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：LabH1_运算器与运用

学生姓名：钟书锐

学生学号：PB19000362

完成日期：2021.3.31

计算机实验教学中心制

2020 年 09 月

【实验题目】

LabH1_运算器与运用

【实验目的】

1.掌握 Vivado 的使用

2.熟悉 Verilog 的语法

3.掌握约束文件与模拟文件的书写

4.了解如何通过 Vivado 产生电路图

5.熟悉综合、仿真与下载的步骤，掌握 Verilog 代码的调试

6.实现运算器，并基于其实现 FIB 等小程序

【实验环境】

硬件:

处理器：i7-10750H @ 2.60GHz 六核

显卡：RTX2060(6GB)

Nexys4DDR 开发板

操作系统:

WINDOWS10 家庭中文版

软件:

Vivado 2020.1

【实验内容】

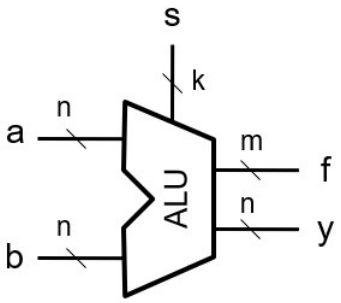
1. **算术逻辑单元（ALU）-6 位**

s：功能选择。加、减、与、或、非、异或等运算

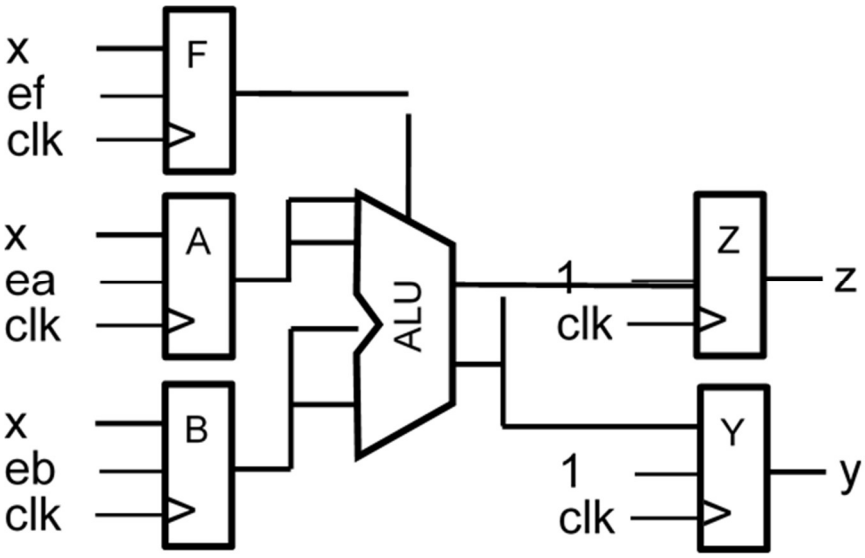
a, b：两操作数。减运算-- a 是被减数；非运算--操作数是 a

y：运算结果。和、差

f：标志:进位/借位（cf)、溢出（of)、零标志（zf)



