计算机组成原理实验 实验报告

实验题目：Lab1_运算器与排序

学生姓名：王章瀚

学生学号：PB18111697

完成日期：2020 年 5 月 4 日

计算机实验教学中心制

2019年09月

1 实验题目

Lab01 运算器与排序

2 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

3 实验平台

Vivado

4 实验过程

4.1 ALU

4.1.1 基本过程

根据老师给出的ALU功能表(下表), 很容易设计出相应的ALU模块.

m	y	cf	of	zf
000	a+b	*	*	*
001	a-b	*	*	*
010	a&b	x	x	*
011	a—b	x	x	*
100	a^b	x	x	*
其他	x	x	x	x

4.1.2 代码讲解

```
module ALU
    #(parameter WIDTH = 4) // 数据位宽
    (
    output reg [WIDTH-1: 0] y,    // 运算结果
    output reg zf,   // 零标志
    output reg cf,   // 进位
```

```
7       output reg of,  // 溢出标志
8       input [WIDTH-1: 0] a, b,     // 两操作数
9       input [2: 0] m // 操作类型
10      );
11
12      always @(*)
13      begin
14          // cf, y
15          case(m)
16              3'b000: {cf, y} = a + b;
17              3'b001: {cf, y} = a - b;
18              3'b010: {cf, y} = {1'b0, a & b};
19              3'b011: {cf, y} = {1'b0, a | b};
20              3'b100: {cf, y} = {1'b0, a ^ b};
21              default: {cf, y} = {(WIDTH+1){1'b0}};
22          endcase
23          // of
24          case(m)
25              3'b000: of = (~a[WIDTH-1] & ~b[WIDTH-1] & y[WIDTH-1]) | (a[WIDTH-1] & b[WIDTH-1] & ~y[
        WIDTH-1]);
26              3'b001: of = (~a[WIDTH-1] & b[WIDTH-1] & y[WIDTH-1]) | (a[WIDTH-1] & ~b[WIDTH-1] & ~y[
        WIDTH-1]);
27              default: of = 1'b0;
28          endcase
29          // zf
30          zf = ~|y;
31      end
32
33  endmodule
```

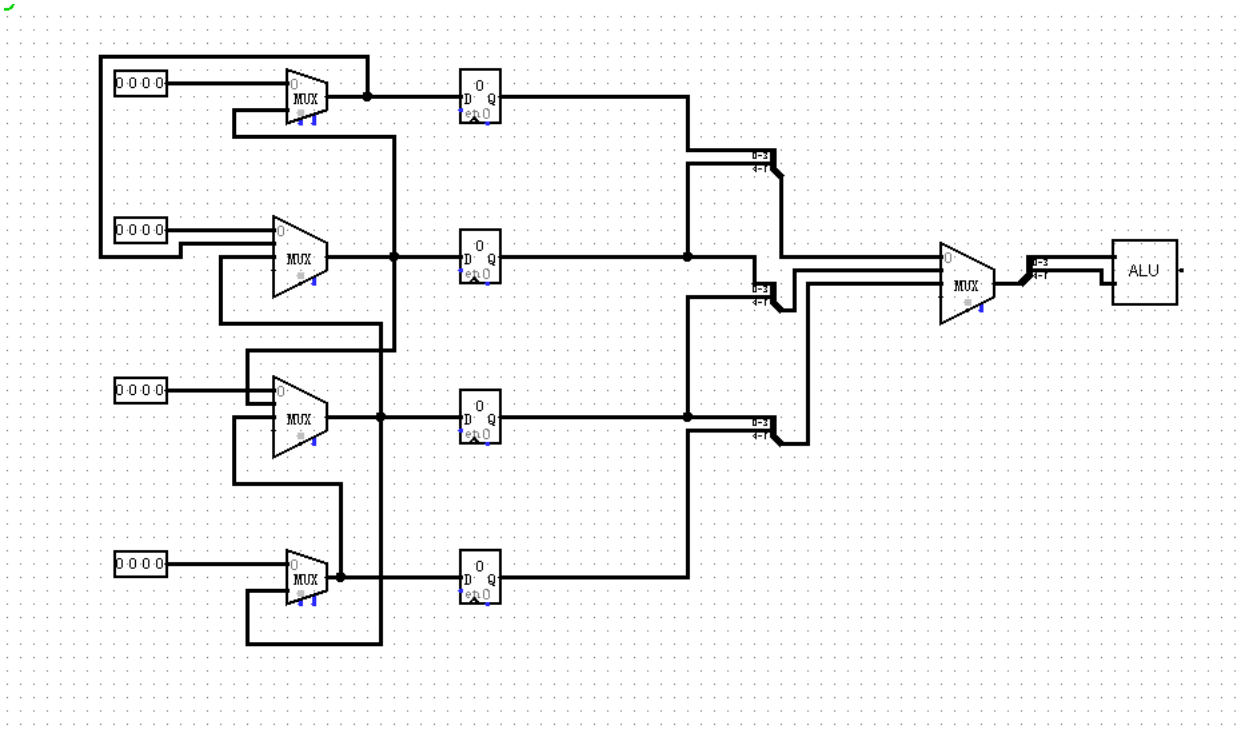
在always语句块中, 第一个case用以对cf和y赋值(按照功能表很容易写出); 而后第二个case对of赋值, 最后对zf赋值. 这样写可以避免产生任何多余的锁存器.

