实验三 算术逻辑单元设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握算术逻辑单元的设计和测试方法。

**二、实验内容**

1 运算模块的设计与测试

2 算术逻辑单元设计与测试

**三、实验要求**

1 掌握Vivado或Logisim开发工具的使用，掌握以上电路的设计和测试方法；

2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

1.一位全加器

设计文件：

module add1(input A, B, cin,

output cout, F

);

assign {cout,F}=A+B+cin;

endmodule

仿真文件：

module sim;

reg A, B, cin;

wire cout, F;

add1 a(A,B,cin,cout,F);

initial begin

{A,B,cin}=0;

end

always@(\*)

fork

forever #1 A=A+1;

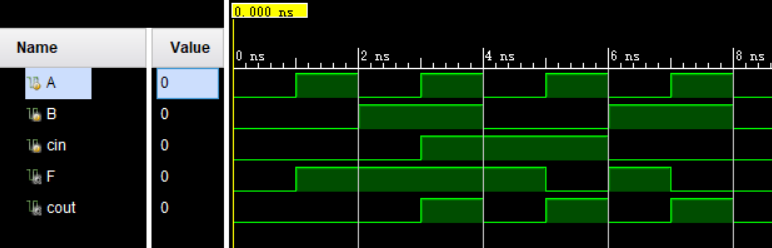
forever #2 B=B+1;

forever #3 cin=cin+1;

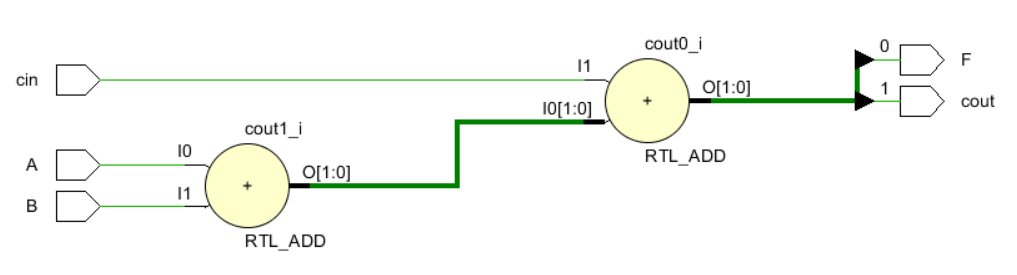
join

endmodule

仿真图形：



电路图：



分析：

在设计一位全加器的过程中，直接使用数据流的形式来描述cout和F，assign {cout,F}=A+B+cin。在仿真图形中，在0到1ns的时候,A,B,cin全部为0，对应的输出F和cout也全为0；在1到3ns的时候，A,B,cin之中只有一个为1，其余两个为0，对应的输出F为1，进位cout为0；在5到6ns的收，A,B,cin之中只有两个为1，对应的输出F为0，进位cout为1；在3到4ns的时候，A,B,cin全部为1，对应的输出F和进位cout也都为1。输出结果符合预期。

2.32位全加器

设计文件：

module add32(input[32:1]A,

input[32:1]B,

output C,

output reg [32:1]F,

input en

);

integer i;

reg [33:1]CI;

assign C=CI[33];

initial begin

CI=0;

end

always@(\*) begin

if(en)

for(i=1;i<33;i=i+1)

{CI[i+1],F[i]}=A[i]+B[i]+CI[i];

else F<=32'bz;

end

endmodule

仿真文件：

module sim;

reg [32:1]A;

reg[32:1]B;

reg en;

wire [32:1]F;

wire C;

add32 a(A,B,C,F,en);

initial begin

{A,B,en}=0;

end

always@(\*)

fork

forever #1 A=A+1;

