对提高 RSA 算法中大数模乘运算速率的思考*

贾斌斌　王忠庆　方炜

（中北大学电气与控制工程学院，太原 030051）

摘要：RSA 算法的核心是大数模乘运算，提高其运算速率不仅对改进 RSA 算法本身有着重要的意义，而且，如果能够通过专用集成电路快速而低成本的实现，将会对电子商务的推广产生积极作用。在研究蒙哥马利算法的基础上，提出一种基于并行前缀加法器架构的基 2-Montgomery 模乘运算，构建了1 024 bit 的 Kogge-stone 加法器。仿真结果表明，该方法可以有效减少模乘运算中操作数的延迟时间，在一定程度上提高大数模乘的运算效率。

关键词：RSA 算法；基 2-Montgomery；Montgomery 算法；Kogge-stone 加法器

中图分类号：TP301. 6　　**文献标志码**：A

引用格式：贾斌斌，王忠庆，方炜．对提高 RSA 算法中大数模乘运算速率的思考[J]．信息通信技术与政策，2023，49(6)：84-90.

DOI：10. 12267/j. issn. 2096-5931. 2023. 06. 012

0　引言

在信息产业和技术日益发展的今天，网络信息安全尤为重要。作为网络信息安全的基础，如何确保密码系统不断改进、升级，成为广大信息工作者研究和关注的话题。目前，国内外密码系统中应用广泛的 RSA（Ron Rivest、Adi Shamir 和 Leonard Adleman）加密算法是较为成熟的公开密钥加密方法，该算法操作的核心是大数模幂运算，即由大数模乘反复运算实现。为了提高安全性，RSA 算法一般都在位宽 1 024 bit 及以上，以如此大的数据位进行计算意味着计算过程非常复杂，可能降低运算速率。笔者认为，提高模乘运算的速率是提高模幂运算的一个重要方法，也是有效加快 RSA 算法整体运算效率的一个重要途径。

1985 年，Perter Montgomery[1] 提出了一种只需运用乘法和数的右移操作就可以实现模乘运算的计算，把这种方法叫做 Montgomery 算法。国内曾有研究提出了其他路径，一种是运用基 2-Montgomery 算法来实现模乘运算[2-3]，优点是基 2-Montgomery 算法对于模乘运算只需加法器、异或运算器、移位器即可，不需要除法运算；不足是运用的硬件资源过多。另一种是运用基于保留进位加法器（Carry-Save Adder，CSA）的 Montgomery 算法，降低时钟个数，从而提高模乘运算速率[4]；不足是随着计算数的位宽不断增加，加法器的进位链也将不断增加，从而影响加法器的性能。

本文以模乘运算模块为切入点，对 RSA 算法中模乘运算进行研究。主要方法是通过将 Kogge-stone 加法器[5] 与基 2-Montgomery 算法结合，形成一种混合运算。结果表明这种混合运算可以在一定程度上提高大数模乘运算效率。

* 基金项目：山西省科技重大专项计划“揭榜挂帅”项目（No. 202101010101017）；山西省重点研发计划项目（No. 201903D111003）

1 RSA 算法中模乘运算原理介绍

1.1 RSA 算法中相关模块

RSA 算法[6]主要包含 RSA 顶层模块、模幂状态控制模块、模乘运算模块(见图 1)。

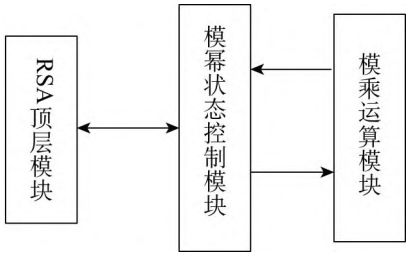

图 1 RSA 算法实现的模块

从图 1 可以看出模乘运算是底层模块,是整个算法的基础,模乘运算的快速实现直接关系着整体 RSA 算法加解密的速度。

1.2 Montgomery 模乘运算

Montgomery 乘法的数学表达式是 $A\times B\times r^{-1} mod\ p$。其中,$A$、$B$ 是与 p 同位长的大数,r^{-1} 是指 r 相对于 p 的模逆,即 r^{-1} 是满足如下条件的数:$r\times r^{-1} mod\ p=1$;这个条件成立的充要条件是 r 与 p 互素,这一点只需要 p 为奇数即可,所以 Montgomery 乘法通常适用于对奇数求模。具体算法描述见表 1。

