**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：**

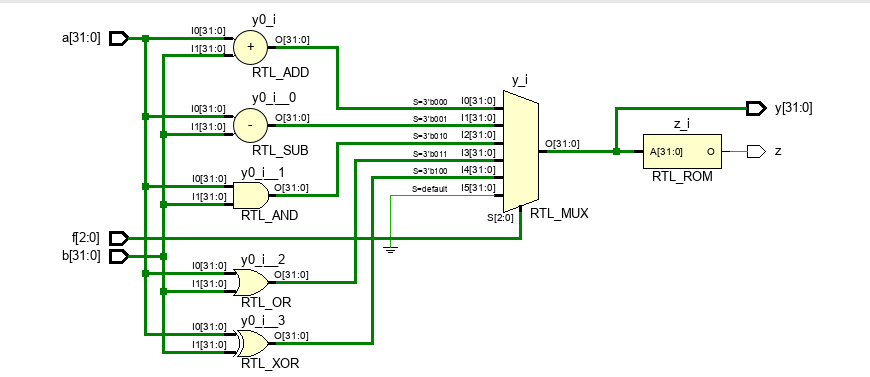
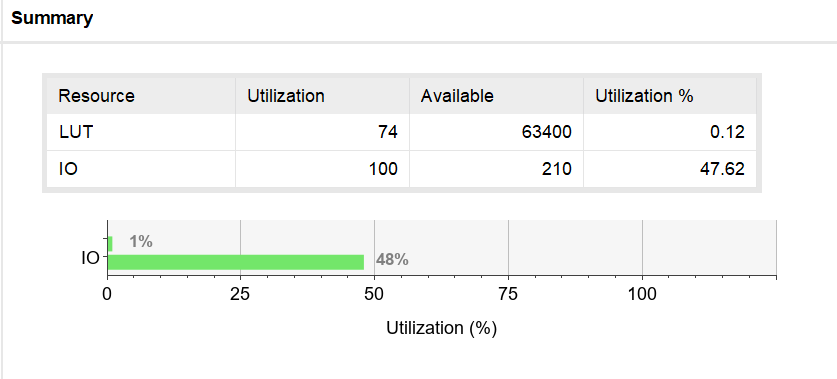
设计一算术逻辑运算单元（ALU），实现加、减、与、或、异或功能；利用前述的ALU模块与适当的硬件电路，完成Fibonacci数列输出功能。

