《数字逻辑与数字系统》实验报告

学院 智算学部 年级 2019 级 班级 一 班 姓名 俞林昊 学号 3019207450

课程名称 数字逻辑与数字系统 实验日期 2021.4.25 成绩

同组实验者 无

实验项目名称 ALU 的设计与实现

一. 实验目的

1. 掌握全加器和行波进位加法器的结构；
2. 熟悉加减法运算及溢出的判断方法；
3. 掌握算术逻辑单元（ALU）的结构；
4. 熟练使用SystemVerilogHDL的行为建模和结构化建模方法对ALU进行描述实现；
5. 为“单周期MIPS处理器的设计与实现”奠定基础。

二. 实验内容

基于 SystemVerilog HDL 设计并实现一个 4 位 ALU 单元。整个工程的顶层模块如图 3-4 所示，输入/输出端口如表 3-2 所示。ALU 单元以拨动开关作为操作数 A、B 以及操作类型控制信号 aluop 的输入，使用 2 个七段数码管显示 ALU 单元的运算结果（1 个数码管用于显示非乘法结果以及乘法结果的低 4 位，另一个数码管显示乘法结果的高 4 位），使用 2 个 LED 灯显示加/减法是否溢出。注意，顶层模块由两个子模块组成，其中，一个是 ALU 单元，另一个是 7 段数码管动态显示扫描单元。同学们只需要实现 ALU 单元即可，动态显示扫描单元在工程中直接提供。

1. ALU 单元的输入 A 和 B 均是补码形式。
2. 实现加法和减法时，不能使用 “+”和 “-”两种运算符，且只能通过一个行波进位加法器和其它必要的逻辑电路实现。
3. 可以使用 “*”运算符实现乘法，但该运算符 在只适用无符号数的乘法，有符号数的乘法需要同学们考虑如何处理。
4. 实现算术右移时， 可以 使用运算符“>>>”。

天津大学本科生实验报告专用纸

三．实验原理与步骤（注：步骤不用写工具的操作步骤，而是设计步骤）

1. 写出全加器 fulladder.sv 代码。

```
module fulladder(
 input logic A,
 input logic B,
 input logic Cin,
 output logic Cout,
 output logic S
 );
 assign {S, Cout} = A + B + Cin;
endmodule
```

2. 给出行波进位加法器 rsa.sv 的代码。

```
module rca(
  input [3 : 0]   A, B,
  input Cin,
  output logic [3 : 0]    S,
  output logic Cout
  );
  logic C0, C1, C2;
  fulladder fulladder_1(.A(A[0]), .B(B[0]), .Cin(Cin), .Cout(C0), .S(S[0]));
  fulladder fulladder_2(.A(A[1]), .B(B[1]), .Cin(C0), .Cout(C1), .S(S[1]));
  fulladder fulladder_3(.A(A[2]), .B(B[2]), .Cin(C1), .Cout(C2), .S(S[2]));
  fulladder fulladder_4(.A(A[3]), .B(B[3]), .Cin(C2), .Cout(Cout), .S(S[3]));
module
```

3. 写出 ALU 模块 alu.sv 代码。

准备工作

```
logic [3:0] addalures;
logic [3:0] a;
logic [3:0] b;
rca muxadd(.A(A),.B(B),.Cin(1'b0),.Cout(),.S(addalures));
```

基本逻辑运算

```
OF = 1'b0;
ZF = 1'b0;
alures[7:4] = 4'b0000;
case(aluop)
4'b0000: alures[3:0] = A & B;
4'b0001: alures[3:0] = A | B;
4'b0010: alures[3:0] = A ^ B;
4'b0011: alures[3:0] = ~(A & B);
4'b0100: alures[3:0] = ~A;
4'b0101: alures[3:0] = A << B[2:0];
4'b0110: alures[3:0] = A >> B[2:0];
4'b0111: alures[3:0] = A >>> B[2:0];
```