数据通路

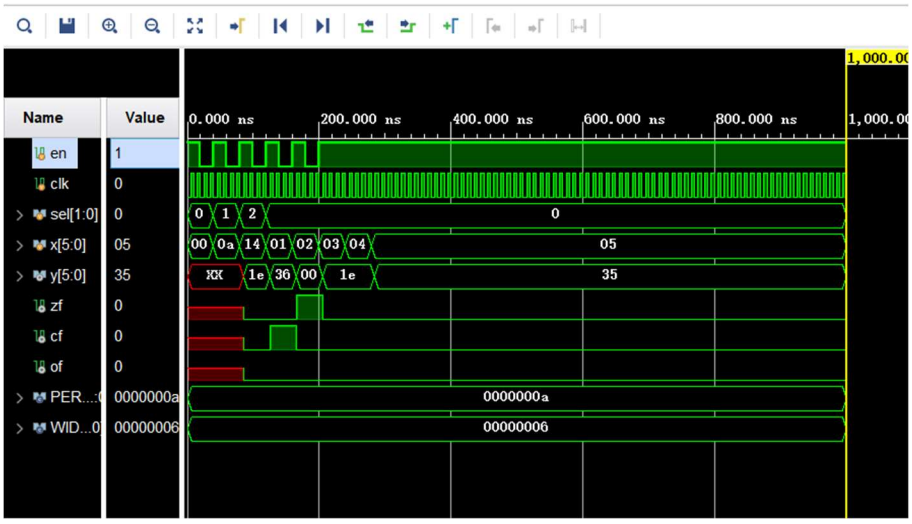


逻辑设计：编写加、减、与、或、非、异或 6 个模块，分别实现 6 个模块，在 top module 中调用 6 个模块通过选择器选择输出结果。

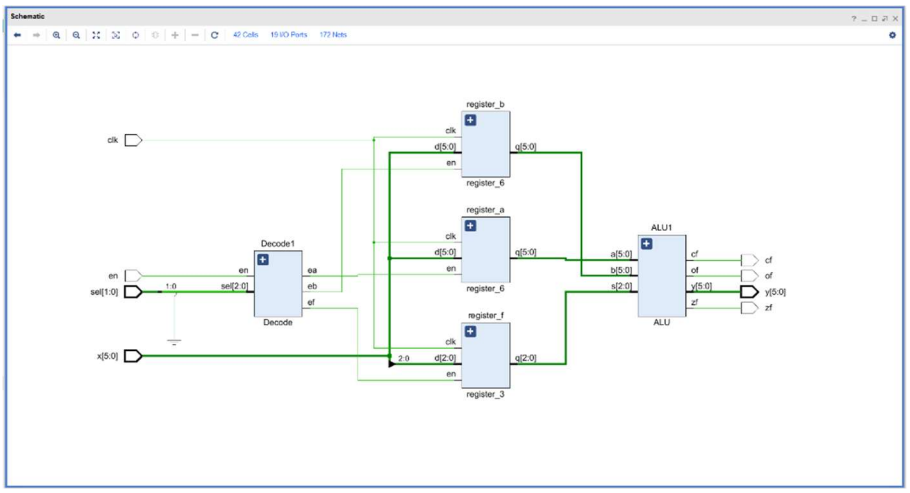
PS:拓展了 cf 作为进位/借位标志(有符号数)

of 作为溢出标准(无符号数)

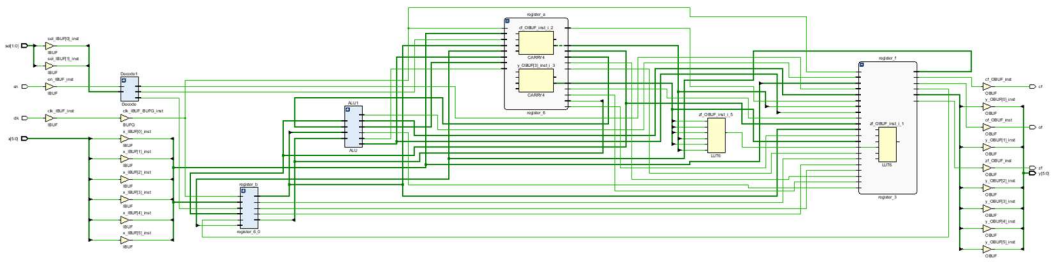
仿真结果



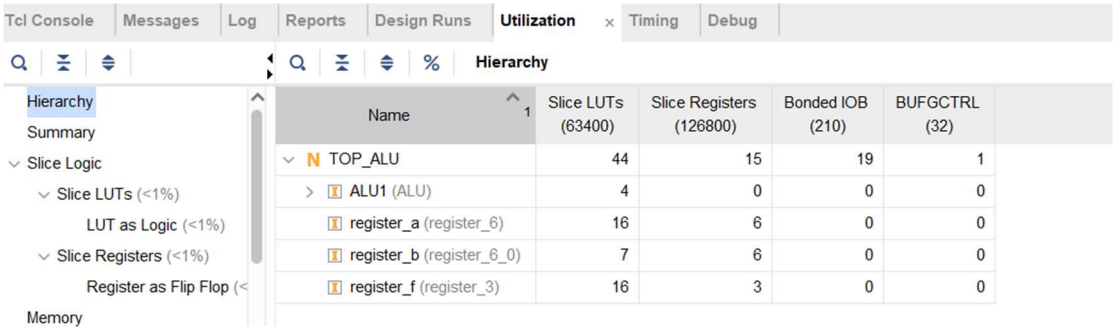
RTL 电路



Synth 电路



资源占用



Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
TOP_ALU	44	15	19	1
ALU1 (ALU)	4	0	0	0
register_a (register_6)	16	6	0	0
register_b (register_6_0)	7	6	0	0
register_f (register_3)	16	3	0	0

