

基于余数系统 RSA 密码算法快速实现

莫一奉, 李树国

(清华大学 微电子学研究所, 北京 100084)

摘 要: 本文基于余数系统实现了 RSA 密码算法. 为每一个计算单元选择形式相近的四个模, 基于这四个模的约减单元比常规方法选模对应的约减单元所需要面积更小, 延时而更短. 按照本文提出的方法选模, 选了 7 组共 28 个位宽为 74 的模, 应用这些模基于 Cox-Rower 结构设计了余数系统蒙哥马利模乘器, 实现 RSA 密码算法. 在 SMIC 0.13 μm 标准单元库下进行综合, 设计的 RSA 电路最高频率 278 MHz. 在该频率下, 中国剩余定理模式下的 1 024、2 048 位 RSA 以及常规模式下 1 024 位 RSA 解密分别需要 710 μs 、99 μs 、350 μs , 快于已发表的同类设计.

关键词: 余数系统; RSA; 快速实现; 蒙哥马利模乘器

中图分类号: TN49

文献标识码: A

文章编号: 1000-7180(2017)11-0011-05

Fast Implementation of RSA Algorithm Based on RNS

MO Yi-feng, LI Shu-guo

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: In this paper, we implement RSA cryptographic using Residue number system (RNS). We choose four modulo similar in the form for every computational unit. The reduction unit supports these four modulo is smaller and shorter than the required reduction unit corresponding to the conventional method selection. According to our proposed method, we select seven groups of modulo with a width of 74, and design a RNS Montgomery multiplier based on Cox-Rower architecture to implement RSA cryptography. In the SMIC 0.13 μm standard cell library, clock frequency reaches up to 278 MHz. At this frequency, 1 024, 2 048-bit RSA decryption in Chinese Remainder Theorem (CRT) mode and 1 024-bit RSA decryption require 710 μs , 99 μs , 350 μs , respectively, faster than the published related designs.

Key words: residue number system; RSA; fast implementation; montgomery multiplier

1 引言

随着信息网络技术的飞速发展, 信息安全变得越来越重要. 用于保证信息安全传输, 许多公钥加密算法被提出, 而 RSA^[1] 就是其中目前应用最为广泛的一种. 它不仅仅用于加密, 还能用于数字签名. 它的安全性主要基于知道两个素数的乘积而求取两个素数困难. RSA 算法核心计算是模幂操作, 而模幂又是通过重复计算模乘得到. 因此, 模乘器的设计成为实现 RSA 设计的关键.

在 RSA 中, 往往需要进行位宽很大模乘操作,

现如今主流的位宽从 1 024 到 2 048, 直接用这样大位宽的乘法直接实现模乘的是不可取的. 因为不仅需要巨大的面积, 还需要很大的功耗, 几乎已经超出了能承受的代价. 面对这个问题, 处理的方法很多, 比如用传统的高基蒙哥马利乘法、Karatusba 乘法等, 这些乘法算法可以让大位宽模乘用小位宽乘法实现, 然而目前更受关注的是余数系统. 它不仅能用小乘法实现大数模乘, 而且由于各模通道计算彼此独立、没有进位等特点, 让计算具有很高的并行性.

Kawamura 提出基于余数系统的模乘算法^[2], 它是利用蒙哥马利模乘技巧^[3]. 它需要两组模个数

相同的余数基,在两组基之间的转换需要基扩展过程.模乘的计算效率往往又跟模的选择有关系.

中国剩余定理(CRT)也可以结合余数系统来加速 RSA 模幂运算.加速的基本原理是将对一个大数的模乘操作转换为模两个小数的模乘操作.

本文针对每一个计算单元,选择了形式相近的模,相应的模约减单元比常规方法约减单元延时更短而面积更少.由于余数系统自身的优势以及我们提出的选模方法,我们实现的 RSA 要比现有文献快很多.

2 RSA 加解密算法与余数系统实现

2.1 RSA 加解密算法

RSA 算法基本参数包括私钥 $p, q, \phi(n) = (p-1)(q-1)$ 和 d , 而公钥包括 $n = pq$ 和 e . 其中 e 和 d 满足 $1 = ed \bmod \phi(n)$.

