实验一：存储器与运算器实现

1 实验目的

i. 了解随机存取存储器 RAM 的原理

ii. 了解算术逻辑单元 ALU 的原理

iii. 熟悉 Verilog 语言，并使用 Verilog 设计简单的模块

2 实验环境（推荐）

i. IDE: vscode

ii. verilog compiler: iverilog

iii. waveform viewer: GTKwave

安装教程参考:

i. 全平台轻量开源 verilog 仿真工具 iverilog+GTKWave 使用教程

ii. 用 VS Code + iverilog + GTKwave 仿真 Verilog

iii. 用 Vscode 代替 Vivado 原生编辑器

3 实验任务

本实验要求编写一个支持 RISC-V ISA 中所有整数算术运算的 ALU，并与一个单端口输出的 Data RAM 连接。

3.1 存储器实验——RAM 的实现

3.1.1 实验描述

图1给出了一个简单的随机存储器，它的存储部分使用由 2^{16} 个 32 位寄存器构成的数组实现。你需要根据图1，使用 Verilog 语言实现 RAM 模块。

3.1.2 实验要求

1. "clk" 端口为 1 位的时钟接口，我们规定寄存器在时钟的上升沿接收数据的变化。
2. "address" 端口为 32 位的地址接口，读写操作共用。
3. "write_en" 信号为 1 位的读写使能信号接口，使能（高电平）代表写，不使能代表读。
4. "write_data" 为 32 位的写数据输入接口，"read_data" 为 32 位的读数据输出接口。

5. ”write_type” 端口用来区分写内存的长度（Byte、Half-Word、Word）。我们规定当写入长度不足 32 位时，使用”write_data” 的低位传输数据。此外，我们在测试数据中保证传入的地址与传入的字节数对齐（举例来说，如果传 Half-word，地址要是 2（字节）的倍数）。这个信号一共有 3 位（可以想想，为什么只需要表示 0,1,2，却保留了 3 位），我们分别用 “00”，“01”，“10” 指代 Byte，Half-word，Word 的数据传输。

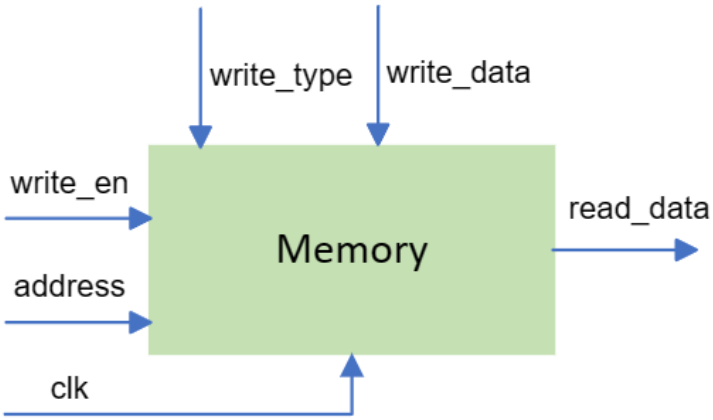


图 1: RAM 模块示意图

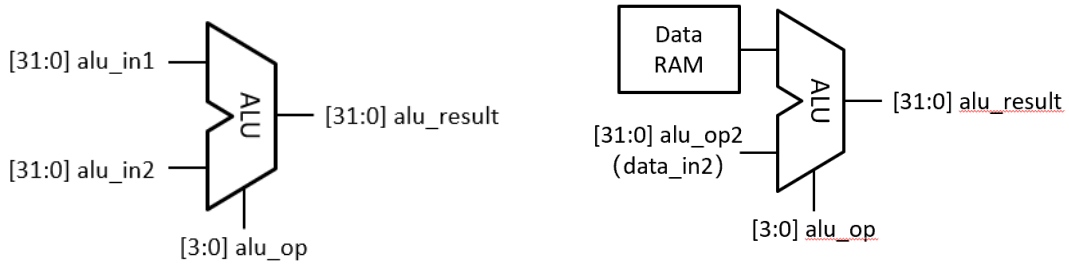
3.2 运算器实验——ALU 的实现

3.2.1 实验描述

图2(a)给出了一个具有两个 32b 输入和 32b 输出的 ALU，其运算符的选择由 4b 信号”alu_op” 决定，具体对应关系如表1所示。

3.2.2 实验要求

1. 根据 ALU 原理图2(a)，使用 Verilog 语言定义 ALU 模块。
2. ALU 读入两个操作数，根据信号”alu_op” 选择具体运算，计算结果并输出。



(a) pic1.ALU (b) pic2.ALU&RAM

图 2: ALU 模块示意图及模块连接规则

运算	**alu_op**	**运算**	**alu_op**
ADD	4'b0000	SRL	4'b0101
SLL	4'b0001	OR	4'b0110
SLT	4'b0010	AND	4'b0111
SLTU	4'b0011	SUB	4'b1000
XOR	4'b0100	SRA	4'b1101

