计算机组成原理 实验报告

实验题目：运算器及其原理
学生姓名：阿非提
学生学号：PB20111633
完成日期：2022.3.26

实验目的

- 掌握算术逻辑单元（ALU）功能
- 掌握数据通路和控制器的设计
- 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
- 了解查看电路性能和资源使用情况

实验平台

- Vivado

实验过程与结果

- **题目1**

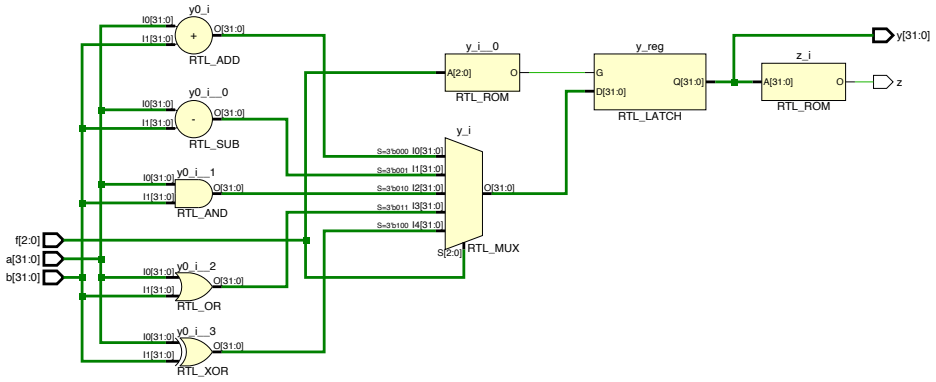
  32位操作数计算逻辑单元

  代码

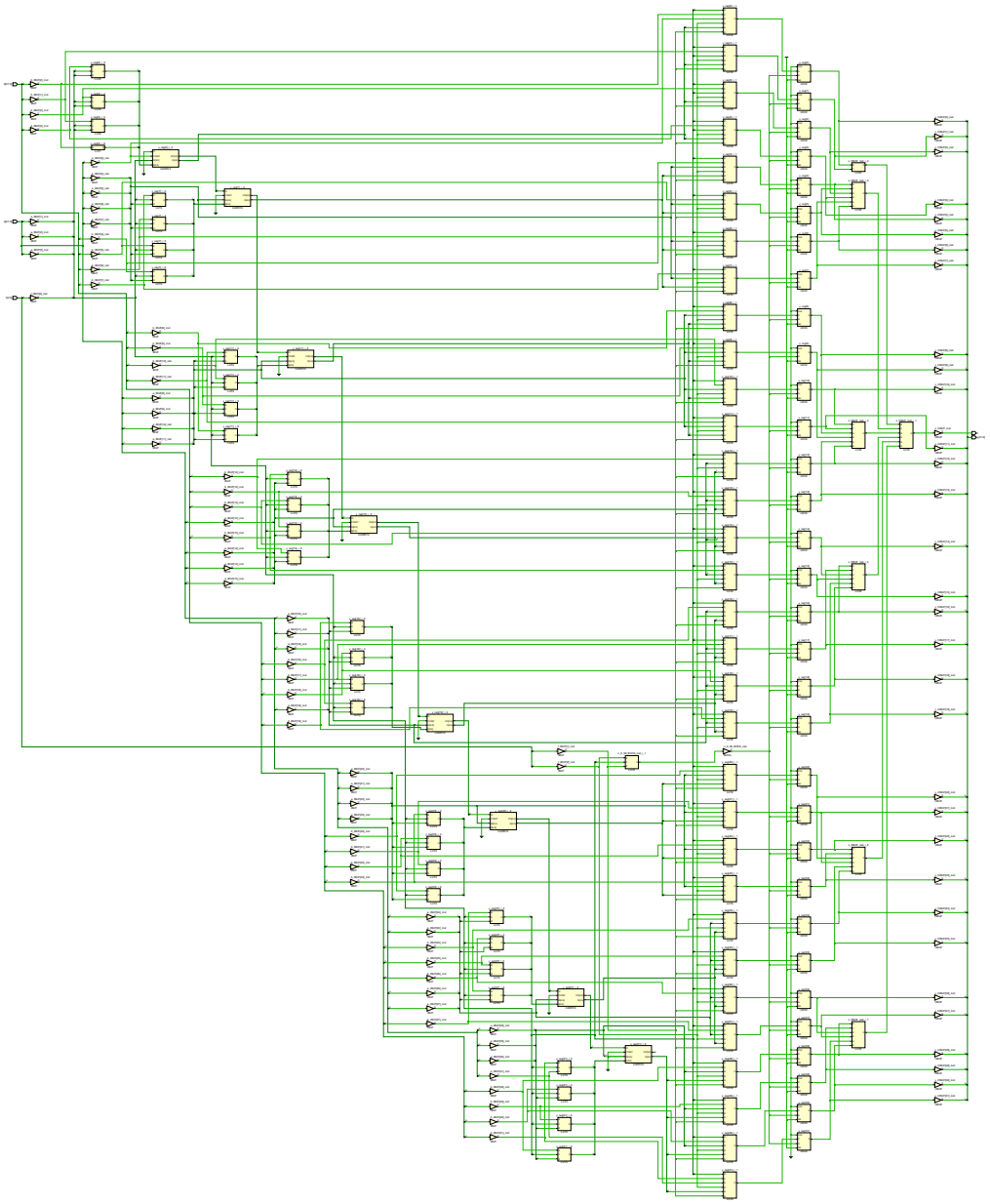
```
module alu#(parameter WIDTH=32) (
input [WIDTH-1:0] a,b,
input [2:0] f,
output reg [WIDTH-1:0] y,
output z
);

    assign z=(y==0)?1:0;
    always@(*)begin
        case(f)
            3'd0: y=a+b;
            3'd1: y=a-b;
            3'd2: y=a&b;
            3'd3: y=a|b;
            3'd4: y=a^b;
            default:;
        endcase
    end
```

```
endmodule
```

