

SM2 高速双域 Montgomery 模乘的硬件设计

郭 晓, 蒋安平, 宗 宇

(北京微电子技术研究所, 北京 100076)

摘 要: 作为由国家密码管理局公布的 SM2 椭圆曲线公钥密码算法的核心运算, 模乘的实现好坏直接决定着整个密码芯片性能的优劣. Montgomery 模乘算法是目前最高效也是应用最为广泛的一种模乘算法. 本文基于 Montgomery 模乘算法, 设计了一种高速, 且支持双域($GF(p)$ 素数域和 $GF(2^m)$ 二进制域)的 Montgomery 模乘器. 提出了新的实现结构, 以及一种新型的 Wallace 树乘法单元. 通过对模块合理的安排和复用, 本设计极大的缩小了时间消耗与硬件需求, 节省了大量的资源. 实现 256 位双域模乘仅需 $0.34\mu s$.

关键词: SM2; 模乘运算; Montgomery 算法; 双域实现

中图分类号: TN918

文献标识码: A

文章编号: 1000-7180(2013)09-0017-05

A High Speed Structure for Dual-Field Montgomery Modular Multiplication in SM2

GUO Xiao, JIANG An-ping, ZONG Yu

(Beijing Microelectronics Technology Institute, Beijing 100076, China)

Abstract: Being the key algorithm of SM2, the special elliptic curve cryptography algorithm presented by OSCCA, modular multiplication defines the system's overall performance. One of the most efficient and widely used modular multiplication algorithms is Montgomery modular multiplication algorithm. This paper presents a new hardware architecture to realize modular multiply in dual field ($GF(p)$ and $GF(2^m)$) based on the improved dual-field Montgomery modular multiplication's algorithm. A new kind of Wallace tree multiplier is also presented in this paper. This multiplier could realize higher performance with less computing resource and lower timing consumption due to the reasonable arrangement and reusing of modules. One complete operation of 256 bit dual-field modular multiplier can be finished in $0.34\mu s$.

Key words: SM2; modular multiplication; Montgomery algorithm; dual field realization

1 引言

密码学已经应用到了社会的各行各业中, 为信息安全提供了强有力的保障. 椭圆曲线加密算法(ECC)相对于其他公钥密码算法具有计算速度快、存储空间小、带宽要求低等优点, 特别适用于各种无线设备和智能卡之类计算资源受限的设备^[1], 成为最有希望的公钥密码算法. 国家密码管理局公布了 SM2 椭圆曲线公钥密码算法, 在我国商用密码体系中被要求用来替换 RSA 算法. SM2 算法是基于椭圆曲线算法(ECC)的一种公钥密码算法, 它具有椭圆

曲线密码算法的多种优点, 更是一种更加先进、更加复杂的算法.

模乘运算是 SM2 椭圆曲线加密算法中的核心运算. 在众多模乘算法中, Montgomery 模乘是应用最为广泛的一种. 它用简单的加法、乘法、移位操作取代了传统模乘算法中繁琐的除法运算, 在加快模乘运算速度和效率的同时, 也非常适合硬件实现^[2]. 本文提出了一种新颖的双域 SM2 算法模乘单元; 设计了基于改进 Montgomery 模乘算法的模乘器, 利用计算单元的复用在有效减小计算单元面积的同时, 提升了计算效率.

2 Montgomery 模乘算法

2.1 多精度双域 Montgomery 模乘

本文需要设计支持素数域和二进制域的双域模乘单元,因此,要在 Montgomery 基本算法的基础上进行改进,使其能够同时实现素数域和二进制域的运算.多精度双域 Montgomery 模乘算法^[3]如下:

```

INPUT: A, B, N, w, k, field
OUTPUT: C ∈ [1, p-1]
C=0, S=0
for i=0 to u-1
    (s|c0) = (aib0) ⊕ c0
    q=f(c0, N0, field)
    (s|c0) = (qN0) ⊕ (s|c0)
    for j=1 to e-1
        (s|cj) = (aibj) ⊕ cj ⊕ s ⊕ qNj
        cj-1 = (cj|cj-1)/r mod w
    ce-1 = (s|ce-1)/r mod w
    ce = 0
    if C>N, then C=C-N when over GF(p)
    GF(p): f(c0, N0, field) = (c0(2k-N0-1)) mod r
    GF(2m): f(c0, N0, field) = (c0N0-1) mod r

