一款 RSA 模乘幂运算器的设计与实现

刘 强[1,2], 佟 冬[1,2], 程 旭[1,2]
(1. 北京大学微处理器研究开发中心, 北京 100871; 2. 北京大学计算机科学技术系, 北京 100871)

摘 要: 通讯技术的高速发展需要更高性能的密码处理设备. 本文介绍的 RSA 模乘幂运算器, 采用蒙哥马利模乘法算法和指数的从右到左的二进制方法, 并根据大整数模乘法运算和 VLSI 实现的要求进行改进, 提供高速 RSA 模乘幂运算能力. 该 RSA 运算器在其模乘法器中使用了进位保留加法器结构以避免长进位链. 我们提出了信号多重备份的方法, 解决大整数运算结构中关键信号广播带来的负载问题.

关键词: 蒙哥马利算法; 模乘法; 模乘幂; RSA; 公开密钥密码系统
中图分类号: TP309.7; TN47 文献标识码: A 文章编号: 0372-2112 (2005) 05-0923-05

The Design and Implementation of a RSA Modular Exponentiator

LIU Qiang[1,2], TONG Dong[1,2], CHENG Xu[1,2]
(1. *Microprocessor Research and Development Center, Beijing* 100871, *China;*
2. *Department of Computer Science and Technology; Peking University, Beijing* 100871, *China*)

Abstract: The rapid advance in communication technology brings a request for cryptoprocessors of higher performance. In the design of the RSA modular exponentiator, the Montgomery modular multiplication algorithm and the right-to-left binary method are used and modified considering large-bit modular multiplication and VLSI implementation. A Carry Save Adder structure is used in the modular multiplier, to avoid the long carry propagation. We propose a Signal Multi-Backup strategy to resolve the problem of large loads that are caused by the signal broadcasting of large-bit operation structures.

Key words: montgomery algorithm; modular multiplication; modular exponentiation; RSA; public-key cryptography

1 引言

在过去的几十年里, 电子通讯技术飞速发展, 国际互联网日益普及, 数据安全一直是保密通讯、电子商务、电子政务等领域密切关注的问题. 为了提供安全服务, 公开密钥密码系统[1] 被广泛应用于各类信息安全系统, 1978 年发明的 RSA 密码系统[2], 是目前应用最广泛的一种. RSA 密码系统的算法并不复杂, 但它依赖于长整数的模乘幂(Modular exponentiation)运算, 很难获得比较高的数据处理速度, 开发一种高速、实时并且成本不高 RSA 密码系统, 仍然是一个挑战.

本文介绍一款优化的 1024 位 RSA 模乘幂运算器(Modular exponentiator) 的设计与实现. 为了提高处理性能, 该 RSA 运算器在不同的层次上进行优化, 包括算法、结构和 VLSI 实现.

模乘幂运算由一系列的模乘法运算完成[3], 在实现长整数的模乘法运算的所有算法中, 蒙哥马利算法[4] 不依赖于长整数的比较和除法, 是一种便于硬件实现的算法, 所以应用最为广泛. 我们将采用文献[5] 中的简化技术, 通过略微增加循环的数量来去掉模乘法中的模化简步骤. 在文献[6, 7] 中, 作者描述了对操作数进行移位以简化计算的方法, 本文将讨论如何对其他操作数进行移位以简化商数的计算, 从而缩短硬件实现中流水线的关键路径, 提高流水线的速度.

为了避免长整数加法中的进位链问题, 常常采用的策略有两种, 一是采用脉动阵列结构[8], 二是采用进位保留加法器(CSA: Carry Save Adder) 结构[9]. 我们采用的是进位保留加法器结构, 在一个时钟周期内执行算法中的一个循环步, 关键路径的长度大约是两个单位全加器的延迟. 模乘法中最后的加法利用 32 位加法器完成. 文[9] 中所采用的移位寄存器结构, 对性能造成很大的影响, 我们采取了按地址索引访问的方式代替移位寄存器结构.

在 1024 位 RSA 模乘幂运算器的硬件实现中, 其数据运算结构非常长, 达到一千位以上, 而若干关键信号需要广播到这种长结构的所有数据位上去. 这种关键信号的广播, 负载非常大, 严重的影响着运算速度的提高. 文[7] 讨论了如何通过改进电路的设计来减小负载, 本文详细探讨了采用信号多重备份来降低负载的方法.