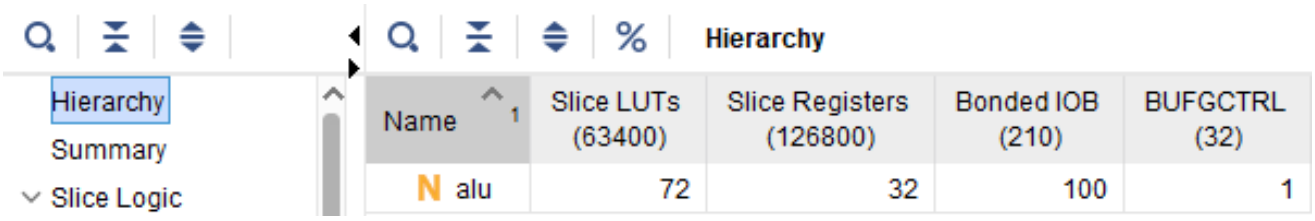
RTL 电路



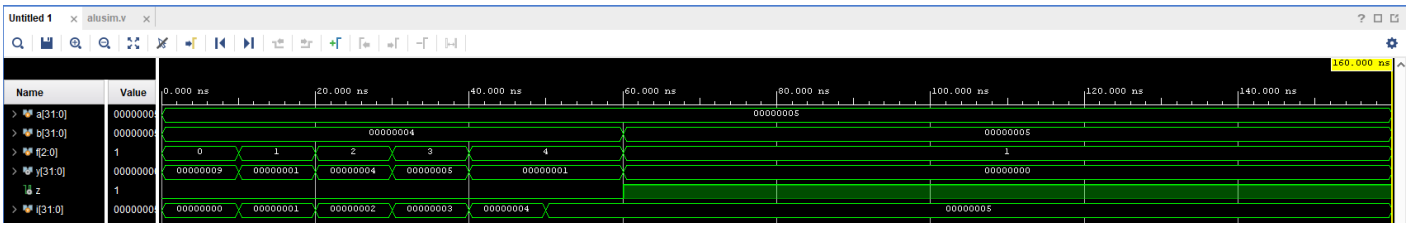
综合电路



资源使用情况



仿真结果



- **题目2**

六位操作数算术逻辑单元

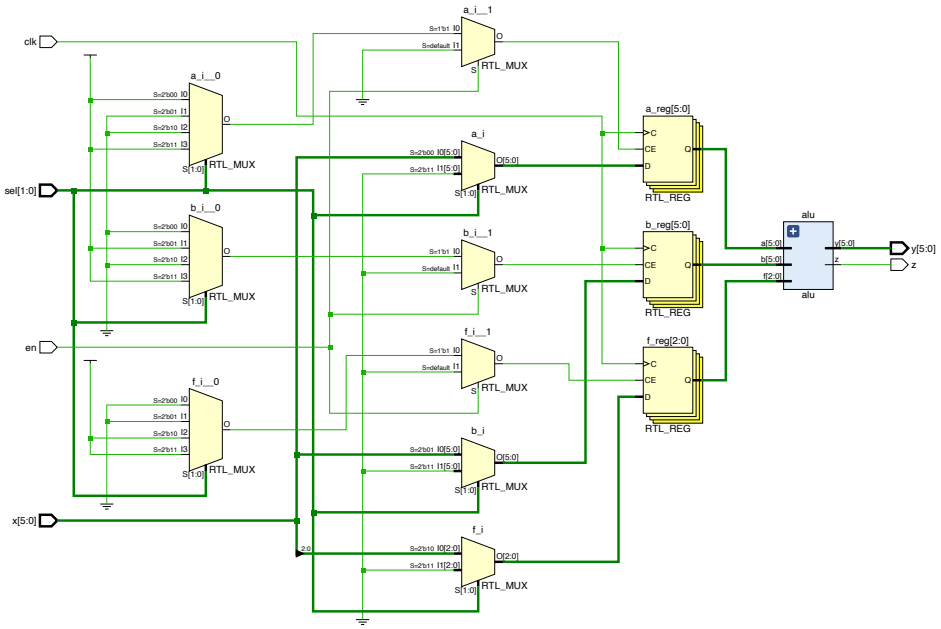
实例化上述ALU单元用于计算，用三个reg类型变量存储a，b，f的值。

代码

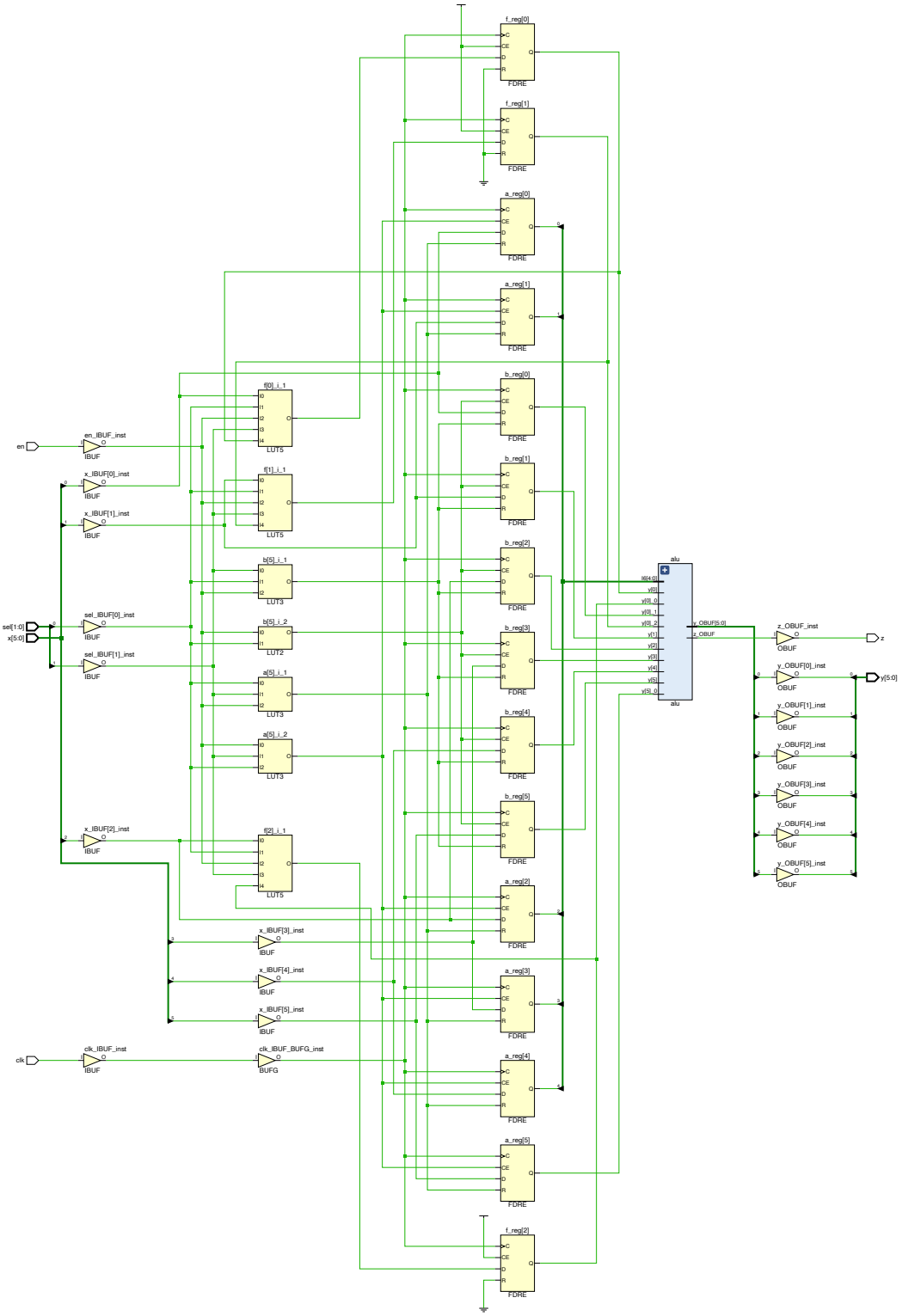
```
module ALU6bit(
    input clk,
    input en,
    input [1:0] sel,
    input [5:0] x,
    output [5:0] y,
    output z
);
reg [5:0] a,b;
reg [2:0] f;
alu #(6) alu(.a(a),.b(b),.f(f),.y(y),.z(z));
always@(posedge clk)begin
    if(en)begin
        case(sel)
            2'b00:a<=x;
            2'b01:b<=x;
            2'b10:f<=x[2:0];
            2'b11:begin
                a<=0;
```

```
                    b<=0;
                    f<=0;
                end
                default:;
            endcase
        end
    end
    endmodule
```

