实验三 层次结构设计方法及应用

**一、实验目的**

1. 进一步掌握Verilog中的基本语法和语句。
2. 熟悉Logisim软件的使用。

**二、实验内容**

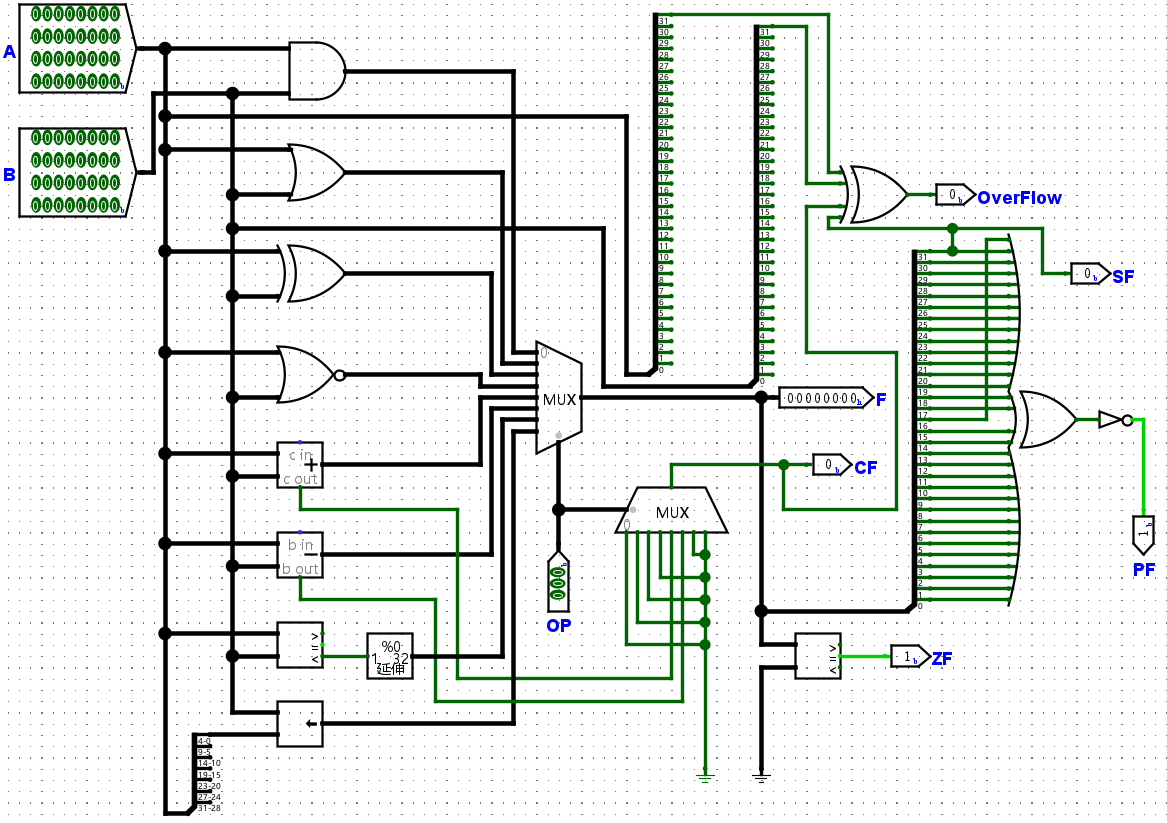
1. 掌握Verilog三种描述方式的使用。
2. 完成1位半加器、32位全加器模块的设计。
3. 设计一个基本的32位算术逻辑运算（ALU）模块。
4. 观察记录分析仿真波形。
5. 或者在Logisim中完成设计并验证。

**三、实验要求**

1. 画出模块的电路图。
2. 分析电路的仿真波形/真值表，标出关键的数值。
3. 记录设计和调试过程。

**四、实验代码及结果**

Logisim实现：



Vivado实现：

1）1位半加器

代码：

module halfadder(S, C, A, B, EN);

input A, B, EN;

output S, C;

reg S, C;

always @(A, B, EN) begin

C = 0;

if(EN) begin

S = A^B;

C = A&B;

end

else S = 1'bz;

end

endmodule

module sim\_half();

reg A, B, EN;

wire S, C;

initial begin

EN = 1;

fork

{A,B} = {1'b0,1'b0};

#10 {A,B} = {1'b0,1'b1};

#20 {A,B} = {1'b1,1'b0};

#30 {A,B} = {1'b1,1'b1};

