第35卷 第22期 计 算 机 工 程 2009年11月
Vol.35 No.22 Computer Engineering November 2009
·工程应用技术与实现· 文章编号：1000—3428(2009)22—0224—03 文献标识码：A 中图分类号：TP301.6

基于 Montgomery 的 RSA 高速低成本实现

王 辉，刘宏伟，张慧敏
(北京科技大学信息工程学院，北京 100083)

摘 要：给出一种支持多种位数 RSA 算法加密芯片的完整设计方案。采用改进的 Montgomery 模乘算法和 LR 模幂算法，根据大数运算的特点和降低资源消耗的需要改进主要运算电路的结构，并采用全定制 IC 的设计流程进行实现。实验结果表明，该方案结构简单，节省了面积，且能达到较高的性能。

关键词：RSA 算法；模乘；模幂；进位保留加法器；Booth 编码；超前进位加法器

High Speed and Low Cost Realization of RSA Based on Montgomery

WANG Hui, LIU Hong-wei, ZHANG Hui-min
(School of Information Engineering, University of Science & Technology Beijing, Beijing 100083)

【Abstract】 This paper proposes displays a complete chip design for multiple-digit RSA encryption algorithm. This design uses improved Montgomery modular multiplication algorithm and LR modular exponentiation algorithm. According to the characteristics of computing of large numbers and the need of reducing consumption of resources, this design improves the main operation circuit structure by using full-customed IC design process to realize. Experimental results show that the structure design is simple to realize and can save space and achieve high performance.

【Key words】 RSA algorithm; modular multiplication; modular exponentiation; Carry Save Adder(CSA); Booth encoding; Carry Look-ahead Adder (CLA)

1 概述

RSA 加密算法是目前在理论和实际应用中较为成功的一种公钥密码体制，由于 RSA 的简单性、强安全性和密钥管理的方便性而被广泛采用。大数模幂运算是很多公钥密码体制的核心运算，它由一系列的模乘运算构成。大数位数需要在数百位以上，运算量极大，是提高加解密运算速度的瓶颈。当前国内外有很多基于 RSA 加密算法开发的硬件算法加速器，但大多数的实现都存在以下缺陷：

(1)RSA 运算处理器芯片的加密速度太慢，达不到一般应用标准(＜10 Kb/s)；

(2)芯片用到的门数过多，使得所占硅片面积太大，功耗大，成本过高。

针对上述设计中的不足，本文提出了一款专用 RSA 协处理器。协处理器中的核心部件是一个多功能的快速模乘器，它采用改进的 Montgomery 算法，使用并行流水技术，大大减少了一次模乘的运算时间，并且具有较小的面积，很好地控制了成本。

2 算法实现

2.1 模乘算法及其优化

在整个 RSA 加密系统中，大数模乘运算是最主要的运算，它决定了整个加密系统的效率。原始的 Montgomery 模乘算法[1]可以描述成如下形式。

算法 1 原始的 Montgomery 模乘算法

设 N 为模数且 $N>1$，R 是与 N 互素的一个基，通常 $R=2^t$，t 是 N 的位数；$R-1$ 和 N 满足 $0<R-1<N$，$0<N[0]'<R$，$RR^{-1}-NN[0]'=1$，即 $RR^{-1}(\mathrm{mod}\ N)=1$ 或 $NN[0]'(\mathrm{mod}\ N)=-1$；对给定大整数 T，且 $0\le T<R\times N$ 的 Montgomery 算法如下：

Step1 $m\leftarrow(T\ \mathrm{mod}\ R)N[0]'\mathrm{mod}\ R$

Step2 $u\leftarrow(T+m\times M)/R$

Step3 if $u\ge N$ then $u-N$ else return u

由于原始的 Montgomery 模乘算法中包含的乘法和加法运算全部是大数运算，因此 LSI 实现时硬件的代价很大。而且，由于进位链太长，关键路径延时很大，制约了系统的时钟频率。本文采用改进的 Montgomery 模乘算法可以有效地解决这些问题。此模乘算法可以描述成如下形式。

