On the Efficiency and Security of Quantum-resistant Key Establishment Mechanisms on FPGA Platforms

Lukas Malina[1][a], Sara Ricci[1][b], Patrik Dobias[1], Petr Jedlicka[1][c], Jan Hajny[1][d] and Kim-Kwang Raymond Choo[2][e]

[1]*Department of Telecommunications, Brno University of Technology, Brno, Czech Republic*

[2]*Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, U.S.A.*

Keywords: FPGA, Hardware Implementation, Key Establishment, Post-quantum Cryptography, Security, VHDL.

Abstract: The importance of designing efficient and secure post-quantum cryptographic algorithms is reinforced in the recent National Institute of Standards and Technology (NIST)'s Post-Quantum Cryptography (PQC) competitions. Seeking to complement existing studies that evaluate the performance of various PQC algorithms, we explore current hardware implementations of third-round finalist key-establishment algorithms (i.e., Kyber, McEliece, NTRU, and SABER) and the five alternate algorithms (i.e., BIKE, FrodoKEM, HQC, NTRU Prime, and SIKE) on Field Programmable Gate Array (FPGA) platforms. Further, we present our pure-VHDL implementation of Kyber and compare it with the hardware implementations of the NIST finalists. Our design offers one universal Kyber component that can operate in 6 different modes. The evaluation findings show that our pure-VHDL Kyber provides less latency than current VHDL-based implementations.

1 INTRODUCTION

In recent years, there have been efforts from the research community to design quantum-resistant key establishment, public-key encryption, and digital signature protocols and schemes, as well as implementing protocols from the NIST's PQC competition on FPGA platforms (Nejatollahi et al., 2019; Malina et al., 2019; Basu et al., 2019; Zhang et al., 2020; Bisheh-Niasar et al., 2021; Dang et al., 2020; Dang et al., 2021). It is expected that such quantum-resistant schemes will eventually replace those based on the discrete logarithm, RSA and/or other conventional assumptions that are known to be vulnerable to attacks from a functional quantum computer. It is also known that post-quantum cryptography schemes are usually more robust in their parameters and cycles. Therefore, hardware-based implementations of PQC schemes should facilitate acceleration, reduce latency, and increase the number of operations per second.

[a] https://orcid.org/0000-0002-7208-2514
[b] https://orcid.org/0000-0003-0842-4951
[c] https://orcid.org/0000-0003-0833-8068
[d] https://orcid.org/0000-0003-2831-1073
[e] https://orcid.org/0000-0001-9208-5336

1.1 Related Work

In the systematization of knowledge (SOK) study of (Basu et al., 2019), the High-Level Synthesis (HLS)-based hardware design methodology was used to implement 26 NIST PQC Competition Round 2 KEM and Signature algorithms on FPGA platforms, with the aim of assessing the latency and hardware requirements associated with their implementations. However, the implementation did not use the pure VHSIC Hardware Description Language (VHDL), a widely used hardware description language, or evaluate the security levels. In a more recent study, (Bisheh-Niasar et al., 2021) empirically evaluated the performance of SIKEp434 (at Virtex-7), Frodo-640 (at Artix-7), LightSaber (at UltraScale+) and their Kyber implementation (at Artix-7). Their findings suggested that Frodo-640 requires the lowest area (6.8k LUTs, 16 DSPs) but their Kyber implementation is the fastest with 31 μs (12/19 μs encapsulation/decapsulation).

In the evaluation work undertaken by (Dang et al., 2021), the authors focused on high-speed hardware architectures for four lattice-based CCA-secure Key Encapsulation Mechanisms (KEMs), representing 3 NIST PQC finalists: CRYSTALS-Kyber, NTRU (with two distinct variants, NTRU-HPS and NTRU-HRSS), and Saber. Their analysis revealed that all

Malina, L., Ricci, S., Dobias, P., Jedlicka, P., Hajny, J. and Choo, K.
On the Efficiency and Security of Quantum-resistant Key Establishment Mechanisms on FPGA Platforms.
DOI: 10.5220/0011294200003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 605-613
ISBN: 978-989-758-590-6; ISSN: 2184-7711

four NIST PQC finalists significantly outperform the other alternate candidates in terms of speed on the hardware platforms. They also found that SABER and Kyber are more efficient in the main phases in comparison to both NTRU and McEliece schemes. However, the security of KEMs is not within the scope of their evaluation.

1.2 Contributions

Seeking to contribute to the knowledge gaps, we will focus on both efficiency and security of KEM PQC NIST finalists and especially on the hardware implementations of the CRYSTALS-Kyber scheme in this paper. Specifically, we seek to answer the following research questions: (1) What is the current state of hardware-based PQC KEMs implementations on FPGA platforms? (2) Which PQC KEM finalist(s) is/are most suitable for FPGA platforms in terms of efficiency and security? (3) How efficient can Kyber be when implemented on FPGA platforms, and if it is coded solely in VHDL?

