DOI:10.3969/j.issn.1003-5060.2024.11.010

基于 NTT 的高效多项式乘法器设计及其 FPGA 实现

刘笑帆，　肖　昊，　赵延睿，　胡　越
(合肥工业大学 微电子学院,安徽 合肥　230601)

摘　要:基于快速数论变换(number theoretic transform,NTT)的多项式乘法运算是后量子密码(post-quantum cryptography,PQC)的重要组件,提高多项式乘法器的运算速度至关重要。文章基于现场可编程门阵列(field programmable gate array,FPGA)提出一种输入位宽为 14 位、长度为 1 024 的高效多项式乘法器硬件加速方案,设计一种无冗余可重用的蝶形运算单元电路。通过提高部分运算的并行度,实现模乘器接近 100% 的利用率,降低整个多项式乘法运算的迭代周期,提高整体运算速度。该乘法器最终被部署在 Xilinx Artix-7 FPGA 开发板上,实验结果表明,电路的最高工作频率为 238 MHz,多项式乘法运算的总体用时为 35.59 μs,对比现有的硬件设计,该文提出的电路运算效率提高 36.9%。

关键词:后量子密码(PQC);多项式乘法器;快速数论变换(NTT);模乘;现场可编程门阵列(FPGA)
中图分类号:TN492　　**文献标志码**:A　　**文章编号**:1003-5060(2024)11-1498-07

Design and FPGA implementation of efficient polynomial multiplier based on NTT

LIU Xiaofan,　XIAO Hao,　ZHAO Yanrui,　HU Yue
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: Polynomial multiplication based on number theoretic transform(NTT) is an important component of post-quantum cryptography(PQC) and it is very important to improve the operation speed of polynomial multipliers. Based on field programmable gate array(FPGA), this paper proposed an efficient hardware acceleration scheme for polynomial multiplier with input bit width of 14 bits and length of 1 024, and designed a non-redundant and reusable butterfly unit circuit. By improving the parallelism of some operations, the utilization rate of modulo multiplier is close to 100%, and the iteration cycle of the whole polynomial multiplication is reduced and the overall operation speed is improved. The multiplier was finally deployed on Xilinx Artix-7 FPGA development board. The experimental results show that the maximum operating frequency of the circuit is 238 MHz, and the overall time of polynomial multiplication is 35.59 μs. Compared with the existing hardware design, the algorithm efficiency of the proposed circuit is improved by 36.9%.

Key words: post-quantum cryptography(PQC); polynomial multiplier; number theoretic transform(NTT); modular multiplication; field programmable gate array(FPGA)

随着量子计算技术不断取得突破和计算机算力的大幅提升,经典密码体系面临危机。后量子密码(post-quantum cryptography,PQC)作为新的解决方案应运而生。PQC 是能够抵抗量子计算对现有密码算法攻击的新一代密码算法,其中基于格的密码算法提供了固件安全和硬件友好的方式去构建[1]。多项式乘法是基于格的方案中关键的计算瓶颈,而利用快速数论变换(number theoretic transform,NTT)可以将其计算复杂度从二次 $O(N^2)$ 降低到近似线性 $O(N\mathrm{lb}\ N)$[2],N

收稿日期:2023-01-09;**修回日期**:2023-03-20
基金项目:国家自然科学基金资助项目(61974039)
作者简介:刘笑帆(1996—),女,河北邯郸人,合肥工业大学硕士生;
肖　昊(1982—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:xiaohao@hfut.edu.cn.

为多项式的长度。因此，国内外围绕基于 NTT 的多项式乘法器的硬件设计展开了大量的工作。

基于 NTT 的经典零填充算法（Schönhage-Strassen algorithm，SSA）[3] 根据卷积理论，采用零填充方式对序列点进行 2 倍扩展以实现多项式乘法运算。文献[4]采用该算法进行多项式乘法器设计，并通过解耦合 NTT 的方式将系数存储量从 $4N$ 降至 $2N$。为避免点数的双倍扩展，文献[5-8]均采用负包裹卷积（negative wrapped convolution，NWC）算法[9]。文献[5]为解决高计算复杂度算法面向低功耗的嵌入式设备实现的问题，提出一种参数可配置的格密码处理器，采用非同址运算的常几何结构 NTT（constant geometry NTT，CG-NTT）[10]，但该结构的 NTT 在运算过程中对数据的读写访问需要采用乒乓策略，因此需要双倍的数据存储资源。文献[6-8，11]中 NTT 均采用同址运算（in-place）结构，该结构在 NTT 运算过程中对于中间数据的缓存不需要增加额外的存储资源，更加适合在资源有限的设备上实现。

