

TTA-EC: 一种基于传输触发体系结构的 ECC 整体算法处理器

赵学秘 王志英 岳虹 陆洪毅 戴葵

(国防科学技术大学计算机学院 长沙 410073)

摘 要 以传输触发体系结构(TTA)为基础,为支持大数运算扩展寄存器堆,增加模乘单元以加速模乘操作,提出一种 ECC 整体算法处理器 TTA-EC. 该处理器具有如下特点:(1)利用 TTA 工具链,可快速开发出基于 TTA-EC 的完整 ECC 公钥系统;(2)模乘单元将以基数为处理字长的高基数 Montgomery 算法与行共享流水结构相结合,具有良好的可扩展性;(3)流水单元实现矢量乘操作,并同时支持 $GF(p)$ 和 $GF(2^n)$ 双有限域;(4)通过调整总线宽度和流水单元个数,可满足不同性能/面积约束. 在 $0.18\mu\text{m}$ 1P6M CMOS 工艺下,其高性能和紧缩面积版本的规模分别为 117.4K 和 40.6K,可分别在 0.87ms 和 7.83ms 内完成一次 $GF(p)$ 或 $GF(2^n)$ 上的 192 位 EC 标量乘运算,峰值功耗分别为 242.1mW 和 28.5mW.

关键词 椭圆曲线公钥系统;大数运算;模乘;有限域;传输触发体系结构;可扩展乘法器
中图法分类号 TP332

TTA-EC: A Whole Algorithm Processor for ECC Based on Transport Triggered Architecture

ZHAO Xue-Mi WANG Zhi-Ying YUE Hong LU Hong-Yi DAI Kui

(School of Computer Science, National University of Defense Technology, Changsha 410073)

Abstract Implementing ECC whole algorithms in hardware has such advantages as more security, less communication bandwidth and more convenient in hardware/software co-design etc. A whole algorithm processor TTA-EC is presented in this paper, which is extended from transport triggered architecture (TTA) by coupling a modular multiplier and long integer registers. TTA-EC has the following characters: (I) ECC whole algorithms can be developed conveniently through the TTA tool chain; (II) the modular multiplier combines a radix-length based version of high radix Montgomery algorithm with a row sharing pipeline design to get high performance and scalability; (III) pipeline elements perform vector production and support Galois field $GF(p)$ and $GF(2^n)$; (IV) different performance/area constraint can be achieved by adjusting the bus width and the number of modular multiplier's pipeline elements. In a $0.18\mu\text{m}$ 1P6M CMOS technology, the high-speed design using 117.4K gates achieves operation time of 0.87ms for a 192-bit elliptic curve scalar multiplication on $GF(p)/GF(2^n)$ field. A compact version requires 40.6K gates and executes the operation in 7.83ms. And their peak powers are 242.1mW and 28.5mW separately.

Keywords elliptic curve cryptosystem; long integer arithmetic; modular multiplication; Galois field; transport triggered architecture; scalable multiplier

收稿日期:2005-11-12;修改稿收到日期:2006-08-28. 本课题得到国家自然科学基金(60173040)资助. 赵学秘,男,1979年生,博士研究生,研究方向为可编程密码处理器设计. E-mail: zhaoxuemi@263.net. 王志英,男,1956年生,教授,博士生导师,研究领域为信息系统安全、异步微处理器设计等. 岳虹,女,1980年生,博士研究生,研究方向为高性能微处理器设计. 陆洪毅,男,1974年生,副教授,研究方向为 VLSI 设计. 戴葵,男,1968年生,副教授,研究方向为安全芯片设计、芯片攻击防御技术等.

1 引言

公钥密码体制思想的提出是密码学史上的一个重要里程碑,它解决了对称密码系统所暴露出的密钥管理困难等缺陷.椭圆曲线密码系统(Elliptic Curve Cryptosystem, ECC)是1985年提出的一种公钥系统^[1],与RSA、DSA等被广泛应用的公钥密码系统相比,ECC具有每一位密钥能够提供更强的安全度、计算量小、密钥存储占用内存少等优点.然而,虽然ECC的数学理论已经基本成熟,但其算法比RSA更难理解和实现,这在一定程度上限制了ECC的应用.

硬件实现ECC算法有更快的速度和更好的安全性.由于ECC算法的复杂性,目前的硬件实现多采用增加协处理器的方法,以加速有限域中的大整数操作,而ECC的整体算法则由主控处理器完成^[2-6].此方法存在如下3点不足:(1)安全性较差,主控处理器仍然可以直接操控密钥,留下安全隐患;(2)与主控处理器之间的通信开销大,限制了协处理器加速性能的发挥;(3)软硬件协同设计困难,主处理器和协处理器所见的主存储器视图不一致,需较多的人工干预.

另外一种方法是定制数据通路和控制通路,并利用微码编程的方法实现ECC整体算法^[7-8].但微码编程方法设计工作量大,正确性验证困难,而且处理器内部设计参数改变后工程变更周期长,难以快速评估设计参数改变所带来的影响.在微处理器上耦合椭圆曲线运算所需的计算资源,通过编译器支持实现ECC整体算法硬件处理器能够克服该问题.

分析表明,ECC实现困难之处除了因为包括复杂的模乘运算外,还在于其所有运算均为大数操作.在传统微处理器架构上进行椭圆曲线(Elliptic Curve, EC)扩展将存在两点困难:寄存器文件难以增加以支持大数运算;当为加速模乘而需要耦合复杂单元时,指令集扩展困难.传输触发体系结构(Transport Triggered Architecture, TTA)是一种新型体系结构^[9],它利用软件直接指定功能单元之间的数据传输,只有一条指令即传输指令,所以在扩展寄存器堆和特殊功能单元时,无需改变指令集,比较适合进行EC扩展.

本文以基本的TTA结构为基础,增加寄存器堆和耦合模乘单元以有效加速包括模乘在内的有限域大数操作,提出一种ECC整体算法处理器

TTA-EC.本文第2节对椭圆曲线密码算法复杂性进行分析;第3节在介绍基于基数长度的双域统一Montgomery算法基础上,讨论了可扩展模乘加速单元的设计;第4节介绍了TTA-EC的整体硬件结构以及基于该处理器的ECC整体算法开发方法;第5节给出不同配置参数下的面积、性能、功耗等实验分析结果;第6节对整篇文章进行了总结.

