IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 May 2024, accepted 13 May 2024, date of publication 15 May 2024, date of current version 22 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3401235

RESEARCH ARTICLE

Near Threshold Computation of Partitioned Ring Learning With Error (RLWE) Hardware Accelerator on Reconfigurable Architecture

PARESH BAIDYA[1,2], SWAGATA MANDAL[3], AND ROURAB PAUL[1,4]

[1]Department of Computer Science and Engineering, Siksha 'O' Anusandhan, Bhubaneswar 751030, India
[2]Department of Mathematics, National Institute of Technology Jamshedpur, Jamshedpur 831014, India
[3]Department of Electronics and Communication, Jalpaiguri Government Engineering College, Maulana Abul Kalam Azad University of Technology at West Bengal, Kolkata 700064, India
[4]Department of Computer Science, University of Pisa, 56126 Pisa, Italy

Corresponding author: Rourab Paul (rourab.paul@unipi.it)

ABSTRACT The Ring Learning With Error (RLWE) algorithm plays a crucial role in Post Quantum Cryptography (PQC) and Homomorphic Encryption (HE). The security of existing classical crypto algorithms is reduced in quantum computers. The adversaries can store all encrypted data and once quantum computers become available, they can potentially expose this encrypted data. Researchers and cryptographers are actively developing and exploring quantum-resistant cryptographic algorithms like RLWE to address this emerging threat. On the other hand, the HE allows operations on encrypted data, which is appropriate for getting services from third parties without revealing confidential plain-texts. Field Programmable Gate Array (FPGA) based Post-Quantum Cryptography (PQC) and Homomorphic Encryption (HE) hardware accelerators, such as Ring Learning With Error (RLWE), offer a much cost-effective alternative to processor-based platforms and Application-Specific Integrated Circuits (ASIC). However, FPGA based hardware accelerators still consume more power compared to ASIC based design. Near Threshold Computation (NTC) may be a convenient solution for FPGA based RLWE implementation. This paper implements a low-power RLWE hardware accelerator in an FPGA, operating at near-threshold biasing voltage. Instead of applying uniform biasing voltage to all 14 subcomponents of RLWE, this paper applies different near-threshold biasing voltages to the different subcomponents of proposed RLWE. Based on the longest design path or critical path of the 14 subcomponents of the proposed RLWE, the entire RLWE is partitioned into different clusters where each cluster is implemented in an FPGA partition. All the subcomponents placed in the same FPGA partition use the same biasing voltage V_{ccint}. The clusters that have higher critical paths use higher V_{ccint} to avoid timing failure. The clusters that have lower critical paths use lower biasing voltage V_{ccint}. The proposed RLWE uses a voltage calibration algorithm to calculate the biasing voltage V_{ccint} required for a certain amount of average critical path of a FPGA partition. Any timing error caused by NTC can be detected by the Razor flip-flop used in each subcomponent of RLWE. This voltage scaled, partitioned RLWE can save $\sim$6% and $\sim$11% power in *Vivado* and *VTR* platforms, respectively. Although low-power RLWE is the primary focus, the resource usage and throughput of the implemented RLWE hardware accelerator are competitive with existing literature.

INDEX TERMS FPGA, low power, post quantum cryptography, ring learning with error.

I. INTRODUCTION

Lattice-based cryptography is currently considered one of the most secure solutions compared to classical cryptography schemes. Classical schemes such as the discrete logarithm

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

(used in Elliptic Curve Cryptography), RSA and ECDSA are employed to secure modern Internet communications. These asymmetric key crypto-systems are based on the hardness of the prime factor and discrete logarithm. However, asymmetric key crypto-systems are no longer secure under quantum attacks. Even in brute force, Grover's quantum algorithm [1] reduces the searching space of symmetric key cryptography: Advanced Encryption Scheme (AES) from $O(2^n)$ to $O(2^{n/2})$, where n is the key size. Hence, the security of AES-256 is considered to be on par with AES-128 [2] when dealing with quantum computers. It is important to note that, as of now, no quantum computer with sufficient computational power has been developed. Nevertheless, companies like IBM and Google have made significant advancements in the development of highly computational quantum computers. Post-quantum crypto algorithms generally deal with non-quantum operations, but they strongly resists both classical and quantum attacks. Mainly, there are four post quantum cryptography schemes available: (i) Code based cryptography, (ii) Lattice based cryptography, (iii) Hash based cryptography and (iv) Multivariate quadratic cryptography. Isogeny-based cryptography is also a solution that resist post quantum attacks [3]. It utilizes the mathematical properties of isogenies between elliptic curves to devise cryptographic protocols. Isogenies are mappings between elliptic curves that preserve specific algebraic structures and their computation is challenging even for quantum computers. Among all these schemes lattice based cryptography (LBC) is computationally efficient [4]. In 2005, Regev [5] first introduced a lattice based cryptography (LBC) scheme named Learning With Error (LWE). Later LBC becomes more popular for its significant theoretical progress [5], [6]. The LBC became more suitable for real-life applications when it was implemented in software and hardware platforms by articles [7], [8], [9], [10], [11]. On the other hand, post-quantum cryptography finds its application in more advanced schemes such as fully homomorphic encryption [12], [13], which allows the operations of the encrypted data without revealing any information to the third party.

For hardware implementation of RLWE as PQC and HE, the designer can choose two platforms: FPGA or ASIC. In practice, FPGA based PQC and HE hardware accelerators like RLWE is much more cost-effective compared to ASIC. However, FPGA based hardware accelerator is less energy efficient compared to ASIC [14] because of its programmable switch. An experimental study [15] demonstrates that the power consumption of an 8-bit full adder in the Xilinx XC4003A, which also serves as a basic subcomponent for the RLWE hardware accelerator, is 100 times higher compared to the CMOS ASIC platform. The implementation of an 8-bit adder in FPGA platform consumed the power of 4.2 mW/MHz at 5 *Volt* and the same implementation in ASIC consumed 5.5 uW/MHz at 3.3 *Volt* [15]. The power consumption of FPGA based design is crucial, especially for battery powered embedded systems. NTC [16] may be an appropriate solution to reduce the power consumption of FPGA based RLWE implementation.

To the best of our knowledge, all existing RLWE implementations [7], [8], [17], [18] primarily emphasize speed and resource optimization. In contrast, this is the first reported work that tries to minimize the power consumption of RLWE using NTC.

