中国科学院大学
《计算机组成原理(研讨课)》实验报告

姓名 张钰🄵 学号 2022K8009926020 专业 计算机科学与技术

实验项目编号 1 实验名称 基本功能部件设计

注 1:	撰写此 Word 格式实验报告后以 PDF 格式保存 SERVE CloudIDE 的 `/home/serve-ide/cod-lab/reports` 目录下 (注意: reports 全部小写)。文件命名规则: `prjN.pdf`, 其中 `prj` 和后缀名 `pdf` 为小写, `N` 为 1 至 4 的阿拉伯数字。例如: `prj1.pdf`。PDF 文件大小应控制在 5MB 以内。此外, 实验项目 5 包含多个选做内容, 每个选做实验应提交各自的实验报告文件, 文件命名规则: `prj5-projectname.pdf`, 其中 "-" 为英文标点符号的短横线。文件命名举例: `prj5-dma.pdf`。具体要求详见实验项目 5 讲义。
注 2:	使用 `git add` 及 `git commit` 命令将实验报告 `PDF` 文件添加到本地仓库 master 分支, 并通过 `git push` 推送到实验课 SERVE GitLab 远程仓库 master 分支(具体命令详见实验报告)。
注 3:	实验报告模板下列条目仅供参考, 可包含但不限定如下内容。实验报告中无需重复描述讲义中的实验流程。

一、 逻辑电路结构与仿真波形的截图及说明(比如 Verilog HDL 关键代码段 {包含注释} 及其对应的逻辑电路结构图 {自行画图, 推荐用 PPT 画逻辑结构框图后保存为 PDF, 再插入到 LaTeX. 中}、相应信号的仿真波形和信号变化的说明等)

- **寄存器堆设计**。verilog 代码:

fpga > design > ucas-cod > hardware > sources > reg_file > reg_file.v

```
1   `timescale 10 ns / 1 ns
2
3   `define DATA_WIDTH 32
4   `define ADDR_WIDTH 5
5
6   module reg_file(
7           input                       clk,
8           input  [`ADDR_WIDTH - 1:0]  waddr,
9           input  [`ADDR_WIDTH - 1:0]  raddr1,
10          input  [`ADDR_WIDTH - 1:0]  raddr2,
11          input                       wen,
12          input  [`DATA_WIDTH - 1:0]  wdata,
13          output [`DATA_WIDTH - 1:0]  rdata1,
14          output [`DATA_WIDTH - 1:0]  rdata2
15  );
16
17          wire    clk;
18          reg     [31:0] rf [31:0];
19          wire    [31:0] rdata1;
20          wire    [31:0] rdata2;
21          wire    [31:0] wdata;
22          wire    [4:0] raddr1;
23          wire    [4:0] raddr2;
24          wire    [4:0] waddr;
25
26          always @(posedge clk)
27          begin
28                  if (waddr!=0&&wen==1) //写入条件
29                          rf[waddr] <= wdata;
30          end
31
32          assign rdata1=((raddr1==0)?32'd0:rf[raddr1]);
33          assign rdata2=((raddr2==0)?32'd0:rf[raddr2]);
34
```

图 1: 寄存器堆设计

寄存器堆的设计比较简单, 在此做简单说明。整个模块需要注意两个细节: 第一个细节是寄存器堆的写入端是时序逻辑, 需要等待时钟信号上升沿并满足"waddr!=0wen==1 "的判断条件才可以写入, 而读出段则无需时序

逻辑，用 assign 赋值的组合逻辑实现即可；第二个细节是区分 rf 前后的两个 [31:0]，第一个表示寄存器堆中的每个寄存器有 32 位，而第二个则表示寄存器堆有 32 个，需要区分清楚。

- **ALU 设计**。verilog 代码：

fpga > design > ucas-cod > hardware > sources > alu > alu.v

```
1  `timescale 10 ns / 1 ns
2
3  `define DATA_WIDTH 32
4
5  `define ALUOP_AND 3'b000
6  `define ALUOP_OR 3'b001
7  `define ALUOP_ADD 3'b010
8  `define ALUOP_SUB 3'b110
9  `define ALUOP_SLT 3'b111
10
11 //加法器模块
12 module ADD(
13         input [`DATA_WIDTH-1:0] A,
14         input [`DATA_WIDTH-1:0] B,
15         input  cin,
16         output [`DATA_WIDTH-1:0] result,
17         output cout
18 );
19
20 assign {cout,result}=A+B+cin;
21
22 endmodule
23
24 //ALU模块
25 module alu(
26         input  [`DATA_WIDTH - 1:0]  A,
27         input  [`DATA_WIDTH - 1:0]  B,
28         input  [              2:0]  ALUop,
29         output                     Overflow,
30         output                     CarryOut,
31         output                     Zero,
32         output [`DATA_WIDTH - 1:0]  Result
33 );
34
```