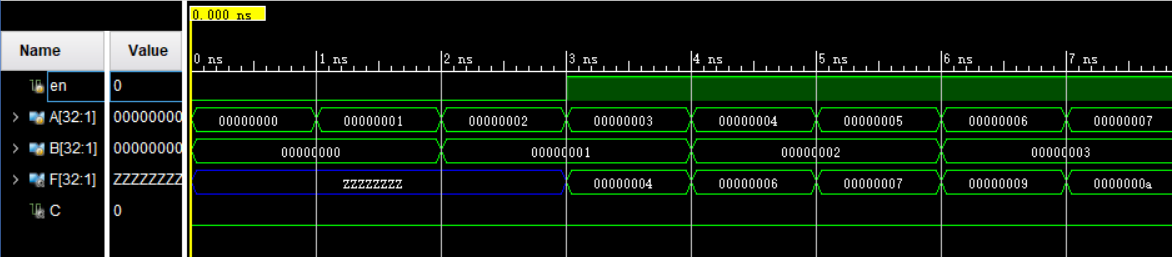
forever #2 B=B+1;

repeat(1) #3 en=en+1;

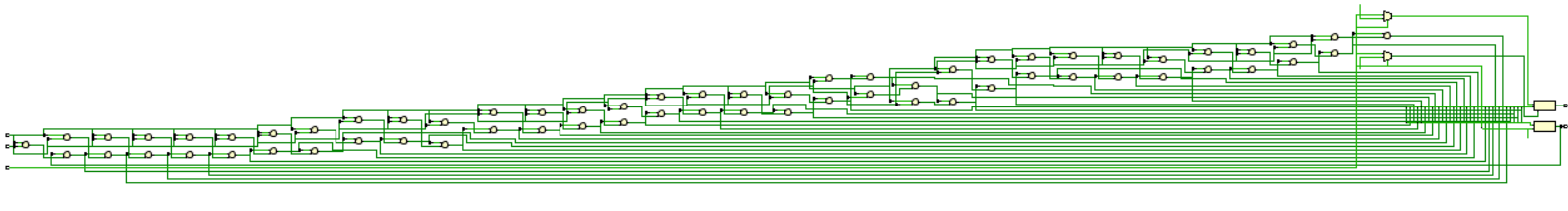
join

endmodule

仿真图形：



电路图：



分析：

在实现32位全加器的过程中，采用了for循环，i从一开始，首先对en进行判断，如果en无效，那么直接将F置为高阻态；如果en有效，那么对A和B的每一位进行相加得到输出F对应未知的值以及进位CI [i],然后i执行+1操作，对应的A[i],B[i]和CI [i]进行相加得到{ CI [i+1],F[i]}的值，如此循环直到i等于33，此时退出for循环，令进位C等于CI [33]。

在仿真图形中，在0到3ns的过程中en始终为0无效，所以对应的F为高阻态，C为0；在3ns时候en变为1有效，此时F等于A+B进行相加的结果4，之后en一直有效，F始终等于A+B的和。符合预期。

3.算术逻辑单元

设计文件：

module ALU(OP,A,B,F,CF );

parameter size=32;

input [3:0]OP;

input[size:1]A;

input[size:1]B;

output reg [size:1]F;

output CF;

wire [size:1]FW;

always@(\*)

begin

case(OP)

4'b0000:F<=A&B;

4'b0001:F<=A|B;

4'b0010:F<=A^B;

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

default:F<=FW;

endcase

end

add32 ma(A,B,CF,FW,(OP==4));

sub32 am(A,B,CF,FW,(OP==5));

slt32 an(A,B,FW,(OP==6));

sll32 ac(A,B,FW,(OP==7));

endmodule

sub32模块：

module sub32(input[32:1]A,

input[32:1]B,

output reg C,

output reg [32:1]F,

input en

);

always@(A,B,en)

if(en) {C,F}<=A-B;

else F<=32'bz;

endmodule

slt32模块：

module slt32(input[32:1]A,

input[32:1]B,

output reg [32:1]F,

input en

);

always@(A,B,en)

if(en) F<=A<B;

else F<=32'bz;

endmodule

sll32模块：

module sll32(input[32:1]A,

input[32:1]B,

output reg [32:1]F,

input en

);

always@(A,B,en)

if(en) F<=B<<A;

else F<=32'bz;

endmodule

仿真文件：

module sim;

reg [3:0]OP;

reg [31:0]A,B;

wire ZF,CF,OF,SF,PF;

wire [31:0]F;

