计算机组成原理 实验报告

姓名：宋玮　学号：PB20151793　实验日期：2022.3.22

一、实验题目：

Lab01 运算器及其应用

二、实验目的：

设计一算术逻辑运算单元（ALU），实现两数的加、减、与、或、异或操作；利用前述的 ALU 模块与适当的硬件电路，完成 6 位操作数 ALU 和计算斐波那契—卢卡斯数列（FLS）功能。

三、实验平台：

Vivado, fpgaol

四、实验过程及结果：

如下所示:

- **算术逻辑单元（ALU 模块）**

1. 32 位操作数 ALU

（1）根据给出的功能进行设计，该模块 verilog 代码如下：

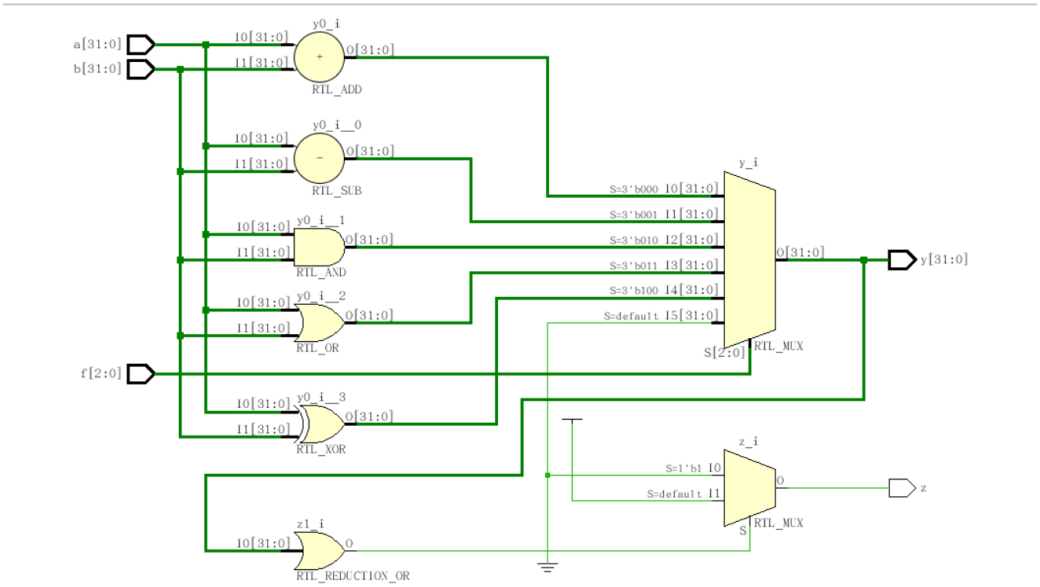
```
module alu #(parameter WIDTH = 32)
(
    input[WIDTH-1:0] a,
    input[WIDTH-1:0] b,
    input [2:0] f,
    output reg [WIDTH-1:0] y,
    output z
);

always@(*)
begin
    case(f)
        3'b000: y = a + b;
        3'b001: y = a - b;
        3'b010: y = a & b;
```

```
            3'b011: y = a | b;
            3'b100: y = a ^ b;
            default: y = 0;
        endcase
end

assign z = y ? 1'b0 : 1'b1;

endmodule
```

