LAMP: Efficient Implementation of Lightweight Accelerator for Polynomial MultiPlication, From Falcon to RBLWE-ENC

Pengzhou He
phe@villanova.edu
Department of Electrical and Computer Engineering,
Villanova University
Villanova, PA, 19085, USA

Ben Mongirdas
bmongird@villanova.edu
Department of Electrical and Computer Engineering,
Villanova University
Villanova, PA, 19085, USA

Çetin Kaya Koç
cetinkoc@ucsb.edu
NUAA, Iğdır University, and UCSB
USA

Jiafeng Xie
jiafeng.xie@villanova.edu
Department of Electrical and Computer Engineering,
Villanova University
Villanova, PA, 19085, USA

ABSTRACT

Post-quantum cryptography (PQC) has drawn significant attention from the hardware design research community. In particular, efficient implementation for major components of PQC algorithms like polynomial multiplication has been a hot topic recently. Following this trend, in this paper, we propose a novel hardware-implemented **L**ightweight **A**ccelerator for the large integer polynomial **M**ulti**P**lication (LAMP) used in PQC schemes. Specifically, we target the polynomial multiplications not bound by fixed fast algorithms like Number Theoretic Transform (NTT), i.e., Falcon (one of the National Institute of Standards and Technology (NIST) selected PQC algorithms) and Ring Binary Learning-with-Errors based encryption scheme (RBLWE-ENC, a promising lightweight PQC scheme). Overall, we have carried out three layers of innovative efforts. (i) A new lightweight computation strategy for the targeted polynomial multiplication is proposed; (ii) The new accelerator is then designed based on the proposed algorithm (applicable for both targeted schemes); (iii) A thorough evaluation process is carried out to showcase the superior performance of the proposed accelerator over the competing designs, e.g., at least 21.2% less area-delay product (ADP) when LAMP is used for RBLWE-ENC (on Virtex-7 device). The proposed work is efficient and interesting, and we hope this outcome can facilitate PQC development.

CCS CONCEPTS

• **Hardware → Hardware accelerators**; • **Security and privacy → Hardware security implementation**; **Cryptography**.

GLSVLSI 2024, June 12-14, 2024, Tampa Bay Area, FL, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Falcon, FPGA platform, hardware design, lightweight polynomial multiplication accelerator, PQC, RBLWE-ENC.

ACM Reference Format:
Pengzhou He, Ben Mongirdas, Çetin Kaya Koç, and Jiafeng Xie. 2024. LAMP: Efficient Implementation of Lightweight Accelerator for Polynomial MultiPlication, From Falcon to RBLWE-ENC. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (GLSVLSI 2024).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

It is known that the existing cryptographic algorithms, such as Rivest Shamir Adleman and Elliptic Curve Cryptography, can be broken by large-scale quantum computers executing Shor's algorithm [16]. Indeed, this is the reason the National Institute of Standards and Technology (NIST) started the post-quantum cryptography (PQC) project for standardization [4] and has selected the first batch of algorithms for standardization in July of 2022 [4]. Among these selected algorithms, Falcon has gained substantial attention due to its unique algorithmic features and related implementation challenges (especially on hardware platforms) [7].

Apart from the general-purpose PQC standardization process, application-specific PQC, such as lightweight PQC, has also attracted the crypto community's interest [3, 18] (the recent National Science Foundation Secure and Trustworthy Cyberspace Principal Investigators' Meeting 2022 (SaTC PI Meeting'22) identified "lightweight PQC" as one of the future research directions [3]). Correspondingly, Ring-Binary-Learning-with-Errors (RBLWE) [6], a ring variant of binary-learning-with-errors (BLWE), is very promising for such applications. Since the initial introduction of the RBLWE-based encryption scheme (RBLWE-ENC) [6], quite a number of research works have been released [5, 9, 14, 15, 19].

Motivation and Challenges. It is observed that large integer polynomial multiplication is the major arithmetic component in both Falcon and RBLWE-ENC. Following the recent trend in the field that efficient and lightweight implementations of the polynomial multiplication deployed in PQC algorithms have become an important topic [12], it will be interesting if a general hardware implementation strategy can be developed for the polynomial multiplication used in the mentioned two PQC. Unlike other PQC

algorithms such as Kyber and Dilithium [4], where the algorithmic operations are directly built-in with number theoretic transform (NTT), the polynomial multiplications of Falcon and RBLWE-ENC are open to other types of implementation strategies. For instance, Falcon's signature verification phase needs a complete polynomial multiplication process, and hence, the related implementation is not limited to necessarily NTT, while the parameter sets of RBLWE-ENC are not in favor of deploying NTT. As a result, researching in this area has become very interesting.

