Analyzing the Potential of Transport Triggered Architecture for Lattice-based Cryptography Algorithms

Latif Akçay[1] and Berna Örs[2]

[1] Bayburt University, Turkey, lakcay@bayburt.edu.tr
[2] Istanbul Technical University, Turkey, orssi@itu.edu.tr

Abstract. Lattice-based structures offer considerable possibilities for post-quantum cryptography. Recently, many algorithms have been built on hard lattice problems. The three of the remaining four in the final round of the post-quantum cryptography standardization process use lattice-based methods. Especially in embedded systems, these algorithms should be operated effectively. In this study, the potential of transport triggered architecture is examined in this sense. We try to compare open source RISC-V processors with our transport triggered architecture processors under fair conditions. Thus, we aim to provide a base architecture for developing application specific processors for post-quantum cryptography, which is becoming an increasingly urgent research area. The tests performed are implemented on the same FPGA and evaluated as performance, resource utilization, average power and total energy consumption. Regardless of the algorithm, our design exhibit better results than RISC-V processors for all tests. It seems to be 2x - 3x faster, 2x - 2.5x smaller and consumes 2.5x - 5x less energy than RISC-V competitors. We also share how the results vary for many different configurations of our processor that can be easily converted. The obtained findings show that the transport triggered architecture is a promising option on developing application-specific processors for lattice-based post-quantum cryptography applications.

Keywords: TTA, RISC-V, lattice-based cryptography, post-quantum cryptography, processor architecture, application specific processor, Kyber, NewHope, Saber, NTRU.

1 Introduction

Quantum computer development studies have been accelerating in recent years. Many large-scale companies are running very high-budget projects and announcing the superiority of quantum computers they have developed one after another [1]. It is clear that this competition has contributed positively to the acceleration of the technology development process. All of these point to the active use of quantum computers in the near future. However, this development poses a very serious threat to the cryptography methods widely used today. Public-key cryptography (PKC) algorithms are based on difficult problems that require quite a long time to solve for classical computers [2]. The factorization problem of large numbers used in the RSA method and the discrete logarithm method used in the ECC algorithm are the most important of these algorithms [3], [4]. However, it has been shown that both problems can be solved much faster by using a large enough quantum computer and Shor's algorithm [5]. To put it more clearly, today's most common PKC systems lose their reliability in the face of quantum computers. Of course, studies have been carried out for a while to develop solutions to this issue. Post-quantum cryptography (PQC), which is a novel research field, refers to safe algorithms against attacks using

both classical computers and quantum computers [6]. Related to this, NIST started a standardization process in 2016 [7]. The main goal is to develop PQC algorithms that can be effectively used effectively in various fields and to create new standards. The first round was started with 69 suitable and completed candidates. In the second round, it was reduced to 26 and then reduced to 7 algorithms in June 2020 in the third round. Draft standards will be announced between 2022 and 2024 according to the process calendar [8].

Lattice-based cryptography, as the name suggests, are structures based on hard mathematical problems defined on lattices [9]. Well-known examples of these are the shortest vector problem or the closest vector problem [10]. The importance of these is that they may provide very appropriate solutions for PQC. Because five of the remaining seven candidate PKC and digital signature algorithms in the last round are based on lattice-based problems [11]. Therefore, it is quite possible that these methods will form the basis of security systems in the quantum age. In addition, the use of these algorithms is a critical requirement not only for the future but also today. Because encrypted information that is somehow recorded today can be decrypted using a quantum computer in the following years. Therefore, PQC is an urgent need to be met.

Another issue that is as important as the security of an algorithm is the effective implementation. This is especially critical in embedded systems that require low power consumption or limited resource usage. PKC is used in most embedded system applications where resource constraints and power consumption are needed to be very low [12]. For example, smart cards, RFID systems, wireless sensor networks are among these applications are among these. In this case, effective processor design stands out as the most important parameter. Developing application specific processors (ASIP) for such areas is an effective solution. Open source and license-free processor architectures are very favorable for ASIP designs in many aspects. In order to develop and use PQC applications in real embedded systems, it is very necessary to research on appropriate processor architectures.

RISC-V is the most popular open source license-free instruction set architecture (ISA) which has a lot of support like compiler, simulator, debugger, etc [13]. Transport-triggered architecture (TTA) is also a well suited method for developing application-specific processors [14]. It is a flexible approach that is very similar to the Very Long Instruction Word (VLIW) architecture [15] but includes some more advantageous features. We consider TTA as an ideal architecture for PQC algorithms, thanks to its feature that enables instruction level parallelism [16].

In this study, TTA processors are compared with the popular RISC-V designs for lattice-based post-quantum cryptography algorithms. We present our sample 64-bit TTA processor and use two different RISC-V cores as competitors. After compiling and running the reference C codes for all processors, we analyze the results from many angles. Also, we rethink the same analyzes for many different configurations of our design.

This work is organized as follows: The second section gives the literature review while the third one explains PQC over lattice-based methods. The fourth section introduces the features and development environment of TTA. In the fifth chapter, we share the design details of the RISC-V and TTA processors used in the study. Experimental tests and comparisons constitute the sixth section. Finally, we conclude the paper and explain our inferences with future plans in the seventh section.

2 Literature Review

There are many studies on the implementation of PQC algorithms from different perspectives. Since the subject is critical and urgent, it can be easily predicted that these studies will increase gradually. There is a comprehensive survey that summarizes the software and hardware PQC implementations found in the literature [17]. Pure software libraries offered as open source code provide a very useful environment for researchers and students

[18]. An optimization study for Number Theoretic Transform (NTT) library [19], using Intel AVX2, was done by Gregor Seiler for [20]. In addition to the projects developed for general purpose processors, there are also studies especially for resource constrained systems [21]. Case studies customized to run on the ARM Cortex-M4 platform can be found in [22], [23] and [24]. Some other interesting software works for limited devices can be seen from [25], [26], [27] and [28].

Hardware-related works can be examined under several different headings: Instruction set extension for PQC, custom hardware designs and hardware-software co-design approach. There are two remarkable studies under the title of instruction set extension for PQC. Fritzman et al. has made extensive customization on Pulpino, a simple RISC-V core, to speed up lattice-based cryptography operations [29]. From a similar point of view, Erdem Alkim et al. achieved very successful results with NTT accelerators built on VexRiscv [30].

To the best of our knowledge, there are not many dedicated hardware studies in the literature. However, a NewHope implementation by Tobias Oder etal can be found in [31]. Similarly, a hardware implementation of Saber is presented in [32]. A compact hardware implementation of Kyber on FPGA is shared in [33]. Another hardware study for NTRU is done in [34].

Hardware-software co-design studies are firstly done for NTRU [35], [36]. Both include a software task running on ARM Cortex-A53 processor with hardware accelerator. A similar study using the Saber algorithm can be found in [37]. [38], [39] and [40] cover the implementation of various PQC algorithms on RISC-V processors with hardware accelerators.

To the best of our knowledge, our previous study is the only one in the literature on PQC using TTA processors [41]. In that work, an example NTRU based public-key cryptosystem is tested on several 32-bit RISC-V and TTA processors and then the results are compared with various parameters. Thus, the efficiency of the two architectures is evaluated for NTRU algorithm.

Our Contribution. This paper provides two contribution. First, we share the test results of performance, power consumption, chip area and total energy comparisons for lattice-based PQC algorithms on two different processor architectures. The second one is that we suggest a base platform which has many advantages on developing application specific processor for lattice-based PQC in embedded systems. We support our ideas with detailed comparative test results.

3 Lattice-based Post-Quantum Cryptography

By the simplest mathematical definition, a lattice is the combination of integer linearly independent base vectors. Lattice-based cryptography is the general term given to structures defined using difficult problems defined on adequately sized lattices. As stated in Section 1, there is no known solution method of these problems that can be accelerated by classical or quantum computers. The Shortest Vector Problem (SVP) is the first of these problems to be raised [9]. In short, it is the problem of finding the shortest non-zero vector in a given lattice and it's base vectors.

Most lattice-based cryptography algorithms use the Learning with Errors (LWE) problem or variants thereof. The LWE, which is a generalized version of the parity learning problem, was introduced by Oded Regev in 2005 [42]. The LWE problem is defined as finding a function over a finite field ring from given samples where some of the samples contain errors. There are two variants of the problem: Ring Learning with Erros (R-LWE) and Module Learning with Errors (M-LWE). In the R-LWE version, polynomials with degrees less than a known n are used. On the other hand, M-LWE is the same as R-LWE, but with the difference, it replaces the elements in the same ring with modulo elements. Thanks to using modulo numbers, it gets easier to arrange security level and efficiency.