A. RELATED WORK

Regev [5] introduced the first hardness proof of LWE cryptography in 2005. Later, various hardware and software implementations of LWE were proposed. A considerable amount of literature has implemented various subcomponents of the RLWE algorithm. The polynomial multiplication, polynomial division and Gaussian sampler are the most challenging subcomponents of RLWE. Howe et al. [17] proposed a hardware architecture for a standard lattice based cryptography (LWE) on a Spartan-6 FPGA platform in 2016. The primary contribution of this paper is the area optimization of the Gaussian Sampler by balancing area and performance. Pöppelmann et al. [19] reported a hardware implementation of the Gaussian sampler of RLWE accelerator. They implemented a Cumulative Distribution Table (CDT) based Gaussian sampler on reconfigurable hardware. The authors also compared CDT based Gaussian sampler and Bernoulli sampler. Roy et al. [10] implemented a Gaussian sampler for sampling from the discrete Gaussian distribution using the Knuth Yao algorithm. Pöppelmann and Güneysu [20] implemented a polynomial multiplier that stores all twiddle factors in a dedicated memory required for NTT computation. Göttert et al. [8] reduced the key size of the conventional LWE schemes and implemented compact and efficient LWE hardware. The article [8] compared LWE hardware and software implementations with Lindner and Peikert's design [21]. Article [8] introduced LWE-matrix and LWE-polynomial and compared these variants of LWE. Article [8] discarded the idea of LWE-matrix because of its large area consumption; they have only implemented RLWE-polynomial. Pöppelmann and Güneysu [7] implements an efficient hardware of RLWE on Virtex 6 FPGA, which can fit on a low-cost Spartan-6 FPGA. This hardware accelerator improved the work of [8]. Article [9] proposed the lightest RLWE encryption scheme by reducing the resource, compared with the high-speed implementation of [7].

Roy et al. [18] implemented a compact ring-LWE coprocessor on Virtex 6 FPGA where they optimized NTT polynomial multiplication and reduced the computation cost of the twiddle factors by avoiding the pre-computation overhead. A variation of the Native Title Research Unit (NTRU) Encrypt system with a quantum security reduction is used in article [22]. In 2019 and 2020, Mert et al. [12] and [13] introduced a software-hardware codesign for homomorphic encryption on Virtex 7 FPGA. This design involves connecting an FPGA and a CPU through a PCI bus, where the required data for homomorphic encryption is processed by the HE hardware accelerator running on

the FPGA. Its security is based on the standard model's LWE problem in polynomial rings. Sarker et al. [23] implemented a lightweight and compact fault detection scheme on various stages of the RLWE encryption hardware accelerator using FPGA. To compete with high speed data communication, hardware accelerators for RLWE on various FPGA platforms were proposed in articles by Thanawala et al. [24], Matteo et al. [25] and Xu et al. [26]. Thanawala et al. [24] explored the design space tradeoff of FFT based polynomial multiplier of RLWE. Matteo et al. [25] proposed a dedicated hardware for RLWE based on DMA to reduce processor overhead. Xu et al. [26] implemented a lightweight and pipelined polynomial multiplier based on Montgomery modular reduction.

Fig. 1 illustrates the timeline of RLWE research, spanning from its theoretical formulation to dedicated hardware implementation. All the above-mentioned RLWE implementations have mostly focussed on area versus speed issues. More speed causes more area and more area causes more power consumption. Without affecting the datapaths (area vs speed), the conventional low power solutions mostly deal with clock gating, power gating, dynamic voltage and frequency islands, retention power gating, save and restore power gating architectures. To the best of our knowledge. this is the first RLWE hardware accelerator based on the dynamic voltage island concept, where different voltage islands run with near-threshold voltages

B. PROPOSED RLWE

The existing literature on RLWE focuses on enhancing the throughput of the architecture to meet the demands of high-speed applications. Typically, this increased throughput is achieved through higher levels of parallelism, which, in turn, results in higher power consumption. Power consumption is a critical concern, particularly in embedded systems. However, existing literature has not explored the possibilities of NTC [16] in any PQC and HE implementations. NTC is an innovative design approach where transistors in a circuit operate at voltages close to their threshold voltage. The threshold voltage is the minimum voltage required for a transistor to switch on and conduct current. Operating at near-threshold voltages enables significant reductions in power consumption, as power consumption is roughly proportional to the square of the voltage. To the best of our knowledge, our implementation is the first attempt at NTC of RLWE on FPGA platforms with the aim of reducing power consumption. In our implementation, we designed a dedicated RLWE hardware, which has 14 subcomponents. The critical path of all these subcomponents is measured and based on the critical path, 4 cluster algorithms create groups with these 14 subcomponents. The FPGA is partitioned and different partitions have different biasing voltage V_{ccint}. The group having a higher value of critical path is placed in a partition which has higher V_{ccint} and the group having a lower value of critical path is placed in a partition which has lower V_{ccint}. It is to be noted that the partition is based on a critical path because the critical path is the path that has the longest delay. The relation between delay (D) and biasing voltage V_{ccint} can be stated by below equation: $D = C_{load}/V_{ccint}$) [30] Where C_{load} is the capacitance of the design path. Therefore, delay increases if the biasing voltage decreases. As the critical path has the longest delay, this path will be affected first and consequently, the other design paths with lesser delay than the critical path will start being affected. As a result, the partitioning of sub components of RLWE must be based on critical path.

The primary contribution of our paper is stated below:

- This paper designs and implements a low-power RLWE crypto algorithm in Artix-7 (xc7a100tcsg324-3, 28nm) commercial FPGA, along with three other academic FPGAs capable of operating in near-threshold biasing voltage. The implemented RLWE hardware accelerator has 14 subcomponents such as Random Number Generator (RND), Poly_div, NTT Controller, Polynomial Multiplier, Read only Memory (ROM), Scanner, Datapath Controller, Convolution Controller, Poly_add, reader, Distance, Row Column Controller and 2 dual-port Random Access Memory (RAM).
- The cluster algorithms create groups with RLWE subcomponents based on similar critical paths. Different groups are placed in different FPGA partitions. Different partitions run with different V_{ccint}s. The groups with a lower critical path are placed in a lower V_{ccint} partition and the groups with a higher critical path are placed in a higher V_{ccint} partition. The proposed voltage calibration algorithm calculates the biasing voltage V_{ccint} required for a certain amount of average critical path of a partition based on two parameters: the available FPGA technology ($V_{ccint_{max}}$, $V_{ccint_{min}}$) and the number of partitions. The timing errors caused by the reducing V_{ccint} are handled by the Razor Flipflops.
- The proposed FPGA based voltage scaled RLWE can save $\sim$ 11% and $\sim$ 6% dynamic power consumption in *VTR* and *Vivado* (Lower bounds of improvement due to the constraint of V_{ccint}) tool respectively. The throughput, resource and power consumption of the proposed RLWE is reasonably better compared to the existing RLWE.

