IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received September 7, 2021, accepted October 29, 2021, date of publication November 8, 2021,
date of current version November 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126208

A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms

PIETRO NANNIPIERI, (Member, IEEE), STEFANO DI MATTEO, LUCA ZULBERTI, (Student Member, IEEE), FRANCESCO ALBICOCCHI, SERGIO SAPONARA, (Senior Member, IEEE), AND LUCA FANUCCI, (Fellow, IEEE)
Department of Information Engineering, University of Pisa, 56122 Pisa, Italy

Corresponding author: Pietro Nannipieri (pietro.nannipieri@ing.unipi.it)

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (European Processor Initiative) under Agreement 826647.

ABSTRACT In recent years, public-key cryptography has become a fundamental component of digital infrastructures. Such a scenario has to face a new and increasing threat, represented by quantum computers. It is well known that quantum computers in the next years will be able to run algorithms capable of breaking the security of currently widespread cryptographic schemes used for public-key cryptography. Post-quantum cryptography aims to define and execute algorithms on classical computer architectures, able to withstand attacks from quantum computers. The National Institute of Standards and Technology is currently running a selection process to define one or more quantum-resistant public-key algorithms and lattice-based cryptographic constructions are considered one of the leading candidates. However, such algorithms require non-negligible computational resources to be executed. One viable solution is to accelerate them totally or partially in hardware, to alleviate the workload of the main processing unit. In this paper, we investigate a solution trading-off performance and complexity to execute the lattice-based algorithms CRYSTALS-Kyber and -Dilithium: we introduce a dedicated Post-Quantum Arithmetic Logic Unit, embedded directly in the pipeline of a RISC-V processor. This results in an almost negligible area overhead with a large impact on the algorithms speed-up and a consistent reduction in the energy required per single operation.

INDEX TERMS Lattice based cryptography, crystals, kyber, dilithium, FPGA, post quantum, security, hardware acceleration, RISC-V.

I. INTRODUCTION

In the last years, a digitalization process is going on in many different areas like industry 4.0, automotive, and healthcare. This fact leads to more and more complex Systems-on-Chips (SoC), requiring a continuous internet connection to the cloud that has to be supported efficiently, especially in mobile systems. Such systems will communicate over an intrinsic insecure channel such as the public 5G infrastructure, therefore secure communication is an essential requirement for all these domains of application [1]. In State of the Art (SoA) systems, the security of the connections relies on the Public Key Cryptography (PKC) which employs a pair of keys, public and private. PKC algorithms are based on hard mathematical problems that are considered infeasible to solve, i.e. it would take far too many resources and time to compute the solution and break the system. However, the advent of quantum computers will strongly compromise the security of these algorithms, since such elaborators will be able to solve the problems in polynomial time using Shor's algorithm [2]. For this reason, in 2017 the National Institute of Standards and Technology (NIST) started a standardization process, which is now at its third round, to find one or more quantum-resistant public-key cryptographic algorithms [3].

Post-Quantum Cryptography (PQC) exploits mathematical elements and operations which are usually not straightforward to implement on standard processors. This is a critical aspect especially in low-power embedded devices that have a limited amount of resources and computational power. Consequently, there is increasing interest regarding hardware

The associate editor coordinating the review of this manuscript and approving it for publication was Ilsun You.

acceleration of PQC [4]. There are several options available in the design space to address this problem.

The most optimised approach is obviously to design and build an Application Specific Integrated Circuit (ASIC) to accelerate the requested algorithm [5]. This solution can reach the best results in terms of performance and power/energy consumption, but it requires a considerable design effort and relative non-recurrent costs, which makes it often an undesired solution, especially for accelerating algorithms that are currently under development or evaluation. A classical approach is to design and embed hardware accelerators connected to the control and elaboration unit as memory-mapped peripherals [6]. Another option is to bring smaller hardware accelerators directly in the processor pipeline [7]. With this approach, the area increment is lower and almost negligible for larger cores. Of course, the timing performance are lower than full algorithm acceleration since the CPU can manage only 32/64-bit operands in its execution stage therefore only a subset of functions can be accelerated. The in-pipeline acceleration approach is applicable for open Instruction Set Architecture (ISA) processors, which can be extended to compute the accelerated functions. A RISC-V processor is perfect for this scope [8] since it is a free and open ISA that provides a set of reserved opcodes specifically created to promote more specialized instruction-set extensions.

Among all the possible post-quantum algorithms that are being developed nowadays and that are currently competing in the NIST standardisation process, Lattice-Based Cryptography (LBC) [9] offers a very good trade-off between security and efficiency. In this work, we propose a first ISA extension to the CVA6 [10] processor, applicable also to other processors of the RISC-V family, to accelerate the execution of the CRYSTALS-Kyber and -Dilithium algorithms, respectively a Key Encapsulation Mechanism (KEM) and a Digital Signature Scheme (DSS). Currently, the RISC-V community is putting a lot of effort to standardise RISC-V cryptographic extensions. At the time of writing the proposed standard is in public review [11]. Unfortunately, the policy is to support only existing standardised cryptographic constructs. Candidate protocols for future standardisation are not currently taken into account, even if it is mentioned that the standard will also deal in the future with the NIST Post-Quantum Cryptography contest. Our proposed work, therefore, aims at paving the way to such future extensions, in case our analysed and accelerated algorithms will be selected by the NIST as future standards.

Number Theoretic Transform (NTT) [12] represents one of the most onerous parts of Kyber and Dilithium algorithms. NTT is a specialized form of Discrete Fourier transform (DFT) for finite fields and is widely adopted to perform polynomial multiplication with large operands [13]. As already indicated in [14], this is one of the bottlenecks of such algorithms, hence it will be investigated in this work for its potential hardware acceleration. The contributions of this work include, but are not limited to:

- Detailed algorithms study to demonstrate which functions are worthy to be hardware accelerated.
- First post-quantum ISA cryptographic extension embedded in the 64-bit CVA6 RISC-V processor.
- Introduction of new hardware functionalities directly mapped to assembly instructions to reduce the execution time of the CRYSTALS suite.
- Evaluation of the Post-Quantum (PQ) ISA extension in terms of power/energy consumption on FPGA technology.

The rest of the paper is organised as follows: Section II provides an overview of LBC and CRYSTALS suite. In Section III the algorithms are analysed to identify the bottlenecks in the RISC-V CVA6 core, and the architecture of the implemented instructions is described. In Section IV, once the achieved performance is considered satisfying, the system has been implemented in real hardware and tested against reference implementation to measure time acceleration, energy efficiency improvement, and resource consumption impact. The obtained results are discussed and compared with solutions available in the SoA in Section V. Finally, in Section VI we conclude this work, highlighting the innovative results achieved and the possible improvement for future research.

