Quantum-resistant Cryptography in FPGA

Renata C. Policarpo *, Alexandre S. Nery *, Robson de O. Albuquerque *
Professional Post-Graduate Program in Electrical Engineering,
Department of Electrical Engineering*, University of Brasília, Brasília 70910-900
Email: renata.policarpo@aluno.unb.br; alexandre.nery@redes.unb.br; robson@redes.unb.br

2022 Workshop on Communication Networks and Power Systems (WCNPS) | 978-1-6654-7456-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNPS56355.2022.9969738

***Abstract*—The use of cryptography techniques to guarantee confidentiality, authenticity and integrity of sensitive data has become mandatory. Besides, advancements in quantum computers are gradually posing a threat to public key encryption technology. Without proper security measures, fraudsters may easily gain access to one's personal and sensitive data. The result of the third round of NIST's Post-Quantum Cryptography (PQC) process for standardization of public-key cryptography systems brought CRYSTALS-Kyber as the first mechanism selected for key encapsulation. The aim of this study is to provide the specification of a reconfigurable CRYSTAL-Kyber accelerator using High-Level Synthesis (HLS) technology. Our architecture requires about 2200 LUTs, 3001 FFs and 28 DSP on a low-cost Zynq FPGA (XC7Z020-1 CLG400C). The total time spent by the accelerator in a key exchange simulation is approximately 0.84 ms, operating at 100 MHz, and the estimated power consumption in this process is 1.695W.**

***Keywords*—Post-Quantum Cryptography; CRYSTALS-Kyber; accelerator; FPGA; HLS.**

I. Introduction

Cryptography techniques have become mandatory in the last couple of years, as the use of connected digital services have also increased substantially in the same time period. The use of cryptography techniques and protocols to guarantee confidentiality, authenticity and integrity of sensitive data is crucial [1]. The standardization of public-key cryptography schemes has enabled their implementation on various devices, such as computers, tablets and smartphones, allowing for secure communication applications [2].

However, quantum computers are gradually posing a threat to public key encryption technology, which is the main cryptography system used to date. For instance, RSA (Rivest–Shamir–Adleman) and Elliptic Curve Cryptography are not quantum-resistant [3], since the underlying mathematical operations (e.g. factoring of large integers and the discrete logarithm problem, that are the basis of these systems) are difficult to solve on conventional computers, but can be easily solved by a quantum computer [4].

978-1-6654-7456-6/22/$31.00 ©2022 IEEE

Considering the time and effort required for the development, standardization and transition to post-quantum cryptography (PQC) technology, NIST (National Institute of Standards and Technology) began in December of 2016 a public process to establish a new standard for public-key cryptography, which purpose is to assure secure new algorithms for digital signatures and key encapsulation in classical and quantum computers, as well as being compatible with existing communication protocols and networks. In July 2022, after three rounds of analysis and evaluation, the first four algorithms selected for standardization were released, being the CRYSTALS – Kyber, the only key encapsulation mechanism chosen [5].

This works presents a FPGA co-processor for Crystals-Kyber, designed and implemented using Xilinx Vitis High-Level Synthesis (HLS) tool, which enables fast and efficient development of hardware accelerators with a high degree of flexibility and performance. This HLS tool makes it possible to synthesize a specification in Register Transfer Level (RTL) architecture using VHDL or Verilog. For this paper, we chose Verilog as the synthesis language of CRYSTALS-Kyber C/C++.

Our paper is organized as follows: Section II describes the state-of-the-art related works. The CRYSTALS-Kyber Key Encapsulation Mechanism and the implemented accelerator is presented in Section III. Experimental results are presented in Section IV. Finally, Section V concludes and presents ideas for future work.

II. Related Works

Studies on post-quantum cryptographic algorithms have intensified in recent years and the algorithms that have passed through the NIST contest rounds have become even more in evidence by researchers.

The pure hardware implementations, such as in Huang [6], Xing [7], Bisheh-Niasar [8] [9] and Guo [10], proved to be more efficient since they presented a better performance with a shorter execution time compared to the implementations in software and in hardware/software codesign. Among these implemen-

tations, the architecture proposed by Bisheh-Niasar [9] has the best value (0.47) of area x time ratio compared with the others, while Huang [6] has the worst value (192.89). Xing [7] and Guo [10] offer reasonable execution time with less use of resource.

