一种应用于BIKE的基于Karatsuba算法的大尺寸多项式乘法器

杨 柳[1],张永真[2],田 静[2],宋苏文[3],王中风[3]
(1. 南京大学电子科学与工程学院,江苏南京 210023;2. 南京大学集成电路学院,江苏苏州 215011;
3. 中山大学集成电路学院,广东深圳 518107)

摘 要: 当前美国国家标准与技术研究院(National Institute of Standards and Technology,NIST)对后量子密码(Post-Quantum Cryptography,PQC)标准化方案的评估已进入第四轮,位翻转密钥封装(Bit Flipping Key Encapsulation,BIKE)协议是目前被评估的四个候选方案之一. 在BIKE的密钥生成算法中,多项式乘法作为众多密码系统中特别耗时的操作之一,耗费了大量的时间和面积资源. 针对此问题,本文设计了一种基于Karatsuba算法(Karatsuba Algorithm,KA)的无交叠多项式乘法器,可高效实现万级比特位宽的多项式乘法,具有低时延、高性能和面积小的特点. 同时,本文将该优化乘法器应用于BIKE密钥生成算法中,并基于现场可编程门阵列(Field Programmable Gate Array,FPGA)对其进行硬件架构实现,改进了原有的紧凑多项式乘法和多项式求逆算法. 本文提出的乘法器通过采用不同的操作数位宽,可适应对面积和延时的不同需求. 与BIKE原本的设计相比,改进的设计使密钥生成模块的延时减小了36.54%,面积延迟积(Area Delay Production,ADP)减小了10.4%.

关键词: 后量子密码(PQC);多项式乘法器;Karatsuba算法(KA);位翻转密钥封装(BIKE)
基金项目: 国家自然科学基金(No.62104097)
中图分类号: TN47 **文献标识码:** A **文章编号:** 0372-2112(2025)01-0084-10
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20231210

A Large-Width Polynomial Multiplier Based on Karatsuba Algorithm for BIKE

YANG Liu[1], ZHANG Yong-zhen[2], TIAN Jing[2], SONG Su-wen[3], WANG Zhong-feng[3]
(1. *School of Electronic Science and Engineering*, *Nanjing University*, *Nanjing*, *Jiangsu* 210023, *China*;
2. *School of Integrated Circuits*, *Nanjing University*, *Suzhou*, *Jiangsu* 215011, *China*;
3. *School of Integrated Circuits*, *Sun Yat-Sen University*, *Shenzhen*, *Guangdong* 518107, *China*)

Abstract: The current evaluation of the post-quantum cryptography (PQC) standardization program by the National Institute of Standards and Technology (NIST) has entered the fourth round. Bit flipping key encapsulation (BIKE) is one of four candidates currently being evaluated. In the key generation of BIKE, the polynomial multiplication consumes a lot of time and area resources, which is also one of the slowest and most area consuming operations in most cryptography systems. In this work, we propose an overlap-free polynomial multiplier based on the Karatsuba algorithm (KA), which can efficiently implement polynomial multiplication of tens of thousands of bits with low latency, high performance and small area. This multiplier is applied to the BIKE key generation algorithm, which is implemented in hardware architecture based on the field programmable gate array (FPGA), improving the original compact polynomial multiplication and polynomial inversion algorithm. The multiplier proposed in this article can adapt to different requirements for area and delay by using different operand bit widths. Compared with BIKE's original design, the improved design reduces the delay of the key generation module by 36.54% and the area delay production (ADP) by 10.4%.

收稿日期:2023-12-25;修回日期:2024-08-26;责任编辑:孙瑶

Key words: post-quantum cryptography (PQC); polynomial multiplier; karatsuba algorithm (KA); bit flipping key encapsulation (BIKE)

Foundation Item(s): National Natural Science Foundation of China (No.62104097)

1 引言

在过去的三十年中,传统公钥密码已成为全球通信数字基础设施不可或缺的组成部分[1]. 经典公钥密码系统包括RSA(Rivest-Shamir-Adleman)[2]、椭圆曲线密码算法[3]等,在信息安全领域得到广泛的应用,以保证网络传输数据的机密性和完整性及对通信双方实体身份认证[4]. 然而,1994年SHOR[5]提出,量子算法可以解决大整数因子分解问题和离散对数问题. 一旦有相当规模的量子计算机成为现实,基于上述问题的公钥密码体制将不再安全[6]. 因此,研究人员开始探索在量子攻击下依旧安全的密码系统. 2016年,美国国家标准与技术研究院(National Institute of Standards and Technology,NIST)宣布了后量子密码标准化项目,旨在寻找合适的后量子密码方案并对其进行标准化,其中包括基于中密度准循环奇偶校验码(Quasi-Cyclic Moderate-Density Parity-Check codes,QC-MDPC)构建的位翻转密钥封装(Bit Flipping Key Encapsulation,BIKE)方案[7]. 目前,NIST公布了第四轮候选方案,BIKE是四个候选方案之一[8].

BIKE主要包括密钥生成、封装、解封装三个部分. 其中密钥生成阶段最耗时,多项式乘法是其中的核心计算之一,具有资源消耗大且延时长的特点. 2021年,文献[9]首先在可编程门阵列(Field Programmable Gate Array,FPGA)上实现了BIKE的整个算法,优化了密钥生成部分的多项式求逆计算算法,并且首次实现了BGF(Black-Gray-Flip)解码器,并实现了参数化设计和恒定时间设计,从而可以抵抗计时攻击.

虽然文献[9]完成了对BIKE的整体实现,但其密钥生成模块中的有限域多项式乘法操作仍较为复杂. BIKE中的有限域多项式乘法的操作数位宽达到10 000 bit以上,是密钥生成的重要步骤,具有面积大、延时长的特点. 如果提升了乘法操作的效率,那么可大大提高整个密钥生成模块的性能. 此外,这一挑战并非仅限于BIKE算法,对于其他加密算法而言,如何设计面积小且延时低的乘法器同样是一个复杂的问题. 因此,设计一种适用于大位宽的高效有限域多项式乘法器是必要的.

目前,针对乘法的改进方法也有诸多研究,如快速数论变换(Number Theoretic Transform,NTT)算法、Karatsuba算法(Karatsuba Algorithm,KA)[10]等,NTT算法应用范围较窄,常用于基于格的密码,如CRYSTALS-Kyber算法[11],而KA应用较广. 在KA基础上也有诸多应用和改进,如文献[12]中提出的基于8级分层KA的多项式乘法,可显著减小面积;文献[13]提出了一种优化的KA结构并将其应用于基于格的密码,以92.06%的额外硬件开销获得了2.09倍的速度.