有/无符号数惩乘法

```
4'b1000: alures = A * B;
4'b1001: begin
    logic [7:0] temp1;
    logic [7:0] temp2;
    logic [15:0] temp3;
    temp1 = {A >>> 3 ,A[3:0]};
    temp2 = {B >>> 3 ,B[3:0]};
    temp3 = temp1 * temp2;
    alures= temp3 [7:0];
end
```

有/无符号数加法

```
4'b1010: begin
      a = A;b = B;
      if(A[3] ^ B[3] == 1)
          OF=1'b0;
      else if(A[3] == 1 && addalures[3]==0)
          OF=1'b1;
      else if(A[3] == 0 && addalures[3]==1)
          OF=1'b1;
      alures[3:0]=addalures;
 end
 4'b1011: begin
      a = A;b = B;
      alures[3:0]=addalures[3:0];  end
```

有/无符号数减法

```
4'b1100: begin
     a = A;b = ~B + 1;
     if(A[3] ^ B[3] == 1)
         OF=1'b0;
     else if(A[3] == 1 && addalures[3]==0)
         OF=1'b1;
     else if(A[3] == 0 && addalures[3]==1)
         OF=1'b1;
     alures[3:0]=addalures;
end

4'b1101: begin
     a = A;b = ~B + 1;
     alures[3:0]=addalures[3:0];
end
```