本文展示了 RSA 模乘幂运算器所采用的算法、体系结构和 VLSI 优化技术, 描述了它们如何结合起来, 取得高的运算性能. 本文的结构如下: 第一节是引言, 第二节描述蒙哥马利

收稿日期: 2003-12-08; 修回日期: 2004-11-12

模乘法算法和模乘幂算法,以及算法如何改进,以简化计算和硬件设计,第三节描述模乘法器和模乘幂运算器的设计结构.第四节描述VLSI实现方法,叙述如何解决全局信号的广播问题,并给出时序和面积结果.第五节根据功能性分析以及物理实现的时序和面积结果进行评估,并与现有的其他设计进行了比较,第六节是全文的总结.

2 蒙哥马利模乘法算法和模乘幂算法

2.1 蒙哥马利模乘法算法

下面是经过优化的蒙哥马利算法[4].其中,n 是模数 N 的位长,A 和 B 是小于 $2N$ 的数,a_i 是 A 的第 i 位.输出结果 $S=AB2^{-(n+2)} \bmod N$.A 的长度是 $n+1$,为了去掉最后的减法,增加了一次循环,为了去掉 B 的4倍,再增加两次循环,所以共有 $n+4$ 次循环.

```
Function MontPro(A, 4B, N) Begin
  S:= 0; q0:= 0;
  For i:= 0 to n+3 do Begin
    S:= (S+ qi× N+ ai× 4B) div 2;
    qi+1:= S mod 2;
  End;
  Return S;
End;
```

模乘幂运算连续利用模乘法运算,模乘法的输出直接用于下一次模乘法运算的输入,其输入输出的数值需要一定的范围限制.原始的蒙哥马利算法,需要根据结果是不是在一定的范围,决定是否进行一次模化简操作,即,一次比较运算和一次(可能的)减法运算,这增加了硬件设计的复杂性,并且处理时间不确定.在这里采用了文献[5]的简化技术,通过增加循环的数量(增加一次)来去掉模乘法中的模化简步骤,将算法的输入输出限定在 $2N$ 的范围之内.

我们要设计的关键运算模块,一个时钟周期执行蒙哥马利算法中一个循环步的计算.需要执行的操作有两个:一个是计算 N 的倍数 q_i,它依赖于一个加法的计算结果;另一个是两个倍数的加入以及一个移位.对于需要广播的到各个数位(1024位以上)的 q_i 信号,计算的独立是很必要的.相对于原始的蒙哥马利算法,将 B 上移1位,可以使得的 q_i 计算不依赖于 a_iB;将 B 上移2位,使得 S 的最低位不依赖于 a_iB,从而使得 q_{i+1} 不依赖于 a_iB.在文[6,7]中,作者描述了对操作数进行移位以简化计算的方法.我们在算法中采用将 B 上移2位的方法,其代价是增加了两次循环,但是商数的计算和商数的使用分别出现在两次循环中,并且商数的计算得到很大的简化,缩短了流水线的关键路径的延时,提高了流水线的速度,更适合于VLSI的实现.

2.2 模乘幂算法

在基于蒙哥马利模乘法算法的RSA密码系统中,执行模乘幂运算 $M^e \bmod N$ 需要两个额外的步骤.一个是映射,将输入数据 M 乘上一个因子 constC= $2^{2(n+2)} \bmod N$,另一个是对乘幂运算的结果执行反映射,去掉额外的因子,得到需要的结果.常数constC需要预先计算.

通过乘法计算乘幂 X^e 的最经济的方式,是一个有趣的数学问题,这方面的内容可以参考文献[3].本设计的目标是高速的RSA模乘幂运算,所以采用从右到左的二进制方法(又被称为平方乘法算法)[3].

```
Function MonExpr-RL(M, e, N) Begin
  P:= MontPro(constC, M, N);
  R:= MontPro(constC, 1, N); {Mapping}
  For i:= 0 to k- 1 do Begin {Exponentiation}
    if (ei= 1) then Begin
      R:= MontPro(R, P, N); {Multiplication}
    End;
    P:= MontPro(P, P, N); {Squaring}
  End
  R:= MontPro(1, R, N); {Re- Mapping}
  Return R;
End;
```

模平方和模乘法运算可以使用同样的模乘法电路.如果采用两个模乘法器并行运算,分别用于乘法操作和平方操作,计算只需要 $k+2$ 步,其中 k 是 e 的长度.这不仅提高了处理速度,而且乘幂运算的时间只依赖于指数的长度,不依赖于指数的具体数值.

