FPGA-Based Design and Implementation of a Code-Based Post-quantum KEM

Andrea Galimberti

Abstract Post-quantum cryptography aims to design cryptosystems that can be deployed on traditional computers and resist attacks from quantum computers, which are widely expected to break the currently deployed public-key cryptography solutions in the upcoming decades. Providing effective hardware support is crucial to ensuring a wide adoption of post-quantum cryptography solutions, and it is one of the requirements set by the USA's National Institute of Standards and Technology within its ongoing standardization process. This research delivers a configurable FPGA-based hardware architecture to support BIKE, a post-quantum QC-MDPC code-based key encapsulation mechanism. The proposed architecture is configurable through a set of architectural and code parameters, which make it efficient, providing good performance while using the resources available on FPGAs effectively, flexible, allowing to support different large QC-MDPC codes defined by the designers of the cryptosystem, and scalable, targeting the whole Xilinx Artix-7 FPGA family. Two separate modules target the cryptographic functionality of the client and server nodes of the quantum-resistant key exchange, respectively, and a complexity-based heuristic that leverages the knowledge of the time and space complexity of the configurable hardware components steers the design space exploration to identify their best parameterization. The proposed architecture outperforms the state-of-the-art reference software that exploits the Intel AVX2 extension and runs on a desktop-class CPU by 1.77 and 1.98 times, respectively, for AES-128- and AES-192-equivalent security instances of BIKE, and it provides a speedup of more than six times compared to the fastest reference state-of-the-art hardware architecture, which targets the same FPGA family.

A. Galimberti (✉)
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy
e-mail: andrea.galimberti@polimi.it

F. Amigoni (ed.), *Special Topics in Information Technology*,
PoliMI SpringerBriefs, https://doi.org/10.1007/978-3-031-51500-2_3

1 Introduction

Public-key cryptography (PKC) allows sending encrypted messages over an insecure channel without sharing a secret key, and it has traditionally been a critical component of secure communication protocols such as TLS and SSH. Quantum computing is, however, expected to break the traditional PKC solutions [5, 10, 30] in the upcoming decades, making it mandatory to design new security solutions that can also resist attacks carried out by quantum computers.

Post-quantum cryptography (PQC) aims to design cryptosystems that can be deployed on traditional computers and are based on problems that are computationally hard also for quantum computers, other than traditional ones, thus being able to resist both traditional and quantum attacks.

The USA's National Institute of Standards and Technology (NIST) is currently undertaking a standardization process to define new standards for PQC. Starting from 82 submissions in 2017, it selected as standards four schemes that can be split into key encapsulation mechanisms (KEMs), which are meant to share secret keys confidentially, and digital signatures, which guarantee the authenticity and integrity of a message to the recipient.

All four schemes selected as standards are lattice-based ones [22, 26], i.e., based on the shortest vector problem (SVP), which requires searching for the non-zero vector of a lattice having minimum norm and that is considered NP-hard for both traditional and quantum computers [27].

NIST claimed, therefore, the need to diversify its portfolio of PQC solutions and expects to select one more KEM among the three remaining code-based ones, i.e., BIKE, Classic McEliece, and HQC. Code-based cryptography dates back to the McEliece cryptosystem, introduced in 1978 and based on the difficulty of decoding a generic linear code [21], which is recognized as an NP-hard problem. Code-based cryptosystems in NIST's PQC standardization process are compared in Figs. 1 and 2,

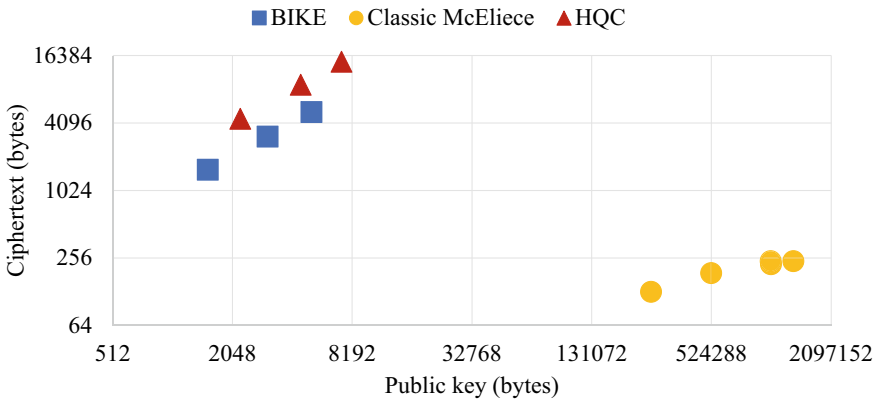

Fig. 1 Size in bytes of the public key and ciphertext of the KEMs advancing to the fourth round of the NIST PQC standardization process [24]

