大整数乘法器的FPGA设计与实现

谢 星[①②] 黄新明[①] 孙 玲*[①] 韩赛飞[①]

[①](南通大学电子信息学院 南通 226019)

[②](南通大学工程训练中心 南通 226019)

摘 要：大整数乘法是公钥加密中最为核心的计算环节，实现运算快速的大数乘法单元是RSA, ElGamal，全同态等密码体制中急需解决的问题之一。针对全同态加密(FHE)应用需求，该文提出一种基于Schönhage-Strassen算法(SSA)的768 kbit大整数乘法器硬件架构。采用并行架构实现了其关键模块64k点有限域快速数论变换(NTT)的运算，并主要采用加法和移位操作以保证并行处理的最大化，有效提高了处理速度。该大整数乘法器在Stratix-V FPGA上进行了硬件验证，通过与CPU上使用数论库(NTL)和GMP库实现的大整数乘法运算结果对比，验证了该文设计方法的正确性和有效性。实验结果表明，该方法实现的大整数乘法器运算时间比CPU平台上的运算大约有8倍的加速。

关键词：全同态加密；现场可编程门阵列；大数乘法；GMP库

中图分类号：TN918.91; TN492 **文献标识码**：A **文章编号**：1009-5896(2019)08-1855-06

DOI: 10.11999/JEIT180836

FPGA Design and Implementation of Large Integer Multiplier

XIE Xing[①②] HUANG Xinming[①] SUN Ling[①] HAN Saifei[①]

[①](*School of Electronic Information*, *Nantong University*, *Nantong* 226019, *China*)

[②](*Engineering Training Center*, *Nantong University*, *Nantong* 226019, *China*)

Abstract: Large integer multiplication is the most important part in public key encryption, which often consumes most of the computing time in RSA, ElGamal, Fully Homomorphic Encryption (FHE) and other cryptosystems. Based on Schönhage-Strassen Algorithm (SSA), a design of high-speed 768 kbit multiplier is proposed. As the key component, an 64k-point Number Theoretical Transform (NTT) is optimized by adopting parallel architecture, in which only addition and shift operations are employed and thus the processing speed is improved effectively. The large integer multiplier design is validated on Stratix-V FPGA. By comparing its results with CPU using Number Theory Library(NTL) and GMP library, the correctness of this design is proved. The results also show that the FPGA implementation is about eight times faster than the same algorithm executed on the CPU.

Key words: Fully Homomorphic Encryption (FHE); FPGA; Large number multiplication; GMP library

1 引言

随着云计算时代的到来和云服务需求的不断增加，用户数据的安全性和隐私性成为人们关注的热点[1]。尽管云平台存放的是用户经过加密的数据，但同时被云服务商掌握的秘钥不能确保用户数据的安全和隐私。同态加密允许云服务器在密文上直接做任意运算，运算过程中数据始终处于加密状态，不会暴露任何原文信息，被认为是保障云时代用户数据安全的有效技术之一[2]。然而，以Gentry为代表的全同态加密(Fully Homomorphic Encryption, FHE)算法为保证同态计算安全，需要的密钥长度很长，从而导致同态加密方案运算复杂度极高，在现有微处理器上运行效率极低。例如，对于维度为2048的最低安全设置而言，1 bit原码加密后的大小约为785000 bit，每加密1 bit需要时间为1 s，并且随后对这些密文进行的操作都是大数运算，计算延时阻碍了FHE方案的实际应用。

大整数乘法是同态加密运算中的基本单元，几乎涉及到其中的每一步。围绕同态加密应用的需

收稿日期：2018-08-27；改回日期：2019-02-15；网络出版：2019-02-25

*通信作者： 孙玲 sun.l@ntu.edu.cn

基金项目：国家自然科学基金(61571246)，江苏省研究生科研与实践创新计划项目(KYCX17-1920)

Foundation Items: The National Natural Science Foundation of China (61571246), The Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX17-1920)

