《计算机组成原理》实验报告

年级、专业、班级	2021级计算机科学与技术03班、05班	姓名	任俊璇、张梓健
实验题目	实验一简单流水线与运算器实验		
实验时间	2023 年 3 月 27 日	实验地点	DS1410
实验成绩	优秀/良好/中等	实验性质	□验证性 ☑设计性 □综合性

教师评价：

☑算法/实验过程正确； ☑源程序/实验内容提交； ☑程序结构/实验步骤合理;

☑实验结果正确; ☑语法、语义正确; ☑报告规范;

其他:

评价教师: 冯永

实验目的

（1）理解流水线 (Pipeline) 设计原理；

（2）了解算术逻辑单元 ALU 的原理；

（3）熟悉并运用 Verilog 语言设计 ALU；

（4）熟悉并运用 Verilog 语言设计流水线全加器；

报告完成时间: 2023年 4月 11日

1 实验内容

1.1 ALU设计实验

实验要求实现以下算术运算功能，其对应的控制码及功能如下：

$F_{2:0}$	功能	$F_{2:0}$	功能
000	A + B(Unsigned)	100	$\overline{A}$
001	A - B	***101***	***SLT***
010	A AND B	110	未使用
011	A OR B	111	未使用

表 1: 算数运算控制码及功能

实验要求：

1. 根据ALU原理图，使用Verilog语言定义ALU模块，其中输入输出端口参考实验原理，运算指令码长度为[2:0]。
2. 仿真时B端口输入为32h'01, A端口输入参照 4.1 中表格
3. 实现SLT功能。
4. 验证表1中所有功能。
5. 给出RTL源程序（.v文件）

1.2 流水线实验

本次实验为仿真实验，设计完成后仅需进行**行为仿真**。

实验要求：

1. 实现4级流水线8bit全加器，需带有流水线暂停和刷新；
2. 模拟流水线暂停，仿真时控制10周期后暂停流水线2周期（第2级），流水线恢复流动；
3. 模拟流水线刷新，仿真时控制15周期时流水线刷新（第3级）。

2 实验设计

2.1 ALU

2.1.1 功能描述

通过输入对应的控制码，对两个操作数进行无符号加法、无符号减法、按位与、按位或、操作数1取非、SLT等操作。

表 2: ALU模块

模块名	功能	接口
|—top.v	顶层文件,将各模块连接。	input:clk(时 钟),reset(重 置),num1(操 作 数1),op(控 制 信号);output:seg(七段数码管),ans(驱动七段数码管显示)
|——alu.v	ALU 模块。	input:num1,op;output:result(运算结果)
|——display.v	七段数码管显示模块文件。	input:clk,reset,s(运算结果);output:seg,ans
|———seg7.v	七段数码管显示模块文件。	input:din(数码管显示数字);output:dout(seg)

2.1.2 接口定义

表 3: 接口定义表

信号名	方向	位宽	功能描述
clk	input	1-bit	系统时钟信号
reset	input	1-bit	重置信号
ins	input	8-bit	操作数1，经过无符号扩展至 32 位
op	input	3-bit	运算器的控制信号
result	output	32-bit	运算结果
seg	output	7-bit	七段数码管
ans	output	8-bit	驱动七段数码管显示

2.1.3 逻辑控制

always 块监视输入的操作码 op，使用case语句根据不同的操作码执行不同的操作，并将操作结果存储在输出寄存器 result 中。num2 被设置为 1，extend 通过将 num1 的低 8 位前面填充了 24 个 0 进行扩展。根据操作码，执行加、减、按位与、按位或、按位取反、比较操作等。如果操作码不是上述任何一个，则将结果设置为 0。

2.2 有阻塞4级8bit全加器

2.2.1 功能描述

1. 实现 4 级流水线 32bit 全加器，每一级进行 8bit 加法运算，需带有流水线暂停和刷新；

2. 模拟流水线暂停，仿真时控制 10 周期后暂停流水线 2 周期（第 2 级），流水线恢复流动；

3. 模拟流水线刷新，仿真时控制 15 周期时流水线刷新（第 3 级）。

2.2.2 接口定义

表 4: 接口定义表

信号名	方向	位宽	功能描述
cin_a	input	32-bit	第一位加数
cin_b	input	32-bit	第二位加数
clk	input	1-bit	系统时钟周期
stop	input	1-bit	流水线暂停信号
reset	input	1-bit	流水线刷新信号
cin	input	1-bit	进位信号
cout	output	1-bit	溢出信号
sum	output	32-bit	运算结果