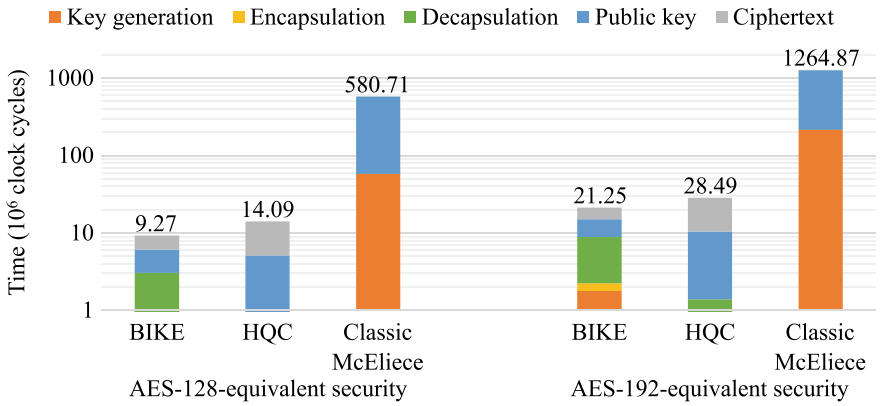

Fig. 2 Performance of NIST Round 4 KEMs on a x86-64 CPU, considering a 2000 cycles/byte transmission cost [25]

respectively, according to their public key and ciphertext sizes, which show how Classic McEliece has a huge public key, in the order of millions of bits, and software performance, which highlights BIKE as the best performing scheme when also considering the cost of transmitting the public keys and ciphertexts between the communicating nodes.

BIKE is a post-quantum code-based KEM using quasi-cyclic moderate-density parity-check (QC-MDPC) codes. These codes are employed in a scheme similar to the well-studied Neiderreiter one, which dates back to the early 1980s. Compared to traditional Niederreiter schemes, whose underlying binary Goppa codes must have sizes in the order of millions of bits to provide quantum resistance, BIKE achieves a significantly smaller public key, in the order of tens of thousands of bits, through its usage of QC-MDPC codes.

Given the complexity of PQC cryptosystems such as BIKE in terms of memory requirements and software performance, providing effective hardware support will be paramount to ensuring a wide adoption and effective deployment of post-quantum security solutions across the computing continuum ranging from embedded devices at the edge to HPC [1]. Indeed, with ever more private, sensitive, and critical data collected and processed in a variety of scenarios, it is mandatory to design computing platforms that not only provide optimal performance for the target applications [13, 33, 34] and the energy and power efficiency required by the specific use case [35] but also guarantee the security of the users' data.

Implementations of BIKE from the literature encompass software, hardware, and hardware-software ones. However, all of them suffer from different drawbacks [16]. Software implementations [3, 7, 8], including those targeting desktop-class Intel CPUs with support for AVX2 instructions and running at more than 4 GHz [2], provide poor performance, whereas hardware ones are custom-tailored to specific target platforms [28, 29].

This research delivers a configurable FPGA-based hardware architecture to support BIKE through two modules dedicated the client- and server-side functionalities of the key exchange. The proposed architecture aims to improve performance over the existing state-of-the-art software and hardware implementations of BIKE, and it is configurable through architectural and code parameters that, through a single parametric design, allow for using the resources available on FPGAs effectively, supporting different large QC-MDPC codes, and targeting the whole Xilinx Artix-7 FPGA family.

2 Components for QC-MDPC Code-Based Cryptography

The hardware components implementing binary polynomial inversion [17], binary polynomial multiplication [4], and Black-Gray-Flip (BGF) decoding [31], i.e., the three most complex operations employed within the BIKE cryptosystem, were specifically designed in a parametric way to exploit parallelism as desired according to the performance requirements and the area constraints given by the target platform. Their designs, meant for FPGA targets, are suitable not only for accelerating the BIKE post-quantum KEM but more in general for other applications making use of large binary polynomials and QC-MDPC codes.

Dense-dense binary polynomial multiplication The dense-dense binary polynomial multiplier [32] performs the multiplication between two large polynomials in $\mathbb{Z}_2[x]/(x^p + 1)$, with degree p in the order of tens of thousands, through a hybrid architecture that mixes the Karatsuba and Comba algorithms [9, 20].

Applying a configurable number of iterations of the Karatsuba algorithm reduces the number of smaller partial products compared to schoolbook multiplication. Each iteration can either compute its three partial products in parallel, on separate internal multipliers, or sequentially, on a shared one. The multipliers employed to compute such partial products either have a Karatsuba architecture themselves or a Comba-based one. At the end of Karatsuba's recursive application, the Comba formula is indeed leveraged to perform the actual computation of the partial products since the size of the operands after the recursive application of the Karatsuba algorithm is still too large to fit into a combinational multiplier. Comba multiplication schedules efficiently the computation of such partial products on a combinational component that performs the carry-less multiplication between two BW-bit digits, where BW corresponds to the datapath bandwidth.