The rest of the paper is organised as follows: Sec. II states the background of the RLWE algorithm followed by Sec. III which gives a preliminary idea about hardware implementation of RLWE. Next, in Sec. IV, the The tool flow of the proposed model is described. In Sec. V, the four cluster algorithm used to partition the FPGA is discussed and in Sec. VI-A and Sec. VII, the proposed algorithm to calculate biasing voltage and results are stated respectively. Finally, Sec. VIII. concludes the paper

II. THE RING-LWE SCHEMES

The Learning With Error (LWE) was proposed by O. Regev in [5] 2005. The LWE became popular because it is developed

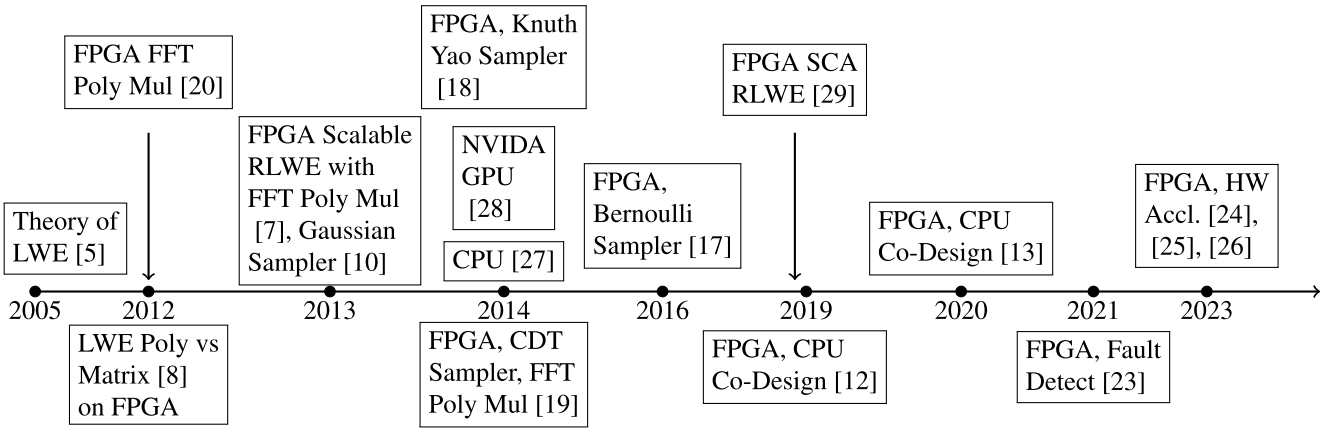

FIGURE 1. Time line of RLWE research.

in a secure lattice based cryptosystem which resists quantum attacks. Later LWE achieves computational efficiency and it reduces the key size by adopting polynomial Ring [31]. The LWE schemes perform in $R_q = Z_q[x]/ < f(x) >$, where $f(x)$=$x^n + 1$ is an irreducible polynomial of degree n, and $n = 2^k \geq 1$ and a prime q is chosen in such a way that $q \equiv 1 \ mod \ 2n$. There is an error distribution called discrete Gaussian distribution χ_σ with the standard deviation $\sigma > 1$ and mean 0. The LWE scheme is defined as follows:

- KeyGEN(a): choose r_1,$r_2 \in R_q$ sampled from χ_σ. Let $p = r_1 - a * r_2$. The public key is (a, p) and the private key is r_2.The polynomial $a \in R_q$ is chosen uniformly during the key generation.
- EnCrypt(a, p, m): choose error polynomials $e_1, e_2, e_3 \in R$ sampled from χ_σ. Then, compute the cipher text as a polynomial.

$$c_1 = a * e_1 + e_2 \tag{1}$$
$$c_2 = p * e_1 + e_3 + \tilde{m} \tag{2}$$

  where $\tilde{m}$ is $encode(m)$ and c_1 and c_2 are the encrypted message.
- DeCrypt(c_1, c_2, r_2): Decryption process requires to compute the equation below.

$$m' = c_1 * r_2 + c_2 \tag{3}$$

  Thereafter, to get the original message m, it needs to decode m'.

III. HARDWARE OF RLWE

The proposed *RLWE* has 14 components such as Random Number Generator (RND), Polynomial Divider (Poly_Div), NTT Controller, Polynomial Multiplier, Polynomial Subtraction (Sub_mod), Scanner, Datapath Controller, Convolution Controller, Polynomial Adder (Poly_add), reader, Distance, Row Column Controller and 2 dual-port Random Access Memory (RAM). The main components in this design are the memory file, Gaussian sampler, random number generator, NTT controller and datapath controller.

A. HARDWARE COMPONENTS

A brief description of all hardware components is stated below:

1) RANDOM NUMBER GENERATOR (RND)

This paper uses Trivium Random [32] number generator to generate error polynomial e_3. In our application, the secret key required for the RLWE is predefined. In real applications, the secret key should be generated randomly, which can be done by the Trivium Random Number Generator. Trivium requires one key and initialized vector to generate random numbers.

2) POLYNOMIAL DIVIDER (POLY_DIV)

The polynomial divider is required to divide polynomials. The resultant value of all the arithmetic operations stated in equ. 1, equ. 2 and equ. 3 need to bring under $R_q = Z_q[x]/x^n + 1$.

3) DUAL PORT RAM

This RAM stores the public key, message and Gaussian sample value required for RLW execution. The dual port memory allows parallel reading and writing operation of polynomial coefficients. The RLWE needs to handle a huge number of polynomial coefficients. The parallel read-write operation of these polynomial coefficients increases the RLWE throughput significantly.

4) DATAPATH CONTROLLER

The Datapath Controller has three sub-components such as:*Polynomial Multiplier*, *Polynomial Adder*(*poly_add*) and *PolynomialSubtraction*(*Sub_mod*) which are used to multiply, addition and subtraction of the polynomials.

(i). Polynomial Multiplier The polynomial multiplier (poly_mul) is required to multiply a and e_1 in equ. 1 and p and e_1 in in equ. 2. The implemented polynomial multiplier is based on Fast Fourier Transform (FFT) which has the lowest timing complexity among all available algorithms for polynomial multiplier.

(ii).Polynomial Adder (poly_add) The polynomial Adder (Poly_add) is required to add polynomials $a * e_1$ and e_2 in equ. 1 and $p * e_1$, e_3 and $\tilde{m}$ in equ. 2. In this implementation, we used the usual process of polynomial addition. This block reads coefficients of polynomials from RAM and adds them one by one.

(iii). Polynomial Subtraction (Sub_mod) The Polynomial Subtraction (Sub_mod) is required to subtract polynomials. This block is required when Polynomial Divider (poly_div) will perform. The working principle of polynomial subtraction is similar to polynomial addition. The negative number is complemented here.

5) CONVOLUTION CONTROLLER

The convolution controller decides whether addition or multiplication will be executed. This block is used to control the Datapath Controller.

