中国科学技术大学计算机学院
《计算机组成原理实验》报告

实验题目：运算器与排序

学生姓名：李昱祁

学生学号：PB18071496

完成日期：2020/04/25

计算机实验教学中心制

2019 年 09 月

【实验目的】

1. 掌握算术逻辑单元(ALU)的功能，加/减运算时溢出、进位/借位、零标志的形成及其应用
2. 掌握数据通路和控制器的设计和描述方法

【实验环境】

1. Vivado
2. Linux 操作系统
3. FPGA OL

【实验过程】

1. ALU 设计

逻辑设计：

a. 根据信号“m”选择对应的操作类型，之后对输入的两个操作数进行相应的运算;由于 m 位宽为 3，要求设计的运算种类小于 8 种，因此未被定义的运算选择均默认进行“按位异或 ^”

b. 使用 parameter 在模块外部定义常量（数据宽度），便于进行修改及维护；

c. 加、减、按位与、按位或、按位异或等运算分别可以通过 verilog 中的+、-、&、|、^ 来实现;

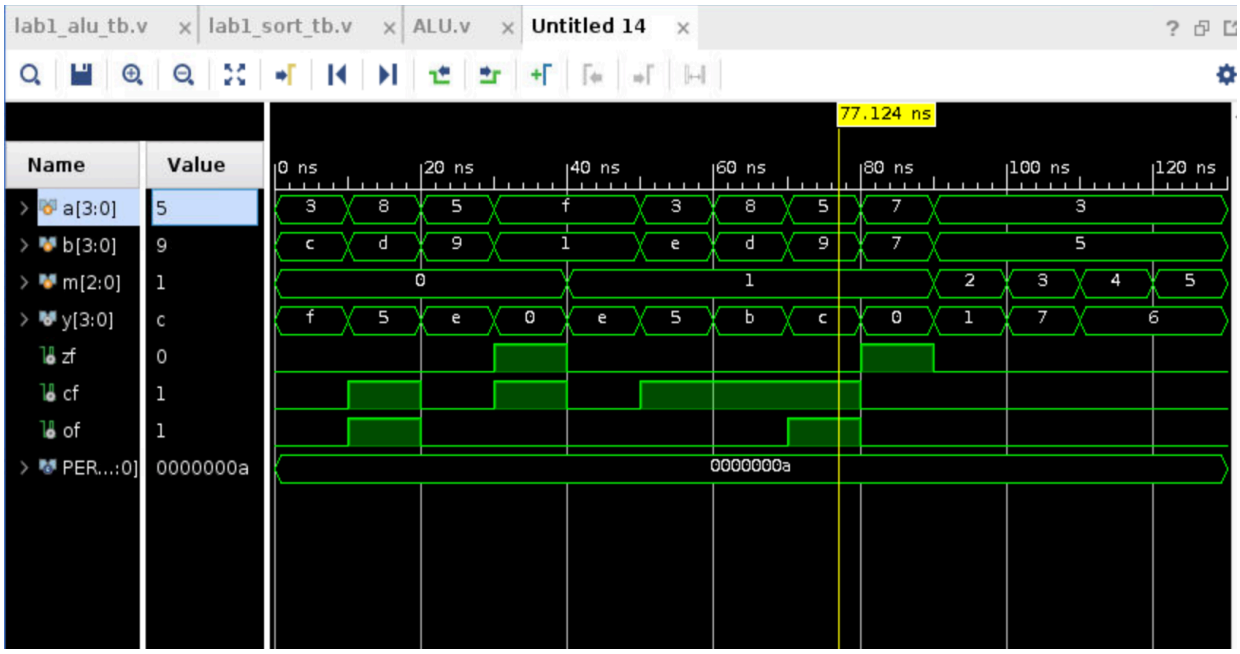
d. zf(等零)信号可以直接由运算结果是否为 0 进行判断；cf(借/进位)信号可以通过额外设置一位信号，用于接收运算结果的 WIDTH 位（最高位+1）来进行判断；of(溢出)信号在进行有符号数加减运算时有效，根据加、减时的不同情况也可进行判断