时间性能报告

Check Type	Slack	Corner	Lib Pin	Reference Pin	Required	Actual	Location	Pin
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X1Y84	register_a/q_reg[1]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X2Y84	register_a/q_reg[2]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X2Y84	register_a/q_reg[3]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X2Y84	register_a/q_reg[4]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X1Y83	register_b/q_reg[1]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X2Y83	register_b/q_reg[4]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X0Y84	register_f/q_reg[1]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X0Y84	register_f/q_reg[2]/C
High Pulse Width	4.500	Slow	FDRE/C	n/a	0.500	5.000	SLICE_X2Y86	register_a/q_reg[0]/C

核心代码

```
module  ALU #(
parameter WIDTH = 32    //数据宽度
)(
input wire [WIDTH-1:0] a, b,    //两操作数
input wire [2:0] s, //操作选择
output reg [WIDTH-1:0] y,   //运算结果
output reg zf,      //零标志
output reg cf,      //进位/借位标志
output reg of       //溢出标志
);
wire [WIDTH-1:0]tmpt_add_y,tmpt_sub_y,tmpt_and_y,tmpt_or_y,tmpt_xor_y,tmpt_not_y;
ADD add1(a,b,tmpt_add_y,tmpt_add_zf,tmpt_add_cf,tmpt_add_of);
SUB sub1(a,b,tmpt_sub_y,tmpt_sub_zf,tmpt_sub_cf,tmpt_sub_of);
AND and1(a,b,tmpt_and_y,tmpt_and_zf,tmpt_and_cf,tmpt_and_of);
OR or1(a,b,tmpt_or_y,tmpt_or_zf,tmpt_or_cf,tmpt_or_of);
XOR xor1(a,b,tmpt_xor_y,tmpt_xor_zf,tmpt_xor_cf,tmpt_xor_of);
NOT not1(a,b,tmpt_not_y,tmpt_not_zf,tmpt_not_cf,tmpt_not_of);
```

ALU 部分

```
module  register_32#(
    parameter WIDTH = 32
    )(
    input wire clk, en,
    input wire [WIDTH-1 : 0]  d,
    output reg  [WIDTH-1 : 0] q
    );
    always @(posedge clk)
    begin
        if (en)
        q <= d;
    end
endmodule
```

寄存器部分

```
module ADD#(
parameter WIDTH = 32    //数据宽度
)(
input wire [WIDTH-1:0] a, b,    //两操作数
output reg [WIDTH-1:0] y,   //运算结果
output reg zf,      //零标志
output reg cf,      //进位/借位标志
output reg of       //溢出标志
);
always@(*)
begin
{cf,y}=a+b;
of = (a[WIDTH-1]==b[WIDTH-1])&&(a[WIDTH-1]!=y[WIDTH-1]);
zf = y==0?1:0;
end
endmodule
```