RTL 电路



综合电路


f_reg[0]
f_reg[1]
a_reg[0]
a_reg[1]
b_reg[0]
b_reg[1]
b_reg[2]
b_reg[3]
b_reg[4]
b_reg[5]
a_reg[2]
a_reg[3]
a_reg[4]
a_reg[5]
f_reg[2]
FDRE
f[0]_i_1
f[1]_i_1
f[2]_i_1
LUT5
b[5]_i_1
LUT3
b[5]_i_2
LUT2
a[5]_i_1
a[5]_i_2
en
en_IBUF_inst
x_IBUF[0]_inst
x_IBUF[1]_inst
sel[1:0]
x[5:0]
sel_IBUF[0]_inst
sel_IBUF[1]_inst
x_IBUF[2]_inst
x_IBUF[3]_inst
x_IBUF[4]_inst
x_IBUF[5]_inst
clk
clk_IBUF_inst
clk_IBUF_BUFG_inst
IBUF
BUFG
alu
z_OBUF_inst
y_OBUF[0]_inst
y_OBUF[1]_inst
y_OBUF[2]_inst
y_OBUF[3]_inst
y_OBUF[4]_inst
y_OBUF[5]_inst
OBUF
z
y[5:0]

资源使用情况

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
ALU6bit	20	15	17	1
alu (alu)	13	0	0	0

时间性能报告

Design Timing Summary

Setup

Worst Negative Slack (WNS):	8.730 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	3

Hold

Worst Hold Slack (WHS):	0.147 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	3

Pulse Width

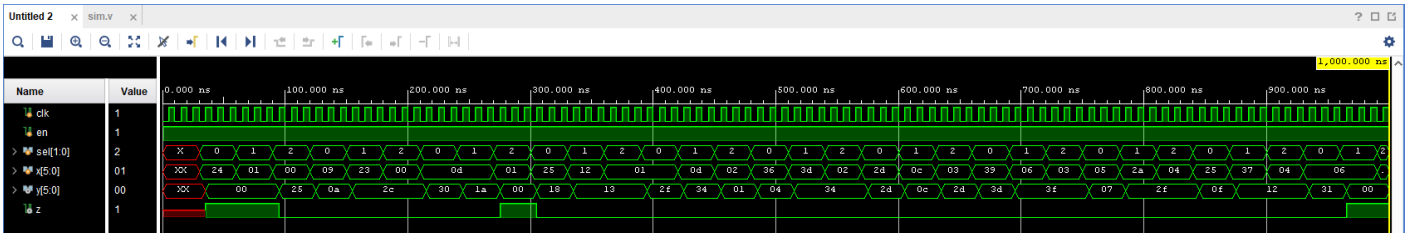
Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	16

All user specified timing constraints are met.