表 1: ALU-op

3. ALU 要求支持表格内的十种运算，各种运算的功能如表2所示（大家可以直接使用 Verilog 内置的运算符实现对应的运算）。

指令	功能
ADD rd, rs1, rs2;	x[rd] = x[rs1] + x[rs2]
SLL rd, rs1, rs2;	x[rd] = x[rs1] << [rs2]
SLT rd, rs1, rs2;	x[rd] = x[rs1] < x[rs2] (Signed)
SLTU rd, rs1, rs2;	x[rd] = x[rs1] < x[rs2] (Unsigned)
XOR rd, rs1, rs2;	x[rd] = x[rs1] $\oplus$ x[rs2] (按位异或)
SRL rd, rs1, rs2;	x[rd] = x[rs1] >> x[rs2] (逻辑右移)
OR rd, rs1, rs2;	x[rd] = x[rs1] | x[rs2] (按位或)
AND rd, rs1, rs2;	x[rd] = x[rs1] & x[rs2] (按位与)
SUB rd, rs1, rs2;	x[rd] = x[rs1] - x[rs2]
SRA rd, rs1, rs2;	x[rd] = x[rs1] >> x[rs2] (算术右移)

表 2: ALU ops and function

3.3 模块组织

将存储器和运算器连接起来。

|–top.v	设计顶层文件，参照图2(b)将各模块连接。
|——alu.v	ALU 模块
|——ram.v	RAM 模块

表 3: module organization

4 测试与评分

请在本地调试完毕后，再将代码提交到测试平台，并查看测试平台中显示的分数。

4.1 平台使用

1. 课程平台地址：微处理器设计与智能芯片课程实践

2. 请按规定字段进行注册，并牢记自己的密码。
3. 从测试平台上下载的代码包包括模板文件和简化的测试文件，请先仔细阅读 README，其中提到了你至少应该上传哪些文件，可以自行定义其他辅助模块一并压缩上传。
4. 测试平台会用自己的测试用例覆盖提供给同学们的用例，可以不再上传与测试相关的文件。
5. 请珍惜实验室服务器资源，切勿恶意提交代码。
6. 与平台、实验有关的任何问题可以与助教联系。
7. 欢迎大家提出有关平台、实验的各种建议，让本课程和配套实习越来越好。

4.2 评分细则

1. 能通过编译不报错，获得一个绿色的PASS和 1 分。这主要是为了帮大家明确自己正确提交了文件并通过了编译，所以请不要获得一个 PASS 之后就结束，务必根据 performance 分数确认自己的完成度。
2. 平台会进行一系列的测试输入与输出，每一段输出对应 Lab 的一个功能。比如，本次 lab 会分别验证十种运算，每有一个运算验证正确获得 10 分，满分 100 分。

注：平台评分主要作为对你完成度的提示，并不等于你最终课程实习部分的分数。

4.3 实验报告

在实验的同时和实验之后撰写实验报告是非常良好的习惯。这不仅有助于提高实验效率，理清实验思路，帮助 debug，也有助于记录实验过程和结果，还可以和其他同学交流和分享。因此，我们需要大家在 lab 完成过后为我们提交一个实验报告，实验报告有以下几点注意事项：

1. 包含你的设计思路，可以包括相应的模块组织层次、一些不同于助教提示的地方等。
2. 你 debug 的过程，或崩溃或大起大落或柳暗花明的心路历程等。
3. 最终用于 Lab 评分的那次提交：包括代码包、分数截图、提交时平台为本次提交生成的 ID。
4. 实验报告不做字数要求，体现思考和实验过程即可（反卷第一名）。即使最后有 bug 没解决，没有以满分通过平台测试，讲一讲你的 debug 过程与思考也会获得一些分数的补偿:）

5 提示与帮助

1. 课上讲授的 RISCV 核是 64bit 的，但本 lab 是32bit的设计。
2. 常见的，如 Intel 兼容机器、Android、IOS 机器是以小端法存储的，但本次 lab 为了简化设计、方便调试，采用了大端法存储。即如 tb 文件和 hex 文件所示，0x01234567 的 01 存储在一个 32bit 寄存器的第一个 Byte，23 存储在第二个 Byte（低地址存放高 bit 数据）。
3. 大端法与小端法详解

4. 图2(b)已更新。
5. 请在 define.v 中添加一些宏定义，使用宏定义是一个更方便编程和 Debug 的习惯，这在后面 lab 规模越来越大的情况下非常重要。
6. 允许自己添加 wire、reg（甚至很多时候不得不添加。）但不要改动已经给出的端口。
7. 对设计一些不太明白的地方，建议参考 tb 文件，这样有助于理解各种定义。