核心代码：

```
module alu
    #(parameter WIDTH = 32)
    (   output reg [WIDTH - 1 : 0] y,   //运算结果
        output reg zf,
        output reg cf,
        output reg of,
        input  [WIDTH - 1 : 0] a,b, //两个操作数
        input  [2 : 0] m           //操作类型
    );
    reg c;

    always@(*)
    begin
        c = 0;
        case(m)
            3'b000:
                begin
                    {c,y} = a + b;
                    of = (~a[WIDTH - 1] & ~b[WIDTH - 1] & y[WIDTH - 1]) |
                    (a[WIDTH - 1] & b[WIDTH - 1] & ~y[WIDTH - 1]);
                end

            3'b001:
                begin
                    {c,y} = a - b;
                    of = (~a[WIDTH - 1] & b[WIDTH - 1] & y[WIDTH - 1]) |
                    (a[WIDTH - 1] & ~b[WIDTH - 1] & ~y[WIDTH - 1]);
                end

            3'b010:
                begin
                    y = a & b;
                    of = 0;
                end

            3'b011:
                begin
                    y = a | b;
                    of = 0;
                end

            3'b100:
```

```
            begin
                y = a ^ b;
                of = 0;
            end
        default:
            begin
                y = a ^ b;
                of = 0;
            end
    endcase
    zf = (y == 0);
    cf = c;
end

endmodule
```

仿真结果：



2. 有符号数排序

逻辑设计：

a. 参考实验讲义中的设计思路，采用冒泡排序的方式，在待排数据个数已知(4 个)的情况下，将“循环比较”设计成有限状态机的 N 个状态；

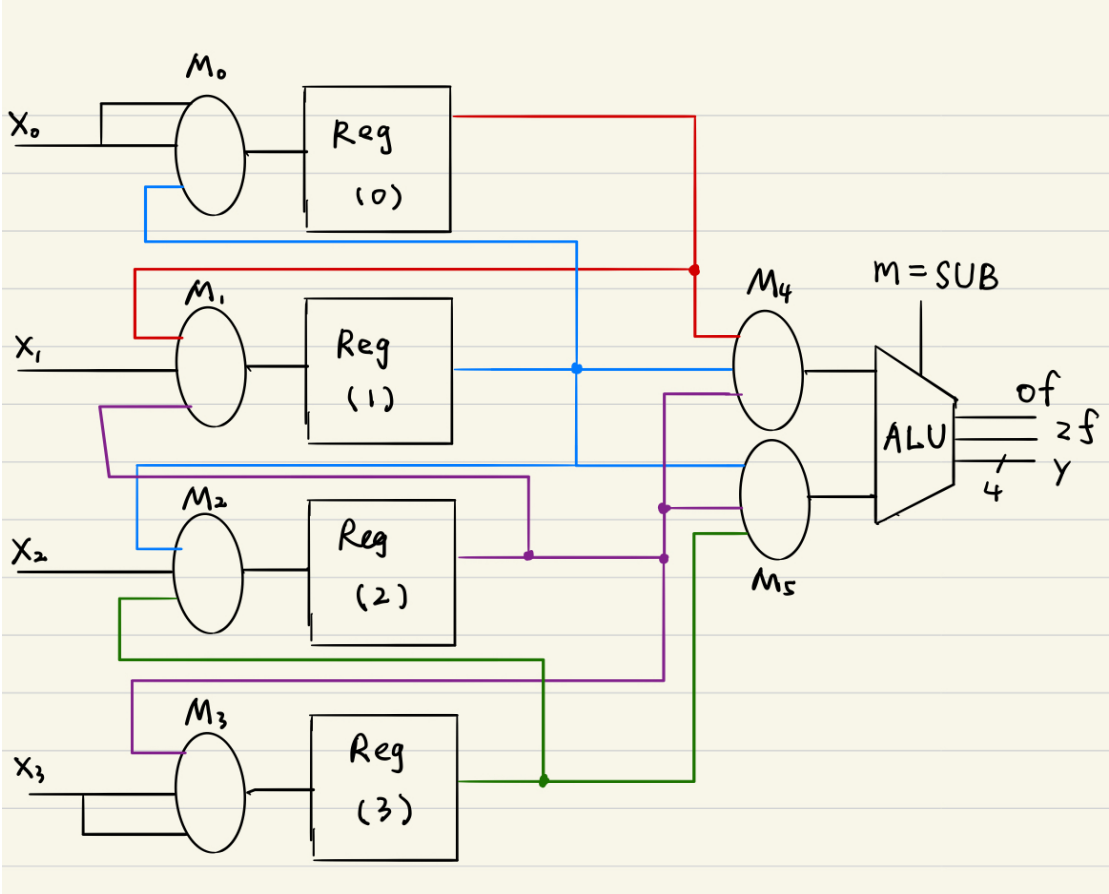
b. 利用已设计好的 ALU 的进/借位、溢出、运算结果等信息，确定两个数据之间的大小关系；由于需要对有符号数（补码）进行排序，所以可以借助 ALU 的 of（溢出）信号与运算结果的正负（最高位是 0 还是 1）进行判断：

```
en0 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
```

