实 验 报 告

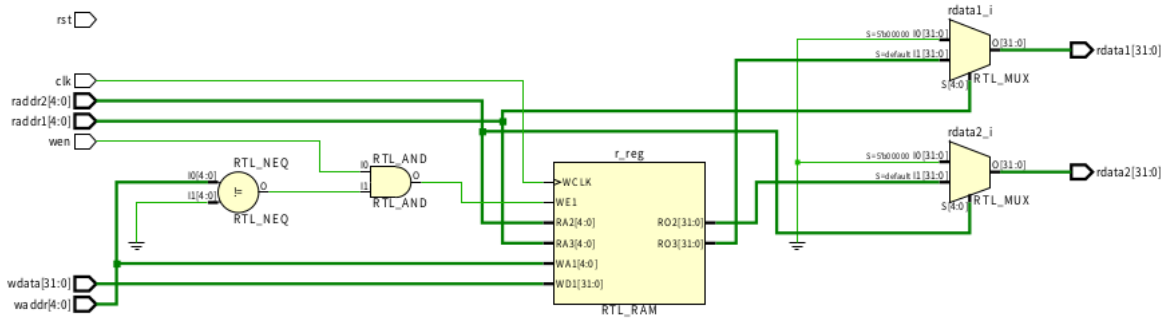
学号：2019K8009915022　　姓名：栗祺菁　　专业：计算机科学与技术

实验序号：1　　实验名称：基本功能部件RF与ALU设计

一、 逻辑电路结构与仿真波形的截图及说明（比如关键RTL代码段{包含注释}及其对应的逻辑电路结构、相应信号的仿真波形和信号变化的说明等）

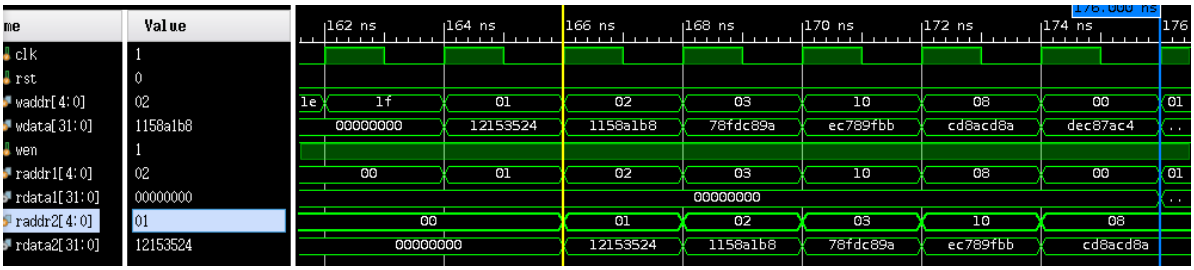
1.RF:

逻辑电路结构：




```
alway @(posedge clk)begin
    if(wen&& waddr!=5'b0)
        r[waddr] <= wdata;
end
//满足要求：当使能信号为真且写地址不为1的时候，进行写操作。这里对应着原理图中的寄存器模块RAM
assign rdata1=raddr1==0?32'b0:r[addr1];
assign rdata2=raddr2==0?32'b0:r[addr2];
//满足要求：读取第0个寄存器的值时，输出为全零；读取其它位置的值时，按照输入情况进行输出。这里对应着原理图中的选择器mux
```

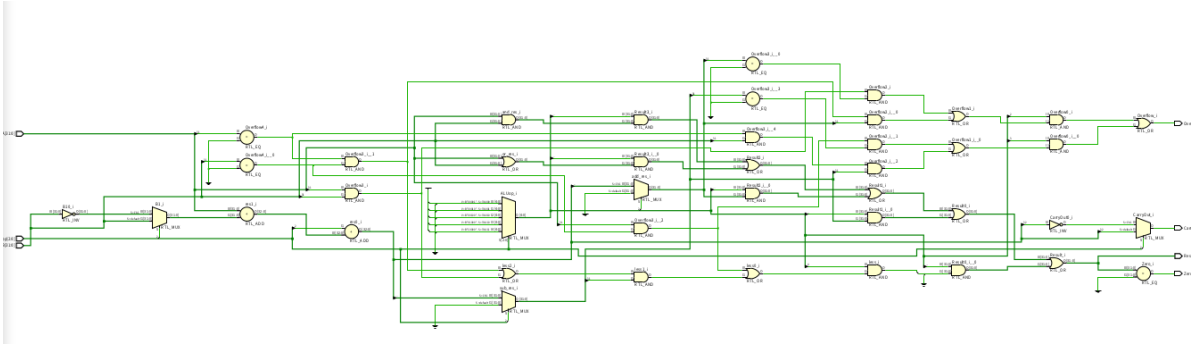
波形：



由图中黄线和蓝线所在位置可以知道，根据读的地址，依次开始输出寄存机中存的值，对照写地址和写数据可知，存的值就是输入的值wdata。且在此之前，读地址为0的时候，读数据1和2输出都是全零。

