计算机组成原理 实验报告

姓名：杨博涵　　学号：PB20000328　实验日期：2022.3.16

一、实验题目：

Lab01 运算器

二、实验目的：

设计一算术逻辑运算单元（ALU），实现加、减、与、或、异或功能；利用前述的 ALU 模块与适当的硬件电路，完成 Fibonacci 数列输出功能。

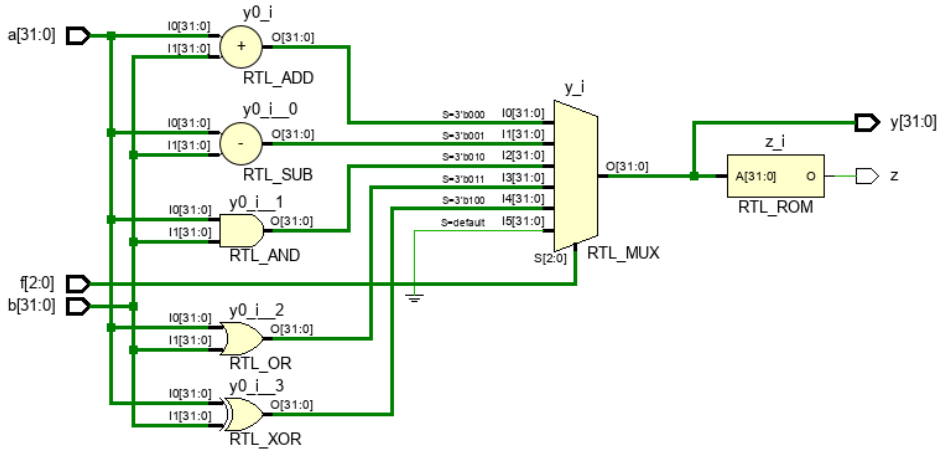
三、实验平台：

Vivado

四、实验过程：

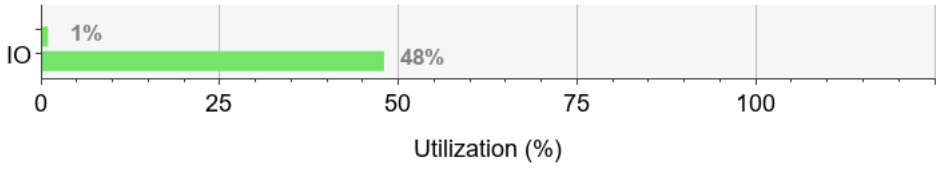
源码在最后给出。

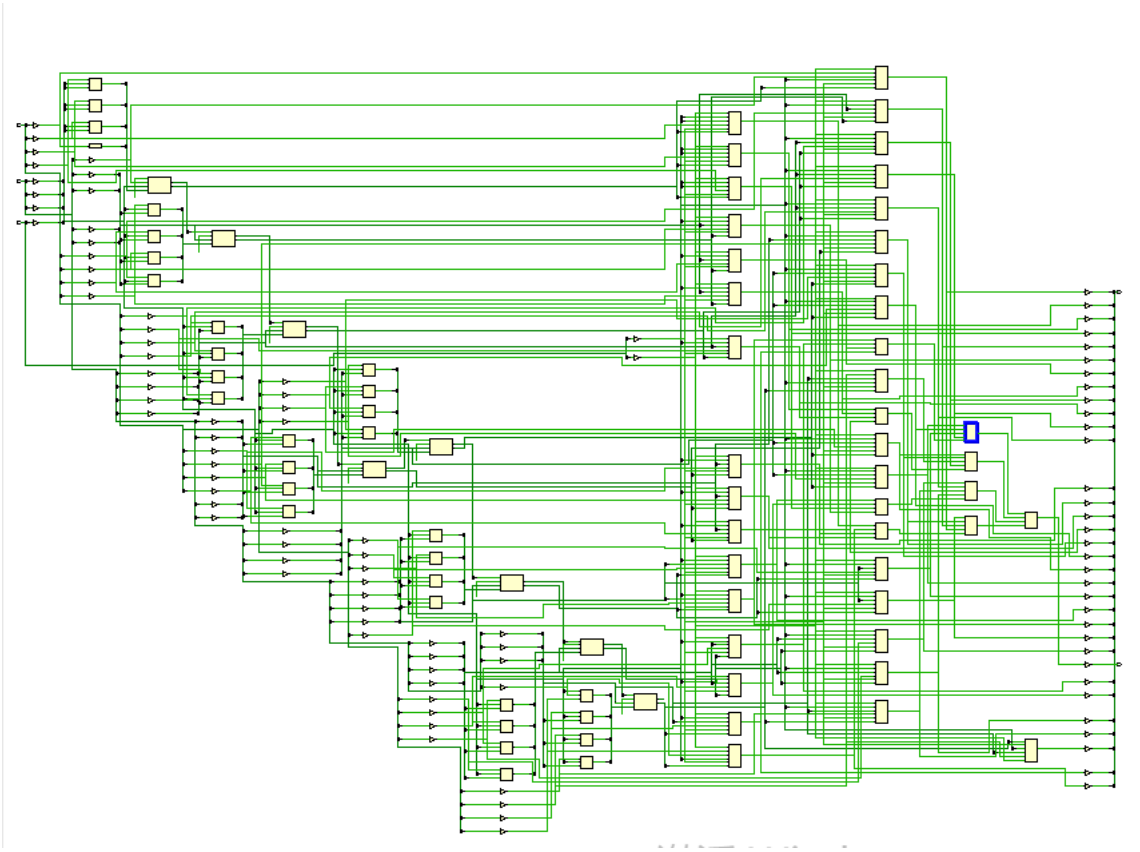
1. 32 位 ALU 设计



Summary

Resource	Utilization	Available	Utilization %
LUT	74	63400	0.12
IO	100	210	47.62