表 1 Montgomery 乘法

输入:A,B,r,p

输出:$A\times B\times r^{-1} mod\ p$

$Monm(A,B,r,p)$

{ $t=A\times B$

$u=(t+(t\times p\ mod\ r))p/r$

if $u\geqslant p$

return $u-p$

}

Montgomery 算法则是将对模数的取余计算转化为对 2 的除法运算,在计算机系统中对 2 的除法运算则是简单的移位操作,从而大大简化了计算的难度。虽然 Montgomery 普通算法有效降低了模乘运算,但是当 A、B、p 的位数超过 1 024 位时,在运算处理和存储时的数值相当大,非常占用硬件资源,所以需要对 Montgomery 算法进行相应的改进优化。

1.3 基 2-Montgomery 算法

一般而言,硬件加密主要从运算速率和资源利用两个方面进行设计。对普通 Montgomery 算法进行优化主要有两个方面:一个是在乘法计算时,每次有选择地计算一部分值,依次输出;另一个是选择基数的优化方法。

计算 A、B 的 Montgomery 乘积的基 2 表达式[7]见表 2。

表 2 基 2-Montgomery 算法关键步骤

输入:A,B,P

$S[0]=0$;

for i 0 *to* $k-1$ *loop*;

$q_i=(S[i]_0+A_iB_0)\ mod 2$;

$S[i+1]=(S[i]+A_iB+q_in)\ div 2$;

end loop;

return $S[k]$

算法中,$A=\sum_{i=0}^{k-1} a_i\ 2^i=(a_0\ a_1\cdots a_{n-1})_2$,$B=\sum_{i=0}^{k-1} b_i\ 2^i=(b_0\ b_1\cdots b_{n-1})_2$ 和 $P=\sum_{i=0}^{k-1} p_i\ 2^i=(p_0\ p_1\cdots p_{n-1})_2$,$k$ 是位宽,$S[i]_0$ 是 $S[i]$ 的最低有效位,A_i 代表 A 的第 i 位,算法的输出结果为 $A\times B\times 2^{-k}(mod\ p)$。从表 2 可知,基 2-Montgomery 算法在步骤 $S[i+1]$ 中 3 个输入相加时有相应的临界延迟,增加了计算的延迟时间。

2 混合 Montgomery 乘法器的设计

2.1 Kogge-stone 加法器设计

Kogge-stone 加法器是利用 Peter M. Kogge 和 Harold S. Stone[8]于 1972 年提出的一种并行算法生成的树形加法器,该加法器具有逻辑层数低和扇入扇出较低的特点。

Kogge-stone 加法器见表 3,Kogge-stone 并行算法见表 4。

1 024 bit Kogge-stone 加法器设计。从表 3 可知 Kogge-stone 加法器的进位链和公式与表 4 中 Kogge-stone 并行算法中的一阶递归问题属于同一种,

表 3　Kogge-stone 加法器

$$S_i = p_i \oplus C_{i-1}(i = 1\cdots N)$$

$$c_i = g_i + C_{i-1} \times g_i$$

$$C_0 = C_{in}, C_{out} = C_{in}$$

进位项与产生项：$P_i = A_i \oplus B_i, g_i = A_i \times B_i$

得出进位链为：$C_i = g_i + C_{i-1} \oplus P_i$

$$x_i = a_i + x_{i-1} \oplus b_i$$

表 4　Kogge-stone 并行算法

对于序列，满足，$x_1 x_2 x_3 \cdots x_N \quad x_i = f(x_{i-1} x_{i-2} \cdots x_{i-m})$ 一阶递归如下：

$$x_i = a_i \times x_{i-1} + b_i ;$$

定义函数：$\widehat{Q} = (m,n) = \sum_{j=n}^{m} (\prod_{r=j+1}^{m} a_r) \times b_r$

其中 $\prod_{r=m+1}^{m} a_r = 1$

以 8 bit 加法器为例

结果：$x_1 = b_1 = \widehat{Q}(1,1)$

$$x_2 = a_2 \times x_1 + b_2 = a_2 \times b_1 + b_2 = \widehat{Q}(1,1)$$

…

$$x_8 = a_8 \times x_7 + b_8 = \widehat{Q}(8,1)$$

即：$\widehat{Q}(1,1) = x_1 = b_1$

$$\widehat{Q}(2,1) = x_2 = a_2 \times b_1 + b_2 = a_2 \times \widehat{Q}(1,1) + \widehat{Q}(2,2)$$