Although they do not present the best performance or architecture, studies carried out with hardware/software codesign approaches or using High-Level Synthesis (HLS) tools, like in Zhao [11] and Basu [12], show to be more flexible and easier to develop, while also presenting better execution time than software centered implementations.

As polynomial multiplication operations have a high computational and time cost for post-quantum algorithms based on lattices, such as Kyber, efforts to optimize implementations of the Number Theoretic Transform (NTT) are carried out in several studies [9] [13] [14].

Our accelerator brings an implementation in HLS of the polynomial multiplication function, based on NTT, with the purpose of optimizing the execution of the algorithms of generation, encapsulation and decapsulation of keys. Further details are presented in Section IV, including comparison with previous works.

III. CRYSTALS-Kyber Mechanism and Accelerator

This section briefly describes Kyber, the first quantum-resistant key encapsulation mechanism (KEM) selected for standardization by NIST [5], and then details the implementation of the proposed accelerator.

A. Kyber Mechanism

Kyber is part of the *Component of the Cryptographic Suite for Algebraic Lattices (CRYSTALS)* package. It is a key encapsulation mechanism (KEM) based on Module-Learning-With-Errors (MLWE) and had its original design presented in [15].

A key encapsulation mechanism is useful when there are the necessity to share symmetric keys securely between two parties. It can be seen as similar as Public Key Encryption scheme. The difference is that in KEM the symmetric key is derived by a Key Derivation Function, such as a cryptographic hash, using a random element, eliminating the need for padding of traditional public key systems.

Kyber is constructed by a CPA-Secure Public Key Encryption (PKE) scheme that, with the application of a variant [16] of the Fujisaki-Okamoto transform [17], results in a CCA-Secure KEM, having as main algorithms the key generation (Kyber.CCAKEM.KeyGen), encapsulation (Kyber.CCAKEM.Enc) and decapsulation (Kyber.CCAKEM.Dec) [18].

This mechanism is presented in three versions, Kyber512, Kyber768 and Kyber1024, which result in different post-quantum security levels according to the adopted parameter set. In the supporting documentation [18] it is possible to find the final version of the algorithms that make up Kyber KEM, with the design modifications and changes in the parameter values that were carried out during the NIST PQC Project rounds.

Figure 1 shows a simple example of a key exchange protocol using Kyber KEM functions. Alice uses the key generation algorithm to generate her public (pk) and private (sk) key pair, sending pk to Bob. Bob performs the key encapsulation algorithm that takes Alice's public key as input parameter and returns a ciphertext (c) and key (K). Alice receives the ciphertext from Bob and she executes the key decapsulation algorithm with the input parameters c and sk and obtains as an output the same key K that Bob has.

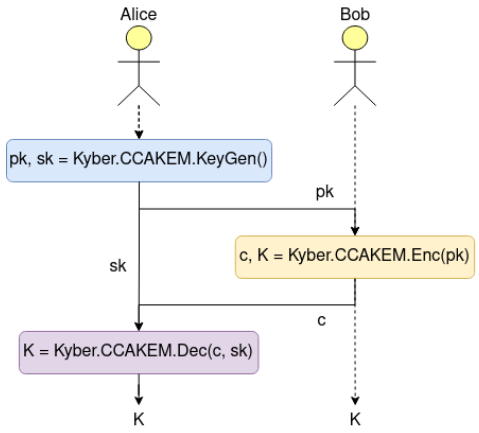

Figure 1: Key Exchange Protocol between Alice and Bob using Kyber.CCAKEM algorithms

B. Kyber Accelerator Architecture

Our Kyber accelerator architecture was designed using Xilinx Vitis High-Level Synthesis (HLS) tool (version 2022.1), targeting a low-power XC7Z020-1CLG400C Xilinx FPGA chip that embeds an ARM Cortex-A9 microprocessor. This RTL architecture results from the HLS synthesis and implementation of the proposed code executed by Vitis HLS, without optimizations. Verilog was the hardware description language selected to model the RTL architecture.

The co-processor architecture, showed in simplified form in Figure 2, uses AXI (Advanced eXtensible Interface) protocol, allowing the ARM processing system to communicate with the proposed co-processor. The interface is automatically produced by the HLS compiler, as shown later in Listing 1.