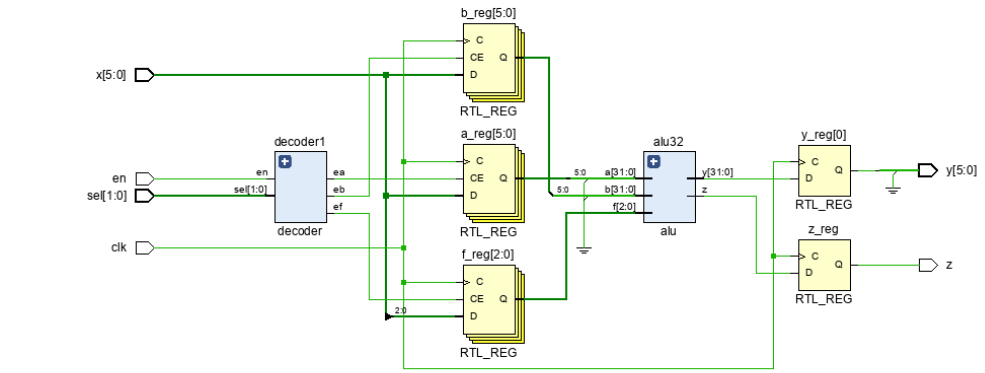
2.6 位 ALU 设计

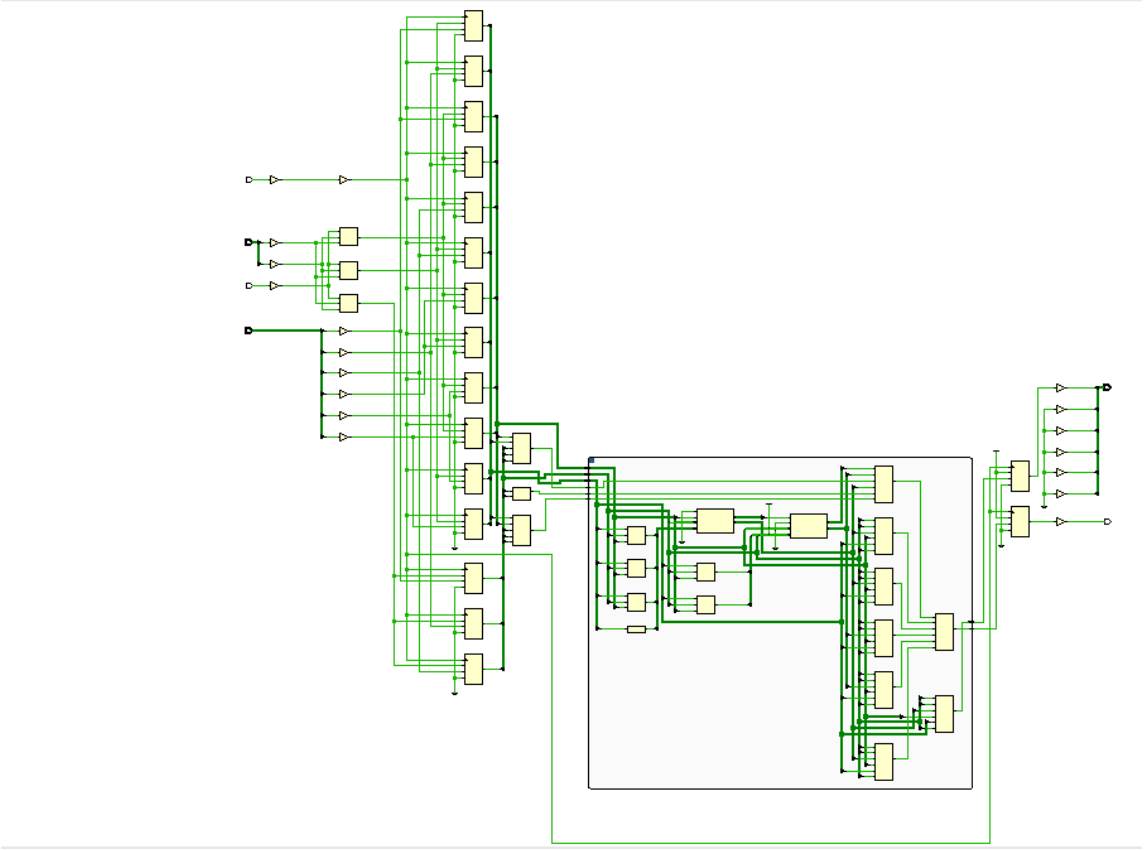
Utilization | Timing

Unconstrained Paths - NONE - NONE - Hold

- General Information
- Timer Settings
- Design Timing Summary
- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clo
Path 11	∞	3	4	33	f[1]	y[0]	2.073	1.396	0.677	-∞	input port cl
Path 12	∞	3	4	33	f[1]	y[10]	2.073	1.396	0.677	-∞	input port cl
Path 13	∞	3	4	33	f[1]	y[11]	2.073	1.396	0.677	-∞	input port cl
Path 14	∞	3	4	33	f[1]	y[12]	2.073	1.396	0.677	-∞	input port cl
Path 15	∞	3	4	33	f[1]	y[13]	2.073	1.396	0.677	-∞	input port cl
Path 16	∞	3	4	33	f[1]	y[14]	2.073	1.396	0.677	-∞	input port cl
Path 17	∞	3	4	33	f[1]	y[15]	2.073	1.396	0.677	-∞	input port cl
Path 18	∞	3	4	33	f[1]	y[16]	2.073	1.396	0.677	-∞	input port cl
Path 19	∞	3	4	33	f[1]	y[17]	2.073	1.396	0.677	-∞	input port cl
Path 20	∞	3	4	33	f[1]	y[18]	2.073	1.396	0.677	-∞	input port cl