II. KYBER AND DILITHIUM OVERVIEW

In the NIST PQC standardization process, seven algorithms have reached the third round as finalists, and five of them are lattice-based. Different mathematical problems can be used to construct cryptography schemes based on lattices, and the most known is the Learning With Errors (LWE) problem. LWE involves the extraction of vector $\mathbf{s}$ from the equation $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$, where $\mathbf{A}$ is a matrix, $\mathbf{t}, \mathbf{s}$ and $\mathbf{e}$ are vectors, and the vector $\mathbf{e}$ must be sampled from specific small error distributions.

CRYSTALS is a CRYptographic SuiTe for Algebraic LatticeS (CRYSTALS) that encompasses Kyber and Dilithium algorithms, which base their security on the hardness of solving the LWE problem in module lattices (MLWE problem [15]). Compared to standard LWE, matrices in MLWE have smaller dimensions and the coefficients are polynomials in R_q.

A. CRYSTALS-KYBER

Kyber is one of the four candidates that have been selected as third-round finalists for KEM of PQC NIST competition, together with Classic McEliece [16], NTRU [17] and SABER [18]. The construction of Kyber follows two steps: first, it encrypts 32-bytes messages following the conventional method to construct INDistinguishability under Chosen-Plaintext Attack (IND-CPA) secure public-key encryption scheme; then, a tweaked Fujisaki–Okamoto (FO) transform [19] is used to build the INDistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) secure KEM.

TABLE 1. Parameter sets for crystals-kyber.

NIST Security Level	n	k	q	η_1,η_2	pk	sk	ct
1	256	2	3329	3, 2	800	1632	768
3	256	3	3329	2, 2	1184	2400	1088
5	256	4	3329	2, 2	1568	3168	1568

TABLE 2. Parameter sets for crystals-dilithium.

NIST Security Level	n	k, ℓ	q	η	γ_1	pk	sign
2	256	(4,4)	8380417	2	2^{17}	1312	2420
3	256	(6,5)	8380417	2	2^{19}	1952	3293
5	256	(8,7)	8380417	2	2^{19}	2592	4595

Kyber works on rings of integer polynomials modulo prime q, which are denoted as $Z_q[X]$. Polynomials modulo both q and $X^n + 1$ compose the ring $R_q = Z_q[X]/(X^n + 1)$. Bold lower-case letters represent vectors and bold upper-case letters are matrices with coefficients in R_q. Noise polynomials in Kyber are sampled from the centred binomial distribution B_η where η is directly related with the range of noise samples.

The main functions to construct the IND-CPA-secure public-key infrastructure in Kyber are key generation, encryption, and decryption. Detailed information about such algorithms can be found on the official documentation of Kyber [20]. The parameter sets of Kyber are reported in Table 1. The parameter n is set to 256 to encapsulate keys with 256 bits of entropy, q is a small prime that allows a fast NTT-based multiplication. The parameter k fixes the lattice dimension and allows scaling security and efficiency to different levels. The parameter η_1 defines the noise of vectors $\mathbf{s}$ and $\mathbf{e}$ in key generation function and of $\mathbf{r}$ in encryption function, while η_2 defines the noise of $\mathbf{e_1}$ and e_2 in encryption function. The columns pk, sk, and ct indicate respectively the size in bytes of the public key, the secret key, and the cyphertext.

Kyber algorithms involve several cryptographic primitives; key generation function requires seed expansion through the SHA3-512 HASH function, the matrix $\hat{\mathbf{A}} \in R_q^{k\times k}$ generation through the eXtendable-Output Function (XOF) SHAKE-128 and rejection sampling to generate elements in R_q that are statistically close to a uniformly random distribution. The noise terms $\mathbf{e}$ and $\mathbf{s}$ are sampled from the centred binomial distribution B_η which requires a Pseudo-Random Function (PRF) implemented in Kyber with SHAKE-256 XOF. In encryption function, matrix generation (i.e. $\hat{\mathbf{A}}$) and vectors sampling (i.e. $\mathbf{r}$, $\mathbf{e_1}$ and e_2) require the same primitives of the key generation function. NTT is adopted for polynomial multiplications. In addition, several auxiliary functions, such as keys encoding (and decoding) to serialize (and deserialize) polynomials in byte arrays (and vice versa), and compression (and decompression) functions, are used to transform elements $\in Z_q$ to integer less than $log_2(q)$ (and vice versa).

B. CRYSTALS-DILITHIUM

Crystals-Dilithium is one of the three digital signature schemes selected in the third round of the NIST PQC competition with Falcon [21] and Rainbow [22]. The mathematical notation reported in Section II-A for Kyber is still valid also for the Dilithium algorithm. The parameter sets of Dilithium are reported in Table 2. Its detailed explanation can be found in the official documentation [23]. In Dilithium $\mathbf{A}$ is a $k \times \ell$ polynomial matrix, while $\mathbf{s}$ and $\mathbf{e}$ become ℓ-dimensional and k-dimensional polynomial vectors, named respectively $\mathbf{s_1}$ and $\mathbf{s_2}$. In Dilithium n and q are fixed, while the dimension of k and ℓ impacts on the security level and performance. The parameter η indicates the maximum size of $\mathbf{s_1}$ and $\mathbf{s_2}$ coefficients. Parameter γ_1 limits the coefficients of the polynomial vector $\mathbf{y}$, which is used as masking vector in encryption function. Dilithium is composed of three main functions: key generation, signature generation, and signature verification. The two main operations that constitute such functions are XOFs and multiplications in the polynomial ring $R_q = Z_q[X]/(X^n + 1)$. The generation of the matrix $\mathbf{A}$ and of $\mathbf{s_1}$ and $\mathbf{s_2}$ adopts SHAKE-128 and SHAKE-256 as XOF. The computation of the public key $\mathbf{t} = \mathbf{As_1} + \mathbf{s_2}$ is performed over R_q. NTT is adopted for multiplications. In the signing procedure, the message to be signed is hashed using the SHAKE-256 as Collision Resistant Hash (CRH).

C. NTT AND MODULAR REDUCTIONS

In Kyber and Dilithium algorithms, polynomial multiplication represents one of the most critical and time-consuming operations. NTT is a special case of DFT which is conducted in the finite field Z_q rather than in complex field C. NTT can be adopted to speed-up polynomial multiplication, reducing the complexity of multiplying two n-terms polynomials from $O(n^2)$ to $O(nlog(n))$. NTT transformation is denoted as:

$$NTT(a) = \sum_{i=0}^{n-1} \breve{a}_i x^i, \quad where\ \breve{a}_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \tag{1}$$

where ω is the n-th primitive root of unity. Since the product of two n-terms polynomials has $2n$ coefficients and should be reduced modulo (X^n+1), the Negative Wrapped Convolution (NWC) [24] procedure can be used to remove this overhead. NWC involves the pre-scale and post-scale of the polynomials with the square root ξ of the n-th primitive root of unity. Pre-scale is the multiplication between the coefficients of the input polynomials and ξ, while the post-scale requires the multiplication between the output polynomial coefficients and ξ^{-1}. Equation 2 refers to polynomial multiplication with the NTT technique. The symbol $\circ$ indicates Point-Wise Multiplication (PWM). NTT computation is typically executed through butterfly operations in Cooley-Tukey (CT) or Gentleman-Sande (GS) configurations [25]. NTT in Kyber is slightly different from the classic one because field Z_q does not contain the $2n-th$ primitive root of unity, and the modulo