```

该算法将操作数看作为基于机器字长的操作序列,然后逐字扫描进行计算.设机器字长为 k 位,操作数 A 长度为 l_a ,操作数 B 长度为 l_b . $u = \lceil l_a/k \rceil$, $e = \lceil l_b/k \rceil$, radix = 2^k . A 和 B 就可以表示为 $A = (A_u \dots A_1 A_0)_{\text{radix}}$, $B = (B_e \dots B_1 B_0)_{\text{radix}}$ 即: $A = \sum_{i=0}^u A_i \cdot \text{radix}^i = \sum_{i=0}^e B_i \cdot \text{radix}^i$.

除了 $f(c_0, N_0, \text{field})$ 表达式的不同,二进制域和素数域上的算法的差异可以体现在两个部分:一是双域加法部分,素数域上为正常的加运算,有进位的传播,二进制域上则只是异或运算无进位,在算法中统一用 $\oplus$ 表示;二是运算最后的处理部分,素数域下需要一个比较单元进行处理,而二进制域下则不需要.

利用该算法,可以在同一个硬件上实现素数域 $\text{GF}(p)$ 与二进制域 $\text{GF}(2^m)$ 的模乘,大大节约了成本.另外,基于该算法具有可伸缩性,能支持任意位长的模乘.

2.2 改进多精度双域 Montgomery 模乘算法

函数 $f(c_0, N_0, \text{field})$ 在有限域 $\text{GF}(p)$ 上为 $(C_0(2^k - N_0^{-1})) \bmod r$, 在有限域 $\text{GF}(2^m)$ 上为 $(C_0 N_0^{-1}) \bmod r$. 通过对 Montgomery 算法的研究发现,函数 f

(c_0, N_0, field) 每一次循环都要计算一遍,浪费时间和资源,其中有相当一部分的计算是完全重复的,如:计算 N_0^{-1} 和 $2^k - N_0^{-1}$, N_0 和 k 的值在整个计算过程中是确定不变的,因此 N_0^{-1} 和 $2^i - N_0^{-1}$ 的值始终不变.为了减少整个运算的消耗,可以预计算 $(2^k - N_0^{-1}) \bmod r$ 和 $(N_0^{-1}) \bmod r$ ^[4]. 改进算法如下:

```

INPUT: A, B, N, w, k, field,
q=(2k-N0-1) mod r, when GF(p),
q=(N0-1) mod r, when GF(2m)
OUTPUT: C ∈ [1, p-1]
C=0, S=0
for i=0 to u-1
    (s|c0) = (aib0) ⊕ c0
    Ti=qc0/r
    (s|c0) = (TiN0) ⊕ (s|c0)
    for j=1 to e-1
        (s|cj) = (aibj) ⊕ cj ⊕ s ⊕ qNj
        cj-1 = (cj|cj-1)/r mod w
    ce-1 = (s|ce-1)/r mod w
    ce = 0
    if C>N, then C=C-N when over GF(p)

```

3 硬件实现

3.1 算法流程图

文中根据改进多精度双域 Montgomery 模乘算法,提出双域模乘单元的算法流程图.以 256 位 Montgomery 模乘为例,算法流程图如图 1 所示.取机器字长为 32 位,采用流水线结构, M1 和 M3 分别表示两种双域乘法器, DFA 表示双域加法器, 4-2 表示双域 4-2 压缩器,具体将在下面介绍.

3.2 模块划分

根据算法流程图,本文将整个模乘计算分为 5 个模块来计算,这五个运算处理单元 A、B、C、D、E,分别如图 2 所示.运算处理单元 A 和 B 完成的是算法中的外层循环,即完成 i 从 0 变到 7, $j=0$ 部分的计算.运算处理单元 C 完成的是内层循环中的一部分计算,即完成 i 从 0 到 7, j 从 1 到 7 时求 $A_i * B_j$ 与 $N_j * T_i$ 之和的计算.剩下的内层循环的计算由运算处理单元 D 和运算处理单元 E 来完成.运算处理单元 D 对运算处理单元 C 的计算结果与 C_i 、 S 进行求和,求出 C_{j-1} 或 C_7 .