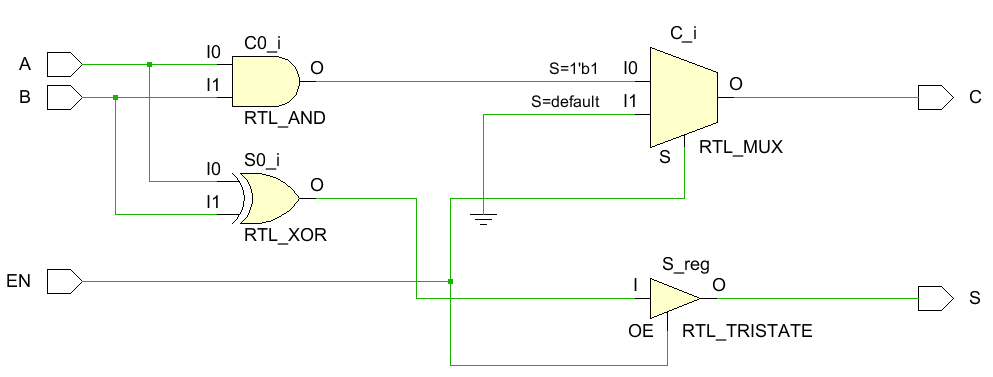
join

end

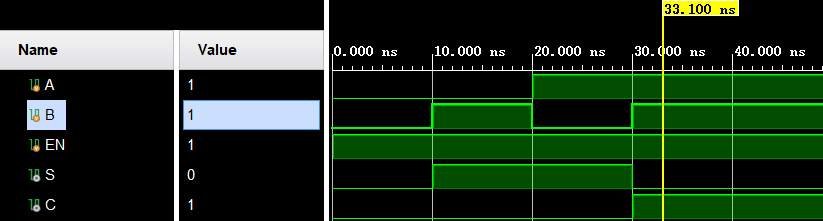
halfadder half(S, C, A, B, EN);

endmodule

电路图：



仿真波形：



2）32位全加器

代码：

module fulladder(S, Co, A, B, Ci, EN);

input A, B, Ci, EN;

output S, Co;

wire S1, D1, D2;

halfadder HA1(S1, D1, A, B, EN);

halfadder HA2(S, D2, S1, Ci, EN);

or G1(Co, D2, D1);

endmodule

module addr\_4bit(S, C3, A, B, Ci, EN);

input [3:0] A, B;

input Ci, EN;

output [3:0] S;

output C3;

wire C0, C1, C2;

fulladder FA0(S[0], C0, A[0], B[0], Ci, EN),

FA1(S[1], C1, A[1], B[1], C0, EN),

FA2(S[2], C2, A[2], B[2], C1, EN),

FA3(S[3], C3, A[3], B[3], C2, EN);

endmodule

module addr\_32bit(S, C31, A, B, Ci, EN);

input [31:0] A, B;

input Ci, EN;

output [31:0] S;

output C31;

wire C3, C7, C11, C15, C19, C23, C27;

addr\_4bit FA4\_0(S[3:0], C3, A[3:0], B[3:0], Ci, EN),

FA4\_1(S[7:4], C7, A[7:4], B[7:4], C3, EN),

FA4\_2(S[11:8], C11, A[11:8], B[11:8], C7, EN),

FA4\_3(S[15:12], C15, A[15:12], B[15:12], C11, EN),

FA4\_4(S[19:16], C19, A[19:16], B[19:16], C15, EN),

FA4\_5(S[23:20], C23, A[23:20], B[23:20], C19, EN),

FA4\_6(S[27:24], C27, A[27:24], B[27:24], C23, EN),

FA4\_7(S[31:28], C31, A[31:28], B[31:28], C27, EN);

endmodule

module sim\_addr\_32bit();

reg [31:0] A, B;

reg Ci, EN;

wire [31:0] S;

wire Co;

initial begin

A = 32'b1111\_1111\_1111\_1111\_1111\_1111\_1111\_0000;

B = 0;

Ci = 1;

EN = 1;

fork

#100 Ci = 0;

repeat(16) #10 B = B + 1;