Algorithm 1 Montgomery Reduction for Crystals-Kyber

```
Input: 32-bit integer $a$, $q_{inv} = q^{-1}$ $mod$ $2^{16}$
Output: 16-bit integer $t$ congruent to $aR^{-1}$ $mod$ $q$, where $R = 2^{16}$
1: $u = aq_{inv}$
2: $t = uq$
3: $t = a - t$
4: $t = t \gg 16$
5: return $t$
```

Algorithm 2 Barret Reduction for Crystals-Kyber

```
Input: 16-bit integer $a$, constant $v = ((1 \ll 26) + q/2)/q$
Output: 16-bit integer $t$ congruent to $a$ $mod$ $q$
1: $t = va \gg 26$
2: $t = tq$
3: $t = a - t$
4: return $t$
```

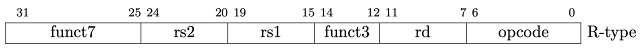


FIGURE 1. RISC-V R-type format instruction.

$(X^n + 1)$ cannot be fully factored into n $degree1$ polynomials but $n/2$ $degree2$ [26], [27]. This means that one 256-term NTT process can be conducted by two separate 128-term, and the PWM requires five Z_q multiplications instead of only one.

$$c = (INTT(NTT(a) \circ NTT(b))) \qquad (2)$$

Butterfly computations and PWM are executed modulo q employing Montgomery [28] and Barret [29] reductions in Kyber (reported respectively in Algorithms 1 and 2). Dilithium adopts Montgomery and a special reduction named *reduce32* (reported respectively in Algorithms 3 and 4).

III. EXTENDED ISA DEFINITION AND IMPLEMENTATION

RISC-V ISA provides four basic instruction formats named R-type, I-type, S-type, and U-type. The R-type format uses two registers as sources and puts the output to a single destination register, I-type format replaces one source register with one immediate, S-type format replaces the destination register with one immediate, and U-type format has no source operands but a larger immediate. The RISC-V ISA has reserved a portion of the encoding space for custom extensions using specific values for the *opcode* field (i.e. bits 6:0) of the instruction, like in the R-type format reported in Figure 1. In the next sections, we will explain our RISC-V custom implementation for CRYSTALS algorithms.

A. ALGORITHMS ANALYSIS ON RISC-V IMPLEMENTATION

The choice of operations to accelerate is based on the analysis of the Kyber and Dilithium algorithms running on the RISC-V CVA6 core. As reported in Section II, the most onerous parts of Kyber and Dilithium algorithms are related

Algorithm 3 Montgomery Reduction for Crystals-Dilithium

```
Input: 64-bit integer $a$ such that $-2^{31}q \le a \le q2^{31}$, $q_{inv} = q^{-1}$ $mod$ $2^{32}$
Output: 32-bit integer $r$ such that $r = a2^{-32}$ $mod$ $q$
1: $r = aq_{inv}$
2: $r = (a - (rq)) \gg 32$
3: return $r$
```

Algorithm 4 Modular Reduction (*Reduce32*) for Crystals-Dilithium

```
Input: 32-bit integer $a$ such that $a \le 2^{31} - 2^{22} - 1$
Output: 32-bit integer $t = a$ $mod$ $q$
1: $t = (a + (1 \ll 22)) \gg 23$
2: $t = a - tq$
3: return $t$
```

to the computation of polynomial multiplication, XOF, and CRH functions. To confirm this assumption, we firstly analysed the contribution of such operations on the three main functions composing Kyber (i.e. key generation, encryption, and decryption) and Dilithium (i.e. key generation, signature generation, and signature verification). In this analysis we are focusing on the indcpa_keypair, indcpa_enc, and indcpa_dec for Kyber and crypto_sign_keypair, crypto_sign and crypto_sign_verify for Dilithium of the reference codes. The goal of this evaluation process is to identify different primitives that can be easily integrated inside the execution stage of the target RISC-V processor. The results of this preliminary analysis are reported in Tables 3 and 4 respectively for Kyber and Dilithium algorithms. Column Funct. reports the functions we are evaluating, column Op. indicates the main subfunctions, columns OP1 and OP2 are the lengths of the input operands and OP3 is the length of the output one. Column % Funct. indicates the contribution in terms of clock cycles taken by the sub-function (i.e. column Op.) over the main one (i.e. column Funct). The reported Keccak sub-function refers to the KeccakF1600_StatePermute function in the code, which is used to compute XOF and CRH operations in the algorithms, while the basemul sub-function refers to the PWM. The presented results refer to the lowest security level of both algorithms measuring the average number of clock cycles based on 10′000 repetitions. Each polynomial has 256 16-bits coefficients in Kyber and 256 32-bits coefficients in Dilithium, thus the operand lengths of the sub-functions are 4096 and 8192 bits when polynomials are involved. In the case of ntt/invntt sub-functions, the OP2 operand refers to the twiddle factors (128 16-bit or 32-bit constants). Aside from Keccak permutation, the other sub-functions are composed mainly by modular multiplications and reductions that can be accelerated by scalar instructions within the RISC-V CPU. In particular, we selected the *fqmul* (multiplication followed by *Montgomery reduction*) and *barret reduction* operations for Kyber algorithm and the *fqmul* and *reduce32* operations for Dilithium one. In Kyber, the *fqmul* and *barret reduction*

TABLE 3. Operands length and clock cycles percentage on CVA6 of the main functions of CRYSTALS-Kyber algorithm.

Funct.	Op.	OP1	OP2	OP3	% Funct.
keypair	ntt	4096	2048	4096	30.79
	basemul	4096	4096	4096	14.76
	poly_add	4096	4096	4096	0.58
	Keccak	1600	–	1600	32.23
enc	ntt	4096	2048	4096	12.30
	basemul	4096	4096	4096	18.10
	poly_add	4096	4096	4096	0.82
	invntt	4096	2048	4096	9.39
	Keccak	1600	–	1600	23.29
dec	ntt	4096	2048	4096	37.29
	basemul	4096	4096	4096	18.29
	poly_add/_sub	4096	4096	4096	0.73
	invntt	4096	2048	4096	28.48
	Keccak	1600	–	1600	0

TABLE 4. Operands length and clock cycles percentage on CVA6 of the main functions of CRYSTALS-Dilithium algorithm.

