

中国科学技术大学计算机学院
《计算机组成原理实验报告》



实验题目：运算器及其应用
学生姓名：林宸昊
学生学号：PB20000034
完成日期：2022. 3. 15

【实验题目】 运算器及其应用

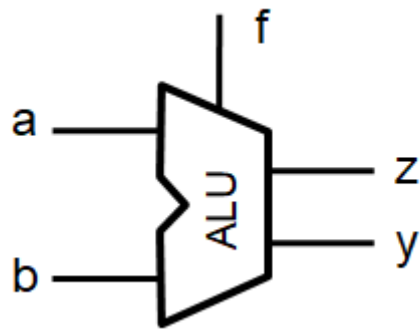
【实验目的】 设计并实现ALU，并在后续中能够利用前述ALU进行复杂设计

【实验平台】 VIVADO FPGAOL

【实验内容】

【一：ALU模块的逻辑设计与仿真】

- 数据通路：



- 状态图:

ALU 模块功能表

f	y	z
000	$a + b$	*
001	$a - b$	*
010	$a \& b$	*
011	$a b$	*
100	$a \wedge b$	*
其他	0	1

- 编写设计文件:

```

module alu_32(
    input [31:0] a, b,
    input [2:0] f,
    output reg [31:0] y,
    output reg z
);
always@(*)
begin
    case(f)          //通过case语句实现操作功能的选择
    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:
        begin
            y = 0;
            z = 1;
        end
    endcase
end
endmodule

```

- 仿真文件

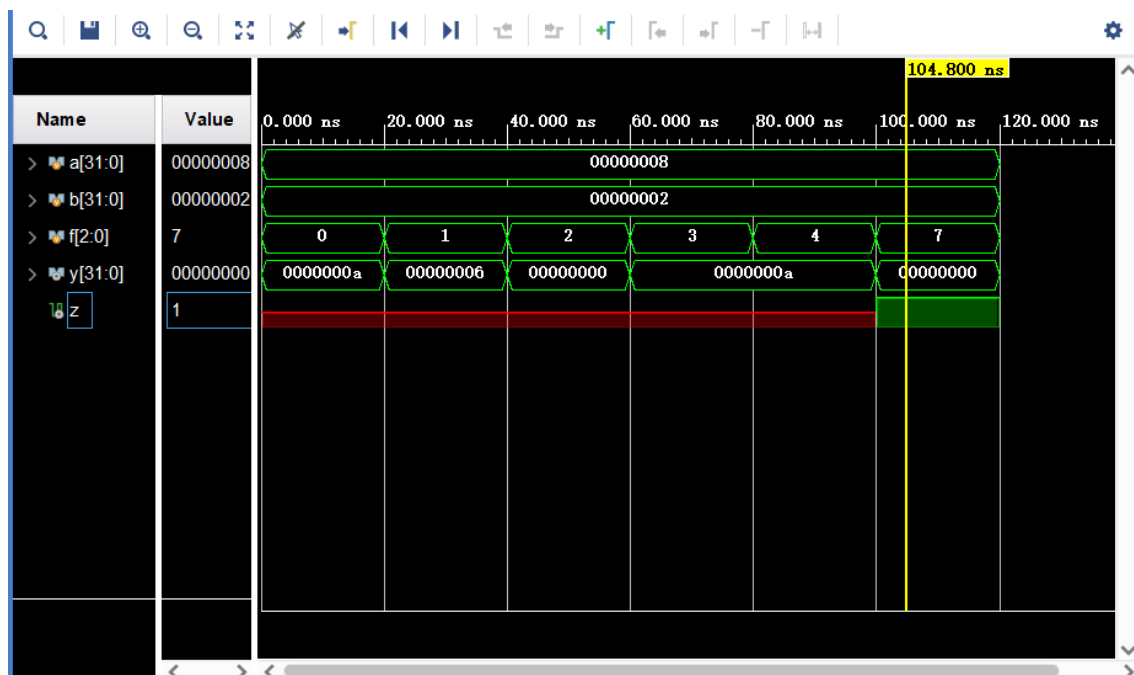
```

module sim1(

);
reg [31:0] a,b;
reg [2:0] f;
wire [31:0] y;
wire z;          //输出必须是线网型
alu_32 alu(a, b, f, y, z);
initial
begin
    a = 32'h8; b = 32'h2;
    f = 3'b0; #20 f = 3'b1; #20 f = 3'b010; #20 f = 3'b011;
    #20 f = 3'b100; #20 f = 3'b111; #20 $finish;
end
endmodule

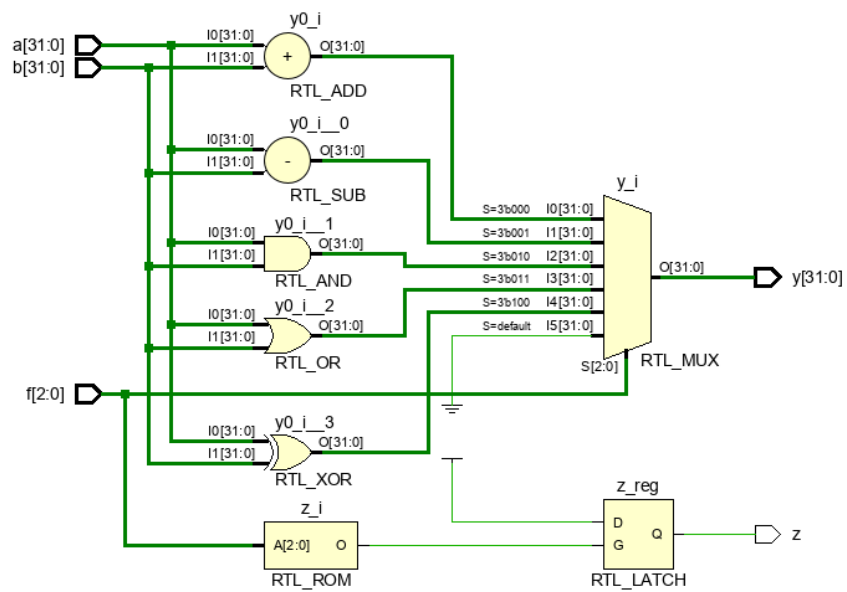
```