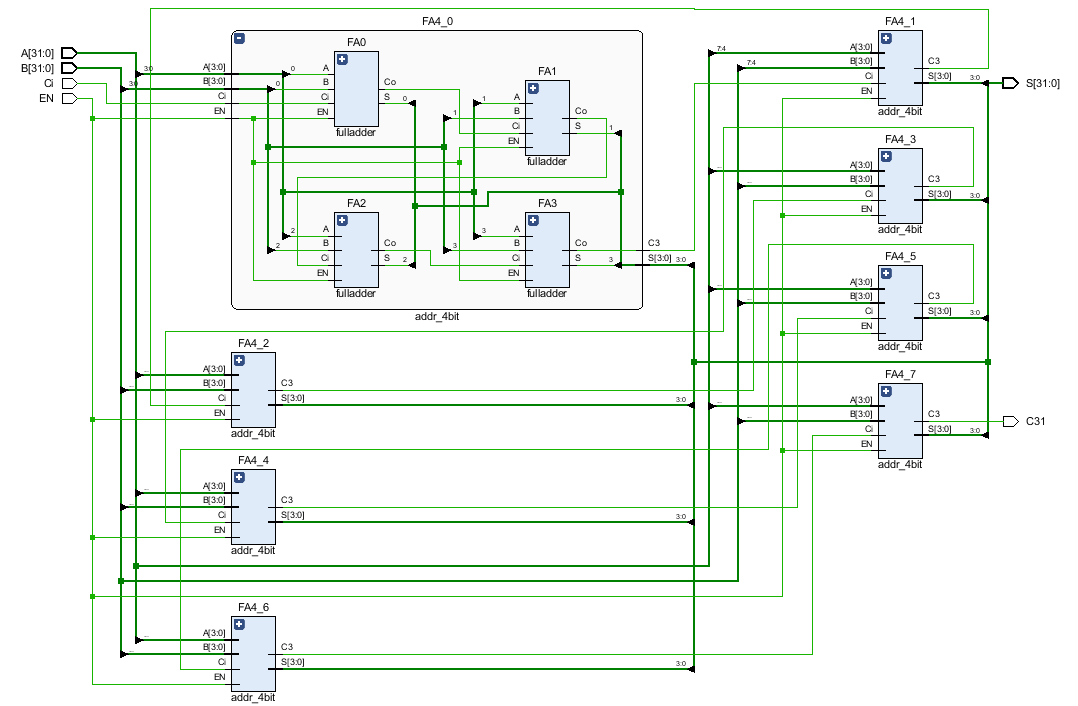
join

end

addr\_32bit FA32(S, Co, A, B, Ci, EN);

endmodule

电路图：



仿真波形：



3）32位算术逻辑运算（ALU）模块

代码：

module ALU(OP, A, B, F, ZF, CF, OF, SF, PF);

parameter SIZE = 32;

input [3:0] OP;

input [SIZE:1] A, B;

output [SIZE:1] F;

output ZF, CF, OF, SF, PF;

reg [SIZE:1] F;

reg C, ZF, OF, SF, PF;

wire [7:0] EN;

wire [SIZE:1] Fw, Fa;

assign Fa = A & B;

always @(\*) begin

C = 0;

case(OP)

4'b0000: begin F = Fa; end

4'b0001: begin F = A|B; end

4'b0010: begin F = A^B; end

4'b0011: begin F = ~(A|B); end

//4'b0100: begin {C,F} = A+B; end

4'b0101: begin {C,F} = A-B; end

4'b0110: begin F = A<B; end

4'b0111: begin F = B<<A; end

default: F = Fw;

endcase

ZF = F == 0;

//CF = C;

OF = A[SIZE]^B[SIZE]^F[SIZE]^C;

SF = F[SIZE];

PF = ~^F;

end

Decoder38 decoder38\_1(OP[2:0], EN);

addr\_32bit add(Fw, CF, A, B, 0, ~EN[4]);

endmodule

module sim\_ALU();

parameter SIZE = 32;

reg [SIZE:1] A, B;

reg [3:0] OP;

wire [SIZE:1] F;

wire ZF, CF, OF, SF, PF;

initial begin

OP = 0;

A = 10;

B = 143;

fork

repeat(7) #10 OP = OP + 1;