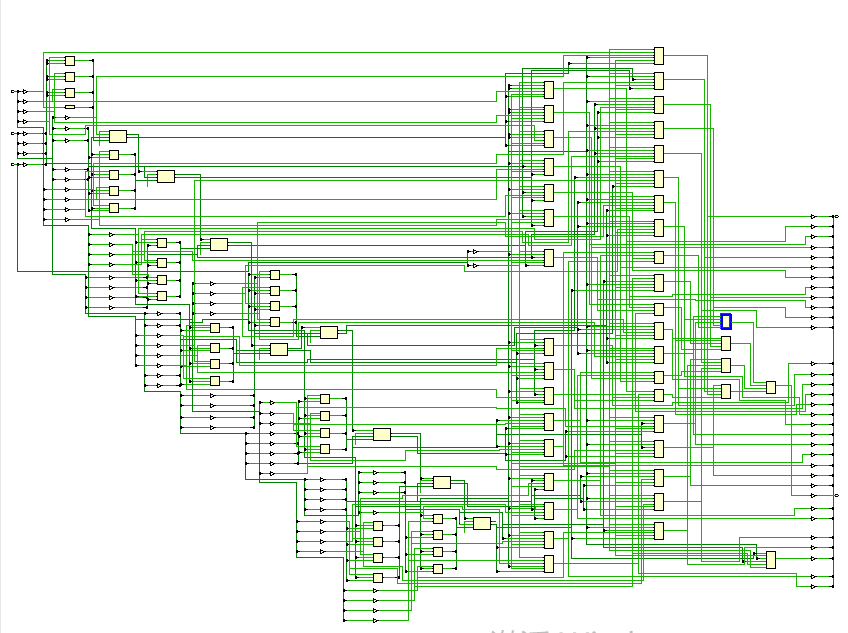
**三、实验平台：**

Vivado

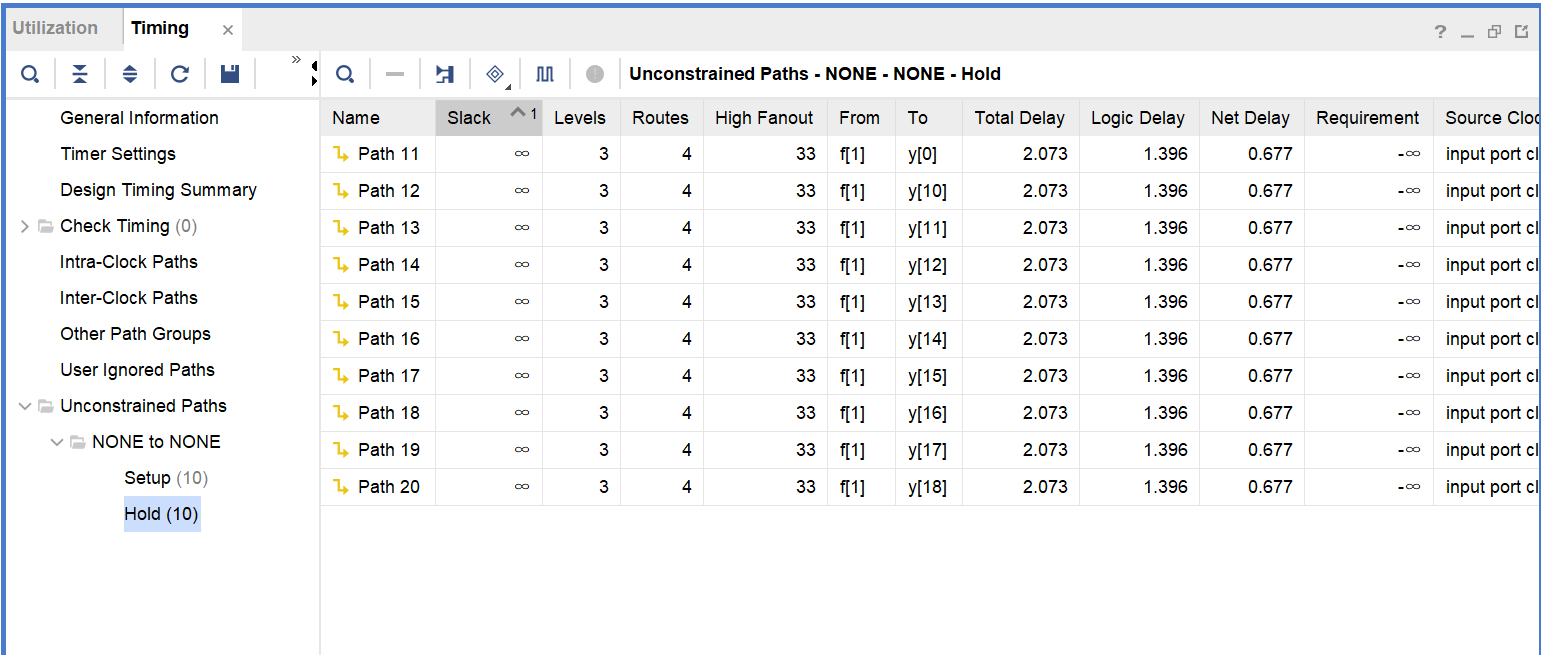
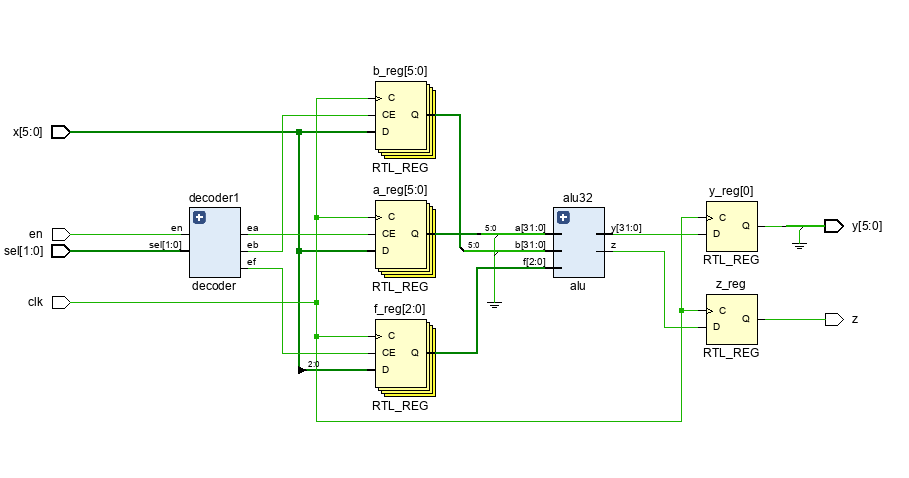
**四、实验过程：**