消息 m 用 RSA 加密: $c = m^e \bmod n$

解密 c 得到消息 m : $m = c^d \bmod n$

RSA 加解密算法的原理主要基于欧拉定理: $m = c^d \bmod n = m^{ed} \bmod n = m^{ed \bmod \phi(n)} \bmod n = m$. RSA 密码算法核心操作是模幂,而模幂又通过重复的计算模乘和模平方来完成.模平方实质是操作数相同的模乘.加速模幂的思路大致有两个,一个是将需要的模乘次数减少(对应 CRT 模式的使用);另一个就是加快单次模乘的计算速度(对应基于余数系统实现和本文提出特定余数基组的选择).

2.2 CRT-RSA 算法

CRT 模式下计算 RSA 模幂,能够将计算复杂减小到原来的四分之一,能够让性能得到提升,而又不需要额外的硬件资源消耗.它的基本原理大致: $m = c^d \bmod n = ((c \bmod p)^{d1} (q \bmod p)q + (c \bmod p)^{d1} (q \bmod p)q) \bmod n$. 其中的两个参数 $d1 = d \bmod (p-1)$, $d2 = d \bmod (q-1)$. 通过这样转换,就能把 $c^d \bmod n$ 的计算转换为 $(c \bmod q)^{d1}$ 和 $(c \bmod q)^{d2}$ 的操作.

2.3 余数系统蒙哥马利模乘

由于余数系统自身高度的内在并行性、无进位等特点,它能让乘法计算复杂度由二进制系统的 $O(n^2)$ 减到 $O(n)$. 而二进制模乘也是由几次乘法实现,因而,在余数系统里,模乘的计算的效率将会提高.但是在余数系统里面计算模乘,需要在两组基里面完成,相应的需要基扩展算法,基扩展算法决定整个余数系统模乘的计算效率. Bajard 提出基于混合基(MRS)实现的基扩展的蒙哥马利模乘,但是那个

算法的基扩展是按串行方式完成的,不利于发挥余数系统的并行性.在文献[3]中, Kawamura 提出了基于 CRT 的余数系统模乘算法.它的基扩展过程是基于 CRT 而实现.基于 CRT 的基扩展算法具有很高的并行性,能够让余数系统蒙哥马利模乘计算的很快.在非对称加密系统(包括椭圆曲线加密算法、ElGamal 算法等)里面,这个余数系统模乘算法的应用范围相当广泛.而实际上在这个算法里面,很多都属于常数乘法,部分乘法能够通过预计算的方式消去.正基于这种想法,在文献[4], Gandino 让整个余数系统模乘的计算复杂度减少了,提出基于原算法的改进余数系统蒙哥马利模乘.改进余数系统模乘算法一般流程:

输入: $xB_N^{-1} \bmod N$, $yB_N^{-1} \bmod N$ in B ,

$xB_N^{-1}A_j^{-1} \bmod N$, $yB_N^{-1}A_j^{-1} \bmod N$ in A

输出: $\hat{z} = xyB_N^{-1}A_j^{-1} \bmod N$ in A ,

$z = xB_N^{-1} \bmod N$ in B

1: $s = (xB_N^{-1}) * (yB_N^{-1})$ in B

2: $t = (xB_N^{-1}A_j^{-1}) * (yB_N^{-1}A_j^{-1})$ in A

3: $\hat{z}$ in $A \leftarrow BE1(s \text{ in } B, t \text{ in } A)$

4: z in $B \leftarrow BE2(\hat{z} \text{ in } A)$

改进的余数系统蒙哥马利模乘需要 $2n^2 + 4n$ 次乘法,相对应最初的蒙哥马利模乘少 $3n$ 次乘法,也是目前需要乘法次数最少的余数系统模乘算法,因此本文的设计使用这个算法.