基于KA的多项式乘法器通常采用分治算法,把大数乘法运算拆分成三个部分积的运算,每个部分积的运算按照同样的算法递归求取,直到将大数拆分成两个小位宽的基的乘法,并用传统的多项式乘法计算. 但这样的架构存在两个问题:一是现有的架构在每次拆分时都需要对输入项进行重排序,导致布局布线较复杂;二是对于较大位宽的多项式乘法,KA的乘法器需要较大的面积和延时.

针对这两个问题,本文提出了三个改进方法:重排序模块、混合串并架构和逐列计算法,提出了一种大尺寸多项式乘法器,可高效实现万级比特位宽的多项式乘法,具有低时延、高性能和面积小的特点. 本文将该乘法器在Xilinx xc7a200tffv1156-1 FPGA上实现,并基于该乘法器在FPGA上实现了BIKE协议中的密钥生成算法. 相比于基于传统乘法器的实现,本文的乘法器使密钥生成模块的延时减小了36.54%,面积延时积(Area Delay Production,ADP)减小10.4%,并且可继续增大乘法器的尺寸,实现该模块更小的延时.

2 基础知识

本节主要介绍了传统的Karatsuba乘法算法以及后续的改进算法,并介绍了基于QC-MDPC码的BIKE协议.

2.1 Karatsuba乘法算法

在加密算法中,乘法运算是消耗资源较多、延时较长的关键模块. 与传统的乘法相比,KA[10]采用额外的加法操作来减少乘法操作,从而可以有效地减少资源消耗. 例如,将多项式u_1X+u_0和v_1X+v_0采用KA进行相乘,可以得到

$$\begin{aligned}&(u_1X+u_0)(v_1X+v_0)\\&=u_1v_1X^2+(u_1v_0+u_0v_1)X+u_0v_0\\&=u_1v_1X^2+((u_1+u_0)(v_1+v_0)\\&\quad-u_1v_1-u_0v_0)X+u_0v_0\end{aligned}\tag{1}$$

对于高阶的KA,可以使用递归的方法来实现. 对于两个n项多项式:

$$U(x)=\sum_{i=0}^{n-1}u_i x^i$$
$$V(x)=\sum_{i=0}^{n-1}v_i x^i \tag{2}$$

其中，$n=2m$. 将两个多项式相乘，可将这两个多项式分别分为高位和低位：

$$U(x)=x^m\sum_{i=0}^{m-1}u_{m+i}x^i+\sum_{i=0}^{m-1}u_i x^i=U_H x^m+U_L$$
$$V(x)=x^m\sum_{i=0}^{m-1}v_{m+i}x^i+\sum_{i=0}^{m-1}v_i x^i=V_H x^m+V_L \tag{3}$$

其中，$U_H=\sum_{i=0}^{m-1}u_{m+i}x^i$，$V_H=\sum_{i=0}^{m-1}v_{m+i}x^i$，$U_L=\sum_{i=0}^{m-1}u_i x^i$，$V_L=\sum_{i=0}^{m-1}v_i x^i$ 分别为输入的两个多项式的高位 $n/2$ bit 和低位 $n/2$ bit. 再将这两个多项式利用KA进行相乘，可得

$$\begin{aligned}U(x)V(x)&=(U_H x^m+U_L)(V_H x^m+V_L)\\&=P_2(x)x^{2m}+[P_1(x)-P_2(x)-P_0(x)]x^m+P_0(x)\end{aligned} \tag{4}$$

其中，

$$P_2=U_H V_H \tag{5}$$
$$P_1=(U_H+U_L)(V_H+V_L) \tag{6}$$
$$P_0=U_L V_L \tag{7}$$

对于 P_2、P_1 和 P_0 将继续采用KA进行计算，如此递归多次，得到最终结果，递归次数的最优解可由实验给出. 这种算法即为递归的KA（Karatsuba-Ofman Algorithms，KOA）[14].

图1和图2分别是采用传统的多项式乘法（Conventional Algorithm，CA）和KOA来实现2 bit和4 bit的多项式乘法. 文献[15]中提出，对于 n bit的多项式乘法，如果用CA实现，需要的单比特加法器（XOR）和乘法器（AND）个数分别为

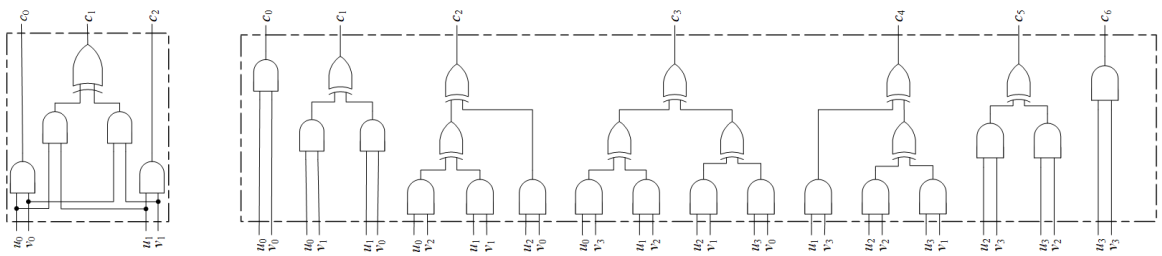
$$\mathrm{CA}_{\mathrm{XOR}}(n)=(n-1)^2 \tag{8}$$
$$\mathrm{CA}_{\mathrm{AND}}(n)=(n)^2 \tag{9}$$

其延时（T_{CA}）为

$$T_{\mathrm{CA}}(n)=T_a+\log_2(n)T_x \tag{10}$$

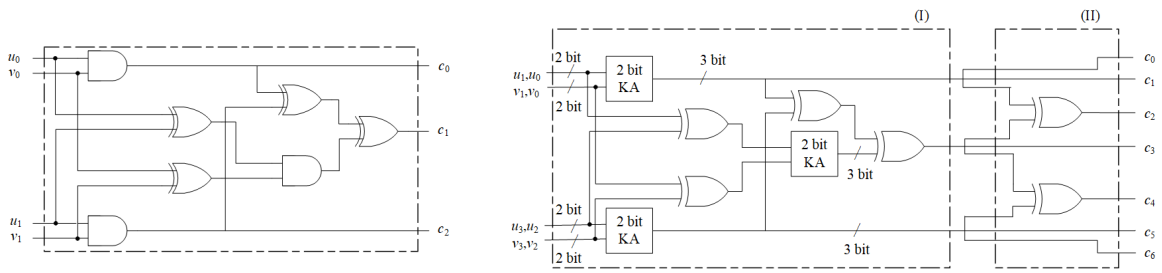
其中，T_a 和 T_x 分别为乘法器和加法器的延时. 如果用KA来实现，需要的加法器和乘法器个数及延时分别为

$$\mathrm{KA}_{\mathrm{XOR}}(n)=6n^{\log_2(3)}-8n+2 \tag{11}$$
$$\mathrm{KA}_{\mathrm{AND}}(n)=n^{\log_2(3)} \tag{12}$$
$$T_{\mathrm{KA}}(n)=T_a+(3\log_2(n)-1)T_x \tag{13}$$



