可伸缩双域 Montgomery 乘法器的优化设计与实现

秦 帆,戴紫彬
(解放军信息工程大学 电子技术学院,河南 郑州 450004)

摘 要:模乘运算是公钥密码算法中的关键运算,本文基于全字运算的 Montgomery 模乘算法,设计了具有可伸缩硬件结构的模乘器。该模乘器可以基于固定的数据路径宽度对任意长度的数据进行运算,并且能够支持两个有限域上的运算。最后用 Verilog 硬件描述语言对该乘法器的硬件结构进行代码设计,并用 Synopsys 公司的 Design Complier 在 Artisan SIMC 0.18 μm typical 工艺库下综合。实验结果表明,相对于其他模乘器设计,本文设计具有较高的时钟频率,并且由于大大减少了运算所需的时钟周期数,模乘运算速度较快。

关键词:公钥密码;Montgomery 模乘;双有限域;可伸缩结构;ASIC
中图分类号:TP332 **文献标识码**:A

An optimized scalable and unified hardware architecture of Montgomery multiplier

QIN Fan, DAI Zi Bin
(Institute of Electronic Technology, The PLA Information Engineering University, Zhengzhou 450004, China)

Abstract: Modular multiplication is the core operation of PKC(public key cryptography). Based on the full-word Montgomery multiplication algorithm, a scalable and unified modular multiplier is proposed, which can work with any precision of the operands and work in both prime and binary fields. It is captured in Verilog and synthesized under 0.18 μm CMOS technology. The result indicates that this work can achieve high clock frequency and perform efficiently than other works, as the clock numbers are reduced greatly.

Key words: PKC; Montgomery multiplication algorithm; prime finite field and binary extension finite field; scalable architecture; ASIC

随着计算机网络的发展和普及,信息安全问题受到越来越多的关注,而对数据进行加密是一种常用且行之有效的保护信息安全的方法。其中,被公认为最有效的是椭圆曲线加密算法。椭圆曲线加密算法(ECC)是一种基于椭圆曲线离散对数问题的公钥密码算法,是 1985 年分别由 Miller[1]和 Koblitz[2]独立提出的。相对于其他公钥密码算法(如 RSA 和 ElGamal),椭圆曲线密码算法具有计算速度快、存储空间小、带宽要求低等优点,特别适用于各种无线设备和智能卡之类计算资源受限的设备,因而受到了人们的广泛关注,成为最有希望的公钥密码算法。而模乘运算是椭圆曲线加密算法中的核心运算,所以如何高效地实现模乘运算是目前的一个研究热点。Montgomery 模乘算法[3]是目前应用最为广泛也最为高效的模乘算法,它将以往模乘运算中的除法运算以简单的移位操作代替,大大加快了模乘运算的速度和效率,同时也非常适合硬件实现。基于 Montgomery 模乘算法的硬件设计非常多,在这些设计中,时钟数消耗最少的是高基 Montgomery 模乘算法[4],可伸缩性最好的是基于字的 Montgomery 模乘算法[5]。本文针对两个有限域即素数域 GF(p)和二元扩域 GF(2^n)上进行模乘运算的差异,提出了一种高速可伸缩 Montgomery 乘法器的设计方案。该方案将高基和基于字的 Montgomery 模乘算法结合在一起,在不降低时钟频率的基础上,大大减少了运算所需的时钟周期数,从而提高了运算效率[6-7]。

1 Montgomery 模乘算法

1.1 素数域和二元扩域

素数域 GF(p)中的元素为整数集合{0,1,2,…,p-1},p为素数。素数域上的加法和乘法由以下两步完成:

(1)整数的加法或乘法运算；

(2)若结果大于等于素数 p，则由 p 进行约简。

二元扩域 $GF(2^n)$ 上的元素可以由次数小于 n 的多项式表示。二元扩域上的加法运算为两个相加元素对应位的异或操作，不存在进位，结果表示为次数不会超过 n 的多项式。与素数域类似，二元扩域上的乘法由以下两步完成：

