# 《计算机组成原理》实验报告

| | | | |
|---|---|---|---|
| **年级、专业、班级** | 2023 级<br>2023 级<br>2023 级 | **姓名** | |
| **实验题目** | 实验一简单流水线与运算器实验 | | |
| **实验时间** | 2025 年 4 月 13 日 | **实验地点** | DS1410 |
| **实验成绩** | 优秀/良好/中等 | **实验性质** | □验证性<br>☑设计性<br>□综合性 |
| **教师评价：**<br>☑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;<br>☑实验结果正确; ☑语法、语义正确; ☑报告规范;<br>其他:<br>评价教师: 任骜 | | | |
| **实验目的**<br>（1）理解流水线 (Pipeline) 设计原理；<br>（2）了解算术逻辑单元 ALU 的原理；<br>（3）熟悉并运用 Verilog 语言设计 ALU；<br>（4）熟悉并运用 Verilog 语言设计流水线全加器； | | | |

**表 1:** 实验报告封面

报告完成时间: 2025 年 4 月 17 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 2:** 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2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

该 ALU 实现加法、减法、按位与、按位或、取反及无符号小于判断等基础运算功能，支持 8 位与 32 位输入，输出 32 位结果，并在数码管上进行显示。

### 2.1.2 接口定义

**表 3:** 接口定义

| 信号名 | 方向 | 位宽 | 功能描述 |
|---|---|---|---|
| num1 | Input | 8-bit | 操作数 1 |
| num2 | Input | 32-bits | 操作数 2 |
| op | Output | 3-bits | 操作码 |
| result | Output | 32-bits | 计算结果 |

### 2.1.3 逻辑控制

本 ALU 模块采用组合逻辑控制方式，不使用时序触发的有限状态机（FSM）。由于 ALU 运算为即时反应型逻辑（无时序依赖），其控制流程可简化为单周期组合逻辑响应，无需状态跳转。模块通过将 8 位输入零扩展为 32 位，并根据 3 位操作码执行加、减、与、或、取反和小于判断等运算，非法操作码输出全 x。ALU 的控制逻辑使用组合逻辑电路实现，具体如下：

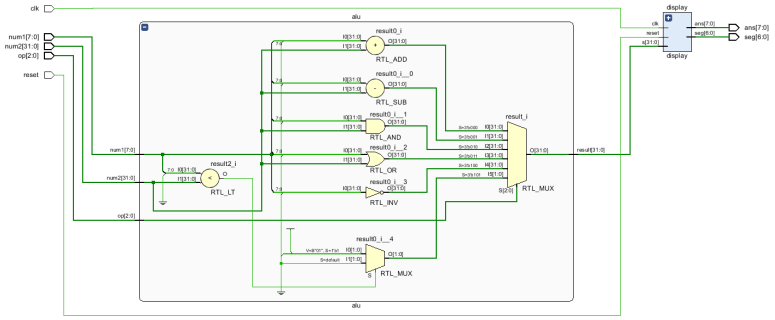


**图 1:** ALU 控制逻辑电路

## 2.2 有阻塞 4 级 8bit 全加器

### 2.2.1 功能描述

### 2.2.2 接口定义

### 2.2.3 逻辑控制

# 3 实验过程记录

## 3.1 问题 1:ALU 计算错误

**问题描述：**误以为运算会自动零扩展，未对 num1 作位数扩展，与 num2 相加时，结果错误。

**解决方案：**对 num1 进行位数扩展，使用 Verilog 的零扩展语句，将 8 位数扩展为 32 位：assign num1_ext = {24'b0, num1};

## 3.2 问题 2:ALU 实现各指令码所对应的功能

**问题描述：**实现过程中，需要根据操作码（op）执行不同的运算逻辑，确保 ALU 能正确完成加法、减法、位运算和比较等功能。

**解决方案：**使用 Verilog 中的 case 语句对操作码进行译码，根据不同的值执行相应的算术或逻辑运算，例如加法、减法、按位与、按位或、取反和无符号小于比较。通过组合逻辑块 always @(*) 实现功能控制。

## 3.3 问题 3: 验证 ALU 各功能正确性

**问题描述：**需要验证 ALU 模块是否按照设计正确执行加法、减法、与、或、取反、小于判断等功能。

**解决方案：**编写 Testbench 仿真文件，采用 initial 块在每个时钟上升沿提供输入数据，包括 8 位 num1、32 位 num2 和 3 位操作码 op，通过 $display 打印输出结果，在控制台查看各功能的执行情况并与预期对比验证正确性。

# 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 4:** ALU 结果表

## 4.2 ALU 仿真

## 4.3 流水线阻塞(暂停)仿真图


**图 2:** 占位图

## 4.4 流水线刷新(清空)仿真图

# A ALU 代码

```
‘timescale 1ns / 1ps

module alu(
    input wire [2:0] op,            // 操作码
    input wire [7:0] num1,          // 8位输入
    input wire [31:0] num2,         // 32位输入
    output reg [31:0] result        // 运算结果
);

    wire [31:0] num1_ext;
    assign num1_ext = {24'b0, num1};  // 8位零扩展到32位

    always @(*) begin
        case (op)
            3'b000: result = num1_ext + num2;
            3'b001: result = num1_ext - num2;
            3'b010: result = num1_ext & num2;
            3'b011: result = num1_ext | num2;
            3'b100: result = ~num1_ext;
            3'b101: result = (num1_ext < num2) ? 32'b1 : 32'b0;
            default: result = 32'hxxxxxxxx;
        endcase
    end
endmodule
```

# B 32bit 流水线全加器代码

仅需要 stallable_pipeline_adder.v，填充至 lstlisting 中 (红字为内容说明，请删除)