BilKa: Design and Implement an SoC Architecture with a CRYSTALS-Kyber Accelerator

Bilge Deniz Özçelik
(bilgedenizozcelik@eskisehir.edu.tr)
Eskişehir Technical University

Selahattin Kaan Tosun
(selahattinkaantosun@eskisehir.edu.tr)
Eskişehir Technical University

Supervisor: Assist.Prof.Dr. İsmail San
(isan@eskisehir.edu.tr)
Eskişehir Technical University

Youtube Link: https://www.youtube.com/watch?v=ujgfLmGZtUk
Github Link: https://github.com/DHLSan/BilKa

Abstract— Encryption is one of the most important and first methods that come to mind in ensuring data security, which has become increasingly important with digitalization. With the discovery of quantum computers, the idea that cryptographic algorithms used today and thought to take billions of years for supercomputers to break can be easily broken, has prompted scientists. We aim to design the SoC design and hardware accelerators of the CRYSTALS-Kyber algorithm, which is one of the three finalists of Round 3 in the post-quantum cryptography competition initiated by NIST (National Institute of Standards and Technology), which is faster than its other two competitors in applications requiring higher levels of security. In the project, which was made using Zynq-7000 ZedBoard SoC, Ubuntu operating system was installed on ZedBoard because of the OpenSSL libraries in the Kyber algorithm and the functions that will work in the Linux operating system. Thus, using embedded Linux, the encryption algorithm is enabled to run on the hardware.

Index Terms- Hardware acceleration, NIST cryptography competititon, PetaLinux, post-quantum cryptography

1 Introduction

In the world, which has entered a new era with the effect of the increasing digitalization process in recent years, using of more complex System-on-Chips (SoC) has become widespread in order to keep up with the speed of technological developments. Also, the importance of data has increased with era of digitalization, the volume of data has grown and the data increase rate has been affected by these technological developments. Thus, it has become a requirement to securely store increasing importance and number of the data. The most widely used method of securely storing the data is encryption. Since the algorithms used for encryption operation at the present time involve complex and intensive arithmetic calculations, they consume a significant amount of time on the standard processors. Therefore, the slow encryption operation needs to be accelerated by developing a SoC architecture in order to keep up with the speed of the changing world.

For many centuries, the desire of human beings to protect confidentiality during the transmission and storage of the data has led to the emergence of the cryptography. The main developments in the field of the cryptography date back to the 20th century, when supercomputers were invented. In that period, quantum cryptography based on public-key encryption was developed, involving difficult mathematical problems that would consume a lot of resources and time to compute the solution and break the system in classical computers. However, the security of public-key system data encryption used nowadays will be greatly compromised if large-scale quantum computers are built. Accordingly, National Institute of Standards and Technology (NIST) initiated a standardization process to develop post-quantum cryptographic systems that are secure against both quantum and classical computers and can interoperate with existing communication protocols and networks in 2017.

Designing a whole complex-system on a single chip is very efficient in terms of power and performance. Thanks to ever increasing size of transistors on a single-die silicon chip, a very complex system can be synthesized into a single silicon device. As the size of the system is increasing, the data bandwidth of such systems reaches very high levels. Data security is of utmost importance and designing high-throughput encryption/decryption hardware IP cores to provide data security is a challenging task especially at RTL-level. Conventional cryptographic systems provide security at very high frequency, but some of asymmetric cryptographic algorithms might be broken with quantum computing. Quantum-processor based systems may become available in near future and

solve some conventially secure systems. Thus, designing a secure system from attacks using quantum computers is a challenging research topic. Hardware acceleration of the computation of quantum-secure algorithms on a conventional computing device is needed to reach higher-bandwidth and quantum-level security and the design space is huge.

In this project, we aim to accelerate the CRYSTALS-Kyber algorithm, which is currently one of the finalists in the third round of the Post-Quantum Cryptography competition initiated by NIST, with an HW/SW codesign methodology and application-specific hardware accelerators by using High-Level Synthesis (HLS). In the study of Dang et al. [1], it was concluded that the hardware architecture of CRYSTAL-Kyber algorithm runs better than other finalist algorithms at higher security levels (e.g., Level 3 and Level 5) in Artix-7 FPGAs and Zynq Ultrascale+ devices. Since the information in security level 3 and 5 categories contain risky social, psychological and financial information that can harm individuals or groups if disclosed, it is a basic requirement that these data be stored securely. We aim to hardware accelerate of the CRYSTAL-Kyber algorithm using HLS to ensure the security of applications with high security level requirements with low latency and explore the design space faster thanks to higher design productivity of HLS.