(a) 2 bit (b) 4 bit

图1 2 bit和4 bit的CA



(a) 2 bit (b) 4 bit

图2 2 bit和4 bit的KOA多项式乘法

由此可得，KOA可以有效地减少多项式乘法器消耗的资源，尤其是当乘法的位宽较大且递归的次数多时，效果更为明显. 但是其延时仍然远大于CA. 为此，文献[16]中提出了一种去重叠的KA（Overlap-free Karatsuba Algorithm，OKA），并且在文献[15]中进行了基于FPGA的实现. 该算法将两个比特位宽的多项式按照奇数项和偶数项分类进行改写：

$$U(x)=\sum_{i=0}^{m-1}u_{2i}y^i+x\sum_{i=0}^{m-1}u_{2i+1}y^i=U_e(y)+xU_o(y)$$
$$V(x)=\sum_{i=0}^{m-1}v_{2i}y^i+x\sum_{i=0}^{m-1}v_{2i+1}y^i=V_e(y)+xV_o(y) \tag{14}$$

其中，$n=2m$；$y=x^2$. 相乘后可得

$$U(x)V(x)=(U_e(y)+xU_o(y))\times(V_e(y)+xV_o(y))$$
$$=G_2(y)y+[G_1(y)-G_2(y)-G_0(y)]x+G_0(y) \quad (15)$$

其中,

$$G_0=U_eV_e \quad (16)$$

$$G_1=(U_o+U_e)(V_o+V_e) \quad (17)$$

$$G_2=U_oV_o \quad (18)$$

对于G_0、G_1、G_2采用递归的OKA继续计算. 图3是采用OKA的8 bit多项式乘法. 由公式和图可知,与KA相比,每次递归OKA都可以减小延迟. 根据文献[15]的计算,对于比特多项式乘法,OKA消耗的资源和延时分别为

$$\mathrm{OKA}_{\mathrm{XOR}}(n)=6n^{\log_2(3)}-8n+2 \quad (19)$$

$$\mathrm{OKA}_{\mathrm{AND}}(n)=n^{\log_2(3)} \quad (20)$$

$$T_{\mathrm{OKA}}(n)=T_a+(2\log_2(n)-1)T_x \quad (21)$$

从而在传统KA的基础上进一步减小延时. 与CA相比,虽然OKA延时略大,但是由于消耗的资源较少,面效比较高. 尤其是对于较大位宽的乘法,OKA算法面效比优势更明显.

2.2　位翻转密钥封装(BIKE)协议

BIKE协议包括三个算法:密钥生成、封装、解封装. 由于密钥生成过程最耗时,因此本工作重点研究对密钥生成算法的改善. 该算法中的符号定义:$\mathbb{F}_2$为二进制有限域;$\mathcal{R}$为循环多项式环$\mathbb{F}_2[X]/(X^r-1)$;$|h|$为二进制多项式$h\in\mathcal{R}$的汉明重量. 各个参数的定义:r为长度,并且为素数;w为行的汉明重量,正偶数且$w/2$为奇数;l为共享密钥的尺寸,正整数;t为误码的汉明重量,正整数.

对于不同保密等级,各个参数的值如表1所示.

表1　BIKE协议不同保密等级对应的参数

安全等级	r	w	t	解码失败率
Level 1	12 323	142	134	2^{-128}
Level 3	24 659	206	199	2^{-192}
Level 5	40 973	274	264	2^{-256}

本文研究的是密钥生成算法. 算法1详细描述了根据BIKE的参数生成私钥和公钥的过程. 在该算法中,步骤3是消耗资源最多和延时最长的计算,包含了一个r项多项式乘法和一个r项多项式求逆,其中求逆也是由无数个多项式乘法实现的.

算法1　密钥生成

输入: BIKE参数n,ω,t,l

输出: 私钥(h_0,h_1,σ)和公钥h

1. 生成$(h_0,h_1)\xleftarrow{\$}\mathcal{R}^2$,其汉明重量$|h_0|=|h_1|=\omega/2$为奇数;
2. 随即均匀生成$\sigma\xleftarrow{\$}\{0,1\}^{\ell}$;
3. 计算$h\leftarrow h_1h_0^{-1}$;
4. 输出(h_0,h_1,σ)和h.

2.3　相关符号定义

本文的实验部分提到了一些名称和缩写,表2中进行了详细定义.

表2　相关符号及其定义

符号	定义
Delay	延时,即模块从开始到执行完毕所需要的时间
LUT	Look Up Table,查找表
Slice	FPGA的基本可配置逻辑单元,Slice内部包含4个LUT等资源
ADP	Area Delay Production,面积和延时的乘积,本文采用Slice的值作为面积值
Freq	模块运行的时钟频率

3　基于KA的高阶多项式乘法器

本节详细介绍了本文提出的基于KA的高阶多项式乘法器,包括重排序模块、混合串并架构和逐列计算法三个部分. 重排序模块可以在乘法器的输入端简化布局布线;混合串并架构可减小乘法器的面积,对于中等位宽(128 bit以上,10 000 bit以下)的乘法器效果明显;逐列计算法可大大减小乘法器的面积,使乘法器适用于超大位宽(10 000 bit以上)的乘法. 结合了逐列计算法的乘法器可适用于超大位宽多项式乘法,若乘法器采用混合串并架构,在资源消耗相当的情况下,能采用更高阶的KA优化,可降低其延时,从而降低ADP.

3.1　重排序

在递归使用KA前,首先利用基于二叉树模型深度优先的递归函数,对输入的大整数各项进行重排序. 重排序函数富有规律,可拓展性强. 假设输入的多项式项数为n,将多项式分为N个位宽为w的基t_i,其中$i=0,1,\cdots,N-1,n/w=N$,从而在顶层将基利用传统多项式乘法计算. 其重排序算法过程如算法2所示.

