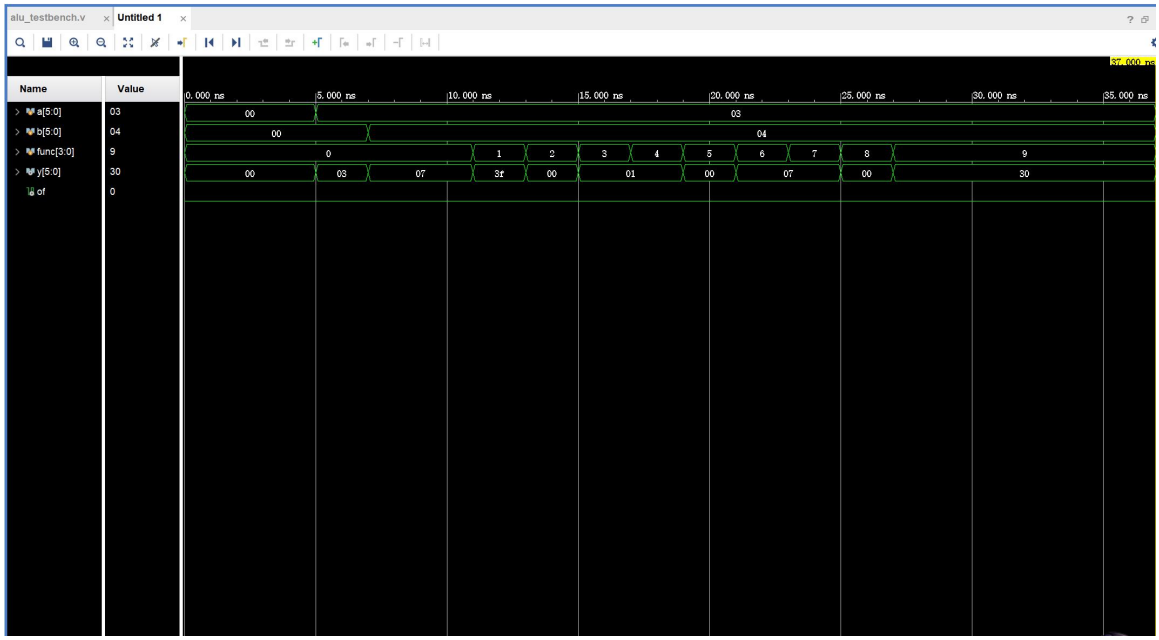
仿真结果与分析

1. 仿真文件截图和仿真结果解释：

alutest 仿真：

```
module alu_testbench(

    );
    reg [5:0] a = 0;
    reg [5:0] b = 0;
    reg [3:0] func = 0;
    wire [5:0] y;
    wire of;
    initial begin
        #5 a = 7'd3;
        #2 b = 7'd4;
        #2 func = 4'b0000;
        #2 func = 4'b0001;
        #2 func = 4'b0010;
        #2 func = 4'b0011;
        #2 func = 4'b0100;
        #2 func = 4'b0101;
        #2 func = 4'b0110;
        #2 func = 4'b0111;
        #2 func = 4'b1000;
        #2 func = 4'b1001;
        #10 $finish();
    end
    alu alu1(
        .a(a),
        .b(b),
        .func(func),
        .y(y),
        .of(of)
    );
endmodule
```

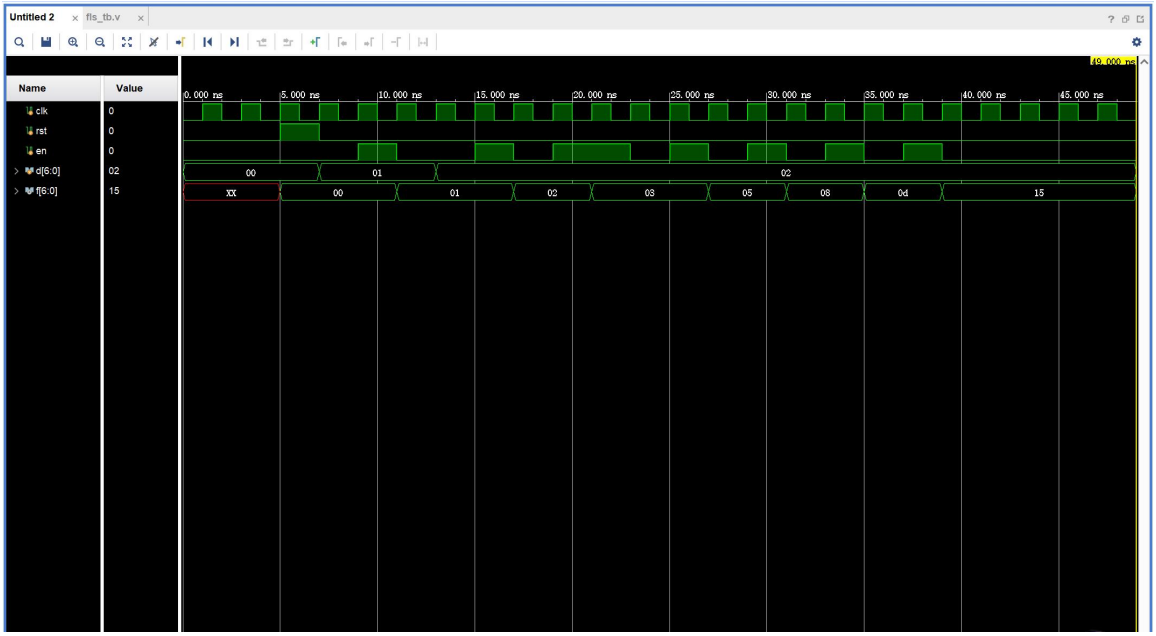
func 初始为 0 时，当 a 输入为 3，b 为 4，结果为 7，of 标志不溢出，当 func 为 1 时结果为-1，6 位补码表示为 3f，同理检查之后的 func，均无误。