In addition to all, the Learning with Rounding (LWR) problem is known as a modified version of the random property of LWE. The LWR problem uses vectors generated with a deterministic change instead of a random error vector. Similarly, the problem is called as Module Learning with Rounding (M-LWR) when the elements are chosen from modulo.

Lattice-based cryptography algorithms in the NIST PQC standardization process are based on different lattice problems. We briefly describe these algorithms in the following subsections.

3.1 Nth Degree Truncated Polynomial Ring Units (NTRU)

NTRU is a ring-based public key cryptosystem based on SVP [43]. There were three different NTRU-based key encapsulation mechanism (KEM) suggested to the NIST PQC standardization process which are: NTRUEncrypt, NTRU-HRSS-KEM, and NTRU Prime. Currently, the latest submission is merger of NTRUEncrypt and NTRU-HRSS-KEM and named just as NTRU [43]. The authors claim that the KEM is indistinguishable under chosen ciphertext attack (IND-CCA2). The submission package contains two narrowly defined families of parameter sets which are called NTRU-HPS and NTRU-HRSS [43]. An NTRU system is parameterized by coprime positive integers (n, p, q), sets of integer polynomials (Lf, Lg, Lr, Lm), and an injection $Lift : Lm \rightarrow Z[x]$. However, there are some minor differences between these two sets of parameters, such as the selection of the value of q and the properties of the polynomial sets. The recommended sets for NTRU-HPSS are ntruhps2048509 ($n = 509$ and $q = 2048$), ntruhps2048677 ($n = 677$ and $q = 2048$) and ntruhps4096821 ($n = 821$ and $q = 4096$). These sets provide different NIST security levels as 1, 3 and 5 respectively. For NTRU-HRSS, the only recommended parameter set is ntruhrss701 ($n = 701$) and it claims the NIST security level 3.

NTRU operates in a truncated polynomial ring $R = Z[X]/(X^n - 1)$ where all polynomials have integer coefficients and degree at most $n - 1$. The general working way of the algorithm can be summarized as follows: To generate a key, one must produce two ternary polynomials f and g in the ring R. Additionally, the polynomial f must have inverses for modulo p and modulo q ($f.fp \equiv 1 \pmod{p}$, $f.fq \equiv 1 \pmod{q}$). Then, the public key h is generated as in 1.

$$h \equiv p.fq.g \pmod{q} \tag{1}$$

One can send a message m by using the public key h and the encryption formula in 2.

$$e \equiv r.h + m \pmod{q} \tag{2}$$

where the r is a random blinding polynomial in the ring R. To decrypt the ciphered message e, 3, 4 and 5 must be applied by the receiver.

$$a \equiv f.e \pmod{q} \tag{3}$$

$$b \equiv a \pmod{p} \tag{4}$$

$$c \equiv fp.b \pmod{p} \equiv fp.f.m \pmod{p} \equiv m \pmod{p} \tag{5}$$

3.2 Crystals-Kyber

Kyber is an IND-CCA2 secure KEM based on M-LWE problem in lattices [44]. It uses a public key encryption (PKE), which is claimed to be indistinguishable under chosen plaintext attack (IND-CPA), called Kyber.CPAPKE, together with Fujisaki–Okamoto (FO) transform [45] to obtain an IND-CCA2 secure KEM, which is called Kyber.CCAKEM [44]. It creates Cryptographic Suite for Algebraic Lattices (CRYSTALS) together with the Dilithium scheme developed for the quantum-resistant digital signature method. Both methods are in the third round finalists of the NIST PQC standardization process.

Kyber.CPAPKE uses few fixed and variable integer parameters $n = 256$, $q = 3329$, $m = 2$, $k = \{2, 3, 4\}$, $du = 10$, $dv = \{3, 4, 5\}$ to set the lattice dimensions and thus the security level. Accordingly, three different Kyber.CCAKEM parameter sets specifying the NIST security level are set with the parameter k as: Kyber512 ($k = 2$, NIST level 1), Kyber768 ($k = 3$, NIST level 3) and Kyber1024 ($k = 4$, NIST level 5).

To put it simply, the key generation process is started by generating a random matrix $A \in Rq^{kxk}$ and vectors s, e by using centered binomial distribution method $B\eta$ [46]. Then, the vector t is computed as in 6. So, the public key pk and the secret key sk is encoded from t and s.

$$t = As + e \tag{6}$$

To encrypt a message $m \in Rq$, matrix $A \in Rq^{kxk}$, vectors e_1, $r \in Rq^k$ and $e_2 \in Rq$ are sampled using the same distribution method $B\eta$. Then, vector $u \in Rq^n$ and vector $v \in Rq$ are computed as in 7 and 8. Vector u and vector v form the ciphertext c.

$$u = ATr + e_1 \tag{7}$$

$$v = tTr + e_2 + m \tag{8}$$

For decryption, the vectors u and v are obtained from the ciphertext c. Then, the original message m is simply computed as in 9.

$$m = v - sTu. \tag{9}$$

3.3 Saber

Saber is an IND-CCA secure KEM which uses the M-LWR problem in lattices [47]. Actually, the name Saber refers to a family of cryptographic primitives. Saber is one of the finalists of the NIST PQC standardization process. The submission package contains Saber.PKE which is an IND-CPA secure encryption scheme. By using FO transform method, Saber.PKE is transformed into Saber.KEM which is an IND-CCA secure key encapsulation mechanism [47].

Saber.KEM has three parameter sets which are called LightSaber, Saber and FireSaber. This ranking also points to NIST security levels 1, 3 and 5 respectively. There are few integer parameters used in Saber methods. The fixed parameters $n = 256$ and $q = 2^{13}$ determines the polynomial ring $Zq[X]/(X^n + 1)$, which are same for all parameter sets. Parameter l shows the module dimension while p (2^{10}, another fixed element for all parameter sets) and T are used for rounding. Coefficients of secret vectors are sampled using binomial distribution with the parameter μ. The LightSaber uses the variable parameters as $l = 2$, $T = 2^3$, $\mu = 10$. For Saber, the same parameters are used as $l = 3$, $T = 2^4$, $\mu = 8$ while the FireSaber uses them as $l = 4$, $T = 2^6$ and $\mu = 6$.

Key generation steps start with randomly sampled matrix $A \in Rq^{lxl}$ by using a pseudo random function with a seed value $seedA$. Then, a secret vector s is generated with the parameter μ by using the binomial distribution. Vector b is computed as in 10. Here, the h is a constant vector defined in the algorithm which is used to replace rounding operations by a simple bit shift. So, public key pk is obtained as $pk = (b, seedA)$ and the secret key sk is $sk = s$.

$$b \equiv (AT.s + h) \pmod q \tag{10}$$

To encrypt a message m, one first need to obtain matrix A by using the $seedA$ in the public key. Then by using the secret vector s' and matrix A, the vector b' is computed as in 11. The other part of the ciphertext v' is obtained using 12. So, the ciphertext c is constructed together with encoding of $cm \equiv (v' + m) \pmod p$) and b'.

$$b' \equiv (A.s' + h) \pmod q \tag{11}$$

$$v' \equiv (bT.s') \pmod{p} \tag{12}$$

The decryption process is rather simpler than the encryption. Vector v is computed with 13 and the original message m is converted from 14.

$$v \equiv b'Ts \pmod{p} \tag{13}$$

$$m \equiv v \pmod{p} \tag{14}$$

3.4 NewHope

NewHope is a KEM whose security relies on the hardness of the R-LWE problem in lattices [48]. It was a NIST PQC candidate until the third round. Although rated as a high performing candidate in the third round evaluation, it was eliminated due to some disadvantages compared to the M-LWE based Kyber [11]. Despite this is the case, we do not exclude NewHope as the main subject of this study is to investigate architectures proper for lattice cryptography.

NewHope package contains two different KEM called NewHope-CPA-KEM and NewHope-CCA-KEM [48]. Similarly, as in Kyber and Saber, NewHope-CCA-KEM is built with FO transform from NewHope-CPA-KEM to provide CCA security model. NewHope offers two NIST security level 1 and 5 with two different parameter sets. This is done by just changing the degree of polynomial paremeter n. Modulos parameter $q = 12289$ and noise distribution parameter $\mu = 8$ don't change while the parameter $n = 512$ for NEWHOPE512 and $n = 1024$ for NEWHOPE1024. Noise generation methodology is also the centered binomial distribution for NewHope.