Funct.	Op.	OP1	OP2	OP3	% Funct.
keypair	ntt	8192	4096	8192	8.13
	basemul	8192	8192	8192	6.86
	poly_add	8192	8192	8192	2.25
	invntt	8192	8192	4096	9.88
	Keccak	1600	–	1600	53.07
sign_gen	ntt	8192	4096	8192	15.96
	basemul	8192	8192	8192	10.04
	poly_add	8192	8192	8192	2.43
	invntt	8192	8192	4096	27.62
	Keccak	1600	–	1600	13.13
sign_ver	ntt	8192	4096	8192	16.44
	basemul	8192	8192	8192	7.70
	poly_add/_sub	8192	8192	8192	2.03
	invntt	8192	8192	4096	8.88
	Keccak	1600	–	1600	47.70

operations take 31 and 22 cycles respectively, contributing for $51-82\%$ of the three high-level *indcpa* functions. For Dilithium algorithm, the *fqmul* and *reduce32* operations take 26 and 19 cycles respectively, contributing for $11-27\%$ of the high-level functions. We remark that the purpose of this work is to investigate the acceleration of polynomial operations even though we know that a considerable contribution to the execution time of the CRYSTALS suite is also due to the Keccak function. The integration of a Keccak hardware accelerator inside the pipeline of the processor would be very costly since it requires several bit manipulation operations and storage resources.

The implemented extension has been inserted into the *custom-0* RISC-V encoding space (opcode = 7'b0001011), using the R-type format for all instructions to simplify the decode stage. `func3` field is used to distinguish between Kyber and Dilithium, and `funct7` field to individuate the

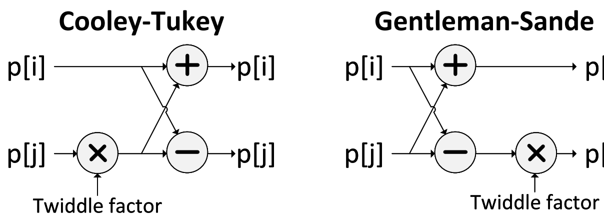


FIGURE 2. CT and GS butterflies configurations.

TABLE 5. RISC-V post-quantum instructions encoding for crystals-kyber and -dilithium.

funct7	func3		arguments
	0	1	
0	`fqmul_k`	`fqmul_d`	`rd, rs1, rs2`
1	`reduce_k`	`reduce_d`	`rd, rs1`
2	`set_twiddle_k`	`set_twiddle_d`	`rs1`
3	`ntt_k`	`ntt_d`	`rd, rs1, rs2`
4	`intt_k`	`intt_d`	`rd, rs1, rs2`

particular instruction. The *fqmul* and *reduce* operations can be straight-forward implemented since they have respectively two and one input operands.

As reported in Section II, NTT and INTT are employed for polynomial multiplication in both Kyber and Dilithium algorithms. Forward NTT requires the CT-butterfly configuration, while inverse NTT employs the GS one. Figure 2 depicts both CT and GS configurations. The integration of the butterfly unit inside the RISC-V pipeline as R-type instructions has two main blocking factors: the butterfly unit requires three input operands, i.e. two 16-bit (or 32-bit) polynomial coefficients and one 16-bit (or 32-bit) twiddle factor, and two 16-bit (or 32-bit) polynomial coefficients as result of the computation. To overcome the first limitation, we exploited the fact that the twiddle factors are compile-time constants, therefore they can be saved in an internal LUT of the PQ ALU. We introduced an appropriate instruction (`set_twiddle_k/_d`), called before the actual butterfly one (`[i]ntt_k/_d`), to select the twiddle factor. The second limitation can be overcome by packing the two 16-bit (or 32-bit) results into a single 64-bit destination register. This solution is valid only on 64-bit architectures for the Dilithium algorithm but can be used in 32-bit architectures to accelerate Kyber ones.

Finally, as result of this work, we have implemented the ten instructions listed in Table 5, five per algorithms, which accelerate the selected operations. In particular, `reduce_k` computes the Barrett reduction for Kyber, `reduce_d` computes the *reduce32* reduction for Dilithium, `ntt_k/_d` perform the CT-butterfly, and `intt_k/_d` perform the GS one.

B. PQ-ALU ARCHITECTURE

We designed two different PQ ALU modules to implement the PQ ISA extension listed in Table 5. Figure 3

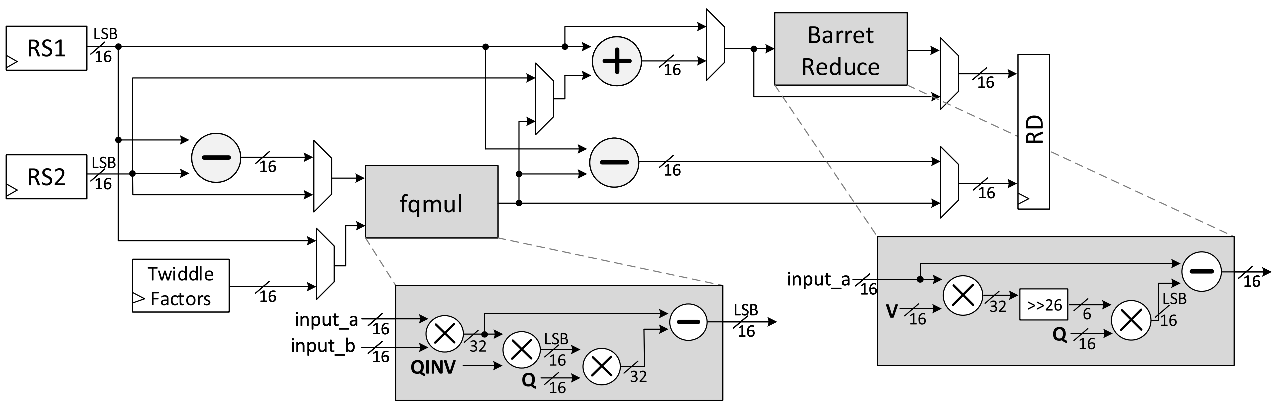


FIGURE 3. PQ-ALU architecture for CRYSTALS-Kyber algorithm. Q = 3329, QINV = 62209, and V = 20159.

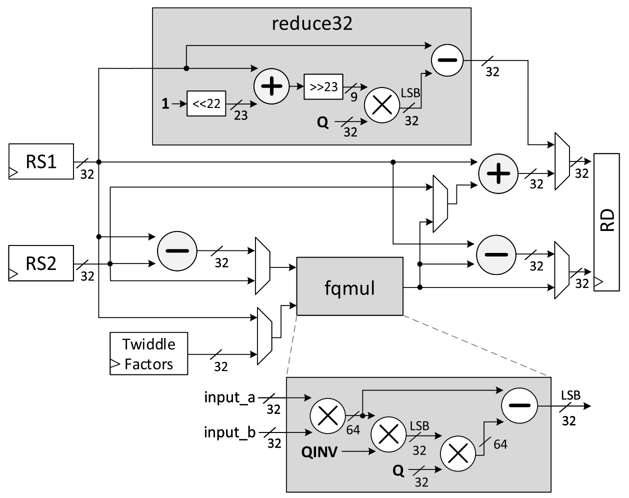


FIGURE 4. PQ-ALU architecture for CRYSTALS-Dilithium algorithm. Q = 8380417, and QINV = 58728449.