To answer the first two questions, we will map existing hardware implementations of all PQC KEM NIST finalists and discuss the current security attacks and potential problems of PQC-KEM hardware implementations in Sections 3 and 4. To answer the third question, we will present and discuss the findings of our pure VHDL-implementation of the CRYSTALS-Kyber scheme (Avanzi et al., 2017), as well as present a comparative summary of its performance with other related works.

2 BACKGROUND: HARDWARE-BASED CRYPTOGRAPHY AND FPGA PLATFORMS

Hardware-based implementations on FPGA platforms are a common trend, partly due to their potential in outperforming software implementations in speed, power consumption, and/or energy usage. In addition, there have been attempts to explore hardware-based implementations that are high-speed or lightweight.

Generally, high-speed implementations seek to minimize the execution times of major operations via protocol optimization and operations/subcomponents parallelization, whereas lightweight implementations tend to achieve minimum resource utilization (by not exceeding predefined maximum execution time). Resource utilization can be represented by the number of Logic Cells, Look-Up Tables (LUTs), Flip-Flops (FFs), Digital Signal Processor (DSP) slices, and Block Random Access Memories (BRAMs). It is also observed that many FPGA-based applications seek to find a balance between efficiency and having minimal hardware resources.

Table 1 describes the hardware requirements of chosen Xilinx FPGA platforms in various hardware implementation studies. We note that small and medium FPGA platforms are typically used for accelerating some particular operations and processes at end nodes (embedded devices, user nodes, routers, servers, etc.), where these boards provide the acceleration of particular cryptography schemes. Large FPGA platforms are often used for high-speed data processing and high-speed communication. These platforms can facilitate dense processing of various computationally heavy cryptography schemes and protocols, e.g., in cloud-based solutions, backbone active network devices, and central servers.

3 PQC KEM SCHEMES: THIRD-ROUND NIST FINALISTS AND ALTERNATIVES

In this section, the current size requirements of the NIST KEM finalist are summarized. Memory consumption is an essential feature that needs to be taken into account, particularly when we consider resource-constrained devices since if the scheme is too memory-demanding, it cannot be directly implemented on such devices. Table 2 describes the private key, the public key, and the ciphertext sizes of NIST KEM schemes which are given in bytes. Note that only the SIKE scheme presents memory consumption close to traditional cryptography standards, followed by Kyber and SABER schemes. In general, lattice-based schemes require smaller key-pair and ciphertext sizes compared to code-based schemes.

From Tables 1 and 2, we observe that for a NIST KEM protocol to be run on an FPGA platform, the device needs to store at least the protocol's keys. In other words, the McEliece scheme requires medium- to large-sized platforms. In fact, Spartan-6 XC6SLX45T and Artix-7 XA7A12T have 261 000 bytes and 90 000 bytes of total memory, respectively, and McEliece key size exceeds these values. The total available memory can be computed by multiplying the number of BRAMs for the block size. It is essential to consider that the keys are just one of the components that need to be stored in the memory. During the

Table 1: Hardware specifications of Xilinx FPGA platforms: A comparative summary.

Platform	Logic Cells / LUTs	FFs	DSP	BRAMs / Block Size	Used in
Large platforms					
Virtex UltraScale+ XCVU7P	**1 724 100 / 788 160**	**1 576 320**	**4 560**	**1 440 / 36 Kb**	**Our work**
Zynq UltraScale+ XCZU9EG	599 550 / 274 080	548 160	2 520	912 / 36 Kb	(Dang et al., 2020)
Virtex-7 VC707 XC7VX485T	485 760 / 303 600	607 200	2 800	1 030 / 36 Kb	(Huang et al., 2020)
Zynq UltraScale+ ZU7EV-3	504 000 / 230 400	460 800	1 728	312 (96 Ultra BRAMs)	(Dang et al., 2021)
Small and medium platforms					
Artix-7 XC7A200T	215 360 / 134 600	269 200	740	365 / 36 Kb	(Chen et al., 2021)
Artix-7 XC7A100T	101 440 / 63 400	126 800	240	135 / 36 Kb	(Henson et al., 2021)
Artix-7 XA7A12T	12 800 / 8 000	16 000	40	20 / 36 Kb	(Xing and Li, 2021)
Spartan-6 XC6SLX45T	43 661 / 27 288	54 576	58	116 / 18 Kb	(Chen et al., 2020)

Table 2: Traditional cryptography memory consumption compared with NIST PQC KEM finalists and alternate candidates with 128-bit level of security. The sizes are given in bytes.

Traditional Cryptography				
Scheme		**Total Key Size**		**Ciphertext**
RSA encryption		384		384
Encryption/KEM NIST PQC Finalists				
Scheme	**Type**	**Private Key**	**Public Key**	**Ciphertext**
Kyber	lattice	1 632 (or 32)	800	768
McEliece	code	6 452	261 120	128
NTRU	lattice	1 452	1 138	1 138
SABER	lattice	1 568	672	736
KEM NIST PQC Alternate Candidates				
BIKE	code	249	2 541	2 541
FrodoKEM	lattice	19 888	9 616	9 720
HQC	code	252	6 170	6 234
NTRU Prime	lattice	1 125	897	1 025
SIKE	isogeny	374	330	346

Table 3: Hardware/Software co-design, HLS, and pure Hardware implementations of NIST PQC KEM finalists and alternate candidates. The table shows the number of found implementations per scheme with the year of publication.