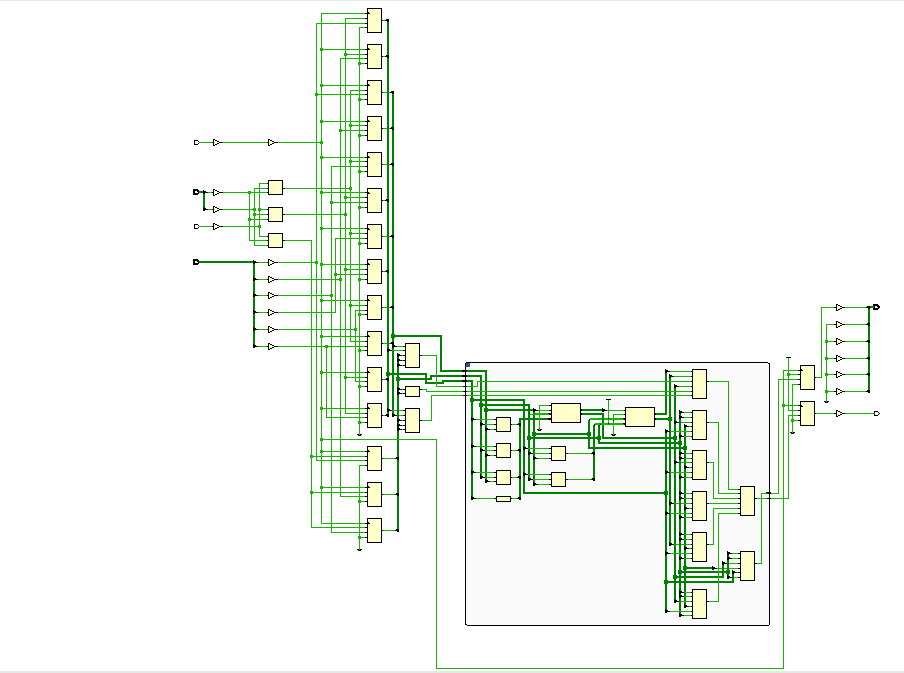
源码在最后给出。

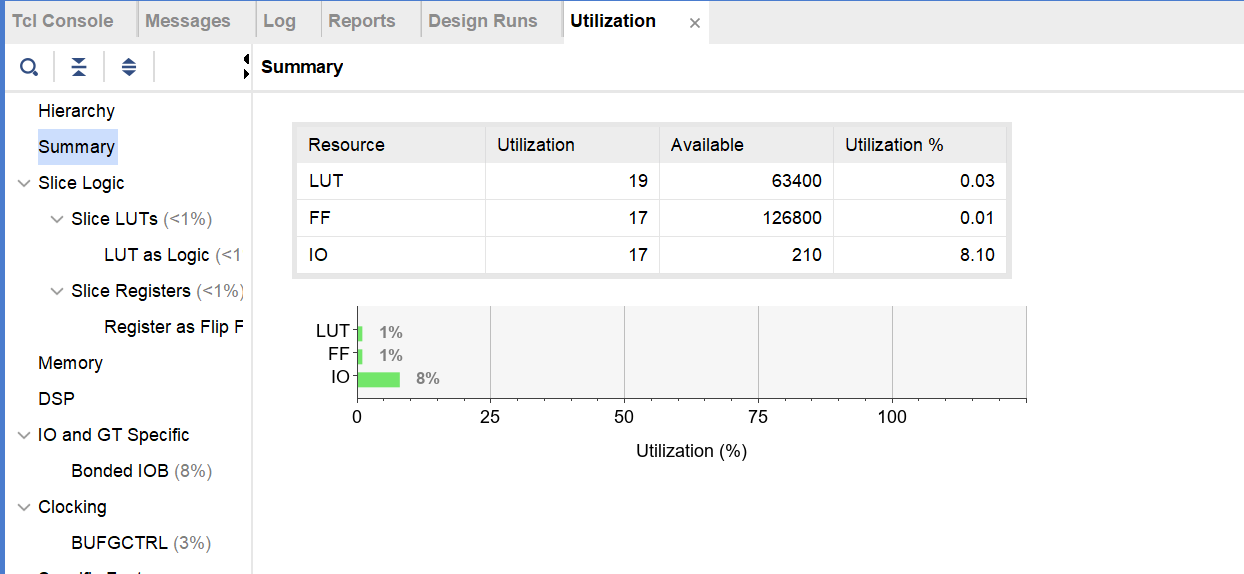
