HLS-based acceleration of the BIKE post-quantum KEM on embedded-class heterogeneous SoCs

Andrea Galimberti
DEIB
Politecnico di Milano
Milano, Italy
andrea.galimberti@polimi.it

Gabriele Montanaro
DEIB
Politecnico di Milano
Milano, Italy
gabriele.montanaro@polimi.it

Davide Zoni
DEIB
Politecnico di Milano
Milano, Italy
davide.zoni@polimi.it

***Abstract*—An effective transition to post-quantum cryptography mandates its deployment on embedded-class devices, guaranteeing adequate performance while satisfying their strict area constraints. This work accelerates BIKE, a QC-MDPC code-based post-quantum KEM, through HLS on embedded-class heterogeneous SoCs that couple a CPU with FPGA programmable logic. The proposed methodology implements HLS-generated accelerators to compute the most time-consuming operations of BIKE, identified by analyzing the software-only execution. The mix of accelerators instantiated in hardware and operations executed in software, as well as the configurable architectural parameters of the former, are then determined, depending on the resources available on the target SoC, to minimize BIKE's execution time. Experiments on AMD Zynq-7000 SoCs highlight a speedup of up to 3.34 times compared to the reference software execution and up to 1.98 times over state-of-the-art HW/SW implementations targeting the same chips.**

***Index Terms*—post-quantum cryptography, QC-MDPC code-based cryptography, BIKE, embedded systems, heterogeneous system-on-chip, hardware accelerators, HLS, FPGA**

I. Introduction

Post-quantum cryptography (PQC) has emerged as a trending research topic due to the threat posed by quantum computers, which are expected to break the currently deployed public-key cryptography solutions. BIKE [1] is a key encapsulation mechanism (KEM) that employs quasi-cyclic moderate-density parity-check (QC-MDPC) codes in a variant of the Neiderreiter scheme and that has the best performance among the remaining candidates in the PQC standardization process led by the USA's National Institute of Standards and Technology (NIST) [2].

The complexity of PQC solutions makes it paramount to optimize their execution to guarantee proper performance even on embedded devices at the edge [3]–[5]. The literature provides the official *x86-64* software implementations of BIKE [1], constant-time versions [6], and works targeting other architectures [7], [8], while state-of-the-art hardware implementations include the official FPGA-based one [9], that instantiates the whole KEM in a unified accelerator, and an FPGA-based architecture [10], split into two cores dedicated to the KEM client and server nodes functionality.

While applications such as PQC get more complex and must be deployed on constrained devices, heterogeneous platforms featuring a CPU and programmable logic have emerged as effective computing solutions in embedded scenarios. The two parts suffer from constrasting drawbacks since the CPU does not provide sufficient performance to execute such applications entirely in software and the FPGA does not contain enough resources to implement them fully in hardware, and ever-tighter time-to-market deadlines must be met.

High-level synthesis (HLS) has emerged to this end as an effective solution for the fast production of digital circuits, although delivering less optimized hardware designs that suffer from inefficient usage of the FPGA resources. Exploiting the shorter design time of HLS and improving performance over software execution within the limited resources of embedded heterogeneous platforms requires thus a careful approach [11].

The lone state-of-the-art work mixing the software and hardware execution of BIKE implements the whole three KEM primitives, i.e., key generation, encapsulation, and decapsulation, in hardware through HLS-generated accelerators [12]. Its hardware-software (HW/SW) co-design approach is limited to executing each primitive either in hardware on a dedicated accelerator or in software on the CPU. Moreover, the results cover only BIKE's instance with AES-128-equivalent security.

Contributions

In this paper, we propose a methodology that, targeting embedded heterogeneous SoCs that feature both a CPU and programmable logic, leverages their limited FPGA resources to implement in hardware, through HLS, solely the most time-consuming operations of BIKE in order to maximize the performance improvement for the overall post-quantum KEM. Depending on the available resources, we select the mix of accelerators to instantiate in hardware and operations to execute in software, as well as the configurable architectural parameters of the former, that minimize BIKE's execution time. The experimental results highlight performance improvements over the state-of-the-art software and HW/SW solutions.

To our knowledge, this is the first work that applies an HLS-based, fine-grained HW/SW approach to the BIKE post-

This work was supported by the European Commission and the Italian Ministry of Enterprises and Made in Italy (MIMIT) under the KDT JU "ISOLDE" project (Grant No. 101112274).