6) READER

The reader block performs a loading operation to load the coefficients of polynomials and Gaussian noise coefficients. It also encodes the message bit before making it into cipher text.

7) NTT CONTROLLER

In the RLWE scheme, polynomial multiplication is one of the main operations of the encryption and decryption process. hardware implementation of coefficient-wise multiplication of two polynomial functions is computationally expensive. Therefore, we convert the coefficient-wise representation to point point-wise representation using the Number Theoretic Transformation (NTT) [33]. *NTTController* performs the NTT operation on the polynomial functions a, p, r_2, e_1, e_2, e_3 to generate $\tilde{a}, \tilde{p}, \tilde{r_2}, \tilde{e_1}, \tilde{e_2}, \tilde{e_3}$.

8) GAUSSIAN SAMPLER: SCANNER, DISTANCE, ROW COLUMN CONTROLLER

The Gaussian Sampler is based on Knuth Yao Sampler [34]. The Knuth Yao Sampler is a tree-based sampler that stores the binary expansion of the samples in a probability matrix. This probability matrix stores the probabilities of the samples. Roy et al. [10] implemented the first hardware of the Gaussian sampler using the Knuth Yao algorithm which used pre-computed table of the probability matrix. Knuth Yao algorithm uses a random walk model for the sampling process. This sampling process creates a discrete distribution generating (DDG) tree using the probability matrix. Our *Gaussian Sampler* has three subcomponents such as *scanner*, *distance*, *row column controller* modules which are dedicated to traversing the tree, scanning the bit from the ROM and calculating the distance in the tree of the visited node and intermediate node. The details of the Knuth Yao sampler and DDG tree are described in [34].

(i). Scanner Scanner block performs the scanning operation on the ROM block. It scans the bits from a ROM word. When all bits are read from a ROM word, it fetches the next ROM word to scan.

(ii). Row Column Controller Row ColumnController has two registers: an up counter named *column_length* and a down counter named *row_number*. The *column_length* stores the length of the different column lengths of the probability matrix. At the first step of the column scanning process, *row_number* is initialized by the *column_length*. If the *row_number* reaches zero, the column scanning process is completed.

(iii). Distance The distance block is required to construct the DDG tree of the probability matrix. For this purpose, a subtracter is used to control the random walk. When the subtracter value is < 0, then it indicates the completion of the sampling operation. After the completion of the sampling operation, the *row column controller* selects the current *row_number* as a sample
output.

B. WORK FLOW OF RLWE

As shown in Fig. 2, the coefficients of polynomials $a, p, r_1, r_2, e_1, e_2, e_3$ are stored in *Dual Port RAM*. The errors: e_1, e_2, e_3 are generated by *RND* and *Gaussian Sample* and stored in *Dual Port RAM*s. The *Convolution Controller* decides on polynomial arithmetic operations and activates the required polynomial arithmetic blocks: *poly_div*, *polynomial multiplier*, and *poly_add* to be executed. The *load* block, which is a part of the *reader wrapper*, reads all the polynomials from the *Dual Port RAM*s and sends them to different blocks for various polynomial arithmetic operations. During the polynomial multiplication in *polynomial multiplier* block, *NTT Controller* interfere the coefficient loading process of *loader* to convert the coefficient-wise representation to point point-wise representation using the NTT. The *Dual Port RAM*s are used to read and write polynomial coefficient simultaneously.

IV. TOOL FLOW

This paper uses two environments 1)*Python-Vivado* and 2) *Python-VTR*. In the *Python-Vivado* environment, *Vivado* generates the synthesis report, which includes the timing report of RLWE. The timing report consists of all timing-related information of RLWE including the critical path length of all 14 subcomponents of RLWE. This timing report is sent to the *Python* environment for three processes: (i) Cluster Algorithm, (ii) Voltage Calibration and (iii) Generate Constraint File. The cluster algorithm clusters the 14 subcomponents of RLWE in different groups. The details of this process are mentioned in Sec. V. After that, the cluster algorithm sends clusters of RLWE subcomponents to the voltage calibration algorithm mentioned in Sec. VI-A. The voltage calibration algorithm calculates suitable biasing voltages V_{ccint} for each cluster created by cluster algorithm. The voltage calibration algorithm may calculate the same V_{ccint} for different clusters. The clusters with the same V_{ccint} are merged into the same

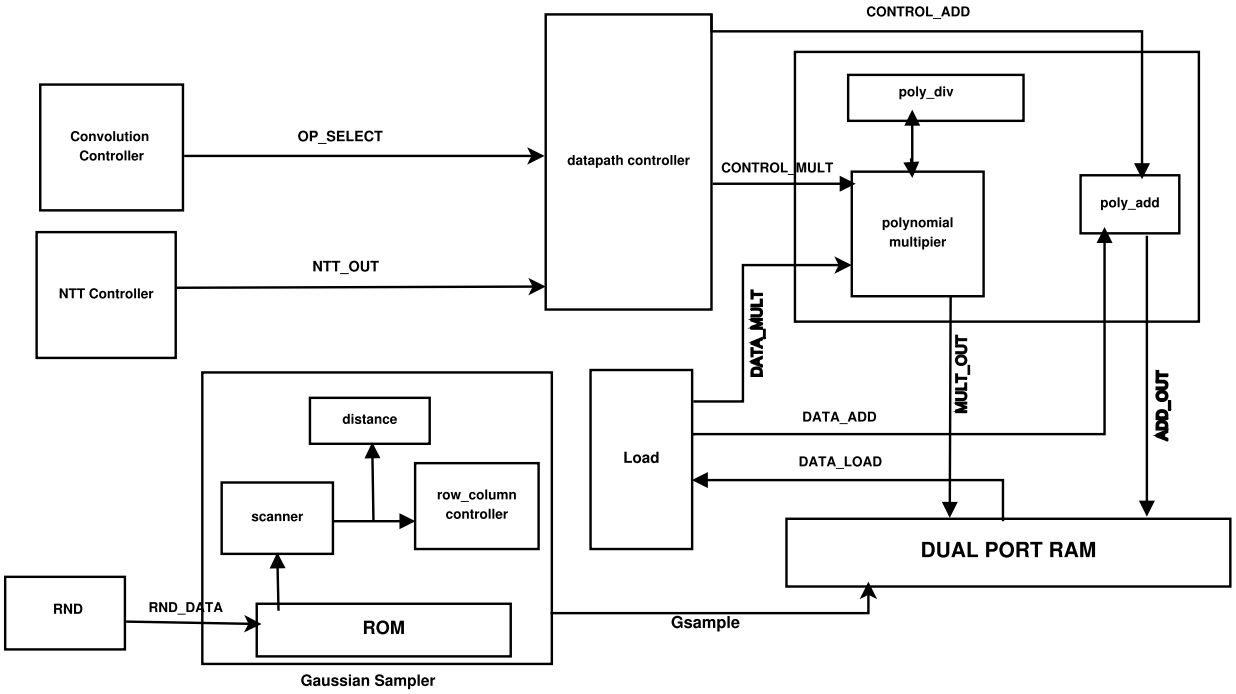

FIGURE 2. Architecture of RLWE.

