FP16乘累加(MAC)单元设计文档

该项目以流水线形式设计了一个支持FP16的规格化数和非规格化数的乘累加单元

1. Step1：input\_check

该步骤需要对三个输入数k，x，b做处理，先分离出符号位S，指数位E(data[14:10])，尾数位M(data[9:0])，根据情况做修改。如果E=0且M=0，即0，E和M不做改动；如果E=0且M!=0，即非规格化数，则需要把指数E置为1(-14+15)；如果E!=0，即规格化数，需要在M的低11位补1.

1. Step2：mcl

该步骤即乘累加中的乘，k与x的符号位相异或，指数位相加-15，位数为相乘得到一个最多22位的新尾数M\_mcl

1. Step3：carry1

十位小数与十位小数相乘应该只有二十位小数，所以规格化的M\_mcl应该只有21位（1位整数部分和20位小数部分）。如果M\_mcl的第22位为1，则需要进位，M\_mcl右移一位，E\_mcl+1

1. Step4：compare

该步骤需要比较乘完的结果mcl和加数b的指数大小比较，比较依据有以下三点（以mcl大于等于b为目标举例）

1. Mcl的指数大于b的指数(E\_mcl > E\_b)
2. b的指数为0(E\_b == 0)，注意，这一点不可以与1）合并，因为存在这样一种情况：b是0，而mcl再step2中明显存在E\_k+E\_x-15小于0的可能，这样的情况下按照1）的判断会是b>mcl，但实际是mcl>0=b

以上三点满足一点即可判断mcl>=b，他们是或关系

判断完成后，保留较大数的符号、指数、尾数，较小数的尾数，计算较小数的尾数将要移位数(shift\_bits)，和两者进行加还是减的标志位(suboradd)。两个尾数都要想左移位直到两个尾数的最高位都是第32位为止，方便后续的对齐

1. step5：shift to align

根据上一步骤中计算出的为一数，对较小数的尾数M\_l进行相应的向左移位即可，超出21，则视为较小数的尾数过小，直接置0

1. step6：add

计算加的结果，符号位取较大数符号位，指数位取较大数指数位，根据step4中计算出的加减标志进行尾数位的加或者减

1. step7：abs

由于非规格化数的存在，step4中光比较阶数大小并不能说明绝对值大小，如果做减法有可能出现负数，这一步骤即检测负数取绝对值并符号取反。

在硬件中符号位若为1，则整个数按位取反并加1，且符号位取反

1. step7：nor\_pre

这一步骤将要根据加完后的尾数位从高往低第一个1的位置，截取出将要保留的尾数位、指数位的修正、和尾数舍入标志

保留的尾数为从高往低第一个1极其更低的10位，共11位；尾数舍入标志则是更低的1位（从高往低第一个1后的第11位）

指数位修正为从高往低第一个1的位置与32的差，第33位则+1，32位+0，31位-1，以此类推

如果从高往低第一个1的位置在第7位或者更低，一般视为0，指数、尾数都置为0

1. step8：carry2

这一步骤需要整理并考虑是否进位

如果指数小于1，则说明是一个非规格化数，按照非规格化数的处理。可供处理的指数范围在[-10:0]，比-10更小说明即使是非规格化数也是在精度之外了，将指数和尾数置0以表示0。具体为根据指数选取应保留的尾数和舍入标志。再根据舍入标志修正尾数，如果修正后尾数的第11位变成1，说明该次舍入发生了非规格化到规格化的变化，指数置1，否则指数还是0.

如果指数大于等于1，则说明是一个规格化数，根据step7中得出的舍入标志对尾数修正。如果修正后尾数第12位是1，说明发生了进位，尾数右移1位并阶数+1；否则不变。

1. step10：output

指数大于30，上溢出，置{符号位，111\_1111\_1111\_1111}，即最大可以表示的值

尾数为0，输出0x0000，即0

其余情况输出{符号位,指数[4:0],尾数[9:0]}