On the other hand, there exist several challenges to obtaining the proposed goal: (i) the coefficients' setup for Falcon and RBLWE-ENC vary from each other; (ii) hardware designs for Falcon are still not abundant [7]; (iii) the polynomial-size of related schemes can be relatively large, e.g., $n = 1,024$ for Falcon, and thus, efficient and lightweight implementation is challenging.

Proposed Plan. Since the parameter sets of RBLWE-ENC are a little bit unique, we consider its implementation strategy first. It is observed that the polynomial multiplication of RBLWE-ENC has unequal-sized coefficients between two input polynomials, i.e., one binary polynomial and the other integers [14]. This setup brings challenges when deploying fast algorithms like Karatsuba as binary polynomial has to be split to be added respectively, and these additions increase the following operations' coefficient size including the related point-wise multiplications. Consequently, the gain of using a fast algorithm in this case may be offset. Therefore, several existing implementations for RBLWE-ENC were based on the schoolbook method or the variants [14, 15, 19]. While polynomial multiplication of Falcon can use NTT, the schoolbook method can also be efficient (according to a recent paper's study [17]). Therefore, we propose to use a schoolbook-based new method to derive an efficient computation strategy for transforming the targeted polynomial multiplications into desired lightweight accelerators.

Major Contributions. Following the above discussions, we propose a novel design strategy, i.e., **L**ightweight **A**ccelerator for polynomial **M**ulti**P**lication (LAMP), for Falcon and RBLWE-ENC. Overall, our main contributions include:

- We have rigorously formulated the major arithmetic operation of Falcon and RBLWE-ENC into a desired computation form for lightweight realization.
- We have designed the proposed LAMP accelerator with the help of a series of optimization techniques fitting respective PQC schemes.
- We have carried out a thorough evaluation to demonstrate the efficiency of the proposed accelerator.

The rest of the paper is organized as follows: Section 2 provides preliminary knowledge of this work. The proposed computation strategy is presented in Section 3. Accelerator and optimization techniques are described in Section 4. Evaluation and conclusions are given in Sections 5 and 6, respectively.

2 PRELIMINARY KNOWLEDGE

Notations. The following notations are used throughout the paper: (i) n is the polynomial size (security level); (ii) the targeted polynomial multiplication relies on the operations over ring $\mathbb{Z}_q/(x^n+1)$; (iii) $n = uv$ (u and v are integers).

Falcon. Falcon is a CCA-2 (Indistinguishability under Adaptive Chosen Ciphertext Attack) digital signature scheme selected by NIST [1, 2]. Overall, Falcon scheme can be described as: Falcon = GPV framework + NTRU lattices + Fast Fourier sampling [2], which is the consummation of many previous years' efforts. The details of Falcon's algorithmic features can be seen at [2].

RBLWE-ENC. RBLWE is a ring variant of BLWE with small key size and computational complexity (see its details in [6]). RBLWE-ENC retains the average hardness of the RBLWE problem, and analyses have ensured that it is secure enough for lightweight applications [6, 10]. As lightweight PQC is highlighted in the recent NSF SaTC PI Meeting'22 [3], the research on RBLWE-ENC becomes ever more important.

Polynomial Multiplication for Falcon and RBLWE-ENC. Both schemes use polynomial multiplication over ring $\mathbb{Z}_q/(x^n+1)$ as one of the major arithmetic components [2, 6]. The coefficients of the input/output polynomials for Falcon are all 14-bit since $q = 12,289$ [2]; while RBLWE-ENC uses one input of 1-bit coefficients and another of 8-bit coefficients (output is 8-bit) since $q = 256$ [6]. Besides that, the polynomial degree of Falcon is $n = 512/1,024$, and $n = 256/512$ for RBLWE-ENC.