(1)多项式的乘法运算；

(2)由不可约多项式 $p(x)$ 对结果表示的多项式进行约简。

1.2 Montgomery 模乘算法

给定整数 a 和 b，以及素数 p，由 Montgomery 模乘算法计算得到 $\bar{c}=\mathrm{MonMult}(a,b)=abR^{-1}(\mathrm{mod}\,p)$，其中 $a,b<p<R$，$R=2^n$，p 为 n bit 素数。Montgomery 模乘并没有直接得到模乘结果 $c=ab(\mathrm{mod}\,p)$，所以在计算前需要对 a 和 b 进行转换，计算完成后也需要将中间结果 $\bar{c}$ 转换回最终要得到的结果 c。转换过程如下：

$\bar{a}=\mathrm{MonMult}(a,R^2)=aR^2R^{-1}=aR(\mathrm{mod}\,p)$

$\bar{b}=\mathrm{MonMult}(b,R^2)=bR^2R^{-1}=bR(\mathrm{mod}\,p)$

$c=\mathrm{MonMult}(\bar{c},1)=cRR^{-1}=a(\mathrm{mod}\,p)$

假设 $R^2(\mathrm{mod}\,p)$ 已经通过预计算得到并存储在寄存器中，所以每次转换只需要一次 Montgomery 模乘运算。素数域上基为 2^k 的 Montgomery 模乘算法如算法 1 所示。

算法 1：

Input: $a,b\in[1,p-1],p,m$

Output: $c\in[1,p-1]$

(1) $c=0,p'_0=2^k-p_0^{-1}$

(2)for $i=0$ to $m-1$

(3) $q=(c_0+a_ib_0)p'_0(\mathrm{mod}2^k)$

(4) $c=(c+a_ib+qp)/2^k$

在算法 1 中，整数 a 以 2^k 为基，表示为 $a=\sum_{i=0}^{m-1}a_i2^{k\cdot i}$，其中 $m=\lceil n/k\rceil$。在该算法的第(4)步中，b、p、c 以完整数参与运算，为设计可伸缩的乘法器结构，本文以字来表示这几个数，在每个时钟对这几个数据的其中一个字进行计算。以 w 表示字的长度，b、p、c 可以采用按字表示的方式，$b=\sum_{j=0}^{e-1}b^{(j)}2^{w\cdot j}$，$p=\sum_{j=0}^{e-1}p^{(j)}2^{w\cdot j}$，$c=\sum_{j=0}^{e-1}c^{(j)}2^{w\cdot j}$，其中 $e=\lceil n/w\rceil$，即以 2^w 为基表示 b、p、c、q、c_0、b_0 以及 p'_0 均为 k bit 的整数。

二元扩域上的 Montgomery 模乘算法如算法 2 所示。

算法 2：

Input: $a(x),b(x),p(x),m$

Output: $c(x)$

(1) $c(x)=0,p'_0(x)=2^k-p_0^{-1}(x)(\mathrm{mod}\,x^k)$

(2)for $i=0$ to $m-1$

(3) $q(x)=(c_0(x)+a_i(x)b_0(x))p'_0(x)(\mathrm{mod}\,x^k)$

(4) $c(x)=(c(x)+a_i(x)b(x)+q(x)p(x))/x^k$

算法 1 和算法 2 主要有两点区别：(1)算法 2 中的加法为按位异或操作；(2)算法 1 在计算完成后如果结果大于素数 p，需要做一次约简，而算法 2 则不需要。不过这一步约简可以避免，所以本文算法 1 中并没有包括这一步骤。两个有限域上的元素以二进制数表示，其表示方式是完全相同的，所以基于算法 1，对关键计算单元稍做修改，设计一个统一的模乘器是切实可行的。

1.3 按字操作的 Montgomery 模乘算法

算法 1 是对每一个完整数进行计算，不具备可伸缩性，本文采用的算法对所有操作数的每一个字进行计算，以提高运算频率和可伸缩性。基于算法 1 改进的 Montgomery 模乘算法[7]如算法 3 所示。