Encryption/KEM NIST PQC Finalists			
Scheme	**HW/SW**	**HLS method**	**Pure Hardware**
Kyber	3 (2019, 2020)	2 (2019, 2021)	2 (2021)
McEliece	1 (2021)	1 (2019)	✗
NTRU	1 (2018)	✗	1 (2018)
SABER	3 (2019, 2020)	1 (2019)	3 (2020, 2021)
KEM NIST PQC Alternate Candidates			
BIKE	✗	✗	3 (2020, 2021)
FrodoKEM	2 (2019)	1 (2019)	2 (2019, 2021)
HQC	✗	1 (2018)	✗
NTRU Prime	✗	✗	2 (2020, 2021)
SIKE	1 (2020)	✗	3 (2017, 2020, 2021)

Note: ✗– no implementation could be found.

protocol, there could be more temporary components that need more memory. Moreover, the numbers of available LUTs, FFs, and DSP need to be taken into account. See Section 5 for a more detailed analysis.

3.1 Quantum Resistant Key Establishment Mechanisms at FPGA

Several partial or whole FPGA implementations of NIST KEM schemes are currently published. The designs can be split into three types, namely: HLS-based design (e.g., ANSI C/C++, and Matlab), software-hardware (SW-HW) co-design, and RTL-based design (e.g., VHDL, Verilog, Chisel). Hardware implementations generally outperform software ones in at least one of the following aspects: latency (operation speed), power consumption, or energy usage. We refer the interested reader to (Dang et al., 2020) for more information.

(Basu et al., 2019) implement 11 NIST PQC semifinalists by using the HLS method and assess them on Virtex-7. Among the NIST finalist schemes, the article covers Kyber, McEliece, SABER, and FrodoKEM. CRYSTALS-Kyber is recognized as the fastest KEM under pipelining directives.

CRYSTALS-Kyber. (Chen et al., 2021) proposed a polynomial ring processor for Kyber by using the HLS method for Artix-7. They developed an optimized Number Theoretic Transform (NTT) that uses a convolution-based polynomial multiplier. (Banerjee et al., 2019) proposed a software-hardware co-design approach based on the lattice cryptography processor with configurable parameters (i.e., sapphire) that can be used to speed up lattice-based protocols such as Kyber and FrodoKEM. They considered an Opal Kelly XEM7001 FPGA development board. (Dang et al., 2020) presented a SW-HW co-design approach to implementing three NIST semifinalists: Kyber, NewHope, and Round5 schemes. They combined C code with Register-Transfer Level (RTL) design methodology on Artix-7. Also, (Fritzmann et al., 2020) used a SW-HW co-design with tightly coupled accelerators to speed up lattice-based cryptography. They evaluated the accelerator by comparing

Kyber performances. (Huang et al., 2020) proposed a Kyber hardware design with NTT optimization with Gentlemen-Sande butterfly on the XC7A200T and XC6SLX45T FPGA platforms. However, no specification on the used language was given. In 2021, (Xing and Li, 2021) and (Dang et al., 2021) developed a pure VHDL implementation of Kyber. (Xing and Li, 2021) used the XC7A12TCPG238-1 FPGA platform, whereas (Dang et al., 2021) used Artix-7 XC7A200T-3 and Zynq UltraScale+ ZU7EV-3.

McEliece. (Kostalabros et al., 2021) proposed a SW-H co-design acceleration of the Classic McEliece KEM scheme on the ZCU102 heterogeneous CPU+FPGA platform. (Wang et al., 2018) provided the fully-RTL (Verilog) Niederreiter cryptosystem implementation on the Virtex-6 XC6VLX240T. This cryptosystem is the dual variant of the Classic McEliece scheme. However, their solution does not provide an end-to-end KEM implementation and cannot be compared with those of (Basu et al., 2019; Kostalabros et al., 2021).

NTRU. Three variants of the NTRU cryptosystem were sent to NIST for standardization (Schanck, 2018), namely: NTRUEncrypt, NTRU-HRSS-KEM, and NTRU Prime. In the third round, NTRU-Encrypt and NTRU-HRSS-KEM were merged in NTRU, which is one of the finalist, whereas NTRU Prime is an alternate candidate. (Fritzmann et al., 2018) presented the first hardware implementation of NTRU using the SVE padding. They also proposed a HW/SW co-design of NTRU where polynomial multiplication and modulo reduction run on hardware, Zynq UltraScale+ MPSoC (ZCU102). (Dang et al., 2021) proposed a hardware implementation of NTRU (i.e, NTRU-HRSS and NTRU HPS scheme) on two FPGA platforms, namely: Artix-7 XC7A200T-3 and Zynq UltraScale+ (ZU7EV-3).

