第34卷 第12期 计 算 机 工 程 2008年6月
Vol.34 No.12 Computer Engineering June 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)12—0209—03 文献标识码：A 中图分类号：TP391

改进的蒙哥马利算法及其模乘法器实现

蒋晓娜，段成华
(中国科学院研究生院信息科学与工程学院，北京 100049)

摘 要：模乘运算的速度决定了公钥加密系统和众多通信系统的系统性能。通过分析Walter等学者对蒙哥马利算法的研究成果，得到运算精简基2-MMM算法，实现基于运算精简算法的线性脉动阵列模乘法器。在验证改进算法正确性后，对模乘法器进行功能仿真和综合。用TSMC 0.18 μm标准单元库综合，可以工作在200 MHz时钟下，等效单元为42 k门，完成1 024 bit模乘法运算需15 370 ns。
关键词：蒙哥马利模乘；线形脉动阵列；公钥方案

Improved Montgomery Algorithm and Implementation of Modular Multiplier

JIANG Xiao-na, DUAN Cheng-hua
(School of Information Science and Engineering, Graduate University of Chinese Academy of Sciences, Beijing 100049)

【Abstract】 Modular multiplication operation is a key factor of determining performance system of publickey cryptography systems and most of communication systems. In this paper, an improved Montgomery Modular Multiplication(MMM) algorithm called operation reduced radix 2-MMM algorithm is developed by combining the available typical Montgomery modular algorithms. And a linear systolic array circuit scheme is used for implementing the modular multiplier. Validation of the modular multiplier functionality is conducted on ModelSim SE 6.0d platform. Based on the TSMC 0.18 μm CMOS technology, area of the modular multiplier is about 42k equivalent gates, the system frequency can up to 200 MHz, and the 1024-bit modular multiplication is 15 370 ns.
【Key words】 Montgomery Modular Multiplication(MMM); linear systolic array; public key schemes

1 概述

模乘作为公钥加密系统和众多通信系统中的核心运算，其运算速度决定了系统的性能。目前，基于蒙哥马利模乘(Montgomery Modular Multiplication, MMM)算法的脉动阵列硬件实现是提高大数模乘运算速度普遍采用的方法。

采用MMM算法的脉动阵列来解决大数模乘问题是由英国学者Walter[1]于1993年提出的，其采用的是二维脉动阵列结构，阵列规模随模乘输入位数(n)呈非线性增长(n^2)，同时每个PE的逻辑电路由若干与门、或门和异或门组成，PE每次运算需要经过5级门电路延时，如果将一个PE的运算时间简单看作T，则完成一次模乘运算需要$(3n+2)T$，T为系统的时钟周期。此后，丹麦学者P.Kornerup[2]提出了基于MMM算法的线性脉动阵列模乘法器，其所需PE数目随模乘输入位数(n)呈线性变化，PE由4个全加器和若干与门组成，因此，所需芯片面积较之Walter提出的模乘法器有较大减少，但其PE的逻辑电路比较复杂，系统最高时钟频率受限。考虑到上述脉动阵列模乘法器PE逻辑电路均比较复杂，N.Nedjah[3]改进了MMM算法，提出一种简洁的PE电路实现方案。该PE电路由1个1 bit FA和1个4-1多路选择器(MUX)实现，PE中完成一次运算经过1个1 bit FA和1个4-1 MUX的延时。

本文首先通过结合Walter等学者[1,3-4]对MMM算法的分析，得到运算精简基2-MMM算法；随后采用改进算法实现线性脉动阵列大数模乘法器，结果表明，将运算精简算法映射到线性脉动阵列结构，模乘法器在速度和面积性能上取得了平衡，适合现代应用的需要。

2 Montgomery算法及其改进

P.L.Montgomery提出的MMM算法采用模加和右移的方法，避免了通常求模算法中费时的除法操作，被认为是一种大数模乘运算硬件实现的有效的算法。

2.1 Montgomery算法

MMM算法给出求解$A\times B\times R^{-1} \bmod M$的快速方法，通过一定的预运算和后运算得到真正形如$S=A\times B \bmod M$模乘运算结果。

