

一种 Montgomery 模乘算法硬件结构

王缔邴, 白国强, 陈弘毅
(清华大学 微电子学研究所, 北京 100084)

摘要: 基于二进制多字 Montgomery 模乘算法, 提出了一种参数可灵活配置的规则的脉动阵列硬件结构, 并使用此结构在 FPGA 上实现了不同位宽的 Montgomery 模乘算法. 该结构成功地在不增加额外电路或运行周期的情况下, 将脉动阵列的关键路径限制在运算单元内部的加法器中. 硬件实现结果表明, 该结构具有更高的电路频率、更少的电路面积消耗及算法运算时间.

关键词: RSA; Montgomery 模乘算法; 硬件; 脉动阵列

中图分类号: TN4

文献标识码: A

文章编号: 1000-7180(2010)05-0001-04

Hardware Architecture for the Montgomery Multiplication Algorithm

WANG Di-li, BAI Guo-qiang, CHEN Hong-yi

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: In this paper, regular and flexible hardware architecture based on the systolic array for implementing the multiple-word radix-2 Montgomery multiplication algorithm is proposed, and it has been used to implement the algorithm in FPGA for different bit-widths. The architecture successfully limits the critical path of the systolic array to the critical path of the adder in a processing element, without any additional circuits or clock cycles needed. According to the hardware implement results, the proposed architecture has higher frequency, less latency and less area.

Key words: RSA; montgomery multiplication algorithm; hardware; systolic array

1 引言

随着公钥密码的提出, 公钥密码在信息安全领域中的应用越来越广泛. 与对称密码算法相比, 使用公钥密码算法除了能保证通信数据的机密和完整性以外, 还能提供身份认证和不可否认性. 同时, 随着集成电路的发展, 密码算法的硬件实现已经具有信息安全方面的重要意义, 目前有公交卡、USB Key 等很多实际应用. 针对密码算法的硬件设计, 如何减小电路面积消耗以及算法运算时间一直是研究的热点.

目前应用最广泛的公钥密码是 RSA (Rivest-Shamir-Adleman) 算法^[1], 其主体操作为大数模幂, 即对位宽为 n (目前 n 一般大于 1024) 的操作数 D 、 e 、 M 计算 $D^e \bmod M$. 大数模幂运算一般被分解为很多模乘运算来实现. 针对大数模乘操作中模运算使

用的试除法需要大量计算量这个问题, 能避免使用试除法的 Montgomery 模乘算法被提出^[2], 它能大大减少模乘运算所需要的运算量, 因此已被广泛应用于 RSA 算法, 在模幂计算中替代原模乘算法.

文中针对 Montgomery 模乘算法, 提出了一种使用脉动阵列实现 Montgomery 模乘的硬件结构.

2 Montgomery 模乘算法

给定 n 位操作数 X 、 Y 、 M , Montgomery 模乘的计算结果为 $MM(X, Y, M) = XY2^{-n} \bmod M$, 所需计算量约为 2 个 n 位数相乘的两倍. 由于操作数的位宽 n 很大, 在硬件上直接实现 n 位数相乘非常困难, 因此需要将操作数划分为位数较少的字或者直接按比特来计算, 这样存在很多种选择, 所以 Montgomery 模乘算法的计算也相应的有很多不同的方法.

文中使用二进制多字 Montgomery 模乘算法, 考虑将 n 位操作数分为 e 个长为 w 的字, 在计算的过程中, 对 X 按比特位扫描, 对 Y 按字扫描, 最终将 n 位数的乘法分解为 n 次 e 个 w 位字的加法. 算法具体流程如表 1 所示.