FPGA partition. Later, the constraint file generation process in the *Python* environment generates the Xilinx Design Constraints (XDC) file, which mentions the coordinates of all 14 subcomponents of RLWE in different partitions of the FPGA floor. This XDC is then included in the implementation process of the *Vivado* environment. The *Vivado* used Artix 7, 28 nm FPGA. The *VTR* [35] also uses the same process as mentioned for *Vivado*. The Verilog-to-Routing or *VTR* generates Synopsys Design Constraint (SDC) file instead of XDC. The VTR tool is an open-source academic CAD toolflow for reconfigurable architecture that supports voltage scaling technology. The VTR tool includes three separate components: Odin II [36], ABC [37], and VPR [38]. The latest version of *VTR* provides libraries for three academic FPGAs [39]: 22nm, 45nm, and 130nm. These academic FPGAs provide many critical features for CAD research that are not available in commercial FPGAs. There is a possibility that after the partitioning of the RLWE subcomponents, the critical paths from the implementation process may differ from the critical paths from the synthesis process. Therefore, changes in the critical path may affect the entire RLWE design and may require re-clustering based on the new critical paths of RLWE subcomponents. Changes in parameters such as FPGA technology, clock frequency, etc., which affect the delays of design paths, can result in different delays in the implementation process and synthesis process. This re-clustering process increases tool execution time. For the particular RLWE hardware accelerator, critical paths from the synthesis process are similar to the critical paths from the implementation process. The *Vivado* tool flow used for the low power RLWE is reported in Fig. 3.

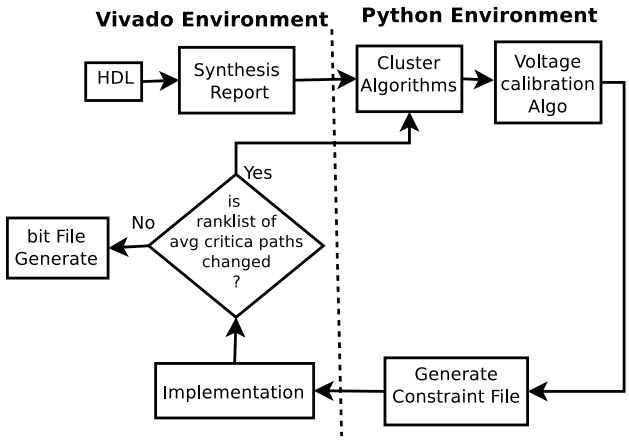

FIGURE 3. Vivado tool flow.

V. CLUSTER ALGORITHMS

This paper analyses four cluster algorithms to group the different sub-components of the RLWE hardware accelerator. All these four algorithms follow different sets of rules to find the similarity of the data in the data distribution. Different symbols used in Fig. 4, represent different clusters. Based on our design requirements, this paper implements four cluster algorithms as stated below:

A. K-MEANS CLUSTERING

K-Means cluster algorithm computes the distance between the data points and the randomly initialized centroids [40]. Thereafter, it creates a cluster based on the distance between the data points and the centroids. This iterative process

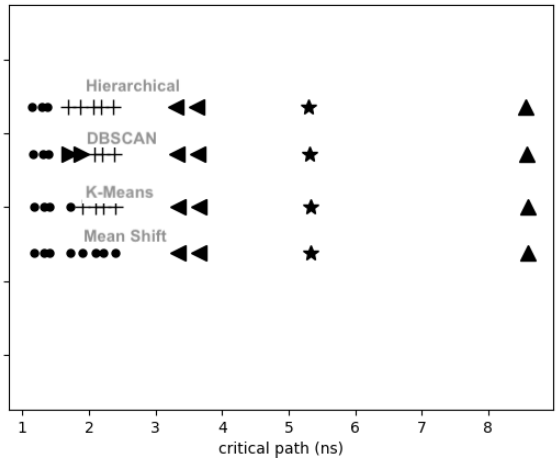

FIGURE 4. Cluster algorithms and partitions.

continues until the criteria of the functions converge to the local minimum. Euclidean distance is used to compute distances between the data. Its time complexity is $\mathcal{O}(Kn)$, where K is the number of clusters and n is the number of data. As shown in Table 1, the K-Means algorithm creates 5 clusters and each cluster is placed in each FPGA partition. The V_{ccint}s of $Partition-1$, $Partition-2$, $Partition-3$, $Partition-4$ and $Partition-5$ are 0.96, 1.00, 0.97, 0.98 and 0.94 respectively. The critical paths of different RLWE components are grouped into 5 clusters using the K-Means algorithm, as illustrated in Fig. 4. The optimal number of cluster in K Means algorithm is based on Silhouette Coefficient. It is to be noted that Silhouette refers to a method of interpretation and validation of consistency within clusters of data.

B. DBSCAN

The DBSCAN clustering algorithm operates under the assumption that clusters exist in high-density regions, while outliers are typically found in low-density regions. Unlike the K-Means algorithm, DBSCAN does not necessitate the pre-specification of the number of clusters.This algorithm finds the number of clusters [41] based on two parameters **epsilon** and **minpoint**. At the first step of DBSCAN choose an arbitrary point p. DBSCAN assumes a circle whose radius is **epsilon** and whose center is at p. All the data points that come under this circle are grouped in a cluster. If there are less number of points than the **minpoint**, p point is considered as noise. In a newly created cluster, if all points are marked as accessed, then the same process is used to deal with unvisited points and create a new cluster. This process will continue until all points are marked in cluster or noise. The main advantage of DBSCAN algorithm over the other algorithms is that it can identify the outlier point as a noise. The time complexity of this algorithm is $\mathcal{O}(n)$ for reasonable epsilon. As shown in Table 2, the DBSCAN algorithm creates 6 clusters. However, the voltage calibration algorithms stated in Sec. VI-A merge 6 clusters into 3 FPGA partitions, such as (i) **FPGA Partition-1**: $Cluster-1$, $Cluster-2$ and $Cluster-6$ placed into FPGA Partition-1 where V_{ccint}=0.96. (ii) **FPGA Partition-2**: contain $Cluster-3$ where V_{ccint}=0.98. (iii) **FPGA Partition-3**: $Cluster-4$ and $Cluster-5$ placed into FPGA Partition-3 where V_{ccint}=0.97. The critical paths of different RLWE components are grouped into 6 clusters using the DBSCAN algorithm, as illustrated in Fig. 4. Different shapes of the points represent different clusters.