2 ECC算法复杂性分析

ECC算法的运算量主要取决于标量乘,该运算可进一步拆分为点加和倍加.点加和倍加所需的基本运算是有限域上的乘、加、减、求逆等操作.求逆是最为费时且难以实现硬件加速的运算,所以人们提出了多种通过坐标系变换方法以消除点计算中求逆操作,有射影坐标系和雅克比坐标系等,其中雅克比坐标系计算量小且易于理解^[10].在雅克比坐标系下,点加需要12次乘法和4次减法,倍加需要4次乘法和6次减法,没有逆操作.求逆运算仅需在标量乘开始和结束时,进行坐标系转换过程中发生,所占运算比例很小.经过坐标系变换后,ECC算法实现的困难主要在于:(1)有限域上的乘法、加法、减法均为大数运算;(2)有限域乘法耗费巨大运算量.本文通过在TTA结构基础上耦合大数寄存器和模乘加速单元,以高效实现ECC整体算法.

3 可扩展双域统一模乘单元

模乘加速单元的设计与实现模乘所采用的算法紧密相关,本节首先给出一种适合于硬件实现的高效算法,然后依据此算法设计行共享流水结构的可扩展模乘单元.

3.1 基于基数长度的双域统一高基数 Montgomery 算法

模乘算法有很多种.Montgomery算法^[11]因避免了除法操作得到广泛应用.文献[12]证明了Montgomery算法同样适合于 $GF(2^n)$ 域中的乘法,只是运算规则由整数运算规则变为多项式运算规则.文献[13]提出一种基于基数长度的高基数Montgomery模乘(Radix-length Based High Radix Montgomery Modular Multiplication, RBHRMMM)算法.该算法以基数长度为处理字长,适合于硬件实现.将其中运算规则扩展,则可以得到适合 $GF(p)$ 和 $GF(2^n)$ 双域的基于基数字长的高基数Montgomery

算法.

算法 1. 以基数长度为处理宽度的双域统一高基数 Montgomery 模乘算法——UniRBHRMMM 算法.

参数: 基数长度 k ; 模数长度 L ; $n = \lfloor L/k \rfloor$

输入: $\tilde{M} = \{\tilde{m}_n, \tilde{m}_{n-1}, \dots, \tilde{m}_0\}$, $A = \{a_{n+2}, a_{n+1}, \dots, a_0\}$,

$B = \{b_{n+1}, b_n, \dots, b_0\}$, $a_i, b_i, \tilde{m}_i \in [0..2^k - 1]$

输出: $R = A \otimes B \otimes 2^{-k(n+2)} \bmod \tilde{M}$

按如下步骤计算输出值 R :

1. $R_{-1} = 0$; $B' = B \gg k$;
2. for ($i=0$; $i \leq n+2$; $i=i+1$) // i 循环
3. $q_i = r_{(i-1)0}$; $q\tilde{M}C_{i(-1)} = 0$; $aB'S_{i(-1)} = 0$; $C_{i(-1)} = 0$;
4. for ($j=0$; $j \leq n+2$; $j=j+1$) // j 循环
5. $q\tilde{M}C_{ij} = (q_i \otimes \tilde{m}_j \oplus q\tilde{M}C_{i(j-1)}) \text{div } 2^k$;
6. $q\tilde{M}S_{ij} = (q_i \otimes m_j \oplus q\tilde{M}C_{i(j-1)}) \bmod 2^k$;
7. $aB'C_{ij} = (a_i \otimes b'_j \oplus aB'C_{i(j-1)}) \text{div } 2^k$;
8. $aB'S_{ij} = (a_i \otimes b'_j \oplus aB'S_{i(j-1)}) \bmod 2^k$;
9. $C_{ij} = (r_{(i-1)j} \oplus q\tilde{M}S_{ij} \oplus aB'S_{ij} \oplus C_{i(j-1)}) \text{div } 2^k$;
10. $r_{ij} = (r_{(i-1)j} \oplus q\tilde{M}S_{ij} \oplus aB'S_{ij} \oplus C_{i(j-1)}) \bmod 2^k$;
11. $r_{i(j-1)} = r_{ij}$;
12. 输出 $R = R_{n+2}$.

在算法 1 中, 对于 $GF(p)$ 域, 其中的加法 \oplus 和乘法 \otimes 均满足整数运算规则; 对于 $GF(2^n)$ 域, 其中的加法 \oplus 和乘法 \otimes 则均满足多项式运算规则.

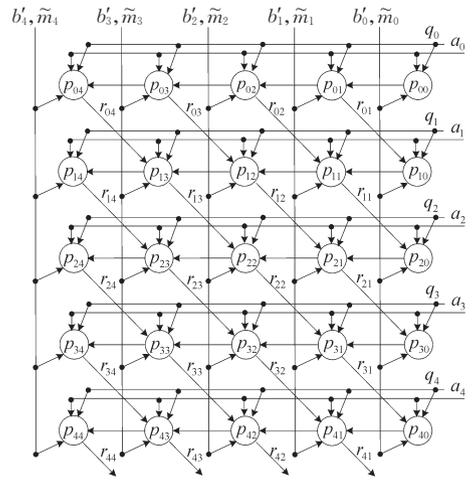
3.2 UniRBHRMMM 算法并行性分析

算法 1 的主体包括两个循环, 内循环称之为 j 循环, 外循环称之为 i 循环. 为便于描述, 定义算法 1 中 j 循环中的一个循环片为一个原子操作, 用 p_{ij} 表示. 图 1(a) 显示了 $n=2$ 时, 算法 1 中原子操作之间的数据共享与依赖关系.

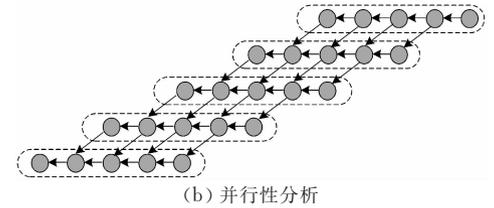
数据共享关系: ①第 i 行原子操作共享 a_i, q_i ; ②第 j 列原子操作共享 b'_j, \tilde{m}_j .