运算处理单元 E 是专门用来在素数域情况下,对之前计算得到的 $C = (C_7, C_6, \dots, C_0)$ 与 N 进行比较的,如果 $C > N$, 则 $C = C - N$, 否则 C 值不变,最后

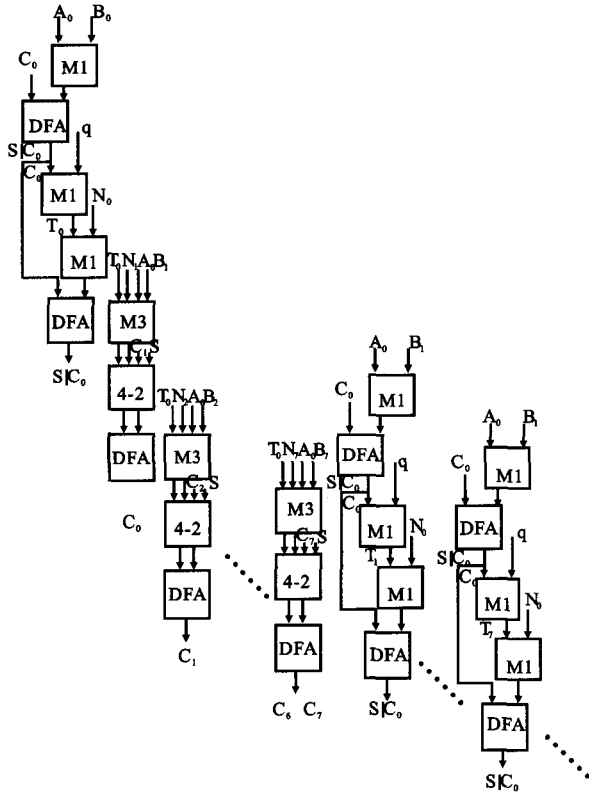


图1 改进多精度双域 Montgomery 模乘算法流程图

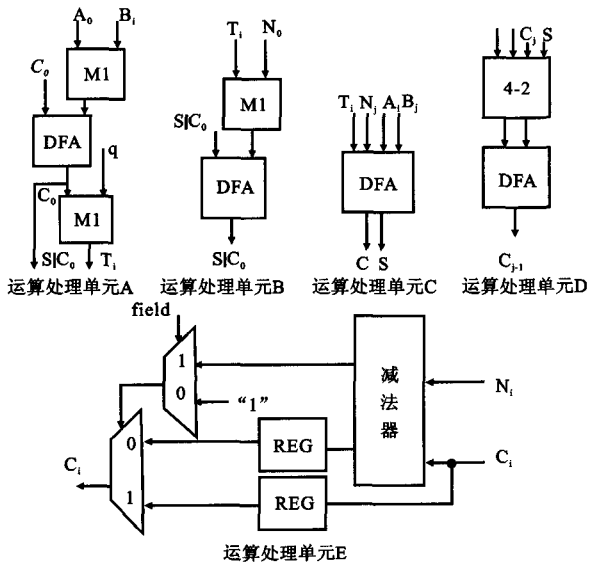


图2 运算处理单元结构图

输出 C。

本文采用的比较器是在计算模乘的同时进行比较的,不仅节省了时钟周期,而且也减小了比较器的面积。由于二进制有限域上并不需要进行比较与减法操作,因此添加了域选择信号控制端,使得二进制域上的模乘运算可以不受此比较单元的影响,直接输出为 C。field=0 时表示是二进制域,field=1 时表示是素数域。在素数域下,减法运算结果和运算处

理单元 D 计算的结果 $C = (C_7, C_6, \dots, C_0)$ 均保存在内部寄存器中,将减法器计算最后的借位信号作为第一级多路选择器的一个输入端,对后续输出进行控制。若 $C > N$,则减法器最后的借位信号为 0,最终选择输出的为减法器的计算结果, $C = C - N$,若 $C < N$,则借位信号为 1,输出 C 值不变。

每一次外层循环开始时,运算处理单元 A 的计算都需要等待前一次外层循环中运算处理单元 B 计算出 C_0 值后才可以开始计算。在每一个内层循环中,运算处理单元 A 计算出 T_0 后,运算处理单元 C 才开始计算,运算处理单元 D 则需要等待运算处理单元 C 和 B 计算出结果之后才可以开始计算,根据这些数据依赖关系得出流水线结构。

整体模乘运算结构示意图如图 3 所示。数据输入、输出单元与外部数据接口宽度均设为 32 位,数据输入输出,数据在各个运算处理单元中传递、计算均在状态控制模块的控制下完成。