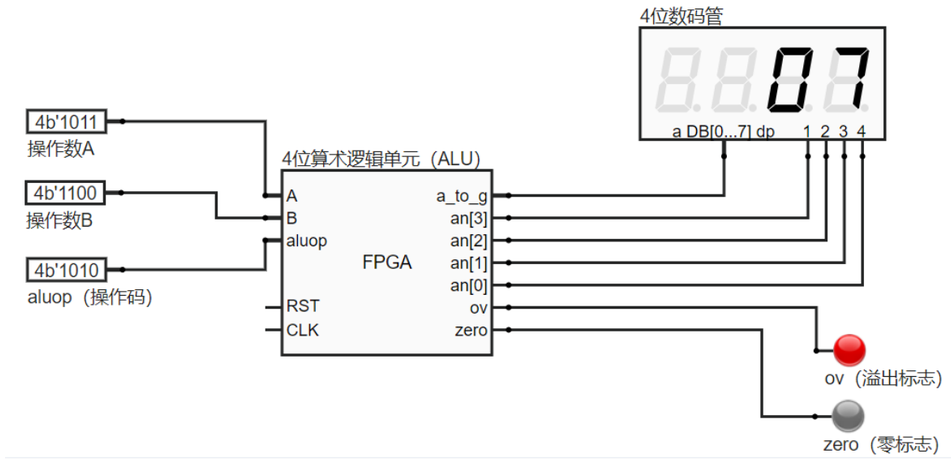
有/无符号数比较

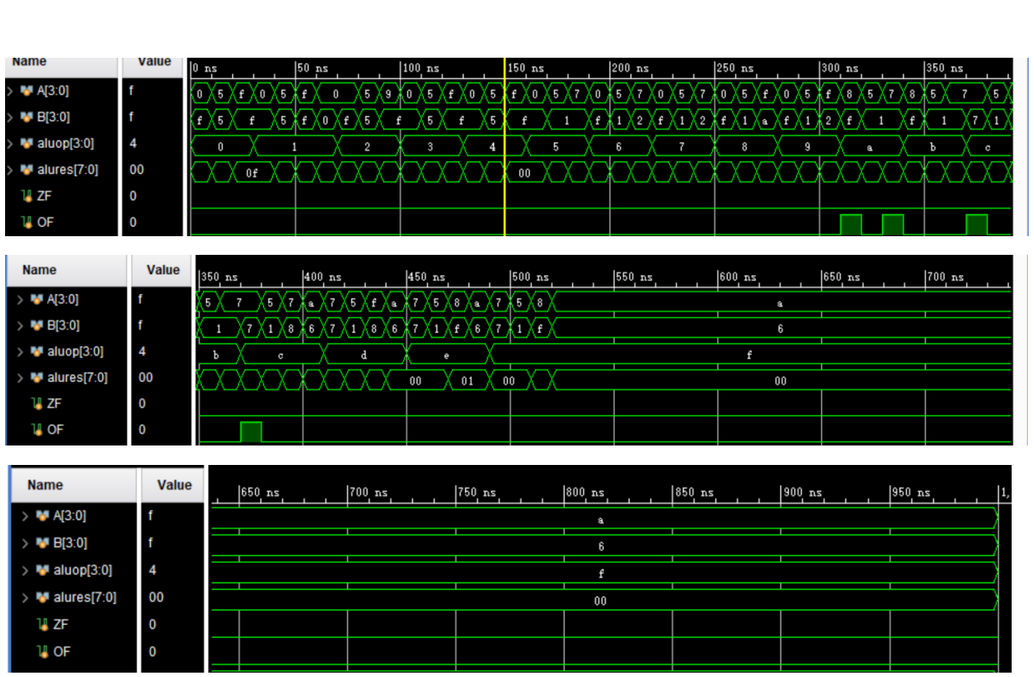
```
4'b1110: begin
    if (A[3] & B[3]) alures[3:0] = ((~B + 1) < (~A + 1));
        else if (B[3]) alures[3:0] = 0;
        else if (A[3]) alures[3:0] = 1;
        else alures[3:0] = (A < B);
end

4'b1111: alures[3:0] = (A < B);
```

四．仿真与实验结果（注：仿真需要给出波形图截图，截图要清晰，如果波形过长，可以分段截取；实验结果为远程 FPGA 硬件云平台的截图）

注：远程 FPGA 硬件云平台截图只需要一个测试激励即可





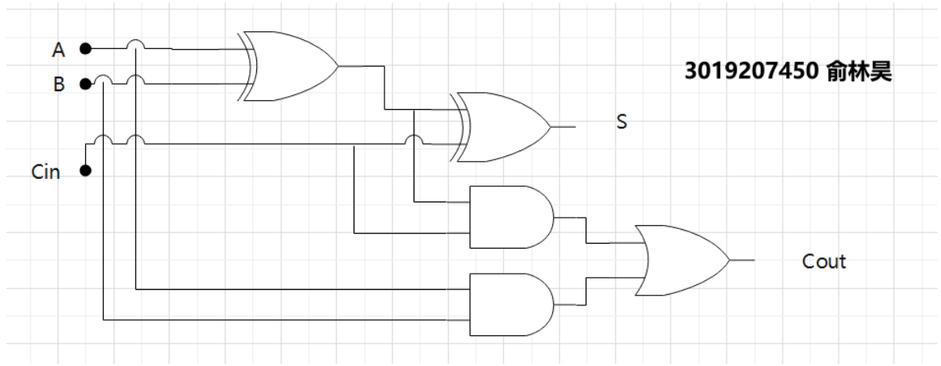
五．实验中遇到的问题和解决办法

1. 在判断有符号数乘法的时候感觉回到了之前上计算机组成原理的时候，采用的方法是先对操作数做符号位的扩展，然后乘完了之后，在截断。还有一种暴力算法就是分别判断两个操作数的正负，然后使用分支。

2. 对于有无符号数的减法在本质上和有无符号数的加法是一样的，只要将 B 设置为~B+1 即可。

六．附加题

1. 画出实现加/减法运算的逻辑电路原理图，并说明为什么加/减法可以只使用一个加法器进行实现？



因为加减法是对于位操作的，一个加法器就足够了。

2. 给出有符号数加/减法溢出的判断规则？

1) 如果两个数一正一负，那么结果不会溢出。
2) 如果两个数都是正数，但是结果的最高位是 1，说明产生了溢出。
3) 如果两个数都是负数，但是结果的最高位是 0，说明产生了溢出。

3. 请说明在自动化仿真测试中，如何合理地设计测试向量，选择测试向量的原则是什么？

1) 测试向量要考虑到所有的情况，特别是在进行有无符号数的加减法的时候，要着重考虑溢出的情况。

教师签字：

年　月　日