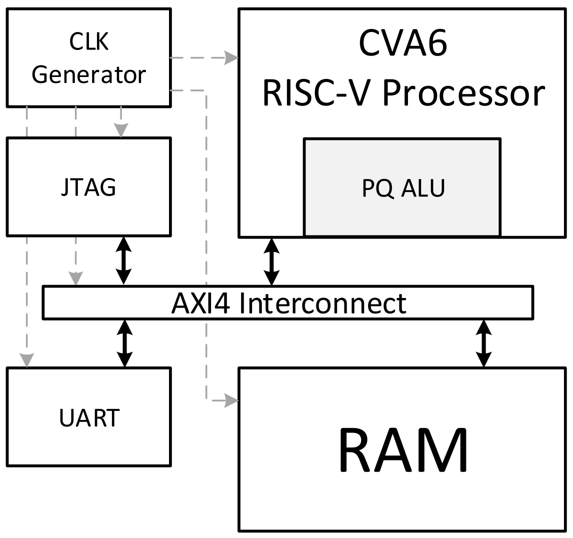


FIGURE 5. FPGA test setup.

shows the hardware design for Kyber acceleration, while Figure 4 shows the design for Dilithium. To limit the hardware resource utilisation, both implementations share the hardware modules for *fqmul* and reduction instructions (i.e. Barrett for Kyber and *reduce32* for Dilithium) to perform the CT and GS butterfly operations. Multiplexers are used to modify the datapath and implement both butterfly configurations. The `set_twiddle_k/_d` instructions change the address of the internal LUT to provide the requested twiddle factor.

All instructions are encoded with R-type format, but reduction instructions do not use *Rs2* and `set_twiddle_k/_d` use only *Rs1*. The unused registers can be set to `x0` in source code to prevent the CPU from performing unnecessary register renaming and/or pipeline stalling due to dependency check between instructions. The output of each instruction is sign-extended to fill the 64-bit register, apart from the `ntt_k/_d` instructions whose output pair must be taken from the packed destination register. This overhead requires just one shift operation to extract the result stored in the most significant part of the 64-bit register.

IV. FPGA DEMONSTRATOR

A. FPGA DEMONSTRATOR TEST SETUP

In order to test and prototype the system, we implemented a basic SoC on the Xilinx ZCU106 evaluation board [30]. As we can see in Figure 5, the SoC is composed by:

- The CVA6 core, extended with our proposed PQ ALU;
- The RAM memory (external DDR4);
- The UART used to display the debug information;
- The JTAG used to program the device with compiled software directly written in the RAM;
- The clock generator;

The CVA6 core includes 32KB and 16KB of cache memory for respectively data and instructions and does not integrate any Floating Point Unit (FPU). We used performance-oriented strategies both for synthesis and implementation on Xilinx Vivado 2020.2, with an operational clock frequency of 100MHz. It is remarkable that the critical path does not include our proposed PQ ALU therefore our ISA extension does not affect the timing performance of the system.

TABLE 6. CVA6 and PQ_CVA6 resource comparison on Xilinx ZCU106 board.

	CVA6				PQ_CVA6 Kyber				PQ_CVA6 Dilithium			
	LUT	FF	DSP	BRAM	LUT	FF	DSP	BRAM	LUT	FF	DSP	BRAM
SoC	61349	60278	19	77	64522	60336	24	77.5	64855	60349	29	77.5
CVA6_noFPU	31670	19076	16	37	33953	19135	21	37.5	34279	19144	26	37.5
ex_stage	10632	6090	16	0	10887	6088	21	0.5	11153	6091	26	0.5
PQ ALU	-	-	-	-	178	0	5	0.5	377	0	10	0.5
issue_stage	9924	4596	0	0	11237	4622	0	0	11298	4630	0	0

B. RESOURCE UTILISATION ANALYSIS

For what concerns the complexity, in the following we briefly report a resource analysis of the extended CVA6. These results have been obtained with the *Flow_PerfOptimized_high* strategy with retiming for the synthesis and the *Performance_ExtraTimingOpt* strategy for the implementation, and with the keep hierarchy attribute on the PQ ALU. Table 6 sums up the utilisation reports.

We can observe that the PQ ALU has a minimal impact on resource utilisation. In fact, it employs only 178 LUTs for Kyber and 377 for Dilithium, no FF, 5 DSP in Kyber and 10 in Dilithium, and only 0.5 BRAM. This means that the area impact of the PQ ALU is almost negligible compared to the CVA6. This result was predictable since our PQ ALU mainly performs arithmetic operations such as multiplications, which in FPGAs are mapped on DSPs, if available.

The PQ ALU is integrated into the CVA6 core as a fixed latency Functional Unit (FU) and its control requires further logic which increases the resource usage. Even if it is not the aim of this paper to describe in detail the architecture of the CVA6 processor (please refer to [10] for further information), in the following paragraph we briefly analyse the area increment due to the control logic.

The largest increment is represented by the *issue stage* of the core which dispatches instructions to the FUs and keeps track of them in a scoreboard like a data structure. This part of the CVA6 processor is composed of the *issue_read_operands* module and the just mentioned *scoreboard*. The latter is a FIFO that keeps track of all decoded, issued, and committed instructions, together with all the registers that are involved during the execution of those instructions. The *issue_read_operands* issues the instructions from the scoreboard and fetches the operands, also handling the forwarding logic in order to execute two instructions back to back (i.e. with no bubble in between).
We introduced a new FU in the execution stage, thus it is obvious that more logic is required to dispatch the instructions to the available FUs, to fetch/forward the operands to other stages of the pipeline, and to decode the instructions themselves.

C. POWER CONSUMPTION

The energy necessary to carry out a specific function is a very important metric, especially if the system is intended to be employed in edge devices where power consumption is a major constraint. Since the resource overhead introduced with our implementation is almost negligible, we expected the instantaneous power consumption to be almost invariant, and the energy necessary to perform a specific function to be reduced by the same percentage of the saved clock cycles to carry out the operation. We performed a specific set of tests where we measured the total power consumption of the FPGA and knowing in advance its static power consumption, we have been able to identify the dynamic one, during the execution of all the six analysed functions, both with the regular CVA6 processor and with the one embedding our PQ ALU. The three main functions for Kyber are: *indcpa_keypair*, *indcpa_enc*, and *indcpa_dec* [26]. While for Dilithium: *crypto_sign_keypair*, *crypto_sign_signature*, and *crypto_sign_verify* [31].

Static power consumption measures have been taken by the implementation results of Vivado, while the total one have been measured applying methods described in [32]: data were collected using the MaxPowerTool instrument from Maxim Integrated for the ZCU106 board. The *VCCINT* power rail has been taken into account to measure the power consumption of the programmable logic, while the *VCCBRAM*, *VCCAUX*, and *VCC1V2* are taken into account for other components of the FPGA and board (e.g. the Block RAM, the I/O, and the DDR memory).