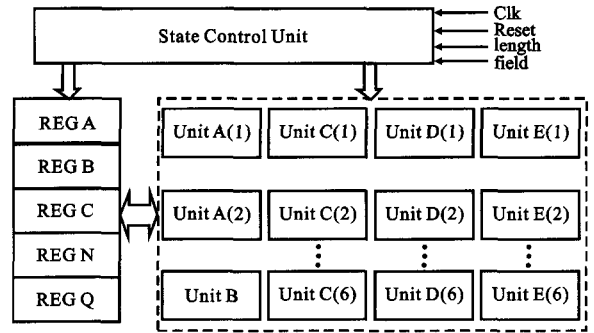


图3 模乘运算整体结构图

3.3 运算单元设计

3.3.1 双域乘法单元设计

根据算法描述,采用 $32\text{bit} \times 32\text{bit}$ 类型乘法器,若采用常规乘法器设计,会产生 32 个部分积,再将这 32 个部分积一一相加,计算量很大。为了减少部分积数量,本文采用 BOOTH 编码来实现乘法。该运算可分为 2 步来实现:(1)产生部分积;(2)部分积的累加。采用二阶 BOOTH 编码对乘数进行编码,可以将部分积数量从 32 个减少为 17 个。二阶 BOOTH 编码部分积产生电路如图 4 所示。

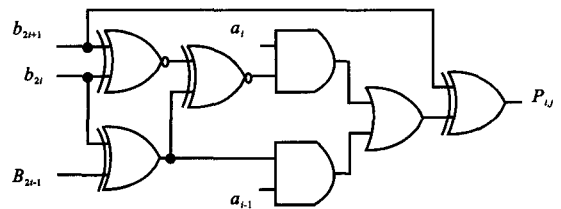


图4 BOOTH 编码部分积产生电路

采用双域 Wallace 树对 BOOTH 编码产生的部

分积进行相加计算. 将部分积按列进行分组, 每一列对应一组加法器, 各列相同权重的值进行压缩相加, 得到的进位传到权值高一位的前一列, 得到的伪和传到同权值的下级. 由于整个算法是支持素数域和二进制域的, 所以设计的乘法单元也是需要支持双域的, 这就需要将实现部分积加法的加法器也要设计为双域加法器.

由于在外部循环中用到的乘法器是求 $A_i * B_0$ 、 $T_i * N_0$ 或者 $q * C_0$, 且得到的结果都是与一个值直接相加, 或作为另一个乘法器的输入. 此种情况下, 我们采用 Wallace 树设计, 在完成最后一个压缩后, 再用一个加法器对得到的伪和和进位相加, 最后输出最终的乘积, 这个乘法器我们命名为 M1, 其结构如图 5 所示. 4-2 表示双域 4-2 压缩器, DFA 表示双域加法器. Control 信号为 BOOTH 编码产生的进位一取反信号, 需要将其与部分积之和相加, 才能得到最终正确结果.

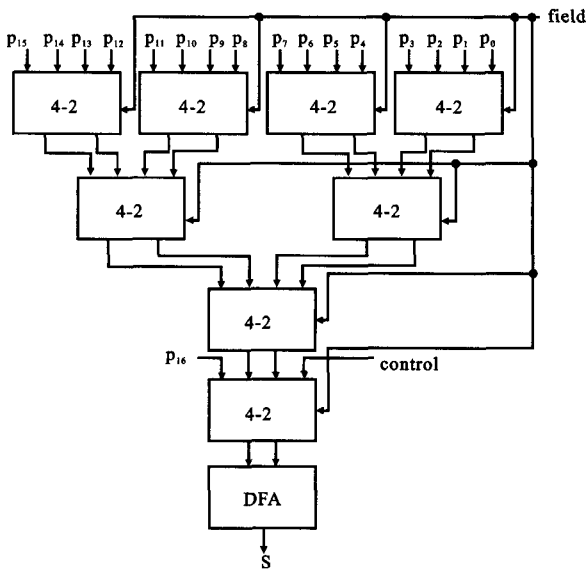


图 5 乘法器 M1

在内层循环中, 用到的乘法器是求 $A_i * B_j$ 和 $T_i * N_j$, 计算结果是直接相加的, 可以将这两个乘法结合起来实现: 首先通过两个 Wallace 树 M2, 如图 6 所示. 对两个乘法产生的部分积分别压缩相加, 在 Wallace 树末端 4-2 压缩计算完成后, 输出 4 个值. 再将这些值通过一个 4-2 压缩器进行相加, 最后, 再通过一个 DFA 将伪和和进位相加, 得到最终结果. 整体的实现结构如图 7 所示. 命名为 M3—改进双域 Wallace 树乘法器. 与直接利用两个乘法器 M1 相加并通过一个 DFA 相加输出结果的方式相比, 本文结构节省资源, 同时提高了速度.

