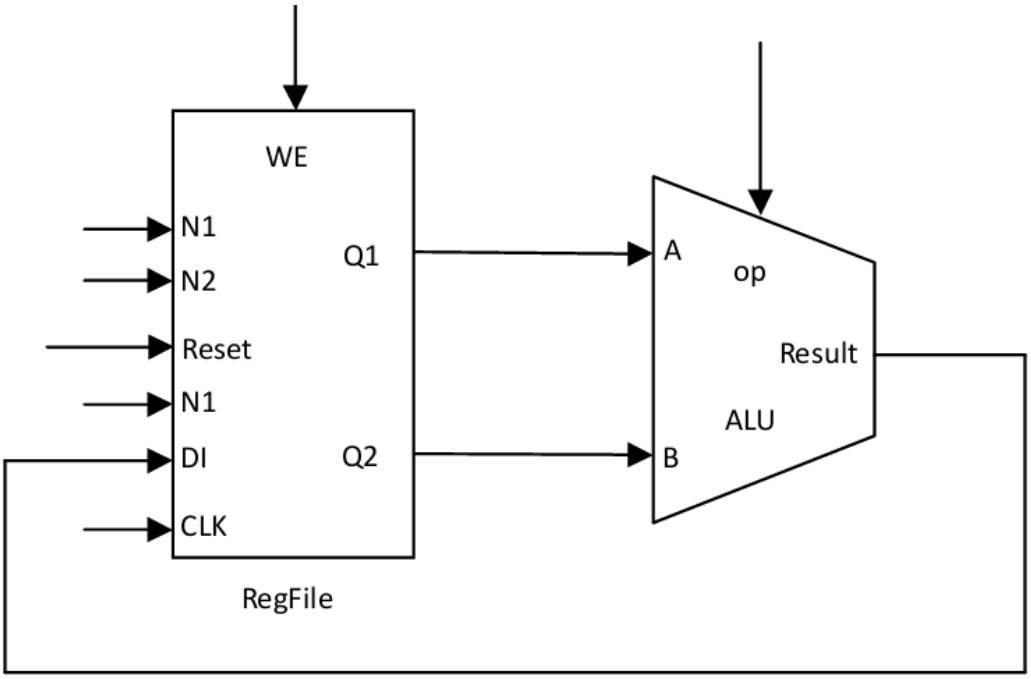
Lab3 寄存器指令设计初步

本次 lab 要求实现一个单周期 CPU 中的简单加（减/乘/取反）指令。

N1、N2 寄存器中存放的是 regfile 的地址，对加法指令 addl REG[N1], REG[N2]而言，需要将 对应信号 N1、N2 的寄存器中的 regfile 地址 Q1、Q2 取出来，进行 ALU 运算（+/-/*/~ ），并 将运算结果的值 Result 作为 DI 输入存到 N1 信号寄存器中。



注意：

1. CLK 为时钟信号，当时钟上升沿的时候读入 REG[N1]、REG[N2]（Q1、Q2 显示），当时钟下降沿的时候将 DI 写入到 REG[N1]中

2. WE 控制写入，为 1 的时候将 DI 写入到 REG[N1]中，0 的时候无操作

3. 寄存器 N1、N2 均为 5 位，Q1、Q2 为 8 位（regfile 地址本来应为 32 位，为了降低工作量这里改成 8 位），op 为 2 位

4. RegFile 里应该为每个寄存器存储一个初始值，寄存器信号为 5 位也就是说一共有 32 个寄存器，32 个寄存器里面都必须预先存有一个 8 位的数值（要求用到二维储存单元（一维数组），关于 VerilogHDL 语言中二维储存单元的构建可以参考网上资料），Reset 为 1 时初始化所有寄存器的值

5. N1、N2、op、WE、Reset 均使用 switch 开关，Q1、Q2 每个均用 2 个 7 段显示表示（0xFF ）

6. 本次 lab 同样要用到分频处理，参考如下代码考虑如何把频率降到可接受标准：

```
always@(posedge clk)
begin
  count = count + 1;
  if(count > *****)   // 这里分频需做成 2-3 秒，以便能看到 Q1 的变化，1MHz=1000000Hz
  begin
    count = 0;
    clk2 = ~clk2;
  end
end
```

应用示例：

寄存器信号从 5'b00000 到 5'b11111，寄存器中初始值都为 1：Data[reg] = 8'b00000001

我们这里选择 00101（N1）和 00110（N2）寄存器：Data[5/b00101] = 8'b00000001，Data[5'b00110] = 8'b00000001

当时钟上升沿来临时，Q1、Q2 均显示 1

设置 op 为 00（假设为加法器），则 Result 值为 8'b00000010，DI 亦为 8'b00000010 当时钟下降沿来临时，并且 WE 为 1，进行读入操作，Q1 = DI，显示为 2