3 PROPOSED ALGORITHM

Without loss of generality, we define the major arithmetic operation (polynomial multiplication) for Falcon and RBLWE-ENC as

$$T = GB \bmod f(x), \tag{1}$$

where $f(x) = x^n + 1$, $T = \sum_{i=0}^{n-1} t_i x^i$, $G = \sum_{i=0}^{n-1} g_i x^i$, and $B = \sum_{i=0}^{n-1} b_i x^i$ (bit-widths of t_i, g_i, and b_i are according to the specific scheme). Then, we have

$$\begin{aligned} T =& Bg_0 + \cdots + Bg_{n-1}x^{n-1} \bmod (x^n+1) \\ =& g_0 \sum_{i=0}^{n-1} b_i x^i \bmod (x^n+1) + \cdots \\ &+ g_{n-1}x^{n-1} \sum_{i=0}^{n-1} b_i x^i \bmod (x^n+1), \end{aligned} \tag{2}$$

where we can substitute with $x^n \equiv -1$ to further have

$$\begin{aligned} t_0 =& g_0b_0 - g_{n-1}b_1 - \cdots - g_1b_{n-1}, \\ t_1 =& g_1b_0 + g_0b_1 - \cdots - g_2b_{n-1}, \\ & \cdots \cdots \cdots \\ t_{n-1} =& g_{n-1}b_0 + g_{n-2}b_1 + \cdots + g_0b_{n-1}, \end{aligned} \tag{3}$$

where it is seen that t_0 can be obtained through the additions of n point-wise multiplications (similar to other t_i, $1 \le i \le n-1$). This strategy was used in [8, 11, 14] to design compact polynomial multiplication for RBLWE-ENC (not Falcon) since the involved point-wise operations have small area usage. This strategy, however, suffers from the major drawback that the processing speed is low and related input routing is expensive (one large-size n-to-1 MUX is used in this case).

Proposed Strategy. We propose a new computation strategy to enhance the processing speed while maintaining low computational complexity. The key ideas of the proposed strategy include: (i) the point-wise multiplications within single t_i of (3) are now computed through multiple channels to speed up the producing of the outputs;

(ii) the input routing resources, with respect to coefficients of B and G, are evenly distributed to minimize the computational complexity. Major steps of this strategy are listed below:

(i) Let us first define $[G^{(n-1)}] = [g_{n-1}, g_{n-2}, \cdots, g_0]$, where $[G^{(n-1)}]_0 = g_{n-1}$, $[G^{(n-1)}]_1 = g_{n-2}$, $\cdots$, $[G^{(n-1)}]_{n-1} = g_0$ (similar definition applies to other $[G^{(i)}]$, $1 \le i \le n-1$).

(ii) Let $n = uv$ (u and v are integers), we can decompose $[G^{(n-1)}]$ into u sub-vectors (each vector has v elements): $[G_0^{(n-1)}] = [g_{n-1}, \cdots, g_{n-v}]$, $\cdots$, $[G_{u-1}^{(n-1)}] = \{g_{v-1}, \cdots, g_0\}$, where $[G_0^{(n-1)}]_0 = g_{n-1}$, $\cdots$, $[G_0^{(n-1)}]_{v-1} = g_{n-v}$, $\cdots$, $[G_{u-1}^{(n-1)}]_{v-1} = g_0$. Note that the same definition applies to other $G^{(i)}$.

(iii) We can also define that $[B_0] = [b_0, \ldots, b_{v-1}]^T, \cdots, [B_{u-1}] = [b_{n-v}, \ldots, b_{n-1}]^T$, where $[B_0]_0 = b_0, \cdots, [B_{u-1}]_{v-1} = b_{n-1}$. Thus, we have the proposed algorithm as

Algorithm 1: Proposed algorithm for polynomial multiplication of Falcon and RBLWE-ENC

Input : G and B (bit-widths are determined by the specific scheme);

Output: $T = GB \bmod f(x)$ ($f(x) = x^n + 1$);

Initialization step

1 $\overline{[T]_0} = \cdots = \overline{[T]_{u-1}} = 0$;

2 Decompose B into $[B_0], [B_1], \cdots, [B_{u-1}]$;

3 Obtain $[G^{(n-1)}]$ as $[G_0^{(n-1)}], [G_1^{(n-1)}], \cdots, [G_{u-1}^{(n-1)}]$;

Main step

4 **for** $i = n-1$ *downto* 0 **do**

5 **for** $j = 0$ *to* $v-1$ **do**

6 $\overline{[T]_0} = \overline{[T]_0} + [G_0^{(i)}]_j [B_0]_j$;

7 $\cdots \cdots \cdots$;

8 $\overline{[T]_{u-1}} = \overline{[T]_{u-1}} + [G_{u-1}^{(i)}]_j [B_{u-1}]_j$;

9 **end**

10 $t_i = \overline{[T]_0} + \cdots + \overline{[T]_{u-1}}$;