- 仿真图象

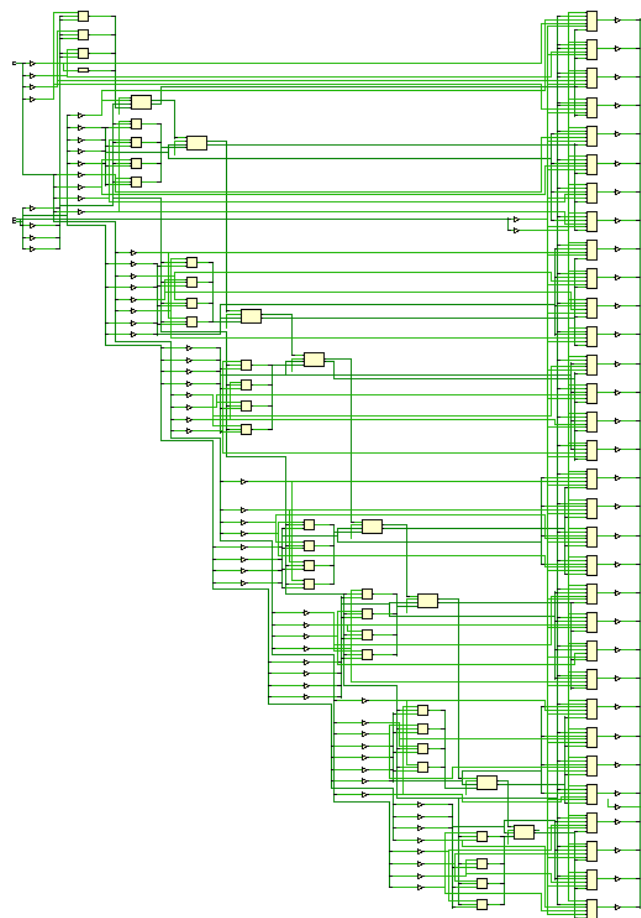


【二：性能报告】

- 生成电路
 - RTL电路：



○ 综合电路：



• 资源使用情况

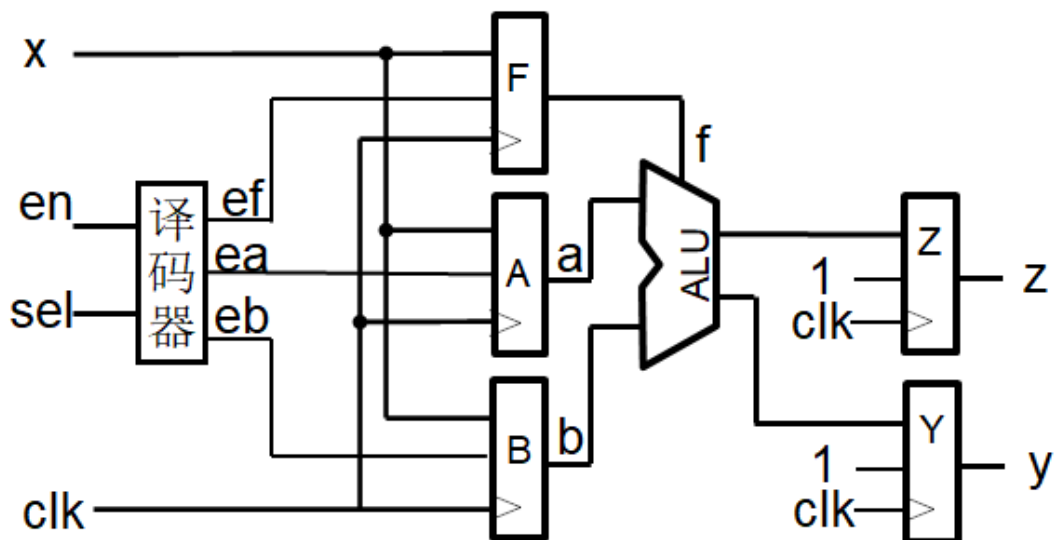
Hierarchy	Name	Slice LUTs (63400)	Bonded IOB (210)
Summary			
▼ Slice Logic	alu_32	64	100
▼ Slice LUTs (
LUT as			
Memory			
DSP			
▼ IO and GT Spec			
Bonded IOB			
Clocking			
Specific Feature			
Primitives			
Black Boxes			
Instantiated Netli			

- 综合电路性能(由于32位ALU未烧在板子上故不做时间性能的特定分析)

General Information	Setup	Hold	Pulse Width
Timer Settings	Worst Negative Slack (WNS): inf	Worst Hold Slack (WHS): inf	Worst Pulse Width Slack (WPWS): NA
Design Timing Summary	Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): NA
> Check Timing (0)	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: NA
Intra-Clock Paths	Total Number of Endpoints: 32	Total Number of Endpoints: 32	Total Number of Endpoints: NA
Inter-Clock Paths			
Other Path Groups			
User Ignored Paths			
> Unconstrained Paths			

【三：6位ALU】

- 数据通路



- 状态图（真值表）

-

en	sel	ea	eb	ef
1	00	1	0	0
1	01	0	1	0
1	10	0	0	1
0	xx	0	0	0

- 端口分配

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

- 设计文件

```

module alu_6(
    input clk,
    input en,
    input [1:0] sel,
    input [5:0] x,
    output reg [5:0] y,
    output reg z
);
    reg [2:0] f;
    reg [5:0] a, b;
    always@(posedge clk)
    begin
        if(en)
            case(sel)
                2'b00:a <= x;
                2'b01:b <= x;
                2'b10:f <= x[2:0];
            endcase
    end

    always@(posedge clk)
    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:
                begin
                    y <= 0;
                    z <= 1;
                end
            endcase
    end
endmodule