算法2　重排序算法

输入: 最大深度$\log_2 N,t_i$, 其中$i=0,1,\cdots,N-1$

输出: $\bar{t}_i$ with $i\in\{0,1,...,N-1\}$

初始化部分:

1. $k=0,\ \mathrm{depth}=0,\ \mathrm{index}=0$

迭代计算部分:

2. Perm(depth, index)
3. IF depth $<\log_2 N$ THEN
4. Perm (depth + 1, index)
5. Perm (depth + 1, index + 2^{depth})

迭代结束后顶层部分:

6. ELSE
7. $\bar{t}_k=t_{\mathrm{index}},\ k=k+1$
8. ENDIF
9. Return: $\bar{t}_i$ with $i\in\{0,1,\cdots,N-1\}$.

利用重排序模块，每次递归KA时均无需再次进行排序. 例如，对于多项式项数为8，基的位宽为2的多项式乘法，先通过重排序模块，将多项式

$$U(x)=\sum_{i=0}^{7}u_i x^i \tag{22}$$

重排序后可得：$\overline{t_0}=u_0,u_4$；$\overline{t_1}=u_2,u_6$；$\overline{t_2}=u_1,u_5$；$\overline{t_3}=u_3,u_7$.

同理，将多项式

$$V(x)=\sum_{i=0}^{7}v_i x^i \tag{23}$$

的系数拆分为$\overline{q}_0,\overline{q}_1,\overline{q}_2,\overline{q}_3$. 如图3所示，8 bit的多项式乘法器输入为$\overline{t}_0,\overline{t}_1,\overline{t}_2,\overline{t}_3$和$\overline{q}_0,\overline{q}_1,\overline{q}_2,\overline{q}_3$. 递归至内部的3个4 bit多项式乘法器输入分别为$\overline{t}_0,\overline{t}_1$和$\overline{q}_0,\overline{q}_1$；$\overline{t}_0+\overline{t}_2,\overline{t}_1+\overline{t}_3$和$\overline{q}_0+\overline{q}_2,\overline{q}_1+\overline{q}_3$；$\overline{t}_2,\overline{t}_3$和$\overline{q}_2,\overline{q}_3$. 如图4所示，第一个4 bit多项式乘法器（图3中标红的乘法器）的输入为$\overline{t}_0,\overline{q}_0$和$\overline{t}_1,\overline{q}_1$.

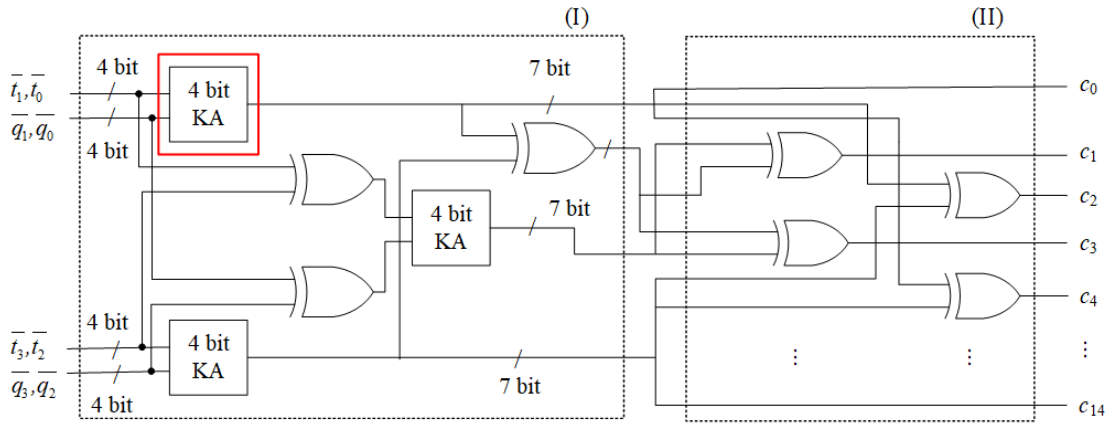


图3 8 bit重排序多项式乘法器

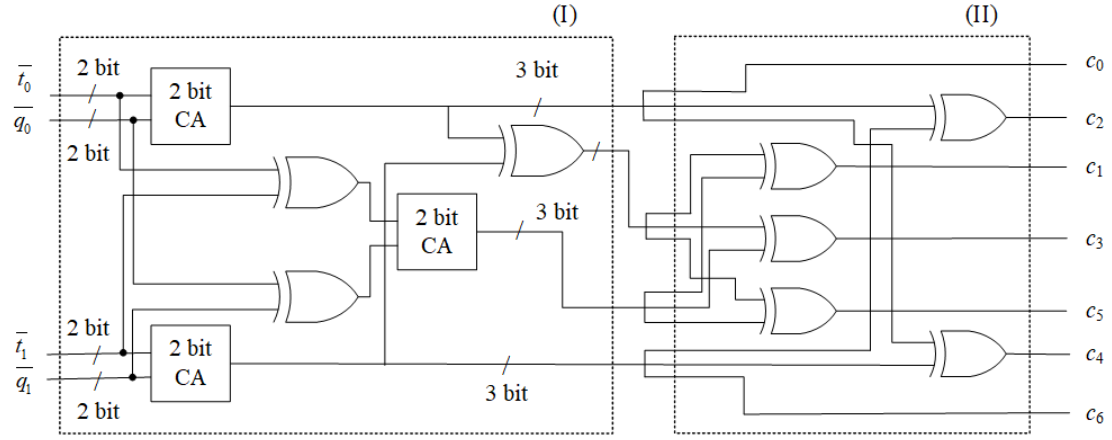


图4 第一个4 bit重排序多项式乘法器

3.2 混合串并架构

以往的设计通常直接用N项的多项式乘法模块并行处理，本文提出一种新的混合串并架构，在顶层串联使用$N/2$项的乘法架构，并在$N/2$点的乘法架构内部使用了并联的$N/4$点的OKA架构. 为了得到$N/4$点的乘法架构的输入，在重排序后，本方案利用预计算模块对重排序的各项进行计算，得到高位和低位的和项，作为作和输入项，并且和经过重排序模块的结果形成高位输入项、作和输入项、低位输入项，利用MUX选择输入到$N/2$点的OKA架构. 通过三次调用$N/4$点的乘法架构得到高位乘法输出、作和项乘法输出、低位乘法输出，并经过一段乘法电路得到最后的偶数项、奇数项和第0项结果. 以$N=8$为例，经过重排序后，本方案的前处理和多项式乘法模块如图5所示. 并行架构在输入和输出处实现了加法器的复用，与传统方案相比，将资源减小至原来的约1/4，而周期只变成了3倍，面效比获得了提升. 此外，还可以递归使用该架构，例如在$N/4$点的乘法架构内部继续串联使用$N/8$点的乘法架构，从而在输入和输出处实现更多的加法器复用，减小电路面积. 灵活递归使用此架构，可以适应对面积和延时的不同需求.