算法 3：

Input: $A=(a_{m-1},\wedge,a_1,a_0)_{2^k}$，$B=(b_{m-1},\wedge,b_1,b_0)_{2^k}$，

$P=(p_{m-1},\wedge,p_1,p_0)_{2^k}$，$q=-p^{-1}\mathrm{mod}2^k=-p_0^{-1}\mathrm{mod}\,2^k$

Output: $C=\mathrm{AB}2^{-n}\mathrm{mod}\,P$

Step1: $C=0$

Step2:for $i=0$ to $m-1$

Step3: $z=0$

Step4: $t_i=(c_0+a_ib_0)q\mathrm{mod}\,2^k$

Step5:for $j=0$ to $m-1$

Step6: $S=cj+aibj+tipj+z$

Step7:if$(j\neq0)$then $c_{j-1}=S\mathrm{mod}\,2^k$

Step8: $z=s/2^k$，$c_{m-1}=z$

Step9:if$(C>P)$then $C=C-P$ else $C=C$

Step10:return C

如算法 3 所示，k 表示基，每次循环读入乘数 A、被乘数 B、模数 P 的 k 位，故该算法非常容易组织成流水线的结构，其按流水线组织结构进行运算的流程如图 1 所示。

2 算法的硬件实现

采用多级处理单元的流水线结构来实现算法，流水线的工作流程如图 1 所示，每一竖列表示一级流水线，每一横行表示一个运算周期，在这里先假设每个运算周期为一个时钟周期。在外部循环 $i=0$、内部 $j=1$ 这一循环经两个时钟周期完成后，才能够得到下一级流水线处理单元 A 所需的 C_0，此时对 A 的第二个字才可以开始扫描。也就是说在第 i 个外部循环的第 1 个内部循环经两个时钟周期完成后才可以开始第 $i+1$ 个外部循环的运算，所以采用这种流水线组织形式，每级流水线之间的延迟为两个时钟周期。对算法实现若采用如图 1 的流水线组织结构，完成一级流水线需要 m 个时钟周期。

若存在不少于 $l=\lceil m/2\rceil$ 个运算单元，则采用该流水线组织结构实现模乘将不间断地进行下去，不会因为没有空闲的运算单元而停止，所以完成两个 n 位数据的模乘运算需要 $3m-2$ 个时钟周期。这里以计算两个 7 bit 的

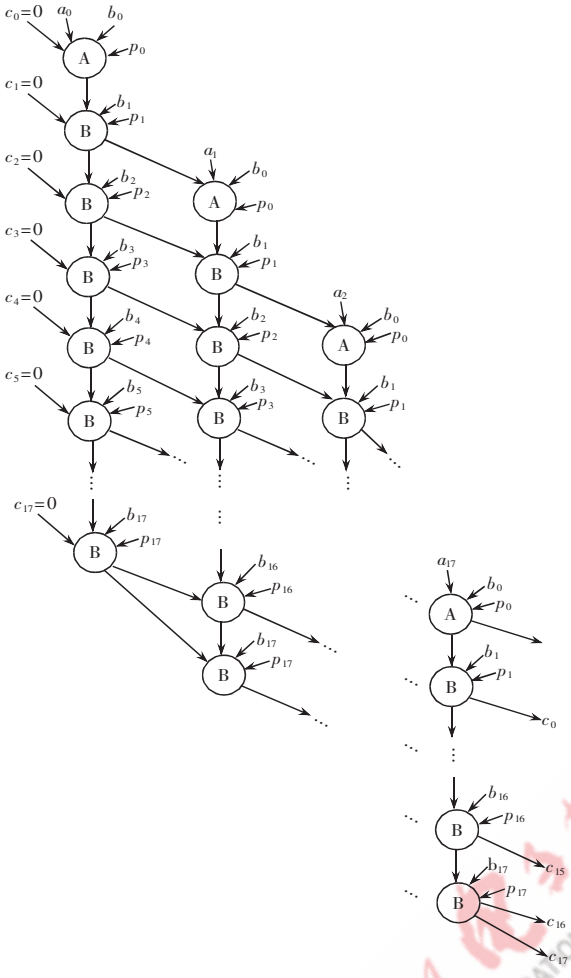


图 1 模乘算法的流水线运算流程图

操作数为例，取基 k=0，有 4 个运算单元，满足 l 不少于⌈m/2⌉的条件。其流水线计算流程如图 2 所示。

若存在的运算单元少于 $l=\lceil m/2 \rceil$ 个，则该流水线组织结构将会因为没有空闲的运算单元而停止部分时钟周期，这时完成算法需要 $3m-2+2(\lceil m/2 \rceil-1)$ 个时钟周期，同样以 7 bit 的操作数为例，若只有 3 个运算单元，即运算单元少于 $l=\lceil m/2 \rceil$ 个，其流水线计算流程如图 3 所示。