For key generation, a uniformly random polynomial $a \in Rq$ is produced using a random seed value $seedA$. Then, secret vector s and error vector e are sampled by using parameter μ. After that, the vector b is computed using 15 and packed with $seedA$ as the public key while the secret key is keeped as s.

$$b = As + e \tag{15}$$

To encrypt a message m, another secret vector s' and errors e', e'' are sampled. Then, the vector v is computed as in 16 and the vector u is computed as in 17. Finally, the ciphertext is obtained as the concatenation of v and u.

$$v = b.s' + e'' + m \tag{16}$$

$$u = a.s' + e' \tag{17}$$

Decryption step is the easiest one of the system. Using its secret key s, one can decrypt the original message m using 18. Both encoding and decoding stages use special functions to convert message space to the ring space or vice versa.

$$m = v - us \tag{18}$$

4 TTA based Co-design Environment (TCE)

TTA is a highly customizable processor design methodology where the existed instructions determines the datapath [14]. Although it is a VLIW-like architecture, there is major difference on register file design approach. The both architecture contains functional units (FUs) in which the desired instructions are existed. In VLIW, multiple port register files are always connected to the FUs while TTA offers separate register files connected to the network via transport buses just as the other FUs. Besides, result of an operation can be transfered to another functional unit instead of a register file. For this reason, register file

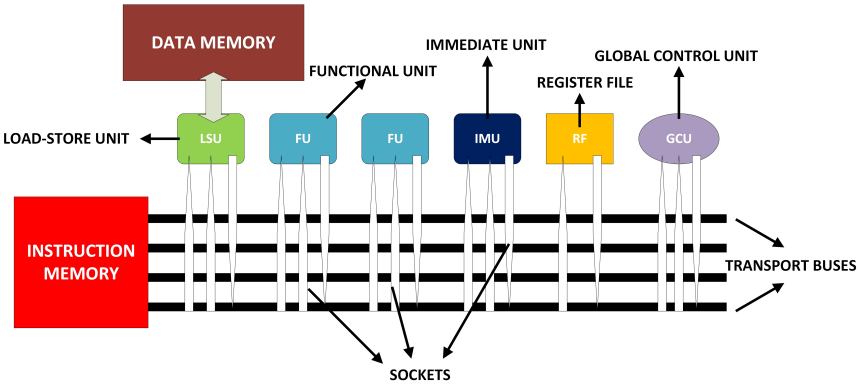

Figure 1: General Structure of a TTA Processor.

access rate is generally low in TTA processors. The foremost feature of the architecture is the multiple transport buses. This capability strongly supports the instruction level parallelism. General structure of a TTA processor is shown Fig. 1.

TCE is an open source and free design environment led by the Customized Parallel Computing (CPC) group at the Tampere University, Finland [49]. It includes several tools that provide compilation, simulation, modeling, profiling and even HDL code generation for TTA processors. The toolset also contains many pre-designed hardware databases (HDB), architecture definition files (ADF) and test codes for the tools.

TCE compiler (tcecc) is an LLVM-based retargetable platform that takes application code and ADF file as inputs and generates parallel binary output. ADF is text-based format that can be written by hand or created by the processor designer (ProDe) which is the processor modeling and RTL generator tool of the TCE. Cycle-accurate simulation and profiling can be done by using TTA simulator (ttasim). These three main tools, and of course the others not detailed here, provide a highly effective development environment for designers.

At the time of this study, TCE supported data types with a maximum width of 32-bit. An experimental 64-bit support was available in the Github repository [1]. However ProDe was still not 64-bit compatible. Also, there was not any FU or load-store unit (LSU) with 64-bit implementation. We use reference C codes written for the lattice-based PQC algorithms in this study [2]. These implementations require 64-bit support. Thus, we needed a fully 64-bit compatible TCE toolset. We solved the issues about ProDe with the help of the TCE developers. Then, we designed several 64-bit arithmetic logic units (ALU) and LSUs. In this way, we became able to design and customize 64-bit TTA processors for all PQC algorithms. All the developments we have achieved are shared with the TCE team.

5 64-bit RISC-V and TTA processors

In this section, we share the technical details of the RISC-V processors used in study. Also, we explain our 64-bit TTA processor design to compare RISC-V rivals. In our previous study[41], we compared 32-bit platforms and obtained results that can give an idea. Therefore, in this study, we develop 64-bit TTA processors that can run reference C implementations of the algorithms in the NIST PQC process.

[1] https://github.com/cpc/tce
[2] https://github.com/PQClean/PQClean

5.1 Rocket Core

Rocket Core is single-issue, in-order, scalar processor core member of the Rocket Chip Generator family developed in University of California at Berkeley [50]. The Rocket Chip Generator is a system written in Chisel hardware construction language [51] and designed to generate cores, caches or interconnects for System-on-Chip construction. Rocket Core implements RV64G (RV64IMAFD) variant of RISC-V ISA [13] and contains a memory management unit (MMU) to support running operating systems. It has a 5-stage classical RISC pipeline and an optional IEEE compliant floating point unit (FPU) which is also enabled in this work. A simple core model of a RISC design can be seen in in Fig. 2. More detailed technical information about the core can be found in [50].

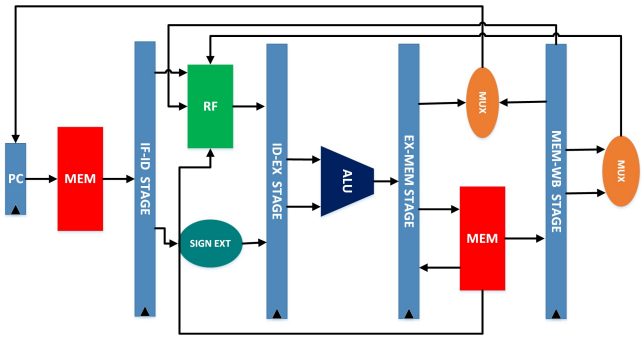

Figure 2: A Simple Model of Classical 5-Stage RISC Pipeline.

In order to achieve consistent results in comparisons of power consumption, resource utilization or total energy consumption, we omit units such as MMU and caches from the Rocket Core design.

5.2 CVA6

CVA6 is a single-issue, in-order,scalar processor core formerly named Ariane, developed by PULP Team, ETH Zurich [52]. It implements RV64GC (RV64IMAFD + compressed instruction set 'C') of RISC-V ISA and has a 6-stage pipeline. CVA6 is called as an application class processor and supports three privilege level to run Unix-like operating systems. It has an optional IEEE compliant FPU just like the Rocket Core. Ariane Core is also used in OpenPiton project [53]. More detailed technical information about the core can be found in [52].

In order to achieve consistent results in comparisons of power for lattice-based PQC, resource utilization or total energy consumption, we omit units such as MMU and caches from the CVA6 design.

5.3 Our 64-bit TTA Design: TTA64

Before going into the details of the designed processor, we want to emphasize that it is not a processor customized for lattice-based PQC. We want to present an example of 64-bit TTA concept that can be used for general purposes just like the RISC-V processors. But of course, we prefer instructions that are executed more frequently when creating FUs. Nevertheless, we are careful to stay in a level that covers a much smaller amount of chip area compared to RISC-V processors.

We simply call our processor TTA64. It has six 64-bit transport buses, one LSU, three FUs, two register files (RFs), a boolen RF and a global control unit (GCU). The structural model of TTA64 which is developed using ProDe is given in Fig. 3.

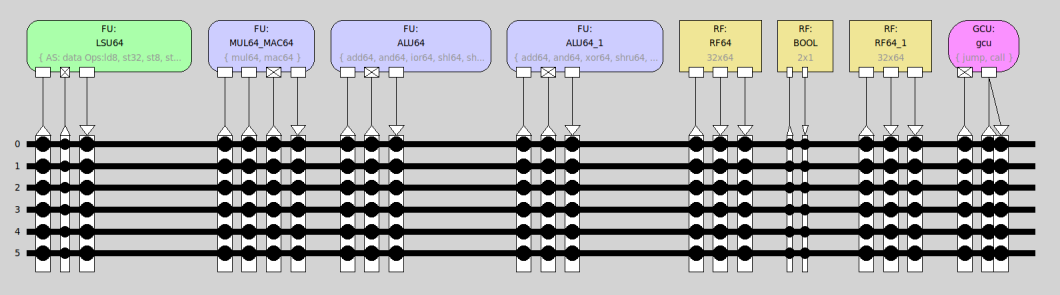

Figure 3: TTA64 Structural Model in TCE-ProDe.