The paper is organized as follows: Section 2 described the technical background related to post-quantum cryptography and PetaLinux Tools of Xilinx. Section 3 provides implementation of system and explanation of the system architecture we developeda and also experimental results. Section 4 provides conclusion. In the final section 5, the paper is closed with a future work.f

2 THEORETICAL BACKGROUND

In this section, information that forms the basis of the studies carried out within the scope of the project and that will facilitate the understanding of the processes is given.

2.1 Post-Quantum Cryptography

Cryptography is the process of encrypting data or converting plain text to scrambled text so that only someone with the right key can read it. Today, quantum encryption based on quantum mechanics is used in the field of cryptography. It takes a long time for classical computers to decode this encryption method. However, with the development of technology in recent years, important researches have been made about quantum computers. While there is still no clear information about when a large-scale quantum computer can be built, it is thought to start working on algorithms that are resistant to quantum computers in the field of cryptography, with the increasing probability. In particular, with most experts predicting that a quantum computer could be built in 9 to 10 years, NIST has sought to establish a standard for post-quantum encryption, as current cryptographic algorithms are thought to be easy to decipher by a quantum computer. If this happens, sensitive data will no longer be protected. Thus, a competition was held to find a secure algorithm against attacks by quantum computers.

2.2 CRYSTALS-Kyber Algorithm

The Kyber algorithm, one of the three finalists of the post-quantum cryptography project in this competition organized by NIST, is a lattice-based key encapsulation mechanism. Lattice-based cryptography, which is thought to be unable to break even quantum computers, constitutes the lion's share of post-quantum cryptography, especially as in the algorithms participating in this competition. With this structure, there are still difficult problems that cannot be solved if a quantum computer is built.

In the final of the Round 3 of the NIST competition, algorithms are aimed to provide different sets for three different security levels. For this purpose, the Kyber algorithm performs encryption at three different security levels, 512, 768 and 1024, and these correspond to the 128, 192 and 256 bit key length versions of the AES (Advanced Encryption Standard) cryptographic algorithm, respectively.

In the category of indistinguishability in terms of security, the Kyber algorithm is the IND-CCA2 (INDistinguishability under adaptive Chosen Ciphertext Attack) key encapsulation mechanism, which has the strongest definition. Kyber has the basic parts of encryption operation such as key generation, encapsulation, and decapsulation. You can find the algorithms for these parts in Algorithms 1, 2 and 3.

In the CCA transform is hash the public key pk into the pre-key K and into the random coins r in Algorithm 2 and hash of the ciphertext into the final key K. [2]

Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$
1: $z \leftarrow \mathcal{B}^{32}$
2: $(pk, sk') := \text{KYBER.CPAPKE.KeyGen}()$
3: $sk := (sk' \| pk \| \text{H}(pk) \| z)$
4: **return** (pk, sk)

Algorithm 1. Kyber.CCAKEM.KeyGen()

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Output: Shared key $K \in \mathcal{B}^*$
1: $m \leftarrow \mathcal{B}^{32}$
2: $m \leftarrow \text{H}(m)$
3: $(\bar{K}, r) := \text{G}(m \| \text{H}(pk))$
4: $c := \text{KYBER.CPAPKE.Enc}(pk, m, r)$
5: $K := \text{KDF}(\bar{K} \| \text{H}(c))$
6: **return** (c, K)

Algorithm 2. Kyber.CCAKEM.Enc(pk)

Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Input: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8 + 96}$
Output: Shared key $K \in \mathcal{B}^*$

$pk := sk + 12 \cdot k \cdot n/8$
$h := sk + 24 \cdot k \cdot n/8 + 32 \in \mathcal{B}^{32}$
$z := sk + 24 \cdot k \cdot n/8 + 64$
$m' := \text{KYBER.CPAPKE.Dec}(\mathbf{s}, (\mathbf{u}, v))$
$(\bar{K}', r') := \text{G}(m' \| h)$
$c' := \text{KYBER.CPAPKE.Enc}(pk, m', r')$
if $c = c'$ **then**
    return $K := \text{KDF}(\bar{K}' \| \text{H}(c))$
else
    return $K := \text{KDF}(z \| \text{H}(c))$
end if
return K

Algorithm 3. Kyber.CCAKEM.Dec(c,sk)

2.3 How To We Use PetaLinux

PetaLinux is a set of software tools that allow for customizing, building and deploying the embedded Linux development on Xilinx processing systems. Also, PetaLinux actually refers to an individual software package, but it is not a standalone embedded Linux development solution. The workflow for PetaLinux consists of many phases where it builds on other Xilinx software such as Vivado and Vitis (before Xilinx SDK, Software Development Kit).

