

智能卡中非对称算法模幂运算单元的设计与验证

邵佳佳, 乌力吉, 张向民

(清华大学 微电子学研究所, 北京 100084)

摘 要: 基于 CIOS 模乘算法和 Montgomery powering ladder 模幂算法, 设计了模幂电路, 能进行密钥长度可配置的模乘、模幂、模加和模减运算. 在 50 MHz 和 SIMC13LL 工艺库条件下对模幂电路进行 DC 综合, 面积为 16.1 千门, 吞吐率为 10 kb/s, 30 MHz 下 Primetime PX 分析功耗 2.87 mW, 满足智能卡对速度、功耗、面积等指标需求. 与近几年文献中的 RSA 模乘器相比, 所设计的模乘电路具有最小的“功耗×面积/吞吐率”, 资源利用率高.

关键词: 智能卡; 蒙哥马利模乘; 模幂; 非对称算法

中图分类号: TN43

文献标识码: A

文章编号: 1000-7180(2015)02-0037-05

Design and Implementation of Long Integer Modular Exponentiation Unit of Asymmetric Encryption in Smart Card

SHAO Jia-jia, WU Li-ji, ZHANG Xiang-min

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

Abstract: In this paper, we design a key-length-configurable modular exponentiation based on CIOS modular multiplication algorithm and Montgomery powering ladder exponentiation algorithm. The long integer modular exponentiation circuit is synthesized by Synopsys Design Compiler under the clock frequency of 50 MHz and SMIC13LL technology at a cost of 16.1k gates and throughput is 10 kb/s and Primetime PX power report shows that the average power consumption of 30 MHz is 2.87 mW, which achieves the requirements of speed, power consumption and circuit area of smart card. The modular multiplication circuit proposed by this thesis achieves high resource utilization rate and smallest "power×area/throughput" in comparison with modular multiplication of systolic array architectures.

Key words: smart card; Montgomery modular multiplication; modular exponentiation; asymmetric encryption

1 引言

在非对称加密算法中, 模幂运算是核心运算单元. 智能卡在面积、速度、功耗等方面的资源有限且要求具备高安全性和可靠性. RSA 算法包含模幂运算, ECC、SM2 还包含模加模减运算. 模幂运算可以拆分为多次模乘和模平方运算, 模乘算法一般采用蒙哥马利模乘, 转化大数乘法为多精度乘法, 多精度加法和移位操作.

2 CIOS 和 FIOS 模乘

蒙哥马利模乘^[1]的参数设定: N 为 k bit 的模数,

$2^{k-1} \leq N < 2^k$, A, B 为 k bit 相乘的大数, $0 < A, B < N$, 模乘结果 $T = \text{Mont}(A, B, N) = A \times B \times R^{-1} \bmod N$, 其中 $R \times R^{-1} - N \times N = 1$. 将大数分割成 s 个字, 假设 w 是字长, 基 $W = 2^w$, 本文中取 $w = 32$, 则基 $W = 2^{32}$. 那么大数 A 可以写成以下形式: $A = a[s-1]W^{s-1} + a[s-2]W^{s-2} + \dots + a[0]W^0$. 为了方便, 直接写成 $A = (a[s-1], a[s-2], \dots, a[0])$. CIOS 模乘算法如下所示^[1-2].

输入数据: $A, B, N, n[0]'$

输出数据: $T = \text{Mont}(A, B, N) = A \times B \times R^{-1} \bmod N$

for $i = 0$ to $s-1$

```

{C = 0
/* * * * * inner loop1 * * * * */
for j = 0 to s - 1
    {(C, S) = t[j] + a[j] × b[i] + C
t[j] = S}
(C, S) = t[s] + C
t[s] = S
t[s + 1] = C
m = t[0] × n[0]' mod W
(C, S) = t[0] + m × n[0]
/* * * * * inner loop2 * * * * */
for j = 1 to s - 1
    {(C, S) = t[j] + m × n[j] + C
t[j - 1] = S}
(C, S) = t[s] + C
t[s - 1] = S
t[s] = t[s + 1] + C}
/* * * * * big number subtraction * * * * */
for i = 0 to s
    u[i] = t[i]
carry = 0
for i = 0 to s - 1
    {(carry, sum) = u[i] - n[i] - carry
t[i] = sum}
(carry, sum) = u[s] - carry
t[s] = sum
if carry = 0 return T
else return U

```

FIOS 模乘算法如下所示^[1-2].