11 Get $[G^{(i-1)}]$ from $[G^{(i)}]$. // until $[G^{(0)}]$;

12 **end**

Final step

13 Obtain output T from the serially produced t_i;

Note that to get $[G^{(i-1)}]$ from $[G^{(i)}]$ (Line 11), only circular-shifting of all related coefficients is needed with one value's sign inverted (see the expressions of t_i in (3)): the first coefficient of the $G^{(i)}$ becomes the negative value, as the nth coefficient of the $G^{(i-1)}$; and meanwhile all the coefficients of $G^{(i)}$ (from $i = n-1$ to 0) are circularly shifted (not counting the signs)). We also observe that all the coefficients of $G^{(n-1)}$ have not been inverted, and thus, we can start the computation with $G^{(n-1)}$ first.

Features. The proposed algorithm allows us to process multiple point-wise operations (multiplications and accumulations, Lines 6-8) simultaneously while reducing the input routing cost, which is desirable for Falcon (especially that n in Falcon is relatively large, e.g., $n = 1,024$ [2]). Following the algorithmic procedure of Algorithm 1, we will design the accelerators in the next section, first for Falcon and then for RBLWE-ENC.

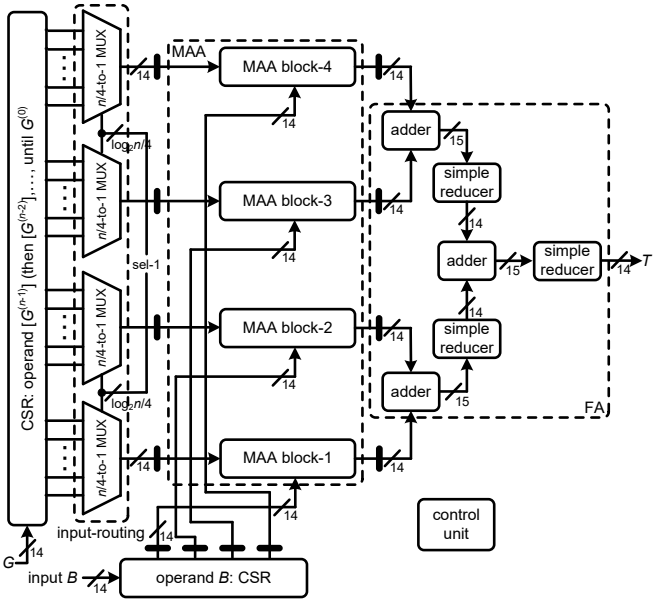

Figure 1: The proposed LAMP for Falcon ($u = 4$), where the black box denotes the register.

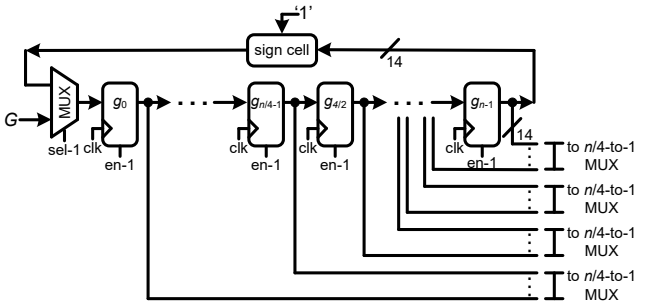

Figure 2: CSR for $[G^{(i)}]$ (first $[G^{(n-1)}]$), $i = n-1$ to $i = 0$. Note the values in the registers are initially loaded data.

4 LAMP: PROPOSED ACCELERATOR

We follow Algorithm 1 to present the design (LAMP) for Falcon first (because of its complicated architectural setup), which is then extended to RBLWE-ENC.

4.1 LAMP for Falcon

The proposed LAMP for Falcon is shown in Fig. 1, where we have used $u = 4$ as a case study example (which can be easily extended to other cases of u). This proposed accelerator consists of five major components: input circular shift-registers (CSRs), input-routing unit, multiplication-and-accumulation (MAA) unit, final addition (FA) unit, and control unit. These components' respective internal structures and functions are introduced below.

