文章编号：1000-8608(2013)02-0294-04

基于改进CSA-蒙哥马利的RSA加密处理器实现

王开宇*，唐祯安，赵　赟，周煜迪

（大连理工大学 电子信息与电气工程学部，辽宁 大连　116024）

摘要：为了提高RSA加密算法的速度和灵活度，对RSA的算法原理和实现方法进行了深入的研究，对普遍采用双CSA的蒙哥马利模乘算法进行了改进.将其中一个CSA的结构进行改变，且优化掉原先CSA-蒙哥马利算法结构中的CPA，不但减少了FPGA资源的使用，而且提高了RSA的处理速度.最后，实现了密钥长度可配置的RSA加密处理器，并对密钥长度为1 024 bits的加密处理器作了性能参数分析.

关键词：RSA；蒙哥马利；进位存储加法器

中图分类号：TP309.7　　**文献标志码：**A

0　引　言

随着信息化时代的不断推进，电子商务、电子通信等诸多领域的信息安全受到越来越多的关注.为了保护电子信息，各种加密算法被应用在信息安全中.其中应用较广泛的RSA加密算法，是一种非对称加密算法，即加密密钥和解密密钥是分开的.RSA在FPGA上的实现主要是利用蒙哥马利算法将求幂取余运算转换成硬件上容易实现的移位和相加运算，本文对算法结构做一些改进，以降低器件资源的使用，提高算法速度，最终在FPGA上实现并进行验证.

1　RSA算法原理介绍

RSA是以Rivest、Shamir、Ad-leman这3位提出者的名字而命名的.其安全性主要是依靠大数的难以分解性质.

首先，选择两个大素数 P、Q，计算 $N=P\times Q$，然后选择一个非$(P-1)$和$(Q-1)$因子的公钥E，通过$(D\times E)\bmod(P-1)(Q-1)=1$，得出私钥$D$.设明文为A，密文为$B$，则加密过程为$B=A^E \bmod N$，相似地，解密过程为$A=B^D \bmod N$.由此可见，加密和解密运算都是求幂取余，当数很大时，如果直接计算幂再取余，无论在硬件还是软件上实现都是不可能的.蒙哥马利算法正是将其转化成移位和相加并且幂和取余运算同时进行，利于在硬件上实现.

2　蒙哥马利算法及其改进

蒙哥马利算法的基本思想是通过移位和相加来实现模乘运算，文献[1-2]中对蒙哥马利算法做了些改进，改进后的算法如下：

算法1

Montgomery Multiplication (A,B,N)

Step 1　$S[0]=0$

Step 2　FOR i in 0 TO $k-1$ LOOP

$t_i=(s[i]_0+A_i\cdot B_0)\bmod 2$

$s[i+1]=(s[i]+A_i\cdot B+t_i\cdot N)$ div 2

END LOOP

Step 3　IF $S[k]>N$ THEN $S[k]=S[k]-N$

RETURN $S[k]$

收稿日期：2011-12-25；　**修回日期：**2013-01-10.
基金项目：大连理工大学基本科研业务费资助项目(DUT10JR01).
作者简介：王开宇*(1973-)，男，副教授，E-mail：wkaiyu@dlut.edu.cn.

其中 $A=\sum_{i=0}^{k-1} A_i \cdot 2^i$，$B=\sum_{i=0}^{k-1} B_i \cdot 2^i$，$N=\sum_{i=0}^{k-1} N_i \cdot 2^i$，$S[k]=A \cdot B \cdot 2^{-k} \bmod N$. 为了避免算法第 3 步中 $S[k]$ 大于 N 的出现，可以增加两次循环，即（$k+2$）次，并且将 A 调整为 $A=\sum_{i=0}^{k+2} A_i \cdot 2^i$，其中 $A_k=A_{k+1}=A_{k+2}=0$，因此调整后 $S[k]=A \cdot B \cdot 2^{-(k+2)} \bmod N$，调整后的算法如下：

算法 2

Montgomery Multiplication (A, B, N)

Step 1　S[0]=0

Step 2　FOR i in 0 TO $k+1$ LOOP

$t_i=(s[i]_0+A_i \cdot B_0) \bmod 2$

$s[i+1]=(s[i]+A_i \cdot B+t_i \cdot N)$ div 2

END LOOP

Step 3　RETURN S[k]

2.1　采用 CSA 的蒙哥马利算法

当位数很大时，算法 2 第 2 步中的加法运算会产生很大的延时，而采用 CSA（Carry-Save-Adder）可以有效地避免长进位链所带来的延时[3]. CSA 中 FA 的逻辑如下：

SUM= X1 xor X2 xor X3

CARRY=（X1 and X2）or（X1 and X3）or（X2 and X3）

CSA 由 k 个 FA 构成，如图 1 所示.