Selecting the number of Karatsuba recursions, whether each computes its partial products sequentially or concurrently, and the datapath bandwidth allows for exploring a variety of performance-area trade-offs.

Binary polynomial exponentiation The exponentiation at the power of k of a polynomial $f(x)$ in $\mathbb{Z}_2[x]/(x^p + 1)$, where k and p are coprime as in QC-MDPC codes

employed by BIKE, corresponds to a permutation in which each i-th bit of the operand $f(x)$ corresponds to the $((i \cdot k) \bmod p)$-th bit of the result $g(x)$.

The exponentiation component [17] implements a two-stage architecture. The first one includes a p-bit memory and outputs E bits per cycle, while the second one contains E p-bit memories, each receiving a bit from the first stage and writing it in the corresponding position. Finally, the contents of the second-stage memories are XORed to produce the actual result of the exponentiation. As an optimization, the usage of lookup tables pre-computed at design time avoids the computation of the bit start addresses and address increments required to obtain the positions of bits in the result polynomial.

The E number of result bits computed per clock cycle, which determines the execution time and area of the exponentiation component, can be selected at design time with any value between 1 and p.

Binary polynomial inversion The binary polynomial inversion component [17] implements a Fermat-based algorithm that computes, by iterating binary polynomial multiplications and exponentiations, the multiplicative inverse of a polynomial in $\mathbb{Z}_2[x]/(x^p + 1)$, which is the most time-consuming operation in BIKE's key generation primitive [19].

The multiplications and exponentiations are carried out on dense-represented operands by two separate parametric components, i.e., the dense-dense binary polynomial multiplication and binary exponentiation components described previously. The two types of operations are computed on their dedicated components by scheduling them in a pipelined fashion, executing independent multiplications and exponentiations concurrently and thus minimizing the execution time of the overall inversion operation.

The dense-dense binary polynomial multiplication and binary polynomial exponentiation components are configurable in their code and architectural parameters, and finding an optimal performance-area trade-off for the inversion one requires balancing their resource utilization and execution time.

Black-Gray-Flip decoding The decoding component implements the BGF decoding algorithm [11], a variant of the baseline QC-MDPC bit-flipping decoding algorithm. The BGF algorithm iterates the computation of two multiplications, performed respectively in the integer and binary domains, between a dense polynomial operand and a sparse one [31]. The two dense-sparse multiplications are performed concurrently in a pipelined fashion, and the number of the bits computed in parallel in both is configurable by the designer [4].

The multiplication between a sparse polynomial $s(x)$ with Hamming weight v, i.e., v coefficients set to 1, and a dense one $d(x)$ corresponds to the addition of v copies of $d(x)$ each shifted by the position of the corresponding 1 in $s(x)$. In the binary domain case, the addition corresponds to XOR, and the result polynomial has binary coefficients, i.e., either 0 or 1. On the contrary, in the integer domain case, it corresponds to integer arithmetic addition, and the result's coefficients are thus integer values comprised between 0 and v. The two integer- and binary-domain

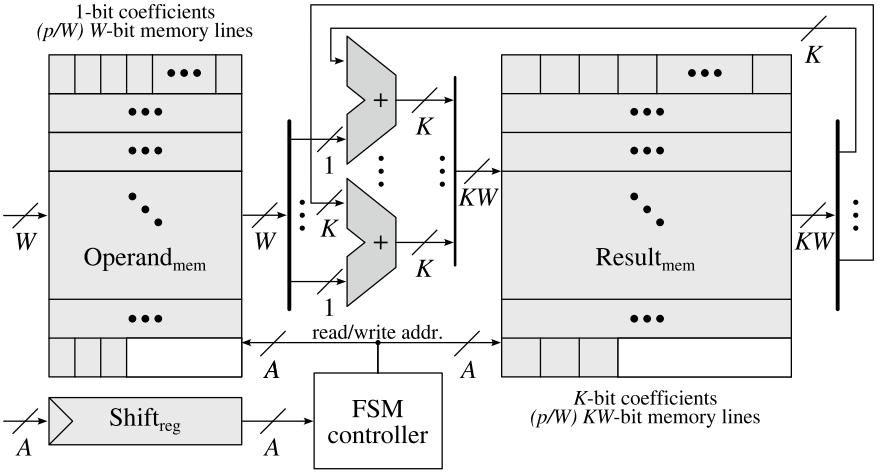

Fig. 3 Baseline architecture of the sparse-dense multiplication components

multiplications are performed by separate components, each dedicated specifically to one of them, but both implement a similar architecture.

