An Efficient Unified Architecture for Polynomial Multiplications in Lattice-Based Cryptoschemes*

Francesco Antognazza[1][a], Alessandro Barenghi[1][b], Gerardo Pelosi[1][c] and Ruggero Susella[2]
[1]*Politecnico di Milano, Milano, Italy*
[2]*STMicrolectronics S.r.l., Agrate Brianza (MB), Italy*

Keywords: Lattice-Based Cryptography, Hardware Accelerators, Polynomial Ring Multipliers.

Abstract: The significant effort in the research and design of large-scale quantum computers has spurred a transition to post-quantum cryptographic primitives worldwide. The post-quantum cryptographic primitive standardization effort led by the US NIST has recently selected the asymmetric encryption primitive Kyber as its candidate for standardization. It has also indicated NTRU, another lattice-based primitive, as a valid alternative if intellectual property issues are not solved. Finally, a more conservative alternative to NTRU, NTRUPrime was also considered as an alternate candidate, due to its design choices which remove the possibility for a large set of attacks preemptively. All the aforementioned asymmetric primitives provide good performances, and are prime choices provide IoT devices with post-quantum confidentiality services. In this work, we propose a unified design for a hardware accelerator able to speed up the computation of polynomial multiplications, the workhorse operation in all of the aforementioned cryptosystems, managing the differences in the polynomial rings of the cryptosystems. Our design is also able to outperform the state of the art designs tailored specifically for NTRU, and provide latencies similar to the symmetric cryptographic elements required by the scheme for Kyber and NTRUPrime.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Public-key cryptography (PKC) plays a fundamental role in today's technology providing the properties of confidentiality, data and origin authentication and non-repudiability, and its diffusion is witnessed by the number of widely-used protocols that rely on it, such as TLS and PGP. PKC primitives are in wide use to encrypt data between two parties without a pre-shared secret over an insecure channel, or to build a Public Key Infrastructure, and to guarantee the integrity and authenticity of data in form of digital signatures. Currently the most used algorithms, RSA and Elliptic Curve cryptography, rely on the hardness of integer factoring, or the hardness of computing discrete logarithm in finite cyclic groups. However, in 1994, Peter Shor designed an algorithm for quantum computers which solves both the prime factoring and discrete logarithm problems with an exponential speedup with respect to classical computers, effectively breaking the corresponding cryptosystems (Sklavos et al., 2017).

Due to the long term confidentiality and data/origin authentication guarantees required from asymmetric cryptographic primitives, and in sight of the recent advancements in the implementation of quantum computers, a significant effort in standardizing quantum-resistant algorithms for public-key cryptography is urgently required. For that reason, the National Institute of Standards and Technology (NIST) in 2016 started the Post-Quantum Cryptography (PQC) standardization process to assess viable candidates either for Public Key Encryption (PKE), in form of a Key Encapsulation Mechanism (KEM), and for digital signatures. The process refined its 69 candidate algorithms, reducing them to to a single KEM and three digital signatures for immediate standardization at the end of the third round (NIST PQC Team, 2022). Furthermore NIST provided a list of candidates which are still under investigation as alternate, as they rely on different computationally hard problems. Arguably the the most successful class of algorithms of this standardization process is

[a] https://orcid.org/0000-0003-3480-486X
[b] https://orcid.org/0000-0003-0840-6358
[c] https://orcid.org/0000-0002-3812-5429
*This Research Was Made Possible Thanks to the Support of STMicroelectronics.

Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

lattice-based algorithms, being attractive in terms of computational latency and with acceptable key and ciphertext sizes.

Besides the candidate selected for immediate standardization, Kyber, three other schemes were deemed particularly interesting in the contest: NTRU, NTRU Prime and Saber. NTRU was officially recommended as the fallback alternative in case patent issues cannot be solved by the end of 2023 (Alagic et al., 2022); NTRU Prime is an NTRU variant with conservative choices in the underlying algebraic structure, which prevent a number of attacks preemptively; Saber is based on a slightly different algebraic problem with respect to Kyber (Module Ring-learning with roundings instead of Module Ring-learning with errors) which is at least as computationally hard as the one of Kyber.

The four aforementioned lattice based cryptosystems rely on the arithmetics of polynomials with integer coefficients modulo q, where q is either a power of two, or a small prime; all considered modulo a polynomial with a low number of terms. Depending on the choices, the polynomial ring obtained may be more or less friendly to sub-quadratic multiplication techniques. Among such techniques, the Number-Theoretic Transform (NTT) is the most efficient way to perform a multiplication, provided that the maximum degree of the polynomial generating the ring is a power-of-two and the coefficient ring is modulo a prime: given an n degree polynomial, it runs in $O(n \log_2(n)))$ sequential steps. By contrast, efficient versions of the schoolbook algorithm, which runs in $O(n^2)$ such as the one by Comba, can always be applied, leading to extremely compact designs but also reduced throughput. Software and hardware implementations of the multiplication algorithms also rely on divide-et-impera techniques such as Karatsuba or Toom-Cook decompositions: these techniques trade off an increased design complexity and larger constants hidden in the O notation for a constant decrease in the complexity exponent. An emerging hardware design approach is the one known in the literature as x-net or LFSR-based multiplier. Its undelrying idea is to perform n coefficient-wise multiplications per clock cycle, resulting in a total computation time which is $O(n)$.