图 2: 加法器模块

加法器是本实验的重要模块，在此做详细说明。

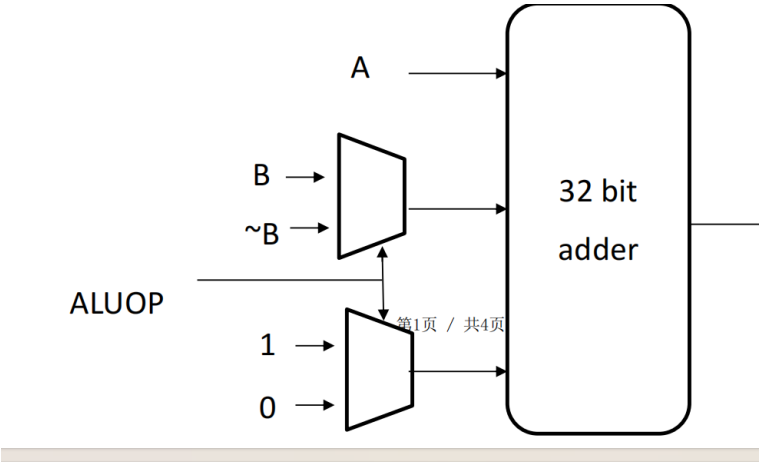

图 3: 加法器电路图

为了只利用一个加法器就完成加减以及比较运算，需要在加法器模块设置三个输入端口 A，B 以及 cin，设置两个输出端口 result 和 cout。在 alu 模块实现加法运算的时候，将两个加数分别输入 A，B 两个端口并将 cin 设置为零即可；而在减法运算与比较运算（实质仍是减法运算）时，需将第二个加数设置为 B 即减数取反，cin 设置

