INT32转化FP16算法说明

完成从INT32到FP16规格数的转换

# step1：pre\_operation

这一阶段主要对输入进行预处理，包括找出**除标志位外**从高到低的第一位1的索引(first one index)、确定十位尾数的范围(rmcache)、根据比十位尾数更低位的内容判断舍入(ifround)、是否上溢出(ifoverflow)、是否为0(ifzero)

* 符号位直接保存为sign
* 根据FP16规划化规则，重点在于找出**除标志位外**从高到低的第一位1的位置
* 输入data\_i = 0是唯一可能的下溢出，以ifzero来标志下溢出
* FP16只有五位表示阶数，最大表示到30，而FP16表示方式本身会加15，故当阶数大于30-15=15时，就已经上溢出了，以ifoverflow来标志上溢出
* 舍入原则为舍入到最近可以表示的数，若距离一样，则偏好舍入到偶数。入的判断依据为 [第一个1的索引(first\_one\_index)后第十一位 && （第一个1的索引(first\_one\_index)后第十位 || 第一个1的索引(first\_one\_index)后第十一位以后是否还有1）]。当第一个1在11位及更低位时，FP16可以包含下尾数，不用再做舍入处理
* 在verilog中可以使用casex来判断无关项，C语言中没有这个功能，使用if-else语句以达到相同的效果

1. **step2：round to nearest even**

这一阶段根据step1的舍入判断结果，进行尾数rmcache的修正

* 若需要入，则尾数rmcache+1即可；舍则不做操作

1. **step3：overflow and carry**

step2的入操作可能导致上溢出，这一阶段判断溢出与否，并进行溢出修正

* 若尾数rmcache的第12位为1，则溢出，需要尾数右移1位，并且阶数+1来重新规格化

1. **step4：result**

这一阶段可以得出结果

* 如果标志位ifoverflow为真，说明上溢出，置0xffff；
* 如果标志位ifzero为真，说明0，置0x0000
* 如果两个标志位都不为真，但阶数大于30，则上溢出，置0xffff；
* 如果两个标志位都不为真，但阶数不大于30，则正常，拼接{符号位，阶数第五位，尾数第十位}