The CSR for G produces $[G^{(n-1)}]$, $[G^{(n-2)}]$, and so on, and finally $[G^{(0)}]$, according to Line-11 of Algorithm 1. The four $n/4$-to-1 MUXes in the input-routing unit function to produce needed

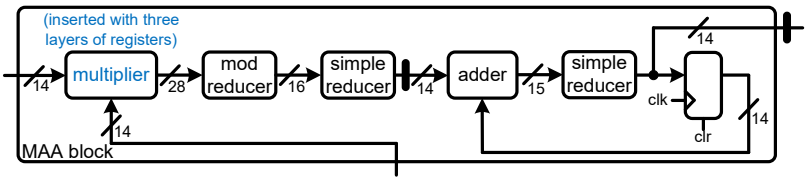

Figure 3: Internal structure of the MAA block.

```
function mod_reducer(C)
C0 ← C mod 2^m
C1 ← C/2^m
  return kC0 - C1
end function
(note: for q=12,289,m=12)
```

Figure 4: Function of the mod reducer, from [13].

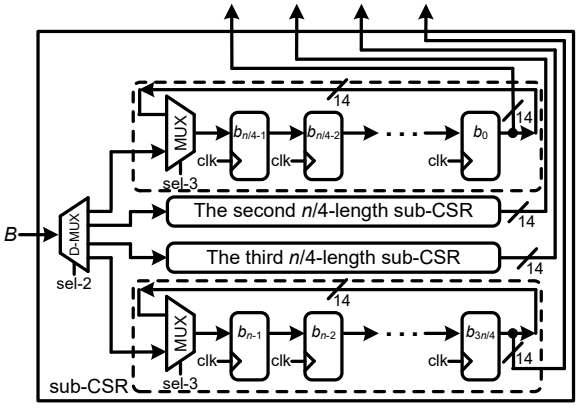

Figure 5: CSR for $[B]$ (where the values in the registers represent those that are loaded with initial values).

$[G_k^{(i)}]_j$ ($k = 0, 1, 2, 3$) based on Lines 5-9 (Algorithm 1). The details for CSR of $[G^{(i)}]$ are shown in Fig. 2. After the CSR for G is loaded with $[G^{(n-1)}]$, the selecting signals to these four $n/4$-to-1 MUXes begin to select each of these connected output data from CSR to be delivered to the following MAA unit for calculation, which lasts $n/4$ cycles. Then, CSR circularly shifts one position (with the help of the sign cell) to produce $[G^{(n-2)}]$ (Line-11 of Algorithm 1) for further MUX-based processing again (this process lasts for n rounds). Note that the sign cell involves a series of inverters followed by 1-bit half adders (and the least significant bit is added with '1') to meet the two's complement computation requirement (see Fig. 2).

There are, in total, four MAA blocks in the accelerator of Fig. 1, and its internal structure is described as follows. As shown in Fig. 3, each MAA block contains one multiplier (inserted with three layers of registers for pipelining), one mod reducer, two simple reducers, one adder, and two registers. Due to the specific q of Falcon, we have deployed the modular reduction method in [13] to obtain the accurate result. The function of the mode reducer is shown in Fig. 4, but it does not always produce the desired outcome after modulo operation (16-bit) [13], and a simple reducer is needed. The simple reducer basically checks the reduced value (mainly on the four most significant bits (MSBs) of its input) and then decides to subtract q (or $5q/4q/3q/2q/0$) to obtain the correct output (which is exactly 14-bit). The MAA block produces one output after all inputs to the $n/4$-to-1 MUX are routed. Related outputs are then added through the FA unit to produce t_i (according to Algorithm 1).

Meanwhile, the CSR for B is shown in Fig. 5, where four $n/4$-length sub-CSRs are involved to produce the needed output (D-MUX: De-MUX). Note that the registers placed at the output of the CSR for B are for lining up with the input routing unit, i.e., the outputs are delayed by registers (see Fig. 1).

Finally, the control unit (mainly a finite state machine (FSM)) generates all the needed signals for LAMP's proper operation. The FSM consists of two counters and other related logic cells. One counter is responsible for the counting of $n/4$ inputs for the two $n/4$-to-1 MUXes, as specified in Fig. 1. While another counter is responsible for counting the repeated operation for the generating of the control signal to the MUXes for n times. Meanwhile, the other control signals such as "clr" (clear), "en" (enable), "sel-1" and other ones in the CSR (see Fig. 2) can be generated by the logic functions related to the two counters.

The accelerator of Fig. 1 can be easily extended to the design with other values of v, where the differences lay mainly on the number of the MUXes (the number of MUXes × the size of the MUX = n) and the increased number of MAA blocks (also the extra adders to add the related outputs together).