…

$$\widehat{Q}(8,1) = x_8 = a_8 \times b_7 + b_8 = a_8 \times a_7 \times a_6 \times a_5 \times \widehat{Q}(4,1) + \widehat{Q}(8,5)$$

Kogge-stone 加法器中进位链 C_i 可以写成 $\widehat{Q}(1,1) = x_1 = b_1$，则 $C_1 \sim C_8$ 生成：$\widehat{Q}(1,1) = C_1$；$\widehat{Q}(2,1) = C_2 = p_2 C_1 \times g_2 = p_2 \widehat{Q}(1,1) \times g_2$ 等。

图 2 为 Kogge-stone 树形图。以 8 bit 加法器为例，其中 $g_1 \sim g_8$ 表示进位产生信号，$C_1 \sim C_8$ 表示产生每一位的进位信号，中间的红心圆表示 Kogge-stone 树状结构，存储产生进位信号 $C_1 \sim C_8$ 所需要的中间结果。

当使用四级结构时，可以实现 8 bit 输入，由此得出，每增加一级结构，输入增加两倍，且延时只增加一级加法运算的延时，所以采用此结构会使得加法器整体性能得到很大的提升，相比其他加法器，Kogge-stone 加法器得到优化。

2.2　改进的 Montgomery 算法

Montgomery 算法中，两个 1 024 bit 的数据相乘需要一个 1 024 × 1 024 bit 的乘法器和一个 2 048 bit 的加法器，中间的临时变量结果最大达到了 2 048 bit，实现所需要的资源开销很大，同时对硬件实现来说编译的时钟频率也无法保证。所以为了减小基 2 - Montgomery 算法中的延迟和硬件资源消耗，本文采用高效的 Kogge-stone 加法器架构，与基 2-Montgomery 算法形成一种混合算法，有效减少操作数的扇出量和延迟。基 2-Montgomery 算法与 Kogge-stone 加法器结合后见表 5。

