小型微型计算机系统
Journal of Chinese Computer Systems

2007年9月 第9期
Vol. 28 No. 9 2007

专用指令集安全处理器设计与VLSI实现

陆荣华，曾晓洋，韩 军，顾叶华，麦 浪

(复旦大学 专用集成电路与系统国家重点实验室，上海 201203)
E-mail：xyzeng@fudan.edu.cn

摘 要：专用指令集处理器(ASIP)结合了ASIC协处理器的高效性与通用处理器的灵活性，在信息安全领域具有广泛的应用前景．本文针对RSA/ECC密码算法，提出了一种专用指令集安全处理器的设计与VLSI实现方案．本文的ASIP基于32位RISC架构，通过采用专用的指令集和特殊的运算单元，以较小的软硬件代价实现了密码算法的高效运算．本设计采用TSMC 0.25μm标准CMOS工艺综合，核心电路等效门为28K，最高时钟频率可达到150MHz，完成一次1024位RSA算法仅需200毫秒．

关 键 词：专用指令集处理器；RISC；RSA；ECC；安全
中图分类号：TP309；TN918 **文献标识码**：A **文 章 编 号**：1000-1220(2007)09-1724-05

Design and VLSI Implementation of an Application Specific Instruction Set Security Processor

LU Rong-hua, ZENG Xiao-yang, HAN Jun, GU Ye-hua, MAI Lang
(*State-Key Lab of ASIC and System, Fudan University, Shanghai* 201203,*China*)

Abstract: The ASIP, which features the high efficiency of ASIC co-processor and the flexibility of General Purpose Processor, has and will have popular application in information security domain. This paper presents a new design and VLSI architecture of Security ASIP for RSA/ECC of cryptographic algorithms. By adopting the special intruction set and computing unit, the proposed 32-bit RISC processor achieves the goal of high-speed and low-cost. Based on TSMC 0.25μm standard CMOS technology, the core circuit of this Security ASIP has about 28k gates, and a max frequency of 150MHz , under which only 200 ms is required when the Security ASIP excutes a 1024-bit RSA algorithm.

Key words: ASIP; RISC; RSA; ECC; security

1 引 言

信息安全领域需要进行高强度的密码学运算，而通用嵌入式处理器在运算性能上有明显的局限性，因此硬件加速方法在信息安全系统中得到了广泛应用．

目前比较普遍的硬件加速方法是针对特定的密码算法设计硬件加速器(ASIC)，并与作为主控制器的通用处理器(GPP)组成SoC系统．这种方法成本高，系统复杂，而且系统运行效率受片上总线传输能力的限制．另外这类ASIC硬件加速器只能实现单一、固定的功能，可配置性和可扩展性十分欠缺．为了克服传统方案的缺陷，目前专用指令集处理器(ASIP：Application Specific Instruction Set Processor)作为一种新的设计方案得到了重视和发展[12]．ASIP具有与GPP类似的架构，软件可配置性强；ASIP能够针对特殊应用优化指令集和数据通路，对算法的关键运算步骤进行硬件加速，因此还具有硬件执行速度快的特点；ASIP还可以灵活地扩展其指令集和相应的硬件单元，在不同的应用条件下具有良好的适应性，从而有效地降低设计成本．

本文采用ASIP的设计方案，在通用32位RISC结构处理器的基础上，设计专用指令集的安全处理器．该处理器用较小的软硬件代价实现高速的模乘和模逆运算，并在此基础上完成完整的RSA、ECC(GF(P))算法．另外本文也提出了ASIP运算单元和指令集扩展的具体方法．为了实现与外部设备的通信，本文的ASIP设计能够支持一个中断．

2 可伸缩的蒙哥马利模乘算法

模乘运算是RSA和ECC算法的核心步骤．目前应用的RSA算法数据位宽已经达到512～4096位，用硬件直接实现如此超长位宽的运算，硬件成本和关键路径延迟都会很大，而字串行架构(digit-serial)可以将位宽为n的模乘运算分解为字为w(w=n/m)的运算，易于将模乘运算集成到位宽较短的处理器结构中．另外，字串行的模乘运算可以实现数据宽度的可伸缩性，无需改变硬件就可以实现算法在不同条件下的应用．模乘运算一般采用蒙哥马利模乘实现[1]，[5]提出了一种适用于硬件实现的可伸缩蒙哥马利模乘算法Scalable_Monpro．下面是对该算法的分析．