Contributions. Our work aims to show that it is possible to have a unified design for an hardware accelerator computing the polynomial multiplication in all the polynomial rings of the four lattice-based cryptosystems: Kyber, NTRU, NTRU Prime and Saber. The structure of such an accelerator stems from an architecture able to achieve efficiency results beyond the state of the art for NTRU-like cryptosystems. We provide efficiency results of a synthesis-time specialized accelerator for the arithmetic used by NTRU HPS and HRSS, NTRU Prime, Saber and Kyber cryptoschemes, namely every round-3 lattice-based KEM proposals at NIST's Post-Quantum standardization contest. Subsequently, we provide a unified design supporting all the polynomial rings, for all security levels of the KEMs, allowing cryptographic agility without the need of replacing the hardware component. We validated the correctness of the results, gathered the performance and resource figures for every parameter set specified by the latest specifications, for both an FPGA design flow. We note that our design uses a sequential memory layout to store polynomial coefficients in memory, and accesses them in a single sweep: our design is thus eligible to be be used in a pipelined fashion, a feature not achievable with current NTT based multipliers.

2 BACKGROUND

In this section, we provide a summary of the polynomial arithmetics for the polynomial rings employed in Kyber, NTRU, NTRU Prime and Saber. Subsequently, we provide a summary of linear-time hardware modular multipliers obtained with the x-net technique. In the following, we will denote polynomials of degree n with lowercase letters, highlighting the variable, as $a(x) = \sum_{i=0}^{n-1} a_i x^i$.

The aforementioned cryptosystems consider the arithmetics over two quotient polynomial rings each, $\mathcal{R}_q$ and $\mathcal{R}_p$ are $\mathbb{Z}_q[x]/\langle p(x)\rangle$ and $\mathbb{Z}_p[x]/\langle p(x)\rangle$. The differences in the ring structures arise from the choice of the values of p, q and $p(x)$, of which a summary is reported in Table 2. In particular, p is always chosen to be a small odd number between 3 and 11; q is either a small power of two (between 2^{11} and 2^{12}) or a prime number of the same order of magnitude. The latter choice yields polynomials with coefficient over a field, $\mathbb{Z}_p$, while the former choice allows a trivial modular reduction $\bmod q$ via truncation of the most significant bits.

The polynomial employed to obtain the quotient ring, $p(x)$ gives $\mathcal{R}_q$ and $\mathcal{R}_p$ a cyclic structure in Kyber and Saber ($x^n + 1$), a nega-cyclic structure in NTRU ($x^n - 1$). The cyclic structure name stems from the fact that, given an element $a(x) \in R_p$, computing the result of $x \cdot a(x)$ is equivalent to cyclically shifting its coefficients towards the higher degrees by one position. Similarly, the nega-cyclic structure implies that the same cyclic shift takes place, but a sign flip of the constant term also takes place after the cyclic shift. The authors of NTRU Prime chose $x^n - x - 1$ as

Table 1: Number of $\mathcal{R}_p \times \mathcal{R}_q$ multiplications in encapsulation and decapsulation primitives of each analyzed cryptoschemes. One further $\mathcal{R}_q \times \mathcal{R}_q$ multiplication is used during the decapsulation in NTRU, denoted by $\star$ symbol. Module-based cryptoschemes Saber and Kyber perform one $k \times k$ matrix-vector and either one or two vector-vector multiplications, where the elements are polynomials in $\mathcal{R}_q$ or $\mathcal{R}_p$, during encapsulation and decapsulation, respectively.

Cryptoscheme	**Module rank** k	**Multiplications**	
		Encap.	**Decap.**
NTRU	/	1	$2^\star$
NTRU Prime	/	1	3
NTRU LPRime	/	2	3
Saber	2,3,4	k^2+k	k^2+2k
Kyber	2,3,4	k^2+k	k^2+2k

Table 2: Summary of the features of the polynomial rings $\mathcal{R}_p = \mathbb{Z}_p[x]/\langle p(x)\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle p(x)\rangle$ for each KEM.

Scheme	**q**	**p**	**n**	$\mathbf{p(x)}$
Kyber	prime valori	5, 7	2^i 256	x^n+1
Saber	2^i valori	7, 9, 11	2^i 256	x^n+1
NTRU	2^i valori	3	prime valori	x^n-1
NTRU Prime	prime valori	3	prime valori	x^n-x-1

the polynomial modulus, thus obtaining polynomial fields for $\mathcal{R}_q$ and $\mathcal{R}_p$: this removes constraints further the ring structure which is present in Kyber and Saber, preventing future attacks which may exploit it.

To align our notation with the one of the cipher specifications, we will consider the representatives of the coefficient ring as the integers balanced around the zero element, for example between $-\lceil (q-1)/2 \rceil$ and $\lfloor (q-1)/2 \rfloor$.