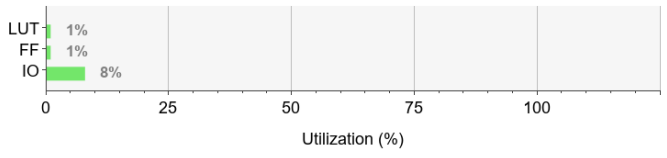


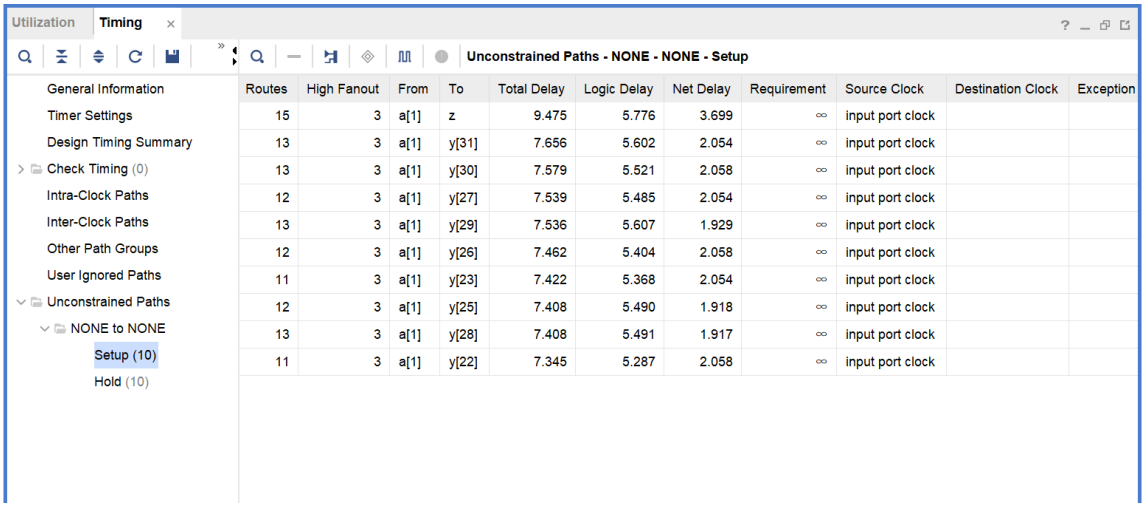
Tcl Console | Messages | Log | Reports | Design Runs | **Utilization** ×

Summary

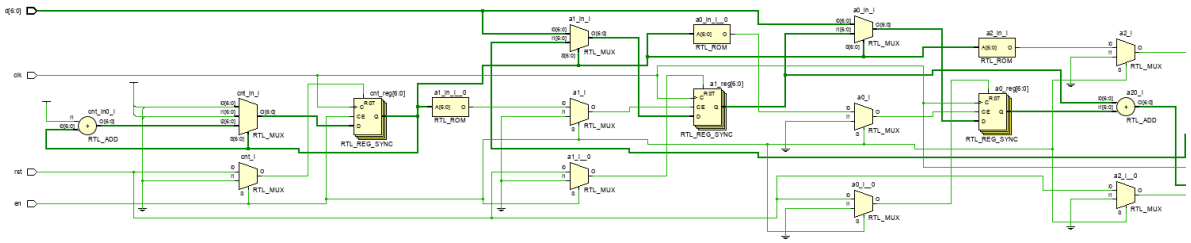
- Hierarchy
- Summary
- Slice Logic
  - Slice LUTs (<1%)
    - LUT as Logic (<1
  - Slice Registers (<1%)
    - Register as Flip F
- Memory
- DSP
- IO and GT Specific
  - Bonded IOB (8%)
- Clocking
  - BUFGCTRL (3%)

Resource	Utilization	Available	Utilization %
LUT	19	63400	0.03
FF	17	126800	0.01
IO	17	210	8.10





Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
15	3	a[1]	z	9.475	5.776	3.699	∞	input port clock		
13	3	a[1]	y[31]	7.656	5.602	2.054	∞	input port clock		
13	3	a[1]	y[30]	7.579	5.521	2.058	∞	input port clock		
12	3	a[1]	y[27]	7.539	5.485	2.054	∞	input port clock		
13	3	a[1]	y[29]	7.536	5.607	1.929	∞	input port clock		
12	3	a[1]	y[26]	7.462	5.404	2.058	∞	input port clock		
11	3	a[1]	y[23]	7.422	5.368	2.054	∞	input port clock		
12	3	a[1]	y[25]	7.408	5.490	1.918	∞	input port clock		
13	3	a[1]	y[28]	7.408	5.491	1.917	∞	input port clock		
11	3	a[1]	y[22]	7.345	5.287	2.058	∞	input port clock		