4.2 Extension to RBLWE-ENC

Due to the specific parameter setup, LAMP involves a simpler structure than Falcon when it is extended to RBLWE-ENC. As shown in Fig. 6 (where we have again used $u = 4$ as the case study example), the accelerator contains similar components as those in Fig. 1, except there is an extra unit involved (to add another 8-bit polynomial, for the sake of comparing with the existing designs like [14]). Note that in this case, we define another polynomial as D, and the final output becomes W (Fig. 6).

The input CSRs in Fig. 6 are similar to those in Fig. 1 except the processing bit-widths. The MAA blocks in the MAA unit involve a much simpler structure than Fig. 1 as the parameter setting of RBLWE-ENC allows natural modular reduction (no specifically designed mod reducer and simple reducer). The multiplier in the MAA block is simply an AND cell, which delivers its result to the following accumulation cell (an adder looped with a register). When the accumulated output of each MAA block is available after the adder ($(n/4-1)$ cycles after the first inputs are fed to the multiplier),

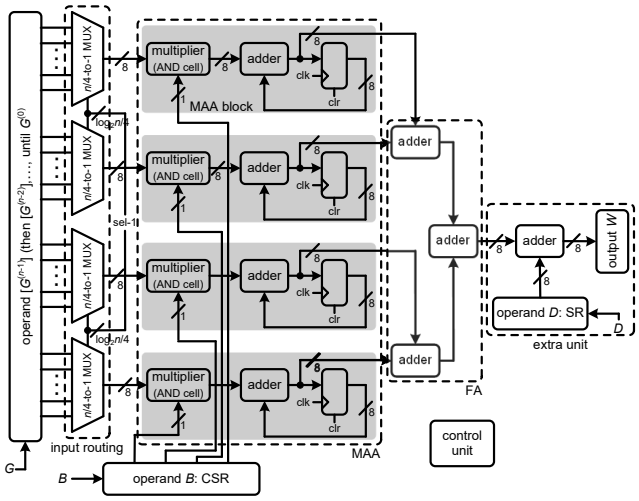

Figure 6: The proposed LAMP for RBLWE-ENC, for $u = 4$.

it is then delivered to the FA unit and go through the extra unit (for another addition) to yield the final result.

4.3 Brief Summary

Overall, LAMP is well designed for lightweight and compact implementation of the polynomial multiplication used in Falcon and RBLWE-ENC. LAMP takes n^2/u cycles to compute (not counting the inserted register layers for Falcon), and its design can be easily extended to other u (see the examples of Figs. 1 and 6). Note that due to the involved large processing bit-width, registers need to be inserted into the complicated operations of the accelerator for Falcon to maintain high frequency but not required for RBLWE-ENC.

5 EVALUATION: IMPLEMENTATION AND COMPARISON

This section gives a thorough evaluation of the proposed LAMP, covering complexity analysis, implementation, comparison, and discussions. Future work is also given at the end.

5.1 Complexity Analysis

In general, the proposed LAMP contains u number of multiplication cells (in MAA blocks), $(2u-1)$ number of adders (not including the one in the extra unit for the accelerator of RBLWE-ENC), two CSRs, u number of n/u-to-1 MUXes, $(u-1)$ registers (not including the ones in the CSRs as well as the inserted pipelined registers for Falcon), and a control unit. The structure has a latency time of (n^2/u) cycles (not counting the pipelined layers for LAMP of Falcon).

Besides that, in the LAMP for Falcon, each MAA block also contains one mod reducer and two simple reducers, as shown in Fig. 5, thus in a total of u mod reducers and $2u$ simpler reducers. Finally, there is a need of $(u-1)$ simple reducers in the FA for the Falcon-based LAMP.

Finally, we have to mention that the actual complexity of the LAMP for Falcon/RBLWE-ENC is also determined by the specific parameter sets, e.g., the bit-width of the related coefficients, which the following implementation results will reflect.

Table 1: Implementation Performance of LAMP (for Falcon) on FPGA Devices

design	ALM/Slice[1]	Fmax(MHz)[1]	latency	delay(μs)[1]
$n = 512$				
u=4	4,724/2984	230.41/187.86	68,608	298/365
u=8	7,056/3,837	227.38/178.48	35,840	158/201
u=16	11,330/5,695	218.53/170.15	20,480	94/120
$n = 1,024$				
u=4	7,237/4,706	242.78/174.46	268,288	1,105/1,538
u=8	9,467/5,576	223.66/172.14	137,216	614/797
u=16	13,781/7,561	221.29/170.10	71,680	324/421

[1]: Performance results are obtained from Stratix-V/Virtex-7 devices, respectively.