收稿日期：2006-06-13 基金项目：国家"八六三"高技术项目(2003AA1Z1270)资助；上海市科委重大攻关项目(03dz15001)资助． 作者简介：陆荣华，男，1982年生，硕士研究生，研究方向为集成电路设计；曾晓洋，男，1972年生，博士，副教授，研究方向为集成电路设计；韩 军，男，1977年生，博士研究生，研究方向为集成电路设计；顾叶华，男，1981年，硕士研究生，研究方向为集成电路设计；麦 浪，男，1983年，硕士研究生，研究方向为集成电路设计．

```
Scalable_Monpro(可伸缩蒙哥马利模乘算法)
Input: A∈[1,N-1],B∈[1, N-1], N=modulus;
Output: ((A×B)×(2^-n) mod N, n=number of bits of N.
reg_S[n+1:0]←0; Qs←0;
for(i=0;i<=n/w-1;i=i+1)
  {op1←A[i×w+w-1:i×w];
   S[2w+1:0]←0;
   for(j=0;j<=n/w-1;j=j+1)
   {op2←B[j×w+w-1:j×w];
    S[2w+1:w]←(S[2w+1:w]+reg_S[j×w+w-1:j×w];
    S←W_Monpro(op1,op2,S[2w+1:0],N[j×w+w-1:j×
      w],Qs,j);
                  /*W_Monpro是字级蒙哥马利模乘算法*/
    if(j>0)then{reg_S[j×w-1:j×w-w]←S[w-1:0];}
    if(j=n/w-1)then{reg_S[n+1:n-w]←S[2w+1:w]+
      reg_S[n+1:n];}
   }
  }
if(reg_S>N)then{reg_S←reg_S-N;}
return reg_S;
```

Scalable_Monpro 将蒙哥马利模乘拆分成字级模乘.

```
W_Monpro(字级蒙哥马利模乘算法)
BN=B+N;/*预处理*/
Input: op1=any w bits of A, op2=any w bits of B, 2w+2 bits
       S_i-1, BN[w-1:0], N[w-1:0], Qs[w-1:0], flag;
Output: S_w-1=2w+2 bits product
for(i=0;i<=w-1;i=i+1)
  {if(flag=0)then Q[i]←(S_i-1[w]+op1[i]×op2[0]) mod 2;
   else {Q[i]←Qs[i]};
   case{op1[i],Q[i]}:
     "00":temp←S_i-1[2w+1:w];
     "01":temp←S_i-1[2w+1:w]+N;
     "10":temp←S_i-1[2w+1:w]+B;
     "11":temp←S_i-1[2w+1:w]+BN;
   S_i[2w+1:w]←temp[w+2:1];
   S_i[w-1]←temp[0];S_i[w-2:0]←S_i-1[w-1:1];
  }
 Qs←Q;
Return S_w-1;
```

W_Monpro 实现位宽为w的字级蒙哥马利模乘.算法包含了重复的加法和右移操作,其中加法运算的操作数必须由参数的值来决定,这在GPP实现中将需要大量的指令和运算时间,从而限制了模乘运算的效率.本文的解决方案是由硬件直接计算参数,避免两次加法之间由于软件计算参数而造成的大量延迟.

由于能够在当前加法运算的同时确定下一次加法运算所需的参数,本文设计了一条多周期指令SCAL完成**W_Monpro**算法,在字长为w的情况下,SCAL指令只需执行w个时钟周期,与GPP相比节省了大量的运算时间.

Scalable_Monpro 算法可以通过循环调用 **W_Monpro** 来实现,下面是其汇编程序的简单示意.

```
        counter1=m;/*参数,运算位宽=64×m*/
   LOOP:
        counter2=m;
SCALLOOP:…/*取第i次运算相关数据,如Ai,Bi,Ni*/
        SCAL
        …/*保存运算结果*/
        counter2=counter2-1;
        if(counter2=0) Jump to SCALLOOP
        counter1=counter1-1
        if(counter1=0) Jump to LOOP
        else           Jump to OUT /*END*/
```

本文设计的ASIP由硬件完成字级运算,而通过软件调度字级运算实现任意长度的蒙哥马利模乘,这样的软硬件划分使本文的ASIP同时具有ASIC的高效性和GPP的灵活性与可编程性.

3 处理器硬件设计

3.1 处理器总体结构

本文ASIP的硬件结构如图1所示,主要分为7个部分,寄存器堆(REGFILE),运算单元(ALU,FU),前推逻辑(FW),译码单元(DECODE),地址产生单元(AGU),存储器(指令存储器ROM/RAM,数据存储器RAM)和中断控制单元(INT).