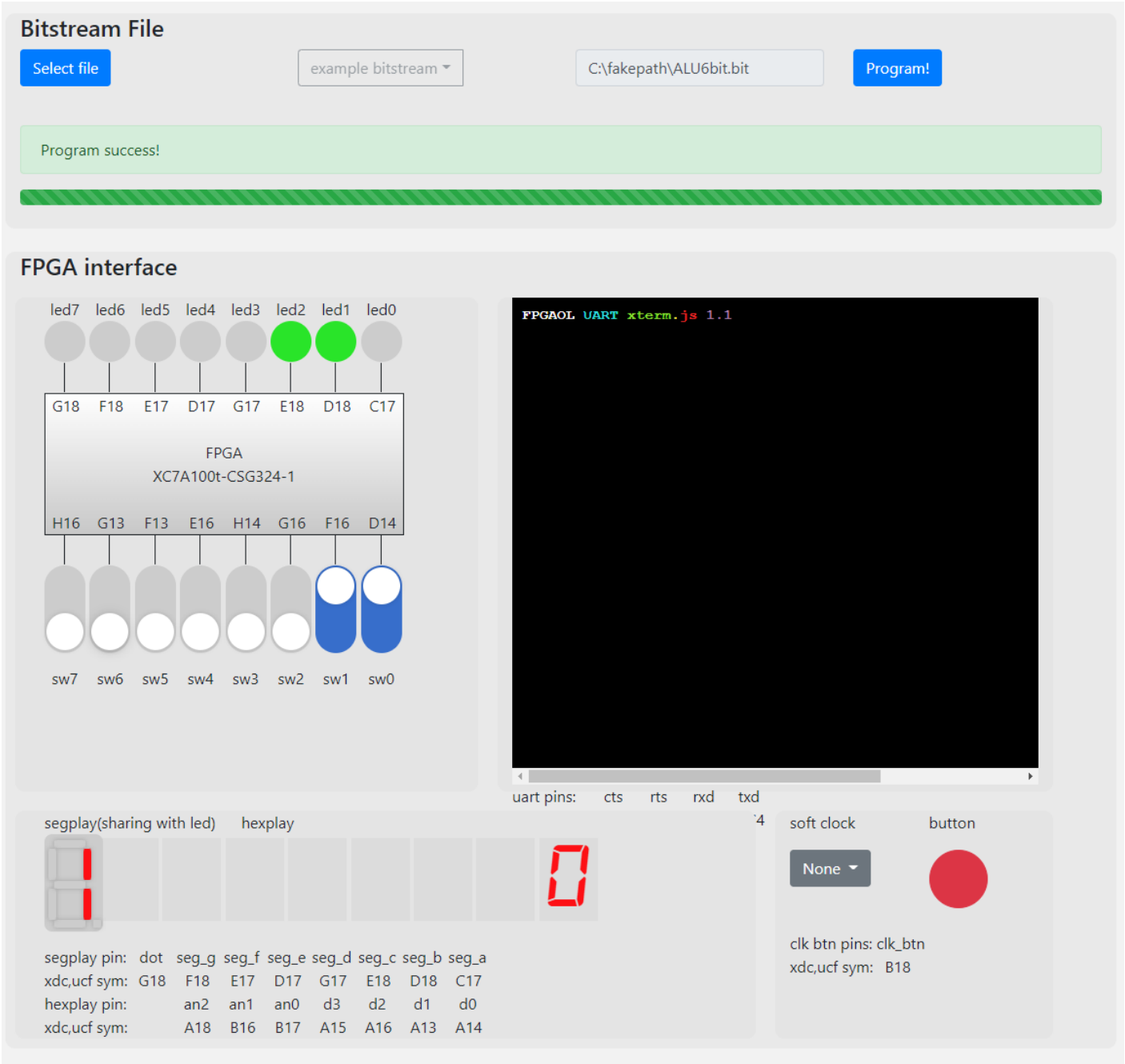
Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Un
Path 1	8.730	1	2	13	f_reg[0]/C	f_reg[0]/D	1.134	0.773	0.361	10.0	sys_clk_pin	sys_clk_pin		
Path 2	8.745	1	2	7	f_reg[1]/C	f_reg[1]/D	1.119	0.773	0.346	10.0	sys_clk_pin	sys_clk_pin		
Path 3	8.745	1	2	7	f_reg[2]/C	f_reg[2]/D	1.119	0.773	0.346	10.0	sys_clk_pin	sys_clk_pin		