SABER. The HLS method is employed by (Basu et al., 2019) for SABER on the Virtex-7 FPGA platform. We could identify three HW/SW co-designs. (Dang et al., 2019) propose a HW/SW co-design with a speed-up that exceeds a factor of 7 compared to software implementation. They consider the ZCU102 Evaluation Kit, based on the Zynq UltraScale+ MPSoC XCZU9EG2FFVB1156E device. Subsequently, (Mera et al., 2020) proposed a domain-specific co-processor to speed SABER where only the polynomial multiplication, i.e., the most expensive operation, is offloaded to the co-processor to obtain a compact design. They consider a Zynq-7000 ARM/FPGA SoC platform. At last, (Fritzmann et al., 2020) explored tightly coupled accelerators to speed up lattice-based cryptography, which was evaluated on SABER performances. Three fully hardware implementations of SABER were developed by (Roy and Basso, 2020), (Dang et al., 2021) and (Zhu et al., 2021). In particular, (Roy and Basso, 2020) proposed parallel polynomial multiplier architecture to overcome memory access bottlenecks common in lattice polynomial multiplication. They employ UltraScale+ (XCZU9EG-2FFVB1156). (Dang et al., 2021) presented four implementations of SABER, where two of them outperformed the best previous design in terms of resource utilization. At last, (Zhu et al., 2021) proposed an energy-efficient configurable crypto-processor for SABER with two improvements of the Karatsuba framework to reduce polynomial multiplication overhead. They considered Virtex UltraScale+ for their performance analyses.

BIKE. (Reinders et al., 2020) presented the first complete implementation of BIKE on older parameters. However, they designed a simplification of the grey-black decoder that makes it constant time. The Intel Arria 10 FPGA platform was used for the analysis. The first complete hardware design of the current BIKE version was developed by (Richter-Brockmann et al., 2021b). Moreover, the first implementation of the black-gray-flip decoder on hardware, an optimized polynomial inversion module, and a scalable multiplier were introduced. Their implementation can run on a low-cost Artix7 (XC7A35T) and on a high-speed XC7A100T FPGA platforms. (Richter-Brockmann et al., 2021a) improved their previous work by introducing an optimized polynomial multiplier and a novel component for polynomial inversion based on the extended Euclidean algorithm. Implementation results run on Artix-7.

FrodoKEM. (Basu et al., 2019) employed the HLS approach for FrodoKEM on Virtex-7. We identified two HW/SW co-designs. (Dang et al., 2019) presented a HW/SW co-design for FrodoKEM where matrix multiplication and SHAKE sequence generations are offloaded to hardware, ZCU102 Evaluation Kit, based on the Zynq UltraScale+ MPSoC XCZU9EG2FFVB1156E device. Moreover, (Banerjee et al., 2019) proposed a lattice cryptography processor with configurable parameters, namely sapphire, to speed up lattice-based protocols and can be applied to FrodoKEM. The Sapphire crypto-processor was coupled with an efficient RISC-V micro-processor. They considered an Opal Kelly XEM7001 FPGA development board. On the pure hardware implementation, (Howe et al., 2019) proposed an alternative hardware design for FrodoKEM that uses an unrolled Trivium as PRNG on Artix-7. Then, (Howe et al., 2021) optimized FrodoKEM hardware implementation by paralyzing the matrix multiplication operation on Artix-7 (XC7A35T).

HQC. A HLS implementation of HQC was proposed by (Melchor et al., 2018) on an Artix-7 FPGA platform. However, neither HW/SW co-design nor pure hardware implementation could be found.

NTRU Prime. (Marotzke, 2020) presented the first hardware implementation of NTRU Prime scheme. The author focused on optimizing the used sources, and the implementation was run on Zynq Ultrascale+. Following, (Peng et al., 2021) proposed two variants of hardware implementations, namely: a high-speed and high-area one and a slower, low-area one. In order to improve the performance, they developed a new batch inversion for key generation, a high-speed schoolbook polynomial multiplier, an NTT polynomial multiplier, a new DSP-free modular reduction method, and a high-speed radix sorting module, and new en- and decoders. They considered Zynq Ultrascale+.

SIKE. Due to SIKE small key size and generally low performance of isogeny-based scheme, several FPGA hardware implementations were presented. The first hardware proposal is from the author of SIKE scheme, i.e., (Azarderakhsh et al., 2017). They used Virtex-7 (xc7vx690tffg1157-3). (Koziel et al., 2020) proposed a fast isogeny accelerator architecture that is then applied to speed up the SIKE scheme. They employed Artix-7, Virtex-7, and Kintex UltraScale+. At last, (Elkhatib et al., 2021) focused on the improvements Montgomery multiplication algorithm and architecture for prime fields to speed up SIKE that is also improved. They considered Artix-7 and Virtex-7 FPGAs. Moreover, (Massolino et al., 2020) presented a HW/SW co-design implementation of SIKE. They focused on making the protocol compact and scalable.

As shown in Table 3, Kyber and SABER schemes have a higher number of implementations, partly because of their (practical) memory consumption and efficiency. SIKE is also a good candidate for FPGA acceleration due to its memory consumption that is comparable to traditional cryptography and its need for better performance.