由于 NTT 运算的输入数据序列与输出数据序列互为倒位序，通常 NTT 与逆 NTT（inverse NTT，INTT）分别采用不同的抽取策略，NTT 选取按时间奇偶抽选（DITNTT），INTT 选取按频率奇偶抽选（DIFINTT），以 DITNTT 与 DIFINTT 相结合的方式来避免倒位序操作，从而减少处理倒位序的运算周期。然而 DIT 与 DIF 算法的计算顺序不同，底层电路结构存在差异，要实现 NTT 与 INTT 共享底层硬件架构，部分工作通过增加冗余计算单元进行电路结构设计，文献[5-6，12]均在运算电路中增加冗余的模加器、模减器或模乘器。另外，在实现 2 个多项式的乘法运算时，通常在 NTT 阶段配置 2 个并行的运算电路同时进行 2 个多项式的 NTT 运算，而在对位系数相乘（point wise multiplication，PWM）以及 INTT 阶段时仅有一个运算电路工作，另一个运算电路处于空闲状态，运算资源的利用率不高。

综上所述，面向 PQC 的多项式乘法算法的设计存在以下问题：① 采用 DITNTT 与 DIFINTT 相结合的运算方式避免位倒序操作，导致运算电路设计存在冗余；② 实现多项式乘法运算的过程中，部分运算电路存在空闲状态。

本文针对 DITNTT 与 DIFINTT 相结合的运算方式，设计无冗余计算单元的可重用蝶形运算单元（butterfly unit，BFU）电路结构。基于所提出的 BFU 电路结构，设计一种高效、低延时的多项式乘法数据调度方案，该方案通过在 PWM 与 INTT 阶段复用 2 个 NTT 运算单元的方法，提高 NTT 运算单元的利用率和多项式乘法的并行度，从而减少运算实现的迭代周期。实验结果表明，本文设计并实现的基于 NTT 的多项式乘法器与现有的相关设计方案对比，电路面积与运算时间的乘积（area-ime product，ATP）较小，运算执行时间最短，运算效率提高 36.9%。

1　算法背景

1.1　NWC 多项式乘法算法

基于格的 PQC 算法中，多项式乘法定义在整系数多项式环 $R_q=Z_q[x]/f(x)$ 上，其中，环 R_q 为模整数 q 的整系数多项式环，环 Z_q 为模整数 q 的商环。若不可约多项式 $f(x)=x^N+1$，N 为多项式的长度，则可使用 NWC 算法，以避免 NTT 点数加倍和额外的模运算[7]。对于 N 维高阶多项式 $\boldsymbol{a}(x)=(a_0,\cdots,a_{N-1})$、$\boldsymbol{b}(x)=(b_0,\cdots,b_{N-1})\in R_q$，其中 $a_0,\cdots,a_{N-1}$、$b_0,\cdots,b_{N-1}\in Z_q$，多项式乘法运算 $\boldsymbol{c}(x)=\boldsymbol{a}(x)\times\boldsymbol{b}(x)$。下面给出基于 NTT 的 NWC 多项式乘法算法具体步骤。

1）进行预计算。计算公式为：

$$\bar{a}_i=a_i\cdot\phi_{2N}^i \bmod q \tag{1}$$

$$\bar{b}_i=b_i\cdot\phi_{2N}^i \bmod q \tag{2}$$

2）进行对位系数相乘，并将结果进行逆变换运算。计算公式为：

$$\bar{c}_i=\mathrm{INTT}(\mathrm{NTT}(\bar{a}_i)\odot\mathrm{NTT}(\bar{b}_i)) \tag{3}$$

3）进行后计算。计算公式为：

$$c_i=\bar{c}_i\cdot\phi_{2N}^{-i} \bmod q \tag{4}$$

其中：i、$j=0,1,2,\cdots,N-1$；ϕ_{2N} 为单位元的 $2N$ 次根；$q\equiv 1 \bmod 2N$。NWC 算法的主要运算单元为 NTT 运算和 INTT 运算，N 点 NTT 与 INTT 的计算公式为：

$$A_i=\mathrm{NTT}(a_i)=\sum_{j=0}^{N-1}a_i\omega_N^{ij} \bmod q \tag{5}$$

$$a_j=\mathrm{INTT}(A_j)=\frac{1}{N}\sum_{i=0}^{N-1}A_i\omega_N^{-ij} \bmod q \tag{6}$$