The baseline architecture, depicted in Fig. 3, stores in a BRAM memory ($\texttt{Operand}_{\texttt{Mem}}$) the dense operand polynomial and in a flip-flop-based register ($\texttt{Shift}_{\texttt{Reg}}$) the position of a bit set to 1 in the sparse one. The content of $\texttt{Operand}_{\texttt{Mem}}$ is shifted according to the value stored in Shift_{Reg} and accumulated in the result polynomial BRAM memory ($\texttt{Result}_{\texttt{Mem}}$) according to the addition operation specific to the implemented arithmetic. In Fig. 3, W corresponds to the number of polynomial coefficients read and written per clock cycle, K refers to the bit length of the coefficients of the result polynomial, and A refers to the width of read and write addresses.

The computation of the overall sparse-dense multiplication can be parallelized, reducing execution time at the cost of additional area, by instantiating multiple shift-and-accumulate modules. Up to v of such modules can be implemented to perform the shift-and-accumulate operation after feeding them different values of positions of bits set to 1 in the sparse operand. The overall product of the multiplication will finally be obtained as the sum of the result polynomials from each of the instantiated shift-and-accumulate modules.

Sparse-dense binary polynomial multiplication The sparse-dense binary polynomial multiplier [4] is employed within all three KEM primitives of BIKE, i.e., key generation, encapsulation, and decapsulation, and it is designed with the same architecture as the one employed by the binary dense-sparse multiplier instantiated in the BGF decoding module. Its parallelism is similarly configurable by selecting the number of shift-and-accumulate operations to compute concurrently, which can be any value between 1 and v, where v is the Hamming weight of the dense operand polynomial.

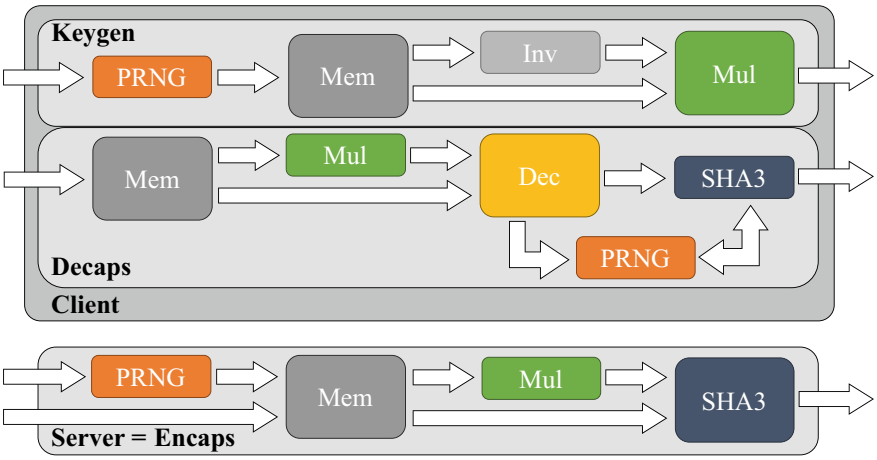

Fig. 4 Top-level architecture of the BIKE client and server cores

Other components The SHA-3 component [14] implements the SHA3-384 cryptographic hash function [12]. It computes the 384-bit digest of the SHA3-384 cryptographic function of the input message according to an architecture similar to the high-speed core detailed in [6], which was modified to support the standard SHA-3 cryptographic hash functions in place of pre-standard Keccak functions.

The pseudorandom number generation (PRNG) component [14] performs the generation of a pseudorandom sequence of bits with fixed Hamming weight by using an internal SHAKE256 module, which implements an architecture similar to the SHA-3 component, albeit producing a variable-length output according to the needs of the surrounding pseudorandom generation logic. The SHAKE256 module expands a seed obtained from a TRNG [18] into a digest output that is broken up into $(\log_2 p)$-bit chunks, each possibly representing the position of a bit set to 1 within a p-bit vector, and the extracted values are evaluated to discard the values which have been generated previously, avoiding cancellations and therefore enabling the generation of a vector with the desired Hamming weight. Moreover, values larger than or equal to p are discarded, providing a uniform distribution of bits set to 1 within the random-generated bit vector.

3 Client-Server BIKE Architecture

Two separate cores target the cryptographic functionality of the client and server nodes of the BIKE key exchange, respectively. The client and server cores, whose architecture is depicted in Fig. 4, make use of the configurable binary polynomial arithmetic and BGF decoding components, the SHA-3 core, and the pseudorandom

number generator that were previously described, and contain additional BRAM-based memories to store the large binary polynomials [15].