```

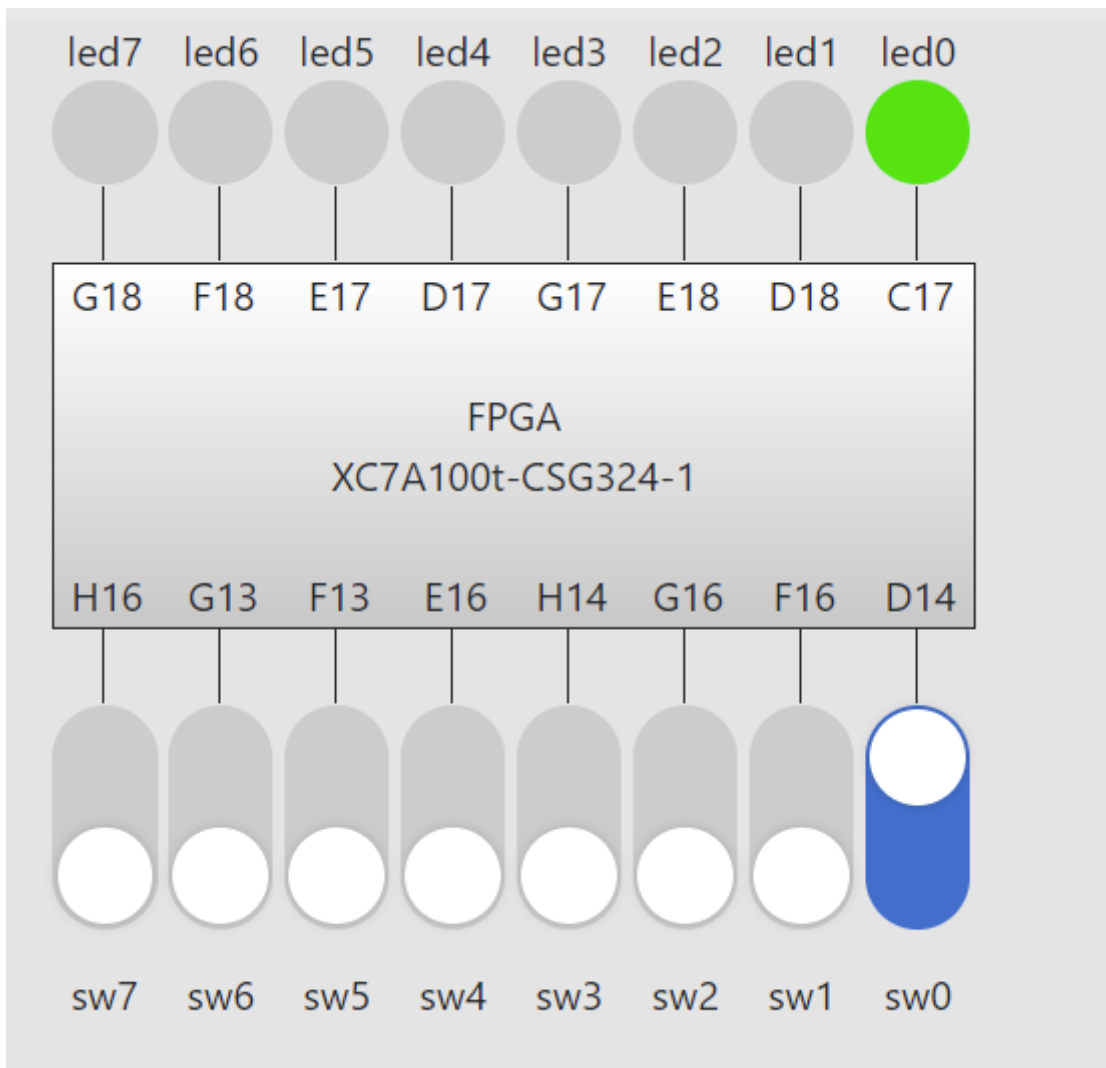
- 约束文件

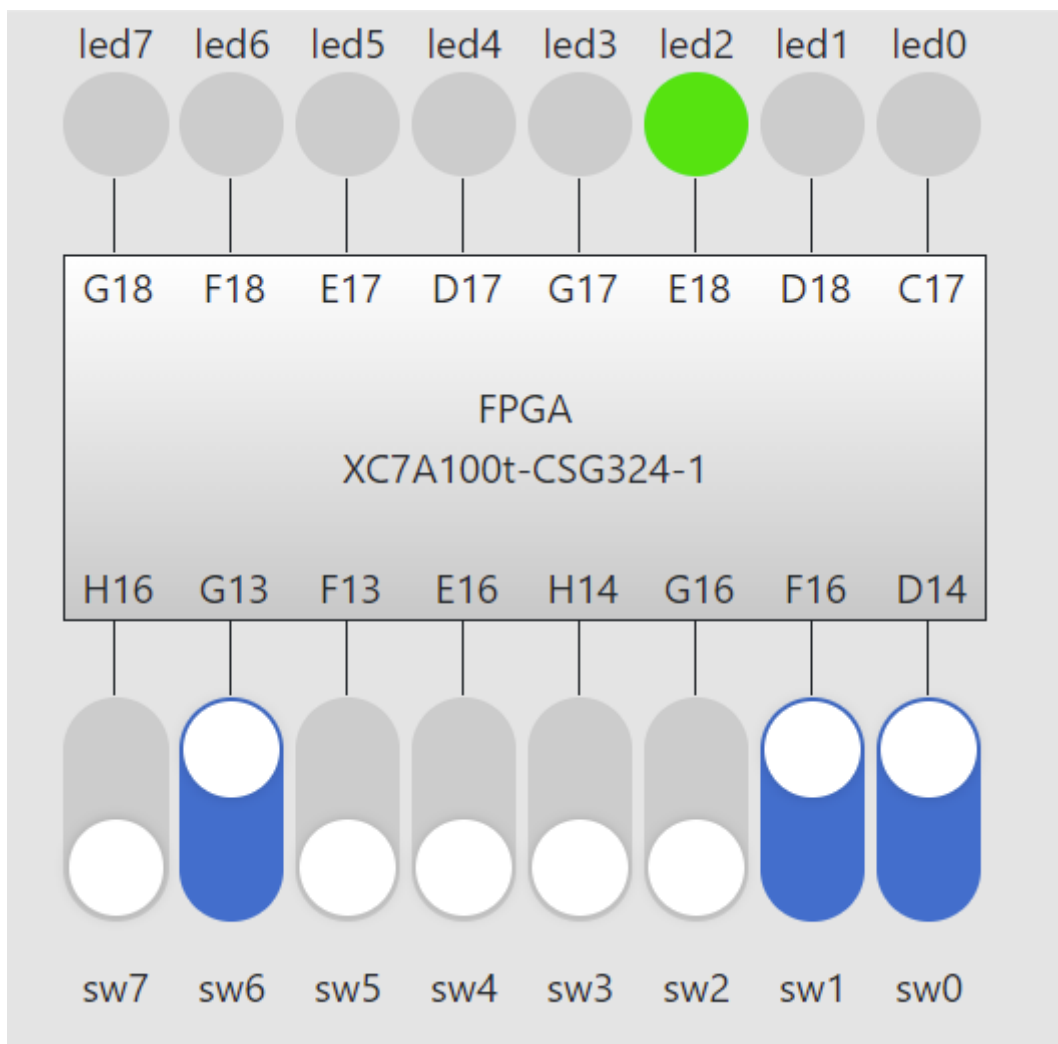
```

6  ## Clock signal
7  set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clk]
8  #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];
9
10
11  ## FPGA0L LED (single-digit-SEGPLAY)
12
13  set_property -dict {PACKAGE_PIN C17 IOSTANDARD LVCMOS33} [get_ports {y[0]}]
14  set_property -dict {PACKAGE_PIN D18 IOSTANDARD LVCMOS33} [get_ports {y[1]}]
15  set_property -dict {PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports {y[2]}]
16  set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {y[3]}]
17  set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {y[4]}]
18  set_property -dict {PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports {y[5]}]
19  set_property -dict {PACKAGE_PIN F18 IOSTANDARD LVCMOS33} [get_ports {}];
20  set_property -dict {PACKAGE_PIN G18 IOSTANDARD LVCMOS33} [get_ports {z}]