Browsing the Xilinx documentation for PetaLinux reveals a few key things. For example, Xilinx provides tools for multiple operating systems, while PetaLinux tools are designed for Linux users only. The Xilinx PetaLinux tool imports the hardware and generates Linux images for the target platform. [3]

When working with PetaLinux, following the hardware design in Xilinx Vivado, there are steps to install the operating system on ZedBoard or any other SoC. These steps will be explained in detail in Section 3. Implementation. Codes written in bare-metal technique work directly on the processor. However, in this case, when the PetaLinux project is created, the codes now run on the installed operating system, not directly on the processor.

3 IMPLEMENTATION AND EXPERIMENTAL RESULTS

Since the CRYSTAL-Kyber algorithm includes OPENSSL libraries and some functions used in its structure are suitable for the linux operating system, they cannot be run in Xilinx Vivado and Xilinx Vitis (or SDK) applications. Because the hardware platform we use is ZedBoard, a member of the Xilinx SoC Zynq-7000 family, which consists of the hard wired ARM 9 dual core processor (PS, Processing System) and a programmable area (PL, Programmable Logic). The codes written in ZedBoard run directly on the processor unless the hardware design is made, that is, the codes are run using the bare-metal technique. However, the OPENSSL libraries required for the Kyber, post-quantum encryption algorithm, whose accelerator is to be designed, and the operating system for linux-based functions are required. Therefore, it is planned to carry out the project with PetaLinux operating system.[4]

Ubuntu operating system (18.04 LTS, Long Term Support), a Linux distribution, was used during the construction phase of the project. As Xilinx Tools (Vivado, Vitis , PetaLinux, Vitis HLS) version 2019.2 has been selected. After all the necessary environment has been set up, the hardware design has been prepared to run the Kyber algorithm completely in the software. At this stage, there is only Zynq processing system in block design. Ethernet input from the I/O peripheral pins is activated so that the code can be downloaded from the internet to ZedBoard, and also it can be update on libraries in processing system.

The operating system required for the Kyber algorithm to work will run on the SD card on ZedBoard, so PetaLinux will be booted from the SD card. For this, you need to make SD Card partititon. At the end of the hardware design, the area where the bitstream and necessary files will be found is allocated as the fat32 file system named label BOOT, while the remaining area is created in the rootfs label in the ext4 file system where the files of the operating system will be found. So the SD card is divided into two parts.

The kernel image must be generated from the hardware designed for the BOOT space allocated on the SD card. U-boot initializes the platform hardware needed to load the linux kernel. It also provides support for a wide variety of CPU archi-