fls 仿真：

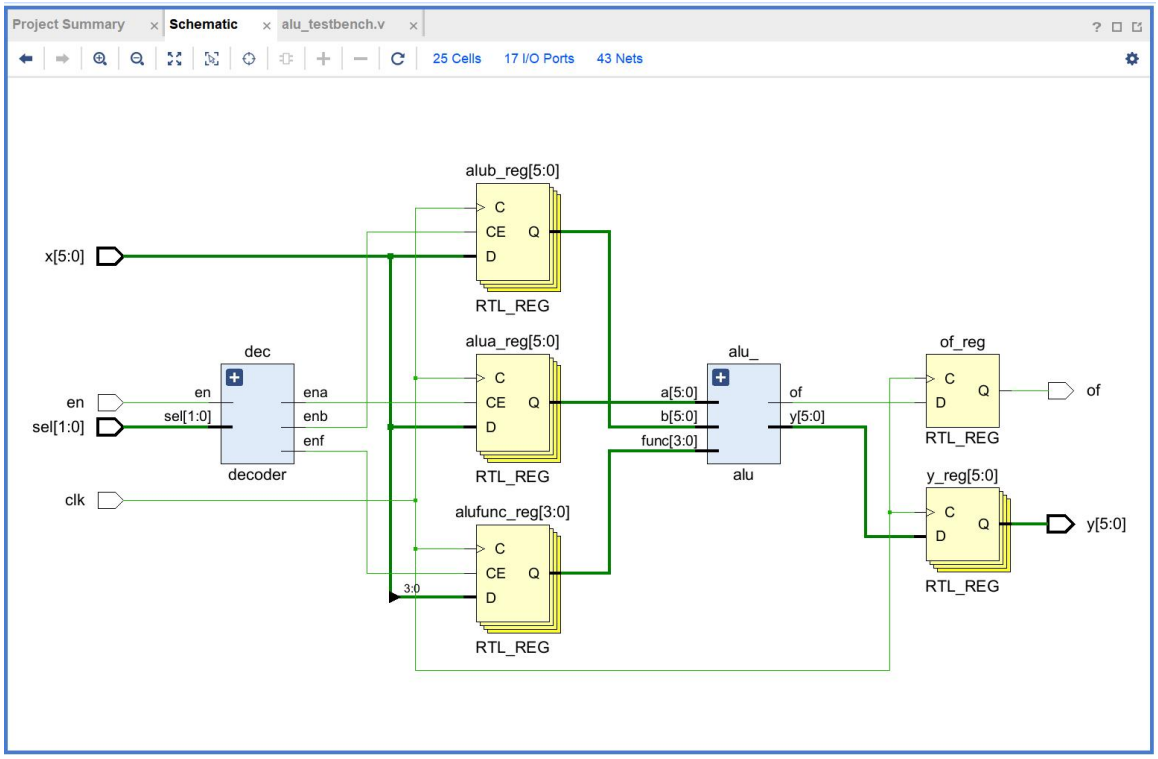
```
module tb1(

    );
    reg clk = 0;
    always #1 clk = ~clk;
    reg rst = 0;
    reg en = 0;
    reg [6:0] d = 0;
    wire [6:0] f;

    initial begin
        #(5) rst = 1;
        #2 rst = 0;
            d = 7'd1;
        #2 en = 1;
        #2 en = 0;
        #2 d = 7'd2;
        #2 en = 1;
        #2 en = 0;
        #2 en = 1;
        #4 en = 0;
        #2 en = 1;
        #2 en = 0;
        #2 en = 1;
        #2 en = 0;
        #2 en = 1;
        #2 en = 0;
        #2 en = 1;
        #2 en = 0;

        #10 $finish();
    end

    fls s(
        .clk(clk),
        .rst(rst),
        .en(en),
        .d(d),
        .f(f)
    );

endmodule
```

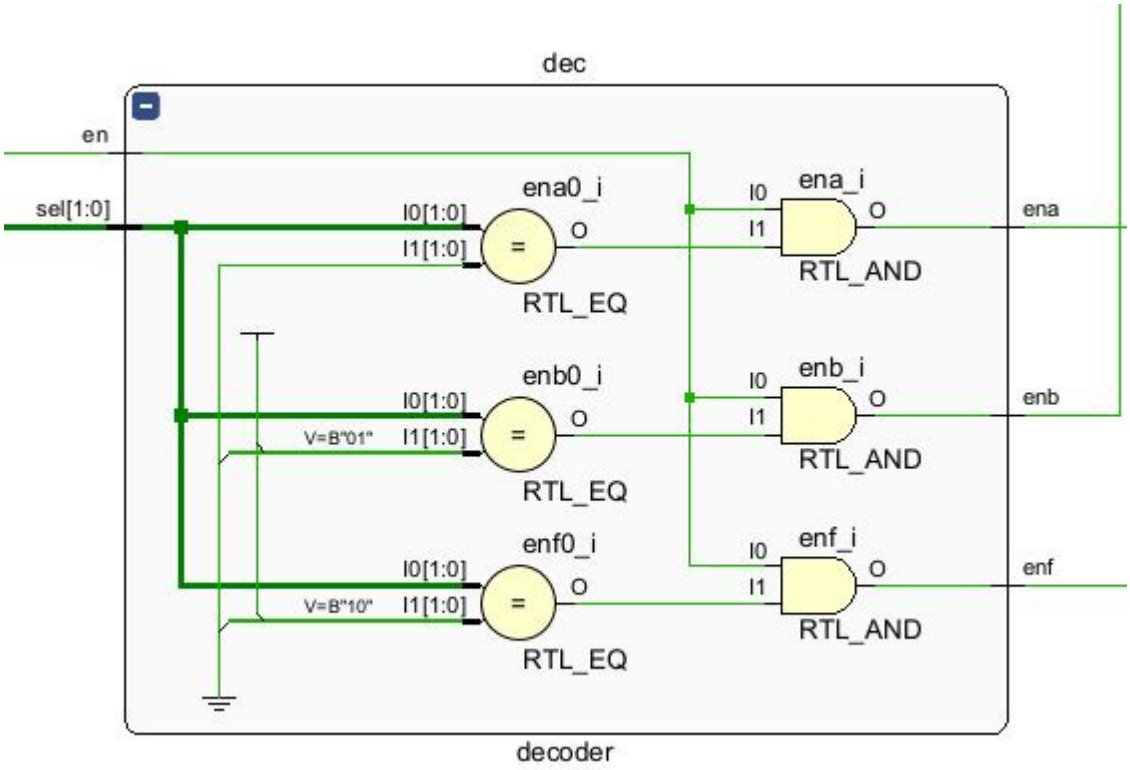
首先 rst，在 rst 之后适时载入第一个数字和第二个数字，之后分别隔可能不一样的时间间隔使 en 有效。仿真结果观察到在 en 有效时，只有当一个时钟上升沿时进行数据的载入或者是数据的计算，载入 1,2 之后得到后续的计算结果也无误，满足要求。