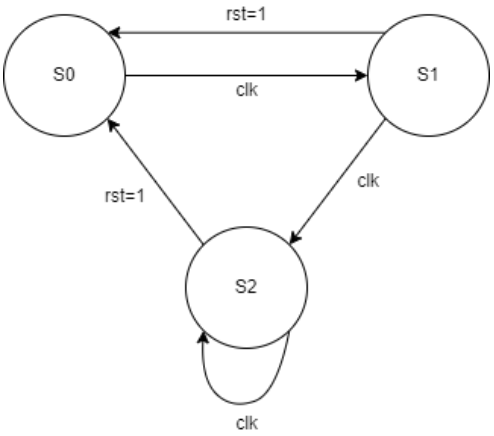
3. FLS 设计

由时序图可知，使能信号是输入控制核心，即当 en=0 时，FSM 不接受输入，故以下讨论均基于 en=1。

复位信号为同步。复位过后的接下来的连续两个输入将作为初项，之后不再接受外部输入，而是自行在时钟上升沿递推下一项。

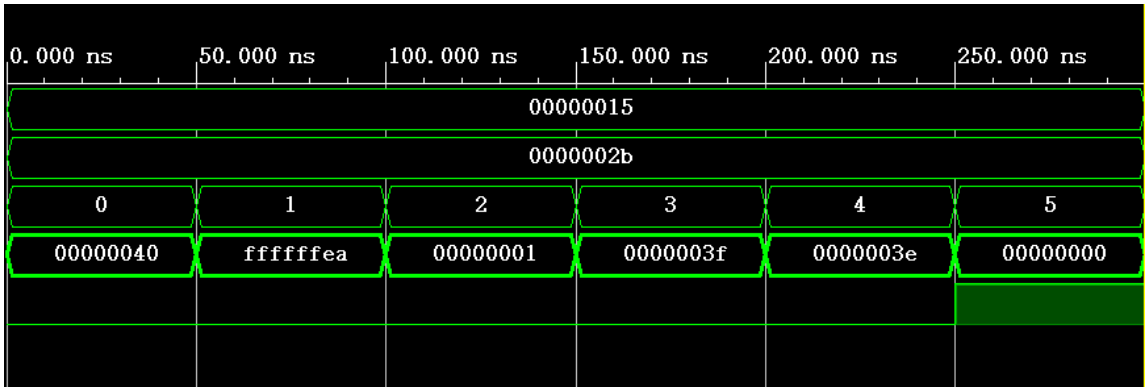
所以我们采用 3 个 32 位寄存器作为连续三项的存储，1 个寄存器作为输入计数，ALU 采用 32 位 ALU 直接计算。当输入计数>=2 时就自行工作。

状态转换图如下



五、实验结果：

32 位 ALU 的仿真结果如下：



6 位 ALU 与 fls 均在 fpgaol 上操作，具体现象已线下检查，均正常工作。

六、心得体会：

作为第一个项目完成了上手的准备工作，提高了熟练度。学习了三段式代码的写作方式和相应的纠错能力。

源码：

1. testbench

```
module alu32sim();
    reg[31:0] a=5'b10101,b=6'b101011;
    reg[2:0] f;
    wire[31:0] y;
    wire z;

    alu alublock(.a(a),.b(b),.f(f),.y(y),.z(z));


    initial
        begin
            f=3'b000;
        #50 f=3'b001;
        #50 f=3'b010;
        #50 f=3'b011;
        #50 f=3'b100;
        #50 f=3'b101;
        #50
        $stop;
        end

endmodule
```