在满足寄存器大小的条件下，可以基于这样按字操作的 Montgomery 模乘算法设计可伸缩的硬件结构，即能够对任意长度的数据进行模乘运算。

2.1 处理单元的设计

由图 1 可以看出，要采用算法 3 设计模乘器，其基本运算单元包括处理单元 A 和处理单元 B。其中处理单元 A 完成算法 3 中循环 $i=0$ 以及 $j=0$ 时的

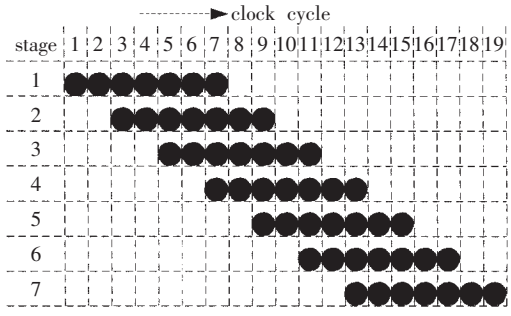


图 2 有 4 个运算单元的流水线计算

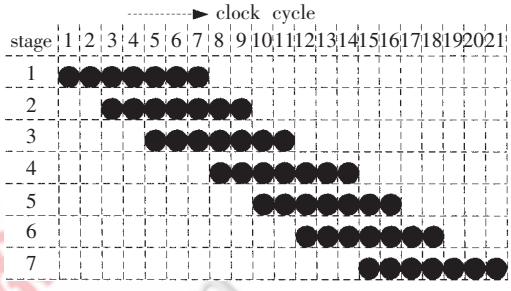
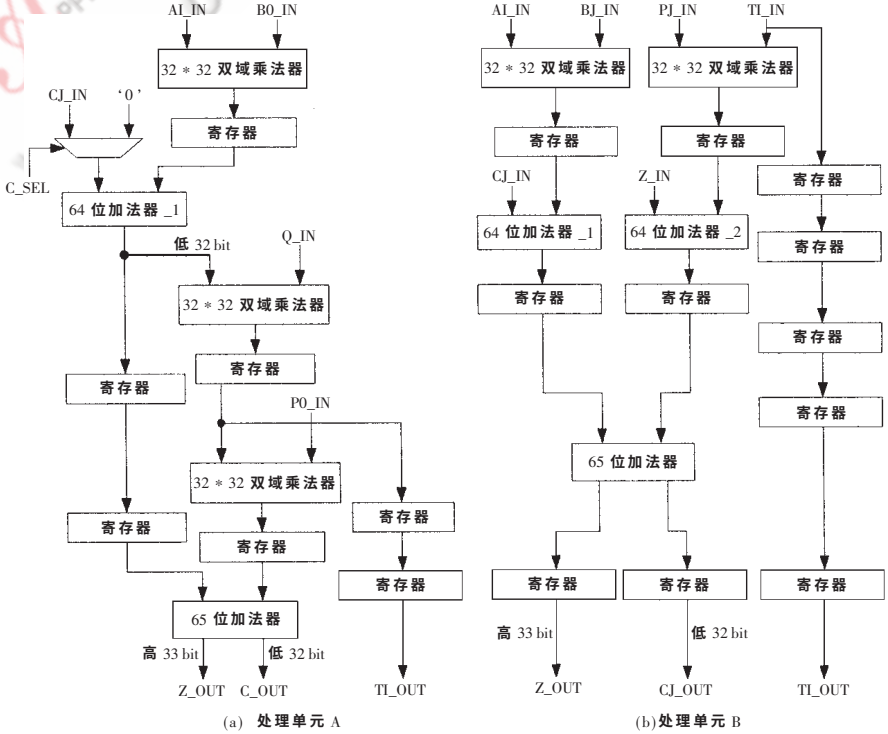