其中，ω_N 为定义在有限域中 N 点 NTT 的原根。$\omega_N{}^{-ij}$ 与 $\omega_N{}^{ij}$ 在模 q 的有限域上互为逆元，ϕ_{2N} 与 ω_N 之间存在如下关系：

$$\phi_{2N}^2\equiv\omega_N \bmod q \tag{7}$$

从计算形式上可以看出，INTT 与 NTT 仅旋转因子不同，根据旋转因子周期性和折半性，

NTT 与 INTT 的旋转因子互为相反数,因此两者可共用同一套旋转因子,即

$$\phi_{2N}^{i} = -\phi_{2N}^{N-i} \bmod q \tag{8}$$

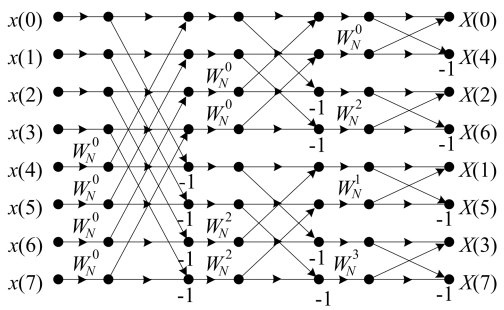
另外,INTT 最后需要模乘 $1/N$ 以修正结果,这一修正过程可拆解至运算的每一阶段,即对每一次的蝶形运算结果模乘 1/2。NTT 与 INTT 具有一致的计算结构,因此在进行硬件电路设计时,可以通过复用底层电路结构以减少电路资源的开销。

1.2 DITNR-NTT 和 DIFRN-INTT 算法

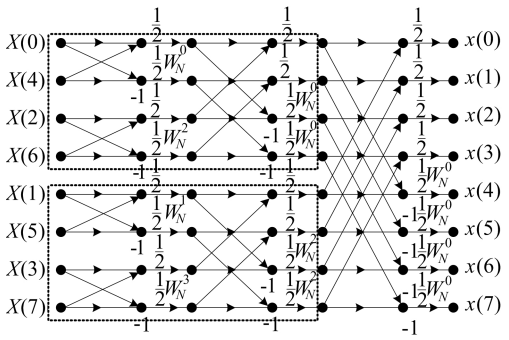
NTT 是有限域的傅里叶变换,其快速计算方式与快速傅里叶变换(fast Fourier transform, FFT)类似,具有多种不同的数据流形式,例如 CG-NTT 在级间和级内具有相同的数据路径,控制逻辑相对简单,但蝶形运算的输入和输出数据地址不同,使得存储空间的需求增加了 1 倍。对于大点数的 NTT,in-place 结构更为合适[4]。

直接执行经典 in-place DITNTT 和 DIFNTT,需要在 NTT 的最开始和 INTT 的结束分别进行倒位序操作,该过程需要 $2N$ 个时钟周期完成。由于经典 DITNTT 以倒位序输入,并以自然序输出,而经典 DIFINTT 以自然序输入、倒位序输出[8]。

本文在基于 NTT 的 NWC 多项式乘法中,根据数据抽取方式及其内部循环顺序,选取按时间抽取,自然序输入、倒位序输出(DIT NTT for input in natural order,DITNR)的方式实现 NTT;选取按频率抽取,倒位序输入、自然序输出的(DIF INTT for input in bit-reverse order,DIFRN)方式实现 INTT,由此避免倒位序操作带来的额外计算,可减少 $2N$ 个时钟周期以提高运算速度。同时,NTT 与 INTT 运算过程中可共用一套旋转因子,由此节省 50% 的 ROM 存储资源。DITNR-NTT 和 DIFRN-INTT 的算法数据流如图 1 所示。



(a) DITNR-NTT



(b) DIFRN-NTT

图 1 DITNR-NTT 和 DIFRN-INTT 算法数据流

采用低计算复杂度的 NWC 算法实现多项式乘法,同时通过对旋转因子进行预处理计算,以实现在 DITNR-NTT 中合并预计算、在 DIFRN-INTT 中合并后计算[6],模乘的计算量与经典算法相比,NTT 从 $(N/2)\text{lb}\ N+N$ 降至 $(N/2)\times \text{lb}\ N$,INTT 从 $(N/2)\text{lb}\ N+2N$ 降至 $(N/2)\times \text{lb}\ N$。对于 1 024 点运算,NTT 与 INTT 的模乘运算量分别减少 16.7% 和 28.6%。