仿真结果



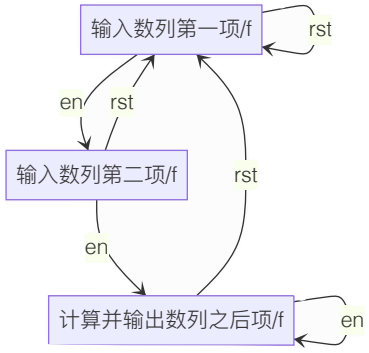
运行结果



- **题目3**

  FLS

  作出状态转化图



由此可知需要三种状态，分别记为state0、state1、state2。

由数列表达式可知 $F_n = F_{n-1} + F_{n-2}$ 可知，只需要两个变量存储当前项和前一项，分别记 f、f_0。计算下一项只需 $f = f + f_0$ 。

需要注意的是 en 输入需要去除毛刺和取边缘。

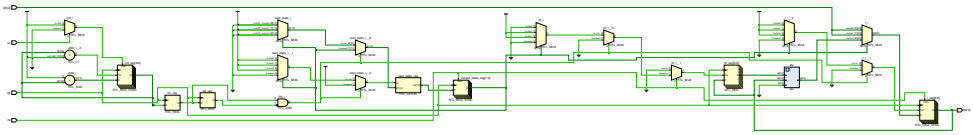
代码

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

reg [6:0] f0;
wire [6:0] sum_of_f0_f;
alu #(7) alu(.a(f0),.b(f),.f(0),.y(sum_of_f0_f));

parameter state0 = 2'b00;
parameter state1 = 2'b01;
parameter state2 = 2'b10;
parameter reset = 7'b0;

reg [1:0] current_state;
reg [1:0] next_state;

//jitter cleaning
reg [3:0] cnt;
wire clean_button;
always@(posedge clk)begin
    if(en==1'b0)cnt<=4'b0;
    else if(cnt<4'h8)cnt<=cnt+4'b1;
end
assign clean_button=cnt[3];

//edge detection
reg b1,b2;
wire EN;

always@(posedge clk)b1<=clean_button;
always@(posedge clk)b2<=b1;
```

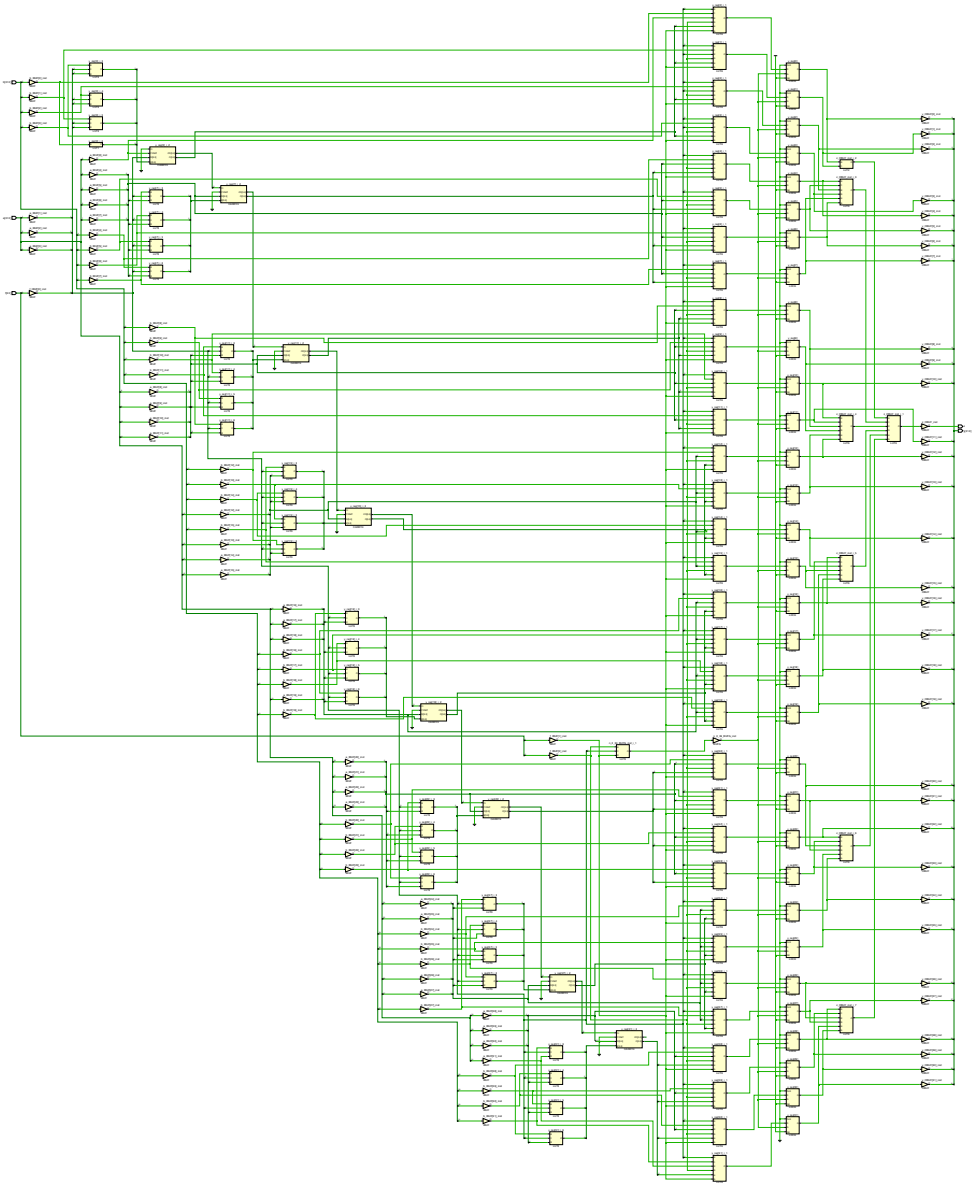
```
assign EN = b1&(~b2);