表 1 二进制多字 Montgomery 模乘算法

输入: 操作数 X, Y, M , 其中 $0 \leq X, Y < M, M$ 为奇数;

位宽 $n = \lfloor \log_2 M \rfloor + 1$, 字长 $w, e = \lceil \frac{n+1}{w} \rceil$; X, Y, M 可

表示为: $X = \sum_{i=0}^{n-1} x_i \cdot 2^i, Y = \sum_{j=0}^{e-1} Y^{(j)} \cdot 2^{wj}, M = \sum_{j=0}^{e-1} M^{(j)} \cdot 2^{wj}$

输出: $Z = \sum_{j=0}^{e-1} S^{(j)} \cdot 2^{wj} = MM(X, Y, M) \equiv X \cdot Y \cdot 2^{-n} \pmod{M}, 0 \leq Z < 2M$

```

1: S = 0
2: for i = 0 to n - 1 step 1 do
3:  $q_i = (x_i \cdot Y^{(0)}) \oplus S_0^{(0)}$ 
4:  $(C^{(0)}, S^{(0)}) = x_i \cdot Y^{(0)} + q_i \cdot M^{(0)} + S^{(0)}$ 
5: for j = 1 to e - 1 step 1 do
6:  $(C^{(j)}, S^{(j)}) = C^{(j-1)} + x_i \cdot Y^{(j)} + q_i \cdot M^{(j)} + S^{(j)}$ 
7:  $S^{(j-1)} = (S_0^{(j)}, S_{w-1}^{(j-1)})$ 
8: end for
9:  $S^{(e-1)} = (C_0^{(e-1)}, S_{w-1}^{(e-1)})$ 
10: end for
11: return Z = S

```

下标 i 表示数的第 i 位, 上标 (j) 表示数的第 j 个字, 标号数值从数的最低位往最高位依次增大. 加法进位 C 的位宽为 2.

此算法的参数(位宽 n 和字长 w) 可根据实际需要选择. 在算法流程中, 3~9 步操作都循环 n 次, $XY2^{-n} \pmod{M}$ 中 XY 的计算在第 4 步、第 6 步的加法中实现, 乘 2^{-n} 的计算在第 7 步、第 9 步的移位中实现. 容易发现, 算法的运算量集中在第 4、第 6 步的加法, 而且此操作重复运行多次, 利用此特点, 算法可以由规则的硬件结构实现.

3 Montgomery 模乘硬件结构设计

3.1 脉动阵列

由于上述二进制多字 Montgomery 模乘算法具有规律性, 在硬件电路中, 使用脉动阵列电路, 将算法分割为一系列非常规则的运算分别执行, 进而实现算法. 同时, 仍旧将位宽 n 和字长 w 当成可选参数, 参数选择的唯一条件为 " $w > 1$ ". 对算法的分割以及用硬件实现算法的流程用规则的数据依赖图表示如图 1 所示.

图 1 中 A、B、G 三种运算的具体操作内容分别如图 2 所示.

图 1 中的 i, j 坐标和表 1 算法中的 i, j 标号保持一致. 由于 i 的范围为 $0 \sim n-1$, j 的范围为 $0 \sim e-1$, 因此整个数据依赖图共有 e 列, 每列有 n 个运