2.2.3 逻辑控制

根据要求将流水线全加器分为4级，因为是32位全加器所以每一级进行8位的加法，将两个加数对应8位值相加得到的值存储起来，将进位保留用于下一级的加法中。每一级由系统时钟上升沿控制，首先判断reset信号和stop信号是否有效，如果reset有效则将存储的加法结果和进位都置为0，如果stop有效，则加法结果和进位值保持不变，如果reset和stop都无效则正常进行运算。最后将四级的加法结果进行拼接得到最终的运算结果。

3 实验过程记录

记录实验的过程，完成了什么样的工作，存在的问题包括哪些，解决方案如何等。subsubsection名称自行设定。

3.1 ALU 设计实验

3.1.1 实验过程

设计ALU主模块：根据实验要求，设计ALU模块，以实现加、减、按位与、按位或、按位取反、比较操作功能。

ALU仿真测试：设计仿真文件，对ALU模块功能进行仿真测试。

导入display、seg7模块：将使用提供的display和seg7添加到项目中，并根据实际情况对模块进行修改。

设计top模块：设计top模块，将所有模块连接。

导入约束文件：将使用提供的约束文件添加到项目中，并根据实际情况进行修改。

下板验证：生成bit流文件，进行下板验证。

3.1.2 出现的问题

问题描述：直接添加提供的约束文件后，生成bit流文件出错

解决方案：约束文件中对于操作数1接口的命名与模块中不一致，所以对顶层文件中操作数1接口名进行修改；约束文件中缺少操作码op，所以添加op的引脚分配。

3.2 流水线实验

3.2.1 实验过程

设计32位流水线全加器模块：根据实验要求，设计4级流水线的32位全加器，每一级进行8位加法运算，并且带有流水线暂停和刷新功能。

设计仿真模块：编写仿真文件，模拟流水线暂停，仿真时控制 10 周期后暂停流水线 2 周期，然后流水线恢复流动；模拟流水线刷新，仿真时控制 15 周期时流水线刷新。

运行仿真验证功能：运行仿真文件，得到结果，与实验要求进行比对，验证功能。

3.2.2 出现的问题

问题描述：直接拼接4级的加法结果所得到的最终运算结果有误。

解决方案：将每一级的加法结果一个个拼接到最终运算结果的相应位上，最终验证结果正确。

4 实验结果及分析

4.1 ALU验证实验结果

操作	Num1	Result
A + B(Unsigned)	8’b00000010	32’h00000003
A - B	8’b11111111	32’h000000FE
A AND B	8’b11111110	32’h00000000
A OR B	8’b10101010	32’h000000AB
$\overline{A}$	8’b11110000	32’hFFFFFF0F
SLT	8’b10000001	32’h00000000