求，本文开展了大整数乘法单元的设计研究。本文首先讨论了全同态加密方案，在比较4种大整数乘法算法的基础上，选取了Schönhage-Strassen算法(SSA)；然后详细给出了SSA算法关键模块64K(1 K=1024 bit)点有限域快速数论变换(Number Theoretical Transform, NTT)单元设计方案，并提出了768 kbit大整数乘法器的硬件设计架构；最后进行了基于FPGA的仿真设计与硬件验证，具体介绍如下。

2 基于整数的全同态加密算法

全同态加密是指能够在密钥保密的情况下，对密文进行任意计算，即对于任意有效的运算f和m明文，有性质$f(\mathrm{Enc}(m))=\mathrm{Enc}(f(m))$[3]。文献[4,5]提出了第1个可信的全同态加密方案，该方案包括密钥生成、加密、解密和重加密(recryption)过程，其密钥可以离线生成。该方案的加密过程如式(1)所示，其中b是1 bit的明文数据，u_i为加密过程中随机生成的“噪声矢量”元素，加密过程是用公钥(d,r)对原文进行多项式求值运算从而得到密文c。解密过程主要完成一个模乘运算，如式(2)所示，其中w是私钥。在Gentry同态加密中，密钥一般为785000 bit，正是因为密钥如此之大，导致同态加密方案的复杂度非常高。因为所有加密、解密以及重加密运算都是基于785000 bit的模乘运算，所以快速的大整数模乘运算是实现该加密方案的关键之一。

$$c=[u(r)]_d=\left[b+2\sum_{i=1}^{n-1}u_i r^i\right]_d \tag{1}$$

$$m=[c\cdot w]_d \bmod 2 \tag{2}$$

3 Schönhage-Strassen大整数乘法算法分析

大整数乘法是目前在FHE方案里最耗时的操作，因此也是加速的主要对象。常用的乘法算法有小学算法(grammar-school)、Karatsuba-Ofman算法、Toom-Cook算法和Schönhage-Strassen算法(SSA)4种，表1给出了这4种算法的时间复杂度对比表[6]。从表中可以看出，小学算法的时间复杂度随乘数位数N呈平方律增加，SSA算法的时间复杂度则相对变化较小。因此，本文选择了Schönhage-Strassen算法作为核心算法来设计全同态加密中的大整数乘法运算。

Schönhage-Strassen算法描述的是一个基于快速数论变换(NTT)的乘法算法[7,8]，它提供了对大整数乘法进行有效并行计算的一个很好的解决方案，其主要实现步骤如下：

表 1 主要大整数乘法算法时间复杂度

算法	时间复杂度
grammar-school	$O(N^2)$
Karatsuba-Ofman	$O(N^{\ln 3/\ln 2})$
Toom-Cook	$O(N^{\ln(2k-1)/\ln(k)})$
Schönhage-Strassen (SSA)	$O(N\log N\log\log N)$

(1) 给定基数p，将两个大整数A与B分解成一系列的$a(i)$和$b(i)$；

(2) 分别计算分解后各序列的NTT，得到$\mathrm{NTT}(A(i))$和$\mathrm{NTT}(B(i))$；

(3) 将A和B序列的NTT计算结果按部分相乘得到$C(i)$，即：$C(i)=\mathrm{NTT}(A(i))\cdot\mathrm{NTT}(B(i))$；

(4) 计算$C(i)$的INTT变换得到$c(i)$，即：$c(i)=\mathrm{INTT}(C(i))$；

(5) 进行进位处理，当$c(i)\ge p$，则$c(i+1)=c(i+1)+(c(i)/p)$，并且$c(i)=c(i) \bmod p$。

SSA的大数算法需要计算操作数A和B的NTT，并且用NTT逆变换(INTT)运算NTT(A)和NTT(B)的点积结果。因此，大点数NTT运算模块是FHE整个架构设计的关键组成部分。本文选择将每个大数分成32k个组，每组有24 bit(24·32k=786432 bit)。两个数的乘法运算类似于环状卷积，每个数包含32k个样本。通常情况下，循环卷积涉及“0填充”，因此卷积结果包含输入信号大约两倍多的样本。为此，本文提出设计一个64k点有限域NTT模块，该模块也可以用于转换INTT结果，具体分析与设计过程如下。