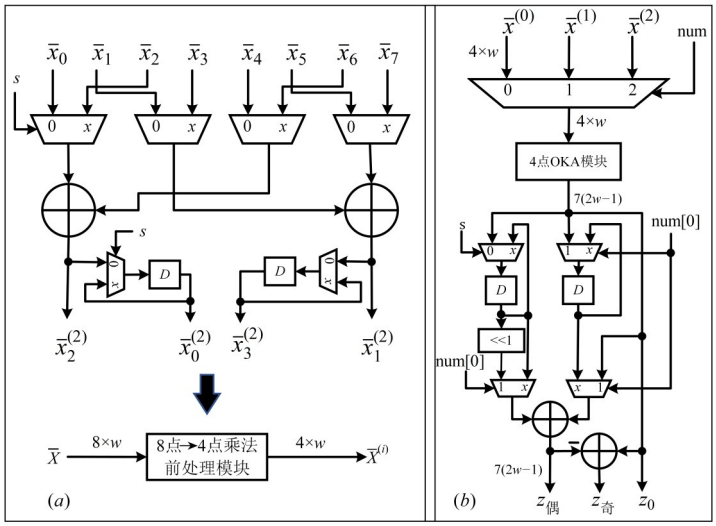


图5　8 bit混合串并架构的OKA乘法

3.3　逐列计算法

文献[9]中提出的逐列计算法避免了CA反复从块随机存取存储器(Block Random Access Memory, BRAM)读取数据的问题. 将此方法和OKA结合,设计出一种基于KA的高阶多项式乘法器(Large-Width Karatsuba Multiplier, LWKA). 相较于OKA, LWKA可采用串行计算,面积较小,从而更适合高阶的多项式乘法. 算法3详细介绍了LWKA的逐列计算法. 其中,OKA输出的结果位宽为$2n-1$,分为n比特的输出$\overline{R}_0$和$n-1$比特的输出$\overline{R}_1$.

算法3　逐列计算法

参数: 乘法器位宽r,OKA乘法器的位宽n,$l=[r/n]$,OVERHANG = $r \bmod n$

输入: 乘数$M[r-1:0]$, $K[r-1:0]$

输出: $R=M*K$

1. FOR $i=0,1,2,\cdots,l-1$ DO
2. $\quad \bar{M}[i]=M[(i+1)*n-1:i*n]$
3. $\quad \bar{K}[i]=K[(i+1)*n-1:i*n]$
4. $\quad \bar{R}[i]=0$
5. END FOR
6. $\bar{M}[l]=M[r-1:r-\text{OVERHANG}]$
7. $\bar{K}[l]=K[r-1:r-\text{OVERHANG}]$
8. FOR $i=0,1,2,\cdots,l$ DO
9. $\quad$ FOR $t=0,1,2,\cdots,l$ DO
10. $\quad\quad$ IF $t=0$ THEN
11. $\quad\quad\quad \bar{R}_0[t],\bar{R}_1[t]=\text{OKA}(\bar{M}[i],\bar{K}[t])$
12. $\quad\quad\quad \bar{R}[t]=\bar{R}[t]\oplus\bar{R}_0[t]$
13. $\quad\quad$ ENDIF
14. $\quad\quad$ ELSE IF $t\leqslant l-1$ THEN
15. $\quad\quad\quad \bar{R}_0[t],\bar{R}_1[t]=\text{OKA}(\overline{M}[i],\overline{K}[t])$
16. $\quad\quad\quad \bar{R}[t]=\bar{R}[t]\oplus\bar{R}_0[t]\oplus\bar{R}_1[t-1]$
17. $\quad\quad$ ENDIF
18. $\quad\quad$ ELSE IF $t=l$ THEN
19. $\quad\quad\quad$ IF OVERHANG = 0 THEN
20. $\quad\quad\quad\quad \text{REG}=\bar{R}_1[t-1][n-2:0]$
21. $\quad\quad\quad$ ELSE
22. $\quad\quad\quad\quad \bar{R}_0[t],\bar{R}_1[t]=\text{OKA}(\bar{M}[i],\bar{K}[t][\text{OVERHANG}-1:0])$
23. $\quad\quad\quad\quad \bar{R}[t]=\bar{R}[t]\oplus\bar{R}_0[t][\text{OVERHANG}-1:0]\oplus\bar{R}_1[t-1][\text{OVERHANG}-1:0]$
24. $\quad\quad\quad\quad$ IF OVERHANG = $n-1$ THEN
25. $\quad\quad\quad\quad\quad \text{REG}=\{\bar{R}_1[t],\bar{R}_0[t]\}[n+\text{OVERHANG}-2:\text{OVERHANG}]$
26. $\quad\quad\quad\quad$ ELSE
27. $\quad\quad\quad\quad \text{REG}=\bar{R}_1[t-1][n-2:\text{OVERHANG}]\oplus\{\bar{R}_1[t],\bar{R}_0[t]\}[n+\text{OVERHANG}-2:\text{OVERHANG}]$
28. $\quad\quad\quad\quad$ ENDIF
29. $\quad\quad\quad$ ENDIF
30. $\quad\quad\quad \bar{R}[0]=\bar{R}[0]\oplus\text{REG}$
31. $\quad\quad$ ENDIF
32. $\quad$ END FOR
33. $\quad$ FOR $t=l-1,l-2,\cdots,2,1$ DO
34. $\quad\quad \bar{K}[t]=\bar{K}[t-1]$
35. $\quad$ END FOR
36. $\quad$ IF OVERHANG = 0 THEN
37. $\quad\quad\quad \bar{K}[0]=\bar{K}[l-1]$
38. $\quad\quad$ ELSE
39. $\quad\quad \overline{K}[0]=\{\overline{K}[l][\text{OVERHANG}-1:0],\overline{K}[l-1][n-1:\text{OVERHANG}]\}$
40. $\quad\quad \bar{K}[l]=\bar{K}[l-1]$
41. $\quad$ ENDIF
42. END FOR
43. $R=\{\bar{R}[0],\bar{R}[1],\cdots,\bar{R}[l][\text{OVERHANG}-1;0]\}$
44. RETURN R