The `Client` core is composed of two main modules, `Keygen` and `Decaps`, devoted to the key generation and decapsulation of BIKE, respectively [14]. The `Keygen` module performs three subsequent hardware operations, namely pseudo-random number generation (executed by the `PRNG` component), binary polynomial inversion (`Inv`), and binary polynomial multiplication (`Mul`). Similarly, the `Decaps` module executes a sequence of four hardware operations, namely binary polynomial multiplication (`Mul`), BGF decoding (`Dec`), computation of SHA-3 hash digest (`SHA3`), and pseudorandom number generation (`PRNG`). The `PRNG` and `Mul` components are notably shared between the `Keygen` and `Decaps` modules to minimize duplicate hardware resources.

The `Server` core only includes the `Encaps` module [14], devoted to the encapsulation primitive of BIKE, which requires performing a sequence of three hardware operations, namely pseudorandom number generation (`PRNG`), binary polynomial multiplication (`Mul`), and computation of the SHA-3 hash function (`SHA3`).

The optimal parameterization, which maximizes performance within the available FPGA resources, of the configurable components, i.e., binary polynomial arithmetic and BGF decoding ones, is identified by using a complexity-based heuristic that leverages the knowledge of such parametric components' time and space complexity to steer the design space exploration. The execution time is selected as a proxy for the time complexity, while the space complexity is modeled by the number of occupied BRAM memory blocks since the design is dominated by BRAM usage due to the large polynomials and the exploited parallelism.

4 Experimental Evaluation

The experimental evaluation aims to gauge the performance and resource utilization improvements of the proposed FPGA-based architectures compared to state-of-the-art software, hardware-software, and hardware implementations.

Experimental setup The proposed components were described in SystemVerilog and then implemented in Xilinx Vivado 2020.2 targeting Xilinx Artix-7 FPGAs, which were selected as the target platform since they are the de-facto standard in research, due to their wide availability and best price-performance ratio among FPGAs, and they were chosen as the hardware target by NIST, to avoid differences due to FPGA technologies and ASIC technology nodes. RTL synthesis and implementation were carried out targeting a 91 MHz clock frequency, i.e., an 11ns clock period.

The proposed architectures were validated from the functional point of view, both through post-implementation simulation, on Artix-7 35, Artix-7 50, and Artix-7 200 FPGAs, and through prototype execution on a Digilent Nexys 4 DDR board, which features an Artix-7 100 FPGA. In each case, the results from the executions of 10000

key generations, encapsulations, and decapsulations on the proposed architectures were compared with the corresponding outputs of software execution.

Reference implementations The experimental evaluation was carried out against state-of-the-art software, hardware-software, and hardware implementations of the BIKE post-quantum KEM.

The additional Intel AVX2-optimized software implementation of BIKE [2] was selected as the software reference. It provides a constant-time execution on Intel x86-64 CPUs that support the Intel AVX2 instruction set extension, i.e., CPUs from the Intel Haswell generation and later ones. Within the experimental evaluation, it was executed on an Intel Core i5-10310U CPU, a desktop-class 64-bit processor implementing the x86-64 ISA and providing support for the Intel AVX2 extension, running at a clock frequency up to 4.4 GHz. Moreover, the PC mounting the Intel CPU ran the Ubuntu 20.04.3 LTS operating system.

The solution proposed in [23], which makes use of HLS-generated accelerators, each implementing a BIKE primitive, was selected as the hardware-software reference. Three different combinations of KEM primitives implemented in hardware, depending on the available FPGA resources, with the remaining ones executed instead in software on the CPU, allow targeting three chips from the Xilinx Zynq-7000 heterogeneous SoC family, which feature ARM CPUs coupled with programmable FPGA logic equivalent to the Artix-7 one.

The official FPGA-based hardware implementation [28] was instead selected as the state-of-the-art hardware reference. The proposed design, targeting Xilinx FPGAs and described in SystemVerilog, delivers a unified architecture that implements the whole BIKE KEM and executes it in constant time. The authors provide three instances ranging from a lightweight one that minimizes resource utilization up to mid-range and high-performance ones.

Area results The area of the proposed architecture is evaluated according to its utilization of the FPGA resources available on the target chips. Table 1 details the look-up tables (LUT), flip-flops (FF), and block RAM (BRAM) blocks occupied by the client and server instances. The proposed architecture's smallest client and server

Table 1 Area results, expressed in terms of LUT, FF, and BRAM resources, and execution times, in milliseconds, for the proposed client and server cores

Core	Equivalent security	Lightweight				High-performance			
		Resources			Exec. time	Resources			Exec. time
		LUT	FF	BRAM		LUT	FF	BRAM	
Client	AES-128	31792	17805	43.5	5.71	126510	51492	357	0.58
	AES-192	31411	20181	45.5	19.27	124891	53067	360	1.71
Server	AES-128	19804	11401	30	0.03	91422	46208	275.5	0.03
	AES-192	19979	12282	28	0.08	72725	37795	235.5	0.06

Table 2 Execution times, in milliseconds, for the state-of-the-art and proposed implementations. Legend: **LW** lightweight, **MR** mid-range, **HP** high-performance instances

Equivalent security	Ref. SW [2]	Ref. HW/SW [23]			Ref. HW [28]			Proposed	
	AVX2	LW	MR	HP	LW	MR	HP	LW	HP
AES-128	1.08	617.31	482.48	288.18	11.13	6.36	3.69	5.74	0.61
AES-192	3.51	–	–	–	37.10	19.71	11.69	19.35	1.77

cores fit in Artix-7 50 and 35 FPGAs, respectively, while the largest instances target Artix-7 200 chips, i.e., the highest-end chips of the FPGA family.