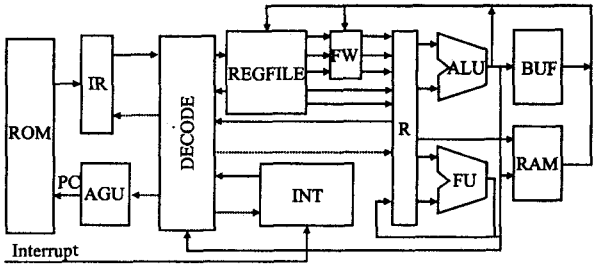

图1 专用处理器总体结构

Fig. 1 Hardware architecture of security ASIP

与经典的五级流水线结构相似,AGU生成指令地址PC,并从ROM取出32位指令;指令寄存器IR对指令进行锁存;译码单元对锁存的指令译码,产生各个单元的控制信号,并从REGFILE读取数据;REGFILE是一个32×32的寄存器堆,用于存放数据,其中寄存器R0在系统复位后恒为0;FW对来自REGFILE的数据进行校正;寄存器R锁存数据以及ALU控制信号;运算单元包括ALU和其它运算模块(FU),ALU输出运算结果状态位至译码单元,FU只在执行特殊算法时工作;ALU产生的数据如果是地址,则访问数据RAM,如果不是则经过BUF缓冲,以使各种指令经过相同级数的流水线;最后REGFILE根据BUF,RAM的输出结果更新寄存器的值.

目前大多数针对密码学算法的处理器都采用MAC单元完成一次字模乘[3],本文不包含乘法器单元,并只对ALU的

结构和主要数据通路做优化，从而减小硬件开支，同时避免乘法器成为关键路径.

3.2 运算单元设计

[5]设计了两个加法器 MSB-CSA 串联的结构用于模乘运算，本文在这种结构的基础上对低位的 MSB-CSA 做了改进，使之能够作为处理器 ALU 的主要部件，并添加了逻辑运算单元执行逻辑操作指令.

运算单元执行蒙哥马利模乘运算时，功能如[5]所述；执行其它运算时，reg_carry 作为进位寄存器. 加法运算时，reg_carry 保存 carry，减法时保存～carry；逻辑操作时，reg_carry 保存逻辑运算单元进位输出 shift_out；中断响应进入或退出时，reg_carry 保存 reg_carry_int. reg_carry_int 为中断响应状态下的进位寄存器.

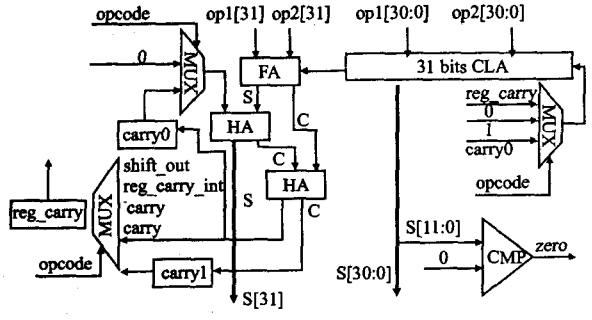

图 2 运算单元主要结构

Fig. 2 Operation unit

GPP 通常对 ALU 的 32 位结果判断是否为 0，产生状态位并返回 DECODE 译码，这种设计主要为了方便编程时设置计数器，但也往往成为系统的关键路径. 本文只对低 12 位结果判断是否为 0，这样做既能够满足软件编程时计数范围的要求($2^{12}=4096$)，也能缩短关键路径(12 位比较器取代了 32 位比较器).

3.3 数据通路优化

图 3 是本文 ASIP 的主要数据通路. 运算单元由一个 MSB_CSA 和一个以 MSB_CSA 为基础构造的 ALU 组成，ALU 的输出 carry1 与 MSB_CSA 的进位输入相连. 保留进位缩短了关键路径，同时在模乘运算右移累加过程中，进位可以直接加入本位位做下一次运算. 虽然字级模乘运算的位宽是 64，但是关键路径只与 32 位加法器有关.

与 GPP 的寄存器堆不同，本文的 ASIP 寄存器 R2-R15 既可以作为通用寄存器，又可以在模乘运算时作为特殊寄存器存放运算所需数据，如 R14，R15 分别保存 64 位 Q 值的低位和高位，R6，R7 保存 64 位的 A 值，并能够实现循环右移，从而在 R6 的最低位输出每个运算周期所需的算法参数 A[i](a_lsb).