Modular Polynomial Multiplication Algorithms. Modular Polynomial multiplications with large operands are extremely common in cryptographic primitives, and have thus seen significant efforts in their optimization. A first classification criterion is the strategy which is employed to perform the modular polynomial reduction: indeed, depending on the algorithm being employed, it is sometimes possible to interleave the reduction operation with the intermediate steps of the multiplication algorithm, saving on the memory elements required for the computation. The second classification criterion is the asymptotic complexity of the multiplication method, counted as the number of coefficient-wise multiplications, as a function of the number of coefficients of the operands, n.

The operand scanning, schoolbook method involves $O(n^2)$ coefficient-wise multiplications as it adds together all the results of multiplying the first polynomial factor by each one of the monomials composing the second factor. Sub-quadratic methods, pioneered by Karatsuba (Karatsuba, 1963), provide an algorithm to compute the polynomial multiplication in $O(n^{\log_a(2a-1)})$, where $a \geq 2$, coefficient-wise multiplications. In particular, Karatsuba proposed the algorithmic variant for $a = 2$, while Toom and Cook generalized the result for $a > 2$. The reason for avoiding the ubiquitous application of such methods is that, while the number of coefficient-wise application decreases, they require an increasing number of polynomial additions and subtractions to compute the result. While additions and subtractions have a linear cost in n, their overhead offsets the gains coming from saving multiplications for small values of n. Given that the ratio between the absolute values of the computational costs of multiplications and additions/subtractions varies depending on the platform it is commonplace to determine the break-even value for a through exhaustive evaluation for a specific design. In our context, Karatsuba was used in (Marotzke, 2020) instantiating three parallel schoolbook Comba multipliers, and (Dang et al., 2021) design involved a 3-way Toom-Cook computing five parallel multiplications recursively with odd-even Karatsuba method.

Finally, it is possible to compute polynomial multiplications in $O(n\log_2(n))$ exploiting Fourier transforms. The method relies on the fact that multiplying two polynomials can be seen as the convolution of their coefficients, interpreted as integer sequences. This allows to perform the multiplication computing the discrete-time Fourier transform of the sequences, performing the element-wise multiplication of the results and computing the inverse Fourier transform. The total cost of the operation depends on the cost of computing the Fourier transform, to which a linear amount of coefficient-wise multiplications must be added. For the special case where n is a power of two, computing the Fourier transform takes $O(n\log_2(n))$, thus resulting in a $O(2(n\log_2(n)) + n) = O(n\log_2(n))$ cost for the entire multiplication. This technique is applied fruitfully to polynomials in a ring $\mathbb{Z}_q[x]/\langle p(x)\rangle$, provided that the degree of $p(x)$ is a power of two, and that $\mathbb{Z}_q$ is a field, thus providing all the required roots of unity, and goes by the name of Number Theoretic Transform (NTT) (Dang et al., 2021). As it is the case for the sub-quadratic multiplication techniques, also the NTT requires some linear-time operations to be computed, and thus the break-even point for the value of n is sought experimentally. Of the four cryptosystems we are considering, only Kyber has a parameter choice which allows the use of

NTT based techniques.

Linear-Time Modular Multiplication Algorithms. An orthogonal approach to the redesign of the multiplication algorithm is the one which exploits the inherent parallelism of the schoolbook approach. Indeed, all the coefficient-wise multiplications involved in a factor-times-monomial product can be computed independently. This observation leads to the design of a linear-time multiplication algorithm which exploits n computation units and n coefficient-wide memories to compute the entire product in $O(n)$.

The first proposal of a linear-time modular multiplication algorithm specialized for the NTRUEncrypt polynomial ring comes from (Liu and Wu, 2015). The work achieves the multiplication in n clock cycles using n parallel multiply-and-accumulate (MAC) units. Furthermore, to reduce the area of each MAC unit, the work replaces the multiplier with a multiplexer, which selects one of the three possible coefficient-wise multiplication outcomes, thanks to the small size of the coefficients of the $\mathcal{R}_p$ operand.

This approach was then separately adapted for different polynomial rings of Saber, NTRU, and NTRU Prime cryptoschemes in (Basso and Roy, 2021; Dang et al., 2021; Farahmand et al., 2019). The authors of (Basso and Roy, 2021) proposed a centralized way to compute the few possible coefficient-wise multiplication results, and distribute them to every MAC unit. In (Peng et al., 2021) it is proposed to postpone the reduction $\bmod q$ of the coefficients of the multiplication result to the end of the multiplication. This approach entails larger accumulators to store them, while allowing to save area as only a single modular reduction unit is required.