21
22
23  ## FPGA0L SWITCH
24
25  set_property -dict {PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports {x[0]}]
26  set_property -dict {PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports {x[1]}]
27  set_property -dict {PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports {x[2]}]
28  set_property -dict {PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports {x[3]}]
29  set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {x[4]}]
30  set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports {x[5]}]
31  set_property -dict {PACKAGE_PIN G13 IOSTANDARD LVCMOS33} [get_ports {sel[0]}]
32  set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports {sel[1]}]
33
34  ## FPGA0L BUTTON & SOFT_CLOCK
35
36  set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports en]

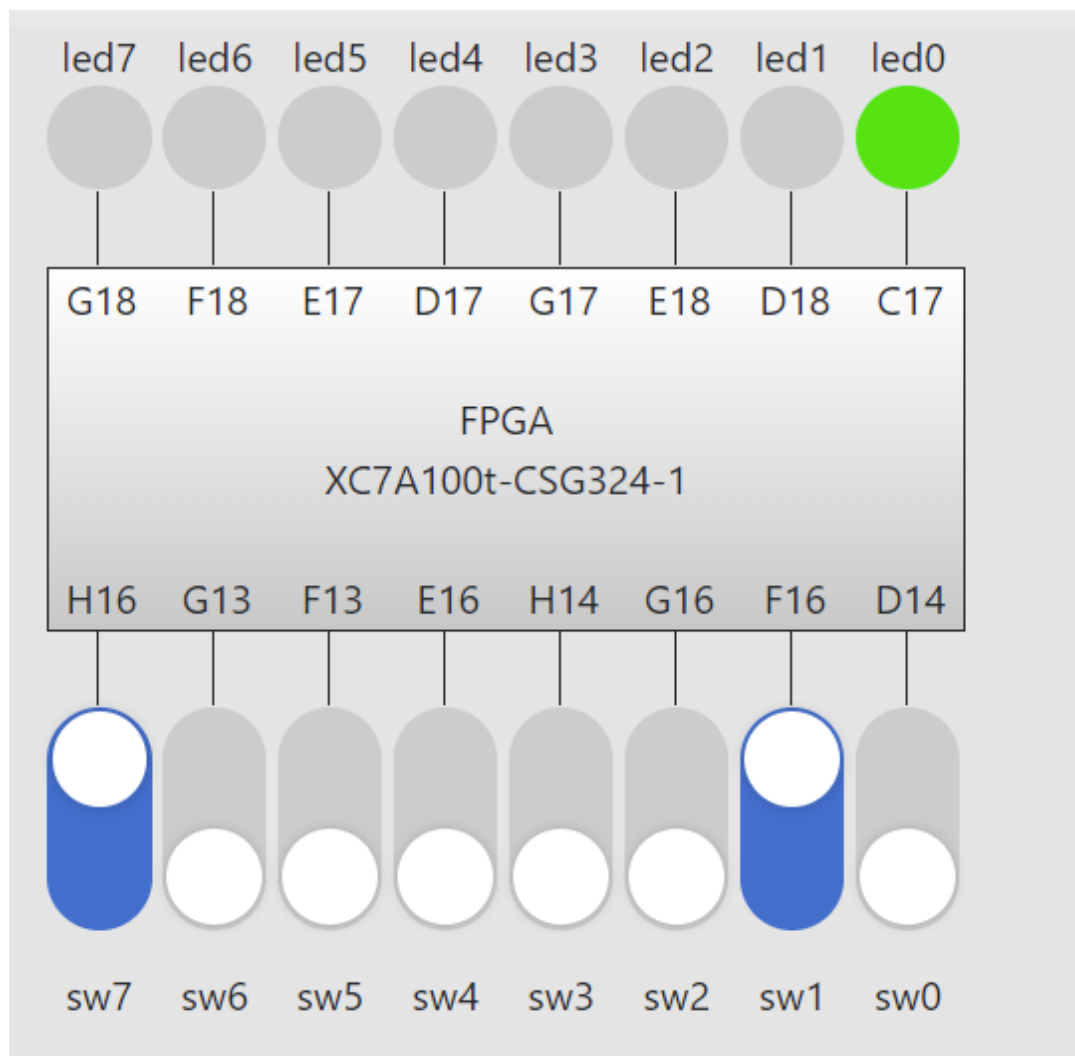
```

- FPGA运行结果 (设a = 1, b = 3)