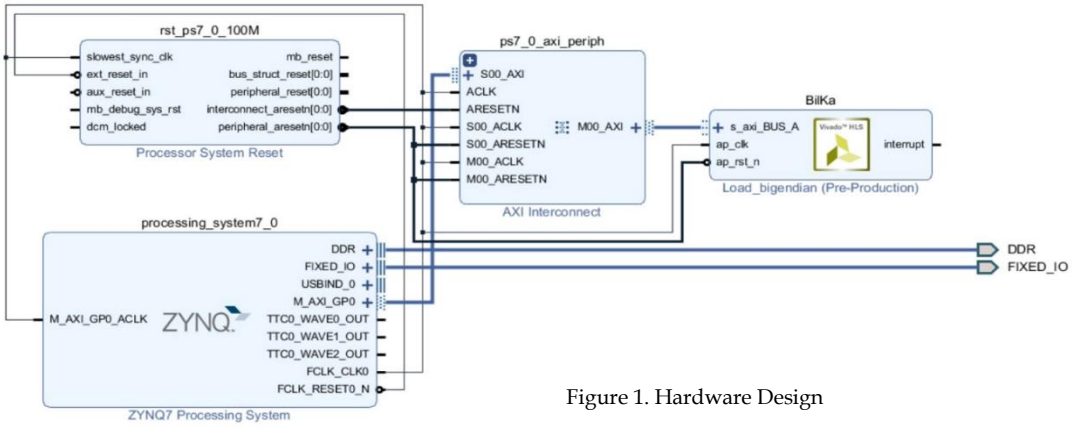

Figure 1. Hardware Design

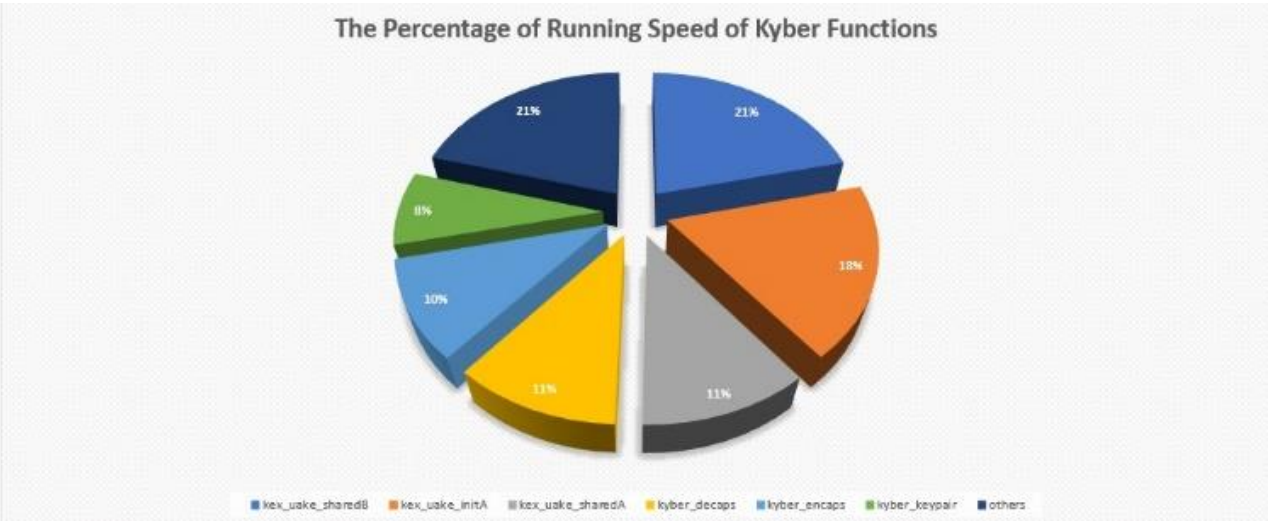

Table 1. The Percentage of Running Speed of Kyber Functions

tectures, including U-Boot, hundreds of embedded boards, and ARMs, the richest, most flexible, and most actively developed open source available for Embedded Linux OS (Operating System). After this kernel is ready, the PetaLinux project is configured. Since the boot process is SD card in our project, the hardware settings should be selected as SD card. After project is successfully built, BOOT.BIN file is generating.[5]

In order to run an Ubuntu operating system on ARM, the file system of the Ubuntu operating system, which is reserved as rootfs on the SD card, has been downloaded. Thus, when the SD card was inserted into the ZedBoard, thanks to the bitstream contained in it, bitsream was installed and the Ubuntu operating system could be run in it. PetaLinux operates on the Processing System (PS) part of the SoC and runs a C application. Thus, the GCC (GNU Compiler Collection), which is required for the C code to work, is loaded from the ethernet and the code is run in the software. [6]

Since the parts of the Kyber code that need to be run to detect the slow-running functions are written for x86 processor architecture, it gives an error in the assembly code for ARM. For this, the profiling process was carried out as a result of writing those parts without using any functions and headers in accordance with the C code. Profiling result appears in Table 1 and Figure 2. By removing the hardware accelerators of the functions that run the slowest here, the system will be accelerated significantly. Since the key exchange functions, which are seen to be the slowest, only call other functions and do not have any arithmetic operations in them, instead of extracting the hardware design of these functions using HLS, the parts of the arithmetic operations called in these functions are put in HLS. You can see the hardware design in Figure 1.