算法1为原始的MMM算法，其中，$T=A\times B$且有$0\leq T<RM$，M为模数，且$M>1$，R是满足式(1)的一个基(通常取R为2^n，n为输入大数的位数)：

$$\gcd(M,R)=1 \tag{1}$$

R^{-1}和M'是满足式(2)和式(3)的数：

$$R\,R^{-1}(\bmod M)=1 \tag{2}$$

$$R\,R^{-1}-MM'=1 \tag{3}$$

算法 1

function1 REDC(T)

REDC $(T,M)=T\times R^{-1}\bmod M$

$m=(T \bmod R)\,M' \bmod R$;

$P=(T+mM)/R$;

If $P>M$ then $P-M$ else return P;

算法1中每次模乘运算总有$A\times B$，TM'和mM 3次大数乘法运算，当A, B和M均为1 024 bit以上的大整数时，带有

基金项目：国家“863”计划基金资助项目(2002AA141041)
作者简介：蒋晓娜(1981－)，女，硕士研究生，主研方向：安全协处理器设计；段成华，教授
收稿日期：2007-06-30 **E-mail：**xnjiang04@mails.gucas.ac.cn

大数乘法的模乘硬件实现仍然是十分困难的，因此，Montgomery 提出变形的 MMM 算法。

在变形的MMM算法中，大整数A, B和M以r为基表示，通常$r=2^w$，w称为字长，则对于输入是n位的大整数有

$$A=\sum_{i=0}^{n-1}a_i r^i,\quad B=\sum_{i=0}^{n-1}b_i r^i,\quad M=\sum_{i=0}^{n-1}m_i r^i$$

在变形的 MMM 算法中，对于式(3)相应地有

$$R\,R^{-1}\text{-}MM'\equiv 1(\bmod r^n) \tag{4}$$

由式(4)易得，$M'\bmod r=(\text{-}M)_r^{-1}=(r\text{-}M[0])_r^{-1}$。在变形的MMM算法中，使用了循环，每次循环对大数中字长的 1 位进行运算，因而较之算法 1 利于硬件实现。

2.2 基 2-MMM 算法

目前，大数模乘运算的硬件实现普遍采用脉动阵列结构，当选取上述变形算法中的基r为 2 时，因为M是奇数(由式(1)易知)，所以$M[0]=1$，则式(4)中的$(r\text{-}M[0])_r^{-1}=1$，得到基 2-MMM算法，见算法 2。

$$A=\sum_{i=0}^{n-1}a_i 2^i,\quad B=\sum_{i=0}^{n-1}b_i 2^i,\quad M=\sum_{i=0}^{n-1}m_i 2^i$$

算法2

function 2 MontPro(A, B, M)

MontPro$(A, B, M)=A\times B\times 2^{-n} \bmod M$

$P=0$;

For $i=0$ to n-1

$Q[i]=(P[0]+a_i\times b_0)\bmod 2$;

$P=(P+a_i\times \mathrm{B}+Q[i]\times M)\mathrm{div}\ 2$;

采用基 2-MMM 算法的脉动阵列模乘法器，每个 PE 完成的是 1 bit 数据的运算，同时右移 1 位的操作就是除 2。因此，模乘法器不仅可以工作在高时钟频率下，而且降低了硬件设计复杂度。

2.3 改进的基 2-MMM 算法

MMM算法作为求解模乘运算普遍采用的算法，被众多学者研究和改进。为了解决线性脉动阵列Montgomery模乘法器面积和速度平衡问题，在重点研究了Walter等学者[1,3-4]对MMM算法理论分析结果基础之上，得到运算精简基 2-MMM算法。运算精简基 2-MMM算法对MMM算法做了以下 3 点改进：

