# FPDIVSQRT舍入情况分析

以半精度(Half-Precision)浮点数为例，其参与除法运算(如果源操作数是denormal数字，则需要先将其左移进行normalize操作)的除数和被除数的尾数形式均为:

记被除数是X, 除数是D, 商是Q, 余数是REM, 于是有:

其中M, N均为自然数.

设舍入前Q在2 ^ -10, 2 ^ -11位置的数字分别为L, G, 所有的余数组成的sticky值为S, 则需要用{L, G, S, RM, sign}来判断是否需要向上舍入

为了正确得到舍入结果，需要计算出13-bit的Q[12:0], 此时小数点的位置在Q[12]和Q[11]之间。设:

exp\_diff[6:0] = opa\_exp - opb\_exp + bias = opa[14:10] - opb[14:10] + 15

向上舍入的+1操作，可能会向2 ^ 0位置的数字产生进位(记为事件C)，此时会改变结果的指数值，下面对不同的情况进行分析。

### case[0]: X >= D, exp\_diff >= 1

如果出现事件C, 即Q[12:2] + 1'b1 = 10.0000\_0000\_00, 则说明Q[12:2] = 1.1111\_1111\_11

先考虑向上舍入条件较为宽松的Round Away From Zero(RA), 则此时必然有:

X = D \* Q[12:2] + REM, REM > 0

D = 1.0时，REM一定是0, 所以上述情况出现时，D的最小值是1.0000\_0000\_01

D \* Q[12:2] = 1.0000\_0000\_01 \* 1.1111\_1111\_11 = 10.0000\_0000\_00\_1111111111 >= 2

在Round to Nearest Even(RN)中，则要求Q[12:1] = 1.1111\_1111\_111, 此时条件比RA更严格, 同样无法被满足，因此不会出现事件C.

### case[1]: X < D, exp\_diff >= 2

与case[0]同理，假设能出现事件C, 在RA中，要求此时Q[12:1] = 0.1111\_1111\_111 = 1 - (2 ^ -11)

REM > 0时，为满足上述条件，可写出不等式:

(1 - (2 ^ -11)) \* D < X < D

(1 - (2 ^ -11)) \* (1 + N \* (2 ^ -10)) < (1 + M \* (2 ^ -10)) < 1 + N \* (2 ^ -10)

为满足右边的不等式，M能取到的最大值是(N - 1), 此时对于左边的不等式来说有:

1 + (N \* 2 ^ -10) - (2 ^ -11) - N \* (2 ^ -21) < 1 + N \* (2 ^ -10) - (2 ^ -10)

移项后可得:

(2 ^ -10) < (2 ^ -11) + N \* (2 ^ -21)

由于N <= (2 ^ 10) - 1, 因此上述不等式的右边:

(2 ^ -11) + N \* (2 ^ -21) < (2 ^ -11) + (2 ^ 10) \* (2 ^ -21) = (2 ^ -10)

矛盾，所以M取到最大值的时候无法令"(1 - (2 ^ -11)) \* D < X "成立，可知不会出现事件C, 同理RN中也不会出现事件C.

### case[2]: X >= D, exp\_diff = 0

结果是denormal数字，此时要先将Q[12:0]右移1位才能开始舍入。假设会出现事件C, 则必定有Q[12:2] = 1.1111\_1111\_11或1.1111\_1111\_10, 由case[0]

可知Q[12:2] = 1.1111\_1111\_11不可能出现，则Q[12:2] = 1.1111\_1111\_10时, 类似的将其和可能产生非0余数的D的最小值相乘，可得:

1.0000\_0000\_01 \* 1.1111\_1111\_10 = 1.1111\_1111\_11\_111111111

因此X必须大于1.1111\_1111\_11\_111111111时才可能出现事件C, 但是max(X) = 1.1111\_1111\_11, 所以上述条件不可能成立，因此事件C不可能出现。

### case[3]: X < D, exp\_diff = 1

若不看尾数相除的结果，则指数相减的结果表明商是normal数字，但是Q < 1, 因此需要先将Q左移1位再舍入，但左移操作会使normal数字变为denormal数字，因此case[3]中无需对Q进行移动，直接开始舍入。若事件C能发生，先考虑RA, 则需满足Q[12:2] = 0.1111\_1111\_11 = 1 - (2 ^ -10), 与case[1]类似, 可得不等式:

(1 - (2 ^ -10)) \* D < X < D

此不等式有且仅有一个解: M = N - 1, 即X[10:0] = D[10:0] - 11'd1, 其中D[10:0] >= 11'b1\_0000\_0000\_01

在RN中，则要求Q[12:1] = 0.1111\_1111\_111, 由case[1]可知此条件不可能满足.

因此只有舍入模式为RA且X[10:0] = D[10:0] - 11'd1时才会出现事件C.

### case[4]: X < D, exp\_diff <= 0; 或X >= D, exp\_diff <= -1

此时舍入前，经过移位之后的Q在2 ^ -1处的数字是0, 所以不可能向2 ^ 0处产生进位

### FPDIV舍入的化简方法

综上所述，事件C只可能在case[3]中出现, 而进位产生之前，舍入前的指数值必然是0, 因此可以对指数的计算逻辑进行一定程度上的化简，伪代码如下:

if(Q[12]), res\_exp\_before\_round = exp\_diff; else, res\_exp\_before\_round = exp\_diff - 1

if(res\_exp\_before\_round <= -1), res\_exp\_before\_round = 0

res\_exp\_after\_round = res\_exp\_before\_round

res\_exp\_after\_round[0] = res\_exp\_after\_round[0] | 事件C

在之前的舍入方法中，因为没有分析事件C出现的条件，所以需要进行res\_exp\_after\_round = res\_exp\_before\_round + "事件C"的计算. 上述化简操作可以省下1个5-bit的全加器, 现在的硬件开销只需要1个OR门.

### FPSQRT中的舍入问题

设SQRT计算的源操作数是opa, 则不难知道，只有opa\_exp是偶数(即opa\_exp - 15是奇数, 源操作数的指数是奇数), X[10:0] = 1.1111\_1111\_11, 并且在RA中才可能出现事件C, 此时有:

sqrt({X[10:0], 0}) = sqrt(11.1111\_1111\_10) = 1.1111\_1111\_11\_011111111111101111111...

因此计算sqrt的时候，在迭代开始之前，仅凭借X[9:0]和opa\_exp[0]的值就可以知道是否会出现事件C, 并且计算出结果的指数. 因此"res\_exp\_after\_round + 1"这个计算可以放在前处理中完成。

### **FPDIVSQRT的合并舍入**

一般的设计中都会将FPDIVSQRT写成一个模块(因为他们的实现方法都是不流水的迭代形式的SRT计算)。因此上述fpsqrt的事件C发生时的前处理中需要的指数+1操作可以和fpdiv中的前处理中需要的opa\_exp - opb\_exp共用一个加法器。在rounding时，若出现事件C，则此时需要区分是哪个操作引起的(只有fpdiv才会引起舍入后的尾数+1)，将之前的伪代码修改如下:

res\_exp\_after\_round[0] = res\_exp\_after\_round[0] | (事件C & IsFPDIV)

上述做法在省掉1个5-bit全加器的同时，也引入了以下代价:

1. 判断X[9:0] == 10'b1\_1111\_1111: 10-bit的AND门

2. res\_exp\_after\_round[0] | (事件C & IsFPDIV): 包含1个OR和1个AND门

明显上面的门电路少于5-bit全加器的门电路，那么这个优化点还是有点价值的，算是聊胜于无吧......