C. MEAN-SHIFT CLUSTERING

The Mean Shift clustering algorithm, based on the concept of Kernel Density Estimation (KDE), assumes that the data points are sampled from a probability distribution. KDE estimates the distribution by applying a weight function named Kernel on each point of the data set. The mean shift algorithm works in such a way that the points climb uphill to the nearest peak on the KDE surface by iteratively shifting each point of the data set. It starts with a selected random point as the center of the kernel. It considers a circle which has a certain radius in the data set. The kernel is then moved toward a higher-density region by shifting the centroid towards the mean of the points within the said circle. The mean shift cluster does not need prior knowledge of the number of clusters. It needs only one parameter: bandwidth to determine the number of clusters. As shown in Table 3, the K-Means algorithm creates 4 clusters and each cluster is placed in each FPGA partition. The V_{ccint}s of $Partition-1$, $Partition-2$, $Partition-3$ and $Partition-4$ are 0.96, 1.00, 0.97 and 0.98 respectively. The critical paths of different RLWE components are grouped into 4 clusters using the Mean-Shift algorithm, as illustrated in Fig. 4.

D. HIERARCHICAL CLUSTERING

In the first step, the Hierarchical clustering algorithm considers each point as an individual cluster. Subsequently, it computes the distance matrix between pairs of clusters using a Euclidean distance measurement method. Then it merges two clusters which have the smallest distance. This process will repeat until all clusters are grouped into a single cluster. The dendrogram creates a binary tree for visualizing the hierarchy of clusters. The number of clusters can be determined from the dendrogram. This algorithm is computationally expensive for large datasets, having a time complexity of $\mathcal{O}(n^3)$ where n is the number of data-points. As shown in Table 4, the Hierarchical Cluster algorithm creates 6 clusters. However, the voltage calibration algorithms stated in Sec. VI-A merge 5 clusters into 3 FPGA partitions, such as (i)**FPGA Partition-1**: $Cluster-1$, $Cluster-5$ placed into FPGA Partition-1 where $V_{ccint}=0.96$.(ii) **FPGA Partition-2**: $Cluster-2$ and $Cluster-4$ placed into FPGA Partition-2 and fall into same voltage island where $V_{ccint}=0.98$. (iii) **FPGA Partition-3**: contain $Cluster-3$ where $V_{ccint}=0.97$. The critical paths of different RLWE components are grouped into 5 clusters using the Hierarchical algorithm, as illustrated in Fig. 4.

VI. VOLTAGE CALIBRATION AND ERROR CONTROL UNIT

The two primary challenges of NTC are (i)Voltage Callibration for FPGA partitions and (ii) Detecting timing errors.

A. VOLTAGE CALIBRATION ALGORITHM

The proposed voltage-scaled RLWE has 14 hardware components. These hardware components have different critical paths. The 4 cluster algorithms mentioned in Sec. V group these 14 components in different voltage islands. Each FPGA partition has a different biasing voltage V_{CCint}. The designer knows that if the critical path of a hardware component is longer, it requires a higher amount of V_{CCint} to avoid timing failures. However, the designer does not know the specific amount of V_{CCint} required for a particular length of critical path. This calibration of V_{CCint} depends on several design constraints, such as number of partition P, critical path C_i and critical region ($V_{ccintmax} - V_{ccintmin}$) as shown in Fig. 5. This Voltage Calibration Algorithm has two parts: (i)Average Critical Path of each Partition shown in Algorithm 1 (ii)Voltage Scaling of each Partition shown in Algorithm 2.

1) AVERAGE CRITICAL PATH OF EACH PARTITION

The calculation of the average critical path CP_k of each FPGA partition is described in Algorithm. 1. The summation of all critical paths of all subcomponents placed in the k^{th} partition is denoted by $A_{criticalk}$. Line 4 of Algorithm 1 calculates this $Acritical_k$. Thereafter, Line 6 calculates the average critical path CP_k of k^{th} partition by dividing $A_{criticalk}$ by the number of subcomponents $N(k)$ in k^{th} partition. Algorithm 1 calculates average critical path $CP[cp_1, cp_2, .., cp_p]$ of all partitions/cluster (calculated from cluster algorithms) of FPGA.

2) VOLTAGE SCALING OF EACH PARTITION

The algorithm 2 calculates the actual V_{ccint} for all P number of partitions. The V_{ccint} for k^{th} partition is denoted by VP_k. It calculates voltage $V_{r/c}$ for per unit critical path from $V_{ccintmax}$ and $V_{ccintmin}$ as shown in Line 5. After that, the biasing voltage VP_k of the k^{th} partition is calculated depends on the $V_{r/c}$, average critical path CP_k and $V_{ccintmin}$. The algorithm 2 may merge FPGA partitions clustered by cluster algorithms. As shown in Table 2 and Table 4, the algorithm 2 merges FPGA partitions created by DBSCAN and the Hierarchical cluster algorithm. The entire RLWE starts running with the calculated biasing voltages VP_k for its different FPGA partitions. It is to be noted that if VP_k falls within the critical region depicted in Fig. 5, timing errors may occur.

B. TIMING ERRORS

Despite of accurate V_{CCint} allocation, NTC on RLWE may cause timing errors. This paper designs a timing error control unit which uses razor flipflop in the datapath of each RLWE subcomponent. The razor or shadow flipflop in the FPGA

Algorithm 1 Average critical path of each partition

Require: $P\ \{k \leftarrow 1, 2, \ldots .P\}$
1: **for** $k \leftarrow 1$ to P **do**
2: $\quad A_{Critical_k} \leftarrow 0$
3: $\quad$ **for** $i \leftarrow 1$ to $N(k)$ **do**
4: $\quad\quad A_{Critical} \leftarrow A_{Critical} + C_i$
5: $\quad$ **end for**
6: $\quad CP_K \leftarrow A_{Critical}/N(k)$
7: **end for**
8: **return** $CP[cp_1, cp_2, .., cp_p]$

Algorithm 2 Voltage Scaling of Each Partition

Require: $V_{ccint_{max}}$, $V_{ccint_{min}}$, $CP[cp_1, cp_2, .., cp_p]$
1: $T_{Critical} \leftarrow 0$
2: **for** $k \leftarrow 1$ to P **do**
3: $\quad T_{Critical} \leftarrow T_{Critical} + CP_K$
4: **end for**
5: $V_{r/c} \leftarrow \frac{V_{ccint_{max}} - V_{ccint_{min}}}{T_{Critical}}$
6: **for** $k \leftarrow 1$ to P **do**
7: $\quad VP_k \leftarrow V_{ccint_{min}} + CP_K * V_{r/c}$
8: **end for**
9: **return** $VP[cp_1, cp_2, .., cp_p]$

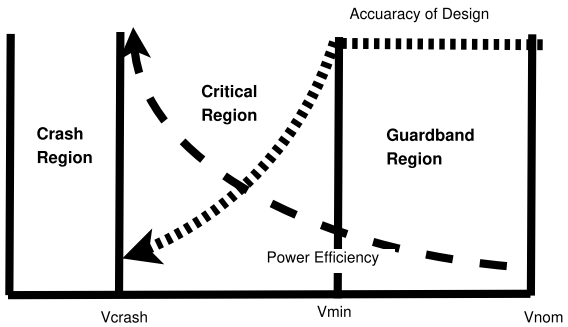

FIGURE 5. Voltage behaviour for V_{CCint}.