(1)简化求解过程。MMM算法引入求q_i，用于保证P是整除的结果，原始算法中该步运算降低了MMM算法的效率。在满足计算P时序要求的同时，为了有效提高算法效率，运算精简算法根据Eldridge和Walter[4]提出的简化求解q_i的方案，将乘数B左移 1 位，且$b_0=0$，这样q_i值仅与$P_{[i]}[0]$有关，初始时$P_{[0]}[0]=0$，因此，$q_0=0$。

(2)减少加法操作次数，简化关键运算。MMM算法中关键运算是P的求解，可以通过判断a_i和q_i当前周期的值[3]，直接得到$a_i\times b_j+q_j\times m_j$的运算结果。对于判断结果中的$m_j+b_j$的值，保证在模乘运算执行到该处时，此数据已获得前提下，可以在模乘任何时间处理得到且只需运算一次，所以，每次循环中求解P只包括一个选择和一个 1 bit加法操作，大大减少了模乘运算中的加法次数。

(3)消除减法操作。原始算法中，当$P\geq M$，为使本次模乘运算的结果可以直接作为下次模乘运算的输入，需要多做一次减法运算，使P满足 $0<P<M$。该减法操作，既增加了运算步骤，又使模乘运算可能受到差分能量分析等的攻击，安全性将降低。为了消除额外的减法操作，被乘数A和乘数B须满足：$A<2M$且$a_n=0$，$B<2M$且$b_{n+1}=0$，$m_{n+1}=m_n=0$[1,4]，易证运算结果$P<2M$。这里增加a_n，即增加了一次循环运算，从而保证输出结果$P<2M$，而增加b_{n+1}和m_{n+1}，m_n是为了保证运算过程中，P的中间结果中的信息没有丢失，保证最后模乘运算结果的正确性。

运算精简基 2-MMM 算法，完成一次 n bit 模乘运算，只需 1 次 n bit 和$(n+1)(n+2)$次 1 bit 加法操作，及$(n+1)(n+2)$次选择操作。

function Modi_MontPro_2(A,B,M)

Modi_MontPro_2$(A, B, M)=A\times B\times 2^{-n-1} \bmod M$

A：$a_n, a_{n-1}, \cdots, a_1, a_0$且$a_n=0$；

B：$b_{n-1}, b_n, b_{n-1}, \cdots, b_1, b_0$且$b_0=b_{n+1}=0$;

M：$m_{n-1}, m_n, m_{n-1}, \cdots, m_1, m_0$且$m_0=1, m_{n+1}=m_n=0$;

$S=B+M$;

$P[0]=0$;

for $i=0$ to n

$q_i=(P_{[i]}[0]+a_i\times b_0)\bmod 2$；

for $j=0$ to $n+1$

switch a_i, q_i {

1, 1 : $Mux[j]\rightarrow s_j$;

1, 0 : $Mux[j]\rightarrow b_j$; (5)

0, 1 : $Mux[j]\rightarrow m_j$;

0, 0 : $Mux[j]\rightarrow 0$;}

$$P_{[i+1]}[j-1]+Ca2_{[i]}[j]=P_{[i]}[j]+Mux[j]+Ca2_{[i]}[j-1] \tag{6}$$

3 线性脉动阵列模乘法器设计

根据运算精简基 2-MMM 算法，结合线性脉动阵列结构特点，实现模乘法器的设计。

3.1 模乘法器整体结构

采用运算精简基 2-MMM 算法的线性脉动阵列模乘法器主要由 5 个模块组成，包括输入单元、模乘运算单元、输出单元、时钟单元和控制单元。其中输入单元接收并为参与运算的数据提供数据通道，特别包括一个完成 n bit 加法运算的加法处理模块；模乘运算单元是模乘法器的核心单元，完成大整数模乘运算；输出单元将运算结果并行输出；时钟单元为其他模块提供时钟；控制单元协调各个单元按时序工作。