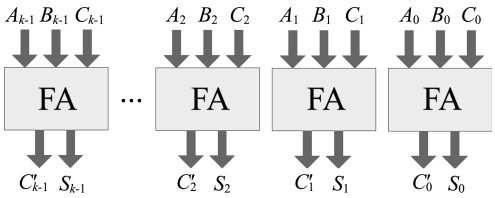

图 1　CSA 的结构

Fig. 1　The structure of CSA

因此采用 CSA 的算法如下：

算法 3

Montgomery Multiplication (A, B, N)

Step 1　S[0]=0，SUM=0，CARRY=0

Step 2　FOR i in 0 TO $k+1$ LOOP

(SUM, CARRY)=CSA(SUM+CARRY+$A_i \cdot B$)

(SUM, CARRY)=CSA(SUM+CARRY+$SUM_0 \cdot N$)

SUM=SUM/2

CARRY=CARRY/2

END LOOP

Step 3　S[k]=SUM+CARRY

RETURN S[k]

在算法 3 第 3 步中含有一个大数的加法，因此文献[4]使用了 CPA（Carry-Propagation-Adder）利用多个时钟周期来完成最后的模乘输出. 算法结构如图 2 所示.

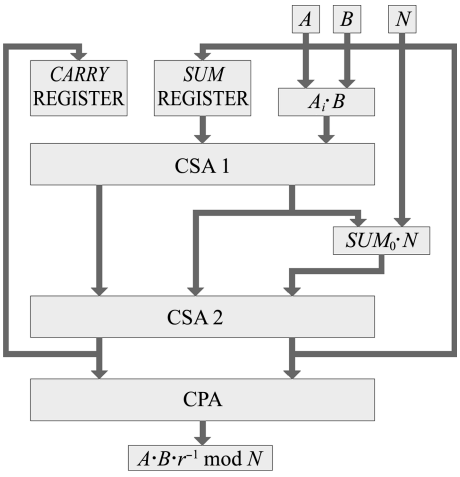

图 2　CSA-蒙哥马利算法结构

Fig. 2　The structure of CSA-Montgomery algorithm

2.2　本文对 CSA-蒙哥马利算法的改进

从 CSA-蒙哥马利算法中可以看出，每次循环中所加的数为 0、B、N、B+N 中的一个，因此将其中一个 CSA 的结构进行改变，即 CSA_ADD，如图 3 所示，并且利用其在第一个时钟周期计算出 B+N，通过{SUM_0, A_i}来判断 CSA 的下一个输入值，循环最后（SUM+CARRY）的值可以再次利用 CSA_ADD 来计算，舍去了 CPA 的使用，不但提高了速度，而且减少了资源使用. 改进后的算法结构如图 4 所示.

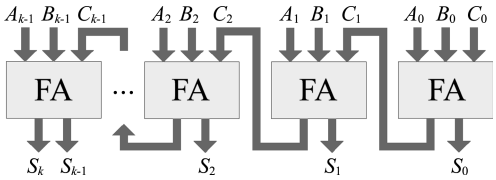

图 3 CSA_ADD 的结构

Fig. 3 The structure of CSA_ADD

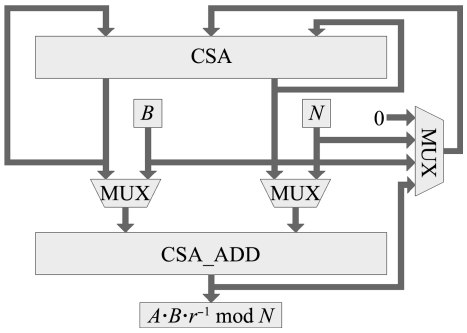

图 4 改进后的 CSA-蒙哥马利算法结构

Fig. 4 The structure of modified CSA-Montgomery algorithm

改进后的算法如下：

算法 4

Montgomery Multiplication (A, B, N)

Step 1 S[0]=0, SUM=0, CARRY=0, B_N=0

B_N=CSA_ADD(B, N)

Step 2 FOR i in 0 TO k+1 LOOP

(SUM, CARRY) = CSA (SUM + CARRY+0/B/N/B_N)

SUM= SUM/2

CARRY= CARRY/2

END LOOP

Step 3 S[k]=CSA_ADD(SUM, CARRY)

RETURN S[k]

3 RSA 的算法实现

本文采用从右到左的方式扫描二进制的指数 E，由于蒙哥马利算法的返回值是 $A \cdot B \cdot 2^{-(k+2)} \bmod N$，为了得到 $A \cdot B \bmod N$，必须先进行一步预处理 $Z = M\_M(M, R, N)$，$P = M\_M(1, R, N)$，其中 R 是与 N 相关的常数，$R = 2^{2(k+2)} \bmod N$，最后再进行一次后处理 $P = M\_M(1, P, N)$，即得到 $P = M^E \bmod N$，其中 M 表示明文，E 表示指数，N 表示模数. 具体步骤如下：

Montgomery Exponent (M, E, R, N)

Step 1 $Z = M\_M(M, R, N)$, $P = M\_M(1, R, N)$

Step 2 FOR i in 0 TO $k-1$ LOOP

IF($E_i = 1$) THEN $P = M\_M(Z, P, N)$

$Z = M\_M(Z, Z, N)$

END LOOP

Step 3 $P = M\_M(1, P, N)$

RETURN P

4 性能分析

本文利用 Verilog 实现了密钥位数可调的 RSA 加解密处理器，利用 Modelsim 仿真工具对 RSA 核进行了仿真，并在 XILINX 的 VIRTEX5 上得到了验证. 如图 5 所示，当 $P = M^E \bmod N = 6^9 \bmod 11 = 2$，与仿真结果一致.