DOI 10.1109/ICECS58634.2023.10382918

TABLE I: Breakdown of the execution times of BIKE, in milliseconds and percentages, on a 32-bit *ARM* CPU. Multiplication and decoding operations are highlighted in **bold**.

KEM primitive	Operation	NIST security level AES-128 [ms]	AES-128 [%]	AES-192 [ms]	AES-192 [%]
Keygen	SHAKE-based PRNG	0.9	0.1%	1.2	0.1%
	Inversion	319.1	39.3%	883.0	41.1%
	→ **Multiplication**	**217.1**	**26.7%**	**696.3**	**32.4%**
	→ Exponentiation	101.9	12.6%	186.7	8.7%
	Multiplication	**12.8**	**1.6%**	**36.6**	**1.7%**
		332.3	40.9%	920.9	42.8%
Encaps	SHAKE-based PRNG	0.9	0.1%	1.2	0.1%
	Multiplication	**12.8**	**1.6%**	**37.2**	**1.7%**
	SHA-3 hash function	0.6	0.1%	1.1	0.1%
	SHA-3 hash function	0.2	$<0.1\%$	0.3	$<0.1\%$
		14.7	1.8%	40.9	1.9%
Decaps	**Decoding**	**463.1**	**57.0%**	**1185.6**	**55.1%**
	SHA-3 hash function	0.6	0.1%	1.1	0.1%
	SHAKE-based PRNG	0.9	0.1%	1.2	0.1%
	SHA-3 hash function	0.2	$<0.1\%$	0.3	$<0.1\%$
		464.8	57.2%	1188.3	55.3%
	Multiplication+Decoding	**705.7**	**86.9%**	**1955.7**	**91.0%**
	Total	812.0	100%	2150.1	100%

quantum KEM and provides results for multiple instances of BIKE with different levels of security.

II. Methodology

The BIKE KEM can be split into three primitives. Key generation produces a public-private key pair, encapsulation generates a shared secret and encrypts it with the public key, and decapsulation retrieves the shared secret with the private key from the exchanged ciphertext. Due to its QC-MDPC code-based nature, BIKE makes wide use of binary polynomial arithmetic operations, preeminently multiplication, exponentiation, and inversion, and of the bit-flipping decoding procedure, in particular the Black-Gray-Flip decoding variant.

The proposed methodology is composed of three steps. After identifying which operations to instantiate in hardware, we generate and optimize the corresponding accelerators through HLS, and finally we integrate them in the heterogeneous SoC and make use of their functionality in the software code.

A. Performance profiling of BIKE software

The first step foresees identifying which portions of BIKE to accelerate in hardware and which to execute in software. Intuitively, we aim to find a small number of operations that occupy a large share of execution time and implement those operations in hardware to speed up their computation.

Table I depicts the performance profile of the execution of the BIKE reference C99 software [1] on an *ARM Cortex-A9* CPU. Notably, bit-flipping decoding and binary polynomial inversion occupy up to 57.0% and 41.1% of the execution time of the whole KEM, respectively. At a finer granularity, binary polynomial inversion is computed as an iterative algorithm composed of binary polynomial multiplications and exponentiations and is primarily dominated by the former. A binary polynomial multiplication also appears in the encapsulation primitive. Cumulatively, bit-flipping decoding and binary polynomial multiplications take 86.9% and 91.0% of the total execution time considering AES-128- and AES-192-equivalent security instances of the BIKE KEM, respectively.

B. High-level synthesis of hardware accelerators

The large fraction of BIKE's execution time covered by Black-Gray-Flip decoding and binary polynomial multiplication operations motivates the design choice to accelerate them through HLS-generated accelerators while keeping the computation of the rest of the KEM in software on the CPU.

Whereas the lone existing state-of-the-art HLS-based HW/SW work designs accelerators that implement the whole KEM primitives of BIKE [12], in this work the HLS effort focuses solely on decoding and multiplication, leveraging all the available FPGA resources for their acceleration.

Binary polynomial multiplication operates on factors with sizes in the order of tens of thousands of bits. The design of the binary polynomial multiplier starts from the C code of a multiplication procedure that leverages the Karatsuba and Comba multiplication algorithms. The Karatsuba multiplication formula is recursively applied a configurable number of times, and, at the innermost level, it makes use of Comba multiplication to compute its partial products. Finally, the Comba multiplier executes its partial products between word-size operands through a combinational multiplier. Configuring the number of Karatsuba recursions enables exploiting the area-performance trade-off. Moreover, unrolling directives in the Comba multiplier allow for reducing latency by setting the length of the combinational multiplier's operands. On the area side, we exploit storage binding and array partitioning HLS directives to use the available resources efficiently, storing the large polynomials to BRAM while reserving flip-flops for variables with sizes up to a few hundred bits.