算法 2 改进的 Montgomery 模乘算法

假设 A 和 B 是 2 个余数，而整数 $r=2$；C_i 保存结果的高 32 位；S_i 保存结果的低 32 位；预计算 $N[0]'$，使得 $N[0]\times N[0]'(\mathrm{mod}\,2^{32})=2^{32}-1$；将模数 N 保存在 N 的数组中，$N[0]$ 是模数 N (2 048 位)的最低 32 位。算法代码如下。

```
MonPro(A,B,N) = A*B*2^-t(mod N)
Step1 MontProMul(A ,B ,N )
  T=0; C2 = 0;
  for i = 0 to e–1 do
    C1 = 0;
    for j = 0 to e–1 do
      if (j = 0) begin
        {C1,S1} = A[0]B[i] + C1 +T[j];
        OP = S1*N[0]'(mod 2^w);
        {C2,S2} = N[0]*OP + S1;
      end
```

作者简介：王 辉(1982－)，男，硕士研究生，主研方向：系统芯片；刘宏伟，副教授；张慧敏，硕士研究生

收稿日期：2009-04-30 **E-mail：**leyan24@163.com

```
        else begin
          {C1,S1} = A[j]B[i] + C1 +T[j];
          {C2,S2} = N[j]*OP + C2 + S1;
          T[j–1] = S2;
        end
      end for
      {C2,S2} = C2 + C1 + T_carry2;
      T[e–1] = S2; T_carry2= C2[0];
    end for
  Step2 ResultReturn (T, N)
    if (Cmp) begin
      for j = 0 to e–1 do
        T[j] = T[j] – N[j];
      end for
    end
    else begin
      for j = 0 to e–1 do
        T[j] = T[j] – 0;
      end for
    end
```

其中，w 代表字长，本文中 w=32；e 代表 N 的字数；Cmp 为 T 和 N 的比较大小的结果。算法复杂度：$(e\times(e+1)+3)\times$字乘周期数，本文字乘周期数为 1。

可以看出，在改进的算法中，位数很长的大数被分解为位长相对较小的数来分别进行计算，改善了由大数的加法和乘法进位链过长引起的问题。

2.2 模幂算法

针对模幂的算法有 3BR 算法[2](分为 LR 算法和 RL 算法)，2^k 进制算法，滑动窗口法。因为幂指数优化编码大约只能获得 20%左右的效果，而且是以牺牲大量的存储空间为代价，所以在模幂的硬件实现上，采用了 LR(从左到右)方法。

算法 2 LR 模幂算法

ModExp(M,e,N)= M^e mod N

Input: M, e, N

Output: C = M^e mod N

Step1 if e[t–1] = 1 then C = M else C = 1

Step2 for i = t–2 to 0

Step2.1 C = C*C (mod N)

Step2.2 if e[i] = 1 then C = C*M (mod N)

Step3 return C

这种方法是从 e 的最高位扫描到最低位，对扫描 e 的每一比特需要执行一次模平方，只有扫描到 e 为 1 的时候，才执行一次模乘。RL 模幂算法的最坏复杂度为$(2n+3)$次模乘，平均情形为$(1.5n+3)$次。

3 LSI 硬件实现

RSA 协处理器包含 4 个部分：数据/命令输入输出接口，基于 Montgomery 算法的模乘运算单元，处理数据传输和调度的控制电路以及数据存储电路。

(1)接口电路

接口电路部分负责数据的接收和返回，并且对 CPU 发来的命令译码，得知 RSA 协处理器所要执行的操作。

(2)模乘运算单元

采用了 2 个模乘器来计算，将改进模乘算法中流水执行，大大提高了运算速度。模乘器中可以完成 a[31:0]×b[31:0]+c[31:0]+d[31:0]+$carry$ 和 a[31:0]×b[31:0]+c[31:0]-d[31:0]-$carry$ 的功能。

(3)控制电路模块

控制电路部分负责输入时的数据选通、数据的存储、状态机的控制及最终结果的返回。

(4)数据存储

数据存储部分除了用于存放中间结果的 RAM 外，还需要一些寄存器作为输入时的缓冲和输出时的暂存以及运算过程中一些中间变量的暂存。

RSA 协处理器的总体结构如图 1 所示。

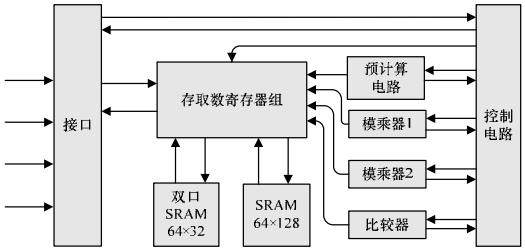

图 1 总体结构

3.1 Montgomery 模乘器

处理 Montgomery 算法中的加法长进位链问题，主要有以下 2 种方法：(1)心动阵列结构[3]；(2)保留进位加法器(CSA)结构[4]。速度和面积是硬件实现的 2 个重要指标。心动阵列结构虽然可以大幅提高速度，但是以牺牲大量面积为代价；而 CSA 结构可以在两者之间达到一种很好的平衡。因此，采用 CSA 来压缩 Booth 编码后产生的部分积，而 64 位 CLA 则用来计算出最后的加法结果。乘法累加器全部由组合逻辑设计实现，一个时钟周期就能计算出一个乘累加结果。模乘器结构如图 2 所示。