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

initial begin

{OP,A,B}=0;

end

always fork

forever #1 A=A+1;

forever #2 B=B+1;

forever

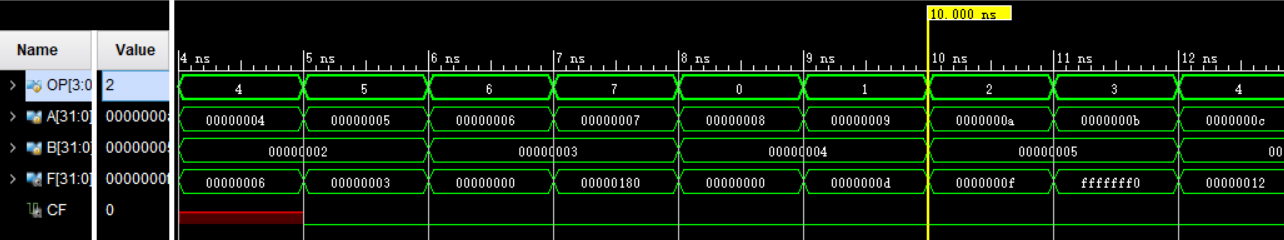
#1 if(OP==7)

OP<=0;

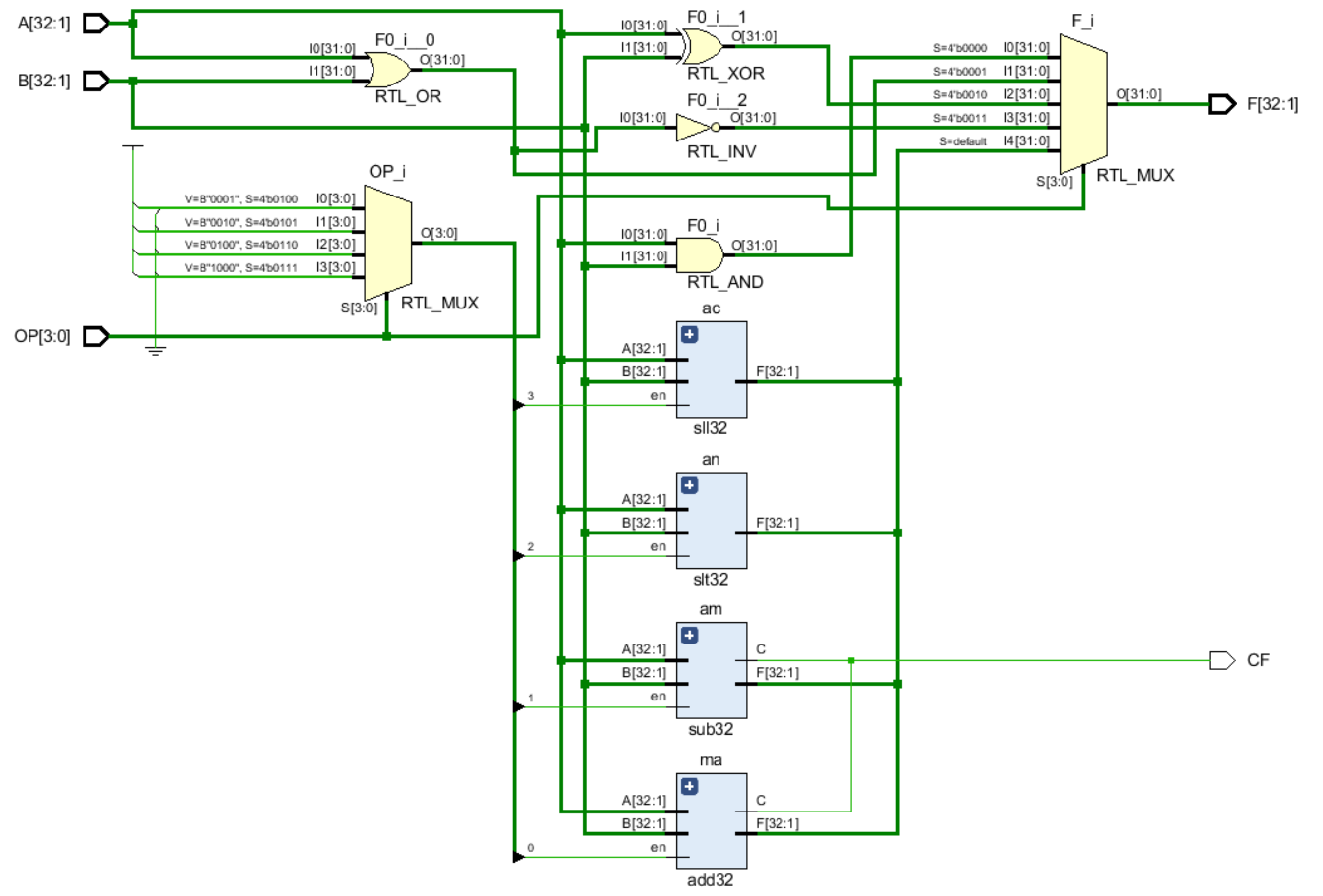
else OP<=OP+1;

