# IDDMM算法实现合理性分析

模乘运算：，在数论、群论、环论、代数、密码学、计算机科学等学科中都有着广泛地应用。

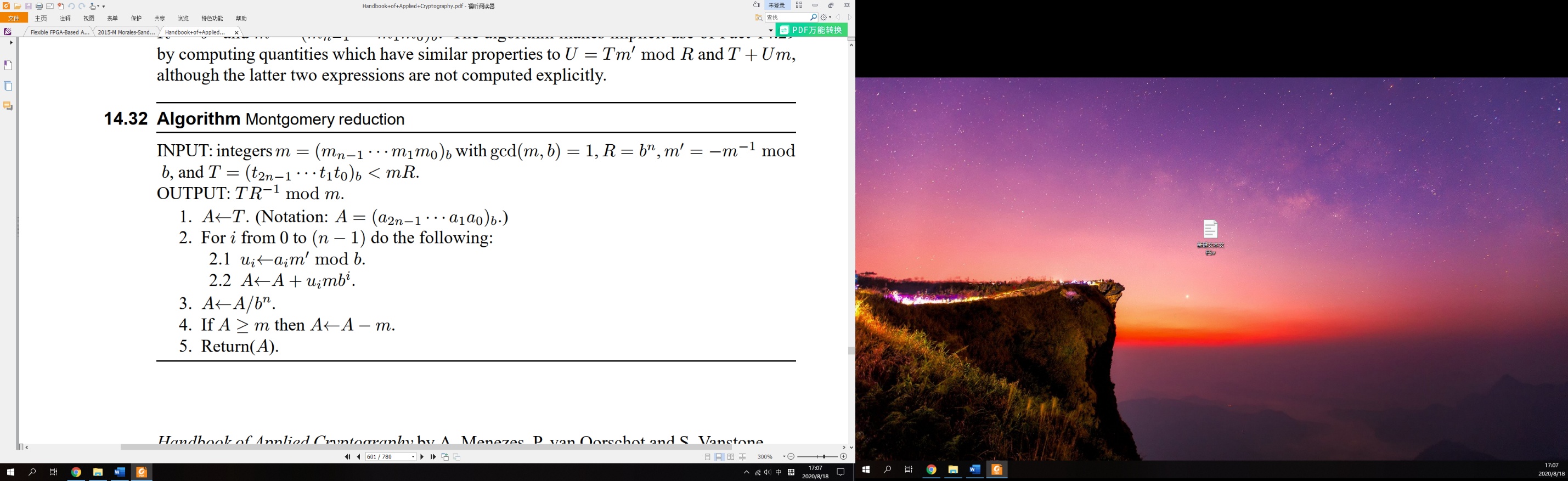
在蒙哥马利约减出现前，经典模乘算法被广泛应用。经典模乘算法使用大数计算方法，首先使用大数乘法，随后使用大数除法取余得到的结果。蒙哥马利约减是一种在不显式地执行经典模乘步骤的情况下实现模乘的一种技术，用于快速幂模运算，例如计算。蒙哥马利约减算法如下图1所示。

图1 蒙哥马利约简

假设和是在范围内的整数。让以及。的蒙哥马利约减结果是： 。考虑到计算，其中。首先计算然后计算的蒙哥马利约减，结果是：。的蒙哥马利约减结果是：最后(的蒙哥马利约减结果是,将它乘以然后模，就能得到的结果。在该算中，是一个进制长度为的整数，T也使用进制表示，的典型取值为，且。是求逆元后取负数模的结果。模值并不能取到任意值，在RSA算法中为奇数。