数据依赖关系: ①原子操作 p_{ij} 依赖于 $p_{(i-1)(j+1)}$ 产生的结果 $r_{(i-1)(j+1)}$ 和 $p_{i(j-1)}$ 产生的进位组; ②第 i 行的共享数据 q_i 依赖于 $p_{(i-1)1}$ 的结果 $r_{(i-1)1}$.

图 1(b) 按照同一列的原子操作之间没有数据相关性的原则对图 1(a) 进行并行性变换, 可以看出最



(a) $n=2$ 时数据相关图



(b) 并行性分析

图 1 原子操作间相关性与并行性分析

大并行性为 3, 对于一般情况为 $\lfloor (n+3)/2 \rfloor$. 所以在设计功能单元时, 流水单元个数最大为 $\lfloor (n+3)/2 \rfloor$.

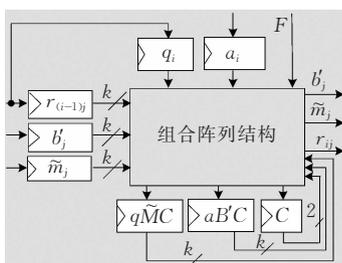
3.3 可扩展双域模乘加速单元结构

采用行共享流水处理方法实现可扩展双域统一结构模乘功能单元 (Unified Modular Multiplier, UMM). 行共享的概念是同一行的原子操作分时共享一个流水单元 (Pipeline Element, PE), 也就是说每个流水单元可完成算法 1 中的一次 j 循环, 具有良好的扩展性. 假设流水单元的个数为 ρ ($1 \leq \rho \leq \lfloor (n+3)/2 \rfloor$), 按照如下步骤生成行共享流水处理结构.

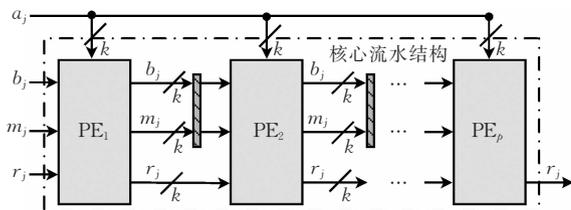
(1) 将 ρ 个处理单元按照递增顺序自左至右排列.

(2) 每个处理单元组成一个流水站, 其结构如下: 根据原子操作所需的源操作数生成寄存器, 将计算映射成计算逻辑阵列; 由于同一行相邻两次操作之间有进位组的依赖, 所以进位组的输出反馈到计算逻辑阵列如图 2(a) 所示.

(3) 生成流水阵列结构. 阵列只有自右向左流



(a) 流水单元结构



(b) 流水结构

图 2 行共享可扩展双域模乘单元

水传递的信号,包括列共享数据 b'_j, \bar{m}_j , 行之间的依赖数据 r 和控制信号. 由于首个 r 数据的产生需要 2 拍, 所以需要一组寄存器同步其它数据. 行共享可扩展模乘单元结构如图 2(b) 所示.

为了实现在 $GF(p)$ 和 $GF(2^n)$ 两个有限域硬件架构的统一, 在两种域中采用的算法形式要尽量相似, 这样可以使额外的硬件消耗减少至最少, 以往的实践证明将整数运算规则的乘法和加法扩展到多项式运算是可行的, 所需的主要是消除加法运算的进位传播并修改一些分支判断的条件. 流水单元中的逻辑运算结构依然采用文献[13]中的处理逻辑结构, 但需要将其中的全加器多加一个域控制位 F , 并将运算规则修改为

$$S_i = A_i \wedge B_i \wedge C_i$$

$$C_{i+1} = (F \wedge A_i \wedge B_i) \vee (F \wedge B_i \wedge C_i) \vee (F \wedge A_i \wedge C_i) \quad (1)$$

4 TTA-EC 整体结构及其软硬件协同设计

本节在对 TTA 概念进行介绍后, 给出了对基本 TTA 进行 EC 扩展后得到的 TTA-EC 处理器, 并讨论了基于 TTA-EC 处理器的 ECC 整体算法软硬件协同设计.

4.1 传输触发体系结构

由 Corporaal 等人提出 TTA, 可以看成传统超长指令字体系结构 (Very Long Instruction Word, VLIW) 的一个超集. 如果 VLIW 被看成单指令多操作类型的体系结构, 那么 TTA 则是单指令多传输类型的体系结构. 所有的操作, 包括 load/store、分

支、跳转等, 都通过传输指令即 MOVE 来完成. 例如, 将一个加法操作变换成 3 个 MOVE 操作:

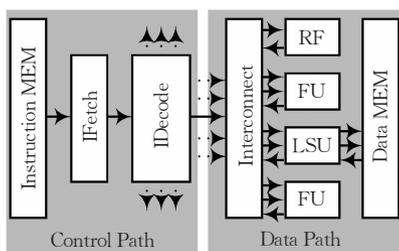
$$\text{ADD } R_3, R_2, R_1 \Rightarrow$$

$$R_1 \rightarrow O_{\text{ADD}}; R_2 \rightarrow T_{\text{ADD}}; R_{\text{ADD}} \rightarrow R_3.$$

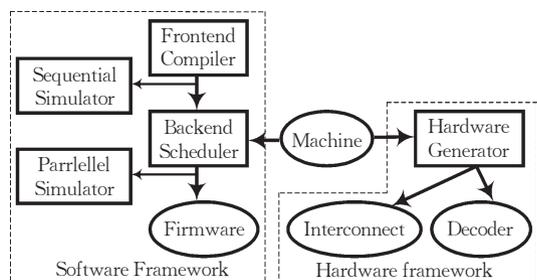
首先, R_1 和 R_2 的值分别被传输到加法器功能单元的操作数寄存器和触发寄存器, 过一段时间 (根据加法器本身的延迟来决定) 之后, 结果就被送入 R_3 . 通过编译调度, 一条 TTA 指令通常包含多个可并行的 MOVE 操作.