LSU design in our processor supports 8-bit, 16-bit, 32-bit and 64-bit load and store operations. There are eleven instructions (**st8, st16, st32, st64, ld8, ld16, ld32, ld64, ldu8, ldu16, ldu32**) in the design. Three of them are used for loading data from memory and zero-extending. We prefer little-endian, but it's not a must in the TCE environment.

We use two RFs which are both 64-bit wide 32 registers. The reason behind the double RF usage is to show the capability of employing separate register units and to increase the performance. The TCE provides a parametric RF design. So, we didn't have to make a new design for 64-bit support. Also, we use a simple boolen RF which just contains two 1-bit registers which are generally used as flags for **jump** and **call** operations by GCU.

TTA64 contains three FUs which are ALU64, ALU64-1 and MULMAC64. ALU64 unit contains following instructions;

- **add64:** Add 64-bit signed values
- **and64:** Logical AND operation
- **eq64:** Set if two 64-bit values are equal
- **gt64:** Set if the first value greater than the second
- **gtu64:** Set if the first value greater than the second (unsigned numbers)
- **ior64:** Logical Inclusive OR operation
- **ltu64:** Set if the first value less than the second (unsigned numbers)
- **ne64:** Set if two value are not equal
- **shl64:** Left logical shift
- **shl1add64:** Array indexing for 64-bit values
- **shr64:** Right logical shift
- **shru64:** Logical shift right (most significant bits zeroed).
- **sub64:** Subtraction for 64-bit signed values
- **sxh64:** Sign extend from 32 bits of the input 1 to 64 bits into output 2.
- **sxw64:** Sign extend from 16 bits of the input 1 to 64 bits into output 2.
- **xor64:** Logical Exclusive OR operation

Table 1: Comparison of RISC-V and TTA64 Processors According to Physical Characteristics

Processor	IWL	Frequency (MHz)	Resources (LUT, FF, DSP)	Power (mW)
ROCKET	64	62.5	16854, 5748, 24	192
CVA6	64	40	19602, 8761, 27	219
TTA64	290	42	7865, 5875, 20	136

ALU64-1 is a subset of ALU64 and includes the most repeated instructions. Although these are not exactly the same for all PQC algorithm, we prefer common ones. Thus, we increase the performance considerably by integrating another separate FU. We determine these instructions using the profiling feature of the ttasim. ALU64-1 contains add64, and64, shl64, shru64, sxh64, xor64. PQC algorithms need many multiplication operations. For fast calculation of these operations, we use MULMAC64 FU. A separate FU design is more useful for these calculations. Because number of needed inputs and operational delay are different from other ALU instructions. MULMAC64 contains following instructions;

- **mul64:** Multiplication of the inputs with lower result bits in the output.
- **mac64:** Multiply and accumulate (signed long).

Much more customization is possible for designing an ASIP for lattice-based PQC. However, the only concept we want to introduce here is just a 64-bit TTA processor which has a very flexible architecture. We believe this will be a more effective solution than RISC alternatives. We support this claim with the tests we present in the next section.

6 Experiments

We choose Xilinx Artix-7 series FPGA (XC7A100T-1CSG324C) as the hardware test platform [54]. First, we arrange all compared processor cores to a similar configuration as indicated in Section V. But of course, instruction word length (IWL) of the processors are inherently different from each other. We obtain maximum clock frequency, resource utilization and estimated average power consumption values by using post-synthesis analysis of the Xilinx Vivado tool. The comparison results in terms of these parameters and IWL of each processor can be seen in Table 1. Maximum clock frequency of the Rocket Core is clearly higher than the other cores. But, when considering the occupied chip area and average power consumption, TTA64 appears to be quite small and lower power than the RISC-V rivals.

For performance analysis, we compile exactly the same reference PQC C codes using the same optimization flags (-O3) on all three platforms. Then, we run these executables on the processors to find out both the accuracy of the results and the total number of clock cycles (NoC) to complete the processes. As seen in Table 2, the NoC of TTA64 is quite best while the RISC-V processors present relatively close values as expected.

From the perspective of algorithms, Kyber parameter sets stands out as the fastest solutions for all NIST security levels they claim. There is an interesting competition between the NewHope and Saber algorithms, which varies according to the processors. While NewHope-512 and NewHope-1024 run faster than LigthSaber and FireSaber on RISC-V processors respectively, the situation is opposite for TTA64. This can be taken as an indication that the C implementation of Saber algorithms are more prone to instruction level parallelism. On the other hand, all NTRU options take the longest time to execute with very high values.

Since one of the main concerns of this study is the performance analysis of the TTA64 processor, we share a more detailed NoC result. We present how many clock cycles the

Table 2: Performance Comparison of RISC-V and TTA64 Processors

PQC ALGORITHM	ROCKET	CVA6	TTA64
KYBER-512-CCA2	3608812	4277745	1301288
KYBER-768-CCA2	5491484	6382248	1963856
KYBER-1024-CCA2	7690692	8829582	2860485
NEWHOPE-512-CCA2	4098804	4166808	2559512
NEWHOPE-1024-CCA2	7707288	7865157	5459508
LIGHTSABER-CCA2	5620808	5804118	1669368
SABER-CCA2	10123716	10071750	3214911
FIRESABER-CCA2	15841608	15557466	5228696
NTRUHRSS701-CCA2	168227892	186277767	75512621
NTRUHPS2048509-CCA2	91814912	105486627	42592928
NTRUHPS2048677-CCA2	161745040	185478378	68401723

Table 3: Detailed Performance Analysis of TTA64 Processor

PQC ALGORITHM	Key. Gen.	Encaps.	Decaps.
KYBER-512-CCA2	270236	482543	548509
KYBER-768-CCA2	427773	725929	810154
KYBER-1024-CCA2	641559	1055373	1163513
NEWHOPE-512-CCA2	565342	892045	1101125
NEWHOPE-1024-CCA2	1195627	1909724	2354157
LIGHTSABER-CCA2	409944	565803	693621
SABER-CCA2	847059	1085971	1281881
FIRESABER-CCA2	1440897	1760122	2027677
NTRUHRSS701-CCA2	71337818	1395228	2779575
NTRUHPS2048509-CCA2	39165957	1559901	1867070
NTRUHPS2048677-CCA2	63501100	2179514	2721109

processor takes for key generation, encapsulation and decapsulation steps for all tested algorithms in Table 3. The key generation is the longest process for all NTRU parameter sets as expected. But, it is generally the shortest operation for other algorithms. The common feature of all options is that the decapsulation takes more time to execute than the encapsulation.

The other fundamental parameters for processor design are average power and total energy consumption. We give comparative values for the three processors in Table 4. While calculating the total energy consumption value, we apply the well-known Eq. 19 where $E, P, t, \text{NoC}, f_{max}$ stands for total energy consumption, average power, total execution time, total number of clock cycles and maximum clock frequency respectively. When the table is examined, it becomes clear that the TTA64 processor is extremely advantageous in terms of energy consumption. Although there are slight differences according to the algorithms, TTA64 is very promising as it consumes approximately 2.5x less energy than Rocket Core and almost 5x less than CVA6.

$$E(t) = P \times t = P \times \text{NoC} \times \frac{1}{f_{max}} \tag{19}$$

6.1 Analysis for Six Different TTA64 Configurations

We set up another test scenario for several configurations of TTA64 processor to analyze the effect of FUs and transport buses on performance and the other results. First, we reduce the number of transport buses of TTA64 from 6 to 4 and name the new processor as

Table 4: Comparison of Total Energy Consumption (mJ) for RISC-V and TTA64 Cores

PQC ALGORITHM	ROCKET	CVA6	TTA64
KYBER-512-CCA2	11.09	23.42	4.21
KYBER-768-CCA2	16.87	34.94	6.36
KYBER-1024-CCA2	23.63	48.34	9.26
NEWHOPE-512-CCA2	12.59	22.81	8.28
NEWHOPE-1024-CCA2	23.68	43.06	17.67
LIGHTSABER-CCA2	17.27	31.78	5.40
SABER-CCA2	31.10	55.14	10.41
FIRESABER-CCA2	48.67	85.18	16.92
NTRUHRSS701-CCA2	516.80	1019.87	244.42
NTRUHPS2048509-CCA2	282.06	577.54	137.86
NTRUHPS2048677-CCA2	496.88	1015.49	221.40