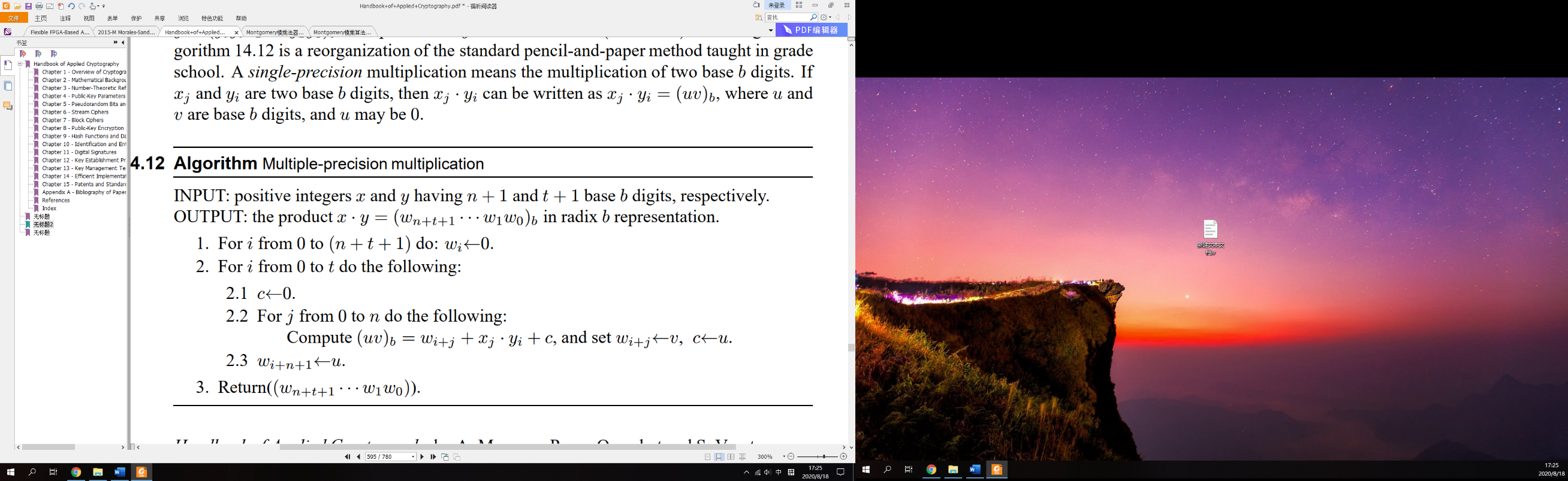
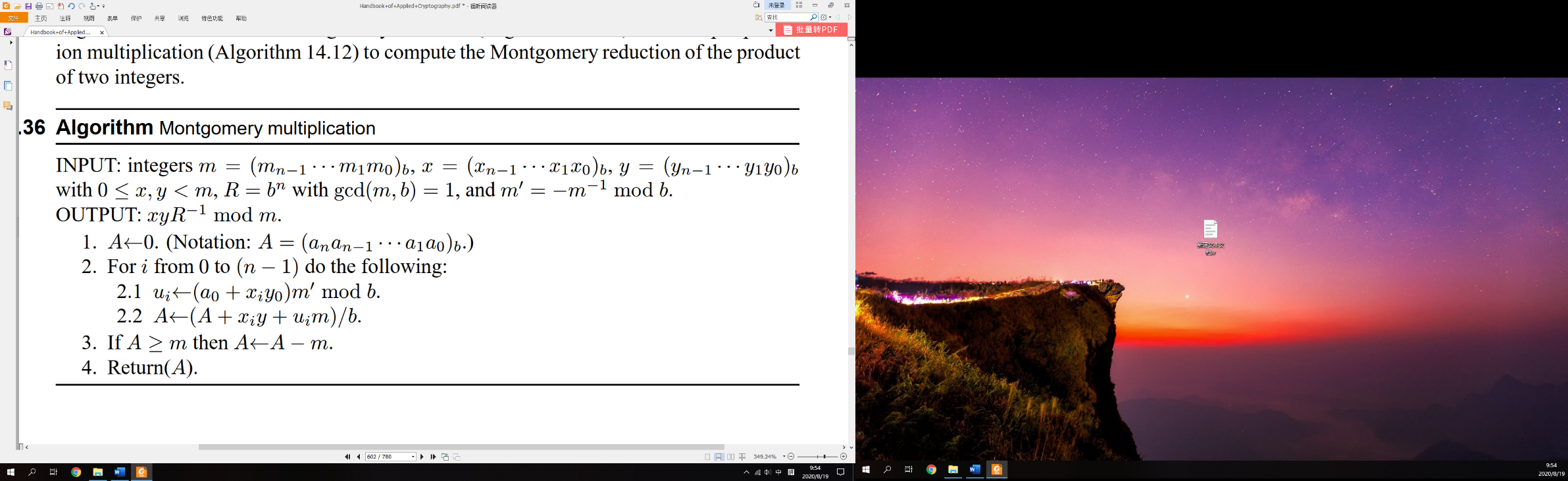
将替换为两个整数乘积后，可以使用手工乘法（中学乘法）图2，将蒙哥马利约减改写为蒙哥马利模乘，图3。