图 3 显示了 TTA 计算结构及其软硬件协同设计框架, 图 3(a) 显示了 TTA 处理器的概念结构示意图, 数据通路部分由寄存器文件 (Register File, RF)、访存单元 (Load Store Unit, LSU)、功能单元 (Functional Unit, FU) 通过互连总线连接组成. 控制通路由取指单元、译码单元组成, 译码单元生成对互连总线的控制信号和立即数. TTA 计算架构下不同面向应用定制处理器 (Application Specific Processor, ASP) 之间的差别在于处理器类型和数量以及互连网络的不同.

如图 3(b) 所示, TTA 工具链框架包括软件框架和硬件框架, 同时定义了一种机器描述语言, ASP 可通过这种语言进行描述, 生成 Machine 文件. TTA 软件框架中的前端编译器根据基本的 TTA 处理器生成串行代码, 后端调度器则根据 Machine 文件来调度串行代码, 生成高效的并行代码. 串行模拟器和并行模拟器则分别用来模拟串行代码和并行代码, 验证其正确性和评估性能. 硬件框架根据 Machine 文件生成译码器和互连网络的 HDL 描述, 给后续 EDA 工具使用.



(a) TTA 处理器结构示意图



(b) TTA 软硬件工具链框架示意

图 3 TTA 计算结构及其软硬件协同设计框架

TTA 结构适合于椭圆曲线扩展源于如下 3 个方面的原因:

(1) TTA 软件工具链分为前端编译和后端调度, 由后端调度完成寄存器分配和指令调度. 后端调度器根据功能单元延迟和数据/控制依赖关系只调度

一种指令即 MOVE, 而无需关心功能单元的语义;

(2) TTA 支持寄存器文件定制, 可以划分为多个寄存器堆, 每个堆均可作为大数寄存器使用;

(3) TTA 硬件工具链可以根据 Machine 文件生成译码器和互连网络的高级语言描述. 设计者可

以将精力集中于特殊功能单元的设计中.

4.2 TTA-EC 硬件整体结构

在基本传输触发体系结构的基础上进行椭圆曲线扩展得到椭圆曲线处理器 TTA-EC. 图 4 显示了 TTA-EC 的数据通路,其总线宽度与模乘算法采用的基数长度 k 一致. 功能单元端口与总线之间的连接称之为 Socket. TTA-EC 处理器包括两个部分:基本 TTA 和 ECC 扩展. 基本的 TTA 包括 RF、LSU、加/减法单元(Add/Subtract Unit, ASU)、移位单元(Shift Unit, SU)和逻辑计算单元(Logic Unit, LU),这些单元的 Socket 均为全相连,所以基本的 TTA 可以实现任意数据处理功能. ECC 扩展部分除了耦合 UMM 外,还增加了 ω 个寄存器堆(Register Unit, RU). UMM 单元用来加速模乘运算,寄存器堆则用以实现大数寄存器.

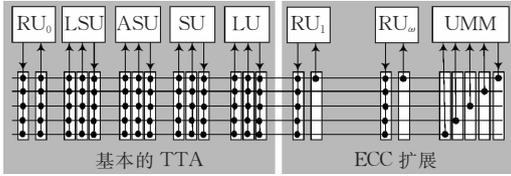


图 4 TTA-EC 整体结构

图 4 中 $RU_i (1 \leq i \leq \omega)$ 均为单读单写寄存器堆,其读 Socket 为全相连,写 Socket 只与一组总线连接. UMM 有 4 个输入端口和 1 个输出端口,这 5 个 Socket 分别与一组总线连接. 这种有选择的与总线连接的方式可在满足算法 1 中数据依赖关系的基础上,减少总线负载,缩短总线传输时间.

UMM 支持 3 种操作:第 1 种是载入 $a_i (1 \leq i \leq \rho) LA[i]$;第 2 种是载入域参数 LF ;第 3 种是流水运算 PPC. 在互连总线资源允许的前提下,这 3 种操作均可并行.

根据图 4 所示的结构,参照 Machine 描述语法生成 TTA-EC 机器描述文件. TTA 硬件框架根据此文件可以生成互连结构和译码单元的高级语言描述,选择多路选择器的方法实现总线控制.

4.3 软硬件协同设计

椭圆曲线密码系统在 TTA-EC 上的实现分为 4 个层次,如表 1 所示,自底向上分别是硬件层、有限域大数操作层、椭圆曲线点操作层和 ECC 整体算法层. 不同的层次采用不同的设计方法实现. 最低层次是基本的传输操作,由硬件来完成. 第 2 个层次是利用加入的特殊功能单元实现大数的载入/载出以及大数模乘,与修改前端编译器相比,嵌入汇编方法不仅能有效利用特殊功能单元,而且复杂度更低. 椭

圆曲线点操作层和 ECC 整体算法层均独立于硬件结构,采用高级语言方法实现.

表 1 ECC 整体算法实现层次

	内容	协同设计
ECC 整体算法	ECDSA、EC-ElGamal	高级语言
点操作	点加,倍加,标量乘	高级语言
大数操作	模乘/加/减,载入/出大数	高级语言/嵌入汇编
硬件	传输	硬件实现

嵌入汇编的 ECC 整体算法通过 TTA 前端编译器得到串行代码,后端调度器则根据串行代码和 TTA-EC 的机器描述文件生成并行代码. 串行模拟器和并行模拟器分别模拟执行串行代码和并行代码,进行正确性验证和性能评估,并将最终的并行代码作为 ECC 整体算法的固件. 大数模乘是 ECC 的关键运算,下一节将讨论基于 TTA-EC 的大数模乘实现.

4.4 基于 TTA-EC 的大数模乘算法

在实现大数操作时,增加的寄存器堆均作为大数寄存器使用,而且遵循使用约定:每个寄存器堆的 r_0 ,均为大数首地址寄存器; r_1 作为大数寄存器是否被修改的标志. 在将大数从存储器载入到大数寄存器之前,通过首地址先判断大数是否已经在 $RU_i (0 \leq i \leq \omega)$ 中,如果在则无需载入. 在将大数写回存储器之前也要判断大数是否被修改,如果没有被修改则无需写回. 假定大数均已采用上述的策略载入到大数寄存器中,算法 2 给出了基于 TTA-EC 处理器的大数模乘算法.