Table 5: Configurations of Six Different 64-bit TTA Processors

Processor	IWL	Bus	Functional Units	Register Files
TTA64-P1	204	4	LSU64, ALU64, ALU64-1, MULMAC64	2xGP, 1 BOOL
TTA64-P2	204	4	LSU64, ALU64, MULMAC64	2xGP, 1 BOOL
TTA64-P3	204	4	LSU64, ALU64, ALU64-1	2xGP, 1 BOOL
TTA64-P4	118	2	LSU64, ALU64, ALU64-1, MULMAC64	2xGP, 1 BOOL
TTA64-P5	204	4	LSU64, ALU64, 2xMULMAC64	1xGP, 1 BOOL
TTA64-P6	204	4	LSU64, ALU64, ALU64-1, 2xMULMAC64	1xGP, 1 BOOL

TTA64-P1. Then, we prepare TTA64-P2, TTA64-P3, TTA64-P4, TTA64-P5 and TTA64-P6 processors by just making similar changes. Structural details of these configurations can be read from the Table 5. In addition, we share the ProDe models of TTA64-P1 and TTA64-P2 in Fig. 4, TTA64-P3 and TTA64-P4 in Fig. 5, and TTA64-P5 and TTA64-P6 in Fig. 6.

Maximum clock frequency, resource utilization and average power consumption values for the six revised TTA64 processors are shared in Table 6. As seen, TTA64-P3 has the highest clock frequency with a big difference. This is because the multiplication instructions that prolong the critical path are not available in this processor. However, the same configuration also has the highest average power consumption value while the TTA64-P4 has the lowest. Another feature of TTA64-P3 that differs from other processors is that it does not contain any DSP. Although this seems to be an advantage in terms of chip area, the best option in terms of LUT and FF numbers seems to be TTA64-P5. However, this configuration includes two MULMAC64 FUs, just like the TTA64-P6. This situation causes the highest number of DSP usage for these two processors among others.

The total NoC for the six different TTA64 configurations can be seen in Table 7. Although the best value in terms of maximum clock frequency belongs to the TTA64-P3

Table 6: Physical Characteristics of Six Different 64-bit TTA Processors

Processor	Frequency (MHz)	Resources (LUT, FF, DSP)	Power (mW)
TTA64-P1	44	6614, 5583, 20	126
TTA64-P2	44	5814, 5370, 20	124
TTA64-P3	106	6209, 5306, 0	154
TTA64-P4	44	5493, 5392, 20	118
TTA64-P5	42	5012, 3553, 26	124
TTA64-P6	42	5828, 3760, 26	126

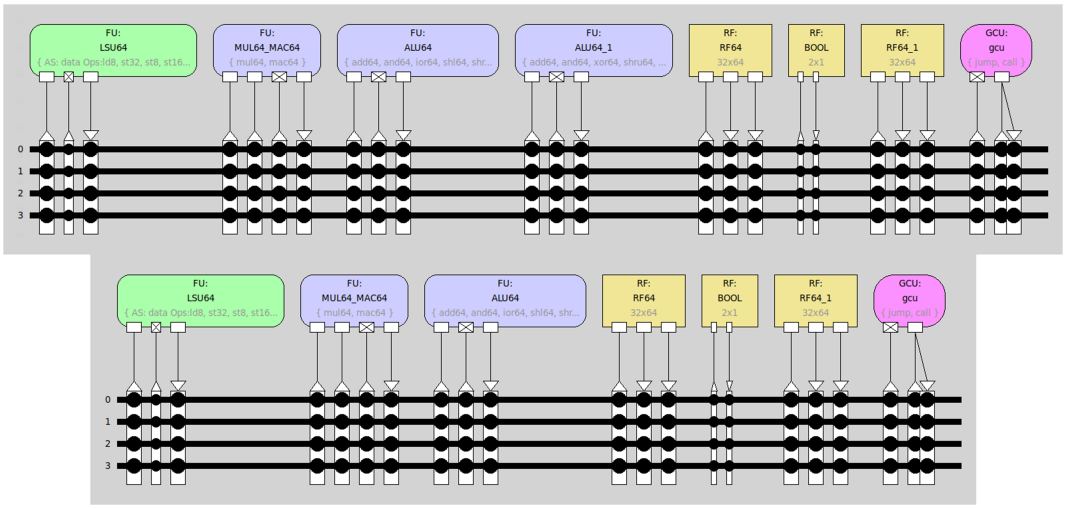

Figure 4: Structural Models of TTA64-P1 and TTA64-P2 Processors in TCE-Prode.

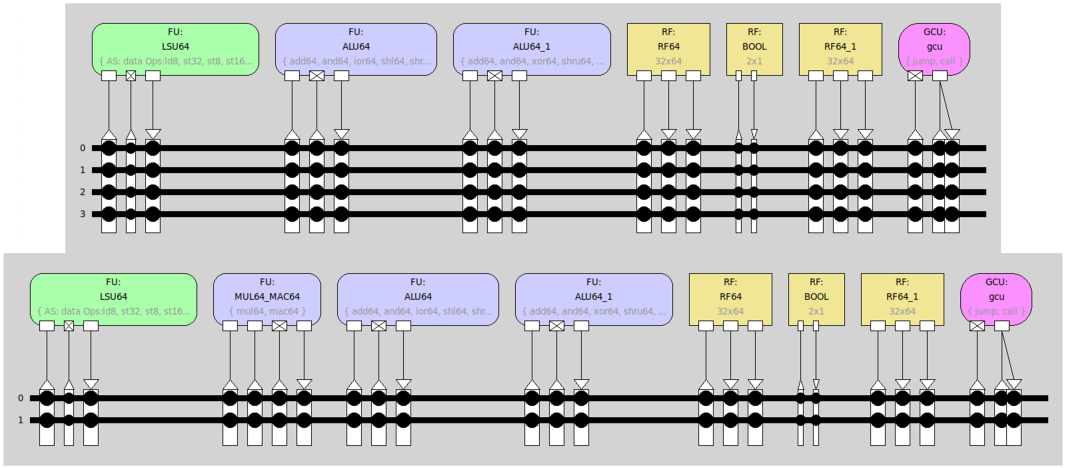

Figure 5: Structural Models of TTA64-P3 and TTA64-P4 Processors in TCE-Prode.

processor, it also exhibits the worst values in terms of the total NoC due to the lack of multiplication operations. However, it is possible to understand from the comparison of TTA64-P1 and TTA64-P6 that the only factor seriously affects the total NoC is not the MULMAC64 FU. As can be seen from the Table 7, the reduction of the RF count worsened the performance despite the increase in the number of MULMAC64. The effect of the number of transport buses on the total NoC can be understood by looking at the values obtained for TTA64-P3.

As expected, average power and total energy consumption values also vary according to the transport bus count and the number of FUs. We apply the same calculations we made for TTA64 and RISC-V processors to the rearranged configurations in the same way. The total energy consumption of these processors can be seen in Table8. The importance of supporting multiplication operations is again evident here. The TTA64-P3 configuration gives the worst results compared to the other processors in terms of total energy consumption. To compare the obtained results all together, we present two different graphs in Fig. 7 and in Fig. 8 which indicate the total NoC and the total energy

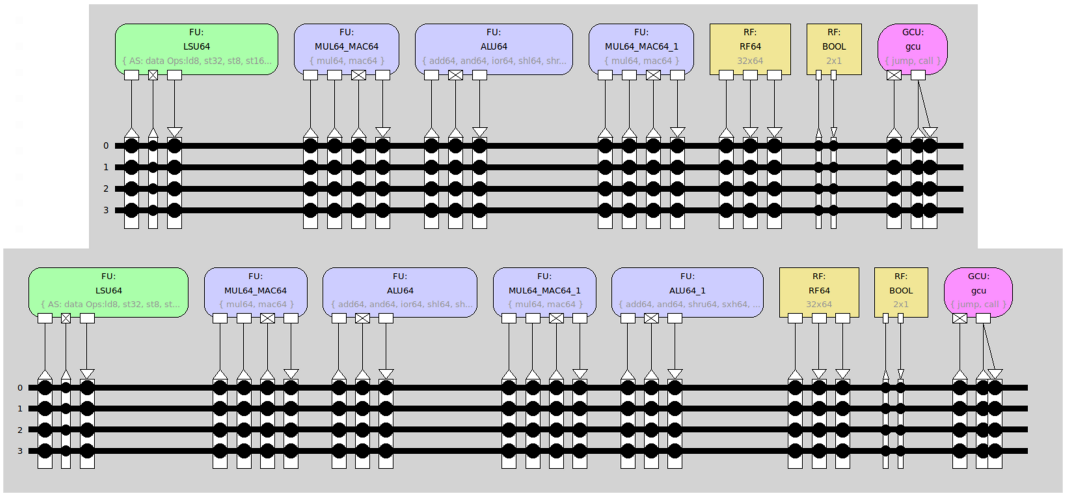

Figure 6: Structural Models of TTA64-P5 and TTA64-P6 Processors in TCE-Prode.