1.32位ALU设计

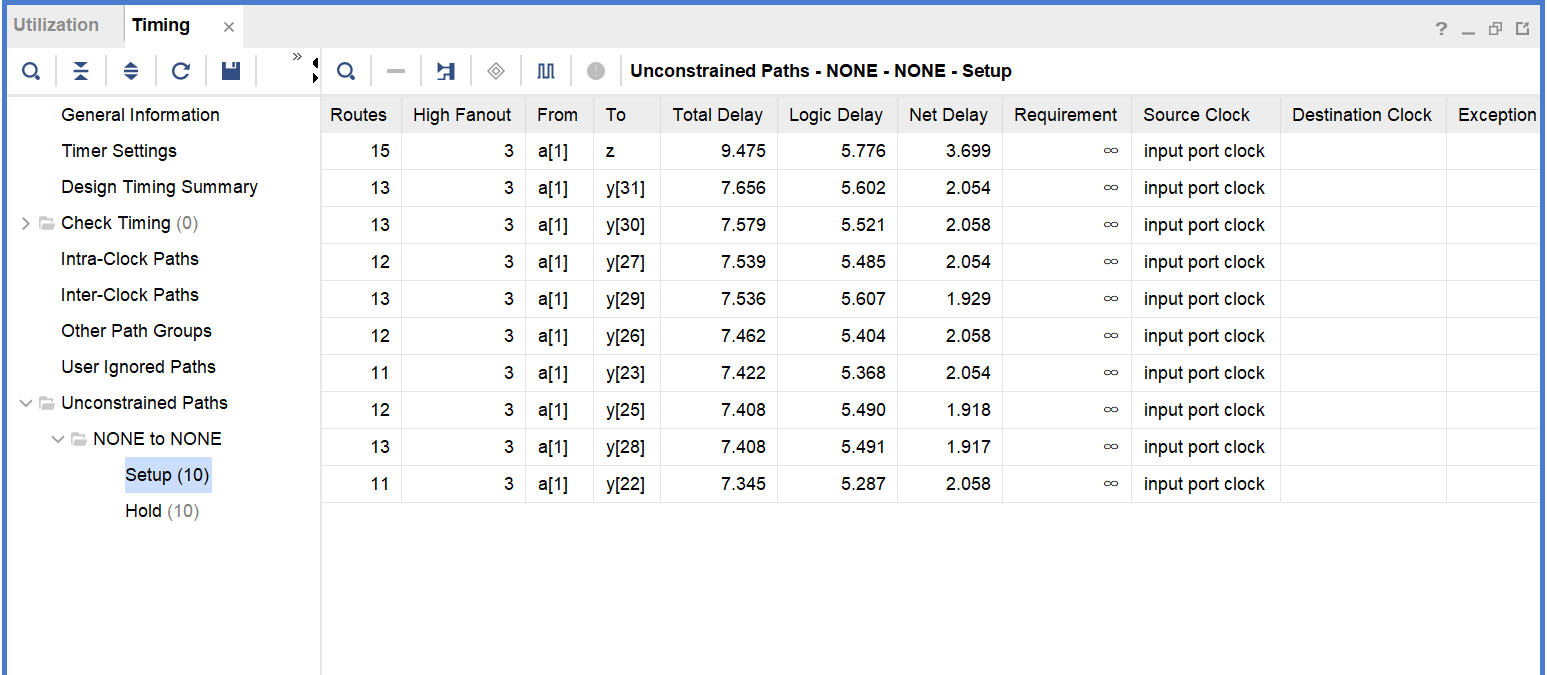
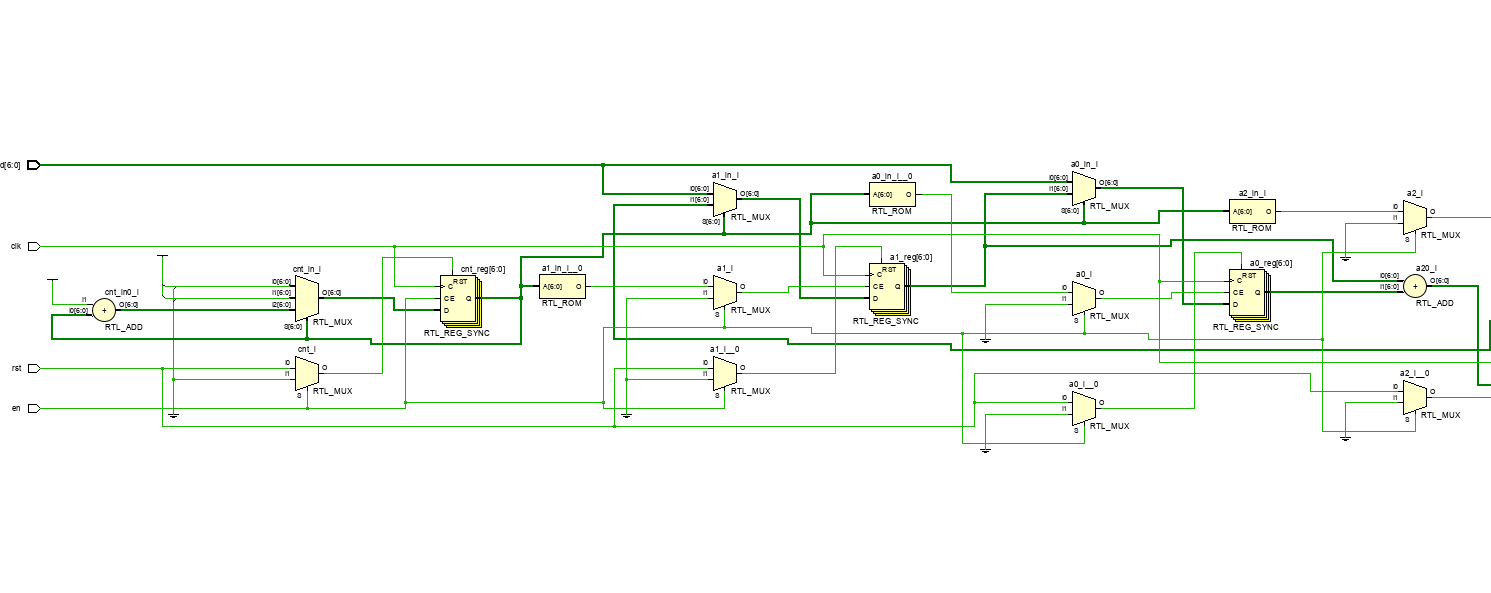