//NS
always@(*)begin
    if(EN)begin
        case(current_state)
            state0: next_state = state1;
            state1: next_state = state2;
            state2: next_state = state2;
            default:;
        endcase;
    end
    else next_state = current_state;
end

//CS
always@(posedge clk)begin
    if(rst)
        current_state <= state0;
    else
        current_state <= next_state;
end

//output
always@(posedge clk)begin
    if(rst)
        f <= 7'd0;
    else if(EN) begin
        case(current_state)
            state0: f <= d;
            state1: begin
                f0 <= f;
                f <= d;
            end
            state2: begin
                f0 <= f;
                f <= sum_of_f0_f;
            end
            default:;
        endcase
    end
```

```
    end

    endmodule
```

RTL电路



综合电路



资源使用情况

Utilization | Find Results | Timing

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
FLS	20	24	17	1
alu (alu)	11	0	0	0

时间性能报告

Design Timing Summary

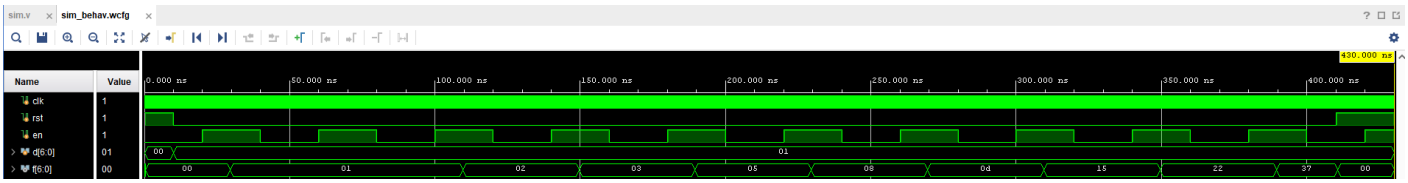
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.063 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	38	Total Number of Endpoints:	38	Total Number of Endpoints:	23

All user specified timing constraints are met.

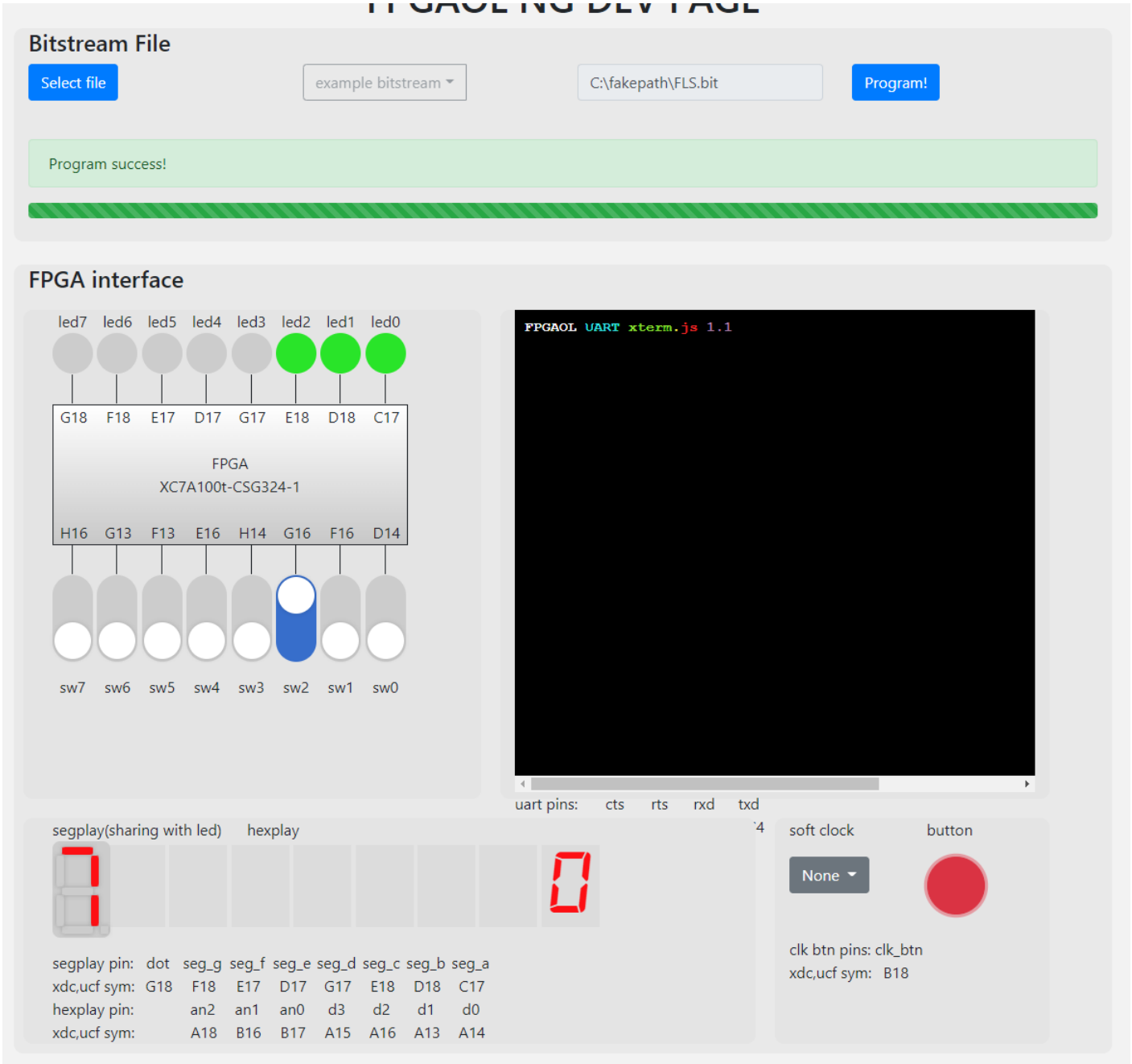