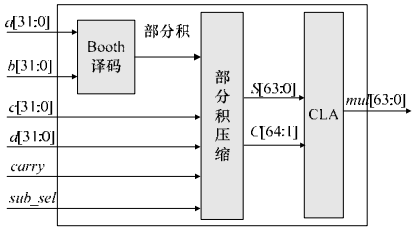

图 2 模乘器结构

3.2 Booth 编码

为实现高速的模幂，采用 3 位的 Booth 编码技术。为了适应 Booth 编码的需要，乘数 B 需要进行扩展，最低位需要扩展 1 位 0，最高位符号扩展 2 位，最终扩展成 n+3 位的数。然后根据表 1 进行 Booth 编码，生成部分积。

表 1 Booth 编码操作列表

B_{2i-1}	B_{2i}	B_{2i+1}	对被乘数操作(部分积)
0	0	0	部分积= $A\times0$
0	0	1	部分积= $A\times(+1)$
0	1	0	部分积= $A\times(+1)$
0	1	1	部分积= $A\times(+2)$
1	0	0	部分积= $A\times(-2)$
1	0	1	部分积= $A\times(-1)$
1	1	0	部分积= $A\times(-1)$
1	1	1	部分积= $A\times0$

3.3 CSA 部分积压缩

为了平衡各个部分的延时，减少关键路径上的时延，采用 Wallace 树结构的 CSA 压缩排列。Pi 为 Booth 编码出的部分积，同时将加数 c 和 d 也作为部分积进行压缩。Wallace 树结构如图 3 所示。