输入数据: $A, B, N, n[0]'$

输出数据: $T = \text{Mont}(A, B, N) = A \times B \times R^{-1} \bmod N$

```

for i = 0 to s - 1
    {CA = 0, CB = 0
(CA, S) = t[0] + a[0] × b[i]
t[0] = S
m = S × n[0]' mod W
(CB, S) = t[0] + m × n[0]
/* * * * * inner loop * * * * */
for j = 1 to s - 1
    {(CA, S) = t[j] + a[j] × b[i] + CA
(CB, S) = S + m × n[j] + CB
t[j - 1] = S}
(S, CA) = CA + CB + t[s]
t[s - 1] = S
t[s] = CA}
/* * * * * big number subtraction * * * * */
同 CIOS 模乘算法的 big number subtraction 部分

```

2.1 大数减法优化

模数乘法的精简形成如下所示.

输入数据: $A, B, N, n[0]'$

输出数据: $T = \text{Mont}(A, B, N) = A \times B \times R^{-1} \bmod N$

```

T = 0
for i = 0 to s - 1
    {q[i] = (t[0] + a[i] × b[0]) × n[0]' mod W
T = (T + a[i] × B + q[i] × N) / W}
if T > N, then T = T - N
else return T

```

(1) 推导模乘每次循环的中间结果 $T < N + B$.

算法中 $T = (T + a[i] \times B + q[i] \times N) / W$, 因为 $0 < a[i], q[i] \leq 2^w - 1$, 所以 $T \leq [N + B + (2^w - 1) \times B + (2^w - 1) \times N] / W = [2^w \times B + 2^w \times N] / W < N + B$.

(2) 推导改进的模乘结果 $T < 2N$.

因为 $0 < A < 2N$, 所以 $a[s - 1] = 1$ 或 0 , 从 CIOS 模乘的最后一步开始推导

$$\begin{aligned}
 T' &= (T + a[s - 1] \times B + q[s - 1] \times N) / W \\
 &\leq (T + B + q[s - 1] \times N) / W \\
 &\leq (T + B + (2^w - 1) \times N) / W \\
 &< [N + B + B + (2^w - 1) \times N] / W \\
 &< (2B + 2^w \times N) / W \\
 &= 2B / W + N \\
 &< 4N / 2^w + N \\
 &< 2N
 \end{aligned}$$

(3) 推导模幂最后一步脱离蒙哥马利数系 $C < N$. 在模幂的最后一步计算 $C = \text{Mont}(T, 1, N)$ 的最后一轮循环开始推导, 此时 $B = 1$.

$$\begin{aligned}
 C &= [T + a[s - 1] \times B + q[s - 1] \times N] / W \\
 &\leq [N + B + 1 \times B + (2^w - 1) \times N] / W \\
 &= [N + 1 + 1 \times 1 + (2^w - 1) \times N] / W \\
 &= N + 2 / W
 \end{aligned}$$

因为 $w > 2$ 很容易满足, 在本文中 $w = 32$, 所以 $C \leq N$. 假设 $C = N$, 即 $M^E \bmod N = N$, 那么 $M = 0$ 或 N , 在 RSA 中一般 $M < N$, 所以假设不成立, 因此 $C < N$ ^[3].