4 有限域快速数论变换NTT

NTT与快速傅里叶变换(Fast Fourier Transforms, FFT)基本思想相同，只是FFT通过n次单位复根来进行运算，而NTT用旋转因子$r^{((p-1)/N)}(\bmod p)$来等价FFT运算中的$e^{(-2\pi i/N)}$，其中r是模素数p的原根(由于p是素数，则原根r一定存在)，即

$$e^{-\frac{2\pi i}{N}}\equiv r^{\frac{p-1}{N}}(\bmod p) \tag{3}$$

N点NTT的计算式为[9]

$$X_k=\sum_{n=0}^{N-1}x_n(r_N)^{nk}(\bmod p) \tag{4}$$

其中，$0\le k\le N-1$，r_N是第N个单位根。

N点NTT逆变换(INTT)的计算式为[9]

$$x_n=\frac{1}{N}\sum_{k=0}^{N-1}X_k(r_N)^{-nk}(\bmod p),0\le n\le N-1 \tag{5}$$

对于整数FHE算法，通常选择Solinas素数$p=2^{64}-2^{32}+1$[10]。该素数p支持高效的取模运算，如$2^{192} \bmod p=1, 2^{96} \bmod p=-1, 2^{64} \bmod p=2^{32}-1$。在进行SSA大整数乘法计算时，为了保证卷积计算结果不溢出，则要满足不等式$N/(2(b-1)^2)<p$。本文设计中同样选择了Solinas素数，并取参数N为65536，基数b为2^{24}。

SSA大整数乘法算法中需要计算64k点NTT，其中$r_{65536}=$ 0xE9653C8DEFA860A9，如果直接计算64k NTT将十分复杂，根据计算所得单位根，表2给出了部分NTT单位根示例。可见，64位、32位和16位等NTT的单位根都是2的幂次方，因此可以做移位操作，而不用直接乘法运算[11–13]，这样可以降低NTT运算的复杂性。因此，本设计选用32点和64点NTT来构建64k点NTT单元。

表 2　素数p的单位根r_N

NTT点数	单位根r_N
2	2^{96}
4	2^{48}
8	2^{24}
16	2^{12}
32	2^{6}
64	2^{3}

4.1　基-32 NTT设计与实现

在有限域Z/pZ中，硬件设计实现NTT所涉及到的模加、模减和模乘都是基于2的幂次方，如果要实现64位宽的操作，则每一个运算结果都要进行模p操作，占用了大量的硬件资源。本文将64位宽的操作数扩展为192位，采用“0填充”的方式进行数据位的扩展，这样就避免了每次操作都进行模p运算[14]，而且有$2^{192} \bmod p=1$。本设计选择 32点NTT作为基本单元，运算过程只需要移位和模加操作。2^6是32点NTT单位根，因此，32点有限域NTT和INTT公式分别如式(6)和式(7)所示，其中“%”为取模运算。

$$X(k)=\sum_{n=0}^{31} x(n)2^{6nk\%192} \bmod p \tag{6}$$

$$x(n)=\frac{1}{32}\sum_{k=0}^{31} X(k)2^{(192-6nk)\%192} \bmod p \tag{7}$$

分析式(6)与式(7)可以发现，每次参与运算的旋转因子都是固定值，而且都为2的幂次方。而当前主流硬件电路恰好非常适合计算2的幂次方运算，因为只需要进行移位就可以得出计算结果。移位处理单元运算结束后接着计算输出数据的累加和，该过程为32个数据相加，本文采用了进位保留加法器(Carry Save Adder, CSA)为基本单元以提高运算效率，CSA有3个位操作数输入端口，其中两个为加法数据a和b，还有一个为来自低位的进位c；输出端口ps为部分和输出，sc为进位输出。给定a, b, c 3个输入数据，其部分和为ps $=a\oplus b\oplus c$，进位为sc $=ab+ac+bc$。使用两个进位保留加法器串联可以实现一个4输入的CSA，在此基础上，设计得到如图1所示的32输入4级串联结构的树形大数求和单元。利用素数p的快速取模性质，图1中Reduction模块将前一级192 bit输出结果转换为64 bit输出，然后经过模加法运算得到求和结果。