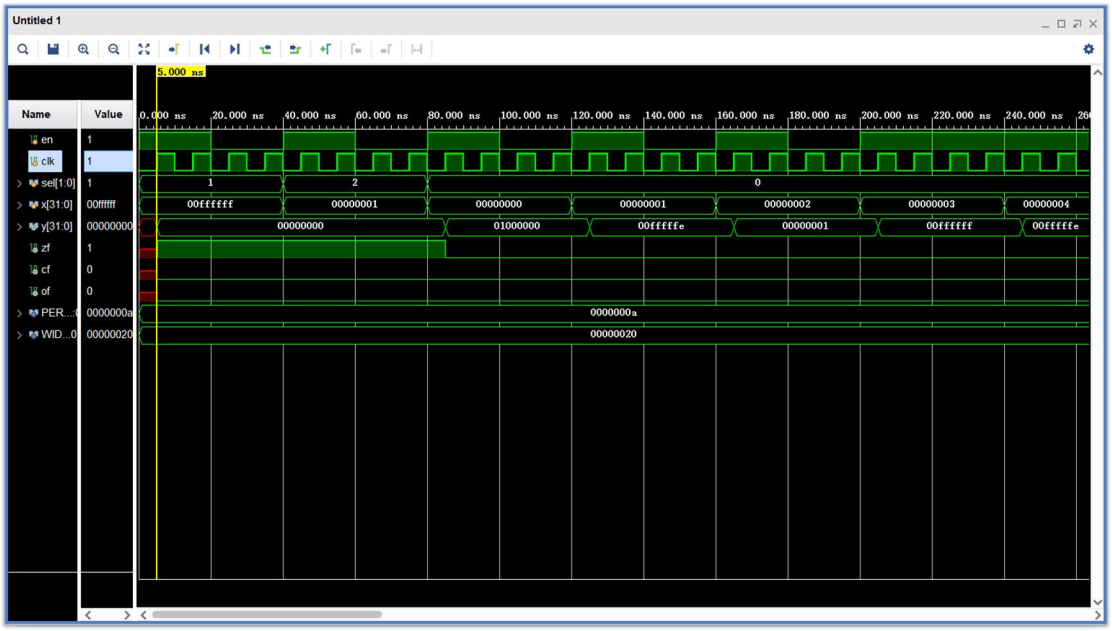
核心运算示例

此处添加了 of 溢出标志 与 cf 进位借位标志

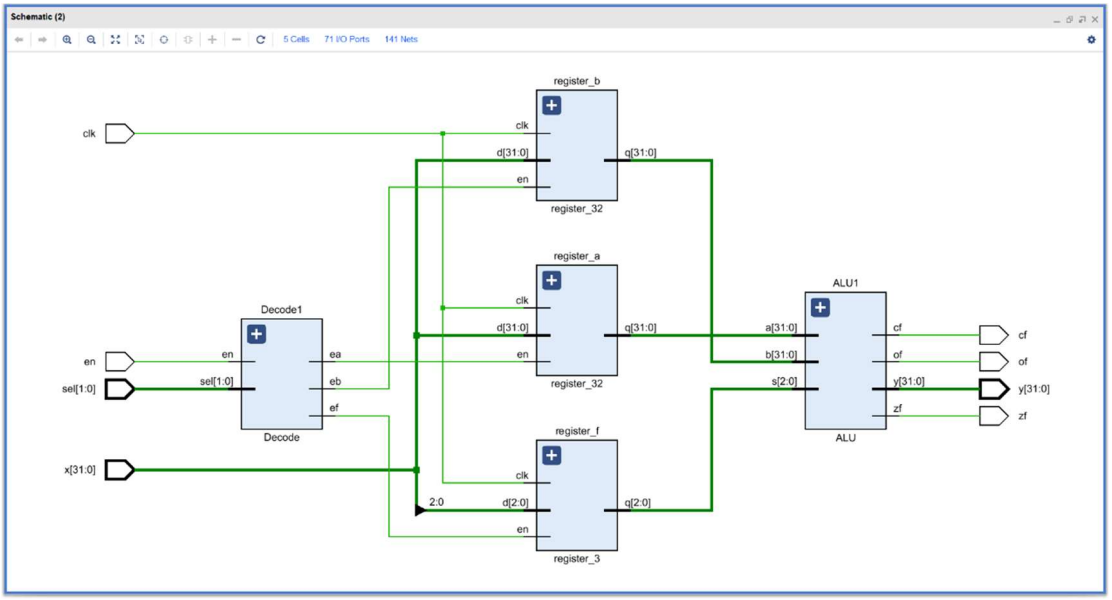
2. 算术逻辑单元（ALU）-32 位

基本设计相同

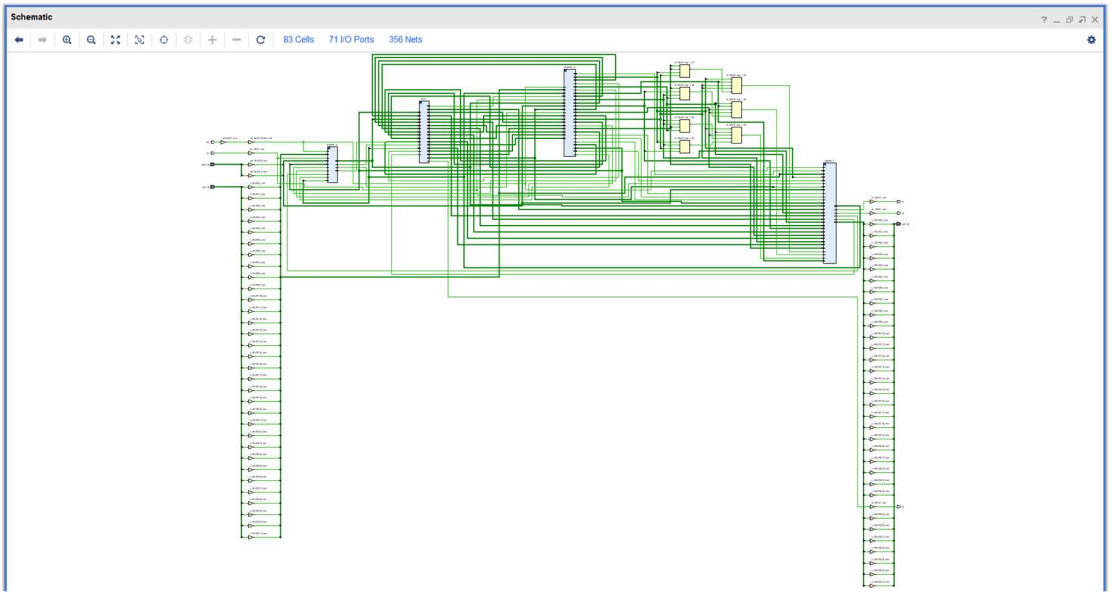