Table 7: Performance Comparison of Six Different 64-bit TTA Processors

PQC ALGORITHM	TTA64-P1	TTA64-P2	TTA64-P3	TTA64-P4	TTA64-P5	TTA64-P6
KYBER-512-CCA2	1463997	1664302	31009699	2136069	1849517	1632415
KYBER-768-CCA2	2218207	2545406	49240559	3286053	2833725	2491214
KYBER-1024-CCA2	3248591	3746299	71005737	4836655	4173794	3654467
NEWHOPE-512-CCA2	2788655	3079498	17350183	3835099	3289969	2993448
NEWHOPE-1024-CCA2	5946513	6504485	37114132	8087257	6915593	6347239
LIGHTSABER-CCA2	2022774	2301096	72343341	3511828	2763006	2481705
SABER-CCA2	3896183	4405555	141601810	6775925	5305394	4761406
FIRESABER-CCA2	6334068	7129859	240265111	11020159	8596350	7794224
NTRUHRSS701-CCA2	83848926	95119293	1509128555	140287666	102293859	91343361
NTRUHPS2048509-CCA2	49210670	55522831	785486018	80762661	57062851	50054799
NTRUHPS2048677-CCA2	79926364	89484623	1396471508	134772177	92938616	82068742

consumption values for both TTA and RISC-V cores respectively. As can be deduced from both graphs, the TTA64-P1 configuration appears to be the most logical option among others for performance and total energy. But, it should be noted that there are better candidates in terms of chip area and average power consumption. On the other hand, all configurations except TTA64-P3 still offer better results than the RISC-V processors.

6.2 Analysis for Different Transport Bus Counts

Finally, we only examine the effect of number of transport buses. To do this, we first take the TTA64 processor as a reference and call it TTA64-R6. We create new configurations as TTA64-R5, TTA64-R4, TTA64-R3, TTA64-R2 and TTA64-R1 by decreasing the number of transport buses one by one from 6 to 1. Then, we repeat performance, average power consumption, resource utilization and total energy consumption analysis. We give the change of processor characteristics in Table 9 for each cores. The total NoC according to the new configurations is given in Table 10. The total energy consumption for all tested

Table 8: Comparison of Total Energy Consumption (mJ) for Six Different TTA64 Cores

PQC ALGORITHM	TTA64-P1	TTA64-P2	TTA64-P3	TTA64-P4	TTA64-P5	TTA64-P6
KYBER-512-CCA2	4.19	4.69	44.99	5.73	5.46	4.90
KYBER-768-CCA2	6.35	7.17	71.43	8.81	8.36	7.47
KYBER-1024-CCA2	9.30	10.55	103.01	12.97	12.32	10.96
NEWHOPE-512-CCA2	7.98	8.68	25.17	10.28	9.71	8.98
NEWHOPE-1024-CCA2	17.02	18.32	53.84	21.68	20.41	19.03
LIGHTSABER-CCA2	5.79	6.48	104.95	9.42	8.15	7.44
SABER-CCA2	11.15	12.41	205.42	18.17	15.66	14.28
FIRESABER-CCA2	18.13	20.09	348.55	29.54	25.37	23.37
NTRUHRSS701-CCA2	240.04	267.98	2189.26	376.11	301.89	273.92
NTRUHPS2048509-CCA2	140.88	156.42	1139.49	216.52	168.40	150.10
NTRUHPS2048677-CCA2	228.81	252.10	2025.83	361.32	274.28	246.11

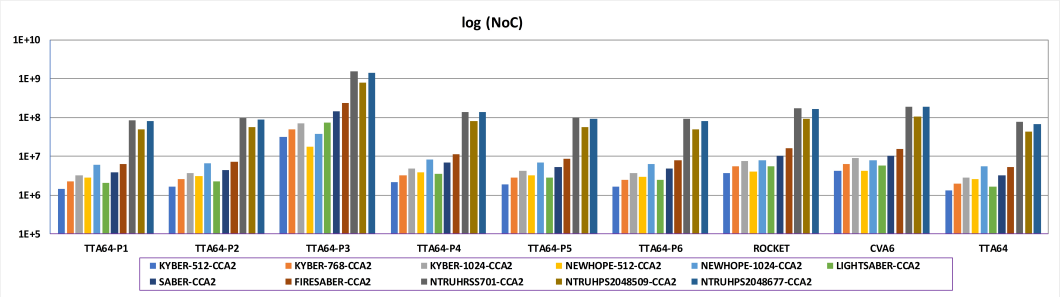

Figure 7: NoC Comparison of RISC-V and Different Configuration TTA Processors.

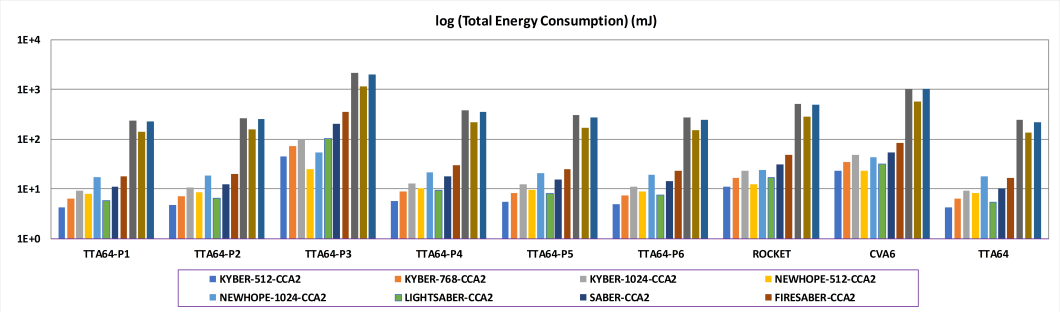

Figure 8: Comparison of Total Energy Consumption for RISC-V and Different Configuration TTA Processors.

algorithms can be found in Table 11.

To visualize the obtained results comparatively, we share two graphs in Fig. 9 and in Fig. 10 which demonstrate the total NoC and the total energy consumption values for both RISC-V cores and TTA processors which have different transport bus counts. We can say that the total NoC increases up to 3x according to the number of transport buses. But, TTA64-R1 which has only one transport bus, is still faster than CVA6. When the number of transport buses are two or more, TTA processors outperform the RISC-V competitors simply. However, as the number of the buses increases, the rate of the rise in performance decreases. This can be seen from the TTA64-R5 and TTA64-R6 which offer almost identical NoC results. A similar picture emerges in the total energy consumption. The TTA64-R1 can still be competitive to the RISC-V cores. TTA64-R2 and the others prove the efficiency of the TTA method clearly. But of course, there is a limit to the advantage of increasing the transport bus count in terms of total energy consumption. That is reached in the TTA64-R5 configuration as it consumes less energy than the TTA64-R6 for all tests.

It is possible to create and analyze many configurations with different FUs or different transport bus counts even more. However, we think that is much sufficient to give an idea for this study. Our main goal here is to demonstrate that TTA processors can be built very flexibly according to the design requirements for various PQC applications.