4 ON SECURITY OF HARDWARE-BASED IMPLEMENTATIONS OF QUANTUM RESISTANT KEY ESTABLISHMENT MECHANISMS

In this section, we present the recent hardware implementations of finalists that add countermeasures and protection into their design.

CRYSTALS-Kyber. (Jati et al., 2021) focused on configurable Kyber hardware implementation with side-channel protection. The authors claimed that their implementation is the first side-channel attack protected and consumes only 5% of HW resources. Their implementation includes various fault protection techniques such as Fault Detection Hashes (FDH), protecting critical signals using Complementary Duplicate Logic (CDL), and protection against control flow (FSM state) modification. The solution also deploys a solid true random number generator and side-channel protection techniques such as random delays, address randomization, and instruction randomization.

SABER. (Abdulgadir et al., 2021) presented a hardware implementation of the Saber scheme that is resistant against side-channel attacks. Their hardware implementation of SABER deploys the masked SABER.KEM.Decaps phase that contains masked CBD Sampler, protected datapath, and masked logical shifting. The complete protection increases approximately 2.9× LUTs and 1.4× latency compared to the baseline unprotected implementation of SABER.KEM.Decaps. The authors used mainly VHDL for hardware description and Chisel for SHA-3.

McEliece. (Colombier et al., 2022) introduced a message recovery attack on the Classic McEliece cryptosystem. They studied power consumption-based side-channel analysis and machine learning techniques. The error weight t is small and is reportedly the Classic McEliece cryptosystem's main weakness; thus, masking can be used instead to have an error vector having a more general pattern. The attack and countermeasure are only general, and the study does not focus more on hardware-based implementations.

NTRU. (Braun et al., 2018) proposed a compact, secure hardware-based architecture of NTRU for Zynq-7000. The NTRU architecture is secure because it avoids CCA attacks by including the SVES padding scheme, defined in IEEE-1363.1, and the protection against SCA attacks using a constant time convolution.

Currently, there are few first hardware-based implementations of Kyber, SABER, and NTRU that should resist side-channel attacks, but require additional hardware resources. Future studies could focus on the detailed practice verification of these implementations and their optimization.

5 ON HARDWARE-BASED IMPLEMENTATIONS OF CRYSTALS-KYBER

In this section, we present our implementation design of the CRYSTALS-Kyber scheme on FPGA, results and comparison.

5.1 Our Design and Hardware Implementation

Our CRYSTALS-Kyber implementation is written solely in VHDL. We design our implementation to be compact, efficient, and optimized. In order to save HW resources, there is only one universal component that can operate in 6 different modes - CPAPKE.KeyGen, CPAPKE.Enc, CPAPKE.Dec, CCAKEM.KeyGen, CCAKEM.Enc and CCAKEM.Dec. The concrete mode is set through the input interface and is reset once all data are outputted. Our proposed architecture of the universal component and connections of its internal modules is shown in Figure 1. To increase throughput, all coefficients are passed in batches of 4 and processed in a parallel manner. The NTT component used for a polynomial conversion to and from the NTT domain is used from the previous work. We assume that NTT can be further optimized for the Kyber scheme in our future work. To reduce resources utilization, the NTT component is used only once. In the related work of (Dang et al., 2021) NTT is used K times, and operations as polynomials sampling and polynomials multiplications are then performed in parallel while NTT is running to reduce waiting. Each internal part and function e.g. Keccak, NTT was carefully optimized. To achieve high frequency, the design was checked multiple times for critical paths, and techniques like registering were used to remove those.

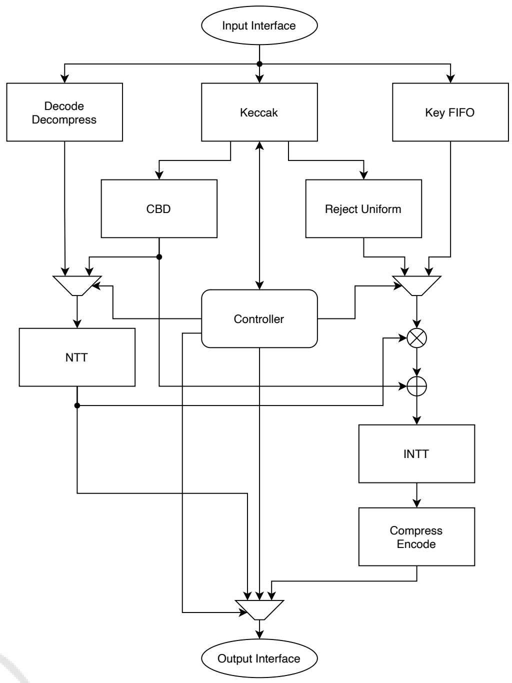

Figure 1: Proposed architecture of universal Kyber component.