2.6位ALU设计







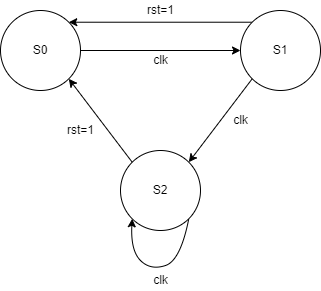
3.FLS设计

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。

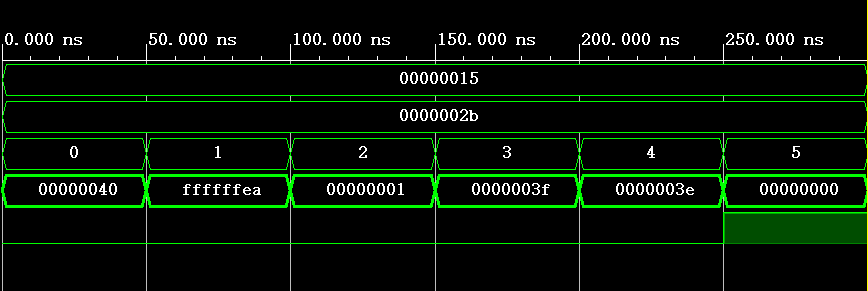
复位信号为同步。复位过后的接下来的连续两个输入将作为初项，之后不再接受外部输入，而是自行在时钟上升沿递推下一项。

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

1. 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

1. 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