若产生溢出，由于 ALU 进行的是减法运算，说明两个运算数为一正一负，则若结果为负数，即 y[N-1]等于 1 时，说明 op1 为正，op2 为负，此时 op1 > op2,需要对两个数进行交换；

若不产生溢出，则直接根据两数只差是正数还是负数进行判断即可，仍比较 y 的最高位。

c. 利用如下电路，来进行数据的保存、交换、输入至 ALU：



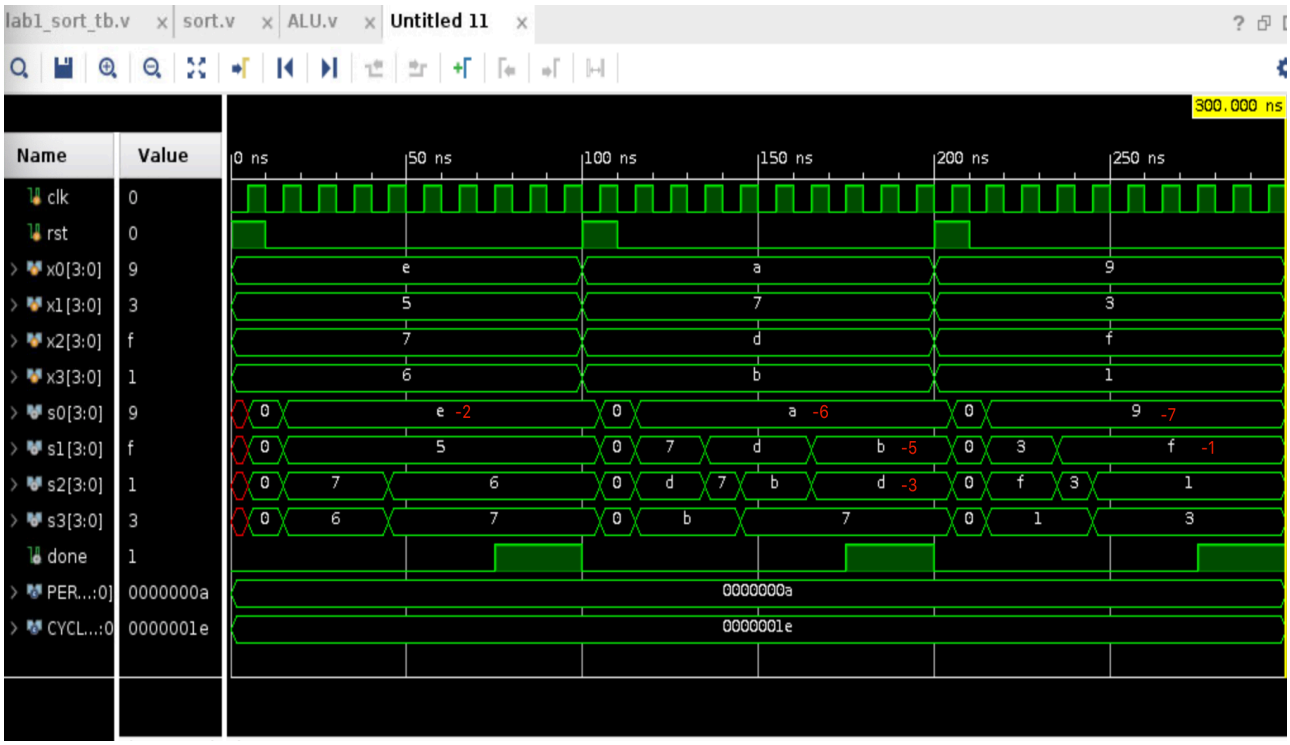
其中，3 个多选器（3 选 1）用于选择寄存器读入自己本身（不交换）、上一个数据（交换）、下一个数据（交换），以及控制 ALU 的两个输入数据来自于哪个寄存器：R0~R2 可以输入到 ALU 的第一个输入端，R1~R3 可以输入至 ALU 第二个输入端。两数比较时，寄存器标号较小的数据进入 ALU 第一个输入，寄存器标号较大的数据进入 ALU 第二个输入。