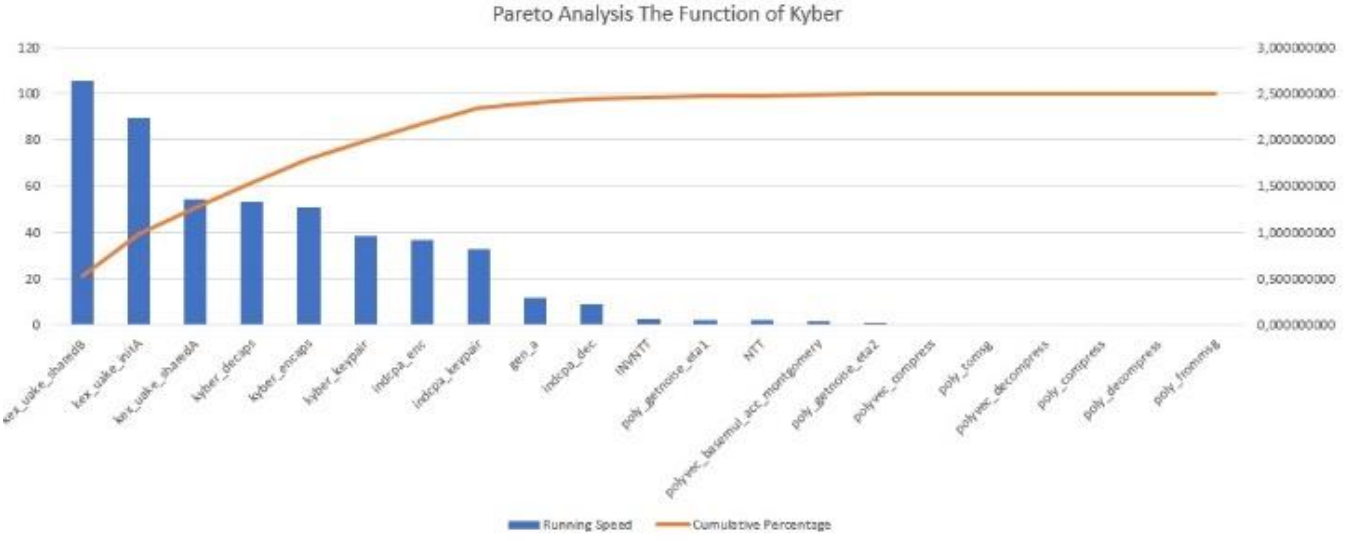

Figure 2. Pareto Anaylsis The Function of Kyber

4 CONCLUSION

In this work, using PetaLinux, the CRYSTAL-Kyber encryption algorithm is run on ZedBoard, which is not suitable for bare-metal technique and ARM processor architecture. Encryption was done successfully using OpenSSL libraries. The speeds of the functions were measured and it was aimed to optimize the system by accelerating the slowest employees, so that the most common functions among the slowest functions were designed using Vitis HLS. The created two Intellectual Properties (IP) has been added to the hardware and the BOOT part that has been created and put on the SD card has been changed to work only in software. Picocom, a serial port based Linux console, was used to access my operating system on ZedBoard. Thus, a hardware project was carried out by running high level C code from the terminal using two Linux operating systems (on the computer and ZedBoard).

5 FUTURE WORK

As one of the finalists of the NIST Post-Quantum Cryptography Competition, the Kyber algorithm is expected to accelerate in the future stages of this project, in which the SoC design and accelerators are made, by running the functions that have the hardware design removed, and the speed will increase as a result of optimizing the code using pipeline or DMA (Direct Memory Access). Thus, the Kyber algorithm, which is faster at higher security levels compared to its other two rivals, Saber and NTRU, is intended to work much faster and with lower latency in these applications, which are of great importance.

REFERENCES

[1] Dang, V. B., Mohajerani, K., & Gaj, K. (2021). High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber. Cryptology ePrint Archive.W.-K. Chen, *Linear Networks and Systems*. Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)

[2] https://pq-crystals.org/kyber/data/kyber-specification-round3.pdf

[3] K. Vipin and S. A. Fahmy, “Automated partial reconfiguration design for adaptive systems with CoPR for Zynq,” in IEEE Int. S ym. on Field Programmable Custom Computing Machines, 2014, pp. 202 205.

[4] Al Kadi, Muhammed, et al. "Dynamic and partial reconfiguration of Zynq 7000 under Linux." 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig). IEEE, 2013.

[5] Hidalgo Martínez, Ginés. "Implementing an Embedded Linux System in Xilinx Zynq." (2015).

[6] Yeniçeri, Ramazan, and Yakup Hüner. "HW/SW codesign and implementation of an IMU navigation filter on Zynq SoC with Linux." 2020 7th International Conference on Electrical and Electronics Engineering (ICEEE). IEEE, 2020.