2.4 Cox-Rower 模乘计算结构

余数系统蒙哥马利模乘算法的经典硬件结构是图 1 所示 Cox-Rower 结构.其中 Cox 是控制单元,由一个加法器便能实现.而 Rower 是计算单元,一般可以分为两个部分,一个计算乘加,另一个部分就是将结果进行约减.约减方法往往和余数基的选择有关系,现有文献里,最为常用的余数基形式是 $2^r - k$ ($r < 2 \lceil r/2 \rceil$) 的伪梅森素数,它的约减过程需要两次乘法和三次加法.本文通过对每一个 Rower 单元选择形式相近的四个模,从而让模约减过程变得简单,加快模乘的计算速度.

3 特定余数基组的快速模约减

余数基的选取对加速余数系统蒙哥马利模乘十分重要.针对需要余数基较多的应用场合,形式如 $2^r - k$ ($k < 2 \lceil r/2 \rceil$) 伪梅森素数的模经常被选取.和一般余数基相比,这种模主要需要一次 $r * w_k$ 和一次 $w_k * w_k$ 共两次乘法,而一般的需要两次 $r * r$ 乘法,因此有了很大改进.

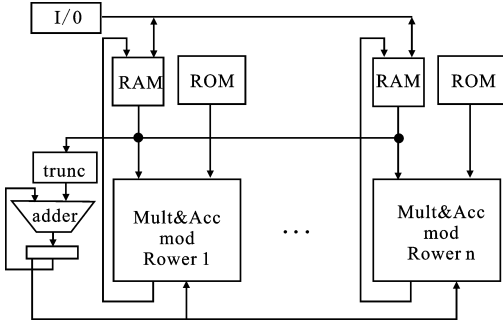


图 1 Cox-Rower 一般结构

3.1 余数基组余数基组形式的选取

本文文化简模约减单元的思路:对每一个 Rower 单元选择 $\{2^r - 2^{t_1} - 2^{t_2} \pm 1, 2^r - 2^{t_1} \pm 1\}$ 四个形式相似的模,对于每一个 Rower 单元 t_1 和 t_2 都是常数,那么它的约减过程只需要拼接和若干次加法就能实现,而其中拼接几乎没有延时和面积消耗的。

这四个模用如下统一形式进行表示

$$M(s_1, s_2) = 2^r - 2^{t_1} - BA(s_1, 2^{t_2}) - BX(s_2, 1) - s_2$$

$$= \begin{cases} 2^r - 2^{t_1} - 1 & s_1 = 0, s_2 = 0 \\ 2^r - 2^{t_1} + 1 & s_1 = 0, s_2 = 1 \\ 2^r - 2^{t_1} - 2^{t_2} - 1 & s_1 = 1, s_2 = 0 \\ 2^r - 2^{t_1} - 2^{t_2} + 1 & s_1 = 1, s_2 = 1 \end{cases}$$

式中, $BA(s_1, 2^{t_2})$ 可以用若干个与门来控制 2^{t_2} 这一项有或者无, $BX(s_2, 1) + s_2$ 是一种补码表示形式,用若干个异或门就能实现这个功能。

3.2 余数基组的快速模约减

特定余数基的约减过程大致如下描述。

输入: $m = 2^r - 2^{t_1} - BA(s_1, 2^{t_2}) - BX(s_2, 1) - s_2$,
 $d, d < 2^r$

输出: $z = d \bmod m, z < m$

1: $T_1 = d[r-1:0] + BA(s_1, \{d[2r-1:r], \{t_2\}\{1d0\}\}) + \{d[2r-1:r], \{t_1\}\{1d0\}\} + s_2$
 $+ BX(d[2r-1:r], s_2)$

2: $T_2 = T_1[r-1:0] + BA(s_1, \{T_1[\frac{3}{2}r-1:r], \{t_2\}\{1d0\}\}) + \{T_1[\frac{3}{2}r-1:r], \{t_1\}\{1d0\}\}$

$+ BX(T_1[\frac{3}{2}r-1:r], s_2) + s_2 + T_1[r-1:0] + BA(s_1, \{T_1[\frac{3}{2}r-1:r], \{t_2\}\{1d0\}\})$