Our aim here is to measure the power consumption on the different configurations under the same environmental condition and to calculate the relative energy saving. Table 7 shows the collected results of the lowest level of security for both algorithms; other security levels follow the same principles. We observe that the dynamic power consumption is slightly lower (less than 248 mW difference) on the PQ CVA6 functions. The difference is probably due to the fact that, even if more hardware is active within the FPGA when the PQ_ALU is added, the overall switching activity is lower, due to the fact that the processor operates more efficiently performing less memory accesses, resulting in an overall small reduction of the dissipated power. Minor influences could also be played by randomness in the place and route algorithms output. What should be taken into account is that the overall power consumption is almost the same, but in the PQ CVA6 the functions require much fewer clock cycles to be executed: the energy necessary to perform each function is reduced by the same factor.

TABLE 7. PQ ALU impact on power performance.

			P Static [W]	P Tot. [W]	P Dyn. [W]	Clock Cycles	Energy [mJ]	Energy Saving
CVA6	Kyber	*Keypair*	0.125	2.788	2.663	654104	18.23	-
		Encode	0.125	2.717	2.592	818886	22.25	-
		Decode	0.125	2.860	2.735	270080	7.723	-
	Dilithium	*Keypair*	0.125	2.860	2.735	1873016	53.56	-
		Sign	0.125	2.824	2.699	8195317	231.4	-
		Verify	0.125	2.896	2.771	2083853	60.34	-
PQ CVA6	Kyber	*Keypair*	0.125	2.612	2.487	419597	10.96	39.9%
		Encode	0.125	2.717	2.592	438280	11.91	46.5%
		Decode	0.125	2.612	2.487	100796	2.632	65.9%
	Dilithium	*Keypair*	0.125	2.681	2.556	1592325	42.69	20.3%
		Sign	0.125	2.752	2.627	5884266	161.9	30.0%
		Verify	0.125	2.717	2.592	1700679	46.21	23.4%

TABLE 8. Kyber functions analysis with our proposed PQ ALU.

Function	**Security Level**	**Standard Execution**		**fqmul + Barrett reduce**			**fqmul + Barrett reduce + butterfly**		
		C.c.	Time [ms]	C.c.	Time [ms]	Speed-up	C.c.	Time [ms]	Speed-up
NTT	1-3-5	50355	0.5036	23742	0.2374	2.12	18488	0.1849	2.72
INTT	1-3-5	76912	0.7691	25664	0.2566	3.00	18488	0.1849	4.16
kem_keypair	1	654104	6.541	440688	4.407	1.48	419597	4.196	1.56
kem_enc		818886	8.189	470395	4.704	1.74	438280	4.383	1.87
kem_dec		270080	2.701	118477	1.185	2.28	100796	1.008	2.68
kem_keypair	3	1103604	11.04	725629	7.256	1.52	694504	6.945	1.59
kem_enc		1327149	13.27	775664	7.757	1.71	731597	7.316	1.81
kem_dec		350815	3.509	153277	1.533	2.29	130348	1.303	2.69
kem_keypair	5	1712079	17.12	1132673	11.33	1.51	1090458	10.90	1.57
kem_enc		1976941	19.77	1183306	11.83	1.67	1126462	11.26	1.76
kem_dec		431106	4.311	187809	1.878	2.30	159639	1.596	2.70

D. PERFORMANCE TESTS

Our aim is to measure and understand how the PQ ALU insertion improves the overall cryptographic performance. To do that, we measured the number of clock cycles to compute each principal cryptographic function in three different scenarios to understand the impact of our acceleration: without PQ ALU, with only *fqmul* and *reduce* accelerations, and with all five instructions per algorithm (including logic for butterfly operations).

Once the functionalities have been verified against the KAT (Known Answer Test) values provided by NIST, we defined the second set of tests to measure the performance of our proposed PQ ALU and the speed-up obtained. Each of these functions has been executed in our hardware prototype by the RISC-V processor, with a total number of 10’000 iterations. In Table 8 we present the average value of clock cycles necessary to execute each function and its sub-functions for Kyber and in Table 9 for Dilithium. Also, the performance improvements are shown, expressed in terms of clock cycle reduction for executing the task. The average speed-up achieved for the main cryptographic functions ranges from 1.2× to 2.7× depending on the security level. Please note that the GS butterfly of Kyber (adopted in the inverse NTT) contains the Barret reduction operation that is not present in CT butterfly (adopted in the forward NTT). This results in a higher speed-up because the software version of the INTT takes more clock cycles than the NTT, while the optimized instructions for the butterflies take just one clock cycle in both cases.

V. RESULTS DISCUSSIONS

Combining the achieved results in terms of complexity, timing performance, and energy per function, what we observe is that with our solution we achieve from 20% to 65% speed-up of the Kyber and Dilithium Functions, with an almost negligible increase of LUT (+3%), no impact on FF and moderate use of DSP (+5 Units) in the FPGA. Thanks to the large timing improvement at low hardware cost, there is also a significant advantage in the energy required for performing a Kyber/Dilithium operation, which is reduced approximately by the same factor of the performance speed-up.

The approach we followed is inspired by [33], where a similar ISA extension is carried out for the LAC algorithms, with remarkable results. Other contributions present a tightly

TABLE 9. Dilithium functions analysis with our proposed PQ ALU.

Function	Security Level	Standard Execution		fqmul + reduce32			fqmul + reduce32 + butterfly		
		C.c.	Time [ms]	C.c.	Time [ms]	Speed-up	C.c.	Time [ms]	Speed-up
NTT	2-3-5	38043	0.3804	21753	0.2175	1.75	18554	0.1855	2.05
INTT	2-3-5	46266	0.4627	24125	0.2413	1.92	21375	0.2138	2.16
dil_keypair	2	1873016	18.73	1624132	16.24	1.15	1592325	15.92	1.18
dil_sign		8195317	81.95	6122761	61.23	1.34	5884266	58.84	1.39
dil_verify		2083853	20.84	1745897	17.46	1.19	1700679	17.01	1.23
dil_keypair	3	3400741	34.07	3012387	30.12	1.13	2974897	29.75	1.14
dil_sign		13933992	139.3	10610866	106.1	1.31	10211677	102.1	1.36
dil_verify		3529953	35.30	3025268	30.25	1.17	2963936	29.64	1.19
dil_keypair	5	5620591	56.21	5045756	50.46	1.11	5001302	50.01	1.12
dil_sign		17633430	176.3	13761208	137.6	1.28	13339255	133.4	1.32
dil_verify		5936553	59.37	5206282	52.06	1.14	5132776	51.33	1.16