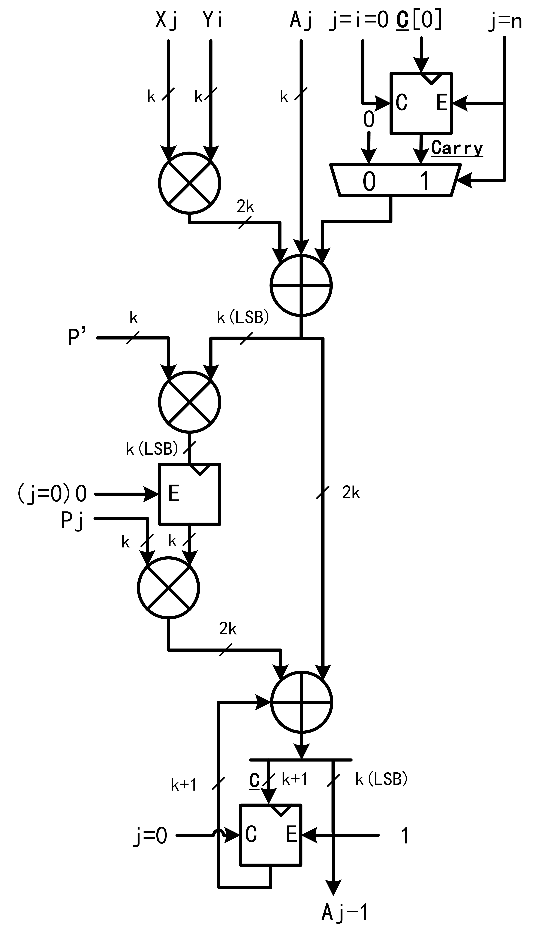
图2 手工乘法（中学乘法）

图3 Alfred J. Menezes蒙哥马利模乘

在图3蒙哥马利模乘中，是进制数的倍数，因此除以的操作可以由右移操作代替。在每次轮迭代中，和都会被计算一次，可以将展开为以下几步：

在步骤2中，结果需要模，是所有操作数的进制数，因此可以只取的最低位用于取模运算，由于小于进制数，因此模操作等效于取最低位。

步骤1和2转换为硬件电路如下图中的黄色区域所示，该部分电路仅当时启用，计算结果保存为。电路中的与上面算法中的进行了交换，电路中的仍然为上面算法中的，不影响运算结果。变量是一个内循环参量，从0到32遍历，总共遍历33次，在以上步骤中没有显示的写出。实现对的分组，用于步骤3中对的分组乘法。



在步骤3、4中，需要使用128比特乘以4096比特的乘法计算，该乘法使用遍历计算，方法是中学乘法，即相乘高位累加取和。步骤3可以改写成（步骤4也类似，这里不再给出）：

上式中，为了减少运算步骤，我们将在最高字加入第33个分组，取值设为0。其中是进制数。步骤5可以和步骤3合并：

步骤6、步骤4和式合并：

可以发现步骤3、4、5、6可以融合为上式。记，，可以遍历j计算上式：

上式中意为分别取高128位和低128位。

然后将步骤1、2、7加入到循环：

当j遍历完毕后，结果应当是，低位被写入，进位，没有被写入，在j结束后加入进行保存。

2015年，M. Morales-Sandoval根据以上部分思想，提出了Iterative digit-digit Montgomery Multiplication(IDDMM)算法。2016年，Dorian Amiet纠正了IDDMM算法中进位链错误并改用了以上运算。本设计采用了Dorian Amiet修正的IDDMM算法。

修正后的IDDMM算法如图4所示。在输入数据中，，。图4中的风别是模数和进制数。是对原数据的分组数。是的模逆元取负数后，再模的结果，该值可以预先计算，它是一个比特数。在IDDMM算法中是个位数，的最高分组位为0。该算法返回蒙哥马利模乘结果：。