The experimental results demonstrate how the proposed cryptographic cores can scale across a range of FPGA chips. Moreover, they show that BRAM memories are the most used resources, relatively to the ones available on the target chip, on the larger Artix-7 200 FPGAs, while instances targeting the smaller chips are bounded by the LUT utilization. The proposed architectures usually employ a large fraction of the available look-up tables while requiring a more limited amount of flip-flops.

Performance results Performance is measured by the execution time of the BIKE KEM primitives on the client and server sides of the key exchange. Table 1 lists the execution times, expressed in milliseconds, for the client and server instances of the proposed architecture, while Table 2 compares the aggregate execution times of BIKE between the state-of-the-art and proposed solutions.

The experimental results highlight significant improvements over the considered state-of-the-art references. The latency of the BIKE KEM can be reduced by almost two times, in the AES-192-equivalent use case, compared to the AVX2-optimized software execution, and the smaller proposed instances outperform even the mid-range state-of-the-art FPGA-based instances. Finally, the best-performing proposed architectures outperform the high-performance state-of-the-art ones by more than six times, as also shown in Fig. 5, which compares the execution time, broken down in the three KEM primitives, between the FPGA-based architectures.

5 Conclusions

This research presented a configurable FPGA-based hardware architecture that implements the BIKE QC-MDPC code-based cryptosystem, aiming to improve performance over the existing state-of-the-art software and hardware solutions.

The proposed architecture provides effective FPGA-based hardware support for QC-MDPC codes suitable to post-quantum cryptography applications. Configurable code and architectural parameters allow using a single design to support different QC-MDPC codes underlying the PQC cryptosystems and to target any FPGA chip from the Xilinx Artix-7 family. Hence, different performance-area trade-offs can be explored through the parametric configurability to satisfy the performance require-

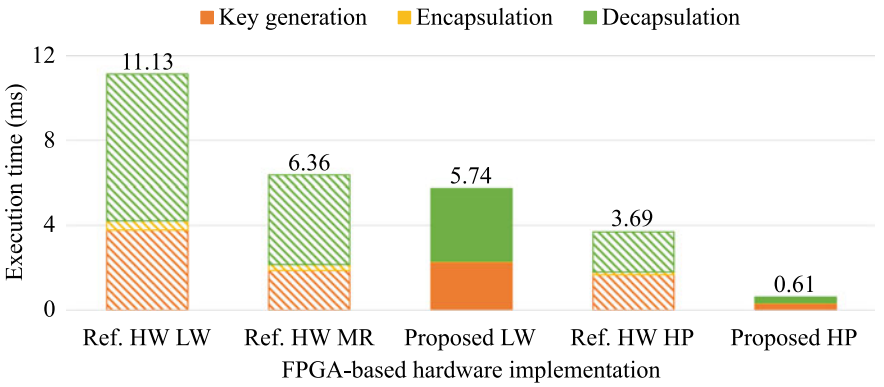

Fig. 5 Execution times of BIKE with AES-128-equivalent security. Legend: **LW** lightweight, **MR** mid-range, **HP** high-performance instances

ments and area constraints set for the overall system that integrates BIKE hardware support. Two modules support the KEM primitives to be executed on the client and server nodes of the key exchange, respectively, and a complexity-based heuristic steers the design space exploration to identify the best parameterization of the configurable hardware components by leveraging the knowledge of their time and space complexity.

The experimental evaluation of the proposed architecture highlighted significant improvements over the state-of-the-art software, hardware-software, and hardware implementations of BIKE from the literature. On the one hand, compared to the reference software implementation, which exploits the Intel AVX2 extension on desktop-class CPUs, AES-128- and AES-192-equivalent security instances of the proposed architecture provide performance speedups of 1.77× and 1.98×, respectively. On the other hand, the proposed FPGA-based BIKE architecture also outperforms the other hardware implementations available from literature, including both HLS-generated and human-designed ones, and provides a speedup over the fastest state-of-the-art FPGA-based instance of more than six times.