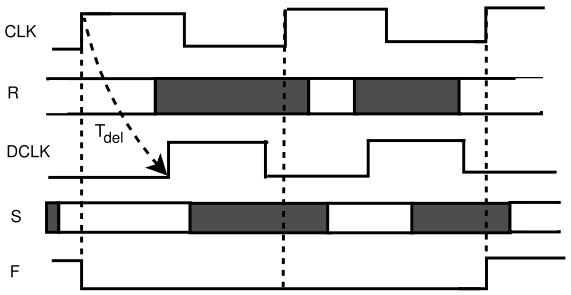

FIGURE 6. Razor timing diagram of fault detection.

platform is driven by a delayed clock [42]. We have assumed that one or more timing pathways leading from any of the source registers terminate at a circuit register R. The shadow registers S samples the same data as the main register R, but it does so on a delayed clock $DCLK$ that is delayed from CLK by T_{del}. Any data that enters the circuit after R samples but prior to S samples will result in a disparity between the two registers, which is identified by the error flag F. This

TABLE 1. FPGA partitions for K-means cluster.

Cluster	Hardwares	Critical Paths(ns)	$V_{ccint}(Volt)$	FPGA Partitions
1	Dual port ram:1 Dual port ram:2 Row Column Controller Distance reader Wrapper Poly_add	2.217 2.217 1.9 1.9 2.394 2.101	0.96	1
2	Datapath	8.603	1.00	2
3	Scanner ROM	3.332 3.654	0.97	3
4	Polynomial multiplier	5.327	0.98	4
5	NTT Controller Convolution Controller Poly_div rand	1.17 1.725 1.323 1.412	0.95	5

TABLE 2. FPGA partitions for DBSCAN cluster.

Clusters	Hardwares	Critical Paths(ns)	$V_{ccint}(Volt)$	FPGA Partitions
1	Dual port ram:1 Dual port ram:2 reader Wrapper Poly_add	2.217 2.217 2.394 2.101	0.96	1
2	Row Column Controller Distance Convolution Controller	1.9 1.9 1.725	0.96	1
3	Datapath	8.603	0.98	2
4	Scanner ROM	3.332 3.654	0.97	3
5	Polynomial multiplier	5.327	0.97	3
6	NTT Controller Poly_div rand	1.17 1.323 1.412	0.96	1

TABLE 3. FPGA partitions for mean-shift cluster.

Cluster	Hardwares	Critical Paths(ns)	$V_{ccint}(Volt)$	FPGA Partition
1	Dual port ram:1 Dual port ram:2 Row Column Controller Distance reader Wrapper Poly_add NTT Controller Convolution Controller Poly_div rand	2.217 2.217 1.9 1.9 2.394 2.101 1.17 1.725 1.323 1.412	0.96	1
2	Datapath	8.603	1.00	2
3	Scanner ROM	3.332 3.654	0.97	3
4	Polynomial multiplier	5.327	0.98	4

TABLE 4. FPGA partitions for hierarchical cluster.

Cluster	Hardwares	Critical Paths(ns)	$V_{ccint}(Volt)$	FPGA Partition
1	Dual port ram:1 Dual port ram:2 Row Column Controller Distance reader Wrapper Poly_add Convolution Controller	2.217 2.217 1.9 1.9 2.394 2.101 1.725	0.96	1
2	Datapath	8.603	1.00	4
3	Scanner ROM	3.332 3.654	0.97	3
4	Polynomial multiplier	5.327	0.98	2
5	NTT Controller Poly_div rand	1.17 1.323 1.412	0.96	1

TABLE 5. Dynamic power consumption of *Vivado* and *VTR*: voltage region*: $V_{ccint_{min}}$ = 0.95 volt to $V_{ccint_{max}}$ =1.05.

Our Design Under 25° Ambient Temparature & 100 MHZ Clock	Cluster algorithms	Partition No.	V_{ccint_i} (volt)	Dynamic Power (mw)			
				Vivado 28nm (mw)	VTR 22nm	VTR 45nm	VTR 130nm
Without Voltage scaling	NA	NA	1.00	22.45	46.44	59.09	190.38
Voltage scaled	K-Means	Partition-1 Partition-2 Partition-3 Partition-4 Partition-5	0.96 1.00 0.97 0.98 0.95	21.48	45.89	58.402	188.49
% of Reduction				4.34*	1.19	1.18	1.005
Without Voltage scaling	NA	NA	1.00	22.45	46.44	59.09	190.38
Voltage scaled	Mean-Shift	Partition-1 Partition-2 Partition-3 Partition-4	0.96 0.97 1.00 0.98	21.62	45.94	58.46	187.2
% of Reduction				3.73*	1.1	1.09	1.69
Without Voltage scaling	NA	NA	1.00	22.45	46.44	59.09	190.38
Voltage scaled	DBSCAN	Partition-1 Partition-2 Partition-3 Partition-4	0.96 0.96 0.97 0.98	21.16	45.87	58.38	188.5
% of Reduction				5.78*	1.24	1.21	0.99
Without Voltage scaling	NA	NA	1.00	22.45	46.44	59.09	190.38
Voltage scaled	Hierarchical	Partition-1 Partition-2 Partition-3 Partition-4 Partition-5	0.96 0.96 0.97 0.98 1.00	21.62	45.94	58.46	188.6
% of Reduction				3.73*	1.1	1.09	0.93

*Lower bounds of improvement due to the constraint of V_{ccint}

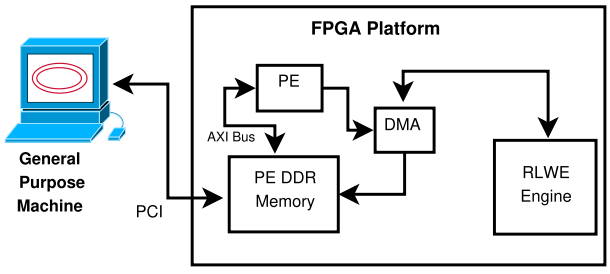

FIGURE 7. System architecture.