算法 2. 基于 TTA-EC 处理器的大数模乘算法.

参数:处理器宽度 k ; 大数长度 L ; UMM 中流水单元个数 ρ ; 域 $FIELD$; $n = \lfloor L/k \rfloor$

输入: $A = \{a[n+2], a[n+1], \dots, a[0]\}$,

$B = \{b[n+2], b[n+1], \dots, b[0]\}$,

$M = \{m[n+2], m[n+1], \dots, a[0]\}$

输出: $R = A \otimes B \otimes 2^{k \cdot \lfloor (n+2)/\rho \rfloor \cdot \rho} \bmod M$

过程:

1. $FIELD \rightarrow UMM.LF_t$; /* 载入域参数 */
2. for ($i=0$; $i \leq \lfloor (n+2)/\rho \rfloor$; $i++$)
3. for ($x=0$; $x < \rho$; $x++$) /* 送 r, m, b 到第一个流水单元;读取上一轮的中间结果;送 $a[4i+x]$ 到第 x 个流水单元 */
4. $b[x] \rightarrow UMM.PPC_o1$, $m[x] \rightarrow UMM.PPC_o2$;
 $r[x] \rightarrow UMM.PPC_t$, $UMM.r \rightarrow r[4j-8+x]$;
 $a[4i+x] \rightarrow UMM.LA[i]_t$;
5. for ($x=0$; $x < \rho$; $x++$) /* 送 r, m, b 到第一个流水单元;读取上一轮的中间结果 */
6. $b[4+x] \rightarrow UMM.PPC_o1$,

```

m[4+x]→UMM.PPC_o2,
r[4+x]→UMM.PPC_t,
UMM,r→r[4j-4+x];
7. for (j=2; j<=⌊(n+2)/ρ⌋; j++)
    /* 运算主体 */
8. for (x=0; x<ρ; x++)
9. UMM.PPC_r→r[4j-8+x],
    b[4j+x]→UMM.PPC_o1,
    m[4j+x]→UMM.PPC_o2,
    r[4j+x]→UMM.PPC_t;
10. 读取最后一轮剩余的 2ρ 个结果.

```

算法 2 中 x 索引的循环体内所有 MOVE 操作组成一条指令,可在一个时钟周期内完成,那么基于 TTA-EC 完成一次模乘所需的时钟周期数为

$$N_{MM} = 1 + (\lceil (n+2)/\rho \rceil + 1)^2 \times \rho + 2\rho \quad (2)$$

5 实验结果

5.1 正确性验证

实验过程中,ECDSA、EC-ElGamal 两种椭圆曲线公钥方案整体算法被用于验证 TTA-EC 的有效性,算法验证采用如下流程:模拟器→高级硬件描述模拟验证→FPGA 验证.首先利用 TTA 软件框架提供的模拟器进行验证,模拟器验证速度快、信息统计方便,某些参数在此阶段确定.在利用硬件框架生成译码器的高级语言描述后,使用 Modelsim5.7a 进行模拟验证.最后在 GiDEL 公司的 PROCSUPERSTAR 设计验证板上完成 FPGA 验证.

PROCSUPERSTAR80-3 设计验证板上包括 3 块 Altera 公司的 Stratix80 FPGA 芯片和 2 块 32MB DRAM.采用 Synplifypro 7.2 进行逻辑综合和 QuartusII5.0 完成布局布线.将硬件下载到 FPGA,将固件和测试数据载入 DRAM 中,测试结果正确,顺利通过 FPGA 硬件验证.

5.2 寄存器堆参数 ω

大数寄存器与存储器之间数据的载入/载出均

为耗时的操作,称之为抖动,其中大数载入称为载入抖动,大数载出称为载出抖动.抖动次数与大数寄存器即寄存器堆的个数 ω 紧密相关,而与总线宽度 k 、UMM 流水单元个数 ρ 无关.在模拟器验证阶段对抖动进行统计,对于 192 为 EC 标量乘运算,得到抖动次数与寄存器堆个数 ω 的关系如图 5 所示.从图中可以看出,载入抖动和载出抖动均随着 ω 的增加而下降,在 $\omega=12$ 时,抖动几乎为 0,所以在 TTA-EC 中,确定大数寄存器参数 ω 为 12.另外,每个寄存器堆所能支持的最大大数长度为 256 位.

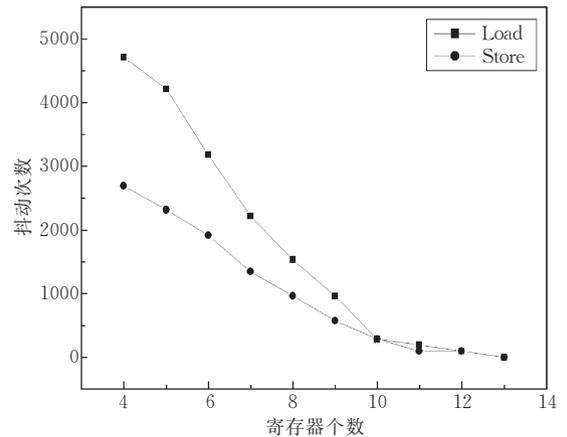


图 5 抖动次数与大数寄存器个数 ω 的关系

5.3 ASIC 设计评估

TTA-EC 的 ASIC 设计采用基于标准单元的设计流程,目标工艺是 $0.18\mu\text{m}$ 1P6M CMOS 工艺,固件存储器由 Memory compiler 得到,总线和功能单元均由标准单元实现.采用 Design Compiler 进行逻辑综合和 SOC Encounter 进行布局布线等物理设计,采用 Formality 进行逻辑等价性检查,PrimeTimeSI 进行静态时序分析.综合后,设计规模随总线宽度 k 的变化如表 2 所示. S_{PE} 代表一个流水单元规模,TTA-EC- ρ 则表示耦合了含有 ρ 个流水单元 UMM 的 TTA-EC 处理器.通过表 2 可以看出, S_{PE} 随着总线宽度 k 的增长是以近似 $O(k^2)$ 速度增长, $S_{TTA-EC-\rho}$ 随着 ρ 的增长是以近似 $O(\rho)$ 速度增长.