核心代码：

```
//output
always@(*)
begin
    {en0,en1,en2,en3} = 4'h0;
    done = 0;
    case(current_state)
        LOAD:
        begin
            {m0,m1,m2,m3} = 8'b10101010;
            {m4,m5} = 4'b0101;
            {en0,en1,en2,en3} = 4'b1111;
        end
        CX01_1,CX01_2,CX01_3:   //第 1、2 个数进行比较
        begin
            m0 = 2'b11;
            m1 = 2'b01;
            m4 = 2'b01;
            m5 = 2'b01;
            en0 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
            en1 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
        end
        CX12_1,CX12_2:          //第 2、3 个数进行比较
        begin
            m1 = 2'b11;
            m2 = 2'b01;
            m4 = 2'b10;
            m5 = 2'b10;
            en1 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
```

```
            en2 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
        end
        CX23_1:                 //第 3、4 个数进行比较
        begin
            m2 = 2'b11;
            m3 = 2'b01;
            m4 = 2'b11;
            m5 = 2'b11;
            en2 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
            en3 = (of & y[N - 1] & ~zf)|(~of & ~y[N - 1] & ~zf);
        end
        HLT:
            done = 1;
    endcase
    {s0,s1,s2,s3} = {r0,r1,r2,r3};
end
```

仿真结果：



（如图，负数补码已经在旁边标出其对应值）

对于测试用例，排序电路在在信号 done 值为 1 时，均将 4 个有符号数进行了排序，验证了电路在行为级上的正确性。

【思考题】

1. 如果要求排序后的数据是递减顺序，电路如何调整？

答：

思路 1：

在 Verilog 中：

只需在赋值时，按与“递增设计”相反的顺序把 4 个寄存器中的值赋给 s0~s3 即可

比如：

```
{s0,s1,s2,s3} = {r0,r1,r2,r3};
```

改为：

```
{s3,s2,s1,s0} = {r0,r1,r2,r3};
```

对应到电路中：

相当于把原来 4 个寄存器的输出值与 sort 模块排序结果 4 个输入值之间的线路关系调整了一下

思路 2：

在 verilog 中，将控制使能信号的表达式变为：

```
en = (of & ~y[N - 1] & ~zf)|(~of & y[N - 1] & ~zf);
```