在该算法中，输入的乘数 M 和 K 的位宽均为 r，为避免面积较大，我们采取串行输入位宽为 n 的 OKA 乘法模块，$l=[r/n]$. 从 BRAM 中读取 M 和 K 的前 n bit $\bar{M}[0]$ 和 $\bar{K}[0]$，采用 OKA 算法相乘，得到结果后不改变 $\bar{M}[0]$，继续读取下一个 n bit $\bar{K}[1]$，继续相乘，直至读取完 K 的值，再重新读取 M 的下一个 n bit. 由于 BRAM 读取优先，存储和读取之间有一个时钟周期的延迟，所以每次计算结束后 $\bar{K}$ 的地址会后移一位，例如 $\bar{K}[0]$ 会储存至 $\bar{K}[1]$. 计算完一列后，$\bar{K}[0]$ 的值将更新为 $\bar{K}[l-1]$ 和 $\bar{K}[l]$ 的组合. 例如图 6 中，当 $r=10$，$n=3$ 时，首次读取 M 和 K 的前 3 bit $\bar{M}[0]$: m_0, m_1, m_2 和 $\bar{K}[0]$: k_0, k_1, k_2，输入乘法模块计算得到 $\bar{R}_0[1]$ 和 $\bar{R}_1[0]$，将 $\bar{R}_0[0]$ 存储至 $\bar{R}[0]$ 并将 $\bar{K}[0]$ 储存至 $\bar{K}[1]$. 接着读取 $\bar{K}[1]$（读取的值并非 $\bar{K}[0]$ 存入的值，以下同理），计算出 $\bar{R}_0[1]$ 和 $\bar{R}_1[1]$，将 $\bar{R}_0[1]$ 和 $\bar{R}_1[0]$ 相加，存储至 $\bar{R}[1]$ 并将 $\bar{K}[1]$ 存储至 $\bar{K}[2]$. $\bar{M}[0]$ 和 $\bar{K}[2]$ 计算得到 $\bar{R}_0[2]$ 和 $\bar{R}_1[2]$，$\bar{R}_0[2]$ 和 $\bar{R}_1[1]$ 相加存储至 $\bar{R}[2]$，并将 $\bar{K}[2]$ 存储至 $\bar{K}[3]$，同时使用寄存器 REG 存储 $\bar{R}_1[2]$ 的最高位（如图 6 中 $m_2 \cdot k_8$）. 读取 $\bar{K}[3]$ 后，截取最低位，和 $\bar{M}[0]$ 相乘后，将最高两位的值（如图 6 中 $m_1 \cdot k_9, m_2 \cdot k_9$）和寄存器 REG 中存储的值相加，再和 $\bar{R}[0]$ 相加，并将 $\bar{K}[2]$ 和 $\bar{K}[3]$ 组合得到 k_7, k_8, k_9，存储至 $\bar{K}[0]$. 此时，第一列的乘法计算完成. 在计算第二列时，读取 $\bar{M}[0]$: m_3, m_4, m_5，$\bar{K}[0]$: k_7, k_8, k_9，依次计算，并与存储的 $\bar{R}$ 相加，得到最终结果.

对于传统的多项式乘法器，当输入的多项式位数过大时，会产生 IO 口不够、乘法器面积过大等问题，这种大量的资源消耗是 FPGA 无法承受的. 我们使用传统多项式进行了实验，但是由于大位宽乘法消耗资源过多，无法完成综合实现. 然而，结合了逐列计算法的乘法器可采用串行输入的方式，其内部利用中等位宽的多项式乘法器，能够大大减小乘法器的面积，从而在高阶的多项式乘法中占有较大优势.

$$
\begin{aligned}
r_0 &= m_0 \cdot k_0 + m_1 \cdot k_9 + m_2 \cdot k_8 + m_3 \cdot k_7 + \ldots \\
r_1 &= m_0 \cdot k_1 + m_1 \cdot k_0 + m_2 \cdot k_9 + m_3 \cdot k_8 + \ldots \\
r_2 &= m_0 \cdot k_2 + m_1 \cdot k_1 + m_2 \cdot k_0 + m_3 \cdot k_9 + \ldots \\
r_3 &= m_0 \cdot k_3 + m_1 \cdot k_2 + m_2 \cdot k_1 + m_3 \cdot k_0 + \ldots \\
r_4 &= m_0 \cdot k_4 + m_1 \cdot k_3 + m_2 \cdot k_2 + m_3 \cdot k_1 + \ldots \\
r_5 &= m_0 \cdot k_5 + m_1 \cdot k_4 + m_2 \cdot k_3 + m_3 \cdot k_2 + \ldots \\
r_6 &= m_0 \cdot k_6 + m_1 \cdot k_5 + m_2 \cdot k_4 + m_3 \cdot k_3 + \ldots \\
r_7 &= m_0 \cdot k_7 + m_1 \cdot k_6 + m_2 \cdot k_5 + m_3 \cdot k_4 + \ldots \\
r_8 &= m_0 \cdot k_8 + m_1 \cdot k_7 + m_2 \cdot k_6 + m_3 \cdot k_5 + \ldots \\
r_9 &= m_0 \cdot k_9 + m_1 \cdot k_8 + m_2 \cdot k_7 + m_3 \cdot k_6 + \ldots
\end{aligned}
$$

图 6 $r=10, n=3$ 时的逐列计算法

3.4 应用了多项式乘法器的 BIKE 协议

本文使用基于 KA 的大尺寸有限域多项式乘法加速器取代了 BIKE 协议密钥生成模块原有的传统多项式乘法器，添加了相关的存储单元，修改了状态机模块和计数器. 采用了本文提出的位宽为 n bit 的混合串并架构的 KA 乘法器后，外部的多项式平方模块位宽由 n bit 拓展为 $2n$ bit，从而减小 BIKE 密钥生成模块运行的周期数和延时. 改进后的 BIKE 密钥生成模块示意图如图 7 所示.