仿真结果



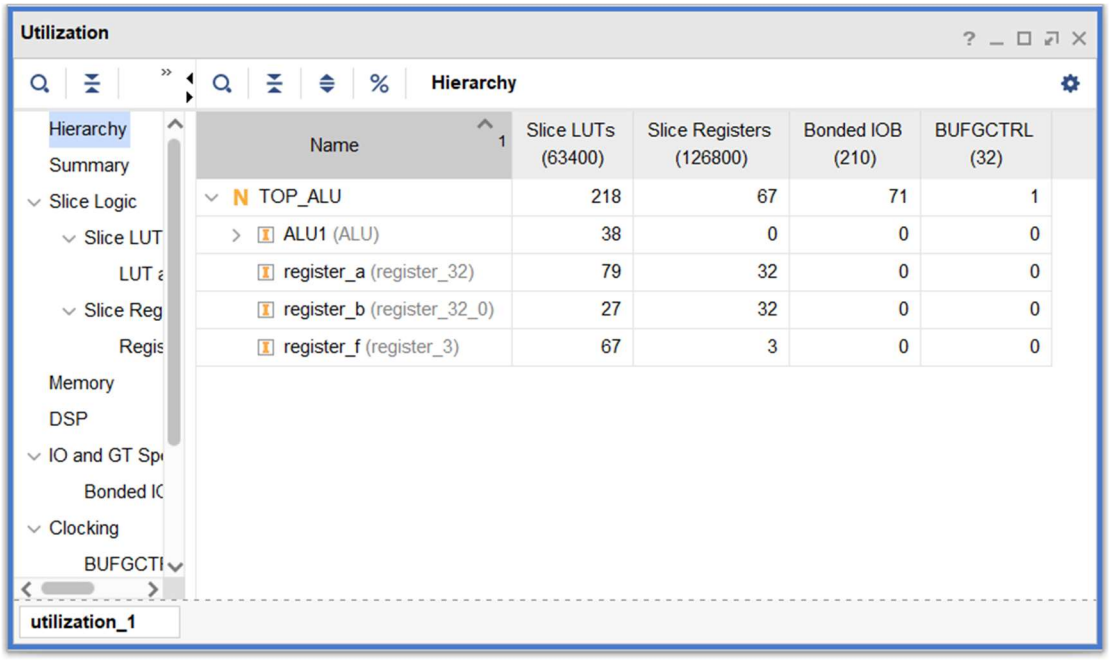
RTL 电路



Synth 电路



资源占用

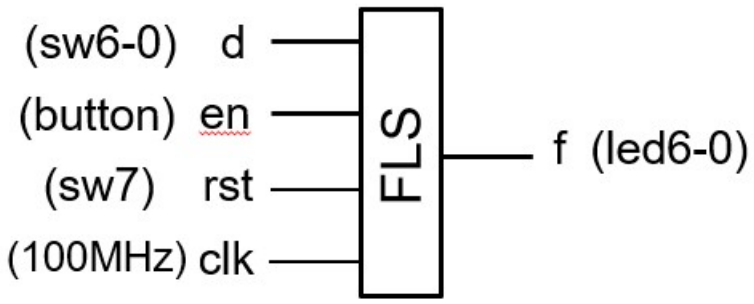


3. **FLS**

fn = fn-2 + fn-1　　(n > 1)