3 PROPOSED ARCHITECTURE

Algorithm 1: x-Net polynomial multiplier.

```
Input: $a(x) \in \mathbb{Z}_p[x]/\langle p(x)\rangle,\ a(x) = \sum_{i=0}^{n-1} a_i x^i$
       $b(x) \in \mathbb{Z}_q[x]/\langle p(x)\rangle,\ b(x) = \sum_{i=0}^{n-1} b_i x^i$
Output: $r(x) = \text{LIFT}\left(a(x), \mathbb{Z}_q[x]/\langle p(x)\rangle\right) b(x)$
Data: $p(x) \in \mathbb{Z}_q[x]$, monic, with degree $n$
1 $r(x) \leftarrow 0$
2 for $i = 0$ to $(n-1)$ do
3     $r(x) \leftarrow r(x) + b_i \cdot a(x)$   // $n$ parallel MACs
4     $a(x) \leftarrow a(x) \cdot x \bmod p(x)$         // via LFSR
5 return $r(x)$
```

In this section we first provide a unified description of the x-net approach to polynomial multiplication, for a generic polynomial modulus $p(x)$. Subsequently, we employ our framework to describe the x-net multiplier design for each one of the four polynomial rings required in Kyber, Saber, NTRU and NTRU Prime. Finally, we describe our unified multiplier architecture.

In the following, we consider the case of the multiplication of two polynomials where the first has coefficients in $\mathbb{Z}_p$, while the second has coefficients in $\mathbb{Z}_q$ and the product has coefficients in $\mathbb{Z}_q$, which is the polynomial multiplication taking place in all the four cryptosystems at hand. The operation is intended to be computed lifting the coefficients of the first polynomial $\mathbb{Z}_p$ simply reconsidering their values as being in $\mathbb{Z}_q$. We note that NTRU also requires a multiplication between two polynomials with coefficients in $\mathbb{Z}_q$: the description in the following also covers this case, simply substituting appropriately sized signals and registers where needed.

The idea underpinning the x-net multiplication is to rewrite the computation of the polynomial ring multiplication $a(x) \cdot b(x) = r(x) \bmod p(x)$, where each polynomial is in the form $p(x) = \sum_{i=0}^{n-1} p_i x^i$, as described in Algorithm 1. In the following, we will assume that $p(x)$ is monic, as it is always the case in practice. The polynomial multiplication is decomposed as a sequence of coefficient-by-polynomial multiplications and polynomial additions (line 3), and multiplications by x and modular reductions (line 4).

The hardware structure of the x-net multipliers, depicted in Figure 1 for the ring $\mathbb{Z}_q[x] / \langle x^n + 1\rangle$ tackles the two operations with two logical component complexes.

The operation $r(x) + b_i \cdot a(x)$ is performed with n independent Multiply and Accumulate (MAC) elements which compute the product of the coefficient b_i by each coefficient of polynomial $a(x)$, and add the result to the corresponding coefficient of $r(x)$. One MAC element is highlighted in grey in Figure 1, and is composed by a multiplier, an adder, a register able to contain a coefficient of the result and a modular reducer $\bmod q$.

The computation of $a(x) \leftarrow a(x) \cdot x$ is efficiently done storing the coefficients of $a(x)$ in a shift register, as the multiplication by x acts shifting the coefficients by one position towards higher degree monomials (to the right, in Figure 1). The computation of the modular reduction of $a(x) \leftarrow a(x) \cdot x \bmod p(x)$ is done efficiently, considering that $a(x) \cdot x$ has degree at most n, and therefore a single polynomial subtraction is sufficient to compute the $\bmod p(x)$ operation. To this end, the portion of the x-net multiplier managing the operation (top portion of Figure 1) subtracts $a_{n-1}p(x)$ from $a(x) \cdot x$, materially adding the coefficients of $(-a_{n-1})p(x)$ to the ones of $a(x) \cdot x$ by performing the addition between any two elements of the shift-register which contains $a(x)$. This network

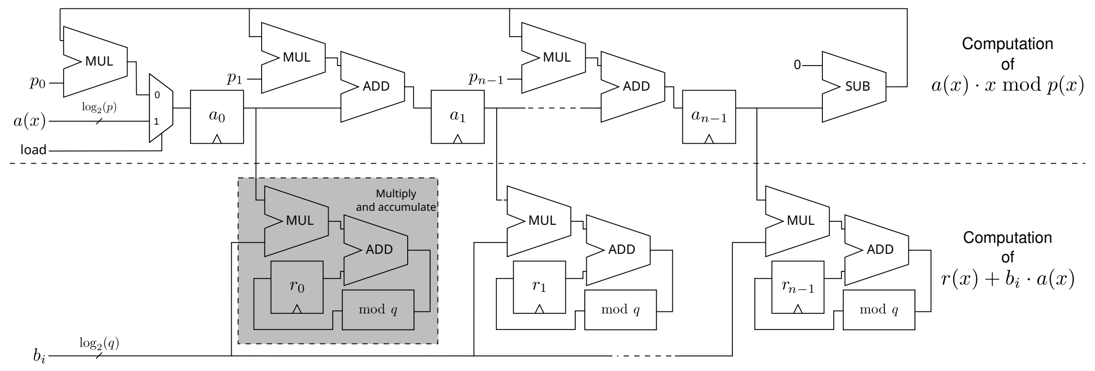

Figure 1: Structure of an x-net multiplier computing the product $r(x) = (a(x) \cdot b(x)) \bmod p(x)$. The top portion of the modular multiplier takes care of computing $x^i \cdot a(x) \bmod p(x)$ at the i-th clock cycle, while the bottom part performs the coefficient-by-polynomial multiplication.