2.ALU:

逻辑电路结构：



感觉比较复杂，注意到上图有两个加法器，实际上用到的是一个带有进位的全加器。{cout，res}=A+B1+Cin,其中A就是输入的A，B1根据是否为减法操作或者比较操作（即ALUop[2]==1），为~B或B，同样地Cin在进行减法或者比较操作时为1，其它情况为0。

Overflow（有符号加减法的溢出判断）的判断有四种情况：(1)加法：两个加数符号位为1，结果符号位为0；两个加数符号位为0，结果符号位为1（2）减法：正数减负数，得到负数；负数减正数得到正数。

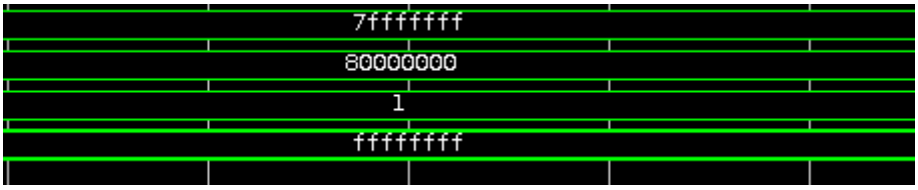
```
        wire op_and=ALUop==3'b000;
        wire op_or=ALUop==3'b001;
        wire op_add=ALUop==3'b010;
        wire op_sub=ALUop==3'b110;
        wire op_slt=ALUop==3'b111;

        wire [`DATA_WIDTH - 1:0] and_res = A&B;
        wire [`DATA_WIDTH - 1:0] or_res = A|B;
        wire [`DATA_WIDTH - 1:0] add_res;
        //assign {CarryOut,add_res} = op_add? A+B:((op_sub || op_slt)? A+~B+1:{0,32'b0});
        wire [`DATA_WIDTH - 1:0] sub_res;
        //wire temp;
        //assign {temp,sub_res} = A+~B+1;
        wire [`DATA_WIDTH - 1:0] B1;
        wire Cin,cout;
        wire [`DATA_WIDTH - 1:0] res;
        assign B1=ALUop[2]?~B:B;
        assign Cin=ALUop[2]?1:0;
        assign {cout,res}=A+B1+Cin;
        assign add_res=op_add?res:0;
        assign sub_res=ALUop[2]?res:0;
        assign CarryOut=ALUop[2]? !cout:cout;

        assign Overflow =( (op_add && ((A[31]==1 && B[31]==1 && add_res[31]==0)||(A[31]==0 &&
B[31]==0 && add_res[31]==1))) || (op_sub && ((A[31]==1 && B[31]==0 && sub_res[31]==0)||(A[31]==0
&& B[31]==1 && sub_res[31]==1))) );

        wire less = (op_slt && ((A[31]==1 && B[31]==0) || (((A[31]==0 && B[31]==0)||(A[31]==1 &&
B[31]==1)) && sub_res[31]==1)));
        wire [`DATA_WIDTH - 1:0] slt_res ={31'b0, less};

        assign Result= {32{op_and}} & and_res | {32{op_or}} & or_res | {32{op_add}} & add_res |
{32{op_sub}} & sub_res | {32{op_slt}} & slt_res;

        assign Zero= (Result==32'b0);
```

波形：

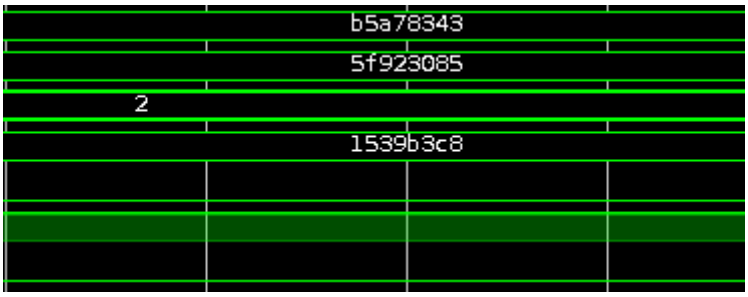
第一行为数据A，第二行为数据B，第三行是操作代码ALUop，第四行是结果result，第五行是Overflow，第六行是Carryout，第七行是Zero