本文 ASIP 在执行普通指令时，译码单元与 ALU 的操作过程和 GPP 一致；而在执行模乘指令时，当前运算的结果 stmp[64:0]直接返回作为下个周期运算单元的一部分操作数，同时译码单元根据 a_lsb，b_lsb(B 的最低位)，q_lsb(如果 flag＝0 则选择 stmp[1:0])的值对特殊寄存器寻址，产生其余的操作数，从而实现高效的模乘运算. 本文的ASIP在规范的 GPP 架构上，结合特殊的应用合理地优化了数据通路，大幅度地提高了运算效率.

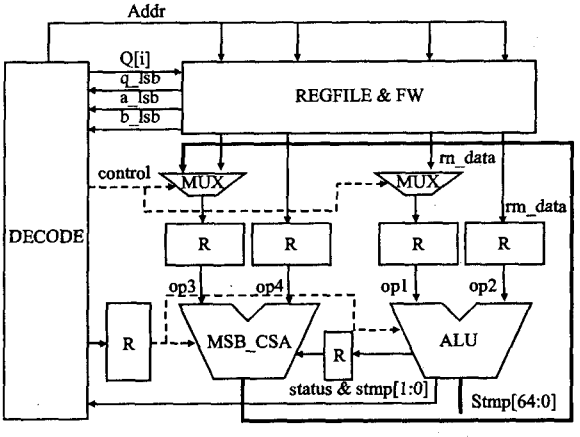

图 3 主要数据通路

Fig. 3 Main data path

3.4 中断控制设计

中断控制电路如图 4 所示. 译码模块输出 int_cs 和 int_over，当系统能够响应中断时 int_cs 有效；当中断响应结束时 int_over 有效；有限状态机输出 int_out，当系统不处于中断响应状态时有效. 当 int_cs 和 int_out 均有效，且收到中断请求(int_en 有效)，则 respond 有效，系统响应中断.

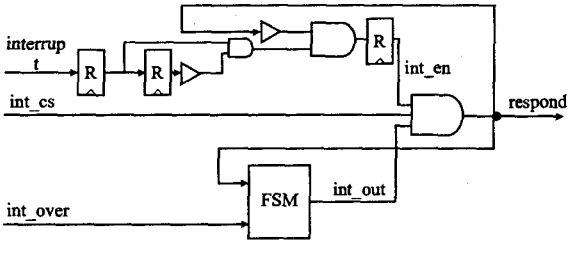

图 4 中断控制电路

Fig. 4 Interruption control block

ALU 内部设置了两组状态寄存器，系统响应中断或从中断响应退出时，两组状态寄存器的值完成一次互换. 为了减小中断现场保护与恢复的硬件代价，寄存器数据的保存通过软件调度完成.

4 专用指令集

本文 ASIP 专用指令集共有二十一条指令，为了提高代码密度，所有指令均为条件执行. 指令集包括数据处理指令 ADD(C)，SUB(C)，MOV，LS(C)，RS(C)，CMP，TST；控制流指令(A)JMP，(A)CALL，RET(I)；存取指令 LOAD，STR；多周期模乘指令 SCAL，模乘调整指令 AMD.

基于该指令集，可以通过编程实现模乘和模逆运算. RSA 算法的模幂以及 ECC(GF(P))算法的点乘因此都可以通过

重复调用模乘与模逆运算实现.

表1 专用指令集

Table 1 Application specific instruction set

指令	说明	指令	说明
ADD	加法	JMP	相对地址跳转
ADDC	带进位加法	AJMP	绝对地址跳转
SUB	减法	CALL	相对地址调用
SUBC	带借位减法	ACALL	绝对地址调用
MOV	寄存器数据传输	LOAD	存储器取数据
LS	逻辑左移	STR	存储器存数据
LSC	带进位逻辑左移	RET	子程序返回
RS	逻辑右移	RETI	中断返回
RSC	带进位逻辑右移	SCAL	模乘指令
CMP	数值比较	AMD	模乘调整指令
TST	最高比特位检测	/	/

4.1 特殊指令

本文的专用指令集包含了若干特殊指令,其中多周期模乘指令SCAL完成W_Monpro算法,它保证了模乘运算的高效性.另外在模乘时,运算单元执行了最高位保留进位加法,因此本文设计了一条AMD指令对运算结果进行调整.AMD指令在整个模乘运算中所占用的执行时间不到0.1%(以1024位为例),但它可以保证运算结果的正确性.