structure will thus need as many multipliers and adder as the number of non-null coefficients in $p(x)$, benefiting from values of $p(x)$ with a very small number of coefficients, as it is the case in the four considered cryptosystems.

x-net Optimizations. The first observation leading to an optimization is that the topmost portion of the x net multiplier may operate entirely with values $\bmod p$, leading to a significant saving in the resource consumption. The lifting required to multiply coefficient in $\mathbb{Z}_p$ by coefficients in $\mathbb{Z}_q$ is done within the multiplier units in the MAC elements simply by sign-extending the two's complement representation of the $\mathbb{Z}_p$ element.

The second observation leading to an optimization is that, in case p is very small, as it is the case in our cryptosystems, the multiplier in the MAC can be substituted by a multiplexer which selects among a small set of fixed multiples of b_i, which are computed by a small number of additions. Taking as an example $p = 5$, the multiplier is substituted by a multiplexer selecting among the values $\{-2b_i, -b_i, 0, b_i, 2b_i\}$ depending on the value of the coefficient of the $a(x)$ polynomial. The values can be either precomputed only once, and distributed, or computed within the MAC unit and selected in place. The former solution trades off area savings for a higher wiring congestion.

A final point concerning the optimization of the x net multiplier is the tradeoff between performing modular reductions in the multiply and accumulate complex managing the coefficients of the result, and performing the reductions upon result readout. The reductions-at-readout approach requires accumulator registers for $r(x)$ of $\lceil \log_2(2npq) + 1 \rceil$ bits, as n values $\bmod q$ will be added by the x net multiplier during its operation. This increase in area is however compensated by the removal of the $\bmod q$ reducers from each multiply and accumulate complex, which typically take more area, unless the reduction by q is trivial (e.g., q is a power of two). We explored both strategies devising modular reducers as follows. In the former case, each accumulator register has $\log_2 q$ bits size, and we perform the $\bmod\ q$ by conditionally applying additions and subtractions. Since the distance between each integer multiplication result and a valid $\mathbb{Z}_q$ element is at most $(q-1) \cdot \lceil (p-1)/2 \rceil$, then $\lceil (p-1)/2 \rceil$ additions and subtractions are carried out in parallel with values multiple of q and the only valid result in $\mathbb{Z}_q$ is selected. In case of the coefficient integer ring reduction operation performed during the readout a single Barrett reduction module is used.

Specialized and Unified x-net Designs. We now describe the specializations which have to be enacted for the design of the x-net multiplier for all the four cryptosystems at hand. For NTRU and Saber, q is a power-of-two, therefore there is no advantage in performing a delayed modular reduction of the coefficients, as it amounts to a simple bit truncation. The feedback network of the computation is very small for all the four cryptosystems. Both Kyber and Saber only require a subtractor to flip the sign of the a_{n-1} coefficient before being fed back to the multipliers and adders; NTRU only requires that a_{n-1} itself is fed back (as $p(x) = x^n - 1$) and NTRU Prime requires feeding back a_{n-1} into two multiply and add elements (as $p(x) = x^n - x - 1$).

Providing a single unified design for all the four cryptosystems was achieved considering the largest among all the register sizes required by the four designs, and inserting multiplexers regulating the kind of coefficient-wise modular reduction being performed, which multiply-add elements are active on the feedback network of the register containing $a(x)$, and whether or not the sign of a_{n-1} should be flipped.

Multiplying in Less than $3n$ Cycles. In our design, we also explored the possibility of reducing the multi-

Table 3: Results of the synthesis targeting an Xilinx UltraScale+ ZCU106 FPGA for every NTRU, NTRU Prime, Saber and Kyber parameter. The results are reported for a design transferring 4 small coefficients per clock cycle and either 1 or 2 large coefficients per clock cycle. Results of each configuration are grouped by security level (from top to bottom: AES-128, AES-192, AES-256 equivalent security, plus three "above AES-256" parameter sets available in NTRU Prime).