图2给出了基-32 NTT运算结构示意图，该模块包括了32个移位单元和树形大数求和处理单元。每个时钟周期内，32组数据样本被送到移位单元进行移位和累加后输出32点NTT结果。

4.2　基-64 NTT设计与实现

同样，由于2^8是64点 NTT单位根，因此，64点NTT运算也只需要移位和模加操作。64点有限域NTT和INTT计算公式如式(8)和式(9)所示。

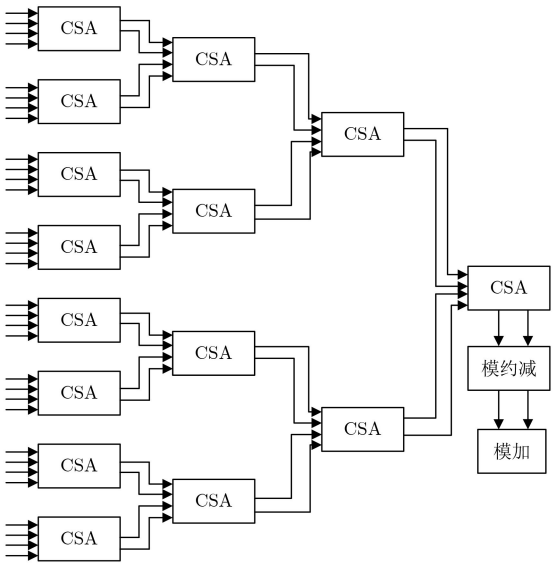

图 1 树形大数求和单元结构图

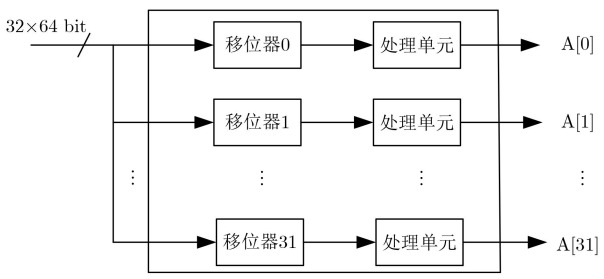

图 2 基-32 NTT运算结构图

$$X(k)=\sum_{n=0}^{63}x(n)2^{3nk\%192}\ \bmod p \tag{8}$$

$$x(n)=\frac{1}{64}\sum_{k=0}^{63}X(k)2^{(192-3nk)\%192}\ \bmod p \tag{9}$$

基-64 NTT在运算结构上和基-32 NTT类似，该模块包括了64个移位单元和树形大数求和单元。每个时钟周期内，64组数据被送到移位单元进行移位和累加后输出64点NTT结果。

4.3 64K点有限域NTT模块设计与实现

用64点和32点NTT构建 64K点 NTT单元的方法为：式(10)表示64k点NTT可以分解为64点与1024点NTT的形式，式(11)将1024点NTT分解为两级32点NTT运算，图3给出了使用2级基-32 NTT构建1024点NTT模块的串行架构。

$$X(k)=\sum_{n_1=0}^{63}W_{64}^{n_1k_1}\left(\sum_{n_2=0}^{1023}x(n)\,W_{1024}^{n_2k_2}\right)W_{64\times1024}^{n_1k_2} \tag{10}$$

$$X(k)=\sum_{n_1=0}^{31}W_{32}^{n_1k_1}\left(\sum_{n_2=0}^{31}x(n)\,W_{32}^{n_2k_2}\right)W_{32\times32}^{n_1k_2} \tag{11}$$

由于所设计的NTT架构在运算过程中需要大量的存储单元，本文采用了同址运算和无冲突算法[15,16]。设置输入数据的地址为$D=[d_{n-1},d_{n-2},\cdots,d_2,d_1,d_0]_r$，其中$n=\lceil\log_r N\rceil$；存储数据的地址为$A=[d_{n-1},d_{n-2},\cdots,d_2,d_1]_r$和$B=(d_{n-1}+d_{n-2}+\cdots+d_2+d_1+d_0)\bmod r$，其中$r$取为32，$N$为1024。首先对输入数据进行排序，排序好的数据存储到双口RAM中，通过读写地址控制单元对数据进行读写操作，读出的数据经过基-32 NTT模块进行运算，运算结果与ROM单元中存储的旋转因子进行模乘运算后再储存到RAM中。