Black-Gray-Flip decoding is an iterative algorithm that takes a syndrome and the private key as inputs and retrieves an error vector. The complexity of decoding is given by the large size of the code, which is in the order of tens of thousands of bits. The decoding algorithm itself contains multiplications between polynomials. Part of those operations are performed in the Galois field of binary polynomials, while others treat the coefficient of those polynomials as natural arithmetic integer values. On the performance side, we optimize the decoding latency by implementing such multiplications through the same hybrid Karatsuba-Comba approach detailed in the previous paragraph. On the area side, the amount of used FPGA resources is reduced by applying storage binding and array partitioning directives.

C. Integration of hardware accelerators in heterogeneous SoC

Depending on the amount of available FPGA resources, the accelerators to be implemented in the programmable logic of the SoC are selected and parameterized to maximize the performance of the BIKE execution, i.e., minimize the aggregate execution time of the three KEM primitives.

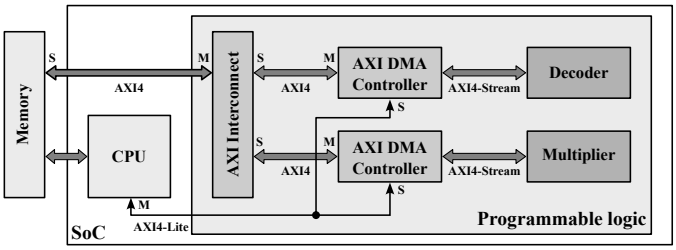

Fig. 1: Top view of the heterogeneous SoC architecture. Legend: **M** AXI master, **S** AXI slave.

Hardware accelerators can be integrated within a heterogeneous SoC that features a CPU and programmable logic as depicted in Figure 1, which shows an example where the FPGA part instantiates both HLS-generated accelerators for binary polynomial multiplication and Black-Gray-Flip decoding.

Each accelerator exposes an AXI4-Stream interface, which connects it to a dedicated DMA controller that provides high-bandwidth access to the DDR memory. DMA controllers are connected to memory through an AXI4 interface, while the CPU accesses their initialization, status, and management registers through AXI4-Lite.

III. Experimental results

In order to provide the fairest comparison, our experimental evaluation adopts a similar setup to the one employed for the reference state-of-the-art HW/SW implementation [12], targeting, as the experimental platform, *AMD Zynq-7000* SoCs.

Zynq-7000 chips pack a 32-bit dual-core *ARM Cortex-A9* processor, running at a 667MHz clock frequency, connected to an external 512MB DDR3 memory and coupled with *Artix-7*-class FPGA programmable logic. We target the *Zynq-7010* SoC, i.e., the smallest *Zynq-7000* chip, whose FPGA part features 17600 look-up table (LUT), 35200 flip-flop (FF), 80 digital signal processing (DSP), and 120 block RAM (BRAM) resources, and the larger *Zynq-7015* one, with 46200 LUT, 92400 FF, 160 DSP, and 190 BRAM resources. *AMD* tools are employed across the whole workflow, including the *Petalinux 2022.1* OS running on the *Zynq-7000* chips, *Vitis HLS 2022.1* for HLS, *Vivado ML 2022.1* for RTL synthesis and implementation, and *Vitis 2022.1* for programming.

The full AXI4 interface connecting the AXI interconnect module to memory is mapped on the HP0 high-performance dedicated bus, while the AXI4-Lite channel between the AXI DMA controller modules and the CPU is mapped on the GP0 general-purpose bus. The accelerators, interconnect, and DMA controllers are implemented at a 100MHz clock frequency.

The executed software, enhanced by calls to the dedicated hardware accelerators, is the reference C99 implementation from the official BIKE NIST submission [1], which was also employed as the initial source code within the HLS process.

We evaluate the proposed architecture in comparison with the state-of-the-art HW/SW proposal, which works at primitive-level granularity through either HLS-generated accelerators or the CPU [12], and the state-of-the-art software execution of BIKE on the CPU of *Zynq-7000* chips [1].

TABLE II: Breakdown of the FPGA resource utilization. **Our** totals include the additional logic to integrate the accelerators in the SoC. Legend: **L** LUT, **F** FF, **D** DSP, **B** BRAM.

