# POSTQUANTUM CRYPTOGRAPHY ON FPGA

**A. Gyõri[1], D. Smékal[1]**

[1]Brno University of Technology, Czech republic

[2]Brno University of Technology, Czech republic

E-mail: 187286@vut.cz, smekald@vut.cz

**Abstract**—This work describes the post-quantum algorithm FrodoKEM, its hardware implementation in VHDL and software simulation of implementation. The issue of postquantum cryptography and the VHDL programming language used to describe the functionality of the hardware was studied. The acquired knowledge was transformed into a functional simulation of all parts of the algorithm. All these parts have already been implemented separately, so that the functionality of every single part can be separately approached. These parts are key generation, encapsulation and decapsulation. After successful simulation. These parts will be synthetised and implemented to FPGA board NEXYS A7.

**Keywords**—Post-quantum cryptography, lattices, Frodo, FPGA, Testbench, LWE, hardware, VHDL

## 1. INTRODUCTION

Once quantum computers become available, it will be possible to solve previously difficult mathematical problems of factorization and discrete logarithms. These two issues underlie many currently used asymmetric cryptographic schemes. As a result,National Institute of Standards and Technology launched a 2017 call for tenders, which are required to submit an asymmetric encryption and signature scheme that can withstand classic attacks, etc. quantum computer. The submitted candidates use very differently approaches to achieving resistance to quantum computers, such as lattices, codes, hash functions, or multidimensional polynomials. Lattice-based cryptographic [5] schemes can be divided into three classes depending on the problem: encryption schemes such as textsf FrodoKEM [1] or Round5 [2] are based on the standard problem of learning with errors, the result is excellent security due to the absence of any structure and scalability for the price reduced efficiency or speed due to large parameters, keys and ciphertext. In contrast, Ring learning with errors -based schemes such as NewHope [3] or NTRU [4] offer better performance, as well as smaller keys and ciphertext than those based on standard lattices. The lattice-based cryptosystem class is based on the Module-LWE problem and offers a trade-off between the security of unstructured lattices and the performance of fully structured lattices. While a high-level C software implementation exists for all the proposals presented, there is virtually no hardware implementation, at least according to my personal research. Software implementations are performed on a processor and can be easily transferred between different types of processors. In contrast, hardware implementations can be performed using Field programmable gate array or Application-Specific Integrated Circuits. They are usually written in the hardware description language HDL, such as VHDL or Verilog. The task of this work is to design and implement the FrodoKEM algorithm directly in HDL.

## 2. ALGORITHM DESCRIPTION

FrodoKEM is a lattice-based key encapsulation scheme. Because FrodoKEM is based on LWE and not RLWE, it has larger parameters and is slower, but provides greater security than other lattice-based algorithms. The Frodo.Gen algorithm generates a matrix A line by line using either the SHAKE hash function or AES. I chose SHAKE in my implementation. Generation is deterministic based on the so-called short sequence and the index of the line just generated. Using Frodo.SampleMatrix, matrices are sampled by elements from a discrete Gaussian distribution. Frodo.Pack transforms the matrix into a bit string, while Fordo.Unpack does the exact opposite operation. Frodo.Encode encodes the bit string into an integer matrix, and Frodo.decode transforms the matrix into a bit string as the name implies.

## 3. ARCHITECTURE OF ALGORITHM IMPLEMENTATION

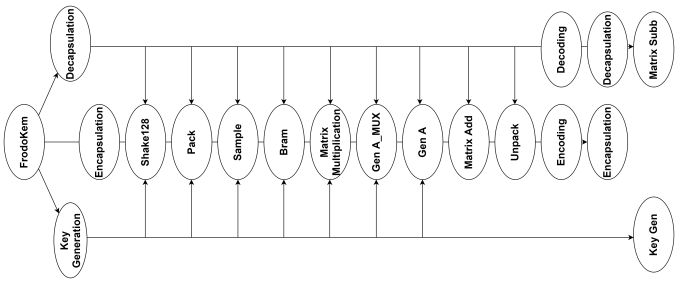


**Figure 1:** Architecture of FrodoKEM VHDL implementation

A scheme describes the way, algorithm has been divided to various components. As there is to be seen, some components are shared by more than one part of algorithm. The way a component works is as followed. Component gets an input, provides computation and sends output to following component. This way a component can be shared across more parts of the algorithm as the functionality is still the same only input is what makes the difference. For the implementation, the strongest and most robust Frodo-1344 parameter set was used.