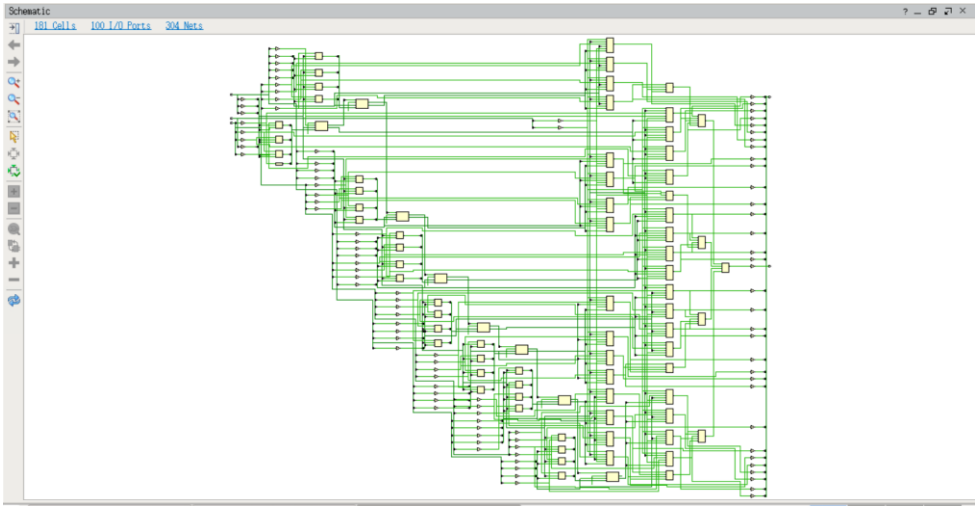
（2）查看 ALU 模块 vivado 生成电路和资源使用情况

①RTL 电路

如下图：

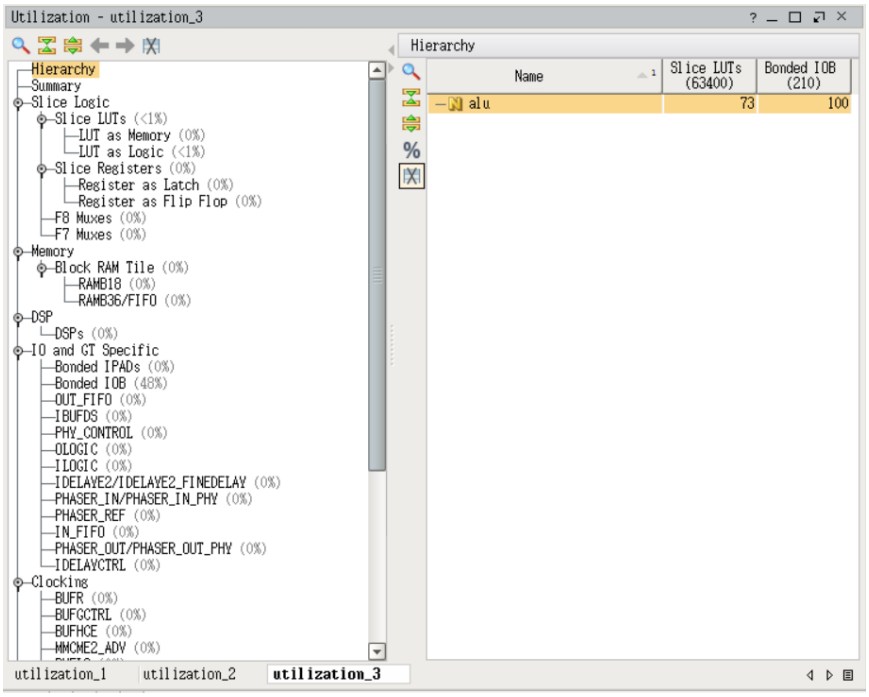


②综合电路

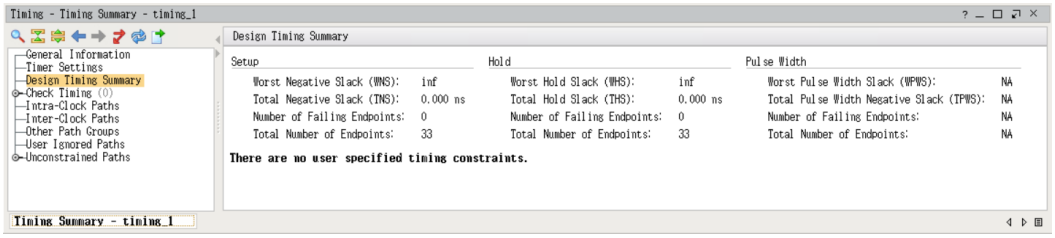


③资源利用情况

资源利用报告如下:



④时间性能报告



2. 6 位操作数 ALU

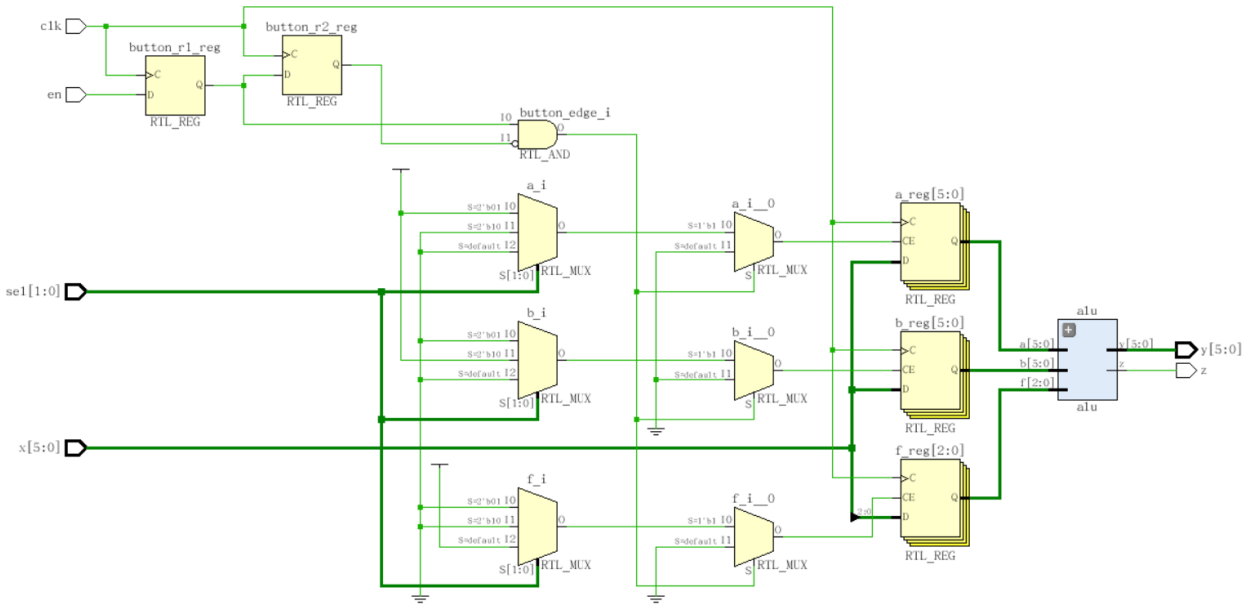
（1）verilog 设计代码如下：

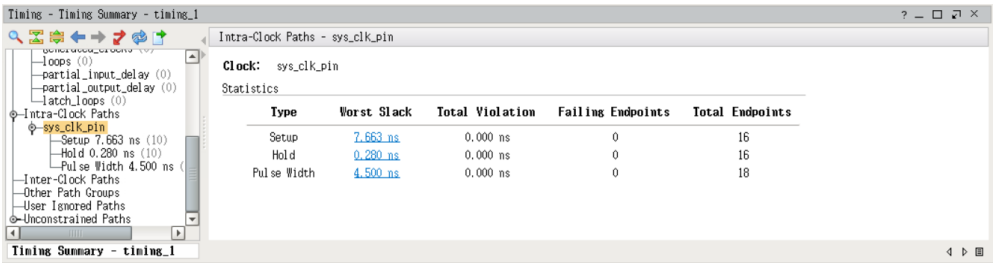
注：设计中使用了去抖动和前面编写的 alu 模块

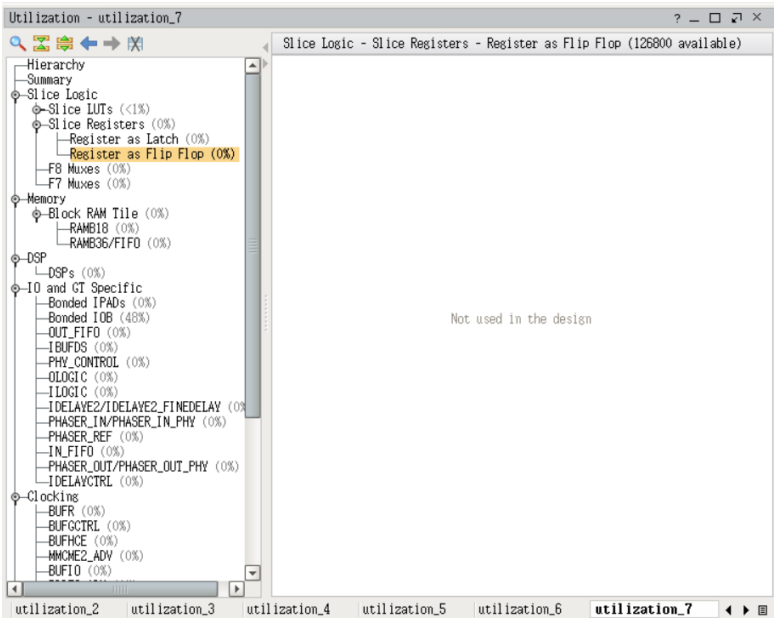
```
module alu6(
input clk,
input en,
input [1:0] sel,
input [5:0] x,
output [5:0] y,
output z
);
```

```
reg [5:0] a;
reg [5:0] b;
reg [2:0] f;
reg button_r1,button_r2;
wire button_edge;

always@(posedge clk)
    button_r1 <= en;
always@(posedge clk)
    button_r2 <= button_r1;
assign button_edge = button_r1 & (~button_r2);

always@(posedge clk)
if(button_edge)
    case(sel)
        2'b00: a <= x;
        2'b01: b <= x;
        default: f <= x[2:0];
    endcase

alu #(6) alu(a,b,f,y,z);
endmodule
```

（2）查看 RTL 电路图，以及实现电路资源和时间性能报告







（3）仿真文件（simulation）

```
module alu_simulation();
        reg clk;
        reg en;
        reg [1:0] sel;
        reg [5:0] x;
        wire [5:0] y;
        wire z;

        alu6 alualu(clk,en,sel,x,y,z);

        initial clk = 0;
        always #5 clk = ~clk;

        initial en = 0;
        always #50 en = ~en;

//起初是选择输入功能，之后选择进行加，减，与，或，异或五个操作
        initial
```

```
        begin
            sel = 0;
        #100 sel = 1;
        #100 sel = 2;
        #100 sel = 2;
        #100 sel = 2;
        #100 sel = 2;
        #100 sel = 2;
        end

//输入的两个操作数分别是 2 和 3，之后输入模式分别为加，减，与，或，异或
        initial
        begin
            x = 2;
        #100 x = 3;
        #100 x = 0;
        #100 x = 1;
        #100 x = 2;
        #100 x = 3;
        #100 x = 4;
        end

endmodule
```