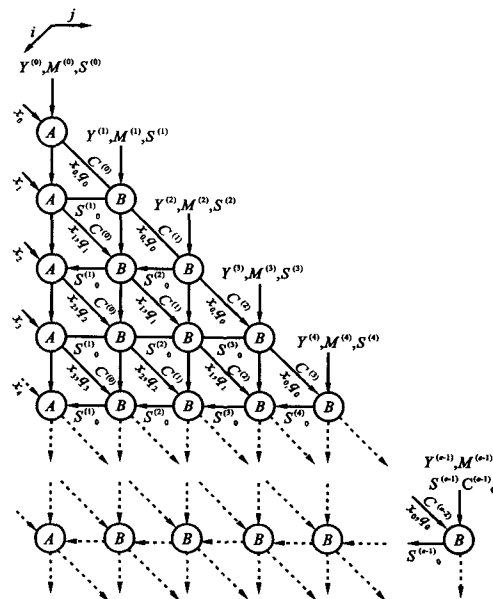


图 1 数据依赖图

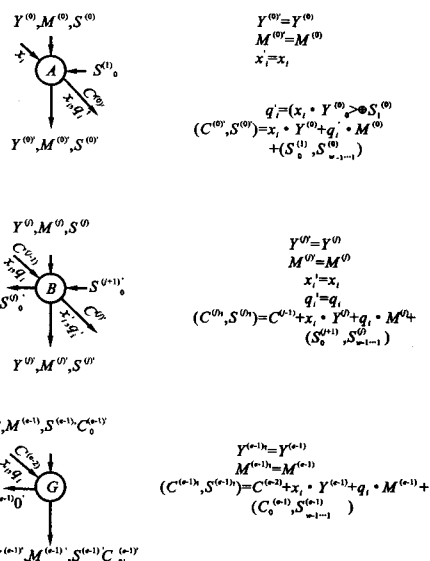


图 2 不同运算的具体操作

算. 表 1 中第 i 次循环第 j 个字的加法包含在图 1 中 (i, j) 坐标处的运算中, 与表 1 中流程不同的是, 移位被放到了每个运算中的加法操作之前实现.

硬件电路实现算法的流程与此数据依赖图相对应. 脉动阵列共有 e 个运算单元, 第 j 个运算单元在 j 个时钟周期之后开始执行图 1 第 j 列的所有运算, 运算执行的时间轴方向垂直于 j 轴向下, 每个运算单元在执行了 n 次运算后结束, 整个算法共需要 $n + e - 1$ 个时钟周期.

脉动阵列结构如图 3 所示. 脉动阵列每个运算单元内部为组合逻辑, 实现的具体功能如图 2 所示; 每个运算单元带 3 个字寄存器(最后一个运算单元多带 1 比特寄存器); 运算单元之间传输 4 个信号共

5 比特,其中 4 比特需要寄存.脉动阵列输入数据 X 从第 0 个运算单元按比特输入, Y 和 M 分为 e 个字

分别输入相应运算单元;输出数据 S 同样分为 e 个字由相应运算单元输出.

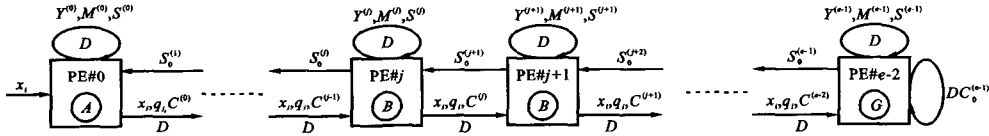


图3 脉动阵列

3.2 脉动阵列的关键路径

只从图3分析,运算单元内部为组合逻辑,而运算单元之间存在一条不带延迟单元的连接路径 ($S_{0}^{(j)}$ 信号传输线),因此可能存在一条无延迟路径将所有的运算单元连接起来,使得关键路径与所有运算单元有关.文中电路结构避免了这种情况发生,具体电路如图4所示.

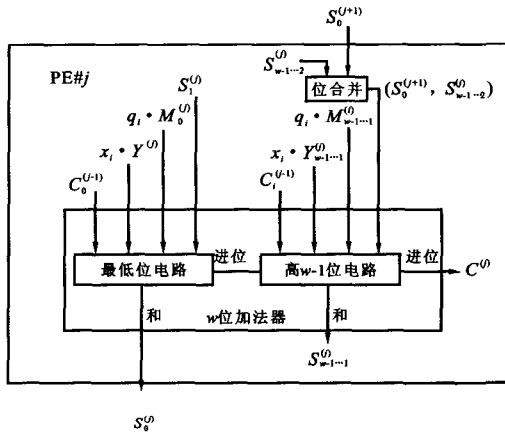


图4 运算单元内部的 $S_{0}^{(j)}$ 信号

加法运算的最低位的和只与加数的最低位有关,因此如图4所示,在 $w > 1$ 的情况下,第 j 个运算单元内部 $S_{0}^{(j)}$ 与 $S_{0}^{(j+1)}$ 之间并不存在通路.结合图3,可以得到包含了 $S_{0}^{(j)}$ 信号传输线的无延迟路径只可能到达第 j 和第 $j-1$ 个运算单元.