Coefficient Reduction done at	**Parameter set**	**Loading 4 small and 1 large coefficient per CC**							**Loading 4 small and 2 large coefficients per CC**						
		CLB	**CC**	**Freq. MHz**	**Latency (μs)**		**AT product**		**CLB**	**CC**	**Freq. MHz**	**Latency (μs)**		**AT product**	
					Enc.	**Dec.**	**Enc.**	**Dec.**				**Enc.**	**Dec.**	**Enc.**	**Dec.**
readout	kyber512	3186	585	328	10.70	14.27	34	45	5460	329	291	6.78	9.04	37	49
readout	sntrup653	8138	1479	288	5.14	15.41	41	125	11867	827	238	3.48	10.43	41	123
readout	ntrulpr653	8138	1479	288	10.27	15.41	83	125	11867	827	238	6.95	10.43	82	123
each CC	kyber512	4226	583	312	11.21	14.95	47	63	6508	327	197	9.96	13.28	64	86
each CC	ntruhps2048509	2150	1153	638	1.81	5.42	3	*11	4455	645	438	1.47	4.42	6	*19
each CC	sntrup653	8411	1477	275	5.37	16.11	45	135	13992	825	200	4.12	12.38	57	173
each CC	ntrulpr653	8411	1477	275	10.74	16.11	90	135	13992	825	200	8.25	12.38	115	173
each CC	lightsaber	2468	583	581	6.02	8.03	14	19	4200	327	375	5.23	6.98	21	29
readout	kyber768	2615	585	312	22.50	28.12	58	73	4733	329	291	13.57	16.96	64	80
readout	sntrup761	9043	1722	312	5.52	16.56	49	149	13762	962	238	4.04	12.13	55	166
readout	ntrulpr761	9043	1722	312	11.04	16.56	99	149	13762	962	238	8.08	12.13	111	166
each CC	kyber768	3202	583	328	21.33	26.66	68	85	5579	327	206	19.05	23.81	106	132
each CC	ntruhps2048677	2825	1531	625	2.45	7.35	6	*20	6208	855	475	1.80	5.40	11	*33
each CC	ntruhrss701	3336	1585	600	2.64	7.93	8	*26	7428	885	425	2.08	6.25	15	*46
each CC	sntrup761	9691	1720	325	5.29	15.88	51	153	16429	960	188	5.11	15.32	83	251
each CC	ntrulpr761	9691	1720	325	10.58	15.88	102	153	16429	960	188	10.21	15.32	167	251
each CC	saber	3019	583	553	12.65	15.81	38	47	4993	327	425	9.23	11.54	46	57
readout	kyber1024	2615	585	312	37.50	45.00	98	117	4733	329	291	22.61	27.13	107	128
readout	sntrup857	10141	1938	312	6.21	18.63	62	188	15404	1082	238	4.55	13.64	70	210
readout	ntrulpr857	10141	1938	312	12.42	18.63	125	188	15404	1082	238	9.09	13.64	140	210
each CC	kyber1024	3202	583	328	35.55	42.66	113	136	5579	327	206	31.75	38.09	177	212
each CC	ntruhps4096821	3712	1855	562	3.30	9.90	12	*36	8052	1035	438	2.36	7.09	19	*57
each CC	sntrup857	11142	1936	312	6.20	18.61	69	207	19034	1080	188	5.74	17.23	109	328
each CC	ntrulpr857	11142	1936	312	12.41	18.61	138	207	19034	1080	188	11.49	17.23	218	328
each CC	firesaber	3245	583	497	23.46	28.15	76	91	5796	327	400	16.35	19.62	94	113
readout	sntrup953	11073	2154	312	6.90	20.71	76	229	18111	1202	238	5.05	15.15	91	274
readout	ntrulpr953	11073	2154	312	13.81	20.71	152	229	18111	1202	238	10.10	15.15	182	274
each CC	sntrup953	12770	2152	312	6.90	20.69	88	264	21165	1200	188	6.38	19.15	135	405
each CC	ntrulpr953	12770	2152	312	13.79	20.69	176	264	21165	1200	188	12.77	19.15	270	405
readout	sntrup1013	12022	2289	312	7.34	22.01	88	264	19201	1277	238	5.37	16.10	103	309
readout	ntrulpr1013	12022	2289	312	14.67	22.01	176	264	19201	1277	238	10.73	16.10	206	309
each CC	sntrup1013	13017	2287	275	8.32	24.95	108	324	22219	1275	188	6.78	20.35	150	452
each CC	ntrulpr1013	13017	2287	275	16.63	24.95	216	324	22219	1275	188	13.56	20.35	301	452
readout	sntrup1277	14735	2883	325	8.87	26.61	130	392	23332	1607	238	6.75	20.26	157	472
readout	ntrulpr1277	14735	2883	325	17.74	26.61	261	392	23332	1607	238	13.51	20.26	315	472
each CC	sntrup1277	16686	2881	262	11.00	32.99	183	550	27283	1605	200	8.03	24.08	218	656
each CC	ntrulpr1277	16686	2881	262	21.99	32.99	366	550	27283	1605	200	16.05	24.08	437	656

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

plication time under $3n$ cycles. Indeed, the described architecture uses n clock cycles to load the $a(x)$ from memory, n cycles to compute the result of the modular multiplication (potentially without coefficient-wise modular reduction), and n cycles to read out the final polynomial multiplication result and store it into the memory. This process can be sped up devising a memory bus transferring multiple polynomial coefficients at once. Transferring α, β and γ coefficients for respectively the small, large and result polynomials, the overall latency of a polynomial multiplication is $\lceil n/\alpha \rceil + \lceil n/\beta \rceil + \lceil n/\gamma \rceil$. Loading α coefficients of $a(x)$ for each clock cycle is achieved transferring them in parallel from main memory, and having the shift register containing a rotate by α positions at each clock cycle through appropriate connections. The same approach is applied for reading out γ coefficients of the result from the accumulator registers, possibly instantiating γ parallel Barrett modules when performing the reductions-at-readout approach. To compute the multiplication of β $\mathbb{Z}_q$ coefficients in parallel, we need a total of $\beta \cdot n$ MACs. Indeed, to compute the result of β multiplication steps, β multiplications and sums need to be computed at each clock cycle, to obtain the result which is to be stored $\beta - 1$ cells to the right of each MAC unit. Furthermore, it is to be noted that β steps of the update of $a(x)$ should be computed in a single step. This in turn requires to perform $\beta - 1$ sign flips of the $\mathbb{Z}_p$ coefficient for specific MAC units of Kyber and Saber, and additional $2(\beta - 1)$ multiply and additions for specific MAC units of NTRU Prime.