2. fls

```
`define S0 1'b0
`define S1 1'b1
`define S2 2'b10


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

wire en_eff,en_cln;
reg [6:0] curr_state,next_state,a2,a1,a0,a2_in,a1_in,a0_in;

jitter_clr clear(.clk(clk),.button(en),.button_clean(en_cln));
signal_edge edger(.clk(clk),.button(en_cln),.button_edge(en_eff));

always@(*)
    begin
//                          case(cnt)
//                              `S0 :    begin a0_in=d; a1_in=a1; a2_in=a2;
cnt_in=cnt+1'b1; f=d; end
//                              `S1 :    begin a0_in=a0; a1_in=d; a2_in=a0+d;
cnt_in=cnt+1'b1; f=d; end
//                              default: begin cnt_in=cnt+1'b1; a0_in=a1; a1_in=a2;
a2_in=a1+a2; f=a1+a2; end
        case(curr_state)
            `S0: if(rst==1'b1)
```

```
                                next_state=`S0;
                        else next_state=`S1;
                `S1: if(rst==1'b1)
                                next_state=`S0;
                        else next_state=`S2;
                `S2: if(rst==1'b1)
                                next_state=`S0;
                        else next_state=`S2;
        endcase
    end

always@(posedge clk)
    begin
        if(en_eff==1'b1)
            begin
                a2<=a2_in;
                a1<=a1_in;
                a0<=a0_in;
                curr_state<=next_state;
            end
    end

always @(*)
    begin
        case(curr_state)
            `S0:begin
                    a0_in=d;
                    a1_in=0;
                    a2_in=0;
                    f=d;
                end

            `S1:begin
                    a0_in=a0;
                    a1_in=d;
                    a2_in=d+a0;
                    f=d;
                end

            `S2:begin
                    a0_in=a1;
                    a1_in=a2;
                    a2_in=a2+a1;
                    f=a2;
                end
        endcase
    end
endmodule




//en 信号处理
module jitter_clr
(
```

```
    input clk,
    input button,
    output button_clean
);
    reg [3:0] cnt;
always@(posedge clk)
    begin
        if(button==1'b0) cnt<=4'h0;
        else if(cnt<4'h8) cnt<=cnt+1'b1;
    end

assign button_clean=cnt[3];
endmodule


module signal_edge
(
    input clk,
    input button,
    output button_edge
);

reg button_r1,button_r2;
always@(posedge clk)
    button_r1<=button;
always@(posedge clk)
    button_r2<=button_r1;

assign button_edge=button_r1&(~button_r2);

endmodule
```

3. alu

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/03/16 17:05:49
// Design Name:
// Module Name: alu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

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

reg [WIDTH-1:0] y;
assign z=y==0 ? 1'b1 : 1'b0;
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

endmodule
```

```
module alu6
(
    input  clk,
    input  en,
    input  [1:0]sel,
    input  [5:0] x,
    output [5:0] y,
    output z
);
wire ef,ea,eb,zin;
wire [5:0] yin;
reg [2:0] f;
reg [5:0] a,b,y;
reg z;

decoder decoder1(.en(en),.sel(sel),.ef(ef),.ea(ea),.eb(eb));
alu alu32(.y(yin),.z(zin),.a(a),.b(b),.f(f));

always@(posedge clk)
    begin
            if(ef==1'b1) f<=x[2:0];
            if(ea==1'b1) a<=x;
            if(eb==1'b1) b<=x;
    end

always@(posedge clk)
    begin
            y<=yin;
```

```
            z<=zin;
    end

endmodule


module decoder
(
    input en,
    input [1:0] sel,
    output ef,
    output ea,
    output eb
);

reg ef,ea,eb;

always @(*)
    begin
        if(en==1)
            begin
            case(sel)
                2'b00:begin ef=1'b0; ea=1'b1; eb=1'b0; end
                2'b01:begin ef=1'b0; ea=1'b0; eb=1'b1; end
                2'b10:begin ef=1'b1; ea=1'b0; eb=1'b0; end
                2'b11:begin ef=1'b0; ea=1'b0; eb=1'b0; end
            endcase
            end
        else    begin ef=1'b0; ea=1'b0; eb=1'b0;  end
    end

endmodule
```