1.3 优化的巴雷特约减算法

广泛使用的模乘算法有蒙哥马利模乘算法和巴雷特约减算法。其中,蒙哥马利模乘算法需要将运算数据转换至蒙哥马利域,最终的运算结果需要进行域转换操作,运算相对复杂。巴雷特约减算法分为两步运算,首先是常规乘法运算,其次是对乘法运算结果进行约减。巴雷特约减过程利用近似商思想,根据模数预计算出一个定值,然后通过乘法和移位操作即可实现模约减,由此避免对硬件不友好的除法运算。对于 32 位以下的模乘运算,采用巴雷特约减的效率优于蒙哥马利模乘[8]。

经典巴雷特约减算法中,首先乘以预计算的定值,然后进行向右移位操作[13],由于右移只取数据高位,部分数据低位与结果无关,因此可以先进行部分移位,该操作可缩减乘数位宽,从而减小乘法器尺寸以节省资源。

下面给出面向硬件优化的巴雷特约减算法描述。算法输入为:A、$B\in(0,q)$,其中,q 数据位宽为 n 位,算法输出为:$C=A\times B \bmod q$。令 $Z=A\cdot B\in(0,q^2)$,数据位宽为 $2n$ 位,预计算数据 $\mu=\lfloor 2^{2n}/q \rfloor$,数据位宽为 $(n+1)$ 位。首先令数据 Z 右移 $(n-1)$ 位,并乘以预计算的常系数 μ;然后将数据右移 $(n+1)$ 位,并乘以模数 q;最后将原始数据相乘结果 Z 与该数据相减得到模约减结果。另外,该算法还需要对结果数据进行修正,若结果

大于模数 q，则减去 q 以修正。

2　多项式乘法器设计方案

2.1　无冗余可重用 BFU 设计

in-place NTT 算法是利用同址运算的特点，以蝶形运算为核心的迭代算法，完成一次 N 点 NTT 运算，需要迭代 $(N/2)\mathrm{lb}\ N$ 次蝶形运算。DITNTT 与 DIFINTT 的核心运算均为蝶形运算操作，因此 BFU 为设计重点。DITNTT 与 DIFINTT 蝶形运算数据流如图 2 所示，由于 DIT 与 DIF 的蝶形运算结构对称，因此通过合理的电路设计，可以实现 NTT 与 INTT 共用同一蝶形运算电路结构。图 2 中：u、v 为蝶形运算的输入数据；U、V 为蝶形运算的输出数据；ω 为旋转因子。

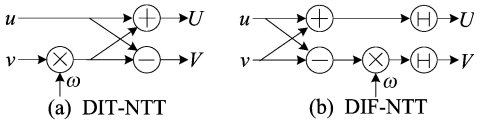


图 2　DITNTT 与 DIFINTT 蝶形运算数据流

为构建可重用 BFU，文献[5-6，12]设计的电路结构存在部分冗余，因此通过优化数据路径，合理排布电路结构以减少计算单元的冗余，BFU 底层模运算单元数量对比见表 1 所列。

表 1　BFU 底层模运算单元数量对比

方法	模加器	模减器	模乘器
文献[5]	2	2	1
文献[6]	1	2	1
文献[12]	1	1	3
本文设计	1	1	1

无冗余的可重用 BFU 结构如图 3 所示。图 3中：U、V 为蝶形运算的输入数据；W 为对应的旋转因子；U_{U}、V_{V} 为蝶形运算的输出数据。该电路可配置为 NTT 运算电路、INTT 运算电路以及 PWM 运算电路。根据选择信号的不同，通过多路选择器来控制数据路径的选通。INTT 需要对蝶形运算结果进行模乘 1/2，即模减半，这一操作并不需要使用模乘器，当数据为偶数时，直接右移 1 位即可；为奇数时，可通过移位和加法操作实现，即

$$\frac{1}{2}x \equiv (x+q) \gg 1 \equiv (x \gg 1) + \frac{q+1}{2} \bmod q \tag{9}$$