7 Conclusion

The rapidly growing quantum computer development projects mean to be a very serious threat to classical public key cryptography. Thus, PQC is a very important research area

Table 9: Change of Physical Characteristics of TTA Processors According to Transport Bus Counts

Processor	Frequency (MHz)	Resources (LUT, FF, DSP)	Power (mW)
TTA64-R1	44	4609, 5292, 20	116
TTA64-R2	44	5524, 5393, 20	119
TTA64-R3	44	6076, 5493, 20	123
TTA64-R4	44	6556, 5579, 20	126
TTA64-R5	44	6958, 5679, 20	130
TTA64-R6	42	7865, 5875, 20	136

Table 10: Performance Comparison of TTA Processors According to Transport Bus Counts

PQC ALGORITHM	TTA64-R1	TTA64-R2	TTA64-R3	TTA64-R4	TTA64-R5	TTA64-R6
KYBER-512-CCA2	3725676	2133413	1638493	1463997	1333680	1301288
KYBER-768-CCA2	5771031	3282389	2502356	2218207	2018243	1963856
KYBER-1024-CCA2	8533787	4831661	3672079	3248591	2939161	2860485
NEWHOPE-512-CCA2	6665724	3835067	3019955	2788665	2617075	2559512
NEWHOPE-1024-CCA2	13981951	8086525	6412038	5946513	5609265	5459508
LIGHTSABER-CCA2	6597332	3511671	2513285	2022774	1787194	1669368
SABER-CCA2	12708510	6776020	4849242	3896183	3444993	3214911
FIRESABER-CCA2	20653585	11020278	7893063	6334068	5607110	5228696
NTRUHRSS701-CCA2	248018569	140289534	100455386	83848926	75093436	75512621
NTRUHPS2048509-CCA2	147184538	80758493	57616802	49210670	44432587	42592928
NTRUHPS2048677-CCA2	243752795	134771801	95272929	79926364	72265352	68401723

Table 11: Total Energy Consumption (mJ) According to Transport Bus Counts

PQC ALGORITHM	TTA64-R1	TTA64-R2	TTA64-R3	TTA64-R4	TTA64-R5	TTA64-R6
KYBER-512-CCA2	9.82	5.77	4.58	4.19	3.94	4.21
KYBER-768-CCA2	15.21	8.87	6.99	6.35	5.96	6.36
KYBER-1024-CCA2	22.49	13.06	10.26	9.30	8.68	9.26
NEWHOPE-512-CCA2	17.57	10.37	8.44	7.98	7.73	8.28
NEWHOPE-1024-CCA2	36.85	21.86	17.92	17.02	16.57	17.67
LIGHTSABER-CCA2	17.39	9.49	7.02	5.79	5.28	5.40
SABER-CCA2	33.49	18.32	13.55	11.15	10.18	10.41
FIRESABER-CCA2	54.43	29.80	22.06	18.13	16.56	16.92
NTRUHRSS701-CCA2	653.66	379.30	280.73	240.04	221.80	244.42
NTRUHPS2048509-CCA2	387.91	218.35	161.01	140.88	131.24	137.86
NTRUHPS2048677-CCA2	642.42	364.38	266.25	228.80	213.44	221.40

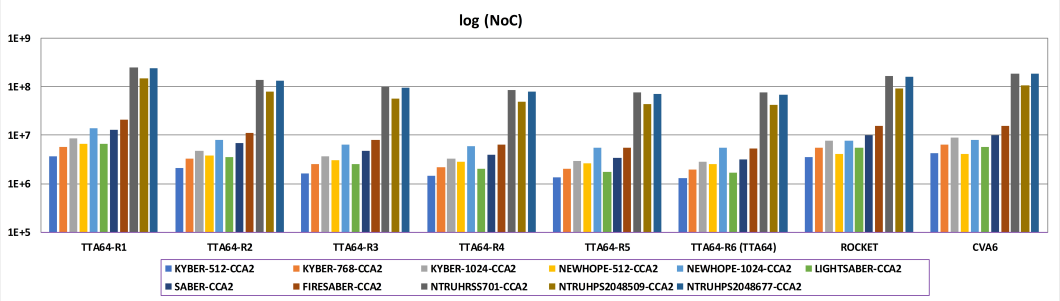

Figure 9: NoC Comparison of RISC-V and TTA64 Processors with Different Bus Counts.

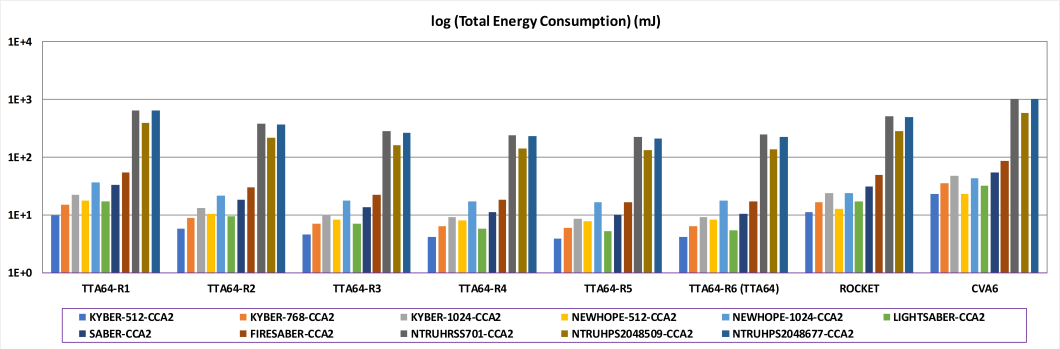

Figure 10: Comparison of Total Energy Consumption for RISC-V and TTA64 Processors with Different Bus Counts.

both today and in the future. In this study, we consider proper processor architectures for PQC in embedded systems. Processor architecture is very critical especially for resource limited devices. Our study analyzes the advantages of TTA as an alternative to classical RISC architecture which is preferred in such systems for PQC.

First, we make the TTA development environment suitable for 64-bit processor design. To do this, we design 64-bit ALU64, LSU64, and MULMAC64 FUs. In addition, we adapt the open source codes of TCE that produce the control unit, so that the fetch and decode units are compatible with the architecture. Finally, we develop TTA64 which is a 64-bit TTA processor designed by integrating FUs with transport buses and RFs. As a competitor to this processor, we identify the Rocket Core and the CVA6 which both have the industry standard 64-bit RISC-V architecture.

Among the PQC methods, the most promising structures are the lattice-based cryptography algorithms. As a matter of fact, three of the four candidate algorithms that remained in the last round of the NIST standardization process are built on hard-lattice problems. In our study, we compare TTA and RISC-V architecture processors in terms of performance, resource utilization, average power and total energy consumption for the reference C implementations of these algorithms. Tests are performed on the same FPGA part by using similar core configurations to provide consistency. We share the obtained results using comparative tables and graphics.

When TTA64 and the compared RISC-V processors are considered, we observe that there is a difference between 2x - 2.5x in terms of resource utilization. By the way, it is possible to say that Rocket Core is a more sparing design than CVA6 in this sense. In addition to the resource utilization data, we see that the Rocket Core has maximum clock frequency value of these three. But the total NoC results are quite impressive. Although 64-bit RISC-V processors offer close values as expected, TTA64 is about 2.5x - 3x faster. When considered together with the maximum clock frequency, the difference in performance is between 2x and 3x. So, it is very promising to deliver at least twice the performance in at least twice the smaller chip area. However, it should be especially noted that while RISC-V competitors have an IEEE compliant FPU, TTA64 does not. Of course, all these seriously affect the average power and the total energy consumption results. When Table 3 is examined, it is easily understood that the TTA processor is more efficient with a very clear difference. As emphasized earlier, TTA64 is not a sufficiently customized design for lattice-based algorithms. In other words, many improvements can be made to obtain much more striking results.

In the second phase of our study, we analyze 64-bit TTA processors with six different configurations, without significantly increasing the chip area or even reducing it even more.

These are not completely different processors but slightly modified versions of the numbers of some units of the first processor. By doing this, we want to show the flexibility of TTA and analyze how much the performance, resource utilization and other parameters change. When we look at the results through configurations, we can easily understand the positive effect of the number of transport buses on performance. Of course, this increases the chip area and total energy consumption a little. Also, a smaller second ALU generated by the most frequently repeated instructions is considered a smart method to further improve the results. However, adding more and more FUs may not notably change the performance, but also negatively affects other factors. Using one more discrete RF is another logical option as it moderately improves the performance.

In the last part of the study, we evaluate the effect of changing only the number of transport buses on the results by keeping FUs same. Thus, we propose a method on how to obtain the most convenient TTA64 configuration according to design requirements. It is very beneficial to increase the number of transport buses up to a certain number, depending on the algorithm and used FUs. It is revealed that the ideal transport bus count for the conditions in this study is 5. More inferences are possible from the results, but we leave more to the readers.

One of the most challenging aspects of application-specific processor design is to provide compiler support. In this respect, TTA again makes things easier thanks to the open source development environment, TCE. Since the compiler automatically adapts to the designed processor, it is possible to compile and run application code instantly. This offers designers a lot of rapid development and debugging opportunities.