3: $T_3 = T_2 + 2^{t_1} + BA(s_1, 2^{t_2}) + BX(s_2, 1) + s_2$

4: if ($T_3[r]$)

$z = T_3[r-1:0]$

else

$z = T_2[r-1:0]$

和一般形式模约减过程相比,常规约减过程中的两次乘法被少量的几次加法代替,从而关键路径的延时变短,资源消耗也变少。

4 余数系统实现 RSA

余数系统实现 RSA 的核心在模乘器单元的设计。而模乘计算速度往往决定于 Rower 单元的设计。本文针对每一个 Rower 单元选择如 3.2 节里面形式相似的四个模,从而能够让约减过程变得简单。

4.1 余数基组

本文选择了 7 组形式如 3.2 里面的模,对应 7 个 rower 单元。第一个 Rower 单元的模: $2^{74} - 2^1 + 1, 2^{74} - 2^1 - 1, 2^{74} - 2^1 - 2^3 - 1, 2^{74} - 2^1 - 2^3 + 1$; 第二个 Rower 单元的模: $2^{74} - 2^6 + 1, 2^{74} - 2^6 - 1, 2^{74} - 2^6 - 2^6 - 1, 2^{74} - 2^6 - 2^6 + 1$; 第三个 Rower 单元的模: $2^{74} - 2^8 + 1, 2^{74} - 2^8 - 1, 2^{74} - 2^8 - 2^4 - 1, 2^{74} - 2^8 - 2^4 + 1$; 第四个 Rower 单元的模: $2^{74} - 2^{10} + 1, 2^{74} - 2^{10} - 1, 2^{74} - 2^{10} - 2^4 - 1, 2^{74} - 2^{10} - 2^4 + 1$; 第五个 Rower 单元的模: $2^{74} - 2^{14} + 1, 2^{74} - 2^{14} - 1, 2^{74} - 2^{14} - 2^4 - 1, 2^{74} - 2^{14} - 2^4 + 1$; 第六个 Rower 单元的模: $2^{74} - 2^{20} + 1, 2^{74} - 2^{20} - 1, 2^{74} - 2^{20} - 2^{12} - 1, 2^{74} - 2^{20} - 2^{12} + 1$; 第七个 Rower 单元的模: $2^{74} - 2^{22} + 1, 2^{74} - 2^{22} - 1, 2^{74} - 2^{22} - 2^4 - 1, 2^{74} - 2^{22} - 2^4 + 1$ 。

我们设计选择的 7 组模,每一个模的位宽 74 位,单组 14 个模的动态范围是 1 036 位,能够满足 1 024 位模乘动态范围的要求。在 CRT 模式下,完成 2 048 位的 RSA,需要计算 1 024 位的模乘,从而需要所有的模,而完成 1 024 位的 RSA,则只需要计算 512 的模乘,14 个模就能构建余数系统蒙哥马利模乘的两组余数基。

4.2 提出的模组合的模约减单元

约减单元如图 2 表示,其中,bit org 提供输入连续若干比特和 0 的组合,比如 $\{d[10+r:r], 10^r d0\}$ 。

一般形式模的约减单元相比,四个特定模对应的约减过程也需要三次约减过程,不同的是常规约减单元里面的乘法被我们用少量几次加法所代替,由于加法器要比同位宽的乘法器面积要小、延时更短,因此,计算起来更快。但是我们的这种设计缺点是同一个 Rower 单元只能完成针对特定四个模的约减过程,然而实际设计里面需要多个 Rower

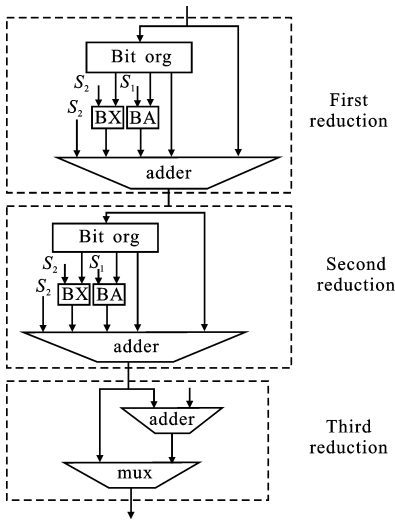


图 2 余数基组的三次约减结构

单元,只要每个 Rower 单元能够完成特定几个模的模乘,所有的模对应模乘就能都能够完成了.其实让所有的 Rower 单元能完成所有模的约减过程是完全没有必要,还会造成很多额外不必要的开销.