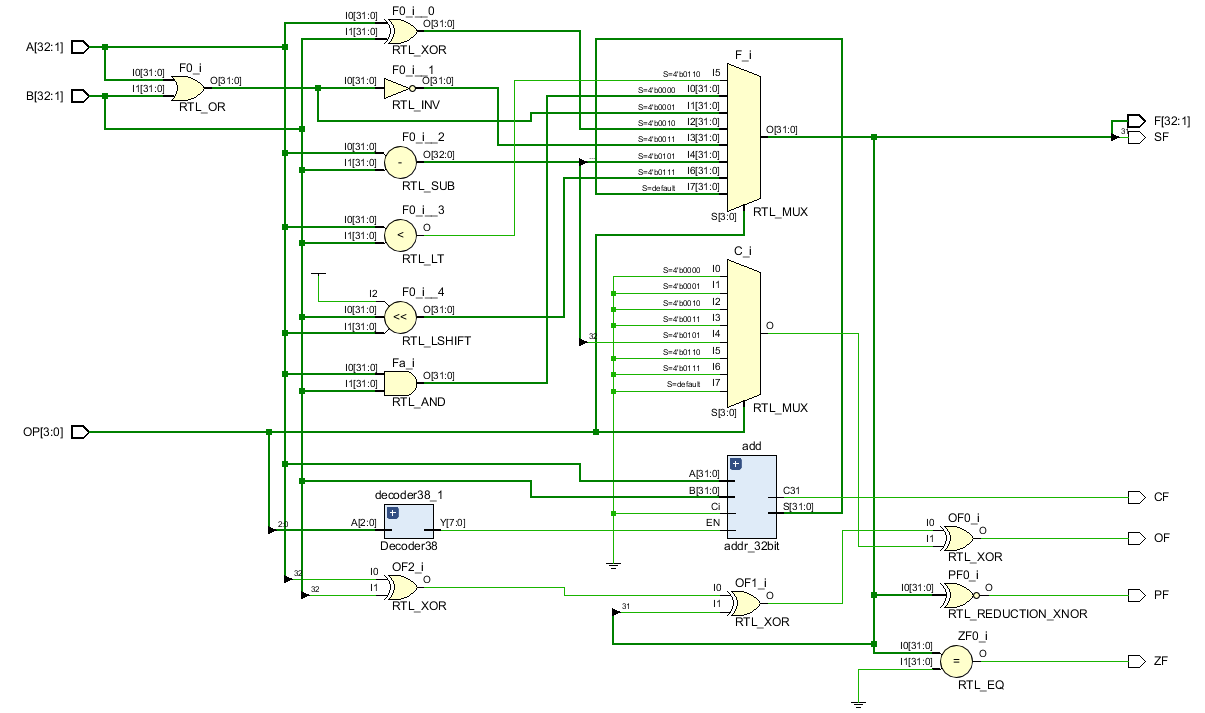
join

end

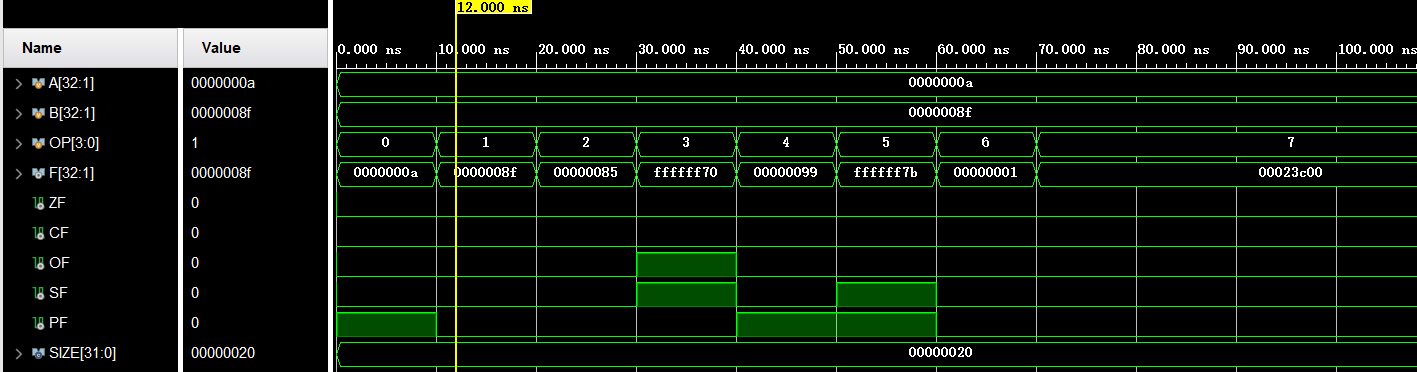
ALU test(OP, A, B, F, ZF, CF, OF, SF, PF);

endmodule

电路图：



仿真波形：



分析：输入A=0000000a，B=0000008f。

当OP=0时，按位与输出F=0000000a，由于F中1的个数为偶数个，故PF=1；

当OP=1时，按位或输出F=0000008f；

当OP=2时，按位异或输出F=00000085；

当OP=3时，按位或非输出F=ffffff70，由于F最高位为1，故SF=1，这里OF=1无意义；

当OP=4时，求和输出F=00000099，没有进位和溢出，故CF=0，OF=0；

当OP=5时，相减输出F=ffffff7b，输出为负数，故SF=1；

当OP=6时，比较输出F=00000001，说明A<B；

当OP=0时，移位输出F=00023c00，B左移A位。

由于所有输出均不为0，故始终有ZF=0。

**五、调试和心得体会**

Logisim踩坑：Logisim中默认的多输入异或门行为是当一个输入为真时输出为真，在计算溢出OF与奇偶个数PF时会出错，需要将其属性改为当奇数个输入为真时输出为真。

纠错：PPT第7页最后一个注释：F有奇数个1，PF=1（根据表达式，这里应是PF=0）。

通过本次实验，我进一步掌握了Verilog三种描述方式的使用，并学会了如何设计一个ALU算术逻辑单元。同时，我也体会到了使用Logisim设计复杂电路的困难，不仅需要合理布局电路，还要选择适当的器件，相比之下Vivado就比较方便灵活。