integrated hardware accelerator in the RISC-V processor. In [34] an NTT & Hash accelerator is embedded in the RISCY processor on an FPGA as a hardware accelerator, with low hardware consumption (886 LUT, 618 FF, and 26 DSP) which is however higher than our PQC ALU, and with comparable performance in terms of NTT clock cycles count (24609 against ours 22866). In [35], the authors follow a completely different approach, designing a domain-specific vector co-processor, integrated with a RISC-V processor, focusing on the NTT transform for the Ring-LWE and Module LWE PQC algorithms, with a complexity in the order of 942 kGates. There are also contributions targeting ASIC accelerators, which achieve higher performance but at the price of more expensive hardware with less flexibility: in [36] a 65nm hardware accelerator oft both Kyber and Dilithium (plus other PQC algorithms) is presented. Finally, there are several contributions on Kyber and/or Dilithium hardware accelerators for FPGAs, which are close to our presented implementation but target slight higher performance with higher complexity: in [37] a Dilithium hardware accelerator is presented, capable of performing all the Dilithium functions (keygen, signature, and verification) in hardware, at the price of about 70k LUT and 86k FF, with performance in the order of 10-20 kOps, depending on the function. In [27] a Kyber hardware accelerator, integrated into a SoC is presented. The entire NTT primitive is performed in hardware, providing better timing performance but also much higher resource consumption (about 7k LUT, 4,6k FF, and 2 DSPs) than our proposed architecture. In [38] a Dilithium hardware accelerator, still based on the NTT primitive is presented; the performance is remarkable (from 1 to 11 kOps), but also the resource consumption is quite high (30k LUT, 11k FF, 45 DSP) compared to our solution.

Even if a detailed comparison with the few alternative solutions available in the literature is not easy, in Table 10 we compare our work with similar solutions. Since many factors are affecting the performance of the systems presented in the various works, we decided to compare only the NTT core designed for FPGA applications, to be used for the Kyber protocol (n = 256, q = 3329). On top of that, we did not consider the execution time for the NTT or the number of clock cycles necessary to carry out the NTT operation, since those numbers are affected by many factors which are not always specified in literature, like security level and code optimisation. We decided to present and compare the speed-up that each circuit achieved using hardware acceleration for the NTT function. Resource utilisation in terms of LUTs, FFs, DSPs, and BRAMs is reported. Even though it is not possible to compute a Speed-up Vs. Resource overhead, since implementations are very different (e.g. some use DSPs and/or BRAM, others do not), it is possible to observe that the speed-up is proportional to the resource utilisation and that our proposed solution is aligned to the state of the art.

TABLE 10. NTT hardware accelerators on FPGA comparison (n = 256, q = 3329).

	NTT Absolute Speedup	Complexity			
		LUT	FF	DSP	BRAM
This work	2.72	178	0	5	0.5
[39]	1.25	104	36	1	34
[40]	6.00	417	462	0	0
[34]	14.00	886	618	26	1
[41]	3.95	609	640	2	4
[42]	2.10	533	514	1	3

VI. CONCLUSION

LBC is one of the most promising candidates to win the NIST standardization process, with Kyber and Dilithium which are finalists of the third round of this process. An important achievement of this work is the proof that PQC algorithms can be significantly accelerated by exploiting the flexibility of RISC-V processors, integrating dedicated accelerators directly in the core pipeline. After the analysis of the Kyber and Dilithium algorithms, their onerous parts have been identified within the polynomial multiplications which use the NTT and modular reductions. We proposed a dedicated architecture to accelerate them directly in the pipeline of a

RISC-V processor, exploiting dedicated instructions for the *fqmul* and Barrett reduction in case of Kyber, and the *fqmul* and *reduce32* in case of Dilithium. In addition, we integrated also the butterfly operations used in NTT and INTT computations for both algorithms. We defined dedicated assembly instruction for each of these functions and we realised two different hardware accelerators, which can be integrated as FUs into the processor pipeline. After that, we designed a SoC implemented on the Xilinx ZCU106 evaluation board for gathering all the performance and power consumption results. As shown in Section V, the number of clock cycles to perform the different functions have been speeded-up consistently, up to 2.7× for *indcpa_dec* with a security level of 5. These results are achieved at an affordable price in terms of resource consumption, leading also to an energy-saving comparable to the obtained speed-up.

This work provides the first available indication of the power consumption of those operations and the impact that in-pipeline hardware acceleration has on the energy required to execute every single function.

REFERENCES

[1] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "Overview of 5G security challenges and solutions," *IEEE Commun. Standards Mag.*, vol. 2, no. 1, pp. 36–43, Mar. 2018.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999.

[3] G. Alagi, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, and R. Perlner, "Status report on the second round of the NIST post-quantum cryptography standardization process," U.S. Dept. Commerce, NIST, Gaithersburg, MD, USA, Tech. Rep., 2020.

[4] K. Basu, D. Soni, M. Nabeel, and R. Karri, "NIST post-quantum cryptography—A hardware evaluation study," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 47, May 2019.

[5] P. Mohan, W. Wang, B. Jungk, R. Niederhagen, J. Szefer, and K. Mai, "ASIC accelerator in 28 nm for the post-quantum digital signature scheme XMSS," in *Proc. IEEE 38th Int. Conf. Comput. Design (ICCD)*, Oct. 2020, pp. 656–662.

[6] F. Yaman, A. C. Mert, E. Öztürk, and E. Savas, "A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 485, Feb. 2021.

[7] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, pp. 239–280, Aug. 2020.

[8] K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for RISC-V," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2014-146, 2014.

[9] P. K. Pradhan, S. Rakshit, and S. Datta, "Lattice based cryptography: Its applications, areas of interest & future scope," in *Proc. 3rd Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Mar. 2019, pp. 988–993.

[10] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2629–2640, Nov. 2019. [Online]. Available: https://github.com/openhwgroup/cva6

[11] A. Zeh, A. Glew, B. Spinney, B. Marshall, D. Page, D. Atkins, K. Dockse, M. J. O. Saarinen, N. Menhorn, R. Newell, and C. Wolf. (2021). *RISC-V Cryptographic Extension Proposals Volume I: Scalar & Entropy Source Instructions*. [Online]. Available: https://github.com/riscv/riscv-crypto/releases

[12] D. Harvey, "Faster arithmetic for number-theoretic transforms," *J. Symbolic Comput.*, vol. 60, pp. 113–119, Jan. 2014.

[13] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *Proc. Int. Conf. Cryptol. Netw. Secur. (CANS)*. Milan, Italy: Springer, Nov. 2016, pp. 124–139.

[14] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–41, Feb. 2019.

[15] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des., Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, Jun. 2015.

[16] A. Cintas Canto, M. M. Kermani, and R. Azarderakhsh, "Reliable architectures for composite-field-oriented constructions of Mceliece post-quantum cryptography on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 999–1003, May 2021, doi: 10.1109/TCAD.2020.3019987.

[17] C. Chen. (2021). *NTRU*. [Online]. Available: https://ntru.org/

[18] J.-P. DâĂŹAnvers, A. Karmakar, S. S. Roy, and F. Vercauteren. (2021). *Saber: Mod-LWR based KEM*. [Online]. Available: https://www.esat.kuleuven.be/cosic/pqcrypto/saber/resources.html

[19] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology*, M. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 537–554.