将大数减法约简的思想运用到 CIOS 和 FIOS 中, 相当于将 1024 bit 的模乘扩展到 1056 bit. 付出的代价: ①每个大数的存储空间增加 32 bit; ②内循环和外循环的计数器上限值增加 1. 得到的性能提升: ①减少了控制流的状态数; ②在硬件规模上也减少了数据通路和存储器地址生成的部分控制逻辑. 一般 A, B, N 存储在外部 SRAM 中, T 存储在模乘器内部的 register file 中. 对于模乘器来说只需考虑内部 register file 的存储资源消耗, 即增加 32 bit 的存储空间, 且运算速度相差不大, 所以这样的优化对于模乘器来说是值得的.

2.2 存储器方案

CIOS 算法结构为一个外循环和两个内循环。在两个内循环中,要完成 1 次乘加操作和 1 次 register file 读写,乘加过程中既要写入该次运算结果又要读取下一个数据,所以采用双口的 register file,这样完成一次模乘需要对 register file 整体读写 $2s^2 + 3s$ 遍。

FIOS 合并了 CIOS 的两个内循环,在一次内循环完成 2 次乘加运算和 1 次 register file 读写,这样在进行第一次乘加操作时,就能读取 register file 中的下一个数据,第二次乘加操作时写入运算结果,所以采用单口的 register file,完成一次模乘只要对 register file 整体读写 $s^2 + 2s$ 遍。

2.3 流水线

在模乘算法的内循环数据通路中插入乘法结果寄存器 mul_result_reg 和加法结果寄存器 add_result_reg,将每个时钟加法和乘法结果都存入相应寄存器中,接着下一轮计算的输入从这两个寄存器中导出。内循环结束时,内部流水线也相应结束。在流水线的结构下,模乘器的关键路径是图 1 中粗线部分所呈现,是 32 bit 的乘法器。

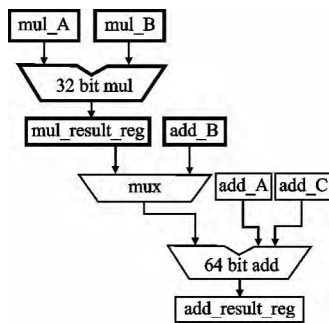


图 1 模乘器的数据通路和关键路径

2.4 功耗分析

本文针对两种模乘器用 Prime Time PX 基于时间模型来做综合后功耗分析。从平均功耗中可以看出,CIOS 模乘器平均功耗小于 FIOS 模乘器。其中,CIOS 模乘器中存储器功耗和组合逻辑的功耗相当,各占 45% 左右,时序逻辑功耗小于 10%,不予考虑。而 FIOS 模乘器,存储器功耗小于 20%,组合逻辑功耗在 70% 左右。在算法分析中,FIOS 相比较于 CIOS 减少了一半的存储器访问量,因此能够解释表中的两种模乘器功耗分布的不一致。

截取一次 1024 bit 模乘的时间段,进行基于时间点的功耗分析,产生相应的波形文件,将其放入 debussy 中观察,能够看到整个运过程中的总体及

各模块的功耗。

用相同的测试向量对 CIOS 模乘器与 FIOS 模乘器进行仿真,得到对应功耗曲线图。观察功耗曲线图可以很明显看出:(1) FIOS 的模乘器平均功耗大于 CIOS 功耗;(2) FIOS 上的乘法器功耗远大于 CIOS 乘法功耗,且 FIOS 乘法器功耗曲线图与 CIOS 乘法器功耗曲线形状有着明显的不同。