图 3 只有 3 个运算单元的流水线计算

运算，处理单元 B 完成算法中作其他循环时的运算。

由于在每轮循环时存在加法和乘法运算，会产生较长的进位传输延迟，所以为提高时钟频率和匹配两种不同处理单元之间的并行性，本文设计的处理单元 A 和处理 B 均采用 5 个时钟周期完成一次运算。处理单元 A 和处理单元 B 的微结构如图 4 中的(a)、(b)所示。



(a) 处理单元 A　　(b)处理单元 B

图 4 处理单元 A 和处理单元 B 的微结构

因为要完成两个有限域上的模乘运算，所以处理单元 A 和处理单元 B 中的乘法器和加法器均能同时完成两个有限域上的乘法和加法运算。乘法运算基本上可以分为两步：首先求出所有的基本乘积项，然后将所有乘积项求和。本文采用二阶 Booth 编码减少乘积项，采用 Wallace 树型加法器求和，以提高乘法器的运算速度。参照文献[7]的设计，只需要将第一级的 CSA（进位保留加法器）用 DFA（双有限域加法器）代替，这样就改造成可以用于双有限域的 Wallace 压缩树，双有限域乘法器的硬件电路结构如图 5 所示。其中 DFA 的电路结构如图 6 所示。

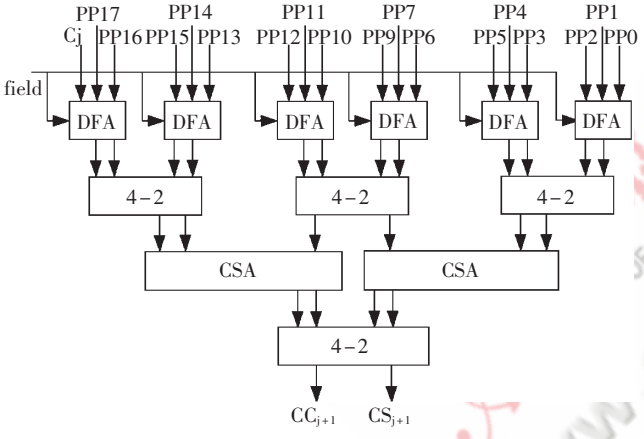


图 5 双有限域乘法器电路结构

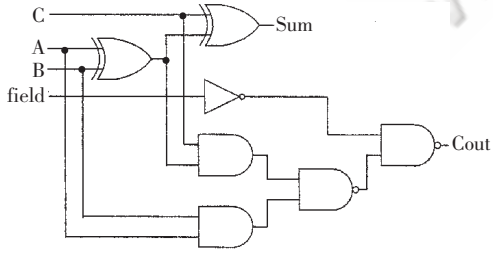


图 6 双有限域加法器 DFA

2.2 模乘单元整体硬件结构

对算法 3，本文选择 32 bit 作为操作数的字长设计高速可伸缩的模乘器，并采用容量为 576 的寄存器，即使该模乘器可以对 576 bit 以内任意长度的数据进行计算，若要支持更高比特数据的计算，只需要增加寄存器的容量。由图 1 可知，最高要计算 576 bit 的数据至少需要 1 个处理器 A 和 8 个处理器 B 才能使并行流水线流程不停顿地进行下去。所以为提高运算速度，本文设计的模乘单元整体结构如图 7 所示。