64k点NTT硬件架构如图4所示。在第1级运算中，需要64个连续的1024点NTT模块进行运算。第1组1024个数据从RAM1s中读取出来，送入到1024点NTT运算模块运算，运算结果被写入到RAM1s

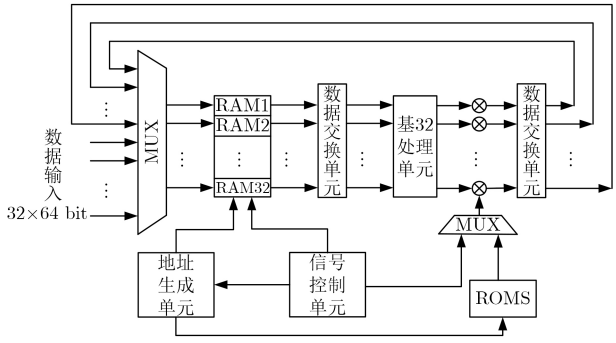

图 3 1024点NTT架构图

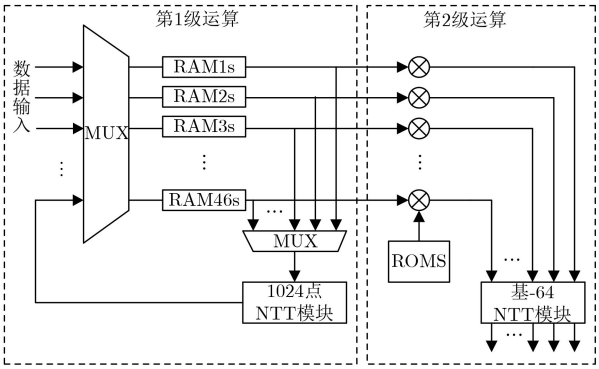

图 4 64k点硬件架构图

中；第2组数据从RAM2s中读取出来，同样送入到1024点NTT模块，运算结果写入到RAM2s中。以此类推，其他组数据进行同样的运算，直到64组1024点NTT都运算结束。

第2级运算，利用64点NTT运算单元进行计算。64个数据分别从64个存储模块中读取出来，每个存储模块RAMs中由32个RAM组成。读出数据与储存在ROM中的旋转因子进行模乘。模乘结果送入到64点NTT单元进行运算，最终输出64k点NTT运算结果。

5 大整数乘法器架构设计与实验结果分析

根据SSA大数算法，768 kbit大整数乘法器硬件架构如图5所示，该模块包括了两个64k点NTT运算单元，分别用来计算Da与Db的NTT变换。为了减少资源开销，将其中的一个NTT处理器单元用来计算NTT变换后的点积运算以及INTT变化，最终结果被送入到进位处理单元。乘法器的完整操作流程如下：

(1) 输入数据排序：对输入的数据Da和Db进行重新排序并存储到相应地址的存储器单元中；

(2) NTT计算：数据分别被送入到两组NTT处理单元进行计算，为了减少ROM的使用，两组NTT处理单元共用旋转因子存储单元；

(3) 点积运算：两个NTT变换后要进行数据的点积模乘运算，由于在设计64k点NTT单元时，已

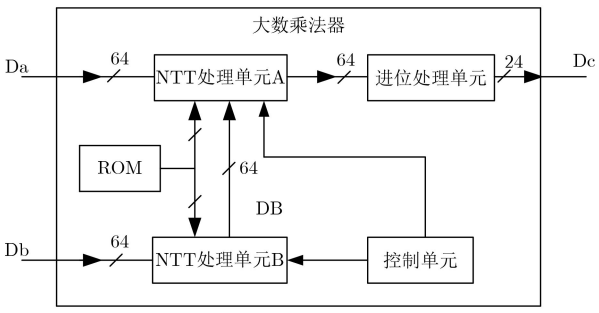

图 5 大整数乘法器架构图

经将64 bit模乘法器嵌入到64k点NTT单元中，因此点积运算时将其中一组NTT变换后的数据DB被送入到NTT单元A中进行点积模乘计算。在整个大数乘法器中，可以重复调用该模乘单元以减少了资源开销；