图 2 和图 3 分别表示 FIOS 与 CIOS 算法内部循环调用乘法器相关代码。从上述代码中,可以很明显看出 CIOS 是将 FIOS 内部一个循环拆成两个并列的循环。在 CIOS 模乘器中,两个内循环 inner_loop1 中和 inner_loop2 中每次 32 bit 乘法中,只有一个乘数在变。而在 FIOS 乘法器部分,每轮内循环,两个乘数都同时发生改变。因此 CIOS 乘法器内部翻转率较低,乘法器功耗较小。而 FIOS 乘法器运算中的内部翻转率较高,乘法器功耗也就很大。在 CIOS 模乘器的乘法功耗曲线中正好出现了 64 个尖峰,和 CIOS 算法内部循环次数吻合。而在 FIOS 模乘器中,内循环中包含两次乘加运算,即每次乘法运算内部翻转率都较高,所以功耗曲线看上去呈现平稳的高水平,相应的平均功耗也较高。

```

/*****fios_inner_loop*****/
for j = 1 to s-1
  {(CA,S) = t[j]+a[j] × b[j]+CA
  (CB,S) = S+m × n[j]+CB
  t[j-1] = S}

```

图 2 FIOS 内部循环调用乘法器部分

```

/*****cios_inner_loop1*****/
for j = 0 to s-1
  {(C,S) = t[j]+a[j] × b[j]+C
  t[j] = S}

/*****cios_inner_loop1*****/
for j = 1 to s-1
  {(C,S) = t[j]+m × n[j]+C
  t[j-1] = S}

```

图 3 CIOS 内部循环中调用乘法器部分

综上,尽管 FIOS 模乘器中存储器访问次数要低于 CIOS 模乘器,但 FIOS 中乘法器功耗要远大于 CIOS 模乘器功耗。因此总体的平均功耗 FIOS 仍然高于 CIOS 模乘器。最终我们选择 CIOS 作为模乘器方案。

3 MontgomeryPowering Ladder 模幂

MontgomeryPowering Ladder 模幂算法如下所示。

输入数据: $M, E, N, n[0]', R_{\text{mod_}N}, R2_{\text{mod_}N}$
 输出数据: $C = M^E \bmod N$

```

 $R[0] = R_{\text{mod\_}N}, R[1] = \text{Mont}(M, R2_{\text{mod\_}N}, N)$ 
for  $i = k-1$  to 0
     $R[1 - e[i]] = \text{Mont}(R[1 - e[i]], R[e[i]], N)$ 
     $R[e[i]] = \text{Mont}(R[e[i]], R[e[i]], N)$ 
return  $C = \text{Mont}(R[0], 1, N)$ 
  
```

3.1 电路结构

模幂电路的结构分为模幂运算控制部分, 模乘顶层模块和 SRAM 开关. 模幂运算控制部分接收外部信号 cmd 来判断完成何种运算, 可单独调用模乘顶层模块完成模乘、模加、模减运算或者频繁调用模乘顶层来完成模幂运算. SRAM 开关受模幂运算模块控制来进行载入密钥或数据读写操作. 模幂运算控制部分含有地址生成单元的组逻辑电路, 产生每次模乘顶层的地址输入. 模幂电路中使用 32 bit 的寄存器 load_key32 来存储从 SRAM 中读取到的部分密钥.

模乘单元能够进行蒙哥马利模乘, 模加模减运算. 其中包含控制流、数据流、存储器地址信号生成模块、register file、乘法器、加法器等运算单元. 控制流控制整个模乘、模加和模减的操作, 控制流部分有三个重要信号: state 信号、循环变量 i 和 j , 通过这三个信号能够确定该时钟周期应该完成的乘法或加法运算和存储器的读写操作.

3.2 安全性分析

现今智能卡追求的不仅仅是性能卓越, 随着侧信道攻击的出现, 安全算法和抗侧信道措施也迫切需求, 这往往伴随着性能的牺牲. 在原始的模乘算法中, 最后一步的条件分支来决定是否舍弃减法结果, 如果结果需要保留, 那么需要搬运结果到外部存储器中. 为了防止时间攻击, 在不需搬运数据的情况下, 则需要增加空循环来填补这部分时间差, 而这两个操作的差异能被功耗攻击识别出来. 在改进的模乘算法中, 由于没有条件分支, 就增强了模乘器的抗时间攻击和功耗攻击的能力. 虽然恢复出 RSA 加密密钥主要集中在对模幂算法的攻击, 但是在模乘层面能够越少的泄露信息, 使每次模乘的时间趋于相同, 功耗曲线更为规整, 也能增强电路的抗侧信道攻击的能力.