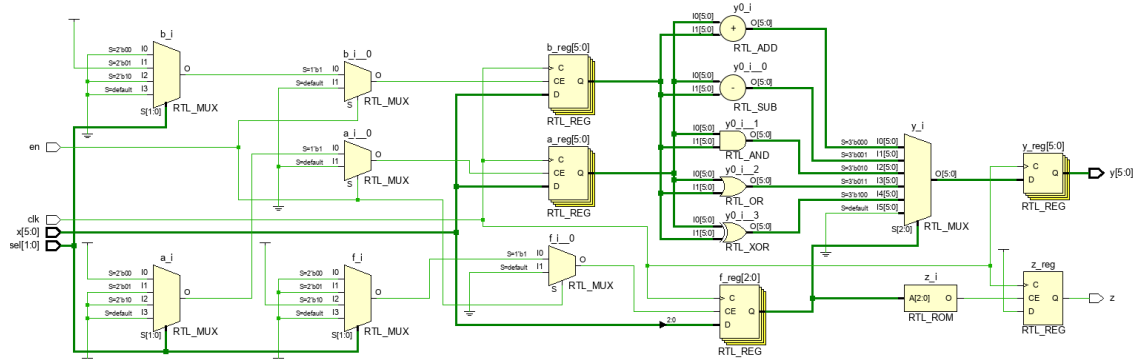




- 测试与功能 (赋值010)



- RTL电路图



- 资源使用情况

Hierarchy

Summary

› Slice Logic

Memory

DSP

Name	1	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
alu_6		15	21	17	1

Hierarchy

Summary

> Slice Logic

Memory

DSP

> IO and GT Specific

> Bonded IOB (8%)

IOB Master Pads

IOB Slave Pads

> Clocking

BUFGCTRL (3%)

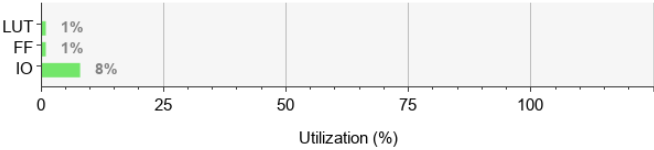
Specific Feature

Primitives

Black Boxes

Instantiated Netlists

Resource	Utilization	Available	Utilization %
LUT	15	63400	0.02
FF	21	126800	0.02
IO	17	210	8.10



• 综合电路时间性能（

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)

> Check Timing (16)

> Intra-Clock Paths

clk

Setup 6.829 ns (7)

Hold 0.214 ns (7)

Pulse Width 4.500 ns (30)

Inter-Clock Paths
Other Path Groups
User Ignored Paths

> Unconstrained Paths

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)

> Check Timing (16)

> Intra-Clock Paths

clk

Setup 6.829 ns (7)

Hold 0.214 ns (7)

Pulse Width 4.500 ns (30)

Inter-Clock Paths
Other Path Groups
User Ignored Paths

> Unconstrained Paths

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	6.829	4	5	2	b_reg[1]C	y_reg[5]D	3.035	1.949	1.086	10.0	clk	clk
Path 2	6.919	3	4	2	b_reg[1]C	y_reg[3]D	2.945	1.723	1.222	10.0	clk	clk
Path 3	6.946	4	5	2	b_reg[1]C	y_reg[4]D	2.918	1.833	1.085	10.0	clk	clk
Path 4	6.986	3	4	2	b_reg[1]C	y_reg[2]D	2.878	1.652	1.226	10.0	clk	clk
Path 5	7.469	3	4	2	b_reg[1]C	y_reg[1]D	2.395	1.309	1.086	10.0	clk	clk
Path 6	7.574	1	2	7	f_reg[1]C	z_reg[CE]	2.044	0.773	1.271	10.0	clk	clk
Path 7	7.744	2	3	2	a_reg[0]C	y_reg[0]D	2.120	1.464	0.656	10.0	clk	clk

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 8	0.214	1	2	7	f_reg[1]C	y_reg[0]D	0.458	0.245	0.213	0.0	clk	clk
Path 9	0.214	1	2	7	f_reg[1]C	y_reg[1]D	0.458	0.245	0.213	0.0	clk	clk
Path 10	0.214	1	2	7	f_reg[1]C	y_reg[2]D	0.458	0.245	0.213	0.0	clk	clk
Path 11	0.214	1	2	7	f_reg[1]C	y_reg[3]D	0.458	0.245	0.213	0.0	clk	clk
Path 12	0.214	1	2	7	f_reg[1]C	y_reg[4]D	0.458	0.245	0.213	0.0	clk	clk
Path 13	0.214	1	2	7	f_reg[1]C	y_reg[5]D	0.458	0.245	0.213	0.0	clk	clk
Path 14	0.482	1	2	7	f_reg[2]C	z_reg[CE]	0.590	0.245	0.345	0.0	clk	clk

slack为正即满足setup time/hold time需求，high fanout为模块直接调用下级模块数量，其后则为各线路时间延迟。

【四：FLS】

- 数据通路

- 设计文件

```
//信号模块用于取按钮边沿
module signal(
    input clk,button,
    output button_edge
);
    reg b1, b2;
    always@(posedge clk)
        b1 <= button;
    always @ (posedge clk)
        b2 <= b1;
    assign button_edge = b1 & (~b2);
endmodule

module fls(
    input clk, rst,
    input en,
    input [6:0] d,
    output reg [6:0] f
);
    reg [6:0] d0, d1; //用于临时储存结果
    reg [1:0] cs; //共四个状态，故只需两位
    reg [1:0] ns;
    wire b_edge; //按钮边沿
    signal signal(clk, en, b_edge);

    //next state
    always @ (*)
    begin
        case(cs)
            2'b00: ns = 2'b01;
            2'b01: ns = 2'b10;
            2'b10: ns = 2'b11; //转入状态二后即进行序列输出，在两个状态间来回切换
            2'b11: ns = 2'b10;
        endcase
    end

    //how current state change
    always @ (posedge clk or posedge rst)
        if(rst)
            cs <= 2'b00;
        else if(b_edge)
            cs <= ns;

    //output
    always @ (posedge clk or posedge rst)
    begin
        if(rst)
        begin
            d0 <= 0;
            d1 <= 0;
        end

        else if(b_edge)
        case(cs)
            2'b00: //状态零，赋值给f0
```

```

        begin
            d0 <= d;
        end
        2'b01://状态一，赋值给f1
        begin
            d1 <= d;
        end
        2'b10://状态二，此时d0排序靠后，累加至d0
        begin
            d0 <= d1 + d0;
        end
        2'b11://状态三，此时d1排序靠后，累加至d1
        begin
            d1 <= d1 + d0;
        end
        endcase
    end

    always @ (posedge clk or posedge rst)
    begin
        if(rst)
            f <= 0;
        else
            case(cs)
                2'b00:
                begin
                    f <= d;
                end
                2'b01:
                begin
                    f <= d;
                end
                2'b10:
                begin
                    f <= d0;
                end
                2'b11:
                begin
                    f <= d1;
                end
            endcase
        end
    end
endmodule