3.2 模乘运算单元

模乘运算单元是模乘法器的核心单元，该单元由一系列结构类似的 PE 构成的线性脉动阵列来实现，具体结构如图 1 所示。在采用改进算法实现 n bit 模乘运算的线性脉动阵列模乘法器中，需要 n+2 个 PE。按位置不同图 1 中的 PE 分为左侧处理单元($PE[0]$)、普通处理单元($PE[1], PE[2], \cdots, PE[n\text{-}1]$)、次右侧处理单元($P[n]$)和右侧处理单元($P[n+1]$)。

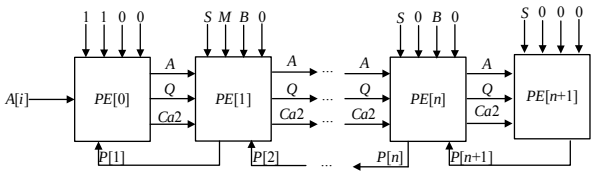

图 1 n bit 模乘运算单元

根据改进算法中式(5)和式(6)，易得包括 1 个 FA 和 1 个 4-1MUX 的普通处理单元示意电路，如图 2 所示。

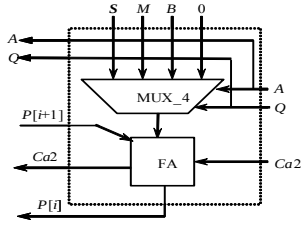

图 2 普通处理单元 PE[i]

3.3 模乘法器工作原理

在第 $0T$(时钟周期)，乘数 $B[1],B[2],\cdots,B[n\text{-}1]$、模数 $M[1],M[2],\cdots,M[n\text{-}1]$输入到相应的 PE 中，被乘数 $A[0],A[1],\cdots,A[n\text{-}1],A[n]$每隔一个 T 从 $PE[0]$输入到运算单元中，如果简单设每个 PE 单元的运算需要一个 T，则经过$(2n+2)T$，运算结果 $P[0]$从 PE[1]输出，随后每个时钟周期 $P[1], P[2],\cdots,P[n]$依次从相应的 PE 输出，经过$(3n+2)T$ 得到模乘运算结果 $P[0,1,\cdots,n]$。

对于 $B+M$ 的运算而言，只要保证时序正确，该运算在输入单元中与运算单元并行求解，并且在整个模乘运算过程中只需计算一次即可。

4 结果与性能分析

4.1 结果验证

用 C 语言执行扩展欧几里德算法对运算精简基 2-MMM 算法做了验证，用 Verilog 语言描述了基于运算精简算法的线性脉动阵列模乘器并用 ModelSim SE 6.0d 对其进行了功能仿真，与 C 语言执行运算精简算法输出结果比较，通过多组数据的比较，结果一致，验证了算法的正确性。

限于篇幅，这里给出 $n=4$ 的验证过程。欲求 $A=11,B=10, M=15$ 的模乘运算结果，根据算法输入 $A=\{01011\}=1, B=\{010100\}=20, M=\{001111\}=15$，验证过程包括以下步骤：

步骤 1 利用扩展欧几里德算法求得：

$R(2^4)$的模M乘法逆元R^{-1}为 1，则$P=11\times10\times1 \bmod 15=5$。

$R(2^5)$的模M乘法逆元R^{-1}为负 7，则$P=11\times20\times(-7) \bmod 15=5$。

步骤 2 利用 C 语言实现运算精简模乘算法，求得的结果：$P=20$。

步骤 3 利用 ModelSim SE 6.0d 对模乘法器进行仿真，在 $T=14$ 时，结果 $P=20$ 输出并保持一个时钟周期的时间。$20 \bmod 15=5$，所以，易知 4 个输出数据是等价的，同时输出结果满足 $P<2M$ 即 $20<30$，当输入多组不同数据进行仿真验证时，结论相同，从而验证算法及其线性脉动阵列模乘法器功能的正确性。