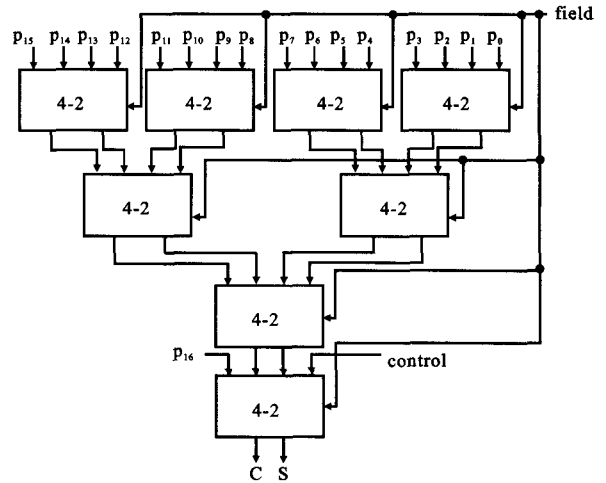


图 6 乘法器 M2

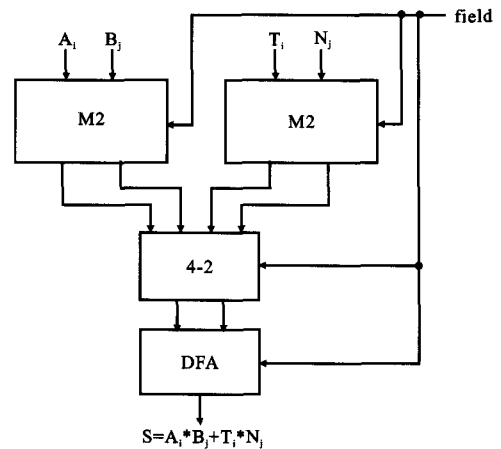


图 7 改进双域 Wallace 树乘法器 M3

3.3.2 双域加法器设计

本设计中涉及到的加法器有两种, 分别为对部分积进行迭加时用到的双域 4-2 压缩器和计算最终结果的 DFA, 它们都是支持双域的. 当域选择信号 $field=1$ 时, 它们执行的是素数域上的加法, 这与基本加法的实现没有差别; 当域选择信号 $field=0$ 时, 它们执行的是二进制域上加法, 即异或运算, 进位输出恒为 0.

本文采用的双域 4-2 压缩器结构如图 8 所示. 该电路由异或门和 2:1 数据选择器组成, 从不同数据输入到输出的延时比较均衡, 能够同时产生 Sum 和 Carry 信号. 这种结构的 4-2 压缩器性能最好, 结构规整, 具有高速、低功耗、节约面积的优点.

DFA 主要是用来处理 64 位加法, 因此对于运算速度有较高的要求. 针对这种需求, 我们的研究重点集中于如何快速产生进位信号以提升运算速度. 在众多加法器实现结构中, 超前进位加法器是最快的. 而在多种超前进位算法中, Kogge—Stone 算法是最快的一种, 其进位信号计算时间仅为 $\log_w n$ 级

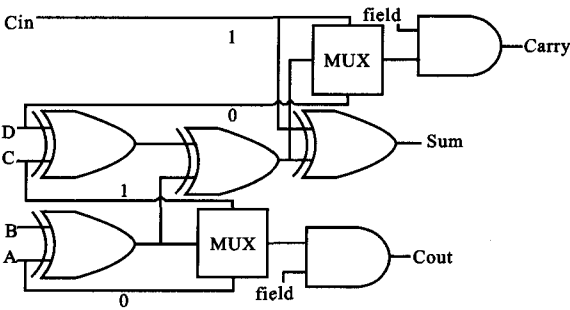


图 8 双域 4-2 压缩器

门的延迟,其中 w 是组的大小, n 是加法器的位数^[5],因此,我们采用 64 位基 4 Kogge—Stone 超前进位加法器来实现 DFA,结构如图 9 所示。