表 2 TTA-EC 逻辑综合后的设计规模(单位是 2 输入与非门)

k (基数 $r=2^k$)	S_{PE}	$S_{TTA-EC-1}$	$S_{TTA-EC-2}$	$S_{TTA-EC-3}$	$S_{TTA-EC-4}$	$S_{TTA-EC-5}$	$S_{TTA-EC-6}$	$S_{TTA-EC-7}$	$S_{TTA-EC-8}$
8	1618	22174	23695	25306	26738	28468	30127	31716	33173
16	7569	29827	37148	44592	52483	59489	67591	74715	81705
32	19695	44453	63456	81670	98938	—	—	—	—
64	65446	96046	160328	—	—	—	—	—	—

表 3 列出了完成布局布线等物理设计后的 TTA-EC 的面积和时钟周期.物理布局时,将标准单元逻辑部分的利用率设定为 85%,为布局后优化留下足够的空间.在 $k \leq 16$ 时,关键路径为互连网络中的总线,

在 $k \geq 32$ 时,关键路径在流水单元中.流水单元的延迟理论上应以 $O(\log_2 k)$ 的速度增长,但由于单个流水单元规模剧增导致 EDA 工具的优化能力下降,实际增长速度在 k 由 32 增长到 64 时,远大于理论值.

表 3 TTA-EC 物理设计后的面积与时钟周期

k (基数 $r=2^k$)	$A_{TTA-EC-1}/(\mu\text{m})^2$	$A_{TTA-EC-2}/(\mu\text{m})^2$	$A_{TTA-EC-3}/(\mu\text{m})^2$	$A_{TTA-EC-4}/(\mu\text{m})^2$	$A_{TTA-EC-5}/(\mu\text{m})^2$	$A_{TTA-EC-6}/(\mu\text{m})^2$	$A_{TTA-EC-7}/(\mu\text{m})^2$	$A_{TTA-EC-8}/(\mu\text{m})^2$	T_{clk}/ns
8	540×445	540×475	540×505	540×535	540×575	540×605	540×635	540×675	5.27
16	550×570	550×710	650×710	750×710	850×710	850×800	850×890	850×980	5.27
32	720×620	720×990	920×990	1120×990	—	—	—	—	7.51
64	1160×785	1160×1355	—	—	—	—	—	—	11.62

TTA-EC 的固件大小与参数无关,每个指令字包括 5 条传输操作共计 $14 \times 5 = 70\text{bit}$,实现标量乘的固件大小为 $87 \times 70 = 6090\text{bit}$,相当于 18.5K 门。

5.4 性能与功耗分析

在给定 TTA-EC 的有限域模数长度 L 、总线宽度 k 、UMM 流水单元数 ρ 等参数条件下,分析完成一次 EC 标量乘法所需的时钟周期数:一次标量乘法可以拆分为 L 次倍加操作和平均 $(L/2)$ 次点加操作;在雅克比坐标系下,倍加可以拆分为 4 次模乘和

6 次模减,点加可以拆分为 12 次模乘和 4 次模减;模乘运算所需的周期数可根据式(2)得到,模减所需的周期数为 $2L/k$ 。所以,完成一次点标量乘主体运算所需的总的时钟周期数理论值为

$$N_{\text{PM}} = L \times (4N_{\text{MM}} + 12L/k) + L/2 \times (12N_{\text{MM}} + 8L/k) = 10L \times N_{\text{MM}} + 16L^2/k \quad (3)$$

表 4 显示了 $L=192$ 时,标量乘计算实际运算周期数。

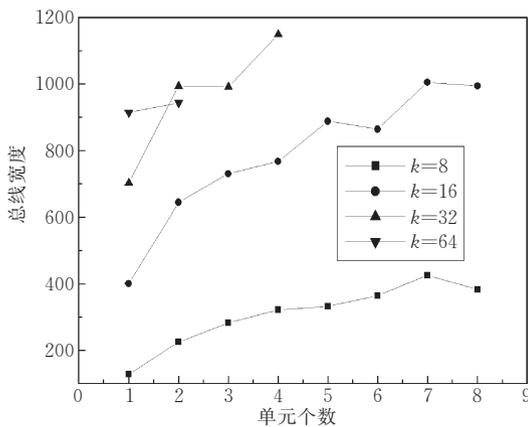
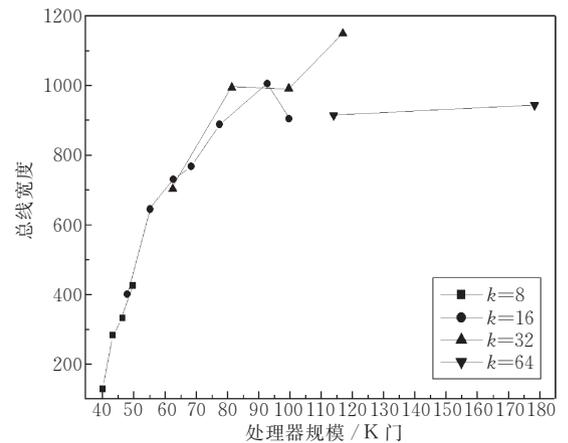
表 4 $L=192$ 时实际运算周期数

k	N							
	$\rho=1$	$\rho=2$	$\rho=3$	$\rho=4$	$\rho=5$	$\rho=6$	$\rho=7$	$\rho=8$
8	1485227	842446	669346	588829	570934	519785	444518	495493
16	473112	294357	259741	247239	213672	248258	188725	209864
32	189781	134127	134326	115822	—	—	—	—
64	94179	91332	—	—	—	—	—	—

上述周期除含运算主体外,还包括使用 UniRB-HRMMM 算法所需的预处理和后处理以及使用雅克比坐标系后所需的坐标系转换计算。表 4 中的实际运算周期基本符合理论值。实际运算性能运算周期数和时钟周期两个因素相关,综合表 2~表 4,得到 $L=192$ 时实际运算性能与处理器参数、电路规模的关系如图 6 所示。