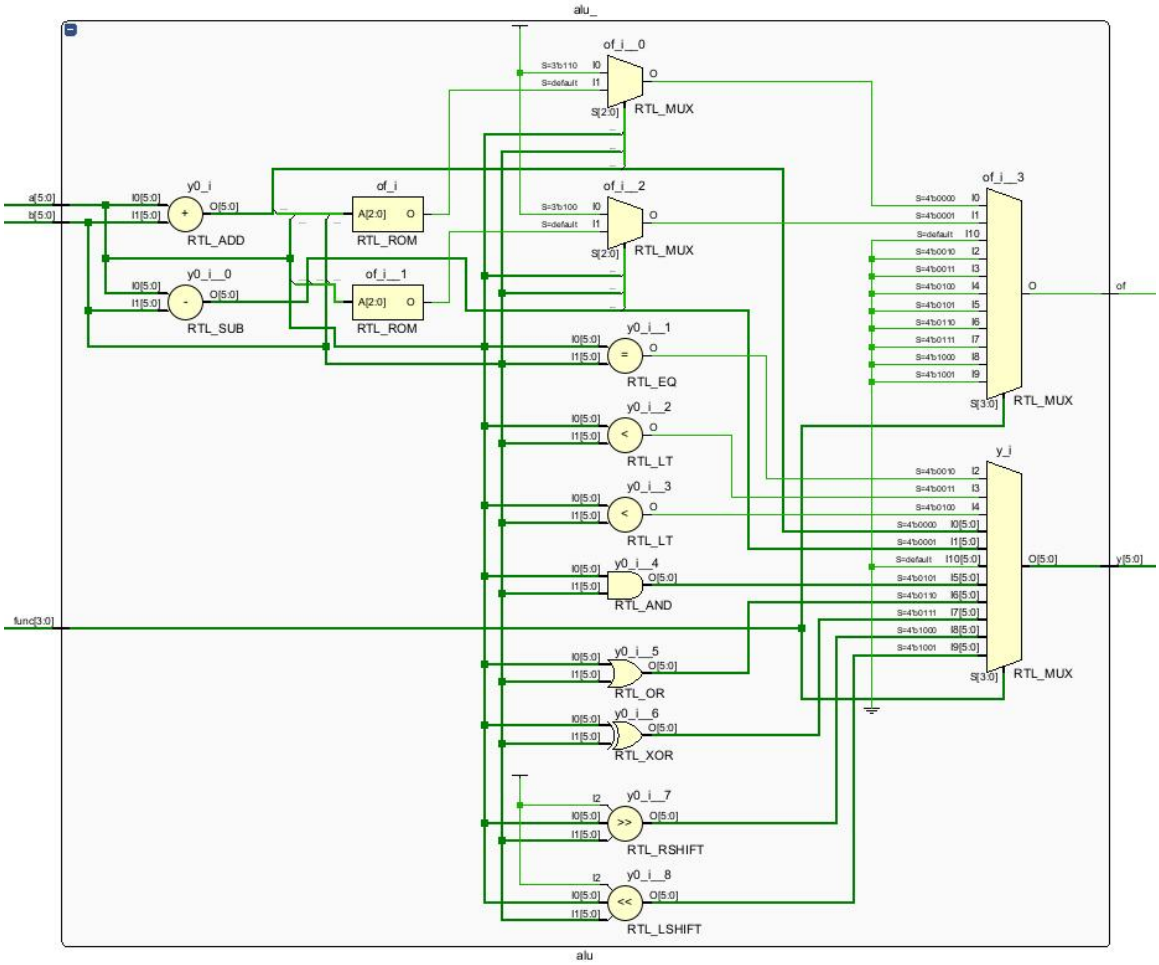


电路设计与分析

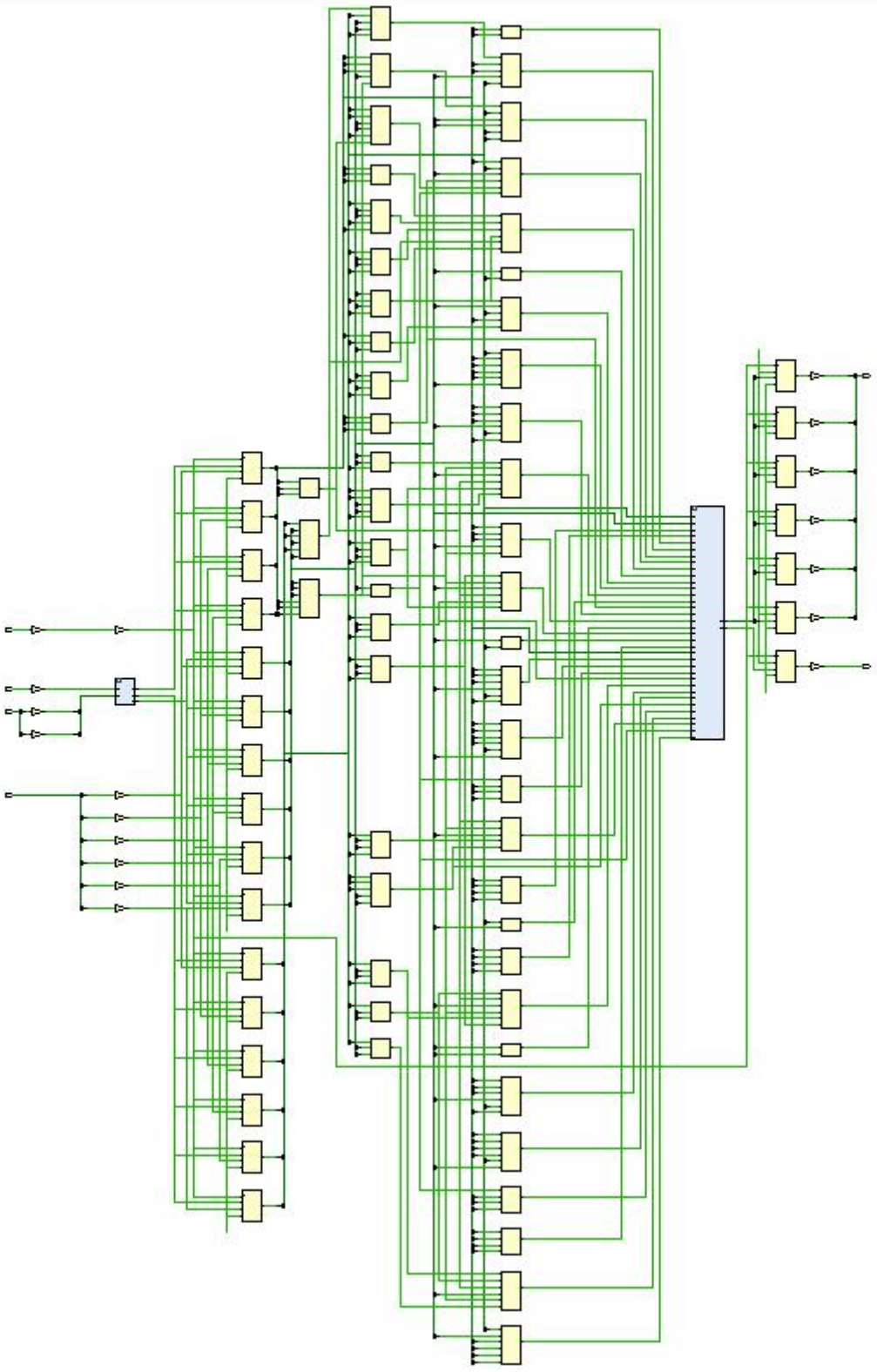
1. RTL 电路图如下：



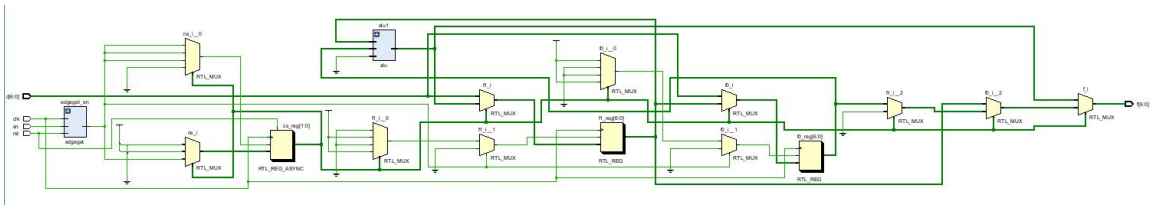


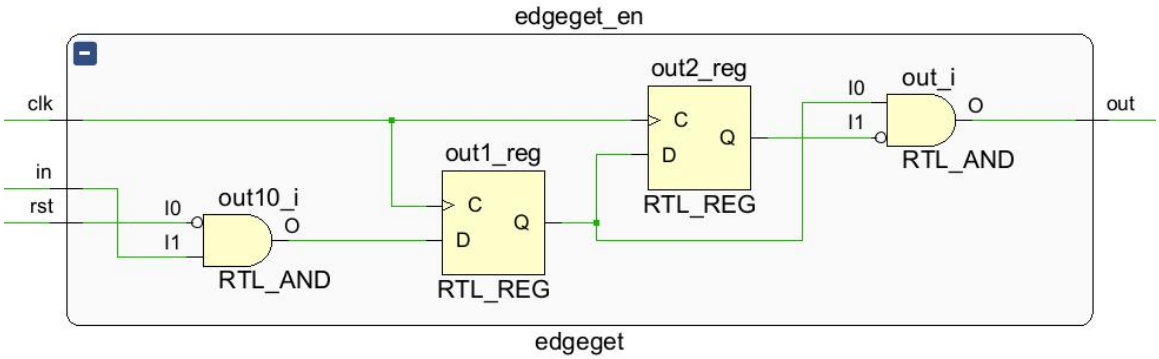


alutest 综合后电路如下:

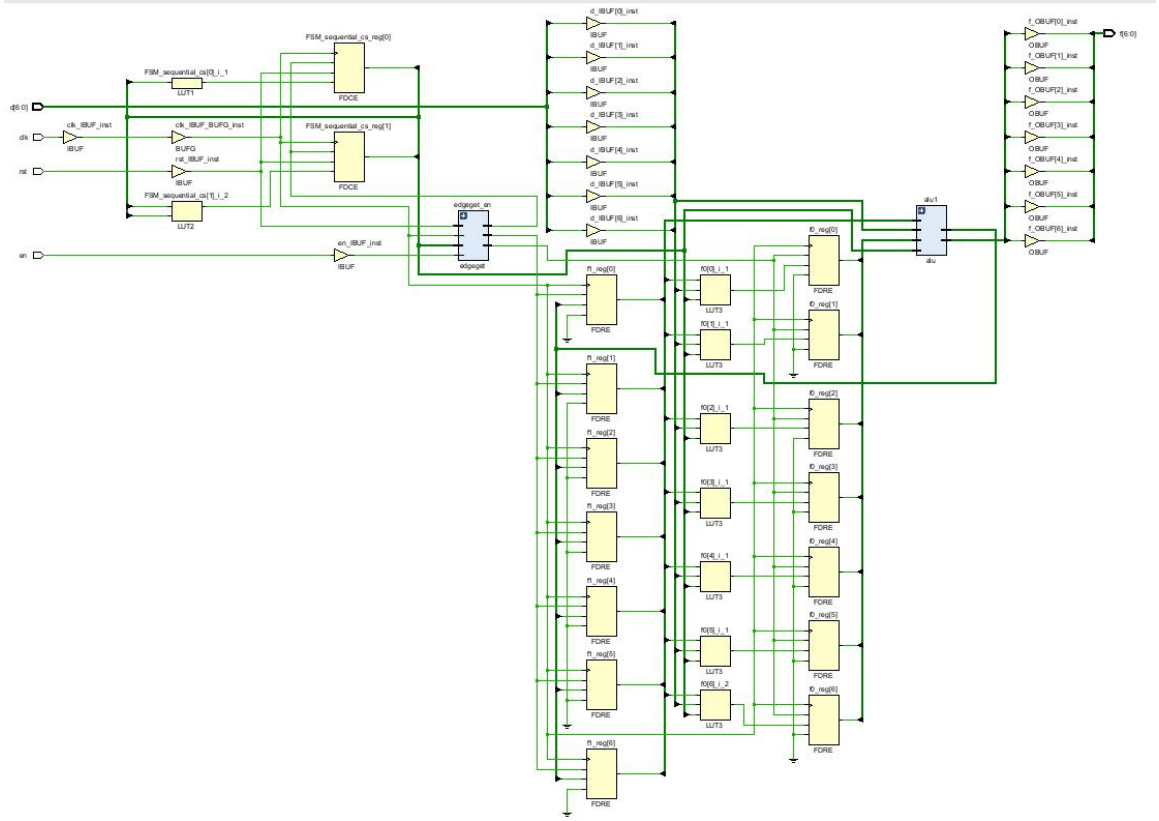


fls 的 RTL 电路如下：




edgeget_en
clk
in
rst
out
out10_i
I0
I1
O
RTL_AND
out1_reg
C
D
Q
RTL_REG
out2_reg
C
D
Q
RTL_REG
I0
I1
out_i
O
RTL_AND
edgeget

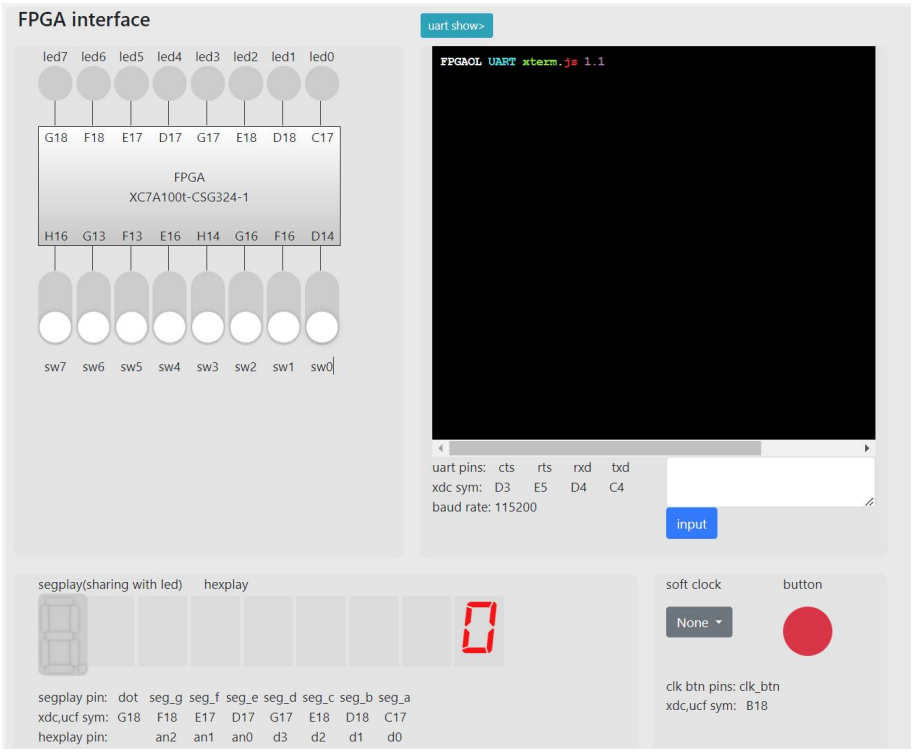
综合后电路如下:



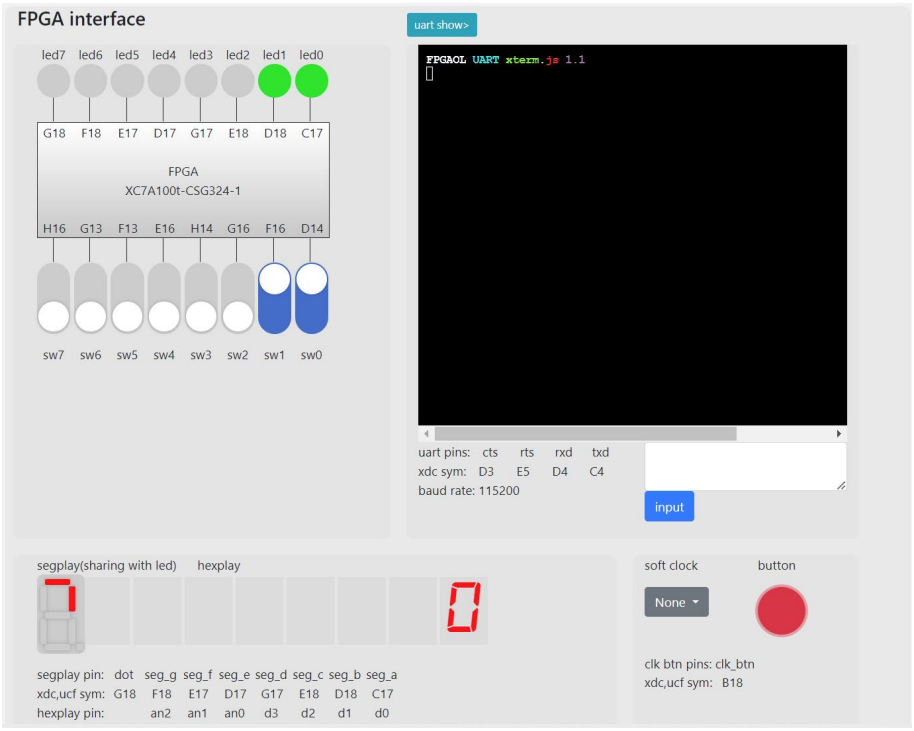
测试结果与分析

1. alutest 上板实验:

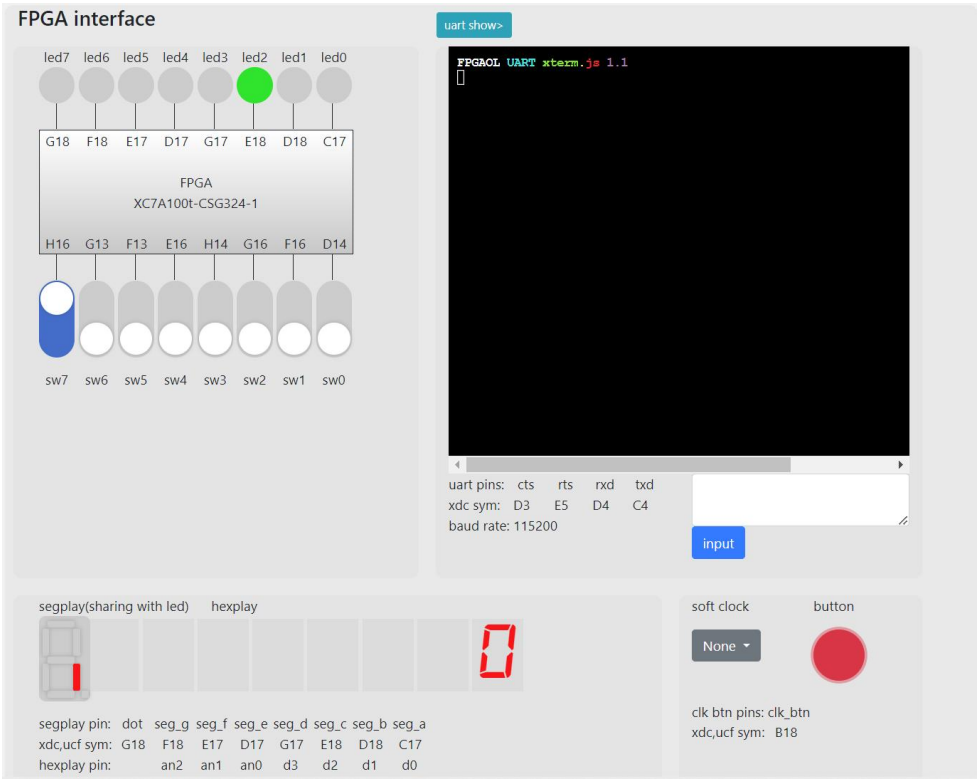
初始:



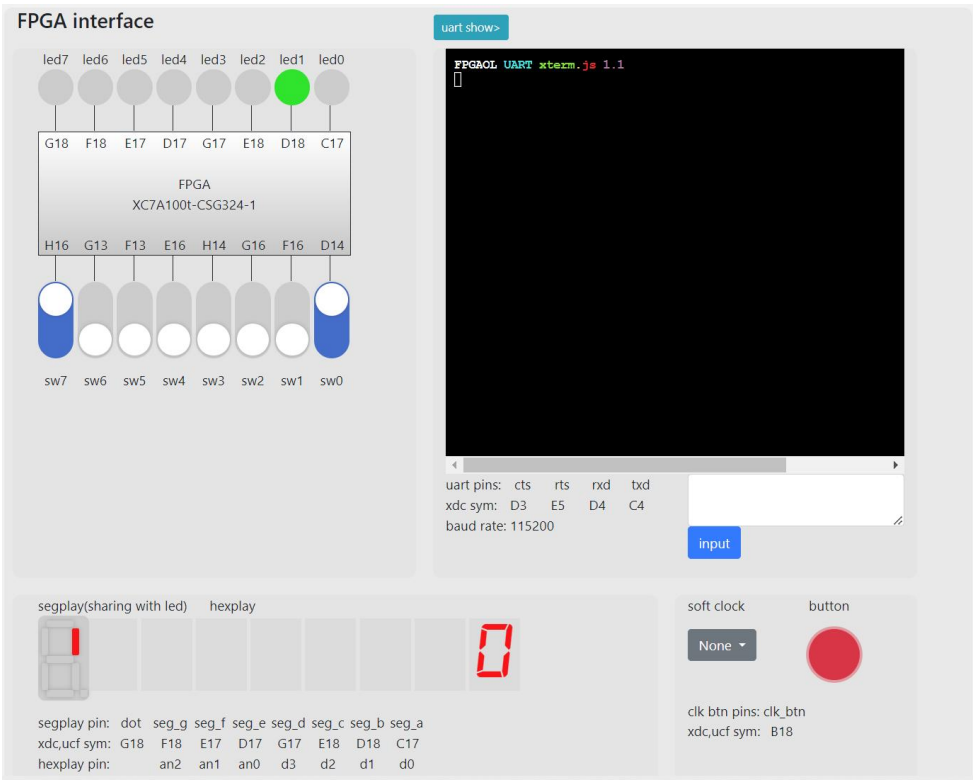
载入 a 为 3:



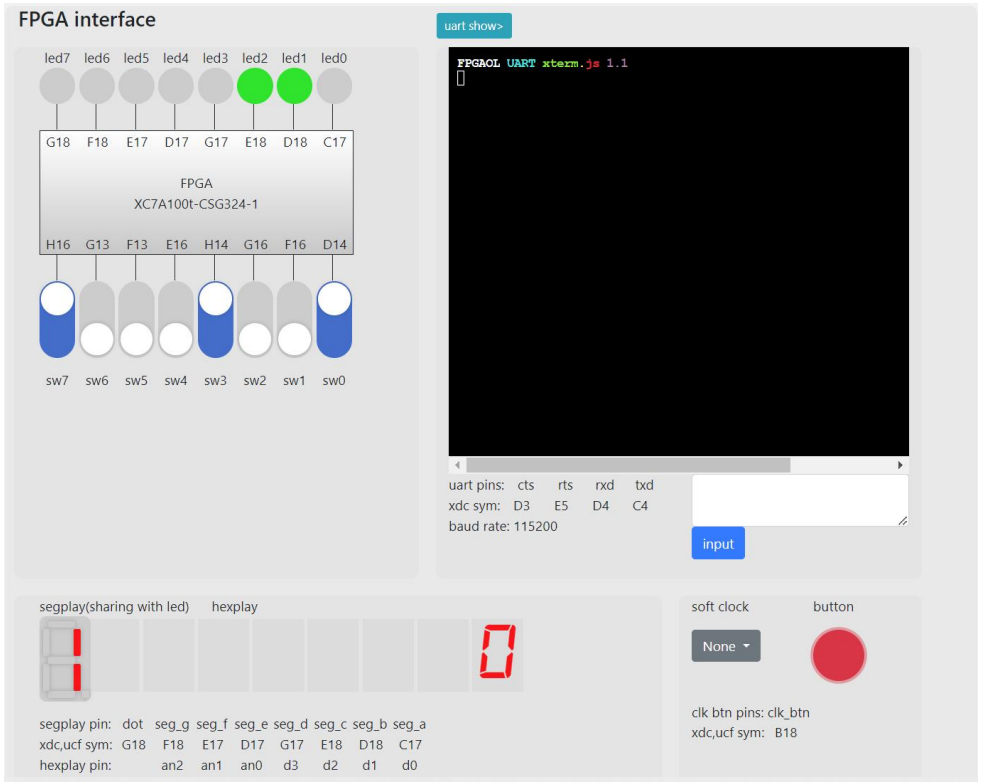
载入 b 为 1 同时由于 func 默认为 0000 完成加法得到结果 4:



载入 func 为 0001 测试减法得到结果 2:

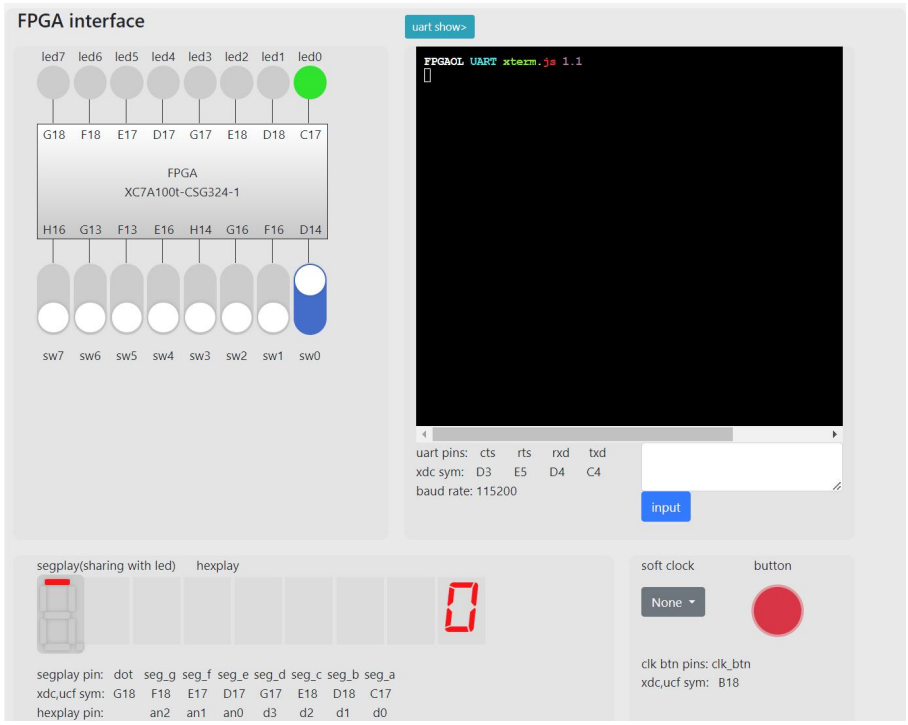


载入 func 为 1001 进行左移操作得到的结果如下,即从 000011 变为 000110:

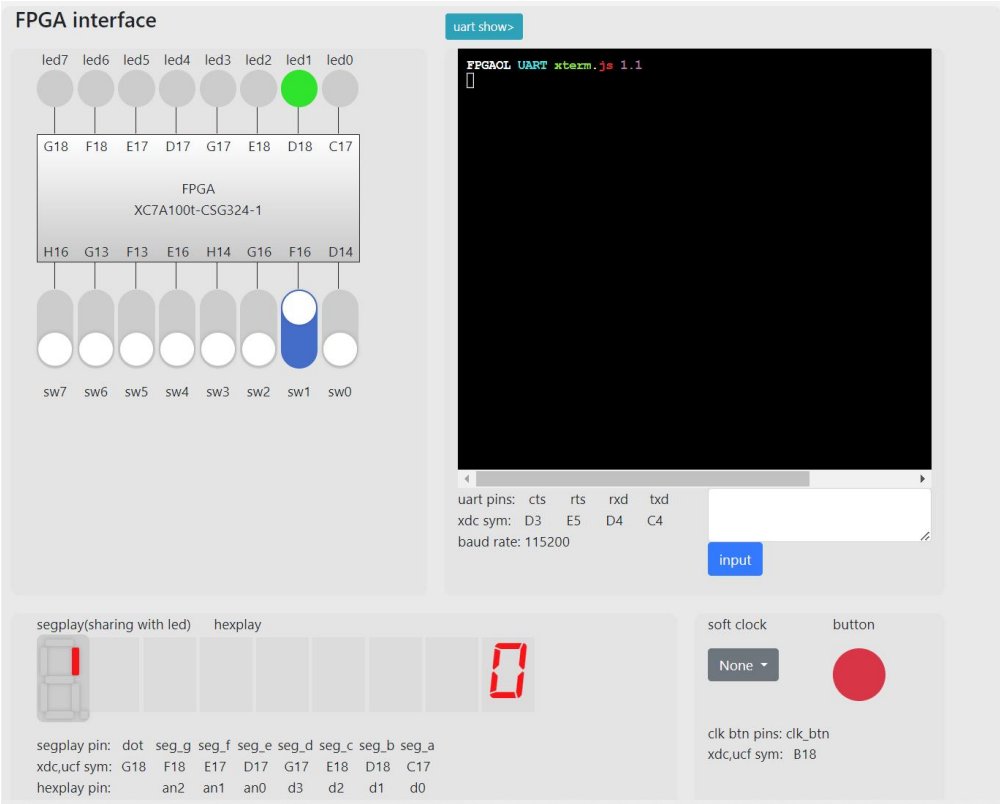


对 fls 上板测试:

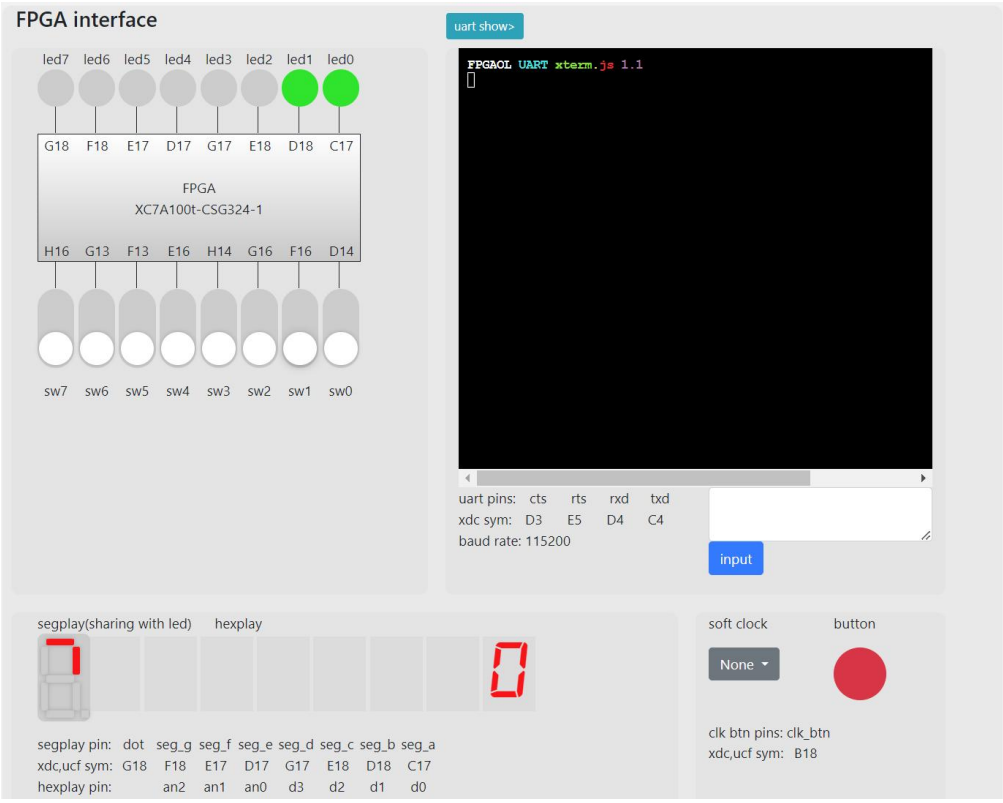
载入第一个 d 为 1:

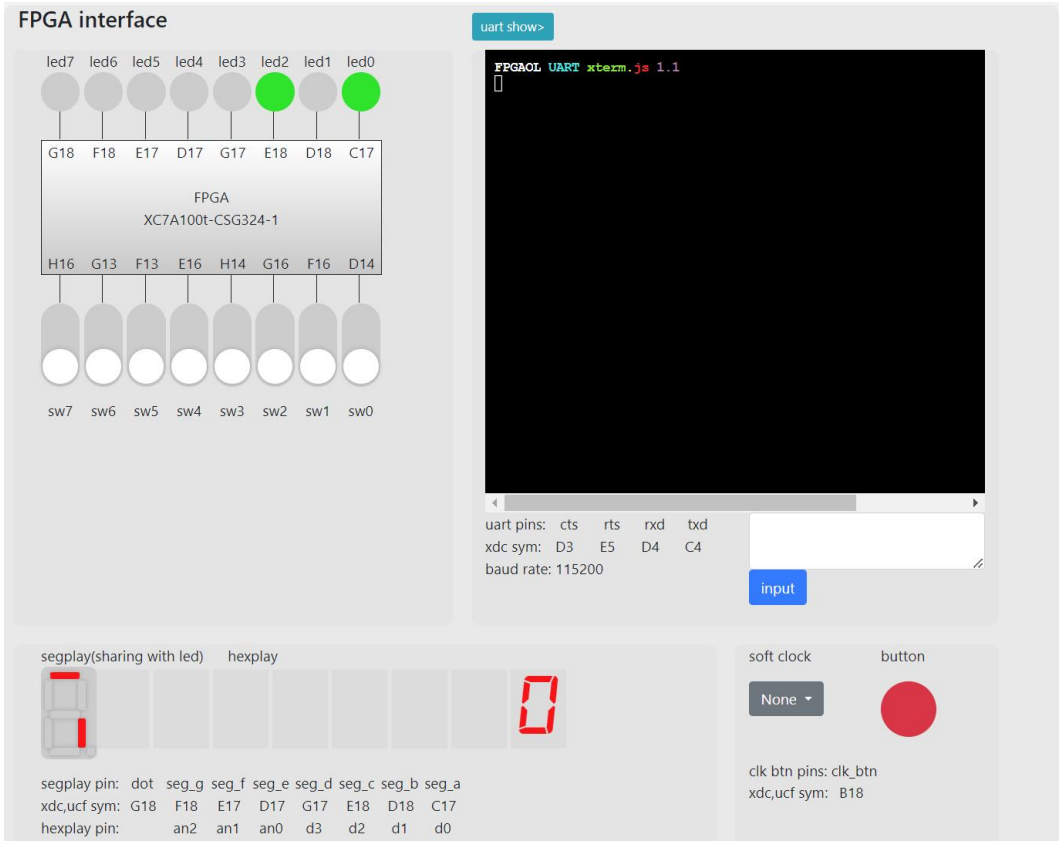


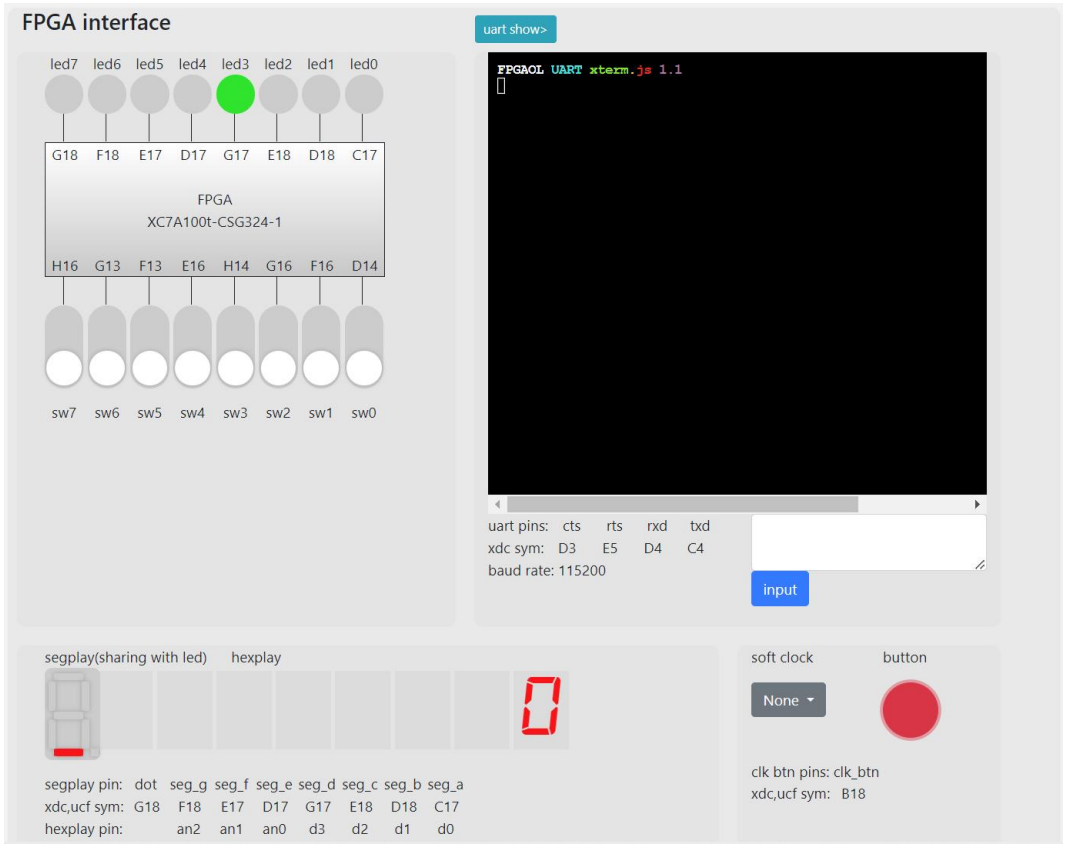
载入第二个 d 为 2:

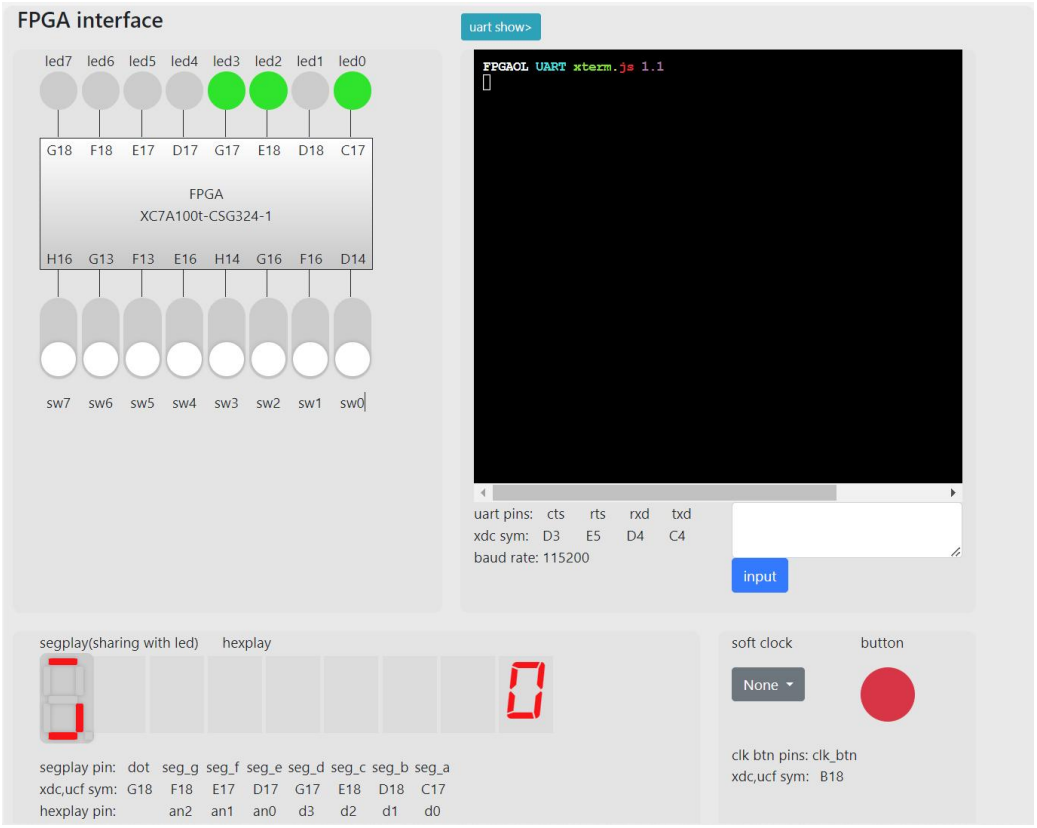


反复按压 button 得到之后的数列为 1 2 3 5 8 13:

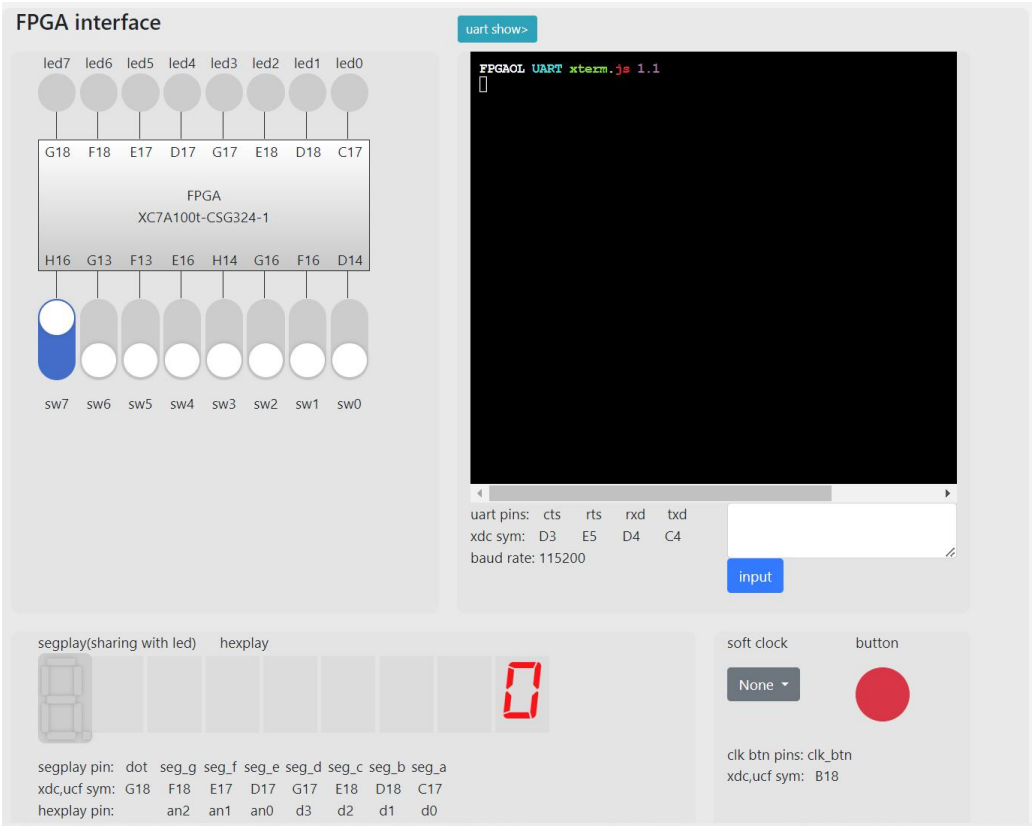



FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
FPGAOL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200
input
segplay(sharing with led)
hexplay
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18


FPGA interface
uart show>
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
FPGAOL 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
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18



按一次 sw7 复位：（重新载入也成功）



总结

1. 本次实验我完成了 alu 的设计和测试，利用加法模块编写代码并上板计算了 fls。再次对 verilog 进行了熟悉和了解，使用 vivado 仿真和生成 RTL 电路，并在 fpga 平台完成了烧写。
2. 实验文档全面，实验讲解翔实。