4.2 性能分析

基于运算精简模乘算法的线性脉动阵列模乘器，求解过程中简化了求解 $Q[i]$的运算，避免了最后除法操作，求解过程中只需进行一次 $B+M$ 运算，减少了加法运算次数，PE 仅由 1 bit FA 和 1 个 4-1 Mux 构成，完成一次 n bit 模乘运算过程需 $3n+2$ 个时钟周期。基于 TSMC 0.18 μm 标准单元库，利用 Synopsys 公司的 DesignCompiler 完成 1 024 bit 模乘法器的逻辑综合，综合结果为：系统时钟可达 200 MHz，等效单元 42 k 门，完成一次 1 024 bit 模乘运算用 15 370 ns。因此，基于运算精简模乘算法的大数模乘法器，在面积和时间上取得了很好的平衡，见表 1。

表 1 本文设计与其他模乘法器的性能比较

设计方案	阵列规模 (*PE* 数目)	*PE* 电路 (*Cell* 数目)	核心单元 (等效门数)	运算时间 (时钟周期)
C.D.Walter [1]	$(n+3)(n+4)$	7and+5xor+2or	14 723 072	$(3n+2)T$
P.Kornerup [2]	$n/2$	4FA+4and	10 240	$(2n)T$
N.Nedjah [3]	$(n+1)(n+1)$	1Mux+1FA	840 500	$(3n+4)T$
Ours	$n+2$	1Mux+1FA	8 208	$(3n+2)T$

5 结束语

综上所述，通过对已有 MMM 算法分析，结合已有算法的改进方案，得到运算精简基 2-MMM 算法，并采用该算法实现了 1 024 bit 的线性脉动阵列模乘法器。结果表明，该模乘法器在速度和时间上达到了很好的平衡。可以看出，模乘法器的硬件利用率不高，所需的硬件资源比较多，在对功耗要求比较高的场合不能取得更佳性能，这将是下一步研究的重点。

参考文献

[1] Walter C D. Systolic Modular Multiplication[J]. IEEE Transactions on Computers, 1993, 42(3): 376-378.

[2] Kornerup P. A Systolic, Linear-array Multiplier for a Class of Right-shift Algorithms[J]. IEEE Transactions on Computers, 1994, 43(8): 892-898.

[3] Nedjah N, De M M L. Three Hardware Architectures for the Binary Modular Exponentiation: Sequential, Parallel, and Systolic[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2006, 53(3): 627-633.

[4] Walter C D. Improved Linear Systolic Array for Fast modular Exponentiation[J]. IEE Computers Digit. Tech., 2000, 47(5): 323-328.

(上接第 208 页)

迹生成算法，并将新点型所对应的操作添加到上述需要分析点型的 3 种情况中即可。

(3)易维护性。采用节点栈和像素栈实现界面显示与后台操作的拨离，程序结构清晰，代码利用率高，易于维护调试。

(4)精巧的结构实现了嵌入式软件设计的高效率与松耦合，整体上满足嵌入式系统的性能要求。实验证明，本算法可应用于复杂的花样编辑操作环境中。

4 结束语

随着嵌入式技术的深入应用，电子花样机已成为新一代智能缝制设备的典型代表，得到越来越广泛的应用。花样编辑则是增强其智能性、方便用户操作的一个重要功能。本文提出了一种在嵌入式平台上构建一个高效、简洁的花样编辑环境的方法。应用表明，该算法对花样编辑软件、绣花软件都有一定的应用价值。

参考文献

[1] 梁 克, 张凯龙, 周兴社. 智能花样缝制设备的主流花样格式分析与仿真[J]. 计算机工程, 2006, 32(3): 259-261.

[2] Zhang Kailong, Zhou Xingshe, Liang Ke, et al. The Customizable Embedded System for Seriate Intelligent Sewing Equipment[C]// Proc. of ICESS'04. Berling, Germany: Springer-Verlag, 2004.

[3] 刘海涛, 郭 磊, 陈世福. 智能化电脑刺绣编程环境的设计与实现[J]. 软件学报, 2001, 12(9): 1399-1405.

[4] Schneider P J, Eberly D H. 计算机图形学几何工具算法详解[M]. 北京: 电子工业出版社, 2005.