join

endmodule



电路图：



分析：

在实现算术逻辑单元的过程中，首先采用always语句，在语句内是一个对OP进行判断的case语句，如果OP的值为0，那么F为A和B进行与操作的结果；如果OP的值为1，那么F为A和B进行或操作的结果；如果OP的值为2，那么F为A和B进行异或操作的结果；如果OP的值为3，那么F为A和B进行或非操作的结果；如果以上都不是，那么F的值等于wire类型的数组变量FW。随后在always语句外分别调用了add32,sub32,slt32,sll32四个模块，模块的使能参数分别为（OP==4）, （OP==5）, （OP==6）, （OP==7）,在四个模块内部，如果使能信号为0无效的话，那么FW会被置为高阻态z，避免总线上的竞争。只有在使能信号有效的情况下各模块的其他参数才能有效。

在仿真图形中，在4ns时，OP为4，F为A（4）和B（2）相加的结果6；在5ns时，OP为5，F为A（5）和B（2）相减的结果3；在6ns时，OP为6，A（6）大于B（3），所以对应表示A小于B的F为0（表示A小于B不成立）；在7ns时，OP为7，F为B（3）左移A（7）位的结果180；在8ns时，OP为0，F为A（1000）和B（0100）相与的结果0；在9ns时，OP为1，F为A（1001）和B（0100）相或的结果D（1101）；在10ns时，OP为2，F为A(1010)和B(0101)异或的结果f(1111)；在11ns时，OP为3，F为A(0000000b)和B(00000005)或非的结果fffffff0；

**五、调试和心得体会**

深入理解加法器和ALU的原理：设计全加器模块是深入理解二进制加法的第一步。这包括理解进位、异或和与门的作用。同时，设计ALU模块需要理解如何执行多种算术和逻辑操作，包括加法、减法、与、或、异或等。将整个ALU分解为多个模块，如加法器、减法器、逻辑运算单元等，有助于提高设计的可维护性和可扩展性。模块化设计使得每个模块都可以单独测试和优化。设计一个32位的ALU需要特别关注时序和性能问题。时钟信号的管理、数据传输速度和操作的延迟都需要考虑，以确保ALU在正常工作时能够满足性能要求。在设计和实现这些模块时，测试和调试是至关重要的步骤。使用仿真工具和模拟器来验证模块的正确性，以及确保它们按照设计规范执行操作。同时，学会分析和解决电路中的问题也是重要的技能。ALU是计算机中的一个核心组件，用于执行各种算术和逻辑操作。通过设计和测试ALU模块，我更好地理解了它在CPU中的作用，以及如何为不同的计算任务提供基本支持。总的来说，我提高了对数字电路和计算机体系结构的理解，同时也锻炼了解决复杂问题的能力。这将为未来的学习和职业发展提供坚实的基础。