Design (Target chip)	KEM prim. or op.	NIST security level AES-128 L	F	D	B	AES-192 L	F	D	B
Ref. HW/SW (Zynq-7010)	Keygen	13567	11621	0	40	N/A	N/A	N/A	N/A
		13567	11621	0	40				
Ref. HW/SW (Zynq-7015)	Decaps	37160	38118	35	90	N/A	N/A	N/A	N/A
		37160	38118	35	90				
Ref. HW/SW (Zynq-7020)	Keygen	13567	11621	0	40	N/A	N/A	N/A	N/A
	Decaps	37160	38118	35	90				
		50727	49739	35	130				
Our (Zynq-7010)	Decod.	12348	11897	14	76	14988	11165	14	125
		12986	11977	14	76	15616	12250	14	125
Our (Zynq-7015)	Mult.	17967	16491	0	36	18760	13691	0	40
	Decod.	24165	21740	14	88	25134	19095	14	133
		42777	39317	14	124	44529	33877	14	173

The FPGA resource utilization of our solutions is obtained after RTL implementation, while the execution times are measured at run time, using the *clock_gettime* function to collect the CPU time at the beginning and the end of the software program or portions of the latter.

A. Area results

Table II details the utilization of the FPGA resources of the proposed solutions and compares it to the reference ones, listing the occupied resources by breaking them down in the three KEM primitives or the HLS-instantiated operations.

Our designs vary to fit within the resources available on the different chips. The *Zynq-7010* instances implement only the decoder, chosen over the multiplier due to its largest reduction effect on the overall execution time. On the contrary, the *Zynq-7015* instances implement both decoding and multiplication accelerators, as in the architecture depicted in Figure 1. The parameters of the HLS designs also differ from each other. Less parallelism is exploited on the smaller *Zynq-7010* instances compared to the larger *Zynq-7015* ones. Moreover, AES-128- and AES-192-equivalent security instances on *Zynq-7015* implement a different number of Karatsuba recursions in the multiplier, five and four, respectively.

Our *Zynq-7015* design occupies a similar number of LUTs and FFs and less DSPs than the reference *Zynq-7015* one, as well as -16% LUT, -21% FF, -60% DSP, and -5% BRAM resources compared to the reference *Zynq-7020* one. Both our *Zynq-7010* and *-7015* instances use more BRAM resources than the corresponding reference ones, thus exploiting more effectively the memories provided by the heterogeneous chips.

B. Performance results

Performance is measured as the execution time of BIKE, i.e., the aggregate execution time of the three KEM primitives.

Table III compares the proposed solution to the reference works, listing, for each design and each considered security level of BIKE, the execution time in milliseconds of the whole

TABLE III: Breakdown of the execution times, reported in milliseconds, at primitive granularity for the BIKE KEM.

Design (Target chip)	KEM primitive	NIST security level AES-128	AES-192
Ref. SW [1] (Zynq-7010/15/20)	Keygen	332	921
	Encaps	15	41
	Decaps	465	1 188
	Total	812	2 150
Ref. HW/SW [12] (Zynq-7010)	Keygen	138	N/A
	Encaps	15	
	Decaps	465	
	Total	617	
Ref. HW/SW [12] (Zynq-7015)	Keygen	332	N/A
	Encaps	15	
	Decaps	135	
	Total	482	
Ref. HW/SW [12] (Zynq-7020)	Keygen	138	N/A
	Encaps	15	
	Decaps	135	
	Total	288	
Our (Zynq-7010)	Keygen	332	921
	Encaps	15	41
	Decaps	138	484
	Total	485	1 446
Our (Zynq-7015)	Keygen	145	423
	Encaps	5	12
	Decaps	93	311
	Total	243	747

BIKE KEM and the breakdown showing the execution times of the three KEM primitives.

Figure 2 details the speedup of the reference and proposed HW/SW solutions over software execution, i.e., the ratio between the software execution time and the execution time of each HW/SW instance. Compared to software AES-128- and AES-192-equivalent security BIKE instances, our solutions provide speedups of up to 3.34× and 2.88×, respectively.

Our *Zynq-7010* solution provides a 1.27× speedup over the reference HW/SW one targeting the same chip, while our *Zynq-7015* one results in a 1.98× speedup over the corresponding state-of-the-art reference and a 1.18× speedup over the *Zynq-7020* one. In addition, the performance of the proposed *Zynq-7010* instance, which implements the decoding operation in hardware, is almost equivalent to the reference *Zynq-7015* one, which instantiates the whole decapsulation primitive, demonstrating the effectiveness of instantiating in hardware solely the most time-consuming operations and dedicating all the FPGA resources to improving their performance.