Table 4: Comparing results of our x-net and x^2-net with (Farahmand et al., 2019) (a), and (Carter et al., 2022) (b) with multiplier architectures adaptable to multiple cryptoschemes targeting a UltraScale+ target. The count of clock cycles do not consider the time to transfer the operands and the result. No DSP were used.

Work	Par. set	CLB	Freq.	CC	LUT	FF	BRAM
x-net	sntrup761	6757	312	762	38798	21768	2
(a)	sntrup761	9699	255	762	65207	32929	6
x-net	ntruhrss701	3088	588	702	18383	10898	2
(a)	ntruhrss701	5476	300	702	33230	32327	6
x^2-net	ntruhps4096821	7766	412	413	46293	12029	2
(b)	ntruhps4096821	8728	187	412	54478	9227	-
x^2-net	firesaber	5602	338	129	30809	4654	2
(b)	firesaber	3427	310	128	22127	7841	-

Table 5: Results of the synthesis targeting an Xilinx UltraScale+ ZCU106 FPGA compatible with parameter sets up to security level 5. The unified design is configured to transfer 4 small coefficients and 1 large coefficients per clock cycle, and each parameter set can be selected at runtime. Two DSPs and two BRAMs were used by both designs.

Supported Ciphers	CLB	Freq.	LUT	FF	CARRY8
NTRU, NTRU Prime Saber, Kyber	15155	250	92575	27171	3452
NTRU, NTRU Prime Kyber	11318	250	72089	25439	3442

4 EXPERIMENTAL RESULTS

In this section we present the results of our synthesis campaign on specialized x-net multipliers for all the four cryptosystems we considered, and compare them with the current state of the art solutions. Furthermore, we report the figures of merit of our unified multiplier design. The correctness of the results of our multipliers was tested through testbenches obtained with a synthetic computation model written in SageMath. We tested the correctness of the polynomial multiplication for every ring defined by the parameter sets of Kyber, Saber, NTRU and NTRU Prime cryptoschemes.

We conducted our syntheses for the Xilinx UltraScale+ ZCU106 FPGA (target `xczu7ev-ffvc1156-3-e`) using Vivado 2021.1 with FLOW_ALTERNATEROUTABILITY and PERFORMANCE_NETDELAY_HIGH strategies for synthesis and implementation. We explored four different choices of the amount of coefficients being loaded namely loading either one or four small coefficients, and one or two large coefficients. As the configurations loading four small coefficients achieve better performance figures (transferring two large coefficients) and better area-time product (transferring one large coefficient) than their alternatives, we report their results for the sake of brevity. We thus have that the first operand is loaded into the registers 4 coefficients at a time, with a transfer of data from 8 to 16 bits per clock cycle depending on the cryptoscheme and parameter set.

Finally, we report the results of our unified multiplier design, able to support all security levels proposed for standardization to NIST, for all the four ciphers in Table 5. We also report the potential advantage of leaving out Saber, as a means of comparison. Our unified design provides complete runtime flexibility at a cost of 36% more area resources, than the largest tailored component required to run the most demanding cipher it also runs. Furthermore, the achieved running frequency is only 15% slower than the slowest component it encompasses, while taking no penalty on the number of clock cycles taken to compute any of the multiplications with respect to a dedicated design. Removing the support to Saber's polynomial rings, the area penalty drops to $< 2\%$.

The results of our exploration are reported in Table 3, in which we report the resource occupation in Cell Logic Blocks (CLBs) of each multiplier, the number of clock cycles taken for an entire modular multiplication, and the maximum target frequency that the design was able to reach, obtained repeating syntheses with a binary search strategy. Furthermore, we also report the total latency taken by all $\mathcal{R}_p \times \mathcal{R}_q$ multiplications in the key encapsulation (encryption) and key decapsulation (decryption) primitives of the scheme, as some schemes require more than a single multiplication (see Section 2). We evaluated both coefficient ring reduction strategies (at each clock cycle, and upon readout) for NTRU Prime and Kyber to determine which solution is to be preferred when targeting an FPGA design.