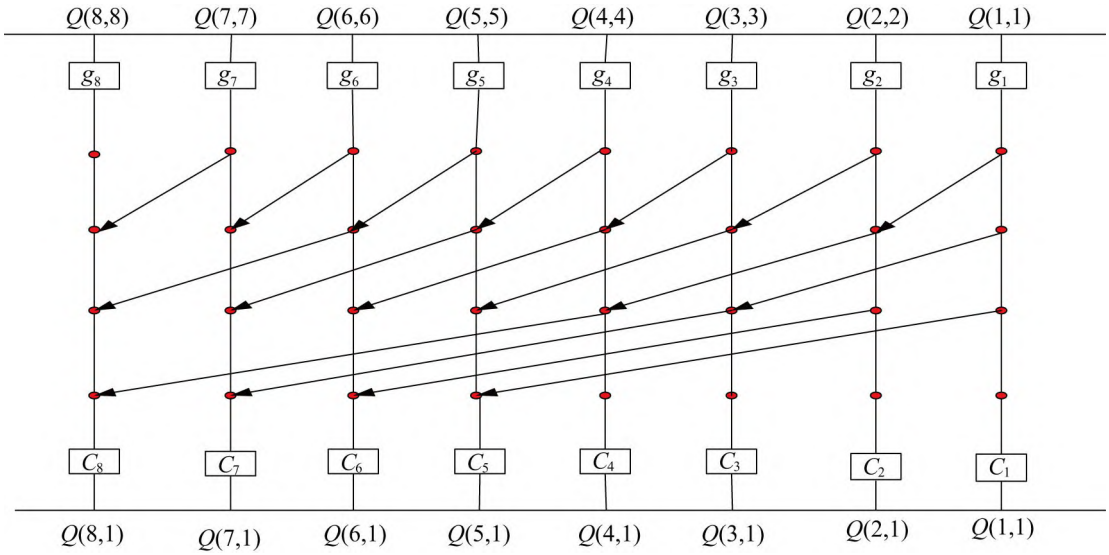

图 2　Kogge-stone 树形图

表 5　混合 Montgomery 算法

$int\ result=0;$
$bit\ carry=0,x$
$for\ i\ to\ n$
$q_i=r_0^i \oplus (a_i+b_0)$
$for\ j\ to\ n$
$start(a_i,q_i)$
$(1,1):x=m\ b_j$
$(1,0):x=b_j;$
$(0,1):x=b_j;$
$(0,0):x=0;$
$r_j^{i+1}=r_{j+1}^i \oplus x \oplus carry$
$end\ for$
$carry=(r_{j+1}^i and\ x_j)\ or(r_{j+1}^i and\ carry)\ or(carry\ and\ x_j)$
$end\ for$
$return\ result$

在表 5 中，第 6~10 行为 4 通道 1 位复用器，用来执行 a_i，q_i 和 Kogge-stone 加法器进行进位传播位的计算以及各进位传播位之和的结果。

Kogge-stone 加法接口与混合 Montgomery 算法接口示意图分别参见图 3、图 4。在图 3 中，输入参数 in_a 是被加数，in_b 是加数，shift 表示移位信号，result 表示结果，start 表示启动脉冲信号，done 表示计算完成信号；在图 4 中，in_a 是被乘数，in_b 是乘数，in_m 表示模数，result 表示结果，start 表示启动脉冲信号，done 表示计算完成信号。

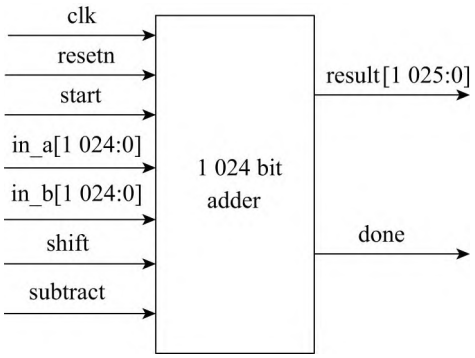

图 3　加法接口示意图

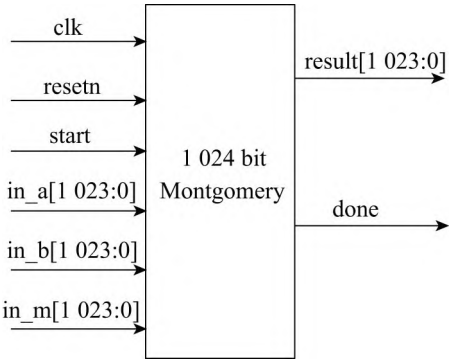

图 4　混合 Montgomery 算法接口示意图

在图 5 中，混合 Montgomery 算法的数据流表示，当输入信号准备好时，复位信号清除 3 个移位寄存器中的数据；同时，KSA_adder_1 加法器将输入 in_b 和 in_m 相加产生和，输入 in_a 加入 shift_reg 1 移位寄存器 1，同时处理单元 pe，它用来产生一个相应的输出有效值 q0，该信号主要取决于中间最低有效值 reg_rji、in_b[0]输入和 in_a[i]在各自时钟周期；当上述输出 out[0]~out[n+1]全部生成，KSA_adder_2 加法器进行 out 与 reg_rji 相加，然后 KSA_adder_2 产生的输出信号加载到 shift_reg 2 移位寄存器 2 中，shift_reg 2 根据 done 的信号将输出值加载到寄存器 3 中进行运算，得出运算结果，其中处理单元 pe 结构如图 6 所示。

3　RSA 算法中 Montgomery 的应用分析

在 RSA 算法中，核心运算是模幂运算，因为模幂运算基础表达式 *abmodn* 相当于[9]：先进行乘法，然后进行取模运算，所以提高模乘运算的速率对 RSA 算法有直接影响。通过对 *abmodn* 分析，可以得出取模运算是最复杂的，因为一次除法运算相当于进行多次加法、减法、乘法，且除法运算在计算机中运算十分复杂，所以如果能将运算中的除法减少或者将除法转变为其他运算甚至不用除法运算，RSA 算法的整体效率会有很大的提高。而 Montgomery 运算的出现较好地解决了此问题。