5.2 Implementation and Comparison

Experimental setup. We have coded LAMP in VHDL with functions verified through ModelSim. The experiment setup is: (i) we selected $u = 4/8/16$ for LAMP; (ii) we followed existing designs' field-programmable gate arrays (FPGAs) (e.g., [19] used Virtex-7 and [11, 14] used Stratix-V) to use AMD-Xilinx Virtex-7 XC7V2000t and Intel Stratix-V 5SGXMA9N1F45C2 devices on Vivado 2019.2 and Quartus Prime 17.0 platforms, respectively; (iii) we used $n = 512/1,024$ and $q = 12,289$ for Falcon [2], and $n = 256/512$ ($q = 256$) for RBLWE-ENC [6]; (iv) the obtained performance, namely the number of slices (or adaptive logic modules (ALMs)), maximum frequency (Fmax, MHz), latency cycles, delay ((1/Fmax)×latency), and area-delay product (ADP), are listed in tables; (v) we do not list any competing designs of Falcon as there are no schoolbook-based similar structures in the literature, but we list those recent compact RBLWE-ENC designs [8, 11, 14, 15, 19] for comparison, excluding the high-speed ones in [8, 14, 19].

Performance Discussion and Comparison. As seen from Table 1, one can see that the proposed LAMP for Falcon has a decent performance in area usage and timing for both cases of $n = 512$ and $n = 1,024$. Besides that, the maximum frequency of the accelerator remains relatively stable when u changes from 4 to 16. Finally, one can also observe that the latency of the accelerator drops linearly while the resource increase does not increase at the same rate. For instance, the latency of the design ($u = 16$ and $n = 512$) reduces to 30% of the one of $u = 16$ and $n = 512$, but the resource usage increase is only about twice (this fully demonstrates the efficiency of the proposed design).

A similar situation happens with the performance of the LAMP for RBLWE-ENC. As shown in Tables 2 and 3, respectively, LAMP obtains the best ADP when $u = 16$. Note that comparing with the data in Table 1, Falcon design incurs a relatively larger area than the RBLWE-ENC one when u becomes bigger. This is due to the fact that Falcon has a larger and complicated parameter sets [2]) than RBLWE-ENC.

Table 2: Comparison on Stratix-V FPGA (RBLWE-ENC)

design	ALM	Fmax(MHz)	latency	delay(μs)	ADP
$n = 256$					
[8]	3,472	201.25	65,792	327	1,135,344
[11]	1,864	316.96	65,536	207	385,848
[14]	846	221.29	66,048	298	252,108
[15]	715	423.55	33.5k	79	56,552
u=4	884	342.35	16,896	49	43,628
u=8	931	378.36	8,704	23	21,413
u=16	1,043	383.14	4,608	12	12,544
$n = 512$					
[8]	6,901	171.32	262,656	1,533	10,579,233
[11]	3,551	296.65	262,144	884	3,139,084
[14]	1,596	203.87	263,168	1,291	2,060,436
u=4	1,614	315.00	66,560	211	341,041
u=8	1,712	338.07	33,792	100	171,124
u=16	1,824	320.82	17,408	54	98,972

ADP=#ALMs×delay.

Table 3: Comparison on Virtex-7 FPGA (RBLWE-ENC)

design	Slice	Fmax(MHz)	latency	delay(μs)	ADP
$n = 256$					
[19]	71	510	66,304	130	9,231
[15]	189	427.095	33.5k	77.11	14,574
u=16	557	352.93	4,608	13.06	7,272
$n = 512$					
[19]	121	443	263,700	595.26	72,026
u=16	1,054	330.96	17,408	52.60	55,439

ADP=Slice×delay.

While considering the comparison with the existing design, RBLWE-ENC LAMP has significantly better area-time complexities than the existing RBLWE-ENC designs. For instance, as shown in Tables 2 and 3, LAMP ($u = 16$) has 21.2% and 23.0% less ADP than [19] on Virtex-7 device, and 77.8% and 95.2% less ADP than the best-competing designs on the Stratix-V platform, for $n = 256$ and $n = 512$, respectively.