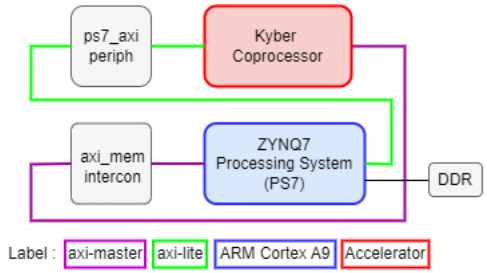

Figure 2: Kyber Architecture and Interface

The module "ps7_axi_periph" allows the connection and arbitration of the periphery by the processing system and "axi_mem_intercon" module enables the access to the external memory. Thus, the first is used for handling simple control signals and memory-mapped register data, while the latter is used for fast data transfer between the co-processor BRAM's and the external DDR memory.

C. Kyber Accelerator HLS Specification

The proposed accelerator implementation is based on the official reference code available in the public git repository [19] with the necessary modifications to enable the generation of the RTL architecture by HLS compiler.

The snippet presented in Listing 1 shows the main Kyber accelerator specification in HLS, while the whole specification is available in [20]. It takes as input parameters the addresses of two vectors of polynomials, *i.e.* a and b, and calculates the polynomial multiplication as specified in the reference code. For the proposed multiplication to be performed, it is necessary that all elements of the vector are in the NTT domain. The value returned by the accelerator into the address of polynomial r results from applying the Barrett reduction to all coefficients of the polynomial resulting from the multiplication.

Listing 1: CRYSTALS-Kyber Polynomial Multiplication Vitis HLS Implementation

```
1
2 #define KYBER_N 256
3 #define KYBER_K 3
4
5 void kyber_accelerator(volatile poly *r, volatile const
      polyvec *a, volatile const polyvec *b)
6 #pragma HLS INTERFACE mode=m_axi depth=1 port=r
      offset=slave
7 #pragma HLS INTERFACE mode=m_axi depth=1 port=a
      offset=slave
8 #pragma HLS INTERFACE mode=m_axi depth=1 port=b
      offset=slave
9 #pragma HLS INTERFACE mode=s_axilite port=r
      bundle=CONTROL
10 #pragma HLS INTERFACE mode=s_axilite port=a
      bundle=CONTROL
11 #pragma HLS INTERFACE mode=s_axilite port=b
      bundle=CONTROL
12 #pragma HLS INTERFACE mode=s_axilite port=return
      bundle=CONTROL
13
14 unsigned int i,j;
15 poly t;
16 poly r_hls;
17 polyvec a_hls;
18 polyvec b_hls;
19
20 //copy data into the accelerator
21 for (i = 0 ; i < KYBER_K_hls ; i++) {
22    for (j = 0 ; j < KYBER_N_hls ; j++) {
23  #pragma HLS PIPELINE
24  a_hls.vec[i].coeffs[j] = a->vec[i].coeffs[j];
25  b_hls.vec[i].coeffs[j] = b->vec[i].coeffs[j];
26    }
27 }
28
29 /********** polynomial multiplication **********/
30
31 poly_basemul_montgomery_hls(&r_hls, &a_hls.vec[0],
      &b_hls.vec[0]);
32
33 for(i=1;i<KYBER_K_hls;i++) {
34    poly_basemul_montgomery_hls(&t, &a_hls.vec[i],
         &b_hls.vec[i]);
35    poly_add_hls(&r_hls, &r_hls, &t);
36 }
37
38 //Barrett reduction
39 poly_reduce_hls(&r_hls);
40
41 /***********************************************/
42
43 //copy computed data back to main processor
44 for (j = 0 ; j < KYBER_N_hls ; j++) {
45    #pragma HLS PIPELINE
46    r->coeffs[j] = r_hls.coeffs[j];
47 }
48 }
```

Lines 6-12 specifies the communication protocol definitions established between the HLS design and external components. The m-axi interface allows fast bursts of data into and out of the FPGA's Block RAM memory (BRAM), as shown in Lines 21-27 and Lines 44-47. This interface follows the AXI standard and allows data transfer in bursts of up to 256 cycles with just a single address phase. This same interface returns the results of the polynomial multiplication calculated by the function from BRAM to the AXI master interface. The "poly_basemul_montgomery_hls" is the function that calculates the multiplication of two polynomials in NTT domain, while the "poly_reduce_hls" function, in line 39, applies Barrett reduction to all coefficients of a polynomial. These functions follow the code available in [19] with the suffix hls to indicate the functions that have been implemented in FPGA.