优化的 BFU 采用 6 级流水线设计，为平衡模乘器带来的电路延迟，保证数据流的时序正确性，需要在电路中插入 4 级寄存器。

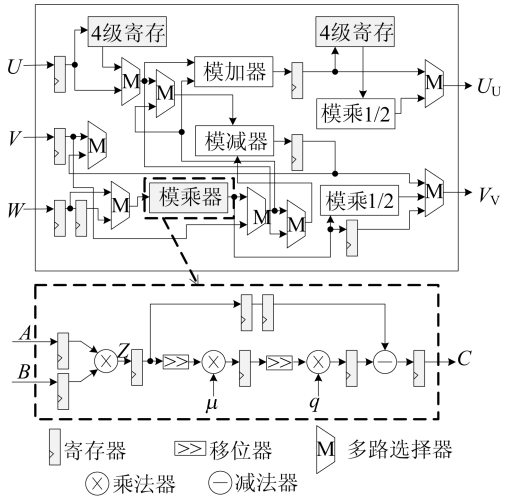


图 3　BFU 运算单元结构

2.2　高效、低延时的多项式乘法器设计

基于 NTT 的多项式乘法器的设计难点在于，in-place 结构带来的多项式系数存取的高复杂度，导致数据内存访问调度的管理复杂。相比于文献[7-8]提出的无冲突内存映射方案更为灵活，且不存在流水线气泡。选取通过第 3 轮 NIST 候选的 PQC 算法 Falcon-1024 的参数 ($N=1\ 024$，$q=12\ 289$)展开设计。根据文献[8]提出适用于本设计的内存映射方案，以 old_address 表示数据流中需要访问的数据点序列号，new_address 表示经映射电路后数据存储于 Bank 中的实际物理地址，bank_num 表示数据存储的 Bank 索引，数据映射方式如下：

$$\text{new\_address}[8:0] = \text{old\_address}[9:1] \tag{10}$$

$$\text{bank\_num}[0] = \wedge\ \text{old\_address}[9:0] \tag{11}$$

另外，NTT 的整个迭代过程需要运算单元与存储单元间不断地进行数据交互，由此引起两者间数据交互的高带宽要求，本文采用 Multi-Bank 策略，每个 BFU 与 2 个由简单双端口 BRAM 构成的存储 Bank 相对应。

多项式乘运算过程中，需要进行 2 次 NTT、1 次PWM 以及 1 次 INTT，因此配置 2 个并行 BFU。首先分别通过 BFU0、BFU1 并行计算多项式 $\boldsymbol{a}(x)$、$\boldsymbol{b}(x)$的 NTT 变换得到 $\boldsymbol{A}(x)$、$\boldsymbol{B}(x)$，对应数据存储于 Bank00、Bank01、Bank10、Bank11。多项式乘法运算的数据存储如图 4 所

示。PWM 与 INTT 阶段根据文献[4]中提出的解耦合思想,将数据分为上、下两部分并行进行。在 PWM 与 INTT 阶段,$\boldsymbol{C}(x)$ 数据存储使用 Bank00、Bank01 的上半部分以及 Bank10、Bank11 的下半部分。PWM_upper0、PWM_lower0 分别表示 Bank00 与 Bank10 上、下半部分数据的 PWM 运算,PWM_upper1、PWM_lower1 分别表示 Bank01 与 Bank11 上、下半部分数据的 PWM 运算。INTT 变换除最后一级外,也可将数据均分为上半部分和下半部分,分别通过 BFU0、BFU1 并行完成点数减半的 INTT 运算,INTT_upper 表示上半部分数据的 INTT 运算,INTT_lower 表示下半部分数据的 INTT 运算。INTT 最后一级单独处理,由此使得模乘器在多项式乘法运算过程中不存在空闲状态,达到接近 100%的利用率。

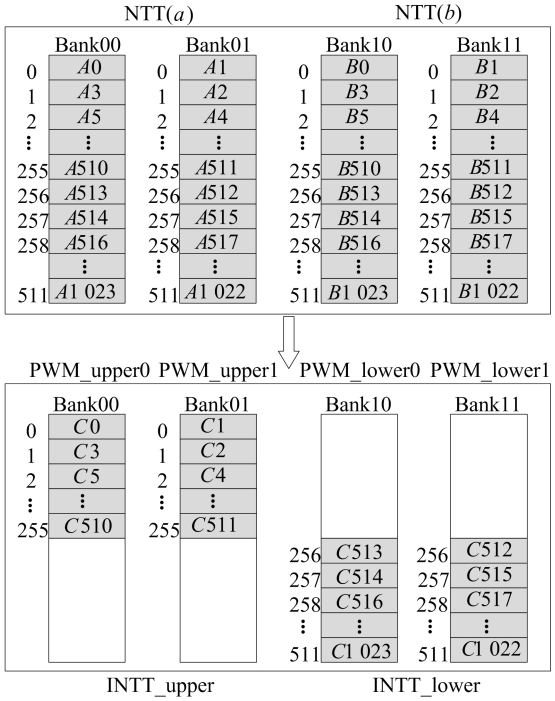


图 4 优化的高效、低延时多项式乘法器数据存储

多项式乘法器的数据调度如图 5 所示。

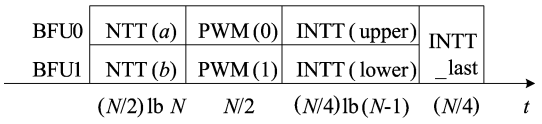


图 5 优化的高效、低延时多项式乘法器数据调度

图 5 中:PWM(0)表示存储于 Bank00 与 Bank10 的对应数据相乘;PWM(1)表示存储于 Bank10 与 Bank11 的对应数据相乘;INTT(upper)表示除 INTT 最后一级 $C(0)\sim C(N/2-1)$ 的 INTT 运算;INTT(lower)表示除 INTT 最后一级 $C(N/2-1)\sim C(N-1)$ 的 INTT 运算;INTT_last 表示 INTT 最后一级运算。