4.3 两级流水 Rower 结构

经过综合发现,我们的约减单元路径延时比 72 位的乘法器要小,因此直接让约减单元一个周期完成,图 3 为我们设计的两级流水 Rower 单元. 每一个 Rower 单元两个 1 比特输入 s_1 和 s_2 四种取值分别对应一个模的约减过程.

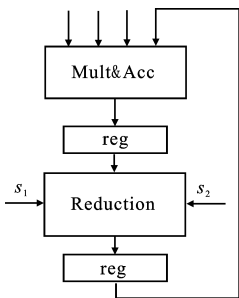


图 3 两级流水 Rower 结构

4.4 CRT-RSA 模幂的调度

用中国剩余定理计算 $\text{mod } n$ 的操作变成 $\text{mod } p$ 和 $\text{mod } q$. 这样就将一个 2 048 的模乘就转换为了两个 1 024 的模乘. 这两组模乘在整个模幂计算过程中彼此独立,没有数据相关关系,可以用同一组余数基完成. 不同的是计算余数系统蒙哥马利模乘过程中涉及到与模操作的模数(p 或者 q)的预计算有所不同而已.

我们选择的用于计算模乘的 Rower 单元有 7 个,而对于蒙哥马利模乘单组余数基模的个数为

14、流水线的级数为两级,因此在计算模乘的时候有两种选择:第一种是 $\text{mod } p$ 和 $\text{mod } q$ 两组计算同时进行,两级流水结构里面轮流进行 $\text{mod } p$ 和 $\text{mod } q$ 的计算;第二种就是计算完 $\text{mod } p$ 后再计算 $\text{mod } q$. 但考虑到 1 024 位的模乘,余数系统蒙哥马利模乘两组模每组只需要 7 个,如果 $\text{mod } p$ 和 $\text{mod } q$ 的计算单独进行,会导致流水线利用不充分. 为了充分利用资源,只能采用第一种方案,即 $\text{mod } p$ 和 $\text{mod } q$ 的操作同时进行计算. 这样能确保计算 2 048 和 1 024 位 RSA 的时候,模乘器利用率都为百分之百,同时内部调度还会简单,因为我们只需要 $\text{mod } p$ 和 $\text{mod } q$ 所需要乘法依次同步的放入流水线,而不需要考虑数据相关问题.

4.5 功能验证

我们将提出的结构用 Verilog HDL 进行了实现. 并用 modelsim 仿真进行功能验证. 用 miracle c 产生测试向量作为 Verilog 的输入,计算结果和 miracle c 直接的计算结果进行比较. 通过大量测试向量测试,本设计正确.

4.6 综合结果

我们的设计使用了 SMIC 0.13 μm 工艺,在 synopsy 公司的 Design Comiler 上进行综合. 为了对比我们提出约减结合相对常规选法的优势,我们常规的约减电路也进行实现综合和我们的约减电路进行对比. 我们用 miracle c 寻找 28 个形式如 $2^{74} - k$ 伪梅森素数的模,结果 k 为 11 比特. 将相应的约减单元用 Verilog HDL 实现 综合,结果如表 1. 按我们提出的选模方式,单个模约减单元延时上要少 1.1 ns,而面积上少了 6 kG,整个设计需要这样的 7 组模约减单元,那么整个设计中的约减单元面积大致减少了 42 kG.