3 结构设计

3.1 模乘法器

长的数据宽度,限制了许多原有的加法、乘法运算结构的使用,因为它们需要很大的面积.对于长整数的模乘法运算,乘法器/加法器结构中的长进位链是个难以解决的问题.一种策略是在乘法器中采用脉动阵列结构[8].另一种策略是中间结果运算采用CSA结构[9],输入输出采用普通表示,这就需要额外的时钟周期将中间结果的冗余表示转换为普通表示.我们在这里采用的是CSA结构进行模乘法运算,中间结果是进位保留形式,最后采用一个高速的32位的超前进位加法器对乘法结果进行格式转换.

模乘法器的结构中包括两个1027位的寄存器(Carry和Sum)用于寄存"进位"与"和",一对1027位CSA用于中间的加法运算,一个32位的超前进位加法器(CLA: Carry Lookahead Adder)用于最后的加法.模乘法器的结构如图1,其中灰色的矩形表示寄存器,白色的矩形表示组合逻辑,a_i,B 和 N 都是从外面输入的信号.

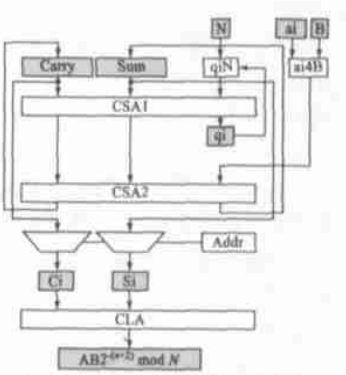

图1 基于CSA结构的蒙哥马利模乘法器

该模乘法器在一个时钟周期内执行算法中的一个循环步,中间结果即CSA2的两个输出存放在Carry和Sum寄存器

中. 模乘法的最后一个循环之后, 存放在 Carry 和 Sum 寄存器中的结果, 需要加起来得到模乘法的结果. 如果使用一般的加法器, 如行波加法器(Ripple Adder, 即串行进位加法器) 或 CLA, 执行 1024 位加法, 需要许多的硬件资源, 执行时间也很长. 我们使用一个 32 位的 CLA, 每个周期计算一次 32 位的加法, 输出 32 位的结果, 共用 32 个周期将 Carry 和 Sum 寄存器中的结果加起来. Carry 和 Sum 寄存器的输出采取按地址索引访问输出的方式, 即在做结果的加法运算时, 每个周期产生一个索引地址(Addr), 用这个地址访问 Carry 和 Sum 寄存器的 32 位. 从寄存器的输出到 CLA 计算完毕, 也是一条关键路径, 我们将它流水化, 首先将索引输出寄存起来, 存放在寄存器 *Ci* 和 *Si* 中, 然后再送到 CLA, 从而减小了关键路径的长度. CLA 每做完一次加法, 都会产生一个进位, 这个进位将用于下一次加法运算, 在图中没有画出相关的电路. 模乘法的输出直接用于下一次模乘法运算的输入, 其中间运算结果是 1027 位的, 输入输出是 1025 位的.

3.2　模乘幂运算器

模乘幂运算的执行经过三个阶段: 映射, 乘幂运算, 反映射. 在从右到左的二进制方法中, 两个模乘法器是并行工作的. 图 2 显示的是采用从右到左的二进制方法、包含两个蒙哥马利模乘法器(Multiplier 和 Square) 的模乘幂运算器.

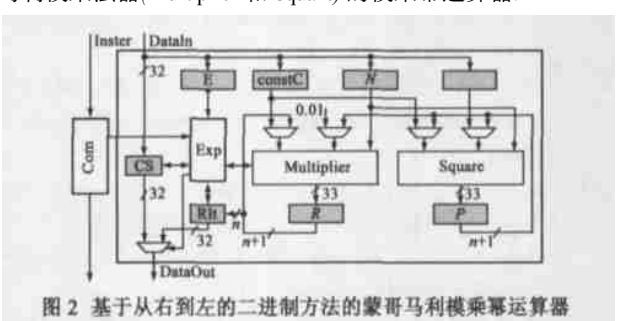

图 2　基于从右到左的二进制方法的蒙哥马利模乘幂运算器

在映射阶段, 初始值 *M* 和 1 被乘上映射因子 const*C*, 这由两个乘法器并行完成. 因此, 一次乘法的时间后, 映射就完成了. 在乘幂运算阶段, 并行计算开始, 乘法和平方相互独立. 模乘幂的临时结果保存在寄存器 *R* 和 *P* 中. 在反映射阶段, 计算最后一个蒙哥马利乘法, 从 *R* 中消除映射因子, 最后结果存放在寄存器 *R* 中, 在结果寄存器(*Rlt*) 可写时写入 *Rlt* 寄存器.