Montgomery powering ladder 可以抵御 SPA 和 safe error. 对于 DPA 的攻击, 可以采用在模数上加掩码来对抗^[4-5].

4 性能对比及分析

从表 1 中可以看出, 本设计的电路方案据有更低的“功耗×面积/吞吐率”指标, 说明在同样的吞吐率下, 本文的设计所用的功耗和占用的面积更小, 性能优越.

表 1 CIOS 模乘电路的性能对比

	面积/ kgates	功耗/ mW	吞吐率 /(kb/s)	(功耗×面积/ 吞吐率) [kgates×mW/(kb/s)]
本文	14.5	1.67	15.16	1.59
文献[5]	74	80	559	10.59
文献[6]	103	60	588	10.51

	时间	工艺	时钟频率/MHz
本文	2014	SMIC 0.13 μm	30
文献[5]	2013	TSMC 90 nm	595
文献[6]	2014	TSMC 90 nm	595

5 系统级验证

为了对模幂运算单元进行系统验证, 需要将模幂运算单元集成到 SoC 系统中. 本文基于苏州国芯科技有限公司提供的 CS0_SDK 开发包, 将模幂运算单元连接到开发平台中的 APB 总线上去. 整体 SoC 验证平台结构图如图 4 所示.

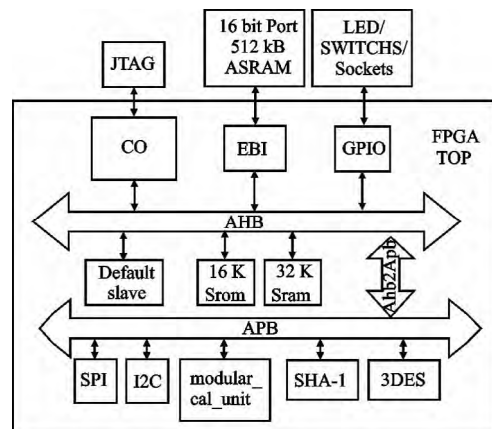


图 4 Co_MCU 结构图

其中 C0 为 32 位低功耗 RISC 核 CPU, 支持 JTAG 调试, 具有抗 DPA、TA 攻击特性, 支持 AM-BA 总线协议. 在图 6 中, modular_cal_unit 为模幂运算单元. 将该模块挂载到 APB 总线中去, 这样就

可以通过 C0 向模幂单元寄存器发送加密指令,实现对模幂单元系统级仿真功能验证。

将模幂运算单元挂载到 APB 总线接口有两个关键模块需要实现:一个是多路选择器,另一个是时钟频率调节模块。该 SoC 系统中共挂载了 5 个 APB 模块,分别为 IIC、SPI、模幂运算单元、SHA-1、DES。多路选择器模块根据输入的 PADDR 地址,选通对应的 APB 从模块,即将对应的 PSELX 置为高电平,如图 5 所示。

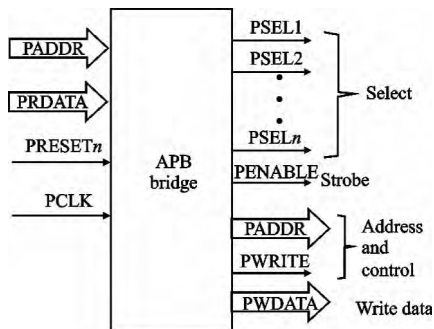


图 5 多路选择器模块

时钟频率调节模块的电路结构图如图 6 所示。当智能卡进行非接触式通信时,智能卡能量由射频前端电路提供。一般地,在 $0.13 \mu\text{m}$ 工艺下,射频前端电路在最坏情况下能够提供 1.8 V 电压和 5 mA 电流,此时对智能卡的功耗要求就十分苛刻。本文提出对模幂运算单元的时钟频率进行动态调整。智能卡在不同的通信工作模式下,模幂模块采用不同的工作频率,从而可以实现对模幂功耗的动态调整,进而可实现对智能卡的整体功耗动态调整。由于受课题资源所限,将 PLL 电路有数字分频电路进行替代。

