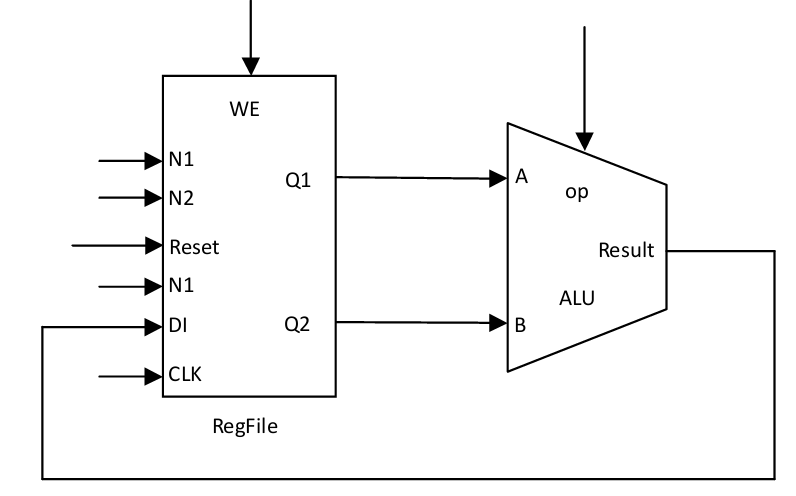
**Lab3 寄存器指令设计初步**

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

实验提交：

1. 实验报告，简要说明一下设计思路和方法
2. 实验源代码