鉴于模逆运算[13]在密码学领域的重要作用,本文ASIP的LS(C),RS(C)指令支持模逆中频繁的移位操作,从而提高了运算效率.

在模幂运算需要对指数的每一比特位进行扫描,本文ASIP的TST指令与移位指令配合后就可以实现这一功能.TST指令避免了将ALU运算结果的最高位作为状态位,虽然需要额外增加一条指令判断数据最高比特位的值,但是缩短了关键路径.

4.2 性能分析与扩展方法

下面分别用本文ASIP专用指令集和ARM7TDMI指令集完成**W_Monpro**算法,见表2.

表2 两种指令集完成W_Monpro的比较

Table 2 Comparision of two instruction sets

专用指令集	ARM7TDMI
	add r0,r0,#1
/*数据载入特殊寄存器*/	cmp r0,#0x20
SCAL	ble loophead
/*数据存储*/	//以上完成for(i=0;i<31;i++)语句
	……省略其它指令
完成一次64位字模乘算法需14条指令,共执行78个时钟周期	完成一次32位字模乘算法需36条指令,共执行约1059个时钟周期

表2的ARM7汇编程序用ARM编译器编译c语言程序得出.设ET为模乘执行时间,m为模乘调用的字模乘次数,n为一次字模乘所需时钟周期数,T为时钟周期,则模乘执行时间的计算方法如式(1)所示.

$$ET=m\times n\times T \tag{1}$$

基于式(1),可以根据表2所列参数对两种处理器进行性能比较(假设ARM7的时钟周期为t(100MHz),本文ASIP的时钟周期为2/3t(150MHz)).由式(2)可见,本文ASIP处理模乘算法的性能明显高于ARM7TDMI.

$$\frac{ASIP}{GPP}\geqslant\frac{32\times32\times1059\times t}{16\times16\times78\times2/3\times t}\cong81 \tag{2}$$

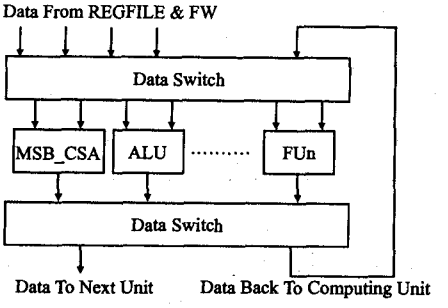

图5 指令集扩展方法

Fig. 5 Method for instruction set extension

本文预留了充裕的编码空间用于扩展指令集,通过扩展指令集可以使本文的ASIP支持更多的密码学算法.扩展方法如下,首先通过抽取新算法的核心运算步骤,将其映射为特殊运算单元FU,并与ALU共享输入数据与输出通路;然后在指令集中扩展出能够调用FU的新指令.基于扩展后的指令集,通过软件编程就可以高效实现新的算法.这种软硬件协同的扩展方式充分发挥了硬件速度快和软件可配置性强的优点,同时对处理器的结构改动很少,能够有效地降低设计成本.

5 ASIP的FPGA验证与VLSI实现

本文的专用指令集安全处理器(以下简称Security ASIP)采用Altera中stratixII系列的EP2S60F1020C3ES开发板进行FPGA硬件验证,并采用如图6所示的测试方案.

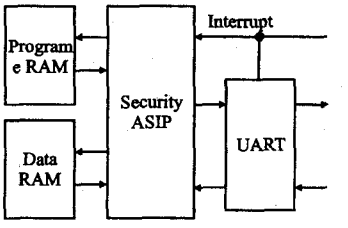

图6 Security ASIP的FPGA验证方案

Fig. 6 Verification by FPGA

Programe RAM和Data RAM分别作为Security ASIP的指令存储器和数据存储器,UART(通用异步串行接口)作为PC和Security ASIP通信的接口.将Security ASIP、Programe RAM、Data RAM和UART的RTL软核用QuartusII软件综合,生成的网表下载到FGPA开发板上.验证结果表明,Security ASIP能够准确完成模乘、模逆、RSA以及ECC算法等

功能.

