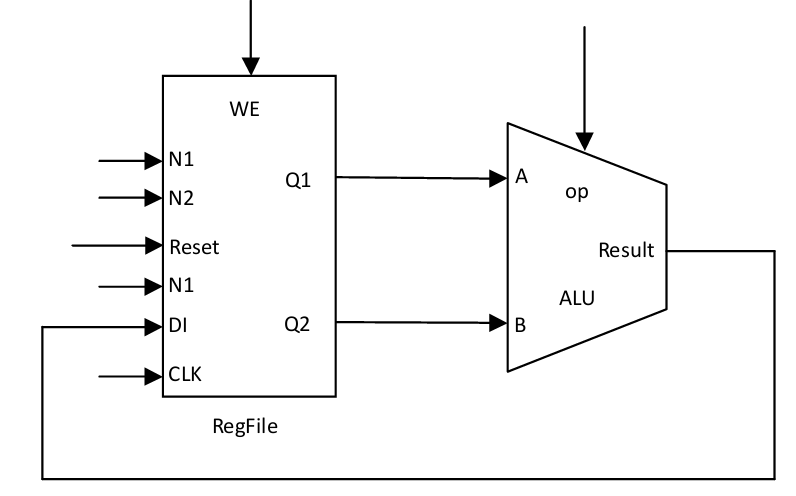
**Lab3 寄存器指令设计初步**

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