```

- 仿真文件

```

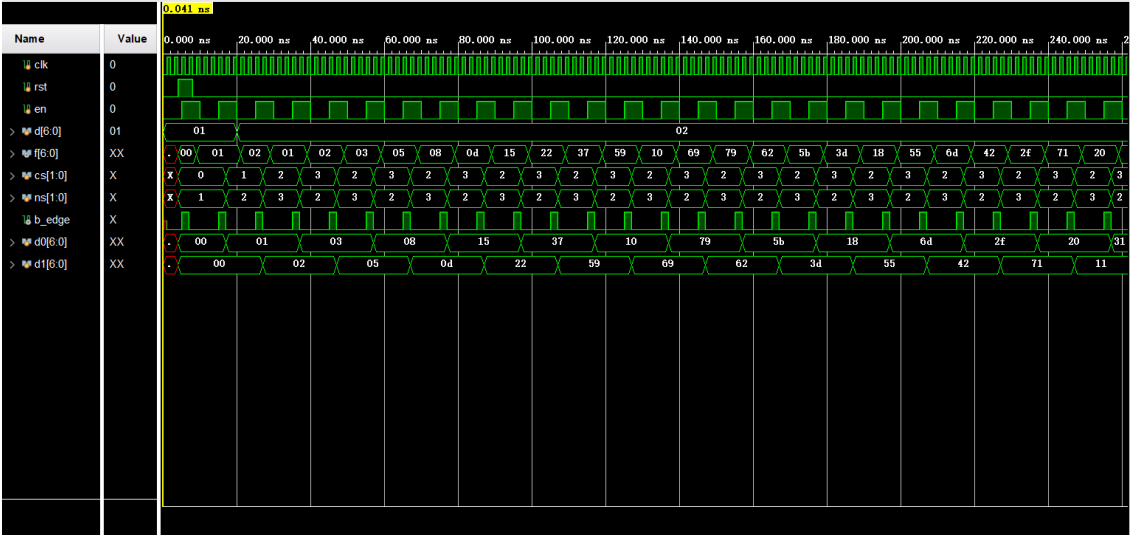
module sim1(

);
    reg clk, rst, en;
    reg [6:0] d;
    wire [6:0] f;
    fls fls(clk, rst, en, d, f);
    initial
    begin
        rst = 0;
        #4 rst = 1;
    end
endmodule

```

```
#4 rst = 0;
end
initial clk = 0;
always #1 clk = ~clk;
initial en = 0;
always #5 en = ~en;
initial
begin
    d = 1;#20 d = 2;
end
endmodule
```