表 6 是三种 Montgomery 运算的表达式。三种表达式各有利弊，在大数 1 024 bit 及以上位宽时，普通 Montgomery 运算处理和存储的数值较大，会耗费过多的硬件资源；基 2-Montgomery 运算从优化面积角度出发，但在加法运算中会有延迟产生；基 8-Montgomery

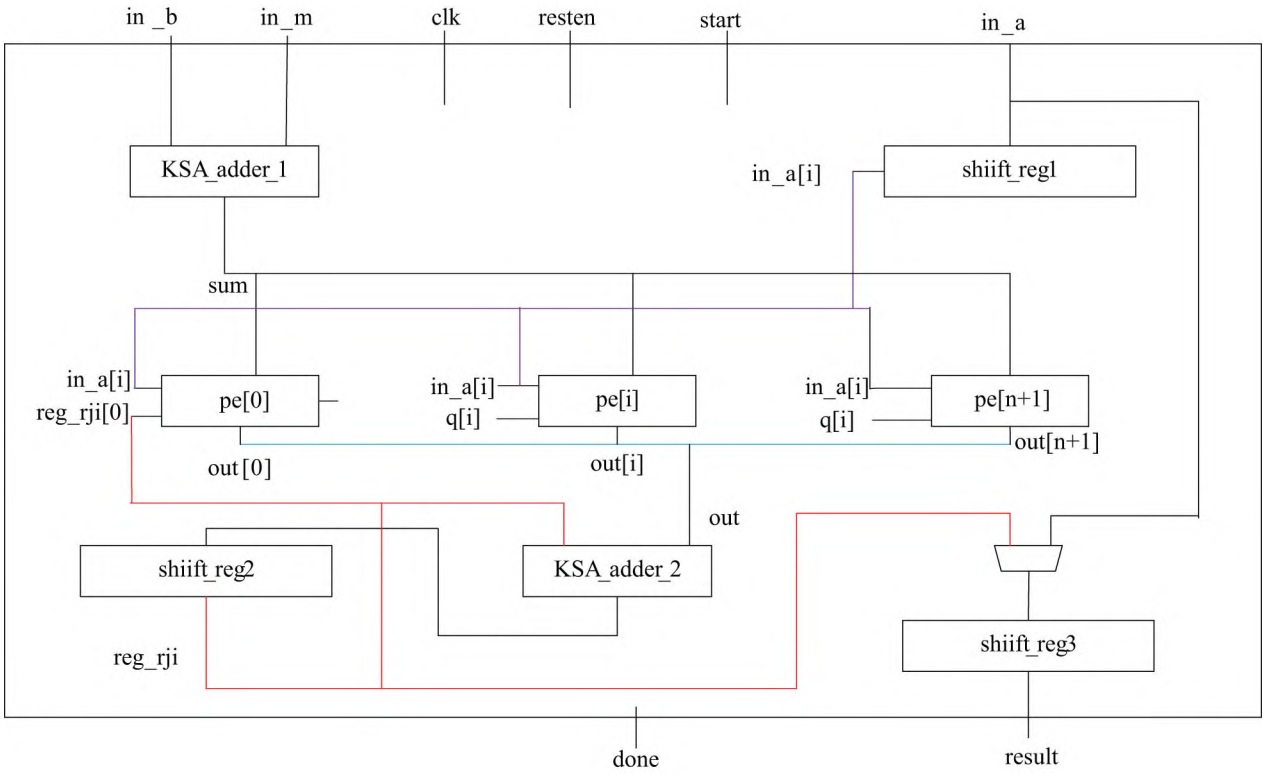

图 5　混合 Montgomery 算法数据流

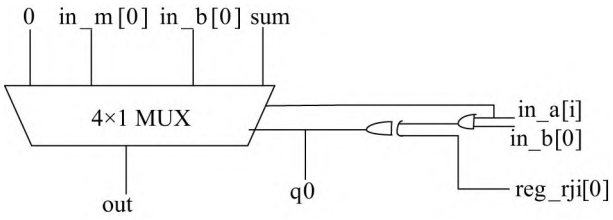

图 6　处理单元 pe 结构

表 6　Montgomery 运算三种表达式

算法名称	公式
普通 Montgomery	$Mon(A,B,r,p)=A\times B\times r^{-1} mod\ p$
基 2-Montgomery	$Mon\_2(A,B,p)=A\times B\times 2^{-n} mod\ p$
基 8-Montgomery	$Mon\_8(A,B,p)=A\times B\times 2^{-3(n-1)} mod\ p$