| | Frodo-640 | Frodo-976 | Frodo-1344 |
|---|---|---|---|
| $D$ | 15 | 16 | 16 |
| $q$ | 32768 | 65536 | 65536 |
| $n$ | 640 | 976 | 1344 |
| $B$ | 2 | 3 | 4 |
| $\text{len}_{\mu} = l$ | 128 | 192 | 256 |
| $\text{len}_{\text{seed}_{\mathbf{SE}}}$ | 128 | 192 | 256 |
| $\text{len}_{\mathbf{s}}$ | 128 | 192 | 256 |
| $\text{len}_{\mathbf{k}}$ | 128 | 192 | 256 |
| $\text{len}_{\mathbf{pkh}}$ | 128 | 192 | 256 |
| $\text{len}_{\mathbf{ss}}$ | 128 | 192 | 256 |
| $\text{len}_{\chi}$ | 16 | 16 | 16 |
| $\chi$ | $\chi$Frodo-640 | $\chi$Frodo-976 | $\chi$Frodo-1344 |
| SHAKE | SHAKE128 | SHAKE256 | SHAKE256 |

**Table I:** FrodoKEM parameter sets

## 4. SIMULATION AND HARDWARE IMPLEMENTATION

All three parts were simulated and subsequently implemented to hardware separately as to show functionality of each one. The way testbench program was designed is as follows. The input of component is a set of hexadecimal values corresponding to certain part of algorithm. For example for key generation the input is initial randomness. These values are read from text file, after that computation processes. Output values are after that compared to values in text form, that are being expected as with the same initial randomness, same key must be generated. Same principles are being used with other parts of algorithm. On the following picture simulation result of key generation is to be seen.The input of this part are hexadecimal values of initial randomness. It is 23 values, that are being read in state read rand. That state is not visible on the screen, because it lasts only 23 clock cycles and is significantly smaller than other states. In state read pk and sk, the private and secret key, that are expected to be result are read.

**Figure 2:** Simulation of algorithm FrodoKEM key generation

Following image shows successfully finished simulation. If all comparisons of computed and initially loaded values were successful, test is evaluated as successful.