5.2 Performance Evaluation and Comparison

Table 4 serves for the performance assessment of our Kyber-768 scheme and all its modes. The results are based on theoretical operating frequency 533.9 MHz obtained using the synthesis targeting the Virtex UltraScale+. Table 5 compares our results with the previous works presenting PQC KEMs at FPGA. We show also our results on the Kintex UltraScale+ with operating frequency 618.8 MHz. *.Enc represents Encapsulation and *.Dec represents Decapsulation. The comparison introduces resources utilization (e.g., LUTs, FFs, DSP slices, and BRAMs) and latency that is based on known frequencies and the number of cycles per operation. The compared results are for setting in the security level 3 which is roughly equivalent to AES-192.

Table 4: Our Kyber-768: Performance of individual modes.

Component	Clock cycles	Op [μs]	# Ops / s
CPAPKE.KeyGen	3297	6.14	162824
CPAPKE.Enc	4059	7.60	131534
CPAPKE.Dec	2339	4.38	228259
CCAKEM.KeyGen	3400	6.36	157029
CCAKEM.Enc	4688	8.78	113886
CCAKEM.Dec	6554	12.27	81461

Table 5: Comparison of PQC KEM NIST finalists for the security level (3). Notation for FPGA families - A7: Artix-7, K7: Kintex-7, VUS+: Virtex UltraScale+, KUS+ Kintex UltraScale+, ZUS+: Zynq UltraScale+. * implemented by the Chisel language.

Work	Component	FPGA	LUT	FF	DSP	BRAM	Clock cycles	Op [μs]
	Kyber.KeyGen						3400	6.36 / 5.49
This work	Kyber.Enc	VUS+ / KUS+	15504	15125	182	16	4688	8.78 / 7.58
	Kyber.Dec						6554	12.27 / 10.59
	Kyber.KeyGen						6316	39.2
(Xing and Li, 2021)	Kyber.Enc	A7	7412	4644	2	3	7925	47.6
	Kyber.Dec						10049	62.3
(Huang et al., 2020)	Kyber.Enc	A7	97085	-	36	200.5	77481	499.88
	Kyber.Dec		110260	-	292	202	102113	658.79
*	Kyber.KeyGen						2600	5.9
(Dang et al., 2021)	Kyber.Enc	ZUS+	10590	10458	6	6.5	3700	8.3
	Kyber.Dec						4900	10.9
	NTRU.KeyGen		50347	44281	45	6.5	67157	268.6
(Dang et al., 2021)	NTRU.Enc	ZUS+	33698	30551	0	5.5	4576	18.3
	NTRU.Dec		38642	33003	45	2.5	10211	24.0
	NTRU.KeyGen.HS		39200	25536	23	33.5	64026	447.7
(Peng et al., 2021)	NTRU.Enc.HS	ZUS+	40879	22382	6	4.5	5007	34.8
	NTRU.Dec.HS		36789	22700	9	3.5	10989	80.2
	SABER.KeyGen		20496	13939	0	1.5	2709	7.3
(Dang et al., 2021)	SABER.Enc	ZUS+	21069	14074	0	1.5	3735	10.1
	SABER.Dec		21342	14233	0	1.5	4682	12.7
	SABER.KeyGen						5453	21.8
(Roy and Basso, 2020)	SABER.Enc	ZUS+	23686	9805	0	2	6618	26.5
	SABER.Dec						8034	32.1
	SABER.Enc		6713	7363	32	0	46705	373.1
(Abdulgadir et al., 2021)	SABER.Dec.U	A7					52758	422.1
	SABER.Dec.P		19299	7363	64	0	72005	576.0
	McEliece460896^{cpa}.KeyGen						5002044	46704.4
(Wang et al., 2018)	McEliece460896^{cpa}.Enc	A7	38669	74858	0	303	3360	31.4
	McEliece460896^{cpa}.Dec						31005	289.5
	McEliece460896^{cpa}.KeyGen						515806	3943.5
(Wang et al., 2018)	McEliece460896^{cpa}.Enc	V7	109484	168939	0	446	3360	25.7
	McEliece460896^{cpa}.Dec						17931	137.1

The most recent (Xing and Li, 2021)'s hardware implementation of Kyber has been designed for the small Artix-7 FPGA chips and achieved interesting results in terms of required hardware resources (7.4k LUTs and 4.6k FFs for all levels). Nevertheless, the frequency of their implementation is only 161 MHz and the execution times of the phases are higher than (Dang et al., 2021) and our implementation. To be noted that (Dang et al., 2021) use Chisel that is an alternative to classic Hardware Description Languages (HDLs), e.g., VHDL, and it adds hardware constructions to the Scala programming language. Dang's Kyber implementation requires fewer clock cycles than our work. Nevertheless, our design is encoded solely in VHDL and is currently the fastest among pure VHDL implementations, to our best knowledge. There are also more Kyber's and other scheme implementations in the literature, but we do not include them in Table 5 as their security level differs. For instance, (Guo et al., 2021)'s HW-based implementation of Kyber-1024 (Sec. level 5) evaluated on the Artix-7 FPGA platform achieves 49.1/52.8/66.0 μs delay when performing KeyGen/encaps/decaps, respectively, with consumption of 7.9k LUTs, 3.6k FFs, 2.3k slices, 4 DSPs and 16 BRAMs. This implementation aims at the low usage of HW-resources and is similar to the (Xing and Li, 2021)'s work, but the latency is less efficient than recent works.