（4）仿真电路

操作数为 2 和 3，五个操作（加，减，与，或，异或）结果分别对应红色方框框出部分。

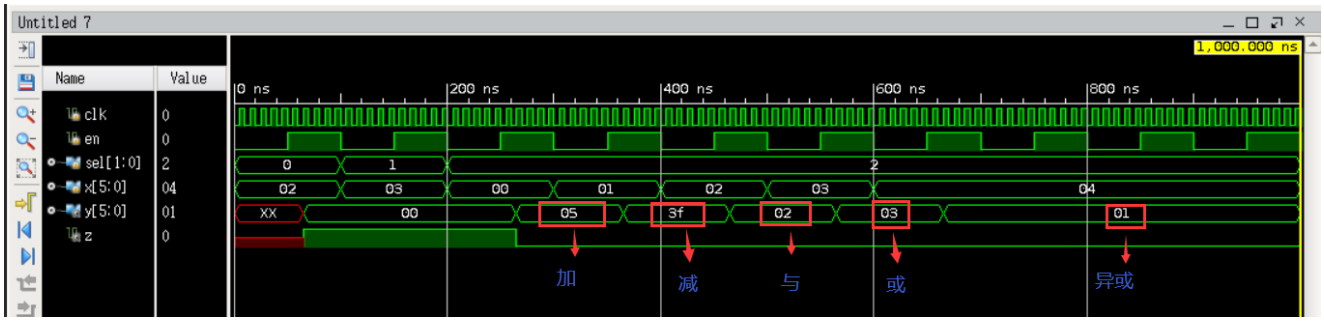
2+3=5

2-3=-1

2&3=2

2 | 3=3

2^3=1



（5）ALU 模块外设端口分配

```
module alu
(
    input clk,
    input  en,
    input  [1:0]sel,
    input  [5:0] x,
    output  [5:0] y,
    output  z
);
```

外设端口分配表

端口	外设
clk	100MHz
en	button
sel	sw[7:6]
x	sw[5:0]
y	led[5:0]
z	led[7]

按照分配表，编写的约束文件如下：

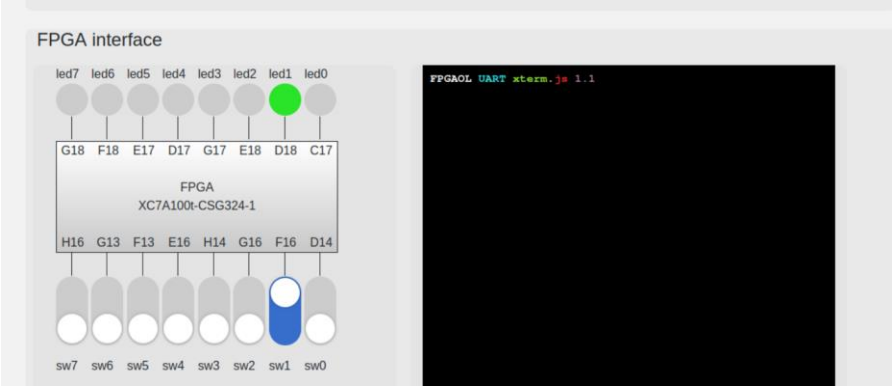
```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk}];
set_property -dict { PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports { en }];
set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports { x[0] }];
set_property -dict { PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports { x[1] }];
set_property -dict { PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports { x[2] }];
set_property -dict { PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports { x[3] }];
set_property -dict { PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports { x[4] }];
set_property -dict { PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports { x[5] }];
set_property -dict { PACKAGE_PIN G13 IOSTANDARD LVCMOS33} [get_ports { sel[0] }];
set_property -dict { PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports { sel[1] }];
set_property -dict { PACKAGE_PIN C17 IOSTANDARD LVCMOS33} [get_ports { y[0] }];
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33} [get_ports { y[1] }];
set_property -dict { PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports { y[2] }];
set_property -dict { PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports { y[3] }];
set_property -dict { PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports { y[4] }];
set_property -dict { PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports { y[5] }];
set_property -dict { PACKAGE_PIN G18 IOSTANDARD LVCMOS33} [get_ports { z }];
```