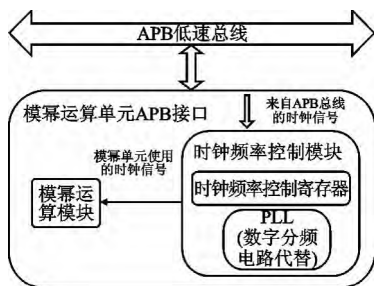


图 6 时钟频率调节模块

基于 CS0_SDK 开发平台,在 Synopsys VCS 上完成了模幂运算单元的 RSA 加解密仿真,并将仿真成功的硬件代码通过 Quartus II 综合下载到 HSNIO2C35 FPGA 开发板上,实现了 RSA 加解密运算^[7]。

6 结束语

文中对比了 CIOS 和 FIOS 模乘算法的性能,虽然 FIOS 模乘器面积比较小,但是功耗方面 CIOS 模乘算法具有优势且电路面积在智能卡可接受范围之内,更加适合于智能卡的应用。智能卡中的公钥算法 RSA、ECC 都可以调用模幂电路来实现数字签名。最后将模幂运算单元挂到 SoC 系统 APB 总线中,实现了对模幂运算单元的系统级仿真和 FPGA 验证。本课题承蒙工信部核高基重大专项“双界面金融卡 SoC 芯片研发与产业化”(项目号:2014ZX01032205)资助,特此致谢。

参考文献:

- [1] 王延斌,叶兵,孙东昱. 基于改进 Montgomery 模乘算法的智能卡协处理器设计[J]. 微电子学与计算机, 2005, 21(12): 104-106.
- [2] Cetin Kaya Koc, Tolga Acar, Burton S Kaliski. Analyzing and comparing Montgomery multiplication algorithms[J], IEEE Micro, 1996, 16(3): 26-33.
- [3] Hachez G, Quisquater J J. Montgomery exponentiation with no final subtractions: Improved results[C]// Cryptographic Hardware and Embedded Systems—CHES 2000. Berlin, Heidelberg: Springer, 2000: 293-301.
- [4] Joye M, Yen S M. The Montgomery powering ladder [C]// Cryptographic Hardware and Embedded Systems—CHES 2002. Berlin, Heidelberg: Springer, 2003: 291-302.
- [5] Yen S M, Joye M. Checking before output may not be enough against fault-based cryptanalysis[J]. Computers, IEEE Transactions on, 2000, 49(9): 967-970.
- [6] Ye J H, Hung T W, Shieh M D. Energy-efficient architecture for word-based Montgomery modular multiplication algorithm[C]// VLSI Design, Automation, and Test (VLSI-DAT), 2013 International Symposium on. Taiwan: Hsinchu, IEEE, 2013: 1-4.
- [7] Lin W, Ye J, Shieh M. Scalable montgomery modular multiplication architecture with low latency and low memory bandwidth requirement[J]. Computers, IEEE Transactions on, 2014, 63(2): 475-483.

作者简介:

邵佳佳 女,(1989-),硕士研究生.研究方向为信息安全算法的数字电路设计与实现. E-mail: sjj11@mails. tsinghua. edu. cn

乌力吉 男,(1965-),副教授,博士生导师.研究方向为民用信息安全与汽车电子.

张向民 男,(1966-),讲师.研究方向为民用信息安全.