FP16\_div算法说明

该模块实现FP16规格化数的除法，输出规格化数

1. State0：rst

这个状态是复位状态，任何状态都可以由rst信号到达该状态，然后前进到state1：idle。**算法中不予显示**

1. State1：idle

这个状态是闲置状态，idle高电平意味着空闲可以输入，经由有效的input\_valid信号会前进到state2：input check，否则停留在该状态。**算法中不予显示**

1. State2：input check

该状态检查输入合法性并分离符号、阶数、尾数，并前进到state3：cal exp，sign

* 除数的阶数和尾数都是0（除数=0），被除数和除数有一个阶数是11111（非法输入overflow或者NaN），以上直接置overflow为1并直接跳到state7：output
* 被除数的阶数和尾数都是0（被除数=0），阶数置全0
* 其他情况正常拼接{1，低十位}作为尾数
* 符号和阶数正常取出
* 除数右移21位（低21位是0）是处于迭代算法需要
* 被除数右移10位（低10位是0）是该除法只能计算出整数商和余数，但需求是小数商而不是余数。经由将被除数乘2^10（左移十位），可以使商也变为2^10倍从而将原本小数点后十位变为可视的整数部分，达到小数商的效果

1. State3：calculate sign，exp

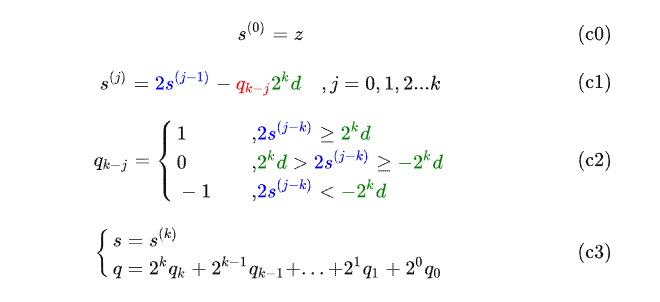
该状态计算符号位和新阶数，并前进到state4 & state5：cal rm

* 符号位异或
* 阶数相减再加bias（15）

1. State4 & State5：calculate rm

该两状态是计算尾数相除的迭代过程，迭代21次，迭代时两状态轮回，迭代完成后由state5前进到state6：normalize

其实我不懂rst除法原理，纯根据算法模拟，放一张算法图



转载自：[直观理解SRT除法，从不恢复余数除法开始！ - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/353010136)

其中z是被除数，d是除数，s是中间寄存器最后即余数，q是每次迭代的结果，k是总迭代次数在整数除法中它应该等于被除数尾数，j是当前迭代次数

根据我的尝试该图的下标有些错误，在算法中已修正

1. State6：normalize

该状态对得到的尾数商进行规格化，寻找从高到低位第一个1，然后前进到state7：output

* 由于规格化的原因，在正常计算下，最小是100\_0000\_0000 / 111\_1111\_1111，最大是111\_1111\_1111 / 100\_0000\_0000，商的第一个1只会在第十一位或者第十位出现（state2中扩大2^10倍）
* 第十位则尾数和阶数都不变
* 第九位则尾数左移一位，阶数-1
* 特殊情况，即被除数是0，商是0，尾数置0，阶数也置0

1. State7：output

该状态组合FP16规格数格式并输出，然后前进到state1：idle

* 如果overflow或者阶数大于30，则溢出（或NaN），输出{符号，0x7fff}
* 如果阶数小于1或者阶数是0，则下溢出（或就是0），输出0x0000
* 正常情况拼接{符号，阶数低5位，尾数低10位}输出