(Jati et al., 2021) present the smallest hardware implementation of Kyber-1024 requiring only 5269 LUTs, 2422 FFs and 250 MHz at Artix-7 where the SHA-3 core is deployed as the co-processor. To be noted that authors also implement multiple side-channel countermeasures, which consume less than 5% of the HW resources. Nevertheless, their implementation is significantly slower than other implementations and performs KeyGen/encaps/decaps operations in 4.59 ms /4.94 ms /4.69 ms. Furthermore, we compare Kyber implementations with other NIST finalists. Table 5 also presents the recent implementations of Saber, NTRU and McEliece460896^{cpa} schemes. (Dang et al., 2021)'s implementation of Saber can be competitive with Kyber in the combination of HW resources and execution times. (Abdulgadir et al., 2021)'s implementation of SABER adds side-channel countermeasures into the decapsulation phase, marked as P (Protected). The countermeasures require additional 12586 LUTs. On the other hand, their unprotected version is lightweight and requires similar numbers of HW resources as balanced Kyber implementations. (Dang et al., 2021)'s implementation of NTRU is comparable to Kyber in the runtimes of the encapsulation and decapsulation phases but requires more HW resources. (Wang et al., 2018)'s implementation of the McEliece460896^{cpa} shows that this scheme is not efficient in the decapsulation phase.

6 CONCLUSIONS

This work studied the current state of hardware-based implementations of PQC KEM algorithms on FPGA platforms, and more specifically on NIST PQC finalists and the Kyber scheme implemented on FPGA platforms. We observed that other than a few studies that focused on PCQ KEMs hardware-based implementations from side-channel attack resilience, most studies generally focused on performance and explored the use of fewer HW resources on FPGA boards. In general, we found that recent HW-based Kyber implementations provide promising results and Kyber is attractive for FPGA implementation. Furthermore, we presented our hardware design of Kyber-768 in VHDL, and showed that our implementation outperforms other pure VHDL Kyber implementations and it is comparable with Dang's efficient implementation of Kyber encoded in Chisel. Our future work will also include side-channel countermeasures in our design, as well as the optimization of hardware resources.

ACKNOWLEDGEMENTS

This work is supported by Ministry of the Interior of the Czech Republic under grant VJ01010008.

REFERENCES

Abdulgadir, A., Mohajerani, K., Dang, V. B., Kaps, J.-P., and Gaj, K. (2021). A lightweight implementation of saber resistant against side-channel attacks. In *International Conference on Cryptology in India*, pages 224–245. Springer.

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2017). Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2:4.

Azarderakhsh, R., Campagna, M., Costello, C., Feo, L., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., et al. (2017). Supersingular isogeny key encapsulation. *Submission to the NIST Post-Quantum Standardization project*, 152:154–155.

Banerjee, U., Ukyab, T. S., and Chandrakasan, A. P. (2019). Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols (extended version). *Cryptology ePrint Archive*.

Basu, K., Soni, D., Nabeel, M., and Karri, R. (2019). Nist post-quantum cryptography-a hardware evaluation study. *Cryptology ePrint Archive*.

Bisheh-Niasar, M., Azarderakhsh, R., and Mozaffari-Kermani, M. (2021). High-speed ntt-based polynomial multiplication accelerator for post-quantum cryptography. In *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, pages 94–101. IEEE.

Braun, K., Fritzmann, T., Maringer, G., Schamberger, T., and Sepúlveda, J. (2018). Secure and compact full ntru hardware implementation. In *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 89–94. IEEE.

Chen, Z., Ma, Y., Chen, T., Lin, J., and Jing, J. (2020). Towards efficient kyber on fpgas: A processor for vector of polynomials. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 247–252. IEEE.

Chen, Z., Ma, Y., Chen, T., Lin, J., and Jing, J. (2021). High-performance area-efficient polynomial ring processor for crystals-kyber on fpgas. *Integration*, 78:25–35.

Colombier, B., Dragoi, V.-F., Cayrel, P.-L., and Grosso, V. (2022). Message-recovery profiled side-channel attack on the classic mceliece cryptosystem. *Cryptology ePrint Archive*.

Dang, V. B., Farahmand, F., Andrzejczak, M., and Gaj, K. (2019). Implementing and benchmarking three lattice-based post-quantum cryptography algorithms using software/hardware codesign. In *2019 International Conference on Field-Programmable Technology (ICFPT)*, pages 206–214. IEEE.

Dang, V. B., Farahmand, F., Andrzejczak, M., Mohajerani, K., Nguyen, D. T., and Gaj, K. (2020). Implementation and benchmarking of round 2 candidates in the nist post-quantum cryptography standardization process using hardware and software/hardware co-design approaches. *Cryptology ePrint Archive: Report 2020/795*.

Dang, V. B., Mohajerani, K., and Gaj, K. (2021). High-speed hardware architectures and fpga benchmarking of crystals-kyber, ntru, and saber. *Cryptology ePrint Archive*.