IV. Conclusions

This work proposed an approach aimed to support the deployment and execution of a complex application such as BIKE on embedded heterogeneous platforms featuring both a CPU and FPGA logic, leveraging HLS to generate accelerators for the operations that require the vast majority of BIKE's execution time, identified through a software profiling phase.

The proposed methodology finds the mix of accelerators to instantiate in hardware and operations to execute in software, as well as the configurable architectural parameters of the former, that minimize the execution time while satisfying the area constraints. Applying HLS to the most time-consuming

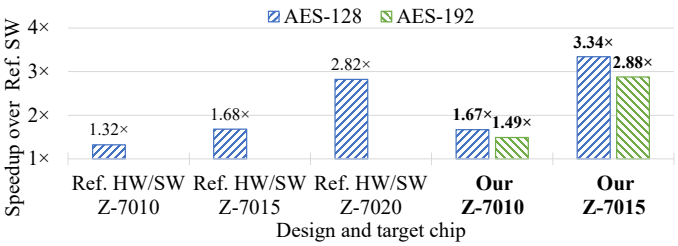

Fig. 2: Speedup over the reference software execution.

operations and computing the other ones in software enabled indeed an effective usage of the limited resources, resulting in better performance than the existing HW/SW solutions.

Experiments targeting *AMD Zynq-7000* SoCs demonstrated a more efficient usage of FPGA resources compared to the state-of-the-art HW/SW reference, which implemented in hardware entire KEM primitives rather than solely the main operations as proposed in this work. The efficient usage of FPGA resources allowed, thus, extracting better performance from the same target SoCs. The proposed solutions reduced the execution time by up to 3.34× and 2.88× for AES-128- and AES-192-equivalent security instances, respectively, compared to the software reference and up to 1.98× compared to reference HW/SW implementations targeting the same chips.

References

[1] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Güneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zémor, "BIKE website," https://www.bikesuite.org/, 2021.

[2] 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.

[3] D. Zoni, A. Galimberti, and W. Fornaciari, "Flexible and scalable fpga-oriented design of multipliers for large binary polynomials," *IEEE Access*, vol. 8, pp. 75 809–75 821, 2020.

[4] ——, "Efficient and scalable fpga-oriented design of qc-ldpc bit-flipping decoders for post-quantum cryptography," *IEEE Access*, vol. 8, pp. 163 419–163 433, 2020.

[5] A. Galimberti, G. Montanaro, and D. Zoni, "Efficient and scalable fpga design of gf(2m) inversion for post-quantum cryptosystems," *IEEE Transactions on Computers*, pp. 1–1, 2022.

[6] Amazon Web Services - Labs, "Additional implementation of bike (bit flipping key encapsulation)," https://github.com/awslabs/bike-kem, 2020.

[7] M.-S. Chen, T. Chou, and M. Krausz, "Optimizing bike for the intel haswell and arm cortex-m4," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, p. 97–124, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8969

[8] M.-S. Chen, T. Güneysu, M. Krausz, and J. P. Thoma, "Carry-less to bike faster," in *Applied Cryptography and Network Security*, G. Ateniese and D. Venturi, Eds. Cham: Springer International Publishing, 2022, pp. 833–852.

[9] J. Richter-Brockmann, M.-S. Chen, S. Ghosh, and T. Güneysu, "Racing bike: Improved polynomial multiplication and inversion in hardware," Cryptology ePrint Archive, Paper 2021/1344, 2021. [Online]. Available: https://eprint.iacr.org/2021/1344

[10] A. Galimberti, D. Galli, G. Montanaro, W. Fornaciari, and D. Zoni, "Fpga implementation of bike for quantum-resistant tls," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 539–547.

[11] A. Galimberti, G. Montanaro, W. Fornaciari, and D. Zoni, "An Evaluation of the State-Of-The-Art Software and Hardware Implementations of BIKE," in *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)*, ser. Open Access Series in Informatics (OASIcs), J. a. Bispo, H.-P. Charles, S. Cherubin, and G. Massari, Eds., vol. 107. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 4:1–4:12. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2023/17724

[12] G. Montanaro, A. Galimberti, E. Colizzi, and D. Zoni, "Hardware-software co-design of bike with hls-generated accelerators," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2022, pp. 1–4.