（6）生成 bit 文件

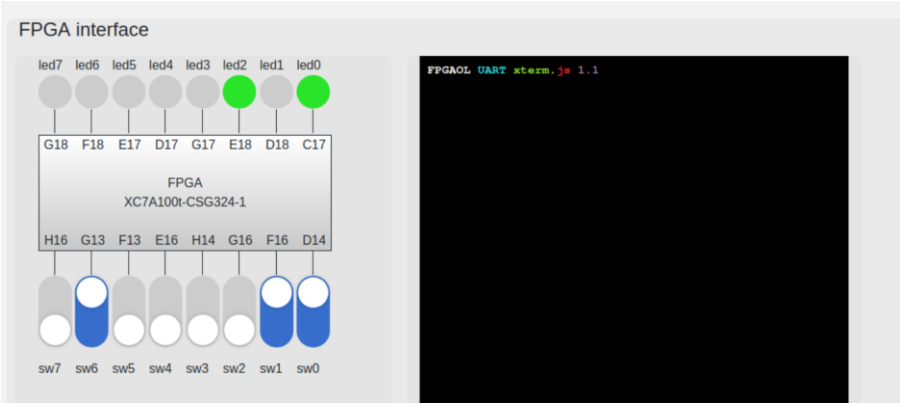
在 fpga 在线平台上的测试：

2+3=5

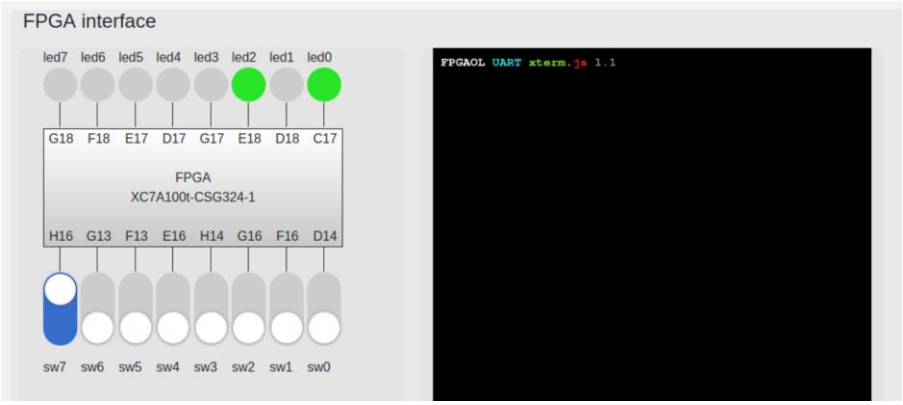
①输入 2



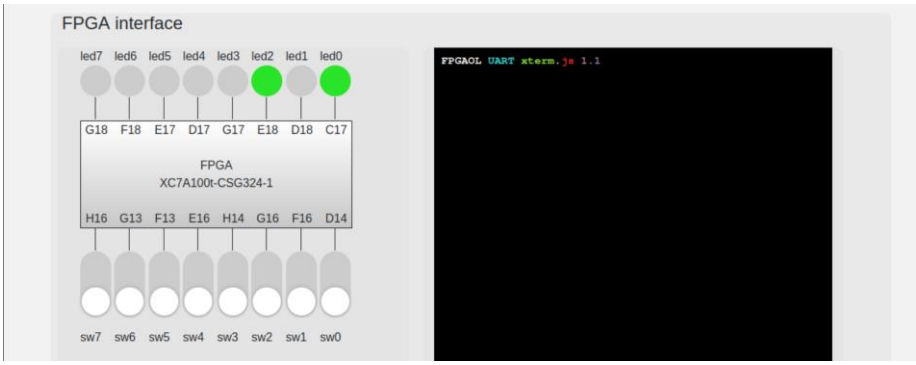
②输入 3



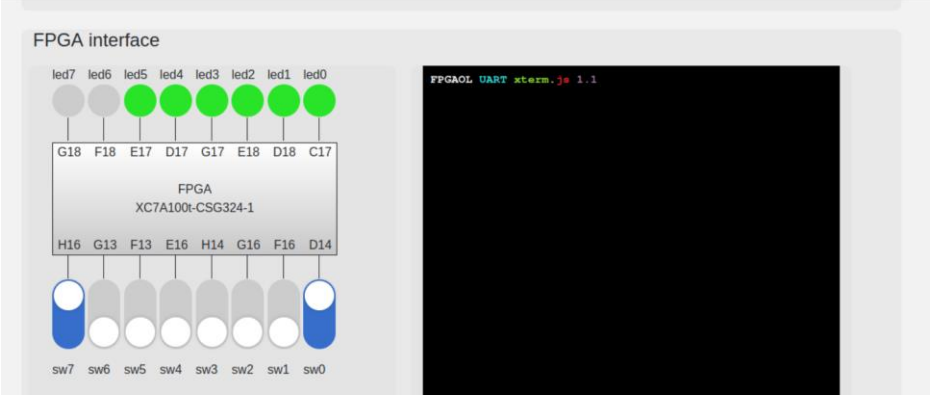
③f:000，表示加法



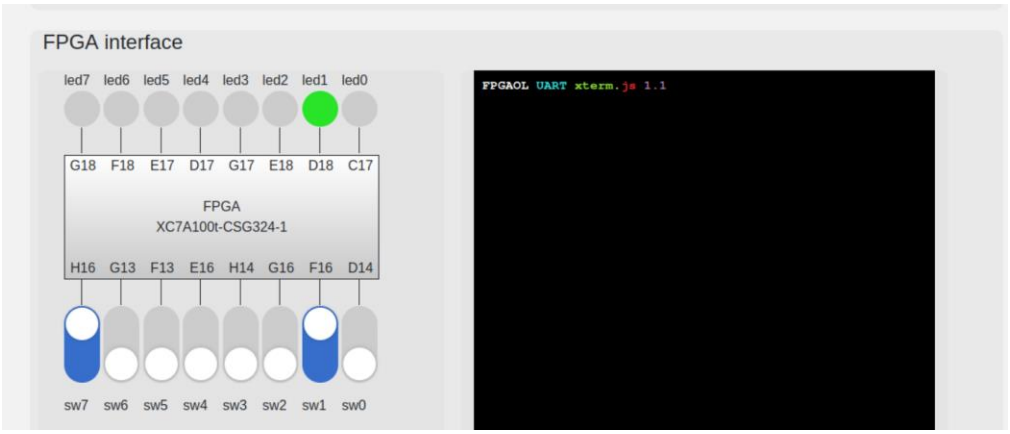
④两数相加，得 5



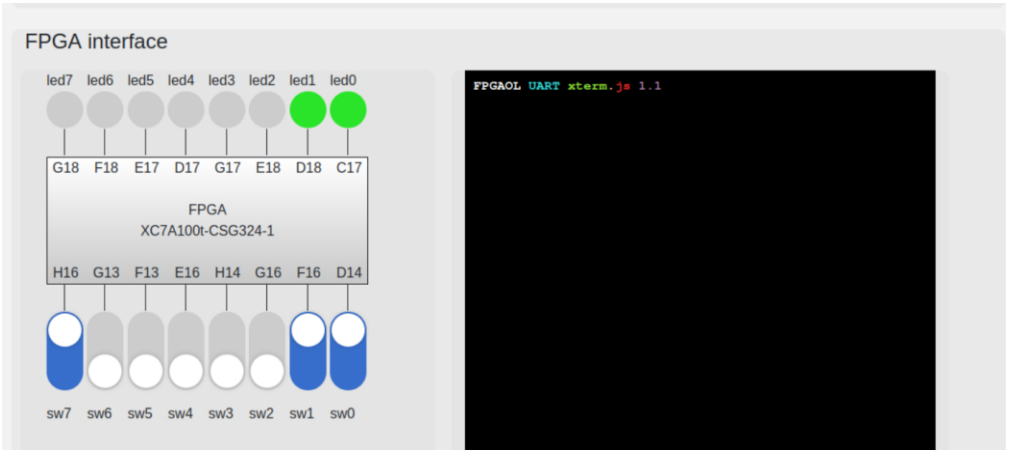
⑤两数相减，得-1



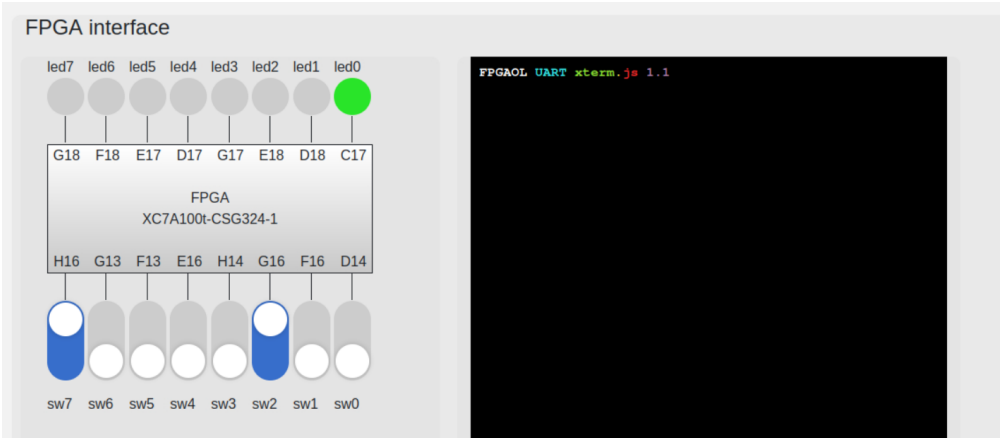
⑥两数相与，得 2



⑦两数相或，得 3



⑧两数相异或，得 1



- **ALU 应用：计算斐波那契—卢卡斯数列（FLS）**

1.结构化描述设计代码：

module fls(

```
    input clk,rst,
    input en,
    input [6:0] d,
    output reg [6:0] f
);

//定义变量
reg [1:0] next_state;
reg [1:0] curr_state;
reg button_r1,button_r2;
reg [6:0] pf = 0;
wire [6:0] y;
wire z;
wire [2:0] s;
wire button_edge;

always@(posedge clk)
    button_r1 <= en;
always@(posedge clk)
    button_r2 <= button_r1;
assign button_edge = button_r1 & (~button_r2);


//描述 cs
always@(posedge clk)
    curr_state <= next_state;


//描述 ns
always@(posedge clk)
begin
if(rst)
    begin
        next_state <= 2'b00;
        pf <= 0;
        f <= 0;
    end
else if(button_edge)
    begin
        pf <= f;
        case(curr_state)
            2'b00:
                begin
                    next_state <= 2'b01;
```

```
                    f <= d;
                end
            2'b01:
                begin
                    next_state <= 2'b10;
                    f <= d;
                end
            default:
                begin
                    next_state <= 2'b11;
                    f <= y;
                end
        endcase
    end
end
```

//套用加法模块，描述输出

```
assign s = 3'b0;
alu #(7) alu2(.a(f),.b(pf),.f(s),.y(y),.z(z));

endmodule
```