1.与操作

2.或操作

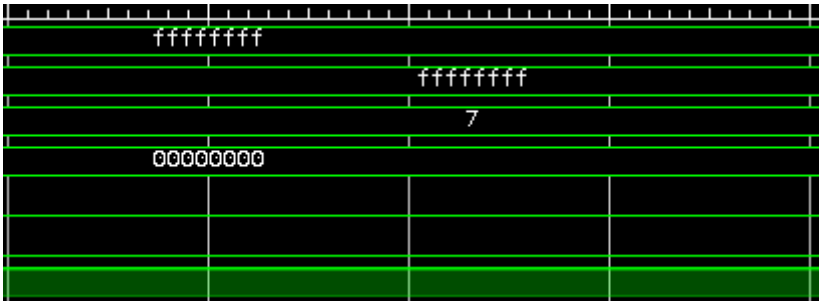


3.加法操作



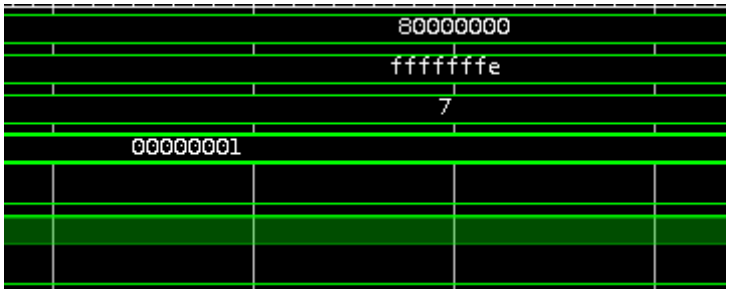
由图可示，十六进制加法。若表示无符号数加法，则b5a78343+5f923085=1,1539b3c8；有进位1（此时指的是十六进制进位），carryout表示无符号数加减法的进位和借位，此时为1；若这个表示有符号数加法，则数A首位为1，表示负数，数B首位为0，表示正数，得到的结果仍然为1,1539b3c8。由溢出判别逻辑，负数加正数，不会溢出，overflow表示有符号数加减法的溢出与否，故此时overflow为0。

4.减法操作



两个相同的数相减，得到的结果为全0，并且最下面一条的Zero为1，表示结果为全0。

5.比较操作



slt是有符号数整数比较，由图所示两个数都是负数，根据我写的逻辑，两个数的正负性相同时，根据减法结果来判断。在补码下1000,0000代表的是-128,11111110代表的是-2，前者小于后者。

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，仿真、云平台调试过程中的难点等）

在写reg_file的时候，我开始将第0号寄存器赋值为0的操作放在always块中，这样每一个时钟上升沿都会执行一次，虽然可以通过测试，但是得到的原理图十分复杂。后来我将其更改为在向外读值的时候多进行一次判断，判断若读地址为0，则输出为0，直接用组合逻辑，简化了电路。我第一次提交的时候用到了initial块进行初始化，经老师提醒，删去了initial块部分，增加了规范性，并由此了解到verilog和c语言的一个很大不同，可以不用初始化，c语言中指针，数组，变量值不初始化的话，很有可能返回出错，但是在verilog中，只要在testbench或者之后的流程中不用到未初始化的值，就不会出错。

在写alu的之前，老师在如何写好verilog中讲到的独热码的概念，我在夏闻宇老师的书上也有看到，他那里是用在状态机上，这次的实验正好让我复习巩固这个知识。独热码的好处之一就是方便地根据各个位置的值来判断操作，这里并不是严格的独热码，但是思想是相同的。sub和slt的第二位都是1，而不是按照编码顺序将ALU_op编为3'b011，这是因为这两个操作有交集，有很多情况都需要用到减法，就可以直接根据ALU_op第二位是否为0来判断。因为要求少用加法器，我最开始用的？：语句，在verilog中和c语言不一样，不是根据条件是否满足而只执行一个操作，而是两个都执行了，最后再通过一个选择器选择一个，这样就会有多个加法器。后来在老师的指导下，进行了更改，最后只用了一个全加器。

三、 对讲义中思考题（如有）的理解和回答

无思考题

四、 在课后，你花费了大约 3 小时完成此次实验。

五、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，对实验项目的建议，对提供帮助的同学的感谢，以及其他想与任课老师交流的内容等）

实验难度适中，我觉得可以增加一点测试案例，看波形的时候感觉有的情况案例不多。感谢刘士祺和王嵩岳助教的耐心指导，课后出现了问题，也都很快地回复我。