复位有效时，f = 0

正常工作时，fn = f0、f1、f2、f3 ……　（f0 = d0, f1 = d1）



- **FLS模块端口定义如下：**

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

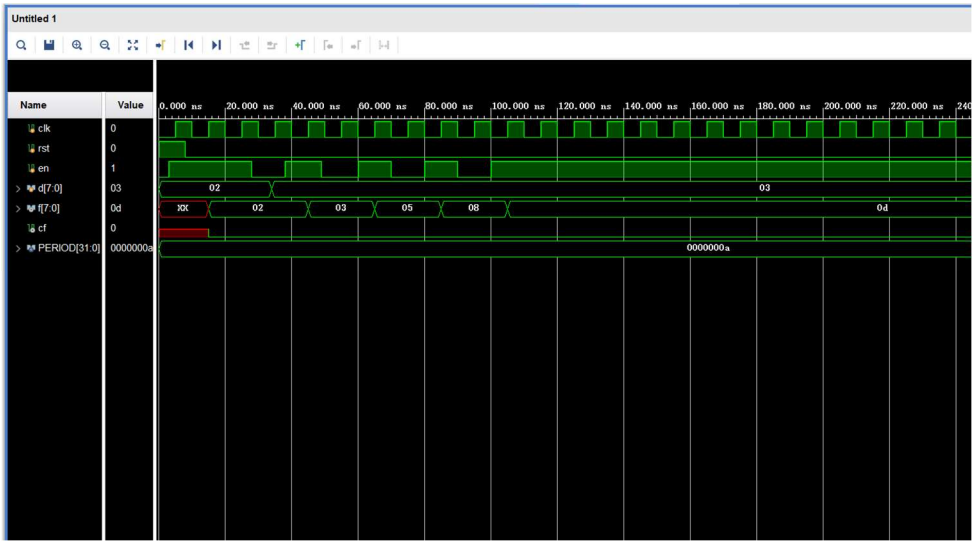
逻辑设计

FSM 描述设计状态机 输出为控制信号

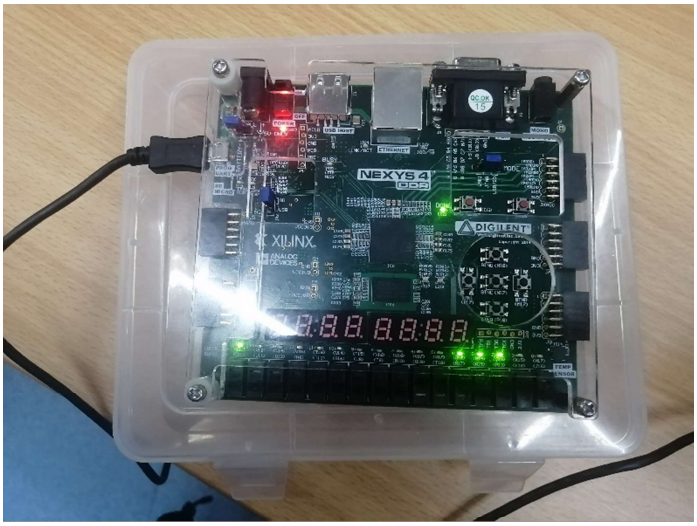
FLS 根据控制信号采取不同措施

调用 ALU 加法模块计算

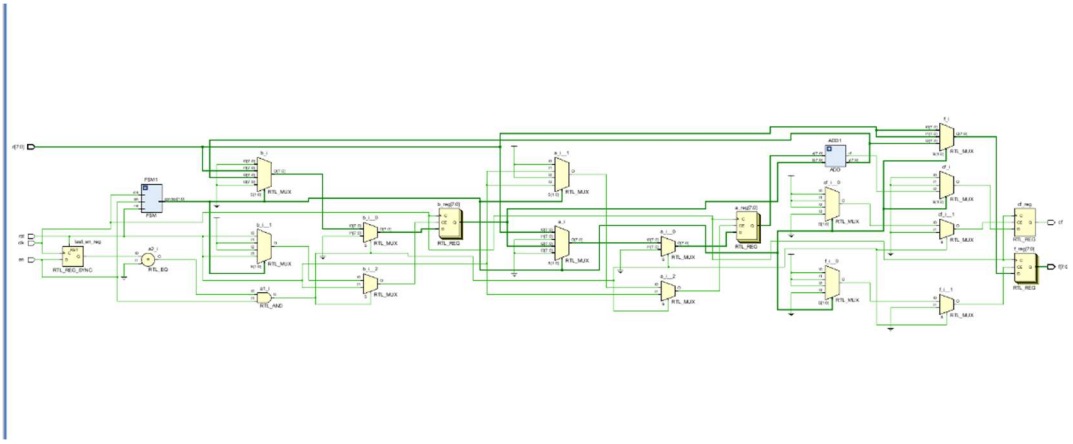
仿真结果



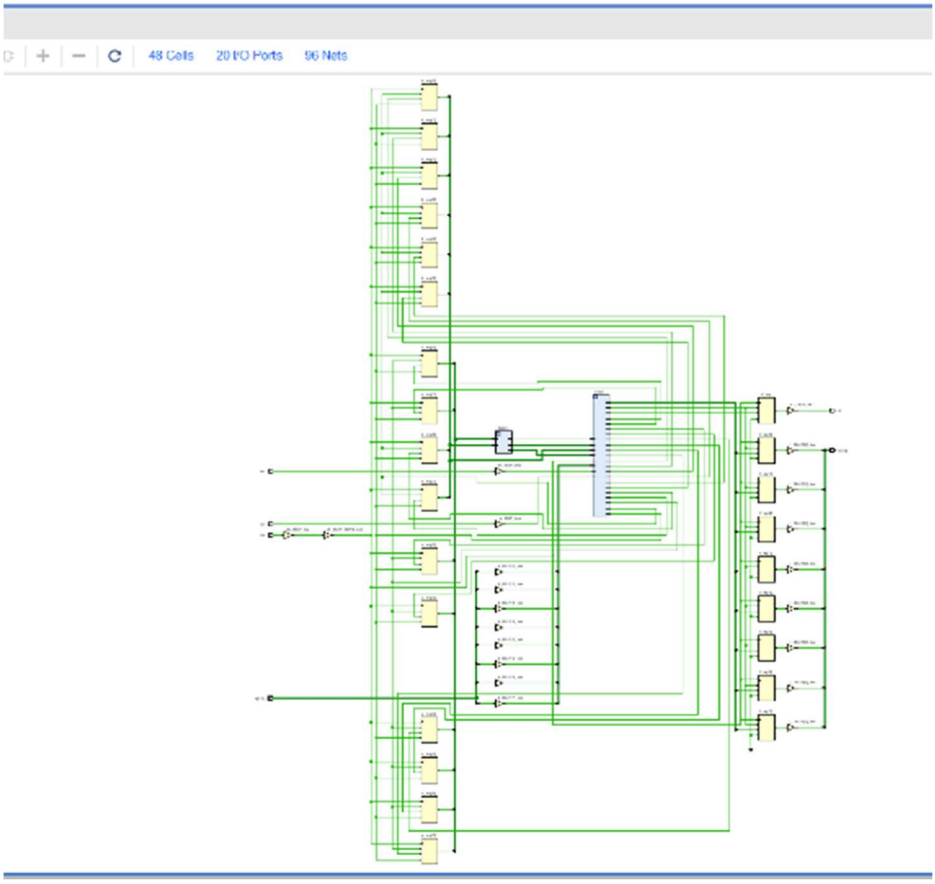
下载结果



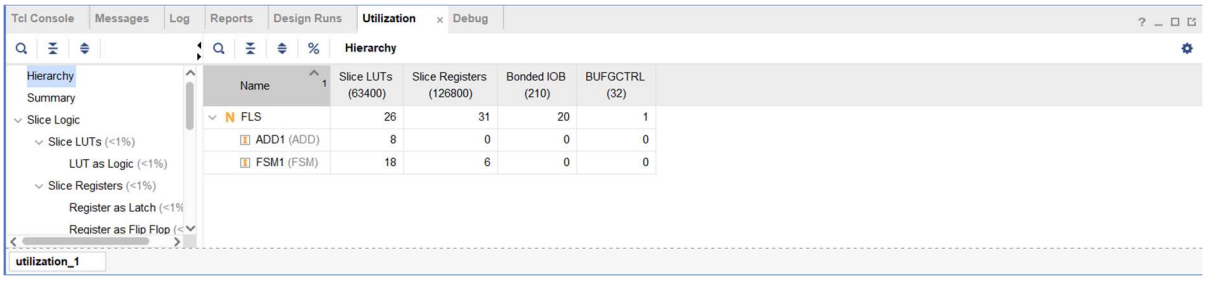
RTL 电路



Synth 电路



资源占用



时间性能分析

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	E
Path 1	6.988	4	5	2	a_reg[1]/C	b_reg[7]/D	2.876	1.936	0.940	10.0	sys_clk_pin	sys_clk_pin	
Path 2	6.988	4	5	2	a_reg[1]/C	f_reg[7]/D	2.876	1.936	0.940	10.0	sys_clk_pin	sys_clk_pin	
Path 3	7.001	5	6	2	a_reg[1]/C	cf_reg/D	2.863	2.071	0.792	10.0	sys_clk_pin	sys_clk_pin	