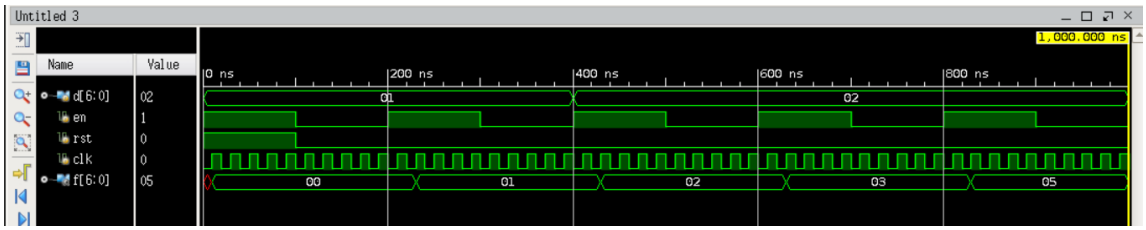
2.仿真文件

```
module fls_simulation(
    );
    reg [6:0] d;
    reg en;
    reg rst;
    reg clk;
    wire [6:0] f;

    fls fls0(clk,rst,en,d,f);
    initial clk = 0;
    always #10 clk = ~clk;

    initial en = 1;
    always #100 en = ~en;

    initial
    begin
```

```
        rst = 1;
    #100 rst = 0;
    end
    initial
    begin
        d = 1;
        #400 d = 2;
    end
endmodule
```

3.仿真测试电路

输出： 0 -> 1(由 d 输入) -> 2(由 d 输入) -> 3 -> 5



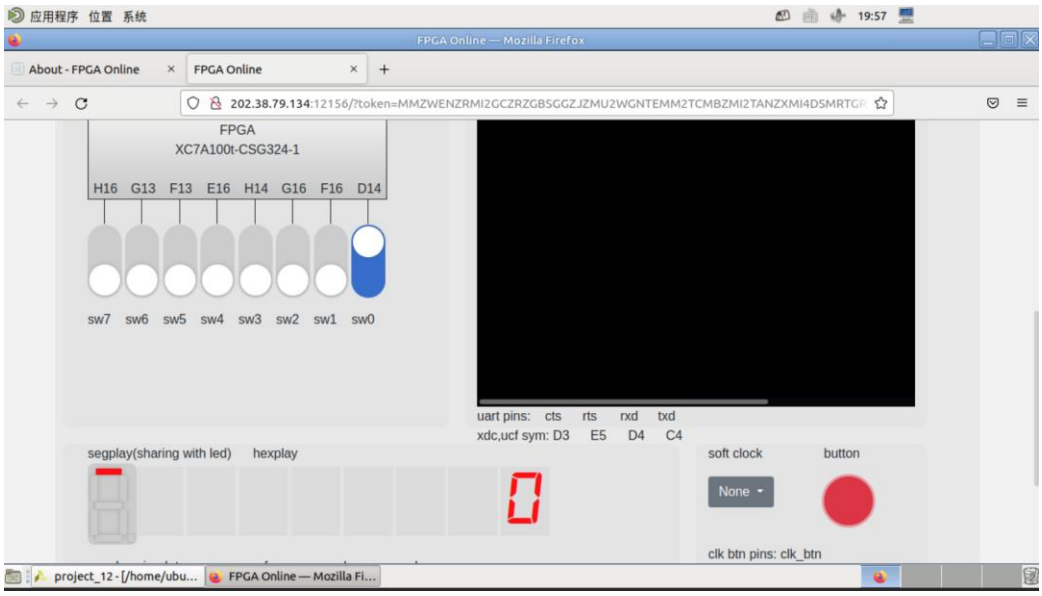
4.约束文件

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk}];

set_property -dict { PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports { en }];
set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports { d[3] }];
set_property -dict { PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports { d[4] }];
set_property -dict { PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports { d[5] }];
set_property -dict { PACKAGE_PIN G13 IOSTANDARD LVCMOS33} [get_ports { d[6] }];
set_property -dict { PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports { rst }];