接口控制模块(Com) 负责外部数据通讯(DataIn 和 DataOut), 解释外部传来的控制命令(Instr), 乘幂运算控制模块(Exp) 负责控制模乘幂运算, 图中的 *CS* 是状态控制寄存器.

4　VLSI 实现

在 1024 位 *RSA* 模乘幂运算器的硬件实现中, 其数据运算结构非常长, 达到一千位以上, 而若干关键信号需要广播到这种长结构的所有数据位上去. 这种关键信号的广播, 负载非常大, 严重的影响着运算速度的提高. 例如, 信号 q_i 和 a_i 分别控制着加法器中 q_iN 和 a_i4B 的计算, 需要广播到一千多位长的进位保留加法器 CSA 的每一个数位上, 一个数据位要控制一千多个数据位, 负载带来的信号衰减(Decay), 要求每四个扇出(Fanout) 就要加一个放大门[7]. 对于 1024 位的模乘幂运算, 广播关键控制信号的树形结构需要五级门电路. 同样的, 寄存器 Carry 和 Sum 的控制比较复杂, 有同步复位、写使能等控制信号, 这些控制信号的负载很大(1024 以上). 针对关键信号的广播带来的问题, 我们在 VLSI 实现中采取了一种“信号多重备份” (SMB: Signal Multi Backup) 的方法:

首先, 关键信号被分发(备份) 到一组“备份寄存器”, 然后, 这些“备份寄存器” 的每一位都控制长结构中的一部分, 这样就可以显著的降低它们的负载. 比如在模乘法器的设计中, q_i 和 a_i 等关键信号都是首先备份到 16 个备份寄存器, 然后这 16 个备份寄存器再分别控制 64 个以上的数据位, 从而将关键信号的扇出从 1024 以上降低到了 64 左右. 我们知道, EDA 工具可以在关键控制信号中插入信号放大电路, 如缓冲器, 但是, 这会使得信号的放大和信号的使用处于同一个时钟周期, 从而产生长的路径. “信号多重备份” 方法的核心思想, 是将信号放大的部分工作转移到前一个周期进行, 从而减小了关键路径的长度.

其次, 如果能够将关键信号的计算提前, 在这些信号被使用之前准备完毕, 可以更方便的采用“信号多重备份” 的方法. 比如在蒙哥马利算法(MontPro) 中, *B* 被上移两位, 使得信号 q_i 被提前一个周期计算出来. 其它关键控制信号的计算也是这样处理的.

RSA 模乘幂运算器使用 VHDL 设计, 在明导(Mentor Graphics) 公司的 ModelSim 环境下验证, 用 Synopsys 公司的 Design Compiler 进行逻辑综合, 采用 Magma 公司的 Blast Fusion 工具进行物理实现. 逻辑综合采用了 TSMC 的 0.25、0.18、0.13μm CMOS 标准单元工艺, 物理实现采用 TSMC 的 0.13μm CMOS 标准单元工艺. 为了验证本文提出的算法和结构, 我们用 VHDL 重新设计了文献[9] 提出的结构, 并在不同工艺下进行逻辑综合和物理实现, 将实验结果与新的设计进行比较. 表 1 列出了两种设计(新的设计 SMB 和文献[9] 中的设计 Original) 在 0.25、0.18、0.13μm 工艺下的逻辑综合结果和 0.13μm 工艺下的物理实现结果(0.13ph), 包括关键路径长度(ns) 和总的单元面积(μm^2). 从 0.13μm 工艺下的物理实现结果可以看出, 基于 SMB 的设计与原有的设计相比, 时钟频率提高 106%, 而面积仅增加 27.8%.