图 6(a)显示了标量乘性能与处理器总线宽度 k

和 UMM 流水单元个数 ρ 的关系,图 6(b)显示了标量乘性能与逻辑硬件规模的关系。从图 6(b)可以看出,对于面积约束较强的应用,可以选择处理器宽度 $k=8$ 或 16,并根据性能方面的约束选择参数 ρ 。对于具有高性能需求的应用,可以选择处理器宽度 $k=16$ 或 32,并根据面积方面的考虑选择参数 ρ 。在 $k=64$ 时,性能规模比差,实际应用价值不高。

(a) 性能与参数 k, ρ 的关系

(b) 性能与处理器规模的关系

图 6 TTA-EC 处理器性能

采用动态功耗估算方法来估计功耗。在仿真工具 Modelsim5.7 上,分别进行 $GF(\rho)$ 和 $GF(2^n)$ 域上各 500 组 192 位标量乘的仿真,获得网表与连线的电平翻转模型,并将这些模型送入动态功耗估算

工具 PowerMill,得到给定参数配置下的针对每一组数据的 TTA-EC 的动态处理功耗,然后取功耗峰值,并计算针对该 1000 组数据的峰值功耗平均值,结果如表 5 所示。

表 5 TTA-EC 的峰值功耗

$k(\text{基数 } r=2^k)$	$P_{\text{TTA-EC-1}}/\text{mW}$	$P_{\text{TTA-EC-2}}/\text{mW}$	$P_{\text{TTA-EC-3}}/\text{mW}$	$P_{\text{TTA-EC-4}}/\text{mW}$	$P_{\text{TTA-EC-5}}/\text{mW}$	$P_{\text{TTA-EC-6}}/\text{mW}$	$P_{\text{TTA-EC-7}}/\text{mW}$	$P_{\text{TTA-EC-8}}/\text{mW}$
8	28.5	33.1	37.8	42.8	47.4	51.2	56.1	61.1
16	58.3	76.7	95.3	113.7	132.5	151.5	169.6	188.7
32	92.6	142.3	192.4	242.1	—	—	—	—
64	180.2	307.7	—	—	—	—	—	—

峰值情况下的 TTA-EC 的功耗主要由四部分组成:寄存器文件读写、UMM 功能单元、总线功耗和控制功耗(取指、译码和控制).寄存器文件的功耗可以在生成时直接得到,UMM 的功耗可以通过差分的方法计算出,这两部分功耗对任何控制结构都是必须的.剩余的功耗是总线功耗和控制功耗.在高性能应用情况下,UMM 的功耗占主要地位,而在小面积应用情况下,总线与控制功耗占了较大的比例.

5.5 与已有工作的比较

表 6 列出了 TTA-EC 与已有工作的比较.文献[3]的数据通路为 256 位,采用进位存储表示形式以提高主频,利用双域全加器支持双域计算,与其相比,TTA-EC 的紧缩面积版本具有面积优势.文献

[5]的数据通路中包括一个由 4 级流水线组成的矢量乘法器,其设计规模没有考虑内部 RAM,在考虑内部 RAM 后,TTA-EC 的面积与文献[5]相似,但具有更高的性能.文献[3,5]均没有考虑功耗问题,文献[6]是一种低功耗 ECC 算法协处理器,依主频不同分为 3 个版本,其功耗与主频成正比,其 100MHz 主频版本与 TTA-EC 的紧缩面积版本性能相似,但功耗更低,其原因在于:首先,文献[6]采用 TSMC 的为功耗优化的 $0.13\mu\text{m}$ 标准单元库;其次,TTA-EC 的总线互连网络耗费了较多的额外功耗,而该设计没有考虑与主控处理器通信引起的功耗.另外,上述设计均为以协处理器方式工作,由于没有实现完整算法,在系统软件级别留下安全隐患.

表 6 与已有工作的比较

	工艺	开发方法	支持域	版本	主频/MHz	规模/K	功耗/mW	标量乘	备注
文献[3]	$0.35\mu\text{m}$	协处理器	双域	—	100	45	—	6.3ms/233b	进位存储表示和双域全加器
文献[5]	$0.35\mu\text{m}$	协处理器	双域	—	25	26	—	60ms/192b	4 级流水线的矢量乘法器
文献[6]	$0.13\mu\text{m}$	协处理器	$GF(p)$	低主频	20	30.3	0.99	31.9ms/168b	可扩展乘法器和可扩展模逆加速单元
				中主频	100	30.4	4.34	6.3ms/168b	
				高主频	200	34.3	9.89	3.1ms/168b	
文献[7]	$0.13\mu\text{m}$	手工编码	双域	高性能	137.7	117.5	—	1.44ms/192b	64 位乘法器
				紧缩面积	363.6	28.3	—	12.4ms/192b	8 位乘法器
本文	$0.18\mu\text{m}$	高级语言 嵌入汇编	双域	高性能	133.1	117.4	242.1	0.87ms/192b	含 4 个 PE 的 32 位模乘单元
				紧缩面积	189.7	40.6	28.5	7.83ms/192b	含 1 个 PE 的 8 位模乘单元

文献[7]是与本文最为接近的研究,与其相比,TTA-EC 除具有更方便的开发方法外,其速度更快,原因在于:TTA-EC 中的流水单元支持矢量乘操作,能够更好地支持 Montgomery 算法;采用雅克比坐标系,避免了费时的模逆运算.本文的紧缩面积版本比文献[7]要稍差,原因在于固件存储器和大数据寄存器的规模与总线宽度无关,而且互连总线占了一定面积,虽然模乘单元规模随着总线宽度减小而减少,但整体规模依然较大.

6 结 论

硬件实现 ECC 整体算法具有安全性好、与主 CPU 间通信开销低和软硬件协同设计方便等优点,但 ECC 算法的复杂性使得硬件实现整体算法存在诸多难点,微码编程方法设计复杂性高,正确性验证困难.本文以传输触发体系结构为基础,进行 EC 扩展,得到一种能有效实现 ECC 整体算法的处理器

TTA-EC. EC 扩展包括两个部分:增加寄存器堆以有效支持大数操作和耦合可扩展双域统一模乘单元以加速模乘运算. TTA-EC 有如下特点:(1)通过 TTA 工具链,可快速开发出 ECC 公钥密码系统;(2)模乘单元采用列共享流水结构,具有良好的可扩展性;(3)流水单元实现矢量乘操作,并支持 $GF(p)$ 和 $GF(2^n)$ 双域;(4)通过调整总线宽度和模乘单元中流水单元个数,可满足不同性能、面积约束要求.