4.2 sort

排序的程序中, 对四个寄存器的输入都有一个mux来选择x或者与之相邻的寄存器(用以数据交换). 而在四个数据寄存器与ALU之间有有一个三选一的mux, 用以选择要比较的数据. 比较的过程即为冒泡排序的过程, 第一轮对(r0, r1), (r1, r2), (r2, r3); 第二轮对(r0, r1), (r1, r2); 第三轮对(r0, r1)即可.

4.2.1 数据通路

按照上面的描述, 容易做出如下数据通路 大致如下图(一些相对不重要或比较明确的线没有连上, 以免过于混乱)




```
// Data Path
//// mux
////// 用于交换的mux
mux2 mux2_0(.select(m0), .in0(x0), .in1(s1), .out(i0));
mux3 mux3_1(.select(m1), .in0(x1), .in1(s0), .in2(s2), .out(i1));
mux3 mux3_2(.select(m2), .in0(x2), .in1(s1), .in2(s3), .out(i2));
mux2 mux2_3(.select(m3), .in0(x3), .in1(s2), .out(i3));
////// 用于输入到ALU的mux
mux3 #(.WIDTH(2*N)) mux_ab(.select(mab), .in0({s0, s1}), .in1({s1, s2}), .in2({s2, s3}), .out({
    alu_a, alu_b}));
//// register
register register0(.data(i0), .en(en0), .rst(rst), .clk(clk), .r(s0));
register register1(.data(i1), .en(en1), .rst(rst), .clk(clk), .r(s1));
register register2(.data(i2), .en(en2), .rst(rst), .clk(clk), .r(s2));
register register3(.data(i3), .en(en3), .rst(rst), .clk(clk), .r(s3));
//// ALU
ALU ALU(.a(alu_a), .b(alu_b), .m(SUB), .cf(cf));
```

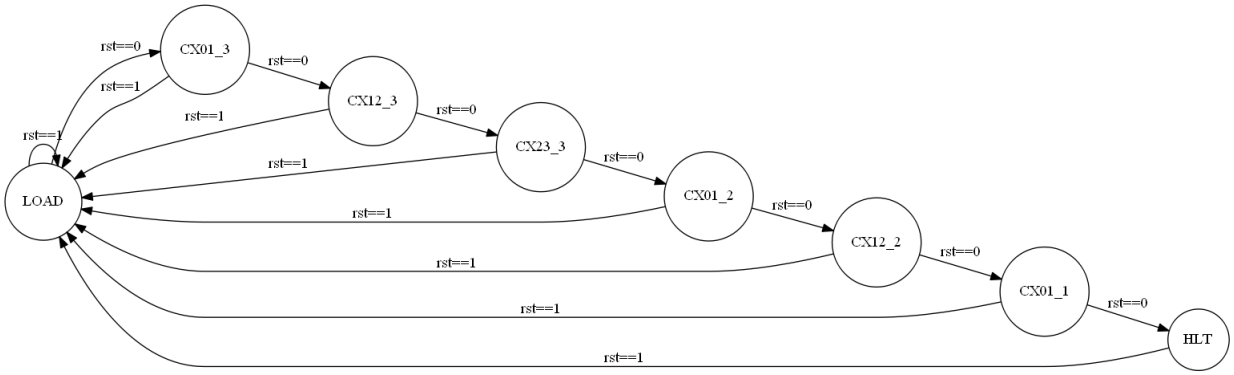
4.2.2 状态机控制

按照冒泡排序的过程, 有如下状态机控制.

```
// Control Unit
always @(posedge clk or posedge rst)
begin
    if(rst) cur_state <= LOAD;
    else cur_state <= next_state;
end

// States
always @(*)
begin
    case(cur_state)
        LOAD: next_state = CX01_3;
        CX01_3: next_state = CX12_3;
        CX12_3: next_state = CX23_3;
        CX23_3: next_state = CX01_2;
        CX01_2: next_state = CX12_2;
        CX12_2: next_state = CX01_1;
        CX01_1: next_state = HLT;
        HLT: next_state = HLT;
        default: next_state = HLT;
    endcase
end
```

其状态转移图大致如下:



4.2.3 控制单元

这一部分是对数据交换的使能, 对ALU输入的选择等. 其中mx表示对x的选择; enx表示对x寄存器的使能; CXab_i表示第i轮中对寄存器a和b的比较与交换.