表 1　新旧设计的关键路径(ns) 和面积(μm^2)

	0.25	0.18	0.13	0.13ph
Original	7.1 3140532	4.2 1934813	4 871659	3.7 787933
SMB	2.8 3301292	2.1 1780169	1.8 1033596	1.8 1006799

5　性能分析和比较

执行一次模乘法运算需要(*n*+ 36) 个时钟周期, 执行一次模乘幂运算需要(*n*+ 36)(*n*+ 2) 个时钟周期, 在 *n* 为 1024 时, 约是 1.09M 个时钟周期.

在进行 RSA 私钥操作时, 由于知道如何将 N 进行因子分解, $N = PQ$, 利用中国剩余定理(CRT: Chinese Remainder Theorem)[10,11], 可以降低模数和指数的长度, 从而提高 RSA 私钥操作的运算速度[12]. RSA 密码系统的性能, 主要取决于长整数模算术的高效实现以及私钥操作时利用 CRT 的能力. 基于 CRT 的 RSA 解密操作, 其计算复杂度(即时钟周期数)高度依赖于 P 和 Q 的长度(见文献[8]的 Table2 和 Fig. 6). 当 P 和 Q 的长度等于 N 的长度的一半时, 时钟周期数最小. 在最坏情况下, P 或 Q 的最大长度假定为 $2n/3$, 最小长度为 $n/3$. 采用 CRT, 一个模乘法器在软件的调度下完成一次 1024 位的 RSA 模乘幂运算, 需要的周期数, 最好情况下是 $0.56M$, 最坏情况下是 $0.62M$.

0.13μm 工艺下的物理实现结果(表 1)显示, 关键路径的延迟是 1.8ns, 主频可以达到 556MHz. 在主频为 556MHz 时, 完成一次 1024 位模乘幂运算的时间是 1.96ms. 1024 位私钥操作的数据吞吐率是 511Kb/s, 采用 CRT 时是 986~892Kb/s; 典型的公钥操作(17 位指数)的数据吞吐率是 27.6Mb/s.

将我们的设计与近年来发表的比较有代表性的相关研究成果进行对比, 结果如表 2 所示.

从表 2 可以看出, 进位保留结构的设计, 比脉动阵列结构的设计[8,13], 需要的时钟周期要少. 相对于文[9]的设计, 我们的设计在算法、结构和 VLSI 实现方面采取了各项优化措施, 主频得到很大的提高, 运算速度提高一倍多, 而面积仅增加 27.8%. 另外, 从文[13,14]中可以看出, 硬件上支持 CRT, 可以将私钥操作的速度提高 4 倍左右, 仅从软件上支持则速度的提高在 2 倍以下.

表 2 不同设计的性能比较

设计	年份	工艺 (μm)	芯片规模 (Kilo Gates)	主频 (MHz)	位长	时钟周期数 (非 CRT)	波特率 (Kb/s) 非 CRT	采用 CRT
[13]	1999	FPGA	–	52	1024	$2(n+4)(n+2)$	25	98
[8]	2001	0.6	109	150	512	$2(n+4)(n+2)$	141	578~328
[9]	2001	0.5	156	50	1024	$(n+34)(n+2)$	45	–
		0.13	155	270	1024	$(n+34)(n+2)$	249	–
[14]	2001	FPGA	–	46	1024	$2(n+4)(n/4+10)$	84	323
[15]	2002	FPGA	–	50	1024	$(n+3)(n+1)$	46	–
[16]	2003	FPGA	–	–	512	$(n/2+3)2(CRT)$	–	91
[17]	2003	FPGA	–	100	1024	$(n/2+2)(n/2+3)(CRT)$	–	389
		0.18	187	200	1024	$(n/2+2)(n/2+3)(CRT)$	–	778
[18]	2004	FPGA	–	78	1024	$2(n+4)(n+2)$	36	–
	2004	0.13	198	556	1024	$(n+36)(n+2)$	511	986~892

6 结论

本文描述了一款 RSA 模乘幂运算器的设计, 将我们的设计与近年发表的比较有代表性的相关研究成果进行比较, 其运算性能不仅优于同类结构的设计, 而且优于在硬件上直接采用 CRT 的设计, 这一是由于物理工艺水平和集成电路设计技术的提高, 也是由于算法、结构和 VLSI 实现技术上的进一步改善.

致谢: 感谢中国科学院数学机械化重点实验室助理研究员马玉杰博士在相关算法方面的建议.

参考文献:

[1] W Diffie, M E Hellman. New directions in cryptography[J]. IEEE Transactions on Information Theory, 1976, 22(6): 644–654.