(4) NTT逆运算：完成两组数据的模乘运算后，计算结果在NTT单元A中进行逆变换；

(5) 进位处理：将经过逆变换后的数据进行处理输出最终结果。

为验证所提设计方法的正确性，本文在CPU平台和FPGA平台分别进行了768 kbit大整数乘法单元的仿真实验，其中，CPU平台为主频3.4 GHz的Intel®coreTMi7-7700处理器，内存64 GB，操作系统为Ubuntu16.04，使用GMP和NTL库实现该大整数乘法器；FPGA平台选用了Altera Stratix-V 5SGXEABN2F45I2，设计软件为Quartus II 13.0和Modelsim 10.4。实验结果表明，相同操作数在这两个平台下的运行结果一致，验证了本文提出的768 kbit大整数乘法器硬件实现方法正确。利用Altera synthesize tool，对该硬件实现方法进行综合，综合结果如表3所示。此时，FPGA硬件条件下电路的最高工作频率为98.02 MHz，大整数乘法运算需要41146个时钟周期，大整数乘法器运算768 k×768 kbit数据的速度是在CPU下运行的8倍左右，如表4所示。

表5给出了本文工作与部分已发表文献的结果比较，其中，文献[17]在设计SSA算法中采用“负卷积”，避免了“0填充”，从而有效减少了FPGA资源占用量，但本文计算速度比文献[17]提高了大约6倍。与文献[18]相比，本文占用FPGA资源少，该文献只给出了乘法器关键组件64k点有限域FFT的计算时间为0.125 ms，并没有给出大整数乘法器运算所需时间。

表 3　Stratix-V FGPA综合结果

逻辑单元	Stratix V		
	占用资源数	总资源数	利用率(%)
ALUTs	240229	718400	33
Logic registers	236088	1436800	16
Total block Memory (bit)	16252928	54067200	30
Total DSP blocks	288	352	82
最大频率(MHz)		98.02	

表 4　CPU和FPGA上性能比较

	计算时间(ms)	加速倍数
I7-7700 CPU	3.350	1
本文设计	0.419	8

表 5　实现结果对比

设计	位数 (kbit)	频率 (MHz)	计算时间 (ms)	占用资源数
文献[17]	768	181	2.30	7.6k ALUTs+3.4k regisiters
文献[18]	768	100	–	463k ALUTs+336k regisiters
本文	768	98.02	0.419	240k ALUTs+236k regisiters

6　结束语

本文设计并实现了基于SSA算法的768 kbit大整数乘法器硬件架构，利用32点与64点NTT单元实现了关键模块64k点NTT的运算；采用了同址运算和无冲突算法实现了存储单元的数据存取。在Altera的Stratix-V综合实现表明，本文大整数乘法器最高工作频率为98.02 MHz，运算速度是基于GMP运算库实现的8倍，但本文工作在乘法器结构上尚有进一步改进空间。

参 考 文 献

[1] 光炎, 祝跃飞, 顾纯祥, 等. 一种针对全同态加密体制的密钥恢复攻击[J]. 电子与信息学报, 2013, 35(12): 2999–3004. doi: 10.3724/SP.J.1146.2013.00300.
GUANG Yan, ZHU Yuefei, GU Chunxiang, *et al*. A key recovery attack on fully homomorphic encryption scheme[J]. *Journal of Electronics & Information Technology*, 2013, 35(12): 2999–3004. doi: 10.3724/SP.J.1146.2013.00300.

[2] FENG Xiang and LI Shuguo. Design of an area-effcient million-bit integer multiplier using double modulus NTT[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017, 25(9): 2658–2662. doi: 10.1109/TVLSI.2017.2691727.

[3] 陈智罡. 基于格的全同态加密研究与设计[D]. [博士论文], 南京航空航天大学, 2015: 1–5.
CHEN Zhigang. Research and design of fully homomorphic encryption based on lattice[D]. [Ph.D. dissertation], Nanjing University of Aeronautics and Astronautics, 2015: 1–5.

[4] GENTRY C and HALEVI S. Implementing Gentry's fully-homomorphic encryption scheme[C]. The 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology, Tallinn, Estonia, 2011: 129–148.