对于系数序列长度为 N 的多项式,优化的多项式乘法器完成一次需要的迭代周期为 $(3N/4)\mathrm{lb}\ N+N/2$,相比于经典 NWC 算法需要 $N\mathrm{lb}\ N+N$ 个时钟周期,本设计迭代周期减少 $(\mathrm{lb}\ N+2)/(4\mathrm{lb}\ N+4)$,当 $N=1\ 024$ 时,减少近 27.3%。

3 硬件架构实现与结果分析

3.1 整体硬件架构

本文设计的多项式乘法器电路共划分为 6 个模块,整体架构如图 6 所示。电路控制采用有限状态机(finite state machine,FSM)模式,生成控制信号 control_signal 和使能信号 en 等控制运算。地址生成模块生成的原始地址 old_address 经内存映射产生数据于存储模块中的实际物理地址 new_address。存储模块分为由 BRAM 实现的数据存储模块和由 ROM 实现的旋转因子 twiddle_factor 存储模块。为减小电路运算延迟,在运算模块配置 2 个 BFU 提高电路并行度。

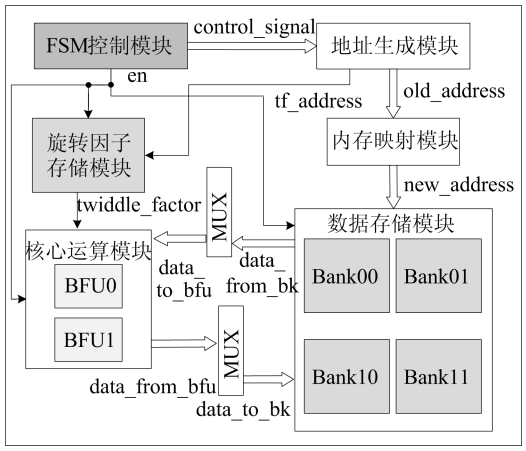


图 6 多项式乘法器整体架构

整个运算过程中 BFU 与 Bank 不断地进行数据交互。首先从 Bank 读取到的 data_from_bk 经 MUX 生成 BFU 的输入数据 data_to_bfu;然后 BFU 完成运算后的输出数据 data_from_bfu 经 MUX 生成存储于 Bank 的数据 data_to_bk;最终多项式乘法器产生的运算结果存储于数据存储模块。

3.2　结果分析

为验证本文硬件设计的正确性，采用基于 python 语言设计搭建多项式运算软件验证平台，硬件设计基于 Verilog HDL 语言，以 Vivado 2020.2 为开发平台，选择 Xilinx Artix-7 现场可编程门阵列（field programmable gate array，FPGA）开发板。2 个平台的实现结果表明，输入相同的数据，输出结果相同，验证了多项式乘法器的功能正确性。

NTT 运算单元资源对比见表 2 所列，给出运算单元电路的各个资源占用量。设计配置 2 个 BFU 运算单元，占用 452 个 LUT、348 个 FF 以及 6 个 DSP 资源。文献[5-6，12]中 BFU 运算电路均存在冗余，文献[5]以节能为目标，NTT 运算单元占用的 FF 数量较少，但占用了大量 LUT 资源，约为设计的 17 倍。文献[6]针对模数 12 289 提出了固定的模乘架构，模约减过程不需要乘法运算，因此仅占用 2 个 DSP，但 LUT 资源为设计的 1.87 倍。文献[12]的 BFU 为计算 INTT 额外增加了 2 个模乘器，占用大量 LUT 与 DSP 资源，其 LUT 数量约为设计的 4.34 倍，DSP 数量约为设计的 8.67 倍。

表 2　NTT 运算单元资源对比

方法	本文设计	文献[5]	文献[6]	文献[12]
器件	Artix-7	Artix-7	Artix-7	Zynq-7000
BFU 数量	2	2	2	1
LUT 数量	452	7 690	847	980
FF 数量	348	16	375	395
DSP 数量	6	11	2	26