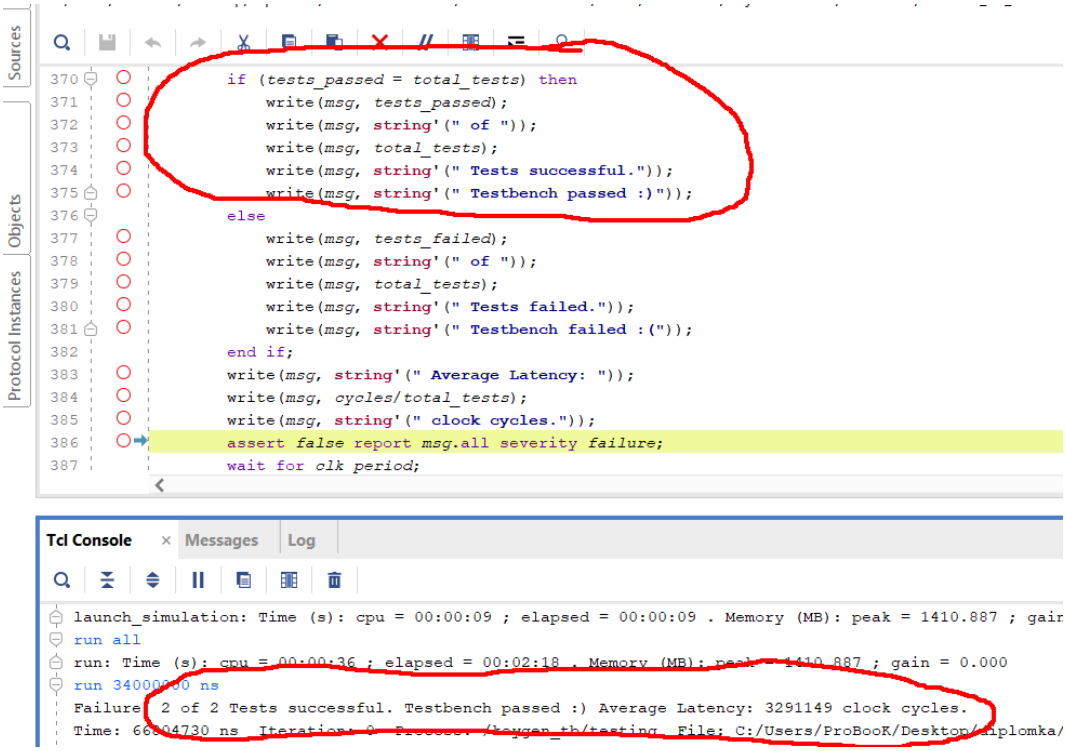


**Figure 3:** Result message of simulation

Hardware implementation is currently the only part of work that is yet to be done. After simulation in development enviroment XILINX Vivado, the three project parts will be synthetised and a bit stream will be generated. The result will be implemented on FPGA board NEXYS A7. The implementation will be run on board. Data will be provided to board vie microSD card and the result of computation will be stored on same micro SD card.

## 5. CONCLUSION

The presented paper deals with the problem of implementation of currently proposed quantum security algorithms. Algorithm that was used was FrodoKEM. It was successfully implemented VHDL ans simulated. Afterwards the solution will be planted on FPGA board and the functionality will be tested.

## REFERENCES


[1] Alkim, Erdem and Bos, Joppe and Ducas, Léo and Longa, Patrick and Mironov, Ilya and Naehrig, Michael and Nikolaenko, Valeria and Peikert, Chris and Raghunathan, Ananth and Stebila, Douglas. FrodoKEM - Learning With Errors Key Encapsulation [cit. 14. 11. 2021]. Dostupné z: URL https://www.designingbuildings.co.uk/wiki/Energy_infrastructure?fbclid=IwAR1CMidlcOQs7Rfu-n2HXK9emn5Q30xyrqj_eyiIsFNS7jHRdxnbW3e0nM0

[2] Baan, Hayo and Bhattacharya, Sauvik and Fluhrer, Scott and Garcia-Morchon, Oscar and Laarhoven, Thijs and Rietman, Ronald and Saarinen, Markku-Juhani O. and Tolhuizen, Ludo and Zhang, Zhenfei (2019) *Post-Quantum Cryptography*, Springer International Publishing, 978-3-030-25510-7.

[3] Alkim, Erdem and Avanzi, Roberto and Bos, Joppe and Ducas, Léo and de la Piedra, Antonio and Pöppelmann, Thomas and Schwabe, Peter and Stebila, Douglas (2019) *New Hope – Algorithm Specifications and Supporting Documentation*

[4] Chen, Cong and Danba, Oussama and Hoffstein, Jeffrey and Hülsing, Andreas and Rijneveld, Joost and Schanck, John M. and Schwabe, Peter and Whyte, William and Zhang, Zhenfei (2019) *NTRU – Algorithm Specifications and Supporting Documentation*

[5] *Chris Peikert. A Decade of Lattice Cryptography [cit. 17. 2. 2016].* Dostupné z: URL https://eprint.iacr.org/2015/939.pdf