set_property -dict { PACKAGE_PIN C17 IOSTANDARD LVCMOS33} [get_ports { f[0] }];
```

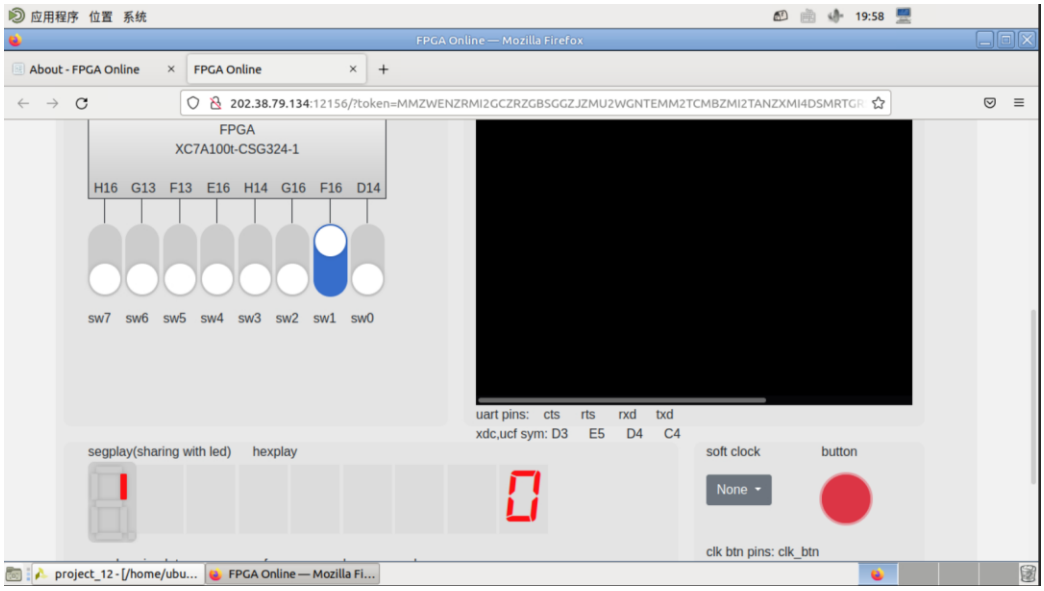
```
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33} [get_ports { f[1] }];
set_property -dict { PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports { f[2] }];
set_property -dict { PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports { f[3] }];
set_property -dict { PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports { f[4] }];
set_property -dict { PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports { f[5] }];
set_property -dict { PACKAGE_PIN F18 IOSTANDARD LVCMOS33} [get_ports { f[6] }];
```

5.生成 bit 文件，并测试

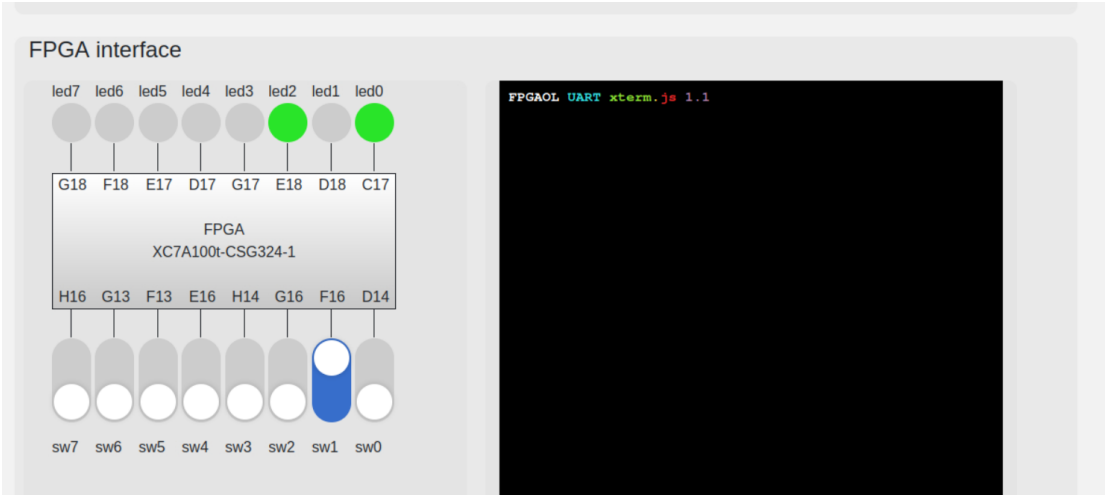
（1）输入 f0=1;



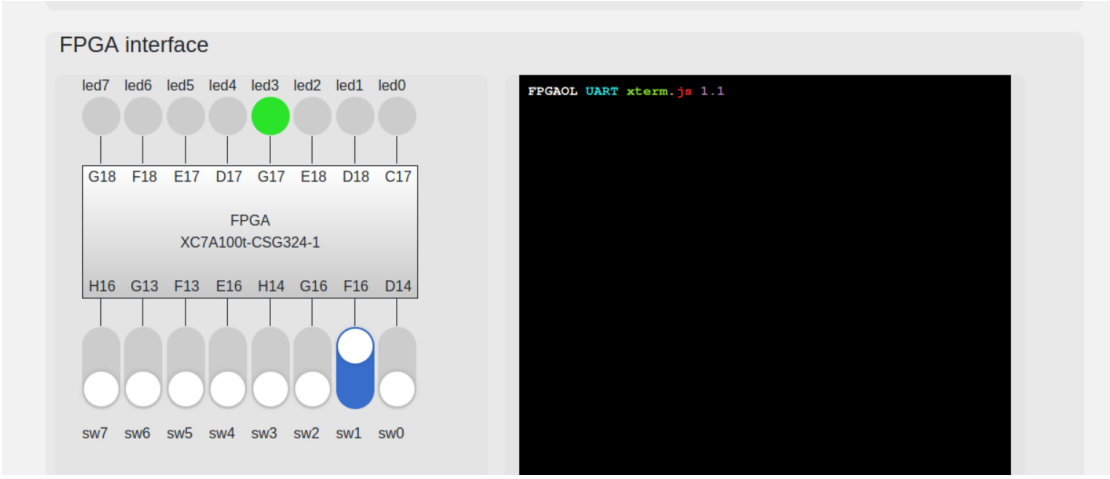
（2）输入 f1=2;



(3) f2=5;


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
FPGAOL UART xterm.js 1.1

(4) f3=8


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
FPGAOL UART xterm.js 1.1

(5) ……