为 1，即可将被减数减减数的运算转化为被减数加减数的相反数的补码的加法运算。两个输出端口则分别输出运算结果和进位。

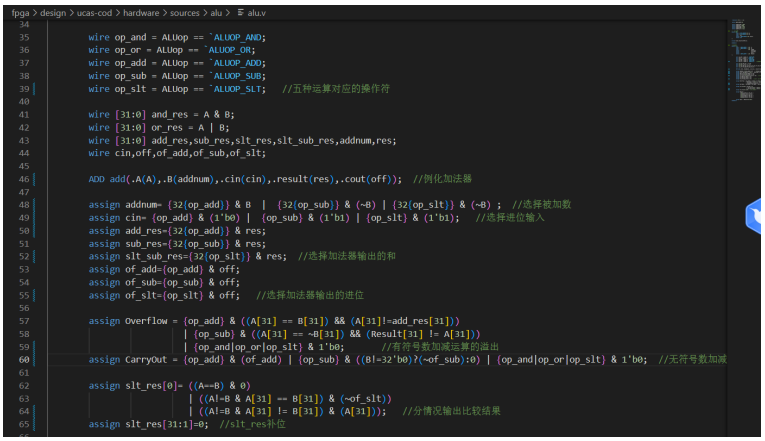

图 4: alu 模块

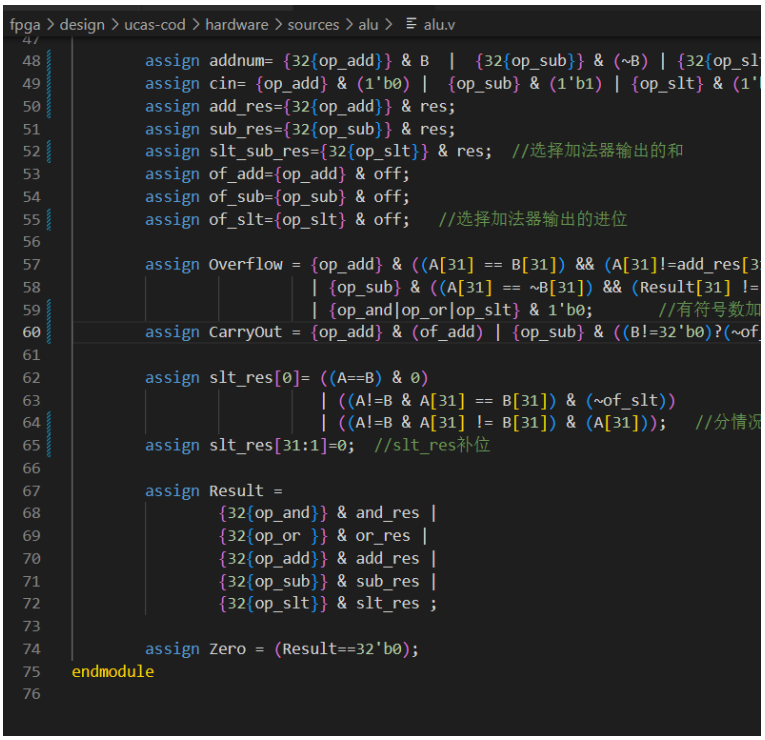

图 5: alu 模块结果输出

alu 中只需例化一次加法器模块，将 A 端口输入的值直接输入加法器模块的第一个加数，而第二个加数 addsum 和进位输入 cin 则由数据选择器选择得到（即在执行加法运算时 addsum=B，cin=0；在执行减法和比较运算时设置 addsum= B，cin=1 以实现将减法转化为补码加法），在两个输出端口则根据操作码选择所需的结果和进位输出（如当操作码对应的是加法时，表示加法器输出的结果的信号 result 被赋值给表示加法结果的变量 add-sum，表示进位的信号被赋值给加法进位变量 of-add，表示其他操作结果和溢出的变量被设置为零）。

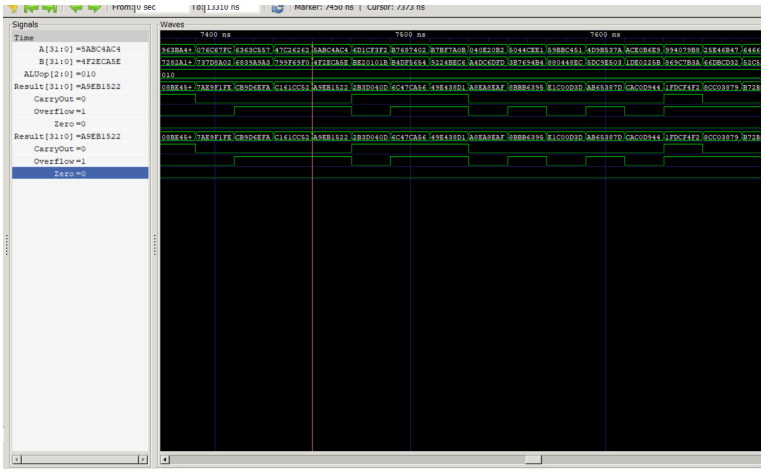

图 6: alu 模块结果输出

这是部分仿真波形图，可以看出 alu 模块得出的 Result，carryout，overflow 以及 zero 结果和参考结果一致，表明结果正确。

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，逻辑仿真和 FPGA 调试过程中的难点等）

• **寄存器堆设计问题**。寄存器堆设计中的问题已经在第一部分中说明。另外经助教提醒，寄存器堆两个输出端口的组合逻辑除了使用三目运算符来实现选择功能外也可以直接使用类似数据选择器中的与或逻辑运算实现，在需要选择的次数较多的时候相比三目运算符更加简洁。

• **ALU 设计问题**。

alu 设计中遇到的第一个问题是如何只例化一次加法器便同时实现加法，减法和比较操作。在最初设计当中我例化了三次加法器模块分别实现三种操作，但是这不符合实验要求。后续我观察到比较运算可以直接利用减法实现，并且可以通过在加法器中设置一个进位输入 cin 直接利用原有的加法器逻辑来实现补码加法运算（即实现了减法）。

第二个问题是如何理解 32op_add 的含义，在实验初期我一直没有明白 32 的含义，只是照抄了课件上面的代码实例，后来观察到 32op_add 后面跟的操作数都是 32 位，上网搜索明白了 32op_add 是用于将 op-add 扩充为 32 位以满足和后面的 32 位操作数按位与的目的。

第三个问题是如何实现 overflow 和 carryout 的计算。课件上面有对于 overflow 和 carryout 的详细说明，但是在具体实现的时候很容易因为忽略一些特殊情况导致结果出错，需要不断根据仿真波形图修改代码。

三、 对讲义中思考题（如有）的理解和回答

本实验无思考题，有关实验的问题已经在前两个部分中说明

四、 实验所耗时间

在课后，你花费了大约____5____ 小时完成此次实验。