Finally, we want to point our future plan. Both this study and our previous study [41] prove us that the TTA is a very powerful method for developing application-specific PQC processors in embedded systems. In the light of these results, we plan to design much more efficient 64-bit TTA processors during upcoming studies. In addition, we will study on techniques to automatically optimize TTA processors.

Acknowledgment

We would like to thank the TCE team for their support for this work and also for the great development environment for TTA processor design.

References

[1] C. S. Calude and E. Calude, "The road to quantum computational supremacy," in *Jonathan M. Borwein Commemorative Conference*. Springer, 2017, pp. 349–367.

[2] A. Salomaa, *PUBLIC-KEY CRYPTOGRAPHY*, ser. EATCS Monographs on Theoretical Computer Science / edited by Wilfried Brauer, Grzegorz Rozenberg, Arto Salomaa. Springer-Verlag, 1990. [Online]. Available: https://books.google.com.tr/books?id=9i-poAEACAAJ

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, Feb. 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[4] V. Kapoor, V. S. Abraham, and R. Singh, "Elliptic curve cryptography," *Ubiquity*, vol. 2008, no. May, pp. 1–8, 2008.

[5] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[6] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[7] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016, vol. 12.

[8] W. Barker, W. Polk, and M. Souppaya, "Getting ready for post-quantum cryptography: Explore challenges associated with adoption and use of post-quantum cryptographic algorithms," National Institute of Standards and Technology, Tech. Rep., 2020.

[9] D. Micciancio and O. Regev, *Lattice-based Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191.

[10] D. Micciancio and S. Goldwasser, *Complexity of lattice problems: a cryptographic perspective*. Springer Science & Business Media, 2012, vol. 671.

[11] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the second round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2020.

[12] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight cryptography for embedded systems – a comparative analysis," in *Data Privacy Management and Autonomous Spontaneous Security*, J. Garcia-Alfaro, G. Lioudakis, N. Cuppens-Boulahia, S. Foley, and W. M. Fitzgerald, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 333–349.

[13] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, "The risc-v instruction set manual. volume 1: User-level isa, version 2.0," California Univ Berkeley Dept of Electrical Engineering and Computer Sciences, Tech. Rep., 2014.

[14] H. Corporaal, "Design of transport triggered architectures," in *Proceedings of 4th Great Lakes Symposium on VLSI*, 1994, pp. 130–135.

[15] J. A. Fisher, "Very long instruction word architectures and the eli-512," *SIGARCH Comput. Archit. News*, vol. 11, no. 3, p. 140–150, Jun. 1983. [Online]. Available: https://doi.org/10.1145/1067651.801649

[16] H. Corporaal and J. Hoogerbrugge, *Code Generation for Transport Triggered Architectures*. Boston, MA: Springer US, 2002, pp. 240–259.

[17] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–41, 2019.

[18] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in *International Conference on Selected Areas in Cryptography*. Springer, 2016, pp. 14–37.

[19] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *International Conference on Cryptology and Network Security*. Springer, 2016, pp. 124–139.

[20] G. Seiler, "Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 39, 2018.

[21] K. S. Roy and H. K. Kalita, "A survey on post-quantum cryptography for constrained devices," *International Journal of Applied Engineering Research*, vol. 14, no. 11, pp. 2608–2615, 2019.

[22] E. Alkim, Y. A. Bilgin, M. Cenk, and F. Gérard, "Cortex-m4 optimizations for {R, M} lwe schemes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 336–357, 2020.

[23] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of kyber on cortex-m4," in *International Conference on Cryptology in Africa*. Springer, 2019, pp. 209–228.

[24] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking nist pqc on arm cortex-m4," Cryptology ePrint Archive, Report 2019/844, 2019, https://eprint.iacr.org/2019/844.

[25] E. Alkim, P. Jakubeit, and P. Schwabe, "Newhope on arm cortex-m," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 332–349.

[26] A. Boorghany, S. B. Sarmadi, and R. Jalili, "On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 3, pp. 1–25, 2015.

[27] Z. Liu, T. Pöppelmann, T. Oder, H. Seo, S. S. Roy, T. Güneysu, J. Großschädl, H. Kim, and I. Verbauwhede, "High-performance ideal lattice-based cryptography on 8-bit avr microcontrollers," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 4, pp. 1–24, 2017.

[28] T. Pöppelmann, T. Oder, and T. Güneysu, "High-performance ideal lattice-based cryptography on 8-bit atxmega microcontrollers," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2015, pp. 346–365.

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

[30] E. Alkim, H. Evkan, N. Lahr, R. Niederhagen, and R. Petri, "Isa extensions for finite field arithmetic," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 219–242, 2020.

[31] T. Oder and T. Güneysu, "Implementing the newhope-simple key exchange on low-cost fpgas," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2017, pp. 128–142.

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

[33] 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.

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

[35] T. Fritzmann, T. Schamberger, C. Frisch, K. Braun, G. Maringer, and J. Sepúlveda, "Efficient hardware/software co-design for ntru," in *VLSI-SoC: Design and Engineering of Electronics Systems Based on New Computing Paradigms*, N. Bombieri, G. Pravadelli, M. Fujita, T. Austin, and R. Reis, Eds. Cham: Springer International Publishing, 2019, pp. 257–280.

[36] F. Farahmand, V. B. Dang, D. T. Nguyen, and K. Gaj, "Evaluating the potential for hardware acceleration of four ntru-based key encapsulation mechanisms using software/hardware codesign," in *Post-Quantum Cryptography*, J. Ding and R. Steinwandt, Eds. Cham: Springer International Publishing, 2019, pp. 23–43.

[37] J. Maria Bermudo Mera, F. Turan, A. Karmakar, S. Sinha Roy, and I. Verbauwhede, "Compact domain-specific co-processor for accelerating module lattice-based kem," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[38] W. Wang, B. Jungk, J. Wälde, S. Deng, N. Gupta, J. Szefer, and R. Niederhagen, "Xmss and embedded systems - xmss hardware accelerators for risc-v," Cryptology ePrint Archive, Report 2018/1225, 2018, https://eprint.iacr.org/2018/1225.

[39] W. Wang, S. Tian, B. Jungk, N. Bindel, P. Longa, and J. Szefer, "Parameterized hardware accelerators for lattice-based cryptography and their application to the hw/sw co-design of qtesla," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 269–306, 06 2020.

[40] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on risc-v," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1148–1153.

[41] L. AKÇAY and S. B. Ö. YALÇIN, "Comparison of risc-v and transport triggered architectures for a postquantum cryptography application," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 29, no. 1, pp. 321–333, 2021.

[42] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, Sep. 2009.

[43] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, and etal, "Ntru algorithm specification and supporting document," March 2019. [Online]. Available: https://ntru.org/f/ntru-20190330.pdf

[44] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, and etal., "Crystals-kyber algorithm specification and supporting document," January 2021. [Online]. Available: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf

[45] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of Cryptology*, vol. 26, no. 1, pp. 80–101, 2013.

[46] A. Viti, A. Terzi, and L. Bertolaccini, "A practical overview on probability distributions," *Journal of thoracic disease*, vol. 7, no. 3, p. E7, 2015.

[47] A. Basso, J. M. Bermudo, J.-P. D'Anvers, and etal., "Saber: Mod-lwr based kem (round 3 submission)," October 2020. [Online]. Available: www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf

[48] E. Alkim, Roberto, Avanzi, J. Bos, L. Ducas, and etal., "Newhope algorithm specification and supporting document," July 2019. [Online]. Available: https://newhopecrypto.org/data/NewHope-2019-07-10.pdf

[49] P. Jääskeläinen, T. Viitanen, J. Takala, and H. Berg, *HW/SW Co-design Toolset for Customization of Exposed Datapath Processors*. Springer International Publishing, 2017, pp. 147–164. [Online]. Available: https://doi.org/10.1007/978-3-319-49679-5-8

[50] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

[51] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: constructing hardware in a scala embedded language," in *DAC Design Automation Conference 2012*. IEEE, 2012, pp. 1212–1221.

[52] F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, Nov 2019.

[53] J. Balkind, K. Lim, F. Gao, J. Tu, D. Wentzlaff, M. Schaffner, F. Zaruba, and L. Benini, "Openpiton+ ariane: The first open-source, smp linux-booting risc-v system scaling from one to many cores," in *Workshop on Computer Architecture Research with RISC-V (CARRV)*, 2019, pp. 1–6.

[54] B. Przybus, "Xilinx redefines power, performance, and design productivity with three new 28 nm fpga families: Virtex-7, kintex-7, and artix-7 devices," 2010.