Path 4	7.057	4	5	2	a_reg[1]/C	f_reg[6]/D	2.807	1.863	0.944	10.0	sys_clk_pin	sys_clk_pin	
Path 5	7.065	4	5	2	a_reg[1]/C	b_reg[6]/D	2.799	1.855	0.944	10.0	sys_clk_pin	sys_clk_pin	
Path 6	7.119	4	5	2	a_reg[1]/C	b_reg[5]/D	2.745	1.941	0.804	10.0	sys_clk_pin	sys_clk_pin	
Path 7	7.119	4	5	2	a_reg[1]/C	f_reg[5]/D	2.745	1.941	0.804	10.0	sys_clk_pin	sys_clk_pin	
Path 8	7.201	3	4	2	a_reg[1]/C	f_reg[3]/D	2.663	1.723	0.940	10.0	sys_clk_pin	sys_clk_pin	
Path 9	7.209	3	4	2	a_reg[1]/C	b_reg[3]/D	2.655	1.715	0.940	10.0	sys_clk_pin	sys_clk_pin	
Path 10	7.236	4	5	2	a_reg[1]/C	b_reg[4]/D	2.628	1.825	0.803	10.0	sys_clk_pin	sys_clk_pin	

核心代码

```
module FLS(
    input  wire clk, rst, en,
    input  wire [7 : 0] d,
    output reg [7 : 0]  f,
    output reg cf
    );
wire [1:0]control;
FSM FSM1(clk,rst,en,control);
reg last_en;
reg [7 : 0] a;
reg [7 : 0] b;

wire [7 : 0] sum_ab;
wire sum_ab_cf;
ADD ADD1(a,b,sum_ab,sum_ab_cf);
```

```
// 描述CS
   always  @(posedge clk)
   begin

       if (rst)
       begin
           cs <= S0;  //同步复位
           last_en<=0;
       end
       else
           begin
               cs <= ns;
               last_en<=en;
           end

   end
```

```
// 描述NS
   always  @*
   begin
       ns = cs;         //默认赋值
   case (cs)
       S0:
       begin
           control = 0;
           if(!(last_en==0 && en==1))
               ns = S0;
           else
               ns = S1;
       end
       S1:
       begin
           control = 1;
           if(!(last_en==0 && en==1))
```

【结果分析】

ALU 设计能分时复用地输入数据和运算选择，同时完成所给出 6 种运算，并且拓展了 of 溢出与 cf 进位标识符。

FLS 设计能完成所要求的计算斐波那契数列，同时拓展了 cf 进位标识符。

【实验总结】

复习了 VERILOG 语法与 vivado 使用

掌握时序逻辑与组合逻辑的基本实现

【实验建议】

建议多提前几天放出题目