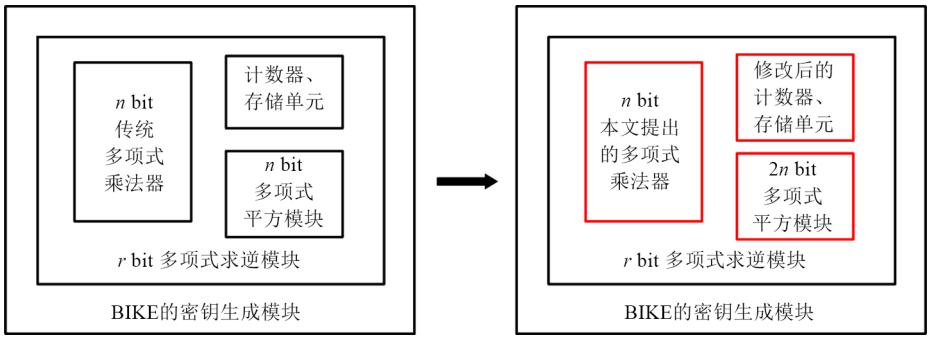


图 7 改进后的 BIKE 密钥生成模块示意图

4 实验结果

本文采用 system verilog 语言，并利用 Vivado 工具，设计了基于 KA 的多项式乘法和传统多项式乘法的全并行架构和混合串并架构，并在 xc7a200tffv1156-1 FPGA 上实现. 由于文献[9]中只说明所使用的 FPGA 系列，并未说明具体型号，为了公平比较，本文采用的是文献[9]中同系列的 FPGA 芯片，并且将文献[9]中开源的代码进行了统一平台的再综合，故表格中的实验数据和文献[9]中给出的结果略有不同. 对于输入多项式位宽为 n bit 的乘法器，本文的混合串并架构的顶层为串行的 $n/2$ bit 的乘法器，该乘法器底层均为并行架构，当递归到位宽为 16 bit 时，采用传统的乘法器进行计算.

表 3 列出了本文设计的混合串并乘法器与全并行乘法器以及传统乘法器的对比，为方便比较，统一将时钟频率设为 100 MHz，表中还给出了各乘法器方案不同

位宽下所消耗的LUT和Slice资源、所采用的Karatsuba级数(以16 bit为基准)以及每个方案对应的归一化Slice资源. 理论上,多项式乘法器的复杂度与阶数r成平方关系,若采用KA优化后,复杂度降低为$O\left(r^{\log_2 3}\right)$. 当时钟频率设置相同,且均采用全并行时,算法复杂度可以简单用资源复杂度来表示. 从传统方案CA和全并行KA设计的归一化Slice资源可见,实验的结果与理论分析基本一致. 为进一步降低资源,本文提出了混合串并方案,可实现在位宽增大时,采用更高一级的Karatsuba等级,而消耗的资源较少,代价为额外增加两个周期. 虽然从目前给出的单个设计结果来看,资源节省的速度略慢于延时增加的速度,但当r足够大,结合下面介绍的逐列计算方案,融合到BIKE加密系统中后,其增加的延时被多层级高并行调用机制弱化了. 从后面的实验结果可以看到,BIKE采用混合串并架构后,能够很容易采用更高等级的KA约简,获得更低的延时,且在部分条件下,能获得更优的面效比.

表3　本文设计的混合串并乘法器和并行乘法器、传统乘法器的对比

r/bit	乘法器方案	LUT	Slice	Karatsuba级数	归一化Slice	Freq/MHz
64	CA	1 565	481	—	1.0	100
	全并行KA设计	1 311	364	2	1.0	100
	混合串并KA设计	746	396	2	1.0	100
128	CA	6 410	1 888	—	3.9	100
	全并行KA设计	4 312	1 187	3	3.3	100
	混合串并KA设计	1 994	926	3	2.3	100
256	CA	24 652	6 317	—	13.1	100
	全并行KA设计	13 902	4 057	4	11.1	100
	混合串并KA设计	5 574	2 076	4	5.2	100

将本文提出的基于KA的高阶多项式乘法器应用于BIKE算法的密钥生成算法,并在xc7a200tffv1156-1 FPGA上实现了密钥生成算法的轻量级应用(乘法器位宽较小,如$n=32$ bit)和高速应用(乘法器位宽较大,如$n=128$ bit). 这一改进使密钥生成算法在延时、面积和面效比等方面都取得了一定的提升,也验证了混合串并架构的KA乘法器在具体应用中对延时的减小作用.

表4对比了本文改进后的BIKE协议密钥生成算法和文献[9]中算法FPGA实现的资源消耗、延迟、ADP和时钟频率. 对于安全系数为1,$r=12\,323$ bit,乘法器位宽$n=64$ bit的密钥生成算法,本文采用全并行乘法器的设计实现了最小的ADP. 乘法器位宽$n=64$ bit时,相比于文献[9]中的设计,本文的全并行乘法器将LUT数量减少了19.08%,面积减小了18.48%,ADP的值降低了8.97%,但是延时有所增加;而相同位宽的混合串并乘法器架构将延时减小了40.65%. 乘法器位宽$n=128$ bit时,相比于文献[9]中的设计,本文的全并行架构将LUT减小了30.39%,面积减小了33.01%,ADP降低了17.68%,但延时有所增加;相同位宽的混合串并乘法器架构延时降低了36.54%,ADP降低了10.4%.

表4　本文的设计和文献[9]中设计的指标对比:$r=12\,323$ bit

n/bit	乘法器方案	LUT	Slices	延时/ms	ADP	Freq/MHz
32	文献[9]中的设计	1 481	467	73.7	34 422.60	100.0
	混合串并KA设计	2 721	934	43.6	40 694.40	100.0
	全并行KA设计	1 462	464	88.5	41 040.70	83.3
64	文献[9]中的设计	3 528	1 012	36.9	37 294.20	83.3
	混合串并KA设计	6 259	2 085	21.9	45 661.50	79.4
	全并行KA设计	2 855	825	41.2	33 949.40	74.6
128	文献[13]中的设计	11 061	3 123	19.7	61 629.30	71.4
	混合串并KA设计	14 240	4 404	12.5	55 217.40	62.5
	全并行KA设计	7 700	2 092	24.3	50 733.10	58.1
256	混合串并KA设计	30 743	9 634	7.9	75 860.00	50.0
	全并行KA设计	20 084	5 392	15.3	82 677.60	45.5

表4最后2行对比了乘法器位宽$n=256$ bit时本文提出的混合串并乘法器架构和全并行乘法器架构的FPGA资源消耗、延迟、ADP和时钟频率. 由于FPGA接口的限制,无法继续仿真更高的乘法器位宽,且由于文献[9]中的设计在位宽大于128 bit时仿真时间过长且无法成功实现,故表4中无$n=256$ bit时文献[9]中设

计的各项指标. 对于安全系数为1,$r=12\,323$ bit,乘法器位宽$n=256$ bit的密钥生成算法,本文的混合串并乘法架构取得了最小的延时,仅7.874 ms,同时,和全并行乘法器架构相比,混合串并乘法器架构的ADP减小了8.25%. 由此可见,当乘法器位宽较小,如$n=64$ bit时,混合串并乘法架构的ADP大于全并行乘法架构,但当乘法器位宽逐渐增大时,混合串并乘法架构的ADP会逐渐改善,并优于全并行乘法架构.