表 5: ALU结果表

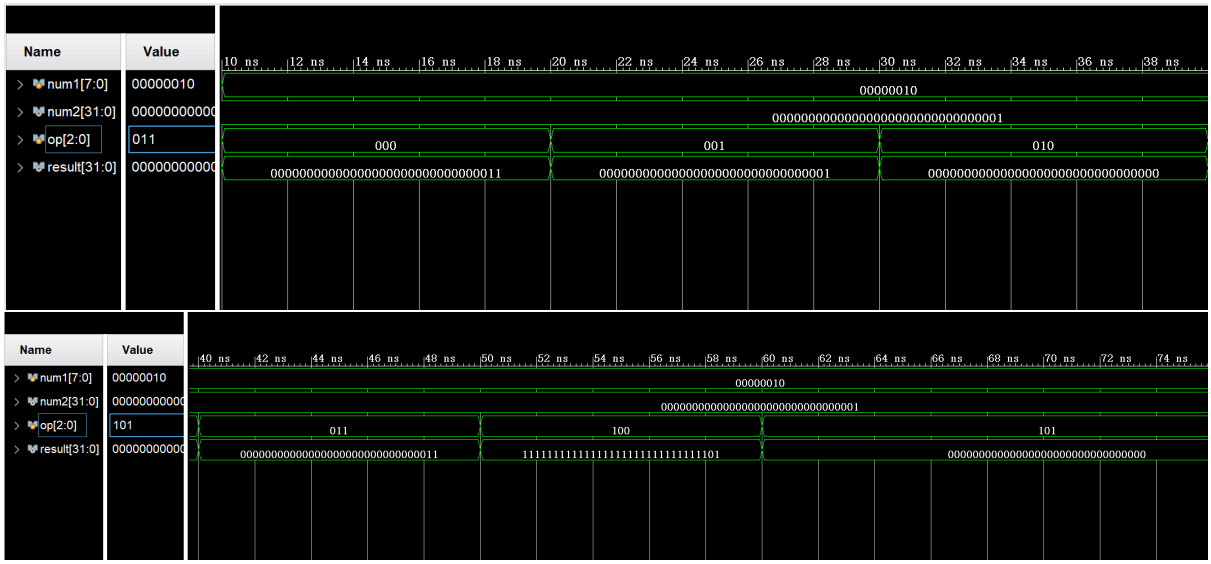


图 1: ALU仿真图

结果分析：由ALU仿真图可以看到，A=2，B=1，根据op值的不同，ALU执行不同的功能，result也随之变化。

当op=000时，result=3；

当op=001时，result=1；

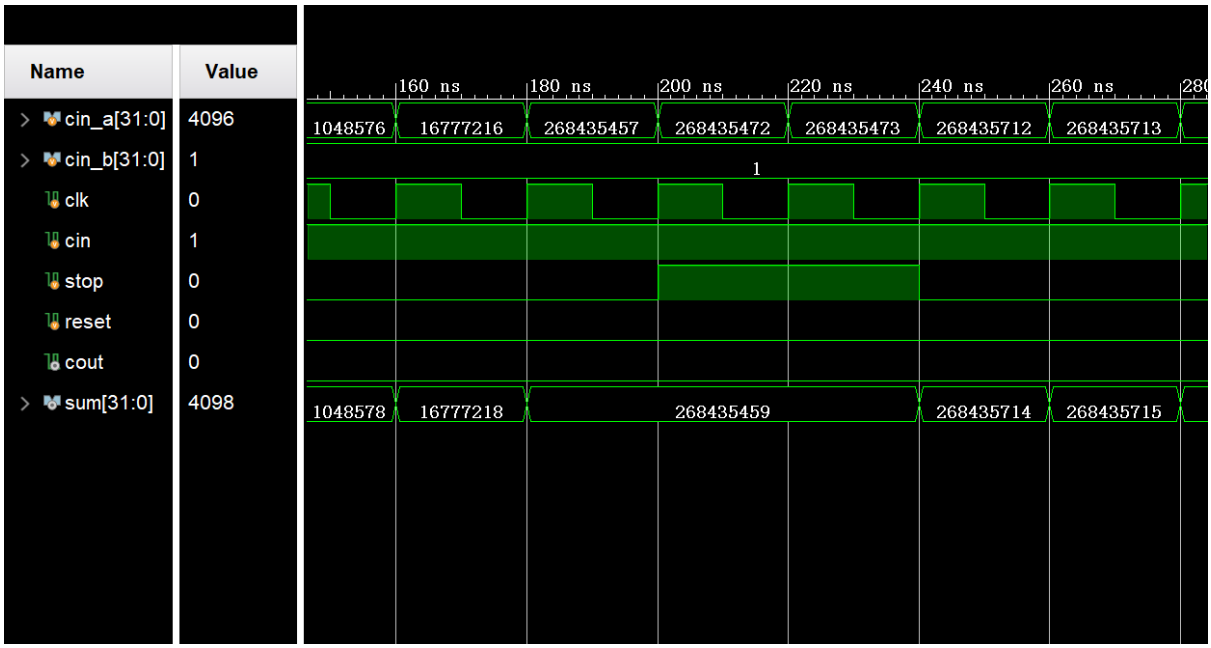
当op=010时，result=00000000000000000000000000000000；