即将寄存器写使能信号 eni 在只有 op1 < op2 时才有效（op1，op2 均为补码表示的有符号数）

对应到电路中：

即产生 en0~en3 信号的组合逻辑电路发生了变化（具体来

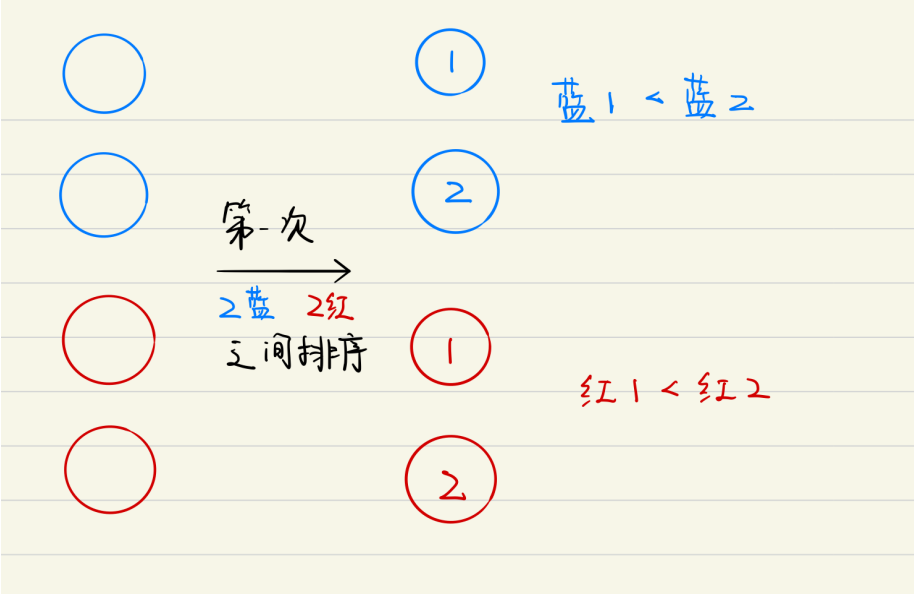
讲，可能是一个非门改变了位置）

2. 如果为了提高性能，使用两个 ALU，电路如何调整？

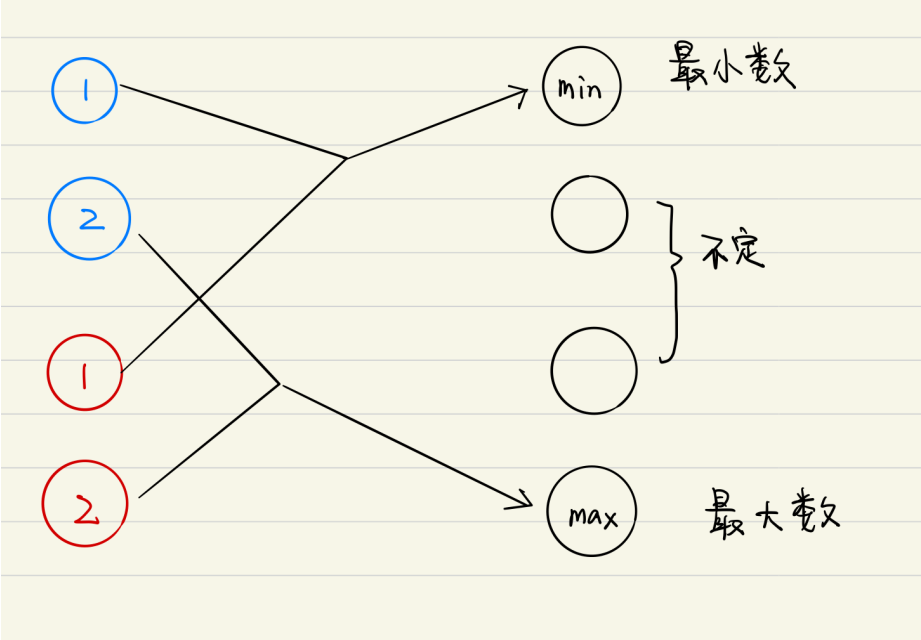
答：并行排序，大致流程如下：

第一次比较：

前两个数(蓝色)之间比较，后两个数(红色)之间比较，分别比较出两组之间的较大者与较小者，并按顺序排放

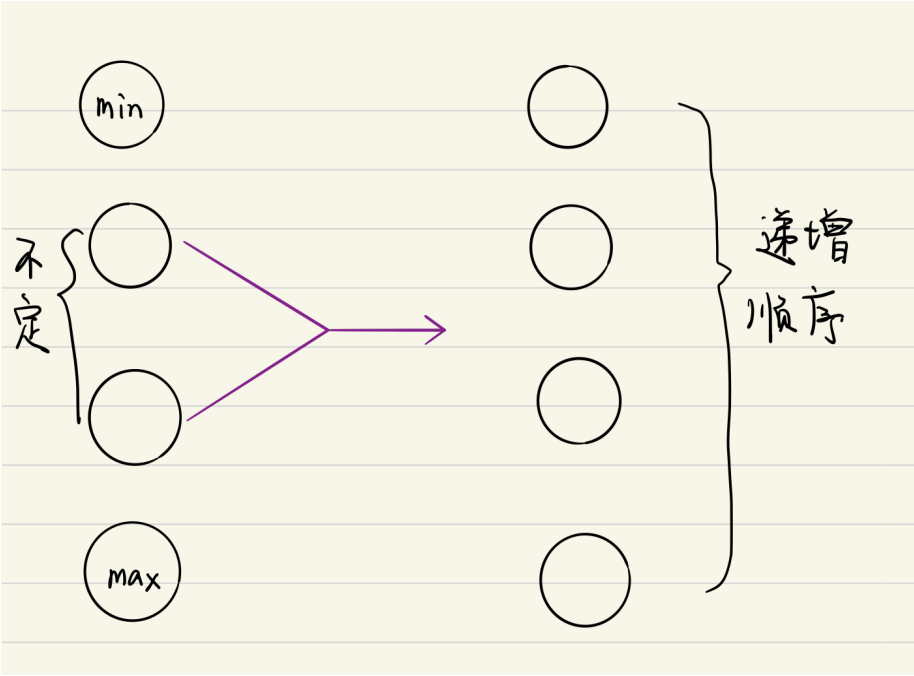


第二次比较：两个较大者(蓝 2 与红 2)之间进行比较，两个较小者(蓝 1 与红 1)之间进行比较，可得到四个数中的最大值、最小值，并按顺序排放



第三次比较：

该周期只用进行一次比较，即比较中间两个数的大小并根据需要进行交换即可



综上分析可知，使用两个 ALU 同样的数据进行排序时，仅需要 3 个时钟周期即可完成排序工作；相比之下，仅用一个 ALU 冒

泡排序需要长达 6 个周期的时间。故使用两个 ALU 可以提高排序的效率。

电路调整：

每个寄存器不止与相邻的寄存器进行数据交换，故 4 个寄存器输入端的多选器需要进行调整；多选器的选择信号与寄存器写入的使能信号也要相应调整，略。

【实验总结】

a. 复习了上学期所学的 Verilog 语法、有限状态机的实现、vivado 项目构成、仿真等知识；

b. 实现了一个基本的 CPU 组成模块——ALU.之后，利用其功能特性实现了有符号数的排序；

c. 建议：本次实验的讲义，相较于上学期数字电路实验的讲义略显简单。希望后期教学能给出更为详细的实验介绍、具体要求细节等