Elkhatib, R., Azarderakhsh, R., and Mozaffari-Kermani, M. (2021). High-performance fpga accelerator for sike. *IEEE Transactions on Computers*.

Fritzmann, T., Schamberger, T., Frisch, C., Braun, K., Maringer, G., and Sepúlveda, J. (2018). Efficient hardware/software co-design for ntru. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, pages 257–280. Springer.

Fritzmann, T., Sigl, G., and Sepúlveda, J. (2020). Risq-v: Tightly coupled risc-v accelerators for post-quantum cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 239–280.

Guo, W., Li, S., and Kong, L. (2021). An efficient implementation of kyber. *IEEE Transactions on Circuits and Systems II: Express Briefs*.

Henson, T. E., Dawoud, A., and Sherif, A. (2021). Privacy-aware and hardware acceleration-based authentication scheme for internet of drones. In *2021 3rd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pages 130–135. IEEE.

Howe, J., Martinoli, M., Oswald, E., and Regazzoni, F. (2019). Optimised lattice-based key encapsulation in hardware. In *Second NIST Post-Quantum Cryptography Standardization Conference 2019*, page 13.

Howe, J., Martinoli, M., Oswald, E., and Regazzoni, F. (2021). Exploring parallelism to improve the performance of frodokem in hardware. *Journal of Cryptographic Engineering*, 11(4):317–327.

Huang, Y., Huang, M., Lei, Z., and Wu, J. (2020). A pure hardware implementation of crystals-kyber pqc algorithm through resource reuse. *IEICE Electronics Express*, pages 17–20200234.

Jati, A., Gupta, N., Chattopadhyay, A., and Sanadhya, S. K. (2021). A configurable crystals-kyber hardware implementation with side-channel protection. *Cryptology ePrint Archive*.

Kostalabros, V., Ribes-González, J., Farràs, O., Moretó, M., and Hernandez, C. (2021). Hls-based hw/sw co-design of the post-quantum classic mceliece cryptosystem. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 52–59. IEEE.

Koziel, B., Ackie, A.-B., El Khatib, R., Azarderakhsh, R., and Kermani, M. M. (2020). Sike'd up: Fast hardware architectures for supersingular isogeny key encapsulation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(12):4842–4854.

Malina, L., Ricci, S., Dzurenda, P., Smekal, D., Hajny, J., and Gerlich, T. (2019). Towards practical deployment of post-quantum cryptography on constrained platforms and hardware-accelerated platforms. In *International Conference on Information Technology and Communications Security*, pages 109–124. Springer.

Marotzke, A. (2020). A constant time full hardware implementation of streamlined ntru prime. In *International Conference on Smart Card Research and Advanced Applications*, pages 3–17. Springer.

Massolino, P. M. C., Longa, P., Renes, J., and Batina, L. (2020). A compact and scalable hardware/software co-design of sike. *Cryptology ePrint Archive*.

Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Persichetti, E., Zémor, G., and Bourges, I. (2018). Hamming quasi-cyclic (hqc). *NIST PQC Round*, 2:4–13.

Mera, J. M. B., Turan, F., Karmakar, A., Roy, S. S., and Verbauwhede, I. (2020). Compact domain-specific co-processor for accelerating module lattice-based kem. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE.

Nejatollahi, H., Dutt, N., Ray, S., Regazzoni, F., Banerjee, I., and Cammarota, R. (2019). Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys (CSUR)*, 51(6):1–41.

Peng, B.-Y., Marotzke, A., Tsai, M.-H., Yang, B.-Y., and Chen, H.-L. (2021). Streamlined ntru prime on fpga. *Cryptology ePrint Archive*.

Reinders, A. H., Misoczki, R., Ghosh, S., and Sastry, M. R. (2020). Efficient bike hardware design with constant-time decoder. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 197–204. IEEE.

Richter-Brockmann, J., Chen, M.-S., Ghosh, S., and Güneysu, T. (2021a). Racing bike: Improved polynomial multiplication and inversion in hardware. *Cryptology ePrint Archive*.

Richter-Brockmann, J., Mono, J., and Guneysu, T. (2021b). Folding bike: Scalable hardware implementation for reconfigurable devices. *IEEE Transactions on Computers*.

Roy, S. S. and Basso, A. (2020). High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 443–466.

Schanck, J. M. (2018). A comparison of ntru variants. *Cryptology ePrint Archive*.

Wang, W., Szefer, J., and Niederhagen, R. (2018). Fpga-based niederreiter cryptosystem using binary goppa codes. In *International Conference on Post-Quantum Cryptography*, pages 77–98. Springer.

Xing, Y. and Li, S. (2021). A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on fpga. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 328–356.

Zhang, N., Yang, B., Chen, C., Yin, S., Wei, S., and Liu, L. (2020). Highly efficient architecture of newhope-nist on fpga using low-complexity ntt/intt. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 49–72.

Zhu, Y., Zhu, M., Yang, B., Zhu, W., Deng, C., Chen, C., Wei, S., and Liu, L. (2021). Lwrpro: An energy-efficient configurable crypto-processor for module-lwr. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(3):1146–1159.