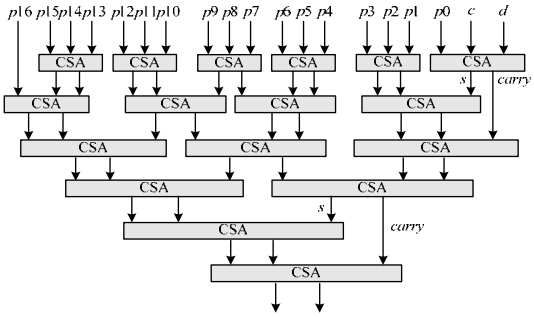

图 3　Wallace 树结构

3.4　高速加法器

为进一步提高速度，采用 64 位 CLA，并用 4 位一组的分组结构，平衡进位传递时间，使传递进位时间和计算结果同时计算出来，如图 4 所示。

在面积方面，4 位一组的 CLA 的最终速度取决于最高位进位产生的速度，而其他的进位产生速度并不直接影响整体 CLA 的速度。

因此，采用图 5 的改进型先行进位链，它的特点是低位的进位全部由行波进位方式产生，这样逻辑电路大为简化，既有利于后期的布线，又能降低成本。

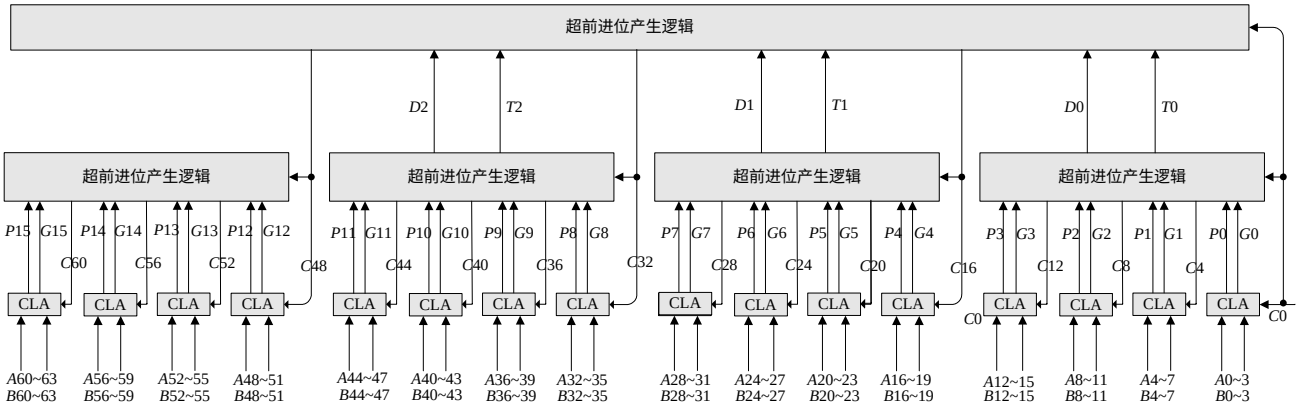

图 4　64 位 CLA 加法器结构

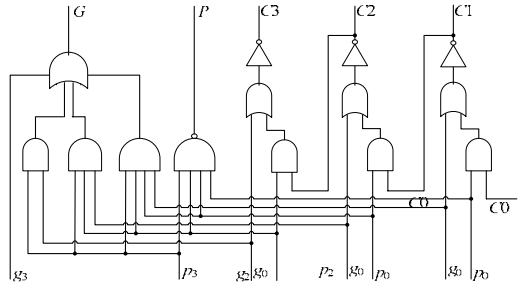

图 5　改进型先行进位链

4　性能分析

FPGA 的 CLB 和门数之间的换算没有固定的公式，依据经验大致估计 1 CLB=10 gate，可以看出该设计在面积和速度上均有优势，而且还取得了很好的平衡。一些 RSA 的实验结果比较如表 2 所示[5-9]。

表 2　一些 RSA 的实验结果比较

设计	年份	频率/MHz	位长/bit	吞吐率/(Kb·s^{-1})	规模
文献[5]	2003	100.5	1 024	94.9	22.0 kCLBs
文献[6]	2003	250.0	1 024	227.0	126 kgate
文献[7]	2004	111.3	1 024	106.0	11.5 kCLBs
文献[8]	2005	129.0	1 024	82.0	7.9 kCLBs
文献[9]	2005	63.0	1 024	39.9	4.0 kCLBs
本文	2008	100.0	1 024	180.8	50.7 kgate
本文	2008	100.0	2 048	47.7	50.7 kgate

5　结束语

本文采用了改进的 Montgomery 模乘算法和 LR 模幂算法，根据大数运算的特点和降低资源消耗的需要改进了主要运算电路的结构，并采用全定制 IC 的设计流程进行实现，使得 RSA 在面积和性能方面达到了很好的平衡。

参考文献

[1] Montgomery P L. Modular Multiplication Without Trial Division[J]. Mathematics of Computation, 1985, 44(1): 519-521.

[2] Knuth D E. The Art of Computer Programming Volume 2: Seminumerical Algorithms[M]. Beijing, China: Tsinghua University Press, 2002.

[3] Wu Chung-Hsien, Hong Jin-Hua, Wu Cheng-Wen. RSA Cryptosystem Design Based on the Chinese Remainder Theorem[C]//Proc. of Asia South Pacific Design Automation Conference. New York, USA: [s. n.], 2001: 39l-395.

[4] Kwon T W. Two Implementation Methods of a 1 024 Bit RSA Cryptoprocessor Based on Modified Montgomery Algorithm[C]// Proc. of the IEEE International Symposium on Circuits and Systems. New York, USA: [s. n.], 2001: 650-653.

[5] Mclvor C, McLoone M, McCanny J, et al. Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures[C]//Proc. of the 37th Asilomar Conference on Signals, Systems, and Computers. New York, USA: [s. n.], 2003: 379-384.

[6] Gong Peijun. A 1 024 Bit RSA Cryptosystem Hardware Design Based on Modified Montgomery's Algorithm[C]//Proc. of the 5th International Conference. Piscataway, USA: IEEE Press, 2003: 1296-1299.

[7] McIvor C, McLoone M, McCanny J V. Modified Montgomery Modular Multiplication and RSA[J]. Exponentiation Techniques, 2004, 151(6): 402-408.

[8] Fournaris A P, Koufopavlou O. A New RSA Encryption Architecture and Hardware Implementation Based on Optimized Montgomery Multiplication[C]//Proc. of International Symposium on Circuits and Systems. [S. l.]: IEEE Press, 2005: 4645-4648.

[9] Hamalainen P, Liu Ning, Marko H, et a1. Acceleration of Modular Exponentiation on System-on-a-Programmable-Chip[C]//Proc. of IEEE International Symposium on System-on-Chip. Tampere, Finland: [s. n.], 2005: 14-17.

编辑　顾逸斐