给出电路的资源使用情况以及电路运行速度等数据，整体电路综合实现性能对比见表 3 所列。采用 ATP 反映硬件设计在面积与性能之间的平衡，ATPx 代表资源 x 使用数量与运行时间的乘积。设计占用 2.5 个 BRAM 和 6 个 DSP，最高工作频率可达 238 MHz，完成长度为 1 024 的多项式乘法运算总共需 8 473 个时钟周期，花费的计算时间为 35.59 μs，运算延时最短。与现有的硬件设计相比，实现的电路运算效率提高 36.9%。文献[14]配置了 4 个并行 BFU，占用的资源约为设计的 2 倍，然而运算过程中存在倒位序操作，最终迭代周期略少于设计。文献[15-16]仅配置 1 个BFU，文献[16]迭代周期过长导致电路延迟最慢。文献[15]中多项式乘法运算的迭代周期为设计的 2.05 倍，设计中 NTT 采用 CG-NTT 结构，控制电路简单，因此 LUT 与 FF 的 ATP 较优，然而乒乓存储策略引起存储量翻倍，故 BRAM 的 ATP 为设计的 2.28 倍。综上所述，相比于文献[14-16]，本文设计的多项式乘法器具有最短的运行时间。

表 3　多项式乘法器硬件性能对比

方法	本文设计	文献[14]	文献[15]	文献[16]
器件	Artix-7	Artix-7	Artix-7	Zynq-7000
运行时间/μs	35.59	52.32	56.62	88.45
时钟周期/个	8 473	7 848	17 382	22 201
LUT 数量	1 024	2 382	389	524
$ATP_{LUT}/10^{-3}$	36.44	124.63	22.03	46.38
FF 数量	658	1 381	346	823
$ATP_{FF}/10^{-3}$	23.42	72.25	19.59	72.79
BRAM 总量	2.5	10.0	3.5	3.0
ATP_{BRAM}	88.98	523.20	198.17	265.35
DSP 数量	6	8	1	3
ATP_{DSP}	213.54	418.56	56.62	265.35
频率/MHz	238	150	307	251

4　结　　论

为适应 PQC 对底层运算电路的要求，本文基于 FPGA 设计并实现一种优化的高效、低时延多项式乘法器硬件架构。乘法器选取 1 024 点、$q=12\,289$的参数设置，采用基于 NTT 的 NWC 多项式乘法算法，在 INTT 阶段以解耦合方式优化电路运算，保证模乘器接近 100%的利用率，从而提高电路实现效率；通过提高 PWM 与 INTT 阶段的并行度，节省 27.3%的多项式乘法运算周期；采用 DITNR-NTT 与 DIFRN-INTT 算法，避免倒位序操作并实现正逆变换旋转因子的共用，减少 50%的 ROM 使用；提出了无冗余计算单元的可重用 BFU 电路结构，减少了底层运算单元的资源使用；利用 in-place NTT 同址运算特点以及无冲突的内存访问策略，实现存储单元的数据存取。本文设计在多项式乘法器结构上尚有进一步改进空间。

[参　考　文　献]

[1] 陶云亭，孔凡玉，于佳，等. 抗量子格密码体制的快速数论变换算法研究综述[J]. 信息网络安全，2021，21(9)：46-51.

[2] PETER J N. Algebraic theory of finite fourier transforms [J]. Journal of Computer and System Sciences，1971，5(5)：524-547.

[3] SCHNHAGE A，STRASSEN V. Schnelle multiplikation groer zahlen[J]. Computing，1971，7(3)：281-292.

[4] ZHANG N，QIN Q，YUAN H，et al. NTTU：an area-effi-

cient low-power NTT-Uncoupled architecture for NTT-based multiplication[J]. IEEE Transactions on Computers, 2020,69(4):520-533.

[5] BANERJEE U,UKYAB T S,CHANDRAKASAN A P. Sapphire:a configurable crypto-processor for post-quantum lattice-based protocols[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems,2019,2019(4):17-61.

[6] ZHANG N,YANG B H,CHEN C,et al. Highly efficient architecture of newhope-nist on FPGA using low-complexity NTT/INTT[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems,2020,2020(2):49-72.

[7] MERT A C,KARABULUT E,OZTURK E,et al. An extensive study of flexible design methods for the number theoretic transform[J]. IEEE Transactions on Computers, 2020,71(11):2829-2843.

[8] CHEN X R,YANG B H,YIN S Y,et al. CFNTT:scalable radix-2/4 NTT multiplication architecture with an efficient conflict-free memory mapping scheme[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022,2022(1):94-126.