实验过程中对 TTA-EC 涉及的参数进行评估,得到以下结果:大数抖动随寄存器堆个数 ω 的增加而急剧下降,在 $\omega=12$ 时,抖动可基本消除;对于面积敏感的应用,处理器宽度选择 $k=8$ 或 16 较为合适,对于性能要求较高的应用, $k=16$ 或 32 较为合适;在 $0.18\mu\text{m}$ 1P6M CMOS 工艺下,其高性能版本和紧缩面积版本规模分别为 117.4K 和 40.6K,可分别在 0.87ms 和 7.83ms 内完成一次 $GF(p)/GF(2^n)$ 域上 192 位 EC 标量乘运算,其峰值功耗分别为 242.1mW 和 28.5mW.

致 谢 本文作者对芬兰 TUT 大学的 Takala 教授和 Cilio 博士为本文研究工作提供的巨大帮助表示感谢!

参 考 文 献

- [1] Miller V. Use of elliptic curves in cryptography//Proceedings of the Advances in Cryptography (CRYPTO'85). Santa Barbara, California, USA, 1985; 417-426
- [2] Shi Yan, Wu Xing-Jun. Hi-speed dual-field crypto coprocessor design. *Microelectronics & Computer*, 2005, 22(5): 8-12(in Chinese)
(史 焱, 吴行军. 高速双有限域加密协处理器设计. *微电子学与计算机*, 2005, 22(5): 8-12)
- [3] Chen Chao, Zeng Xiao-yang, Zhang Qian-ling. A new hardware-reconfigurable public-key cryptographic coprocessor. *Journal on Communications*, 2005, 26(1): 6-12 (in Chinese)
(陈 超, 曾晓洋, 章倩玲. 一种新型硬件可配置公钥密码协处理器的 VLSI 实现. *通信学报*, 2005, 26(1): 6-12)
- [4] Savas E, Tenca A F, Koc C K. A scalable and unified multiplier Architecture for finite fields $GF(p)$ and $GF(2^n)$ //Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'00). Worcester, USA, 2000; 277-292
- [5] Wu Xing-Jun, Bai Li-Chen, Sun Yi-Le, Chen Hong-Yi. A modular processor for multi public-key cryptography. *Microelectronics*, 2005, 35(5): 549-552(in Chinese)

- (吴行军, 白立晨, 孙怡乐, 陈弘毅. 一种适用于多种公钥密码算法的模运算处理器. *微电子学*, 2005, 35(5): 549-552)
- [6] Ozturk E, Sunar B, Savas E. Low-power elliptic curve cryptography using scaled modular arithmetic//Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'04). Cambridge, MA, USA, 2004; 92-106
- [7] Satoh A, Takano K. A scalable dual-field elliptic curve cryptographic processor. *IEEE Transactions on Computers*, 2003, 52(4): 449-460
- [8] Leong P H, Leung I K. A microcoded elliptic curve processor using FPGA technology. *IEEE Transactions on VLSI Systems*, 2002, 10(5): 550-559
- [9] Corporaal H. *Microprocessor Architecture from VLIW to TTA*. West Sussex, England; John Wiley & Sons Ltd, 1998
- [10] Cohen H, Miyaji A, Ono T. Efficient elliptic curve exponentiation using mixed coordinates//Proceedings of Advances in Cryptology (ASIACRYPT'98). Beijing, China, 1998; 51-65
- [11] Montgomery P L. Modular multiplication without trial division. *Mathematics of Computation*, 1985, 44(170): 519-521
- [12] Koh G K, Acar T, Kaliski B S. Montgomery multiplication in $GF(2^h)$. *Designs, Codes and Cryptography*, 1998, 14(1): 57-69
- [13] Zhao Xue-Mi, Lu Hong-Yi, Dai Kui, Wang Zhi-Ying, Tong Yuan-Man. SEA: A high-performance modular long integer exponentiation coprocessor. *Journal of Computer Research and Development*, 2005, 42(6): 924-929(in Chinese)
(赵学秘, 陆洪毅, 戴 葵, 王志英, 童元满. SEA: 一种高性能大数模幂协处理器. *计算机研究与发展*, 2005, 42(6): 924-929)



ZHAO Xue-Mi, born in 1979, Ph. D. candidate. His research interest is programmable cryptographic processors design.

research interests include computer security and asynchronous microprocessors design.

YUE Hong, born in 1980, Ph. D. candidate. Her research interest is high performance microprocessors design.

LU Hong-Yi, born in 1974, Ph. D., associate professor. His research interest is VLSI design.

DAI Kui, born in 1968, Ph. D., associate professor. His research interests include security chip design, anti-attack technologies for chip.

WANG Zhi-Ying, born in 1956, Ph. D., professor. His

Background

Compared with other commonly used public key cryptosystems such as RSA and discrete logarithm, Elliptic Curve Cryptography (ECC) has several benefits that make it particularly suitable for embedded applications: ECC offers the highest security per bit; a smaller memory can be used; ECC needs less computation power. Although the mathematical theory of ECC has become perfect, understanding and implementing its algorithms is much more difficult. Previous research of ECC processors always designed a coprocessor for performing the underlying field operations and implemented the whole algorithm by a host processor. This method has several shortcomings in security, speed and hardware/software co-design. An ideal ECC processor should fulfill the following requirements: (I) handling the whole algorithm to

prevent attacks from software; (II) being convenient to develop the whole public key cryptosystem; (III) implementing ECC in real time; (IV) supporting Galois fields $GF(p)$ and $GF(2^n)$; (V) being scalable in power and area for different application environment. In this paper, an ECC processor TTA-EC that has these five characters is presented, which is extended from transported triggered architecture by coupling a modular multiplier and long integer registers. This work is supported by the National Natural Science Foundation of China grant No. 60173040. The aim of this project is finding an efficient way to design application specific microprocessors. One of its applications is a high speed modular exponentiation processor SSX26, which has been sold in China. And this work is another application for ECC algorithms.