运算则会相应减少加法和乘法运算次数，但是资源则会相应增加。RSA 算法的加解密运算[10]可以表示为 $C=M^e mod\ p$，其中指数 e 的大小和基数的大小成反比，基数越大，指数越小，乘法运算层次越低；但并不是基数越大性能越好，需要充分考虑增加的运算对资源的占用情况。综上分析，本文从运算速率和硬件资源角度出发，选用基 2-Montgomery+KSA 加法器的混合运算来实现模乘运算，一方面引入基于优化硬件资源的基 2-Montgomery 实现形式，另一方面引入多比特并行前缀加法器，它有着高性能的并行处理方式，能有效减少大的扇出和在加法运算时产生的延迟。

4　大数模乘运算仿真与分析

本文仿真选用的软件为赛灵思公司的 Vivado，仿真采用 Vivado 软件自带的仿真器，选用 ZYNQ7020 系列芯片进行仿真测试。

4.1　仿真时间

本文在上述设计后对 1 024 bit 位 Kogge-stone 加法器和 KSA+Montgomery 混合模乘进行仿真验证。验证结果表明：本文运算时间为 111.15 μs。经汇总比对其他研究成果（见表 7、表 8），可以看出本文 1 024 bit 模乘运算时间进一步加快。

4.2　结果分析

如表 7 和表 8 所示，其中 LUT（查找表）和 FF（触发器）代表资源利用情况，可以看出本文采用 KSA+Montgomery 的混合模乘运算使用 5 158 LUT 和 10 491 FF，模乘运算速率相对于其他已有研究方式[11-14]均有

表 7　模乘运算资源对比结果

设计	LUT/FF 数量	优化方法	器件
本文	5 158/10 491	KSA+基 2-Montgomery	ZYNQ-7020
其他方式一	27 820	CSA+Montgomery	Virtex-5

表 8 模乘运算时间对比结果

设计	数据长度/bit	时间/μs	时钟周期	优化方法	器件
本文	1 024	111. 15	735	KSA+基 2-Montgomery	ZYNQ-7020
其他方式一	1 024	—	1 028	CSA+Montgomery	Virtex-5
其他方式二	1 024	452. 49	22 634	SMM 方法	EP3SL340F1760C
其他方式三	1 024	2 200	—	普通 Montgomery	—
其他方式四	1 024	90 350	—	大整数平方改进后的 Montgomery	—

提高,可以更好地平衡模乘运算速率和资源利用。

4.3　本文算法优势

本文采用一种 KSA+Montgomery 的混合算法来实现模乘运算,主要优势体现在模乘运算速率和硬件资源利用率两个方面。在运算速率方面,本文采取 KSA 结构的加法器,可以有效减少乘法运算时调用加法时产生的延迟时间,本文运算速率相比普通 Montgomery 模乘运算提高约 94%;在硬件资源利用率方面,本文实现的模乘运算占 ZYNQ-7020 开发板中 LUT 和 FF 资源分别为 9. 7%和 10%,相较于其他运算,有效减少了硬件资源占用情况。

5　结束语

本文对 RSA 算法中模乘运算进行研究,提出一种基 2-Montgomery 算法和 Kogge-stone 加法器相结合的混合运算,并且构建了 1 024 bit Kogge-stone 加法器,该算法采用了 Kogge-stone 加法器,有效减少基 2-Montgomery 算法中加法延迟的问题。仿真结果表明,在数据位宽为 1 024 bit 时,该算法与其他模乘运算相比较,大大提高了模乘的运算速率。

RSA 算法广泛应用于信息加解密、签名验签等数据交互的工作中,现实中对 RSA 算法的计算速率有一定的要求。对于 RSA 算法来说,大数模乘的运算时间大致决定了模幂运算的时间,从而确定整个 RSA 算法的运算时间,综合考虑其他因素导致的运算延迟和资源消耗,应用本文的混合模乘运算来实现 RSA 算法,应会提高 RSA 算法整体运算速率。

参考文献

[1] 邬贵明, 谢向辉, 吴东, 等. 高基 Montgomery 模乘阵列结构设计与实现[J]. 计算机工程与科学, 2014,36(2):201-205.