Intra-Clock Paths - sys_clk_pin - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	7.063	4	5	4	f_reg[1]/C	f_reg[6]/D	2.801	1.855	0.946	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 2	7.109	4	5	4	f_reg[1]/C	f_reg[5]/D	2.755	1.949	0.806	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 3	7.199	3	4	4	f_reg[1]/C	f_reg[3]/D	2.665	1.723	0.942	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 4	7.234	4	5	4	f_reg[1]/C	f_reg[4]/D	2.630	1.825	0.805	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 5	7.274	3	4	4	f_reg[1]/C	f_reg[2]/D	2.590	1.644	0.946	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 6	7.503	1	2	7	current_state_reg[0]/C	f0_reg[0]/CE	2.115	0.773	1.342	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 7	7.503	1	2	7	current_state_reg[0]/C	f0_reg[1]/CE	2.115	0.773	1.342	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 8	7.503	1	2	7	current_state_reg[0]/C	f0_reg[2]/CE	2.115	0.773	1.342	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 9	7.503	1	2	7	current_state_reg[0]/C	f0_reg[3]/CE	2.115	0.773	1.342	10.0	sys_clk_pin	sys_clk_pin		0.035
Path 10	7.503	1	2	7	current_state_reg[0]/C	f0_reg[4]/CE	2.115	0.773	1.342	10.0	sys_clk_pin	sys_clk_pin		0.035

仿真结果



运行结果



总结与思考

- 通过本次试验我学会了如何通过Vivado仿真工具进行Verilog 进行仿真，查看并分析波形文件 。
- 本次试验难易程偏低。
- 本次试验任务量适中。