References

1. Agosta G, Aldinucci M, Alvarez C, Ammendola R, Arfat Y, Beaumont O, Bernaschi M, Biagioni A, Boccali T, Bramas B, Brandolese C, Cantalupo B, Carrozzo M, Cattaneo D, Celestini A, Celino M, Colonnelli I, Cretaro P, D'Ambra P, Danelutto M, Esposito R, Eyraud-Dubois L, Filgueras A, Fornaciari W, Frezza O, Galimberti A, Giacomini F, Goglin B, Gregori D, Guermouche A, Iannone F, Kulczewski M, Lo Cicero F, Lonardo A, Martinelli AR, Martinelli M, Martorell X, Massari G, Montangero S, Mittone G, Namyst R, Oleksiak A, Palazzari P, Paolucci PS, Reghenzani F, Rossi C, Saponara S, Simula F, Terraneo F, Thibault S, Torquati M, Turisini M, Vicini P, Vidal M, Zoni D, Zummo G (2022) Towards extreme scale technologies and accelerators for eurohpc hw/sw supercomputing applications for exascale: the textarossa approach. Microprocess Microsyst 95:104679. https://doi.org/10.1016/j.micpro.2022.104679. https://www.sciencedirect.com/science/article/pii/S0141933122002095

2. Amazon Web Services - Labs: Additional implementation of bike (bit flipping key encapsulation). https://github.com/awslabs/bike-kem (2020)
3. Aragon, N., Barreto PSLM, Bettaieb S, Bidoux L, Blazy O, Deneuville JC, Gaborit P, Gueron S, Güneysu T, Melchor CA, Misoczki R, Persichetti E, Sendrier N, Tillich JP, Vasseur V, Zémor G (2017) BIKE website. https://www.bikesuite.org/
4. Barenghi A, Fornaciari W, Galimberti A, Pelosi G, Zoni D (2019) Evaluating the trade-offs in the hardware design of the ledacrypt encryption functions. In: 2019 26th IEEE international conference on electronics, circuits and systems (ICECS), pp 739–742. https://doi.org/10.1109/ICECS46596.2019.8964882
5. Bernstein DJ (2006) Curve25519: new diffie-hellman speed records. In: Yung M, Dodis Y, Kiayias A, Malkin T (eds) Public key cryptography–PKC 2006. Springer, Berlin, pp 207–228
6. Bertoni G, Daemen J, Peeters M, Van Assche G, Van Keer R (2011) Keccak implementation overview. https://keccak.team/obsolete/Keccak-implementation-3.1.pdf
7. Chen MS, Chou T, Krausz M (2021) Optimizing bike for the intel haswell and arm cortex-m4. IACR Trans Cryptogr Hardw Embed Syst 2021(3):97–124. https://doi.org/10.46586/tches.v2021.i3.97-124, https://tches.iacr.org/index.php/TCHES/article/view/8969
8. Chen MS, Güneysu T, Krausz M, Thoma JP (2022) Carry-less to bike faster. In: Ateniese G, Venturi D (eds) Applied cryptography and network security. Springer International Publishing, Cham, pp 833–852
9. Comba PG (1990) Exponentiation cryptosystems on the IBM PC. IBM Syst J 29(4):526–538. https://doi.org/10.1147/sj.294.0526
10. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory 22(6):644–654. https://doi.org/10.1109/TIT.1976.1055638
11. Drucker N, Gueron S, Kostic D (2020) Qc-mdpc decoders with several shades of gray. In: Ding J, Tillich JP (eds) Post-quantum cryptography. Springer International Publishing, Cham, pp 35–50
12. Dworkin M (2015) Sha-3 standard: permutation-based hash and extendable-output functions. https://doi.org/10.6028/NIST.FIPS.202
13. Fornaciari W, Agosta G, Cattaneo D, Denisov L, Galimberti A, Magnani G, Zoni D (2023) Hardware and software support for mixed precision computing: a roadmap for embedded and hpc systems. In: 2023 design, automation & test in Europe conference & exhibition (DATE), pp 1–6. https://doi.org/10.23919/DATE56975.2023.10137092
14. Galimberti A, Galli D, Montanaro G, Fornaciari W, Zoni D (2022) FPGA implementation of bike for quantum-resistant TLS. In: 2022 25th euromicro conference on digital system design (DSD), pp 539–547. https://doi.org/10.1109/DSD57027.2022.00078
15. Galimberti A, Galli D, Montanaro G, Fornaciari W, Zoni D (2022) On the use of hardware accelerators in qc-mdpc code-based cryptography. In: Proceedings of the 19th ACM international conference on computing frontiers. CF '22, Association for Computing Machinery, New York, NY, USA, pp 193-194. https://doi.org/10.1145/3528416.3530243, https://doi.org/10.1145/3528416.3530243
16. Galimberti A, Montanaro G, Fornaciari W, Zoni D (2023) An evaluation of the state-of-the-art software and hardware implementations of BIKE. In: Bispo Ja, Charles HP, Cherubin S, Massari G (eds) 14th workshop on parallel programming and run-time management techniques for many-core architectures and 12th workshop on design tools and architectures for multicore embedded computing platforms (PARMA-DITAM 2023). Open Access Series in Informatics (OASIcs), vol 107. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 4:1–4:12. 10.4230/OASIcs.PARMA-DITAM.2023.4, https://drops.dagstuhl.de/opus/volltexte/2023/17724
17. Galimberti A, Montanaro G, Zoni D (2022) Efficient and scalable FPGA design of GF(2m) inversion for post-quantum cryptosystems. IEEE Trans Comput 71(12):3295–3307. https://doi.org/10.1109/TC.2022.3149422
18. Galli D, Galimberti A, Fornaciari W, Zoni D (2022) On the effectiveness of true random number generators implemented on FPGAs. In: Orailoglu A, Reichenbach M, Jung M (eds) Embedded computer systems: architectures, modeling, and simulation. Springer International Publishing, Cham, pp 315–326