表 3 Security ASIP 与 RSA 协处理器的性能比较

Table 3 Comparision with RSA coprocessors

作者	Liu[6]	Sun[7]	Ours
密钥长度(bits)	1024	2048	不限
工艺(μm)	0.18	0.35	0.25
频率(MHZ)	450	60	150
规模(gates)	148K	23K	28K
吞吐率(kbps)	214	18	5
可伸缩性	否	是	是
可编程性	否	否	是
可扩展性	难	难	易

Security ASIP 基于 TSMC 0.25μm 标准 CMOS 工艺进行了逻辑综合,下面是一些主要的性能参数. Security ASIP 核心电路的规模为28Kgates,最高时钟频率150MHz,数据

表 4 Security ASIP 与其它安全处理器的性能比较

Table 4 Comparision with security processors

Company	Product	1024-bit RSA
ARM Limited	Secure Core SC200[4]	594ms
MIPS Technologies,Inc	SmartMIPS 4KSc[8][9]	320ms
NEC Electronics,INC	V-WAY32 uPD79215000[10]	436ms
STMicroelectronics	SmartJ ST22XJ64[11]	380ms
Ours	Security ASIP	200ms

吞吐率为 5Kbps. 表 3,4 分别为 Security ASIP 与近年来的一些 RSA 协处理器[6,7]和其它安全处理器[3]的性能比较. 与协处理器相比,Security ASIP 具有更强的灵活性和更低的成本,与其他安全处理相比,Security ASIP 具有更快的速度和更高的时钟频率.

6 结 论

本文基于 RISC 架构处理器,提出了一种高性能的专用指令集安全处理器设计与实现方案. 为了高效地实现密码算法,本文设计了专用的指令集和特殊的运算单元. 与通用处理器相比,本文的设计具有更高的运算性能和更低的软硬件成本;与 ASIC 协处理器相比,则具有可配置和可编程的优点,灵活性和使用效率更高. 基于 ASIP 的软硬件协同设计是目前嵌入式系统设计的一个重要发展方向,其设计方法学也是一项重要的研究内容. 本文提出的软硬件协同扩展方法对于完善 ASIP 的设计流程具有一定的借鉴意义.

References:

[1] Rivest R L, Shamir A, Adleman L A. Method for obtaining digital signatures and public-key cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126.

[2] Montgomery P L. Modular multiplication without trial division [J]. Mathematics of Computation, 1985, 44(170): 519-521.

[3] Johann G, Guy-Armand K. Architectural enhance-ment for montgomery multiplication on embeded risc processors[J]. ACNS 2003,LCNS 2846, 2003, 418-434.

[4] ARM Limited. ARM SecurCore solutions[EB/OL]. Product brief, available for download at http://www. arm. com/aboutarm/4XAFLB/$File/SecurCores. pdf, 2002.

[5] Han Jun, Zeng Xiao-yang, Lu Rong-hua, et al. Attack-resisted coprocessor intergrating multiplicationand inverse[J]. Journal of Chinese Computer Systems, 2007,28(4):753-758.

[6] Liu Qiang, Ma Fang-zhen, Tong Dong-tong, et al. A regular parallel RSA processor[C]. In: Proceeding of the 2004 47th Midwest Symposium on Circuits and Systems (MWSCAS'04), Hiroshima, Japan, 2004, 467-470.

[7] Sun Yi-le, Wu Xing-jun. An area efficient modular arithmetic processor[C]. In: Proceeding of the 5th International Conference on ASIC, Beijing China, 2003, 1273-1276.

[8] MIPS Technologies, Inc. SmartMIPS architecture smart card extensions[EB/OL]. Product brief, available for download at http://www. mips. com/ProductCatalog/P SmartMIPSASE/SmartMIPS. pdf, 2001.

[9] MIPS Technologies, Inc. Making smart cards secure[EB/OL]. The Pipeline (Technology Newsletter), Fall 2001, p. 4, Available for download at http://www. mips. com/content/PressRoom/Newsletter, 2001.

[10] NEC Electronics, Inc. V-WAY32 32-bit security cryptocontroller[EB/Ol]. Product letter, available for download at http://www. nec. com. sg/es/Smartcard. htm, 2000.

[11] STMicroelectronics. ST22 smartJ platform smartcard ICs[EB/Ol]. Product brief, available for download at http://www. st. com/stonline/prodpres/smarcard/insc9901. htm, 2002.

[12] Bajot Y, Mehrez H. A macro-block based methodology for ASIP core design[C]. In: Proceedings of the ICSPAT, Orlando,1999.

[13] Gutub A A A, Tenca A F. Efficient scalable hardware architecture for montgomery inverse computation in GF(P)[C]. In: Proceeding of IEEE Workshop on Signal Processing Systems (SIPS), Seoul, Korea, 2003, 93-98.

附中文参考文献:

[5] 韩 军,曾晓洋,陆荣华,等. 集成模乘求逆双重运算的抗攻击 RSA 协处理器[J]. 小型微型计算机系统, 2007,28(4):753-758.