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

[3] D E Knuth. The Art of Computer Programming Volume 2: Seminumerical Algorithms (Third Edition)[M]. 北京: 清华大学出版社, 2002: 461–481.

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

[5] C D Walter. Montgomery exponentiation needs no final subtractions[J]. Electronics Letters, 1999, 35(21): 1831–1832.

[6] E F Brickell. A fast modular multiplication algorithm with application to two-key cryptography[A]. Chaum et al. Advances in Cryptology-CRYPTO'82[C]. New York: Plenum, 1983. 51–60.

[7] S E Eldridge, C D Walter. Hardware implementation of Montgomery's modular multiplication algorithm[J]. IEEE Transactions on Computers, 1993, 42(6): 693–699.

[8] C-H Wu, J-H Hong, C-W Wu. RSA cryptosystem design based on the Chinese remainder theorem[A]. Proceedings of Asia South Pacific Design Automation Conference 2001 (ASP-DAC 2001)[C]. New York: IEEE press, 2001. 391–395.

[9] T-W Kwon, et al. Two implementation methods of a 1024 bit RSA cryptoprocesor based on modified Montgomery algorithm[A]. Proceedings of the 2001 IEEE International Symposium on Circuits and Systems (ISCAS 2001)[C]. New York: IEEE press, 2001, 4: 650–653.

[10] A Kondracki. The Chinese remainder theorem[J]. Formalized Mathematics, 1997, 6(4): 573–577.

[11] 潘承洞, 潘承彪. 初等数论[M]. 北京: 北京大学出版社, 1992. 164–174.

[12] M Shand, J Vuillemin. Fast implementations of RSA cryptography[A]. Proceedings of 11th IEEE Symposium on Computer Arithmetic[C]. New York: IEEE press, 1993. 252–259.

[13] T Blum, C Paar. Montgomery modular exponentiation on reconfigurable hardware[A]. Proceedings of 14th IEEE Symposium on Computer Arithmetic (ARITH-14)[C]. New York: IEEE press, 1999. 70–77.

[14] T Blum, C Paar. High-Radix Montgomery modular exponentiation on reconfigurable hardware[J]. IEEE Transactions on Computers, 2001, 50(7): 579–764.

[15] A Daly, W Marnane. Efficient architectures for implementing Montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic[A]. Proceedings of Tenth ACM International Symposium on Field-Programmable Gate Arrays (FPGA'02)[C]. New York: IEEE press, 2002. 44–49.

[16] C McIvor, M McLoone, J V McCanny. A high-speed, low latency RSA decryption silicon core[A]. Proceedings of 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003)[C]. New York: IEEE press, 2003, 4: 133- 136.

[17] C McIvor, M McLoone, et al. Marnane. Fast Montgomery modular multiplication and RSA cryptographic processor architectures[A]. Proceedings of 37th Asilomar Conference on Signals, Systems, and Computers[C]. New York: IEEE press, 2003, 1: 379- 384.

[18] Cilardo, A Mazzeo, et al. Carry-save Montgomery modular exponentiation on reconfigurable hardware[A]. Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'04)[C]. New York: IEEE press, 2004, 3: 206- 211.

作者简介:

刘　强　男, 1978 年生, 博士研究生, 2000 年毕业于北京大学计算机科学技术系, 同年成为计算机系统结构教研室硕士研究生, 2002 年转为北京大学微处理器研究开发中心博士生, 作为主要技术人员参与完成多项国家级有关处理器核心技术的研究课题, 主要研究方向为高性能微处理器、VLSI 设计与验证、安全芯片、系统芯片、计算机运算、浮点处理. E-mail: LiuQiang@mprc.pku.edu.cn.

佟　冬　1971 年生, 博士, 北京大学微处理器研究开发中心副教授, 主要研究方向为计算机系统结构、可重构计算、互联网、存储系统、系统芯片设计.

程　旭　1967 年生, 博士, 教授, 博士生导师, 北京大学微处理器研究开发中心主任、计算机科学技术系主任、计算机系统结构研究所所长、国家"十五"八六三计划超大规模集成电路设计专项专家组成员、中国通信学会专用集成电路委员会副主任委员, 主要研究方向为高性能微处理器、系统芯片、嵌入式系统、指令级并行、优化编译、软硬件协同设计等.