19. Itoh T, Tsujii S (1988) A fast algorithm for computing multiplicative inverses in GF(2m) using normal bases. Inf Comput 78(3):171–177. https://doi.org/10.1016/0890-5401(88)90024-7. https://www.sciencedirect.com/science/article/pii/S0141933122002095
20. Karatsuba A, Ofman Y (1962) Multiplication of many-digital numbers by automatic computers. Proc USSR Acad Sci 145:293–294
21. McEliece RJ (1978) A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, pp 114–116 (1978)
22. Micciancio D, Regev O (2009) Lattice-based cryptography. In: Post-quantum cryptography, pp 147–191. Springer (2009)
23. Montanaro G, Galimberti A, Colizzi E, Zoni D (2022) Hardware-software co-design of bike with hls-generated accelerators. In: 2022 29th IEEE international conference on electronics, circuits and systems (ICECS), pp 1–4. https://doi.org/10.1109/ICECS202256217.2022.9970992
24. National Institute of Standards and Technology (NIST)—U.S. Department of Commerce: Nistir 8309, status report on the second round of the nist post-quantum cryptography standardization process (2020). https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf
25. National Institute of Standards and Technology (NIST)—U.S. Department of Commerce: Nistir 8413, status report on the third round of the nist post-quantum cryptography standardization process. https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf (2022). 10.6028/NIST.IR.8413
26. Nejatollahi H, Dutt N, Ray S, Regazzoni F, Banerjee I, Cammarota R (2019) Post-quantum lattice-based cryptography implementations: a survey. ACM Comput Surv 51(6). https://doi.org/10.1145/3292548, https://doi.org/10.1145/3292548
27. Peikert C (2009) Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the forty-first annual ACM symposium on theory of computing. STOC '09, Association for Computing Machinery, New York, NY, USA, pp 333–342. https://doi.org/10.1145/1536414.1536461, https://doi.org/10.1145/1536414.1536461
28. Richter-Brockmann J, Chen MS, Ghosh S, Güneysu (2021) Racing bike: improved polynomial multiplication and inversion in hardware. Cryptology ePrint Archive, Paper 2021/1344. https://eprint.iacr.org/2021/1344
29. Richter-Brockmann J, Mono J, Güneysu T (2021) Folding bike: scalable hardware implementation for reconfigurable devices. IEEE Trans Comput. https://doi.org/10.1109/TC.2021.3078294
30. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126. https://doi.org/10.1145/359340.359342
31. Zoni D, Galimberti A, Fornaciari W (2020) Efficient and scalable FPGA-oriented design of QC-LDPC bit-flipping decoders for post-quantum cryptography. IEEE Access 8:163419–163433. https://doi.org/10.1109/ACCESS.2020.3020262
32. Zoni D, Galimberti A, Fornaciari W (2020) Flexible and scalable FPGA-oriented design of multipliers for large binary polynomials. IEEE Access 8:75809–75821. https://doi.org/10.1109/ACCESS.2020.2989423
33. Zoni D, Galimberti A (2022) Cost-effective fixed-point hardware support for risc-v embedded systems. J Syst Arch 126:102476. https://doi.org/10.1016/j.sysarc.2022.102476, www.sciencedirect.com/science/article/pii/S1383762122000595
34. Zoni D, Galimberti A, Fornaciari W (2021) An FPU design template to optimize the accuracy-efficiency-area trade-off. Sustain Comput: Inform Syst 29:100450. https://doi.org/10.1016/j.suscom.2020.100450, www.sciencedirect.com/science/article/pii/S2210537920301761
35. Zoni D, Galimberti A, Fornaciari W (2023) A survey on run-time power monitors at the edge. ACM Comput Surv. https://doi.org/10.1145/3593044