IV. Experimental Results

This section presents performance, circuit-area and power consumption results of the Kyber accelerator, when compared with the part of the system that is executed by the ARM Cortex-A9 processor. The performance results were obtained when using the accelerator for the processing of the Kyber768 version, that has estimated 128-bit post-quantum security. Circuit-area and performance results are provided by the synthesis

process of Xilinx Vitis HLS tool (v2022.1) targeting the XC7Z020-1CLG400C Xilinx FPGA (Pynq-Z1).

A. Performance

Table I shows the average values related to latency, interval and the total execution time, in clock cycles, of the accelerator. As the proposed design was synthesized at 100 MHz, the total time spent processing the accelerator was approximately 0.84 milliseconds. This value corresponds to the time spent during a key exchange simulation, being the sum of all the times the accelerator was executed during one execution of Kyber.CCAKEM.KeyGen(), Kyber.CCAKEM.Enc() and Kyber.CCAKEM.Dec().

Table I: Results of accelerator performance

RTL	Latency (clock cycles)	Interval (clock cycles)	Total execution time (clock cycles)
Verilog	6983	6983	83804

B. Circuit-area

The summary of resources used during the synthesis process performed by Vitis HLS is shown in Table II. Given the information provided by the reports it is possible to observe that there is still room to implement more operations into the FPGA logic.

Table II: FPGA Resources utilization

Resource	Available	Utilization	%
LUT	53200	2200	4.14
LUTRAM	17400	215	1.24
FF	106400	3001	2.82
BRAM	140	3.50	2.50
DSP	220	28	12.73

As expected, the DSP slices is the resource with the highest percentage of use (12.73%), which shows that the HLS compiler was able to take advantage of DSP slices to accelerate the polynomial multiplications. Also, as expected, the LUTRAM is the least used (1.24%) resource, as most of the design implemented in FPGA is focused in accelerating polynomial multiplications and handles small portions of data.

C. Power consumption

Using Vivado HLS vectorless power analysis it was possible to estimate the power consumption of the implemented design and the power analysis of the project is given in Table III. This analysis shows that the estimated total power consumption is about 1.695 W during the key exchange simulation using Kyber768 algorithms.

The power consumption related to the polynomial multiplication functions implemented in the programmable part of the FPGA can be interpreted as the sum of the power consumed by the dynamic components clocks, signals, logic, BRAM and DPS. Thus, the accelerator consumed approximately 0.03 W of power. The highest power consumption (1.527 W) was by the ARM Cortex-A9 processor, defined as PS7, for the execution of the other functions of the algorithms that were not synthesized in the accelerator.

Table III: Power consumption of implemented design

	Component	Power (W)
Dynamic	Clocks	0.005
	Signals	0.006
	Logic	0.003
	BRAM	0.003
	DSP	0.014
	PS7	1.527
Static	-	0.137
Total	-	1.695

D. Comparison with related works

Table IV shows a comparison of our results with related works. In addition to our work, only Basu [12] used HLS as a tool for RTL design development. Their work is implemented using the Kyber512 version parameters and uses almost 900x more LUTs and 65x more FFs than our work, but they had implemented all operations related to encapsulation and decapsulation of keys.

Table IV: Comparison with previous works

Related Works	Function	HLS	LUT	FF	DSP	BRAM	Freq (MHz)
[6]	Enc / Dec	No	110260	-	292	202	155
[7]	KeyGen / Enc / Dec	No	7412	4644	2	3	161
[8]	KeyGen / Enc / Dec	No	16000	6000	9	16	115
[9]	NTT	No	801	717	4	2	222
[12][2]	Enc / Dec	Yes	1977896	194126	-	-	-
[13]	NTT / PM[1]	No	9508	-	16	35	172
[14]	NTT / PM[1]	No	5181	4833	16	-	227
Our	PM[1]	Yes	2200	3001	28	3.5	100

[1] Polynomial Multiplication
[2] Kyber512 version

Xing [7] and Bisheh-Niasar [8] present a pure hardware implementation for all main CRYSTALS-Kyber algorithms. Between the two works, Xing [7] consumes less resources of area circuit, with the consumption of LUTs and FFs being approximately 3.4x and 1.5x higher than our results. Their implementation uses less DSP and BRAM resources than our work, that are a hardware/software co-design proposal, with part of the algorithms running on FPGA and part on ARM processor.