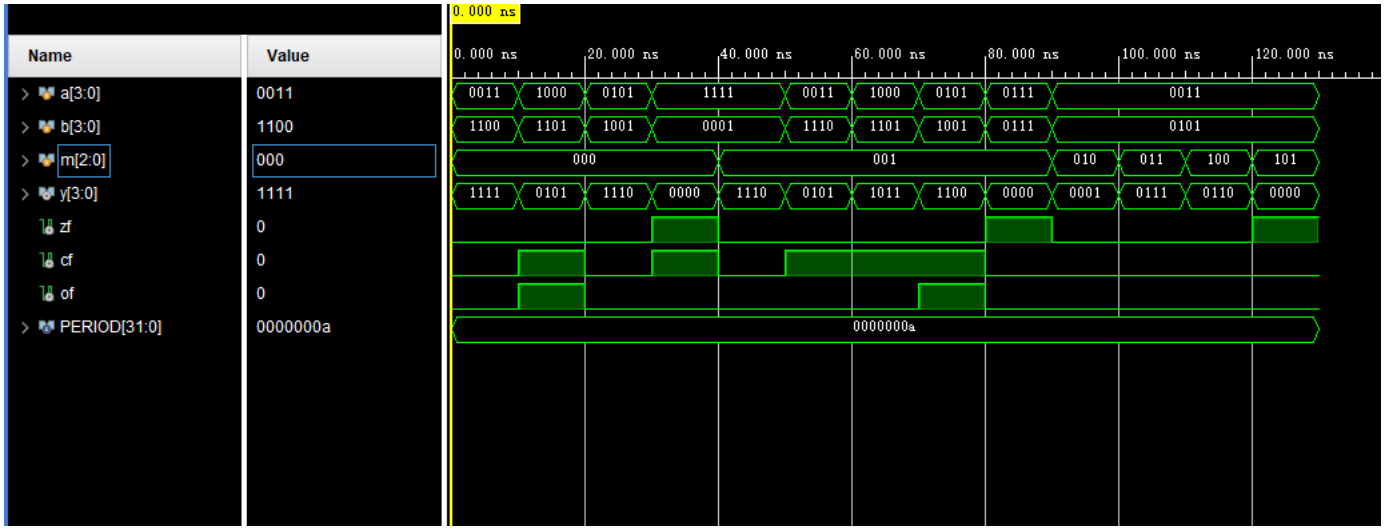
```
always @(*)
begin
    // 初始化
    {mab, m0, m1, m2, m3, en0, en1, en2, en3, done} = 13'h00;
```

```
    case(cur_state)
        LOAD:
        begin
            {en0, en1, en2, en3} = 4'b1111;
        end
        CX01_3, CX01_2, CX01_1:
        begin
            mab = 2'b00;
            en0 = ~(y[N-1] ^ of); en1 = ~(y[N-1] ^ of);
            m0 = 1'b1; m1 = 2'b01;
        end
        CX12_3, CX12_2:
        begin
            mab = 2'b01;
            en1 = ~(y[N-1] ^ of); en2 = ~(y[N-1] ^ of);
            m1 = 2'b10; m2 = 2'b01;
        end
        CX23_3:
        begin
            mab = 2'b10;
            en2 = ~(y[N-1] ^ of); en3 = ~(y[N-1] ^ of);
            m2 = 2'b10; m3 = 1'b1;
        end
        HLT: done = 1'b1;
    endcase
end
```

5 实验结果

5.1 ALU

用老师给出的testbench可以得到如下波形图,



可以看到当m为000时的加法, 为001时的减法等都得到了相应的正确结果.

而对于m=101的情况, 则默认对{cf, y}赋值为{(WIDTH+1){1'b0}}, 其余的仍按原过程实现.

5.2 sort

由于老师给出的testbench只有3个数据, 于是自己编写了一个testbench, 随机生成4个数来排序. 仿真结果如下:

其中sx显示为0的部分是rst的结果. 可以看到每组输入数据最终都完成了从小到大的有符号排序.

6 思考题

6.1 如果要求排序后的数据是递减顺序，电路如何调整？

递减顺序的话, 直接将输入ALU的两个操作数对调即可.

6.2 如果为了提高性能，使用两个ALU，电路如何调整？

可以使用奇偶并行排序算法. 先比较交换第1,2个和第3,4个; 再第2,3个; 再第1,2个和第3,4个.
这时s1,s2,s3,s4就变为有序的了.

7 心得体会

本次实验完成了ALU的设计, 这对后期设计CPU将有很大帮助. 此外这个排序器的实现方式, 让我回忆起了状态机的设计流程.

8 意见建议

本次实验相对简单，没有什么意见建议。

但希望老师讲课的网络环境可以得到改善，否则时不时就卡一下，很影响效率。