表 1 本文选模和常规选模的对比

模的选法	工艺/ μm	延时/ns	面积/kG
本文选法	0.13	3.4	8
常规选法	0.13	4.5	14

我们的设计 RSA 电路在 SMIC 0.13 μm 工艺上面进行综合,电路最高频率为 278 MHz,在 SMIC 0.18 μm 工艺上面进行综合,电路最高频率为 147 MHz. SMIC 0.13 μm 工艺下,CRT 模式 2 048、1 024 位的 RSA 以及非 CRT 模式的 1 024 RSA 一次解密分别需要 710 μs 、99 μs 和 350 μs . 非 CRT 模式的 RSA 实验结果及对比放在表 2,而 CRT 模式的 RSA 实验结果以及对比放在表 3 中. 从表中可以看

出,我们的设计解密所需要的时间快于参考文献.

表 2 RSA 实验结果及比较

设计	工艺/ μm	位宽	解密/ μs	面积/kG
本文	0.13	1 024	350	423
文献[5]	0.25	1 024	4 200	221
文献[8]	0.18	1 024	1 440	255
本文	0.18	1 024	660	410
文献[9]	0.13	1 024	4 100	139

表 3 CRT-RSA 实验结果及比较

设计	工艺/ μm	位宽	解密/ μs	面积/kG
本文	0.13	2 048	710	423
本文	0.13	1 024	99	423
文献[5]	0.25	1 024	2 400	221
文献[5]	0.25	2 048	8 900	221
文献[6]	0.09	1 024	2 380	14
文献[7]	0.18	1 024	1 670	200
本文	0.18	1 024	187	410
文献[8]	0.18	2 048	2 050	395
本文	0.18	2 048	1 341	410

5 结束语

本文基于余数系统实现 1 024、2 048 位 CRT 模式下 RSA 以及 1 024 位非 CRT 模式的 RSA. 通过对每个计算单元选择形式相近的模、让模约减计算单元比常规选择方法延时路径更短、面积消耗更小. 我们的设计用 SMIC 0.13 μm 工艺综合. 其结果为在 CRT 模式下 2 048 位和 1 024 位的 RSA,一次解密需要 710 μs 和 99 μs ,而非 CRT 模式的 1 024 解密需要 350 μs ,比现有公开文献的实现速度更快.

参考文献:

[1] Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems [J]. Communications of the ACM, 1978, 21(2): 120-126.

[2] Kawamura S, Koike M, Sano F, et al. Cox-rower architecture for fast parallel montgomery multiplication

[C]// Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques. Bruges, Belgium, 2000:523-538.

[3] Montgomery P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44 (17):519-521.

[4] Gandino F, Lamberti F, Paravati G, et al. An algorithmic and architectural study on Montgomery exponentiation in RNS[J]. IEEE Transactions on Computers, 2012, 61(8): 1071-1083.

[5] Nozaki H, Motoyama M, Shimbo A, et al. Implementation of RSA algorithm based on RNS montgomery multiplication[C]//International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2001: 364-376.

[6] Néto J C, Tenca A F, Ruggiero W V. CRT RSA decryption: modular exponentiation based solely on montgomery multiplication[C]//Signals, Systems and Computers, 2015 49th Asilomar Conference on. Pacific Grove, CA, USA,IEEE, 2015: 431-436.

[7] 麻永新, 曾晓洋, 吴敏,等. 基于 Barrett 模乘算法的 RSA 密码协处理器设计[J]. 系统工程与电子技术, 2006, 28(6):830-833.

[8] Tao W U, Shuguo L I, Liu L T. Fast RSA decryption through high-radix scalable montgomery modular multipliers[J]. Science China Information Sciences, 2015, 58(6):1-16.

[9] Shieh M D, Chen J H, Wu H H, et al. A new modular exponentiation architecture for efficient design of RSA cryptosystem[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16 (9): 1151-1161.

作者简介:

莫一奉 男,(1990-),硕士研究生. 研究方向为信息安全算法的大规模数字集成电路设计与实现.
E-mail:873087176@qq.com.

李树国 男,(1963-),教授,博士生导师. 研究方向为信息安全算法的大规模数字集成电路设计与实现.