[2] 车文洁, 董秀则, 高献伟, 等. Montgomery 模乘法器的实现与优化[J]. 计算机应用与软件, 2017,34(3):312-315+333.

[3] 彤丽, 姜明富. RSA 加密方式中 Montgomery 算法的研究与改进[J]. 信阳农业高等专科学校, 2013,23(4):107-109.

[4] 高献伟, 张晓楠, 董秀则. 基于 FPGA 的 Montgomery 模乘器的高效实现[J]. 计算机应用研究, 2017,34(11):3424-3427.

[5] 李泉龙. 基于 Kogge-Stone 算法与多米诺逻辑的 64 位高性能加法电路设计[D]. 成都:西南交通大学, 2016.

[6] ZHAO S L, HUANG H, LIU Z W, et al. An efficient signed digit montgomery modular multiplication algorithm [J]. Microelectronics Journal, 2021(3):105099.

[7] PARIHAR A, NAKHATE S. High-speed high-throughput VLSI architecture for RSA montgomery modular multiplication with efficient format conversion[J]. Journal of The Institution of Engineers (India): Series B, 2019, 100(2):217-222.

[8] LEE M X, MUHAMMAD M A Z, AINY H A, et al. VLSI implmentation of a fast Kogge-stone parallel-prefix

adder[J]. Journal of Physics Conference, 2018:1049.
[9] 邵佳佳, 乌力吉, 张向民. 智能卡中非对称算法模幂运算单元的设计与验证[J]. 微电子学与计算机, 2015,32(2):37-41.
[10] 程碧倩, 刘光柱, 肖昊. 改进的蒙哥马利模乘算法及FPGA 实现[J]. 电子科技, 2022,35(7):58-63.
[11] 王开宇, 唐祯安, 赵赟, 等. 基于改进 CSA-蒙哥马利的RSA 加密处理器实现[J]. 大连理工大学学报, 2013,53(2):294-297.
[12] 孙建林. 基于 FPGA 的 RSA 快速加密算法的改进[D]. 保定:河北大学, 2016.
[13] 陈逢林, 苏厚勤. Montgomery 算法的改进及其在RSA中的运用[J]. 计算机应用与软件, 2006, 23(6):109-111.
[14] 靳蓓蓓, 张仕斌. Montgomery 模幂运算的一种改进方案[J]. 长春大学学报(自然科学版), 2006, 16(4):48-51.

作者简介:

贾斌斌 中北大学电气与控制工程学院硕士研究生在读,主要从事可信计算中国密算法研究等方面的研究工作

王忠庆 中北大学电气与控制工程学院副教授,硕士研究生导师,主要从事火炮自动调平系统、高精度步进电机位置控制系统的开发等方面的研究工作

方炜 中北大学电气与控制工程学院副教授,主要从事嵌入式开发、智能控制和工控系统可信安全等方面的研究工作

Thoughts on improving rate of large number modular multiplication in RSA algorithm

JIA Binbin, WANG Zhongqing, FANG Wei

(School of Electrical and Control Engineering, North University of China, Taiyuan 030051, China)

Abstract: The core of RSA algorithm is large number modular multiplication. Improving its operation speed is not only of great significance to the improvement of RSA algorithm, but also will have a certain positive effect on the promotion of e-commerce if the speed can be achieved quickly and cheaply through application specific integrated circuit. Based on the study of Montgomery algorithm, this paper proposes a basic 2-Montgomery modular multiplication operation based on the parallel prefix adder architecture, and constructs a 1 024-bit Kogge-stone adder. The simulation results show that the proposed method can effectively reduce the delay time of operand and improve the efficiency of large number modular multiplication to a certain extent.

Keywords: RSA algorithm; basic 2-Montgomery; Montgomery algorithm; Kogge-stone adder

(收稿日期:2022-08-10)