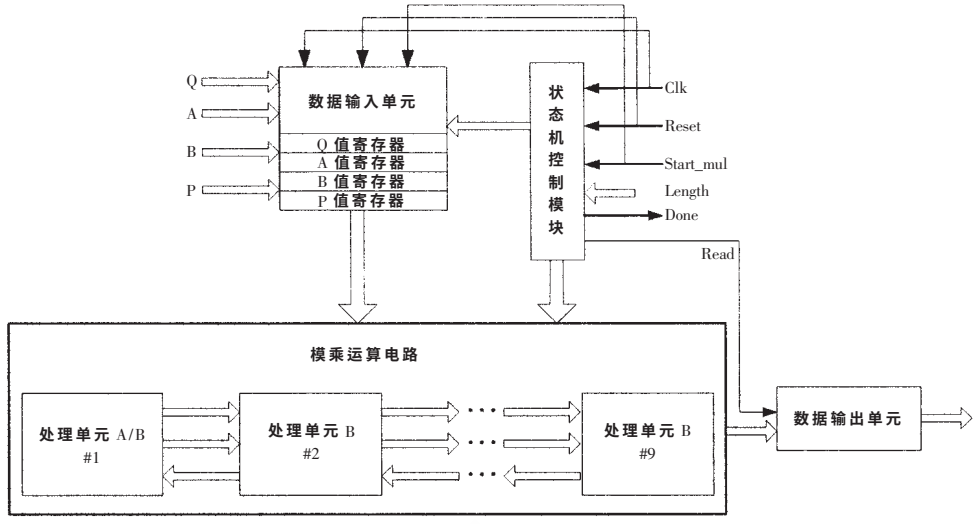


图 7 模乘单元整体电路结构

为适配一些处理器的字长，本文将数据输入单元和数据输出单元与外部的数据接口宽度均设计为 32 位，外部数据的写操作和输出结果数据的读操作均在状态机控制模块的控制下完成，数据输入单元中各个操作数寄存器的数据输出同样也在状态机控制模块的控制下完成，控制信号由输入的数据长度值 Length 产生。

3 性能指标

对本文设计的模乘器采用 Verilog HDL 硬件描述语言进行 RTL 描述后，使用仿真软件 ModelSim SE 6.0c 进行功能仿真。

对本文设计功能仿真正确后，使用 Synopsys 公司的 Design Complier 在 Artisan SIMC 0.18 μm typical 工艺库下综合，除去寄存器堆后的等效与非门为 8.1 万门，工作频率可以达到 210 MHz。忽略初次的数据数入与最后的数据输出所用的时间，完成一次 GF(p)上的 256 bit 输入数据的模乘运算只需要 0.24 μs。表 1 是本文设计的模乘器与已发表文献中同类设计的比较结果。

表 1 模乘器设计比较结果

设计方式	操作数长度/bit	工艺库	模乘时间/(μs@256bit)	时钟频率/MHz
Tenca[6]	256	0.5	6.6	80
Satoh[7]	160~256	0.13	0.48	137.7
史焱[8]	0~512	0.18	0.81	384
本文	0~576	0.18	0.24	210

由表 1 可以看出，相对于其他的模乘器设计，本文设计在时钟频率上具有一定的优势，并且因为对乘数和被乘数均按字进行计算，所以需要时钟周期数较少，能够达到较快的运算速度。

参考文献

[1] MILLER V S.Use of elliptic curves in cryptography[C]. CRYPTO'85, 1986: 417-426.

[2] KOBLITZ N.Elleptic curve cryptosystems[J].Mathematics of computation, 1987, 48(4): 203-209.

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

[4] ORLANDO G, PAAR C.A Scalable GF(p) elliptic curve processor architecture for programmable hardware[J].Proc. Cryptogr-aphic Hardware and Embedded Systems(CHES 2001), 2001: 349-363.

[5] TENCA A F, KOC C K.A scalable architecture for mont-gomery multiplication[J].Proc.Cryptographic Hardware and Embed-ded Systems.(CHES 1999), 1999: 94-108.

[6] SAVAS E, TENCA A F, KOC C K.A sscalable and unified multiplier architecture for fields GF(p) and GF(2^m)[J].Proc. Cryptographic hardware and embedded systems(CHES 2000), 2000: 1-20.

[7] SATOH A, TAKANO K.A scalable Dual-Field elliptic curve cryptographic processor[J].IEEE.Trans.Computers, 2003, 52: 449-460.

[8] 史焱, 吴行军.高速双有限域加密协处理器设计[J].微电子学与计算机, 2005, 22(5): 8-12.

(收稿日期: 2008-11-29)