[20] L. Ducas. (2021). *Crystals-Kyber, Algorithm Specifications and Supporting Documentation, Version 3.01*. [Online]. Available: https://pq-crystals.org/kyber/resources.shtml

[21] K. Kiningham, P. Levis, M. Anderson, D. Boneh, M. Horowitz, and M. Shih, "Falcon—A flexible architecture for accelerating cryptography," in *Proc. IEEE 16th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Nov. 2019, pp. 136–144, doi: 10.1109/MASS.2019.00025.

[22] T. Yasuda and K. Sakurai, "A multivariate encryption scheme with rainbow," in *Proc. 17th Int. Conf. (ICICS)*, in Lecture Notes in Computer Science, vol. 9543, Beijing, China, Dec. 2015, pp. 236–251, doi: 10.1007/978-3-319-29814-6_19.

[23] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, and D. Stehlè, "Crystals-dilithium," Submission NIST Post-Quantum Cryptogr. Standardization [NIS], Gaithersburg, MD, USA, Tech. Rep., 2017.

[24] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in *Progress in Cryptology* (Lecture Notes in Computer Science), A. Hevia and G. Neven, Eds. Berlin, Germany: Springer, 2012, pp. 139–158.

[25] P. Longa and M. Naehrig. (2016). *Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography*. [Online]. Available: https://eprint.iacr.org/2016/504

[26] C. Team. *Kyber Website*. Accessed: Sep. 7, 2021. [Online]. Available: https://pq-crystals.org/kyber/index.shtml

[27] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 328–356, Feb. 2021.

[28] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, Apr. 1985.

[29] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Advances in Cryptology*, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 311–323.

[30] Xilinx. *ZCU106 Evaluation Kit*. Accessed: Sep. 7, 2021. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/zcu106.html

[31] C. Team. *Dilithium Website*. Accessed: Sep. 7, 2021. [Online]. Available: https://pq-crystals.org/dilithium/index.shtml

[32] Xilinx. (2019). *Accurate design Power Measurement Made Easier*. [Online]. Available: https://developer.xilinx.com/en/articles/accurate-design-power-measurement.html

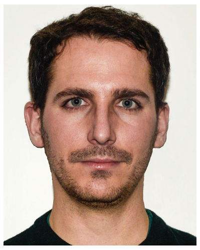
[33] T. Fritzmann, G. Sigl, and J. Sepúlveda, "Extending the RISC-V instruction set for hardware acceleration of the post-quantum scheme LAC," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1420–1425.

[34] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on RISC-V," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1148–1153.

[35] G. Xin, J. Han, T. Yin, Y. Zhou, J. Yang, X. Cheng, and X. Zeng, "VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2672–2684, Aug. 2020.

[36] M. Imran, Z. U. Abideen, and S. Pagliarini, "A systematic study of lattice-based NIST PQC algorithms: From reference implementations to hardware accelerators," 2020, *arXiv:2009.07091*.

[37] S. Ricci, L. Malina, P. Jedlicka, D. Smekal, J. Hajny, P. Cíbik, and P. Dobias, "Implementing crystals-dilithium signature scheme on FPGAs," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 108, Aug. 2021.
[38] G. Land, P. Sasdrich, and T. Güneysu, "A hard crystal-implementing dilithium on reconfigurable hardware," IACR Cryptol. ePrint Arch., Tech. Rep. 2021/1451, 2021, p. 355, vol. 2021.
[39] E. Alkim, H. Evkan, N. Lahr, R. Niederhagen, and R. Petri. (2020). *ISA Extensions for Finite Field Arithmetic-Accelerating Kyber and NewHope on RISC-V*. Cryptology ePrint Archive, Report 2020/049. [Online]. Available: https://eprint.iacr.org/2020/049.
[40] E. Karabulut and A. Aysu, "RANTT: A RISC-V architecture extension for the number theoretic transform," in *Proc. 30th Int. Conf. Field-Program. Log. Appl. (FPL)*, Aug. 2020, pp. 26–32.
[41] C. Zhang, D. Liu, X. Liu, X. Zou, G. Niu, B. Liu, and Q. Jiang, "Towards efficient hardware implementation of NTT for kyber on FPGAs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.
[42] Z. Chen, Y. Ma, T. Chen, J. Lin, and J. Jing, "High-performance area-efficient polynomial ring processor for CRYSTALS-kyber on FPGAs," *Integration*, vol. 78, pp. 25–35, May 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016792602100002X


PIETRO NANNIPIERI (Member, IEEE) received the Ph.D. degree *(cum laude)* in information engineering from the University of Pisa, in 2020. He is currently a Postdoctoral Researcher with the University of Pisa. He spent several months in 2019, as a Visiting Researcher with the TEC-EDP Section, ESTEC (ESA), where he carried out different qualification tests on the spacefibre technology. He is also working on the European Processor Initiative (EPI) Project. His interests are digital and VLSI design, electronics for space applications, and cryptography. His research interests include hardware IPs for satellite onboard data handling, signal processing, and hardware cryptography.

STEFANO DI MATTEO received the B.Sc. degree in electronic engineering from the University of Salerno, in 2015, and the M.Sc. degree in electronic engineering from the University of Pisa, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include digital systems design and hardware design of cryptographic primitives with particular focus on elliptic curve cryptography (ECC) and ring learning with errors (RLWE).

LUCA ZULBERTI (Student Member, IEEE) received the M.Sc. degree *(cum laude)* in electronic engineering from the University of Pisa, in 2021. Since March 2021, he has been working as a fellow with the Department of Information Engineering, University of Pisa, on design of embedded systems based on RISC-V architecture. He has got interest for the computer architecture and micro-architectural designs.

FRANCESCO ALBICOCCHI received the M.Sc. degree in electronic engineering from the University of Pisa, in 2021, with a thesis on a preliminary post-quantum computing instruction set architecture extension to an RISC-V processor. He is currently working as an FPGA Designer in an Italian company in Pisa.

SERGIO SAPONARA (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Pisa, Italy. He was a Marie Curie Fellow with imec, Belgium. He is currently a Full Professor of electronics with the University of Pisa. He has coauthored about 300 scientific articles and holds 18 patents. He is also a Founding Member of the IoT CASS SiG. He has been a TPC member of over 100 international IEEE and SPIE conferences. He is also an associate editor of several IEEE and IET journals. Since 2017, he has been an IEEE IMS Distinguished Lecturer.

LUCA FANUCCI (Fellow, IEEE) received the Ph.D. degree in electronic engineering from the University of Pisa, in 1996. From 1992 to 1996, he was with the European Space Agency as a Research Fellow. From 1996 to 2004, he was a Senior Researcher with the Italian National Research Council, Pisa. He is currently a Professor of microelectronics with the University of Pisa. He is the coauthor of more than 400 journal and conference papers and a co-inventor of more than 40 patents. His research interests include several aspects of design technologies for integrated circuits and electronic systems. He served in several technical program committees of international conferences.