In [9], Bisheh-Niasar proposes a Number Theoretic Transform optimization that uses low resources of a Artix-7 platform, while Yaman [13] and Ma [14] have the NTT and polynomial multiplication functions in

their implementations. Our co-processor initially brings the implementation of the polynomial multiplication function, which makes it difficult to directly compare the results with the other works, as the results are not from the separate operations. In future work, more functions, like NTT, should be implemented in our co-processor and optimized using HLS tools.

V. Conclusions and Future Work

The evolution in the development of quantum computers is a threat to the cryptography used in several current protocols, making it necessary to evolve to quantum cryptography to ensure the continuity of data security.

In this paper, an accelerator for the polynomial multiplication for the CRYSTALS-Kyber Post-Quantum Cryptography algorithm is presented. The accelerator is synthesized for XC7Z020-1CLG400C Xilinx (Pynq-Z1) FPGA using High Level Synthesis tool to translate C specification to Register Transfer Level architecture in Verilog.

The results showed that the use of resources by the accelerator was low, with the DSP slice being the resource with the highest percentage of use (12.73%). This makes it possible to expand the use of the FPGA to implement other functions that can speed up the execution of Kyber, the first algorithm selected for standardization by NIST. The time taken for the accelerator to run during a key exchange simulation was 0.84 milliseconds and the estimated power consumption was only 0.03 W.

Our future work will focus on optimizing the polynomial multiplication function and also implementing in hardware the functions needed to perform the NTT operations.

ACKNOWLEDGMENT

This research is funded by the CNPq (Conselho Nacional de Pesquisa e Desenvolvimento) 426729/2018-8 project and the FAPDF (Fundação de Apoio a Pesquisa do Distrito Federal) 00193-00002139/2018-79 project. The authors acknowledge the support of ABIN (Agência Brasileira de Inteligência) grant 08/2019. R.d.O.A. gratefully acknowledges the General Attorney of the Union - AGU grant 697.935/2019; the General Attorney's Office for the National Treasure - PGFN grant 23106.148934/2019-67. Last but not least, the authors would like to thank Xilinx University Program for the donation of the licenses that allowed the development of this work.

References

[1] M. T. Gençoğlu, "Importance of cryptography in information security," *IOSR J. Comput. Eng*, vol. 21, no. 1, pp. 65–68, 2019.

[2] L. Chen, "Cryptography standards in quantum time: new wine in old wineskin?" *IEEE security & privacy*, vol. 15, no. 4, p. 51, 2017.

[3] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep 2017. [Online]. Available: https://doi.org/10.1038/nature23461

[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[5] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the third round of the nist post-quantum cryptography standardization process," National Institute of Standards and Technology Gaithersburg, MD, Tech. Rep., 2022.

[6] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of crystals-kyber pqc algorithm through resource reuse," *IEICE Electronics Express*, pp. 17–20 200 234, 2020.

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

[8] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Instruction-set accelerated implementation of crystals-kyber," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, pp. 4648–4659, 2021.

[9] ——, "High-speed ntt-based polynomial multiplication accelerator for post-quantum cryptography," in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2021, pp. 94–101.

[10] W. Guo, S. Li, and L. Kong, "An efficient implementation of kyber," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1562–1566, 2022.

[11] Y. Zhao, Z. Chao, J. Ye, W. Wang, Y. Cao, S. Chen, X. Li, and H. Li, "Optimization space exploration of hardware design for crystals-kyber," in *2020 IEEE 29th Asian Test Symposium (ATS)*, 2020, pp. 1–6.

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

[13] F. Yaman, A. C. Mert, E. Öztürk, and E. Savaş, "A hardware accelerator for polynomial multiplication operation of crystals-kyber pqc scheme," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1020–1025.

[14] L. Ma, X. Wu, and G. Bai, "Parallel polynomial multiplication optimized scheme for crystals-kyber post-quantum cryptosystem based on fpga," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. IEEE, 2021, pp. 361–365.

[15] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367.

[16] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the fujisaki-okamoto transformation," in *Theory of Cryptography Conference*. Springer, 2017, pp. 341–371.

[17] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual international cryptology conference*. Springer, 1999, pp. 537–554.

[18] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber algorithm specifications and supporting documentation (version 3.02)," Aug 2021, https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[19] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Pq-crystals/kyber," https://github.com/pq-crystals/kyber, 2018.

[20] R. Policarpo, "Kyber coprocessor hls," https://github.com/renatacpolicarpo/KyberCoprocessorHLS/, 2022.