By comparing the results among equivalent security level, we can see that the time spent in polynomial multiplications is larger in Kyber (a module RLWE scheme) and Saber (a module RLWR) with respect to NTRU-based schemes. Moreover, this difference increases with the security level. This fact is balanced by the flexibility of Kyber and Saber schemes, which have an almost identical polynomial multiplier usable in every parameter set. As it can be clearly seen, given the large amount of parameter sets for NTRU Prime, the latency of the multiplication for our design and this cryptoscheme is linear in the degree of the polynomials. As a consequence the performance penalty imposed by larger security levels grows more slowly than for Kyber and Saber.

When comparing the coefficient ring reduction techniques, we note that delaying the reduction to the readout yields a massive gain in logic resources utilization and slight increase of working frequency, at the cost of a moderate increase in Flip Flops. The count of CLBs provides a joint indicator of the silicon area employed since combines the number of used memory elements (Flip Flops) and boolean logic implementation resources (Lookup Tables), and confirms that using this coefficient reduction approach decreases substantially the consumed FPGA area. We report in tables the Area-Time (AT) product as an efficiency indicator to compare the designs, and computed as the number of occupied CLBs times the execution time in milliseconds. The gathered data suggests that the x-net architecture is one order of magnitude more efficient when employed to compute polynomial multiplications during encapsulations in NTRU rings. During the decapsulation, this no longer holds, as we recall that one of the three multiplications specified in round 3 submission of NTRU does not have one operand with small coefficients, thus requiring an additional cost (indicated with a $\star$ marker in the table).

Table 4 reports the comparison of our cryptosystem-specialized designs with the existing state of the art on NTRU and NTRU Prime linear time multipliers. We note that our design achieves a 30% to 40% reduction in the required CLBs for both cryptosystems, when comparing our solution which loads a single large coefficient (x-net) with the one in (Farahmand et al., 2019). Furthermore, we also obtain a 28% to 96% gain in working frequency with respect to the same design, therefore achieving also a higher area-time efficiency. We compare our solution loading two large coefficients at once, with the only currently available datapoint in the public technical report (Carter et al., 2022). The solution reported in the technical report, where it is denoted as x^2-net, is 10% larger in area a 2.2× slower in the working frequency for the design for NTRU. These results show how the x-net design is a remarkable fit for the $\mathcal{R}_p \times \mathcal{R}_q$ multiplications in NTRU and NTRU Prime.

5 CONCLUSION

In this work, we analyzed a flexible design for linear-time polynomial multiplications, applicable to accelerate four post-quantum cryptographic primitives: Kyber, Saber, NTRU and NTRU Prime. We reported quantitative results of the efficiency of primitive-tailored designs, obtaining area savings (10%–40%) and significant frequency gains (96%–120%) with respect to the state of the art of NTRU and NTRU Prime multipliers. Our unified design provides the first hardware implementation of a polynomial multiplier able to accelerate the computation of Kyber, Saber, NTRU and NTRU Prime at all security levels in a single component with a 15% frequency reduction, and only a third of a dedicated multiplier in area increase.

REFERENCES

Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., and Liu, Y.-K. (2022). . https://doi.org/10.6028/NIST.IR.8413-upd1.

Basso, A. and Roy, S. S. (2021). Optimized polynomial multiplier architectures for post-quantum KEM saber. In *58th ACM/IEEE Design Automation Conference, DAC 2021, San Francisco, CA, USA, December 5-9, 2021*, pages 1285–1290. IEEE.

Carter, E., He, P., and Xie, J. (2022). High-performance polynomial multiplication hardware accelerators for KEM saber and NTRU. *IACR Cryptol. ePrint Arch.*, page 628.

Dang, V. B., Mohajerani, K., and Gaj, K. (2021). High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber. *IACR Cryptol. ePrint Arch.*, page 1508.

Farahmand, F., Dang, V. B., Nguyen, D. T., and Gaj, K. (2019). Evaluating the potential for hardware acceleration of four ntru-based key encapsulation mechanisms using software/hardware codesign. In Ding, J. and Steinwandt, R., editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 23–43. Springer.

Karatsuba, A. (1963). Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596.

Liu, B. and Wu, H. (2015). Efficient architecture and implementation for ntruencrypt system. In *IEEE 58th International Midwest Symposium on Circuits and Systems, MWSCAS 2015, Fort Collins, CO, USA, August 2-5, 2015*, pages 1–4. IEEE.

Marotzke, A. (2020). A constant time full hardware implementation of streamlined NTRU prime. In Liardet, P. and Mentens, N., editors, *Smart Card Research and Advanced Applications - 19th International Conference, CARDIS 2020, Virtual Event, November 18-19, 2020, Revised Selected Papers*, volume 12609 of *Lecture Notes in Computer Science*, pages 3–17. Springer.

NIST PQC Team (2022). PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4.

Peng, B., Marotzke, A., Tsai, M., Yang, B., and Chen, H. (2021). Streamlined NTRU prime on FPGA. *IACR Cryptol. ePrint Arch.*, page 1444.

Sklavos, N., Chaves, R., di Natale, G., and Regazzoni, F. (2017). *Hardware Security and Trust: Design and Deployment of Integrated Circuits in a Threatened Environment*. Springer Publishing Company, Incorporated, 1st edition.