5 总结

本文基于传统的Karatsuba乘法算法及其后续的改进算法,提出了一种基于KA的高阶多项式乘法器,分别采用全并行架构和混合串并架构,将其应用于后量子密码学中BIKE协议的公钥加密算法并在FPGA上实现. 与传统的多项式乘法器、并行架构的KA乘法器相比,基于KA的混合串并架构乘法器可以有效减少资源消耗,并且在位宽增加时效果更加显著. 将基于KA的乘法架构应用于BIKE的公钥加密算法后,本文的全并行乘法架构将其LUT减小了30.39%,面积减小了33.01%,ADP降低了17.68%;本文的混合串并乘法架构延时降低了36.54%,ADP降低了10.4%. 文献[9]中的设计无法提高乘法器位宽,而本文的设计可以解决此问题,同时降低延时,其中混合串并乘法器架构可将延时降低至7.874 ms. 此外,随着乘法器位宽的增加,混合串并乘法器架构的ADP降低速度大于全并行乘法器架构,因此更适用于大位宽的乘法器.

后续,我们计划继续改进乘法器内部的设计和参数,比如乘法器位宽、递归次数等,并适当加入流水线,取得更好的面效比,进一步增加乘法器位宽,测试位宽更大时混合串并乘法架构的各项指标. 同时,我们会将改乘法算法应用到其他后量子加密算法,改进算法的延时和面积.

致谢 感谢朱丹阳博士、宋逸峰博士给本文提出的参考意见.

参考文献

[1] CHEN L, JORDAN S, LIU Y K, et al. Report on post-quantum cryptography[M]. Gaithersburg: US Department of Commerce, National Institute of Standards and Technology, 2016.

[2] 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.

[3] MILLER V S. Use of elliptic curves in cryptography[M]// Advances in Cryptology - CRYPTO'85 Proceedings. Berlin: Springer Berlin Heidelberg, 2007: 417-426.

[4] 黎明, 吴丹, 戴葵, 等. 高性能可扩展公钥密码协处理器研究与设计[J]. 电子学报, 2011, 39(3): 665-670.
LI M, WU D, DAI K, et al. Research and design of a high-performance scalable public-key cipher coprocessor[J]. Acta Electronica Sinica, 2011, 39(3): 665-670. (in Chinese)

[5] SHOR P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J]. SIAM Journal on Computing, 1997, 26(5): 1484-1509.

[6] 张晓涵, 程池, 余天润. 对密钥不匹配攻击的进一步理论分析: 以NTRU-HRSS为例[J]. 电子学报, 2023, 51(4): 1081-1092.
ZHANG X H, CHENG C, YU T R. Further theoretical analysis of key mismatch attacks: A case study of NTRU-HRSS[J]. Acta Electronica Sinica, 2023, 51(4): 1081-1092. (in Chinese)

[7] ARAGON N, BARRETO P, BETTAIEB S, et al. BIKE: bit flipping key encapsulation round 3 submission[C]// NIST Post-Quantum Cryptography Standardization Conference. Florida: Fort Lauderdale, 2018: 1-14.

[8] NIST. Post Quantum Cryptography[EB/OL]. (2024-08-13) [2024-08-20]. https://csrc. nist. gov/Projects/post-quantum-cryptography.

[9] RICHTER-BROCKMANN J, MONO J, GÜNEYSU T. Folding BIKE: Scalable hardware implementation for reconfigurable devices[J]. IEEE Transactions on Computers, 2022, 71(5): 1204-1215.

[10] KARATSUBA A A, OFMAN Y P. Multiplication of many-digital numbers by automatic computers[J]. Doklady Akademii Nauk. SSSR, 1962, 145(2): 293-294.

[11] 陈韬, 李慧琴, 吴艾青, 等. 基于2KNTT的多项式乘法单元设计[J]. 电子学报, 2024, 52(2): 455-467.
CHEN T, LI H Q, WU A Q, et al. A polynomial multiplier design based on 2KNTT[J]. Acta Electronica Sinica, 2024, 52(2): 455-467. (in Chinese)

[12] ZHU Y H, ZHU M, YANG B H, et al. LWRpro: An energy-efficient configurable crypto-processor for module-LWR[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(3): 1146-1159.

[13] WONG Z Y, WONG D C, LEE W K, et al. High-speed RLWE-oriented polynomial multiplier utilizing karatsuba algorithm[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 68(6): 2157-2161.

[14] EL HADJ YOUSSEF W, MACHHOUT M, ZEGHID M, et al. Efficient hardware architecture of recursive Karatsuba-Ofman multiplier[C]//2008 3rd International Confer-

ence on Design and Technology of Integrated Systems in Nanoscale Era. Piscataway: IEEE, 2008: 1-6.

[15] HEIDARPUR M, MIRHASSANI M. An efficient and high-speed overlap-free karatsuba-based finite-field multiplier for FGPA implementation[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2021, 29(4): 667-676.

[16] FAN H, SUN J, GU M, et al. Overlap-free Karatsuba - Ofman polynomial multiplication algorithms[J]. IET Information Security, 2010, 4(1): 8-14.

作者简介

杨　柳　女，1999年出生于江苏省南通市. 2024年毕业于南京大学电子科学与工程学院. 现从事数字IC设计与验证工作.
E-mail: 18851987916@163.com

张永真　女，2001年出生于河南省杞县. 现为南京大学集成电路学院硕士生. 主要研究方向为后量子密码.
E-mail: yongzhenzhang@smail.nju.edu.cn

田　静　女，2020年毕业于南京大学电子学院信息与通信工程专业并获得工学博士学位. 现为南京大学集成电路学院助理教授、特聘研究员. 主要研究方向包括后量子密码技术、现代纠错码等计算密集型算法的高性能集成电路设计. 在领域主流期刊和会议上发表论文近40篇.
E-mail: tianjing@nju.edu.cn