[9] CHEND D, MENTENS N, VERCAUTEREN F, et al. High-speed polynomial multiplication architecture for Ring-LWE and SHE cryptosystems[J]. IEEE Transactions on Circuits and Systems Ⅰ,2015,62(1):157-166.

[10] POLLARD J M. The fast Fourier transform in a finite field[J]. Mathematics of Computation, 1971, 25(114):365-374.

[11] XIN G Z,HAN J,YIN T Y,et al. Vpqc:a domain-specific vector processor for post-quantum cryptography based on risc-v architecture[J]. IEEE Transactions on Circuits and Systems Ⅰ,2020,67(8):2672-2684.

[12] FRITZMANN T,SEPULVEDA J. Efficient and flexible low-power NTT for lattice-based cryptography[C]//2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). [S. l.]:IEEE,2019:141-150.

[13] 车文洁,高献伟. 基于FPGA的进位保留Barrett模乘法器设计与实现[J]. 电子设计工程,2016,24(4):7-9.

[14] KUO P C,CHEN Y W,HSU Y C,et al. High performance post-quantum key exchange on FPGAs[J]. Journal of Information Science and Engineering,2021,2021(5):37-53.

[15] 陈朝晖,马原,荆继武. 格密码关键运算模块的硬件实现优化与评估[J]. 北京大学学报(自然科学版),2021,57(4):595-604.

[16] JATI A,GUPTA N,CHATTOPADHYAY A,et al. SPQ-Cop:side-channel protected post-quantum cryptoprocessor [EB/OL]. (2019-06-30)[2022-11-20]. https://eprint.iacr.org/2019/765.pdf.

(责任编辑 张 镅)

(上接第1485页)

[2] KOBLITZ N. Elliptic curve cryptosystems[J]. Mathematics of Computation,1987,48(177):203-209.

[3] MILLER V S. Use of elliptic curves in cryptography[C]// Advances in Cryptology-CRYPTO 85 Proceedings. Berlin: Springer-Verlag,1986:417-426.

[4] 侯惠芳,王云侠. 基于CPK和改进ECDH算法的可证安全的认证协议[J]. 计算机科学,2011,38(9):55-58.

[5] BERNSTEIN D J. Curve25519: new Diffie-Hellman speed records[C]//International Workshop on Public Key Cryptography. Berlin:Springer,2006:207-228.

[6] DIMOV V,KIRDAN E,PAHL M O. Resource tradeoffs for TLS secured MQTT-based IoTmanagement[C]//NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. [S. l.]:IEEE,2022:1-6.

[7] 成娟娟,郑昉昱,林璟锵,等. Curve25519椭圆曲线算法GPU高速实现[J]. 信息网络安全,2017(9):122-127.

[8] SASDRICH P,GÜNEYSU T. Efficient elliptic-curve cryptography using Curve25519 on reconfigurable devices[C]// International Symposium on Applied Reconfigurable Computing. Berlin:Springer,2014:25-36.

[9] SASDRICH P, GÜNEYSU T. Implementing Curve25519 for side-channel-protected elliptic curve cryptography[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS),2015,9(1):1-15.

[10] SASDRICH P, GÜNEYSU T. Exploring RFC 7748 for hardware implementation:Curve25519 and Curve448 with side-channel protection[J]. Journal of Hardware and Systems Security,2018,2(4):297-313.

[11] KOPPERMANN P, SANTIS F D, HEYSZL J, et al. X25519 hardware implementation for low-latency applications[C]//2016 Euromicro Conference on Digital System Design (DSD). [S. l.]:IEEE,2016:99-106.

[12] KOPPERMANN P,SANTIS F D,HEYSZL J,et al. Low-latency X25519 hardware implementation: breaking the 100 microseconds barrier[J]. Microprocessors and Microsystems,2017,52:491-497.

[13] KARATSUBA A,OFMAN Y. Multiplication of multidigit numbers on automata[J]. Soviet Physics Doklady,1962,7: 595-596.

[14] TURAN F, VERBAUWHEDE I. Compact and flexible FPGA implementation of Ed25519 and X25519[J]. ACM Transactions on Embedded Computing Systems (TECS), 2019,18(3):1-21.

[15] MONTGOMERY P L. Speeding the pollard and elliptic curve methods of factorization[J]. Mathematics of Computation,1987,48(177):243-264.

[16] ROY D B,MUKHOPADHYAY D. High-speed implementation of ECC scalar multiplication in GF(p) for generic montgomery curves[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27(7): 1587-1600.

(责任编辑 张 镅)