同时,两个相邻运算单元之间的组合逻辑连通并不增大运算单元的关键路径.如图4所示,假设加法器的输入除 $S_{0}^{(j+1)}$ 以外所有信号都在寄存器输出时刻同时到达,因此加法器的 $S_{0}^{(j)}$ 输出信号的延迟约为 1 位加法的延迟,同理, $S_{0}^{(j+1)}$ 输入信号的延迟也约为 1 位加法的延迟,而 $S_{0}^{(j+1)}$ 为加法器最高位的输入,由于加法器字长 $w > 1$,因此 $S_{0}^{(j+1)}$ 信号能先于加法器进行最高位计算的时刻到达,其延迟不会对加法器的总延迟造成影响.

所以,脉动阵列的关键路径约为运算单元内部加法器(带进位的 3 输入 w 位加法器)的关键路径.

文献[3-5]中同样提出了基于二进制多字 Montgomery 模乘算法的脉动阵列硬件结构.为了避

免脉动阵列的组合逻辑运算单元之间的信号传输线含有无延迟路径,文献[4-5]中都增加了一级延迟,使得整个算法需要 $2n + e - 1$ 个时钟周期才能够完成;而文献[3]中利用了进位选择加法器的思想,为每个运算单元增加了额外的组合逻辑电路和寄存器,但同时关键路径的延迟也被增大了.

3.3 整个 Montgomery 模乘模块

由于脉动阵列在不同时钟周期执行不同操作,而电路状态的转换并不能完全由脉动阵列电路自身来完成,因此要完整地实现模乘运算需要额外的流程控制电路.文中实现的 Montgomery 模乘电路如图5所示.

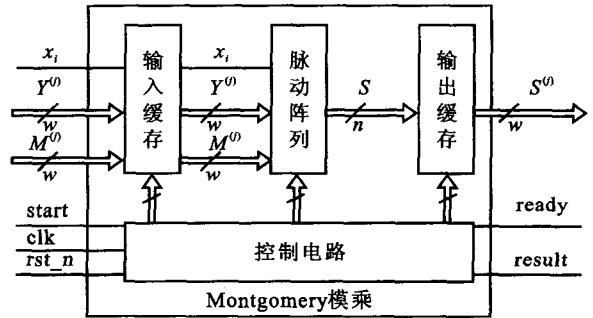


图5 整个模块框图

整个模块与外界接口包括系统时钟信号 clk 及复位信号 rst_n ;数据接口 $x_i, Y^{(j)}, M^{(j)}$ 和 $S_{0}^{(j)}$ 传输数据;控制信号 $start, ready$ 和 $result$ 控制通信过程, $start$ 信号表示外界开始调用此模块, $ready$ 信号表示此模块可接受外界调用, $result$ 信号表示此模块开始输出数据.模块内部的输入缓存电路由寄存器组成,用于输入数据寄存;输出缓存电路由多路选择器和寄存器组成,用来从脉动阵列的运算单元中选定需要的输出数据,由于运算单元数目有可能很大,因此使用多级多路选择器并插入寄存器以避免连线延迟带来关键路径;控制电路处理输入输出的控制信号,并控制整个电路的工作流程,对其他部分电路发出相应的内部控制信号.

外界调用模乘模块的过程如下:在 $ready$ 信号有效的情况下,外界需要计算模乘时,给出 $start$ 信号,并同时分别从最低位开始按比特输入 X 、按字输入

Y 和 M;之后外界等待模乘模块运行,当 result 信号有效时,外界从最低位开始按字读出 S.

模块内部,在 ready 信号有效的情况下,接到 start 信号时,开始依次读入数据,脉动阵列开始计算模乘,ready 信号被置为无效;脉动阵列第一个运算单元执行结束时,开始依次输出数据,result 信号有效;整个脉动阵列执行结束后,数据输出结束,result 信号无效,ready 信号有效.整个模乘流程除了脉动阵列计算需要的 $n + e - 1$ 个时钟以外,多使用了 5 个时钟周期用来输入和输出缓存,因此共需要 $n + e + 4$ 个时钟周期.