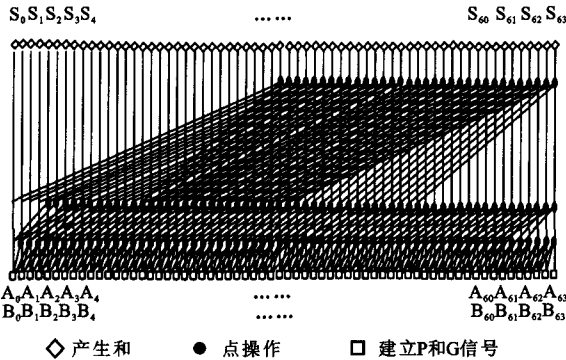


图 9 64 位基 4Kogge—Stone 超前进位加法器

4 仿真结果

利用 Modelsim SE PLUS 6.5 对编写的 Verilog Vhdl 代码进行仿真,实现 256 位模乘计算总共需要两个运算处理单元 A,一个运算处理单元 B,6 个运算处理单元 C,6 个运算处理单元 D 和 6 个运算处理单元 E,花费周期为 49 个周期。

在设计的功能得到验证后,我们又利用 Synopsys Design Compiler 对设计进行综合,选用 SMIC 0.18 μ m 工艺库,频率可达 140MHz,计算 256 位模乘仅需 0.34 μ s。在表 1 中,将本设计综合得到的结果与其他文献中模乘实现的结果进行比较。

表 1 模乘器性能比较

	位数/bit	工艺/ μ m	时间/ μ s	频率/MHz
本文	256	0.18	0.34	143
文献[3]	256	0.5	6.60	80
文献[6]	256	0.18	0.81	384
文献[7]	256	0.18	0.28	250

	周期数	面积/ 10^3	域
本文	49	150	双域
文献[3]	528	—	双域
文献[6]	312	—	双域
文献[7]	70	360	GF(p)

本文设计的模乘单元,与可参考的设计相比,在速度上有较大改进,可以将模乘运算时间减少至 0.34 μ s,同时由于模块的复用设计,使得整体模乘单元的面积得到有效控制。与文献[7]相比,本设计的速度稍低,但是却极大的缩小了模乘单元的面积。因此,综合起来看,本设计既保证了较小的面积开销,又实现了高速的模乘运算,具有优越的性能。

5 结束语

本文提出了一种基于改进型 Montgomery 模乘算法的新型模乘运算硬件实现结构,通过对模块合理的安排和复用,在提升计算速度的同时减少了面积需求。将两个 Wallace 乘法器结合成一个新型 Wallace 树来实现两个乘积相加的运算,进一步的节省了整体运算的时间消耗。还利用 4-2 压缩器、Kogge—Stone 超前进位加法器、BOOTH 编码来减少 Wallace 树所需级数,以及乘法部分积数目。所有运算单元都采用双域设计,节省了资源消耗。

参考文献:

[1] Janagan M, Devanathan M. Area compactness architecture for elliptic cryptography[C]// 2012 International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME). USA: Hawaii, 2012:131-134.

[2] Dimitris Schinianakis, Thanos Stouraitis. A RNS montgomery multiplication architecture[C]// 2011 IEEE Symposium on Circuits and System. France: Nice,2011:1167-1170.

[3] Tenca F, Savas E, Koc C K. A design framework for scalable and unified multipliers in GF(p) and GF(2^m) [J]. International Journal of Computer Research, 2004,13(1):68-83.

[4] 童元满. 高性能密码协处理器的设计与实现[D]. 长沙:国防科学技术大学,2004:60-64.

[5] 赵翠华,姜冕,张洵颖,等. 一种改进的基于 Kogge—Stone 结构的并行前缀加法器[J]. 微电子与计算机, 2011,28(2):47-50.

[6] 史焱,吴行军. 高速双有限域加密协处理器设计[J]. 微电子学与计算机, 2005, 22(5): 8-12.

[7] 刘建国,张军,杨晓辉,等. 有限域模乘专用指令设计[J]. 计算机工程,2011,37(21):105-107.

作者简介:

郭 晓 女,(1988—),硕士研究生. 研究方向为 VLSI 与 SOC 设计。