5.3 Discussion

Though the proposed design strategy is based on the schoolbook method, it nonetheless obtains efficiency for both Falcon and RBLWE-ENC. To the authors' best knowledge, this is the first this kind of implementation work for the two schemes. Following the current PQC development trend that multiple schemes will be selected for future usage, developing a general method for different schemes may also be a new trend for implementation research.

5.4 Future Work

Future work can thus focus on developing novel complexity reduction methods to develop a more appropriate general implementation strategy for the mentioned two schemes. Other future work can also be deploying the proposed LAMP for a complete cryptographic processor building as well as side-channel attacks.

6 CONCLUSION

This paper has presented a novel design method to implement a general polynomial multiplication accelerator (LAMP) for Falcon and RBLWE-ENC. Key efforts include a detailed algorithmic derivation process, a dedicated architectural innovation and designing procedure, and a thorough evaluation (analysis, implementation, and comparison). The comparison showcases the superior performance of the LAMP over the existing ones. We hope the outcome of this research will stimulate more follow-up work in the field.

ACKNOWLEDGMENT

The work of J. Xie is supported by NIST-60NANB20D203 and NSF SaTC-2020625. This work was also supported by the Villanova Match and Villanova Undergraduate Research Fellowship Programs.

REFERENCES

[1] Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized- and-round-4.
[2] Falcon. https://falcon-sign.info/, 2022. (Falcon. https://falcon-sign.info/, 2022).
[3] National Science Foundation (NSF) 2022 Secure and Trustworthy Cyberspace Principal Investigators' Meeting – Break Out Group Reports/Slides: Security in a Post-Quantum World . (National Science Foundation (NSF) 2022 Secure and Trustworthy Cyberspace Principal Investigators' Meeting – Break Out Group Reports/Slides: Security in a Post-Quantum World). https://cps-vo.org/group/satc-pimtg22/breakouts
[4] Post-quantum cryptography project. US National Institute of Standards and Technology. . https://csrc.nist.gov/Projects/post-quantum-cryptography/
[5] Aydin Aysu and et al. 2018. Binary Ring-LWE hardware with power side-channel countermeasures. In *DATE*. 1253–1258.
[6] Johannes Buchmann and et al. 2016. High-performance and lightweight lattice-based public-key encryption. In *ACM international workshop on IoT privacy, trust, and security*. 2–9.
[7] Samuel Coulon and et al. 2023. Efficient hardware RNS decomposition for post-quantum signature scheme FALCON. *57th Asilomar Conference on Signals, Systems, and Computers* (2023), 1–8.
[8] Shahriar Ebrahimi and et al. 2019. Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT. *IEEE IoT-J* 6, 3 (2019), 5500–5507.
[9] Florian Göpfert and et al. 2017. A hybrid lattice basis reduction and quantum search attack on LWE. In *PQCrypto*. 184–202.
[10] Florian Göpfert and et al. 2017. A hybrid lattice basis reduction and quantum search attack on LWE. In *PQCrypto*. 184–202.
[11] Pengzhou He and et al. 2021. Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary Ring-LWE post-quantum cryptography. *IEEE JETCAS* 11, 2 (2021), 383–394.
[12] Safiullah Khan and et al. 2022. Area-Optimized Constant-Time Hardware Implementation for Polynomial Multiplication. *IEEE Embedded Systems Letters* 15, 1 (2022), 5–8.
[13] Patrick Longa and Michael Naehrig. 2016. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *Cryptology and Network Security: 15th Int. Conf.* 124–139.
[14] Benjamin Lucas and et al. 2022. Lightweight hardware implementation of binary ring-LWE PQC accelerator. *IEEE CAL* 21, 1 (2022), 17–20.
[15] Karim Shahbazi and et al. 2023. An Optimized Hardware Implementation of Modular Multiplication of Binary Ring LWE. *IEEE TETC* (2023), 817–821.
[16] Peter Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Ann. symp. found. of comp. sci.* 124–134.
[17] Yazheng Tu, Shi Bai, Jinjun Xiong, and Jiafeng Xie. 2024. Novel Schoolbook-Originated Polynomial Multiplication Accelerators for NTRU-based PQC. In *5th NIST PQC Standardization Conference*. 1–13.
[18] Jiafeng Xie and et al. 2020. Special session: The recent advance in hardware implementation of post-quantum cryptography. In *VTS*.
[19] Dongdong Xu and et al. 2022. A More Accurate and Robust Binary Ring-LWE Decryption Scheme and Its Hardware Implementation for IoT Devices. *IEEE TVLSI* 30, 8 (2022), 1007–1019.