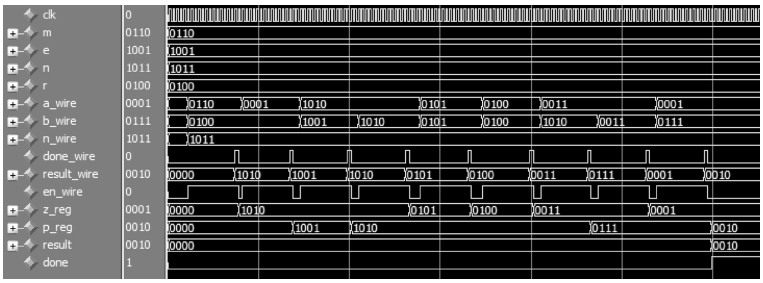

图 5 RSA 核的仿真

Fig. 5 The simulation of RSA core

当加密处理器配置为 1 024 位时，一次蒙哥马利模乘运算需要 1＋1 026＋1＝1 028 个时钟周期. 如果指数 E 的 0、1 出现次数相当，则一次 RSA 加解密需要大约 1 024＋500＋3＝1 527 次模乘运算，最差情况需要 1 024×2＋3＝2 051 次模乘运算，则最差需要 2.1×10^6 个时钟周期，相比文献[5]减少了 2.84%. 当时钟周期为 120 MHz 时，1 s 内可以进行约 75 次加解密运算. 密钥长度为 1 024 bits 时的性能参数如表 1 所示.

表 1　密钥长度为 1 024 bits 的加密处理器性能参数

Tab. 1　The performance of cryptoprocessor with 1 024 bits key length

位宽/bit	最大时钟/MHz	面积/LUTs	一次 RSA 耗时/ms
1 024	117.5	27 820	13.49

5　结　语

本文对采用 CSA 的蒙哥马利模乘算法结构进行了改进，省去了 CPA 的使用，不但节省了器件资源，而且提高了算法速度. 同时，利用 Verilog 编写了可配置密钥长度的 RSA 算法的 IP 核，应对不同工程的需要，可以灵活地改变密钥长度.

参考文献：

[1] Manochehri K. Modified radix-2 Montgomery modular multiplication to make it faster and simpler [C] // **Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC′05)**. Piscataway: IEEE Computer Society, 2005: 598-602.

[2] Blum T. Montgomery modular exponentiation on reconfigurable hardware [C] // **Proceedings of 14th IEEE Symposium on Computer Arithmetic**. Piscataway: IEEE Computer Society, 1999: 70-77.

[3] McIvor C, McLoone M, McCaany J V. Fast Montgomery modular multiplication and RSA cryptographic processor architectures [C] // **Conference Record of the Thirty. Seventh Asilomar Conference on Signals, Systems and Computers**. Piscataway: IEEE, 2003: 379-384.

[4] 王　旭，董　威，戎蒙恬. 基于改进 Montgomery 模乘算法的 RSA 加密处理器的实现[J]. 上海交通大学学报，2004，**38**(2)：240-243.
WANG Xu, DONG Wei, RONG Meng-tian. Implementation of RSA cryptoprocessor based on modified Montgomery modular multiplication algorithm [J]. **Journal of Shanghai Jiaotong University**, 2004, **38**(2): 240-243.

[5] Kwon Tack-won, You Chang-seck, Heo Won-seok, *et al*. Two implementation methods of a 1024-bit RSA cryptoprocessor based on modified Montgomery algorithm [C] // **IEEE International Symposium on Circuits and Systems-ISCAS**. Piscataway: IEEE, 2001: 650-653.

Implementation of RSA cryptoprocessor based on modified CSA-Montgomery

WANG Kai-yu*, TANG Zhen-an, ZHAO Yun, ZHOU Yu-di

(Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China)

Abstract: To improve the speed and flexibility of RSA encryption algorithm, the basal principle and implementation methods of RSA are studied in detail. The common two-CSA-Montgomery multiplication algorithm structure is modified by removing the CPA of CSA-Montgomery algorithm structure. It not only reduces the resources of FPGA, but also improves the speed of RSA algorithm. Finally, the RSA cryptoprocessor that key length is reconfigurable is realized. And the performance of cryptoprocessor with 1 024 bits key length is also analyzed.

Key words: RSA; Montgomery; CSA