[5] GENTRY C. A fully homomorphic encryption scheme[D]. [Ph.D. dissertation], Stanford University, 2009.

[6] 施佺, 韩赛飞, 黄新明, 等. 面向全同态加密的有限域FFT算法FPGA设计[J]. 电子与信息学报, 2018, 40(1): 57–62. doi: 10.11999/JEIT170312.
SHI Quan, HAN Saifei, HUANG Xinming, *et al*. Design of finite field FFT for fully homomorphic encryption based on

FPGA[J]. *Journal of Electronics & Information Technology*, 2018, 40(1): 57–62. doi: 10.11999/JEIT170312.

[7] ÖZTÜRK E, DORÖZ Y, SAVAŞ E, *et al*. A custom accelerator for homomorphic encryption applications[J]. *IEEE Transactions on Computers*, 2017, 66(1): 3–16. doi: 10.1109/TC.2016.2574340.

[8] YE J H and SHIEH M D. Low-complexity VLSI design of large integer multipliers for fully homomorphic encryption[J]. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2018, 26(9): 1727–1736. doi: 10.1109/TVLSI.2018.2829539.

[9] POLLARD J M. The fast Fourier transform in a finite field[J]. *Mathematics of Computation*, 1971, 25(114): 365–374. doi: 10.1090/S0025-5718-1971-0301966-0.

[10] WANG Wei, HUANG Xinming, EMMART N, *et al*. VLSI design of a large-number multiplier for fully homomorphic encryption[J]. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2014, 22(9): 1879–1887. doi: 10.1109/TVLSI.2013.2281786.

[11] RAFFERTY C, O'NEILL M, and HANLEY N. Evaluation of large integer multiplication methods on hardware[J]. *IEEE Transactions on Computers*, 2017, 66(8): 1369–1382. doi: 10.1109/TC.2017.2677426.

[12] ROY S S, VERCAUTEREN F, VLIEGEN J, *et al*. Hardware assisted fully homomorphic function evaluation and encrypted search[J]. *IEEE Transactions on Computers*, 2017, 66(9): 1562–1572. doi: 10.1109/TC.2017.2686385.

[13] DORÖZ Y, ÖZTÜRK E, and SUNAR B. Accelerating fully homomorphic encryption in hardware[J]. *IEEE Transactions on Computers*, 2015, 64(6): 1509–1521. doi: 10.1109/TC.2014.2345388.

[14] WANG Wei, HU Yin, CHEN Lianmu, *et al*. Accelerating fully homomorphic encryption using GPU[C]. 2012 IEEE Conference on High Performance Extreme Computing, Waltham, USA, 2012: 1–5. doi: 10.1109/HPEC.2012.6408660.

[15] HUANG Xinming and WANG Wei. A novel and efficient design for an RSA cryptosystem with a very large key size[J]. *IEEE Transactions on Circuits and Systems II*: *Express Briefs*, 2015, 62(10): 972–976. doi: 10.1109/TCSII.2015.2458033.

[16] JOHNSON L G. Conflict free memory addressing for dedicated FFT hardware[J]. *IEEE Transactions on Circuits and Systems II*: *Analog and Digital Signal Processing*, 1992, 39(5): 312–316. doi: 10.1109/82.142032.

[17] FENG Xiang and LI Shuguo. Accelerating an FHE integer multiplier using negative wrapped convolution and Ping-Pong FFT[J]. *IEEE Transactions on Circuits and Systems II*: *Express Briefs*, 2019, 66(1): 121–125. doi: 10.1109/TCSII.2018.2840108.

[18] WANG Wei and HUANG Xinming. FPGA implementation of a large-number multiplier for fully homomorphic encryption[C]. Proceedings of 2013 IEEE International Symposium on Circuits and Systems, Beijing, China, 2013: 2589–2592. doi: 10.1109/ISCAS.2013.6572408.

谢 星：男，1985年生，博士生，研究方向为信息安全.

黄新明：男，1974年生，教授，研究方向为VLSI设计、高性能计算.

孙 玲：女，1976年生，教授，研究方向为专用集成电路设计、系统集成技术.