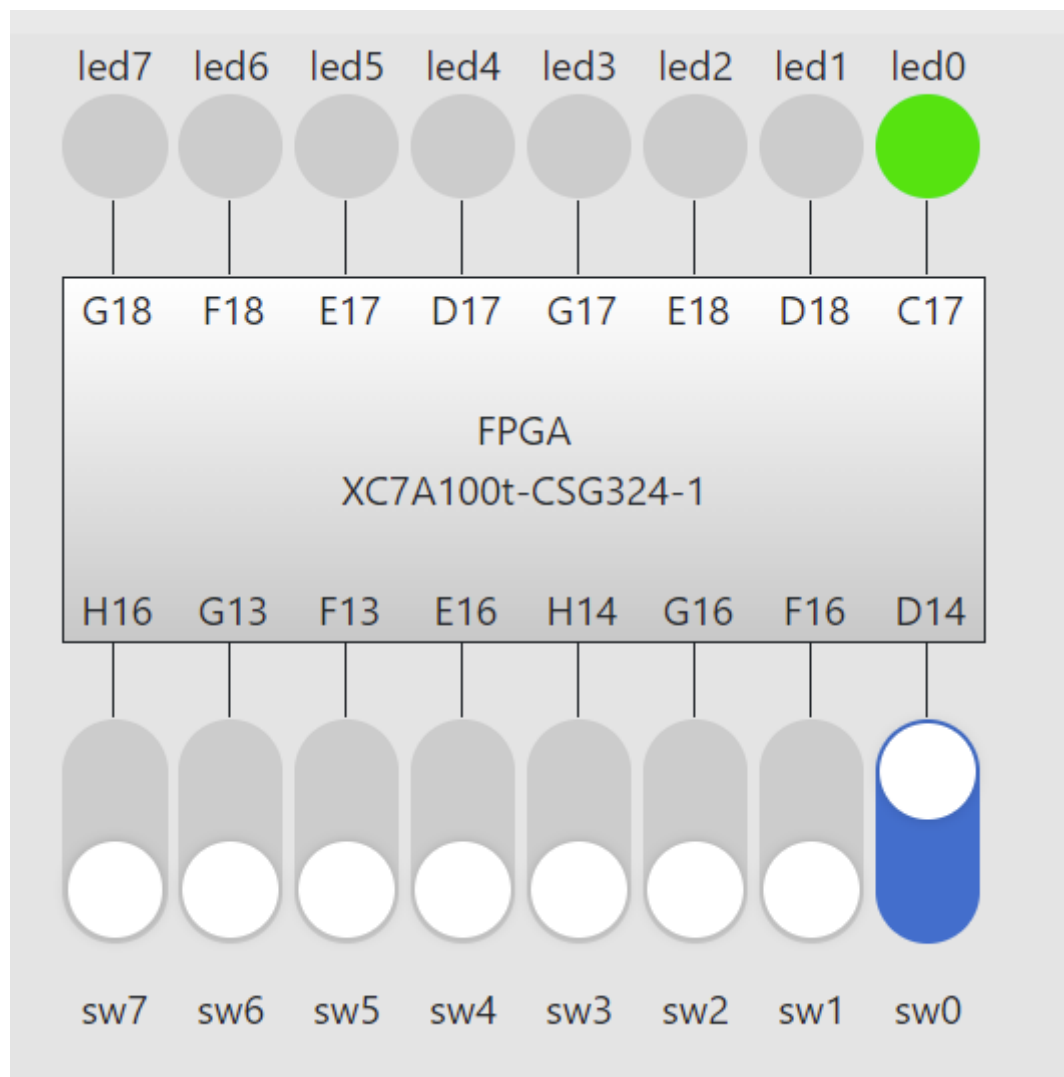
- 仿真结果

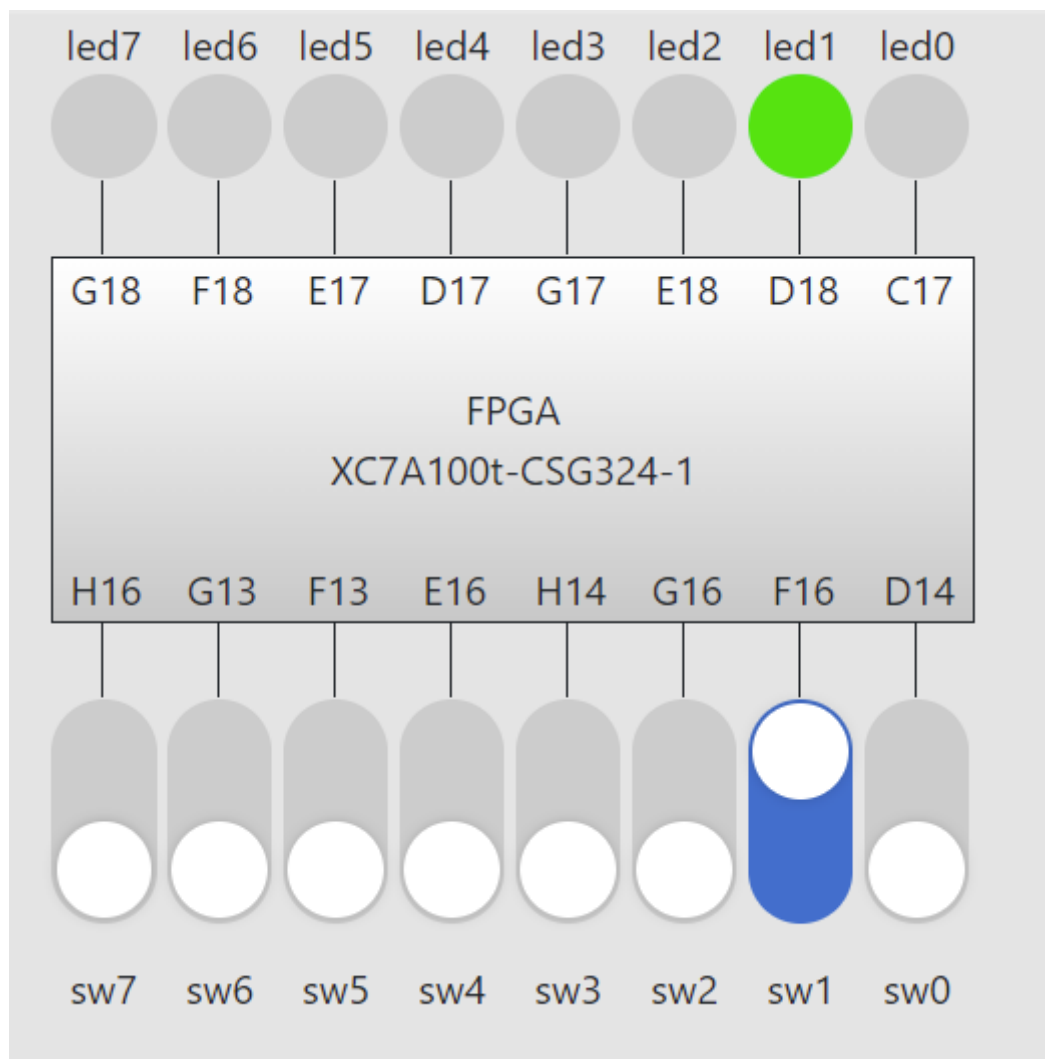


可见在输入初始的两项之后f按照斐波那契序列输出。

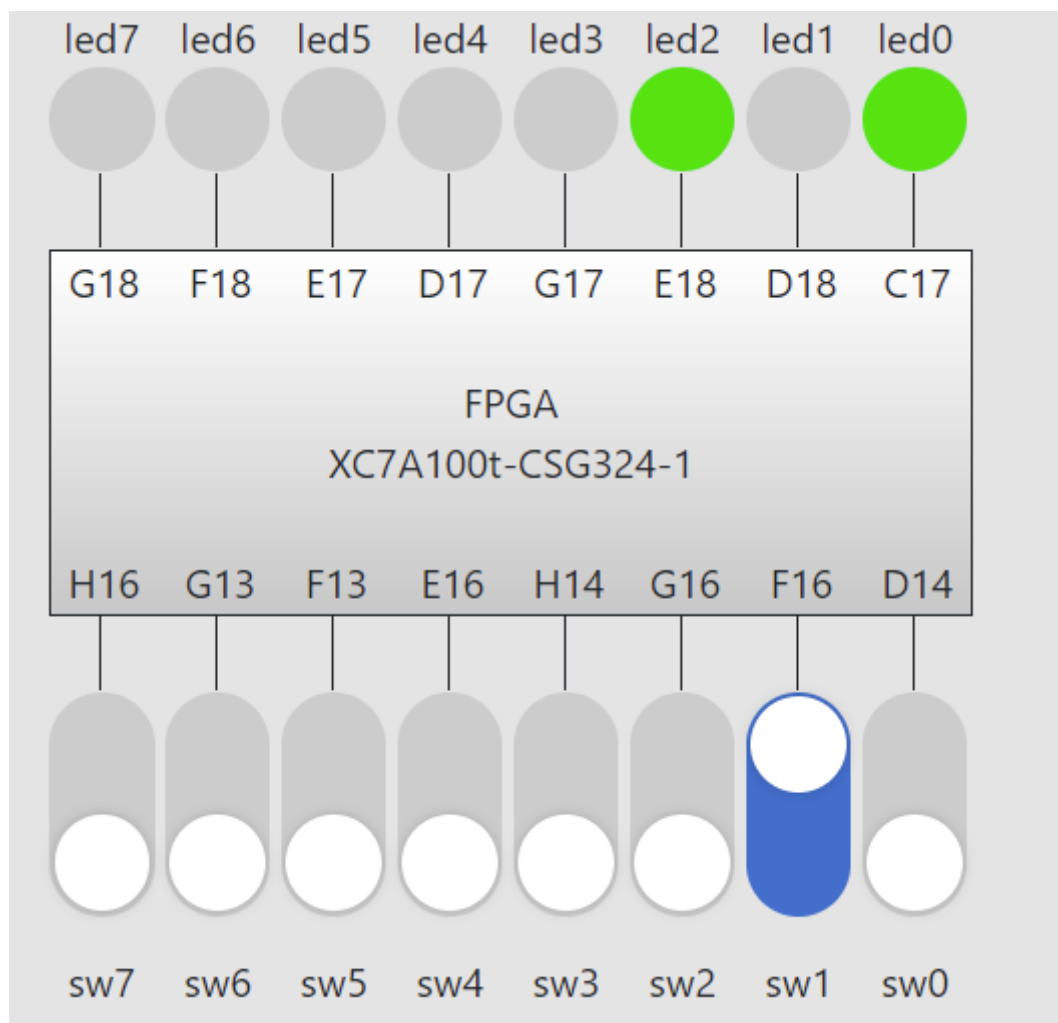
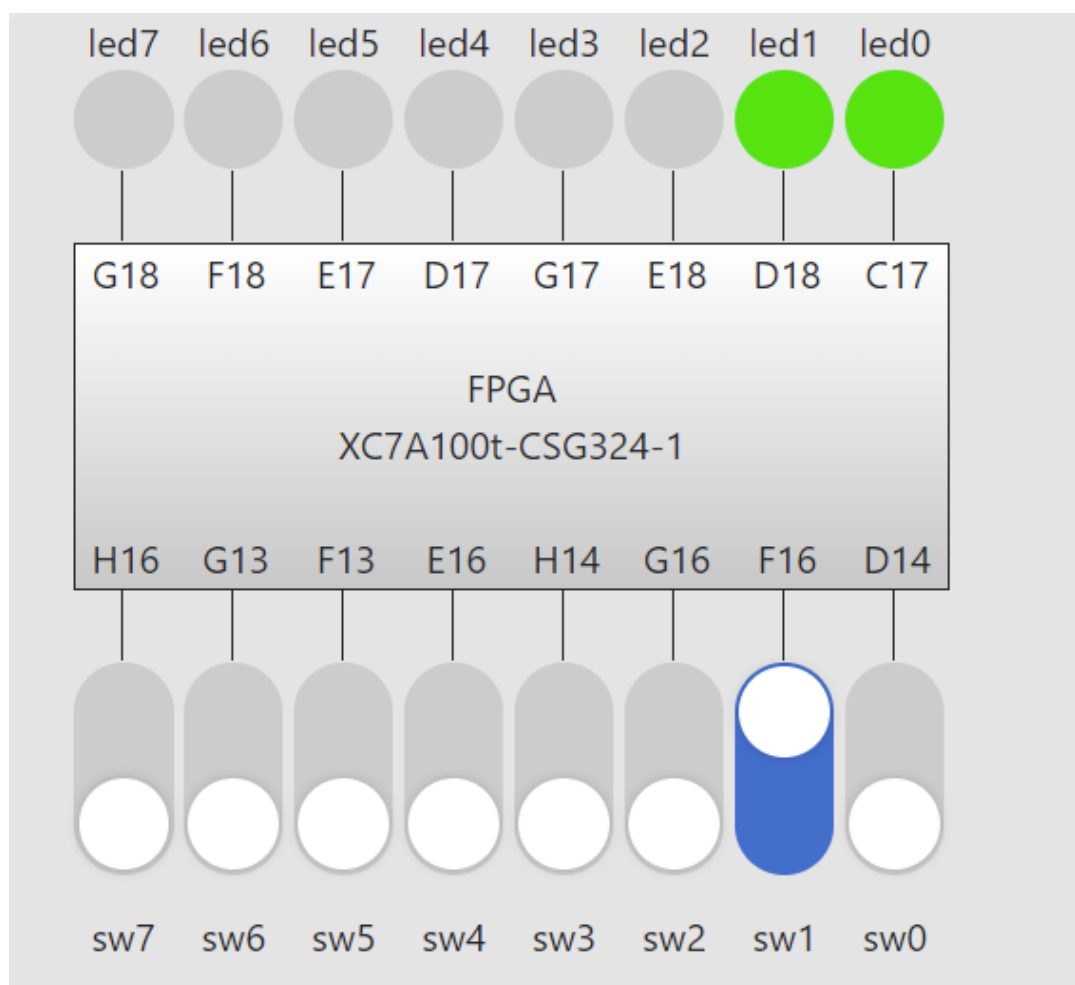
- FPGA测试结果

- 输入 $a = 1, b = 2$





- 此后每一次均输出斐波那契的下一项



【总结与思考】

- 实验总结
 - 总体来说难度不高，主要用于复习上学期所学的模电实验内容——尤其是三段式有限状态机，以及学习如何对当前电路性能进行查看和评测；
- 实验建议（吐槽）
 - 作为一个复习模电的实验感觉没啥问题，确实花了蛮久复习代码怎么写（比如取边沿），但是画数据通路略有些折磨。