在第9行中，位宽为比特，由于模值是进制数，模的结果就是后取低位，因此可以取的低位与相乘。同样技巧应用在第15、16行，这使得该算法十分适合硬件计算。通过改变分组数，IDDMM拥有灵活的缩放特性，可以设置不同的运算位宽以适应不同的硬件配置。较小的运算周期依赖于较大的运算位宽，同时会造成时序收敛困难。使用较小的运算位宽，易于时序收敛，但是运算周期会成倍增加。选取一个合适的分组数十分重要。在本设计中，输入位宽，分组数，单组比特数。算法核心部分需要经过1056次迭代。

将伪代码转换为硬件电路实现，见图5，该电路忠实的还原了IDDMM算法主循环部分的全部细节。本设计同时在了IDDMM算法Python模型的基础上进行了随机验证。验证结果复合预期。

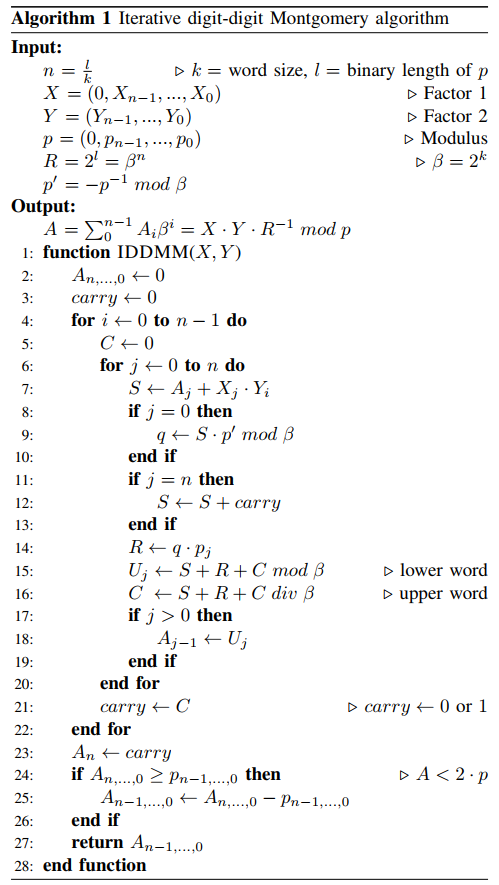


图4 本设计采用的蒙哥马利域模乘算法

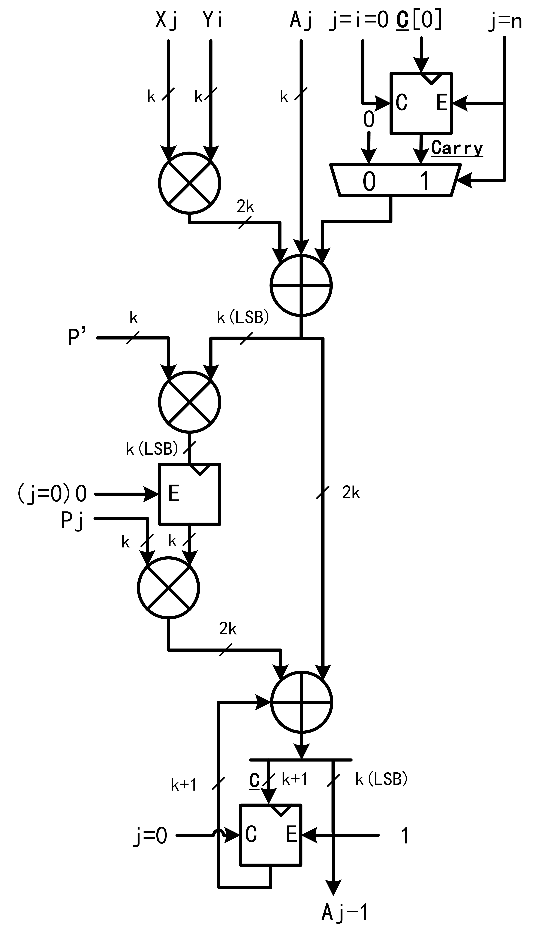


图5本设计采用的蒙哥马利域模乘算法硬件实现（不包含减法器）