当op=011时，result=00000000000000000000000000000011；

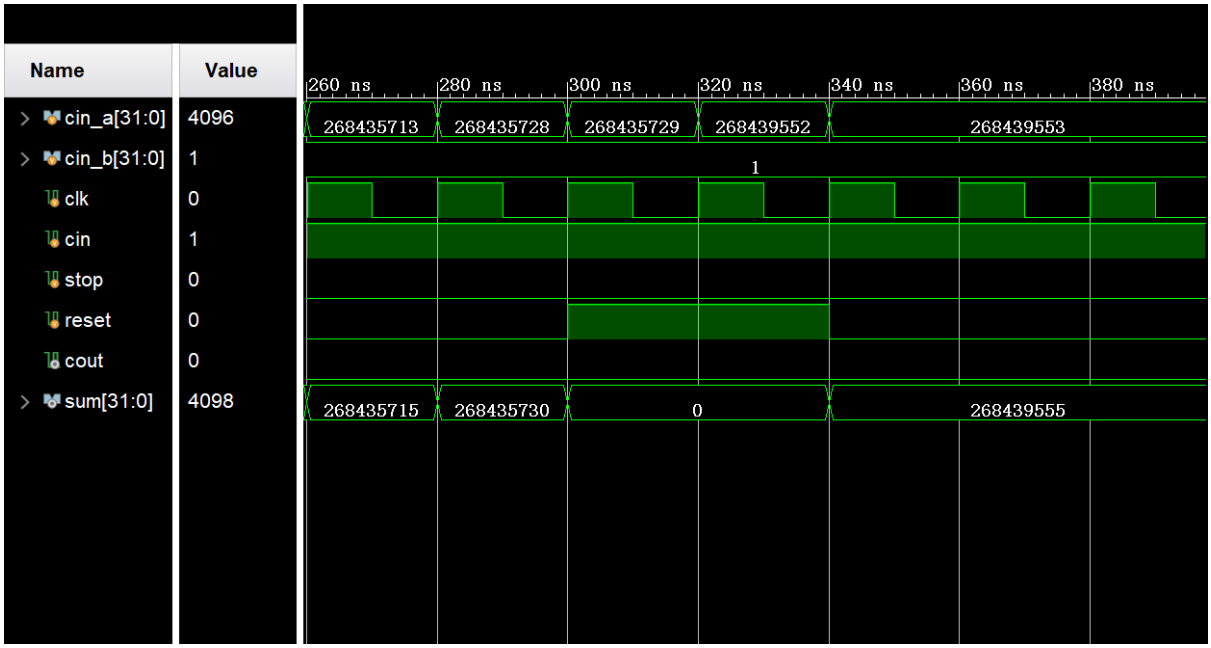
当op=100时，result=11111111111111111111111111111101；

当op=101时，result=00000000000000000000000000000000。

4.2 流水线阻塞（暂停）仿真图



4.3 流水线刷新（清空）仿真图



结果分析：由流水线阻塞（暂停）仿真图可以看到，当stop信号有效时，全加器暂停了2周期，sum的值增加了2周期的延迟；由流水线刷新（清空）仿真图可以看到，当reset信号有效时，sum的值被重置为0，之前暂停的延迟也被刷新了。

A ALU代码

```
module alu(
input [7:0] num1,
input[2:0]op,
output reg [31:0] result
    );
    reg [31:0] num2;
    reg [31:0] extend;
    always@(op) begin
        num2=32'h01;
        extend={24'h00,num1[7:0]};
            case(op)
                3'b000 : result = extend + num2 ;
                3'b001 : result = extend - num2 ;
                3'b010 : result = extend & num2 ;
                3'b011 : result = extend | num2 ;
                3'b100 : result = ~ extend ;
                3'b101 : result= (extend<num2) ? {1}:{0};
             default : result = 32 'h00000000 ;
            endcase
    end
endmodule
```

B 32bit流水线全加器代码

```
module adder_32bits(cin_a,cin_b,clk,stop,reset,cin,cout,sum);
    input [31:0] cin_a;
    input [31:0] cin_b;
    input clk;
    input stop;
    input reset;
    input cin;
    output wire cout;
    output wire [31:0] sum;
    reg cout1,cout2,cout3,cout4;
    reg [7:0] sum1,sum2,sum3,sum4;

    //first level
    always@(posedge clk) begin
        if(reset) begin
            cout1<=0;
            sum1<=0;
        end
        else if(stop) begin
            cout1<=cout1;
            sum1<=sum1;
```

```
    end
    else begin
        {cout1,sum1} = cin + cin_a[7:0] + cin_b[7:0];
    end
end

//secend level
always@(posedge clk) begin
    if(reset) begin
        cout2<=0;
        sum2<=0;
    end
    else if(stop) begin
        cout2<=cout2;
        sum2<=sum2;
    end
    else begin
        {cout2,sum2} = cout1 + cin_a[15:8] + cin_b[15:8];
    end
end

//third level
always@(posedge clk) begin
    if(reset) begin
        cout3<=0;
        sum3<=0;
    end
    else if(stop) begin
        cout3<=cout3;
        sum3<=sum3;
    end
    else begin
        {cout3,sum3} = cout2 + cin_a[23:16] + cin_b[23:16];
    end
end

//forth level
always@(posedge clk) begin
    if(reset) begin
        cout4<=0;
        sum4<=0;
    end
    else if(stop) begin
        cout4<=cout4;
        sum4<=sum4;
    end
    else begin
        {cout4,sum4} = cout3 + cin_a[31:24] + cin_b[31:24];
    end
```

```
    end

    assign sum[7:0] = sum1;
    assign sum[15:8] = sum2;
    assign sum[23:16] = sum3;
    assign sum[31:24] = sum4;
    assign cout=cout4;

endmodule
```