4 硬件实现结果比较及分析

用 Verilog HDL 语言描述整个模块的硬件结构,并用 Modelsim SE 6.2b 仿真及验证通过.为了使得最终硬件实现结果具有比较意义,文中选择与文献[3]中相同的后端设计工具,并在相同的 FPGA 上实现我们的电路,具体的硬件实现环境为:FPGA 型号 Xilinx Virtex-II 6000 FF1517-4、综合工具 Synplify Pro 8.1、布局布线工具 Xilinx ISE 8.1,均与文献[3]保持一致.对于参数 n 和 w ,选取了与文献[3]相同的字长 $w = 16$,并分别比较了在位宽 $n = 1024, 2048, 3072$ 和 4096 的情况下的硬件实现结果.

表 2 显示了本文硬件实现结果与文献[3]及文献[4](此数据来自于文献[3])之间的比较结果.

表 2 硬件实现 Montgomery 模乘的结果比较

	位宽	最大 频率/ MHz	模乘 总时 钟数	模乘 所需 时间/ μs	面积 Slices	面积 - 延时积 $\mu s * Slices$	面积 - 延迟积 与文献 [4]比值
文 献 [4]	1024	110.1	2113	19.186	3 937	75 535	1
	2048		4225	38.363	7 756	297 543	1
	3072		6337	57.540	11 576	666 083	1
	4096		8449	76.717	15 393	1 180 905	1
文 献 [3]	1024	100.0	1088	10.880	4 178	45 457	60.18%
	2048		2176	21.760	8 337	181 413	60.97%
	3072		3264	32.640	12 495	407 837	61.23%
	4096		4352	43.520	16 648	724 521	61.35%
本 文	1024	117.9	1092	9.260	4 184	38 744	51.29%
	2048	114.4	2180	19.051	8 185	155 932	52.61%
	3072	111.1	3268	29.412	12 238	359 944	54.04%
	4096	110.1	4356	39.552	16 280	643 907	54.53%

文中的电路面积略大于文献[4]中的原因是文献[4]中运算单元个数只为文中的 $1/2$.但由于其脉动阵列执行运算的过程不如文中规整,每个运算单元中都需要额外的电路控制自身的运算,所以其面积远大于文中的 $1/2$,同时,它实现模乘所需时间约为本文的 2 倍,因此文中的硬件效率仍然较优.

由表 2 容易看出,文中的电路实现 Montgomery 模乘所需时间最少,电路可达到的频率最高,面积-延迟积最小.

5 结束语

文中提出了一种基于二进制多字 Montgomery 模乘算法的脉动阵列硬件结构,在不增加额外的电路逻辑的基础上,使脉动阵列的关键路径不受 Montgomery 算法中移位操作的影响,从而在不增加面积消耗的情况下提高电路的工作频率,进而提高算法运算速度.

参考文献:

- [1] Rivest R L, Shamir A, Adleman L. Method for obtaining digital signatures and public key cryptosystems[J]. Communications of the ACM, 1978, 21(2):120-126.
- [2] Peter L Montgomery. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170):19-521.
- [3] Miaoling Huang, Kris Gaj, Soonhak Kwon, et al. An optimized hardware architecture for the montgomery multiplication algorithm[J]. Public Key Cryptography - PKC, LNCS, 2008(4939):214-228.
- [4] Alexandre F Tenca, øetin K. A scalable architecture for montgomery multiplication[J]. CHES 1999, LNCS, 1999(1717):94-108.
- [5] Alexandre F, Tenca, øetin K, et al. A scalable architecture for modular multiplication based on montgomery's algorithm[J]. IEEE Transactions on Computers, 2003, 52(9):1215-1220.

作者简介:

王缔娜 女,(1986-),硕士研究生.研究方向为 RSA 密码芯片设计.

白国强 男,副教授.研究方向为信息安全芯片.

陈弘毅 男,教授.研究方向为集成电路(IC)设计.