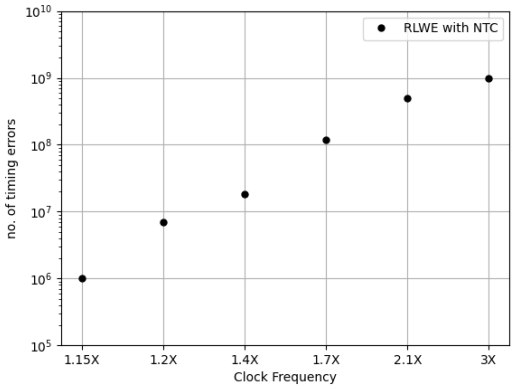

FIGURE 8. Normalized performance.

razor flipflop is put in the datapaths of the subcomponents. Razor increases the resource consumption, but it also has the ability to detect runtime timing errors in RLWE caused by near-threshold biassing voltage. Fig.6 displays the timing diagram for the razor. If Razor finds any error in particular partition of the FPGA the biasing voltage of that partition will be increased by one step. Fig. 5 showsh three voltage region in reconfigurable hardware. In crash region, the voltage below V_{crash} may cause timing failure in RLWE. In the critical region, the closer the voltage is to V_{crash}, the higher the power efficiency and the higher the probability of timing failure. The guard band region between the nominal voltage V_{nom} and the minimum voltage V_{min} has the least power efficiency, but the probability of timing failure due to biasing voltage is zero.

VII. RESULT AND IMPLEMENTATION

As stated in Sec. V, IV cluster algorithms are implemented by using *Scikit − learn* library of *Python*. The entire framework

TABLE 6. Dynamic power consumption for VTR, voltage region: $V_{ccint_{min}}$ = 0.5 volt to $V_{ccint_{max}}$ =1.3.

Our Design Under 25° Ambient Temperature & 100 MHZ Clock	Cluster algorithms	Partition No.	V_{ccint_i} (volt)	Dynamic Power (mw)		
				VTR 22nm	VTR 45nm	VTR 130nm
Without Voltage scaling	NA	NA	1.00	46.44	59.09	190.38
Voltage scaled	K-Means	Partition-1 Partition-2 Partition-3 Partition-4 Partition-5	0.58 0.86 0.62 0.7 0.54	41.41	52.69	179.55
% of Reduction				10.83	10.82	5.68
Without Voltage scaling	NA	NA	1.00	46.44	59.09	190.38
Voltage scaled	Mean-Shift	Partition-1 Partition-2 Partition-3 Partition-4	0.58 0.62 0.86 0.7	41.71	53.28	181.2
% of Reduction				10.19	9.83	4.82
Without Voltage scaling	NA	NA	1.00	46.44	59.09	190.38
Voltage scaled	DBSCAN	Partition-1 Partition-2 Partition-3 Partition-4	0.56 0.56 0.65 0.77	41.27	52.77	178.2
% of Reduction				11.15	10.71	6.38
Without Voltage scaling	NA	NA	1.00	46.44	59.09	190.38
Voltage scaled	Hierarchical	Partition-1 Partition-2 Partition-3 Partition-4 Partition-5	0.58 0.58 0.62 0.70 0.86	41.48	53	181.1
% of Reduction				10.69	10.32	4.89

uses two environments (i) Commercial *Vivado* tool with Artix-7 FPGA (ii) *VTR* tool for 22nm, 45nm and 130nm academic FPGAs. This paper proposes two different designs of RLWE encryption [18] scheme for the parameter set (n, q, s): (256, 7681, 11.32). The first approach implements an RLWE hardware accelerator without voltage scaling, whereas the second RLWE is designed with different voltage partitions. Presently, there is no FPGA which supports multiple V_{ccint}. Therefore, the power results are carried out separately where one partition is considered as an individual circuit. Voltage change strategies are implemented by Inter-Integrated Circuit (I2C) command from a PC-based Digital Power Designer software [43]. It is to be noted that the Artix-7 board hosts a power system based on the Texas Instruments (TI) UCD90120A power supply sequencer and monitor and the TPS84K and LMZ22000 family voltage regulators. As UCD90120A chips comes with Artix-7 PCB, this manuscript did not measure any additional overhead of run-time voltage calibration. Both of these two designs should only be studied for critical regions as shown in Fig. 5. However, the design and tool constraints, especially in *Vivado*, have forced us to study both the designs for the critical region as well as the guard band region. The V_{ccint}s of each partition are calculated by algorithms 1 and 2. The system design of the implemented RLWE is shown in Fig. 7 where general purpose machine is connected with FPGA based RLWE design through PCI communication.

A. VIVADO AND VTR: VOLTAGE REGION: $V_{CCINT_{MIN}}$ = 0.95 VOLT TO $V_{CCINT_{MAX}}$ =1.05 VOLT

The guard band voltage region of Artix-7 is $V_{ccint_{min}}$ = 0.95 volt to $V_{ccint_{max}}$ =1.05 volt. Though academic FPGAs on *VTR* platform support voltages of the critical region, for the sake of better comparative study we have implemented

TABLE 7. Comparison of our design with other RLWE encryption schemes.

Designs	Parameters (n,q,s)	Device	Slice Register/FF	Slice LUT	DSP	BRAM	Critical Path (ns)	Throughput (bps)
Our Design	(256,7681,11.32)	Artix-7	665/ 309	846	2	1	2.539	$\sim$ 7.2 $\times$ 10^6
Thomas et al. [7]	(256,7681,11.32)	V6LX75T @262 MHz	1506/ 3624	4549	1	12	-	$\sim$ 9.77 $\times$ 10^6
	(512,12289,12.18)	V6LX75T @251 MHz	1887/ 4760	5595	1	14	-	$\sim$ 9.33 $\times$ 10^6
Sujay et al. [18]	(256,7681,11.32)	V6LX75T @313 MHz	-/860	1349	1	2	-	$\sim$ 12.74 $\times$ 10^6
	(512,12289,12.18)	V6LX75T @278 MHz	-/953	1536	1	3	-	$\sim$ 10.69 $\times$ 10^6
Norman et al. [8]	(256,7681,11.32)	V6LX240T	143396/-	298016	-	-	-	$\sim$ 31.8 $\times$ 10^6
Timo et al. [29]-V1	(256,7681,11.32)	XC7A200 @208 MHz	1624/-	4365	1	5	-	$\sim$ 10.58 $\times$ 10^6
Timo et al. [29]-V2	(256,7681,11.32)	XC7A200 @250 MHz	2122/-	5616	6	8	-	$\sim$ 16.95 $\times$ 10^6
Howe et al. [17]	(256,4096,3.39)	S6LX45	1866/ 4804	6152	1	73	-	$\sim$ 0.32 $\times$ 10^6
Neil et al. [24] v1	(256,4096,-)	Artix-7	1336/ 2585	3572	45	19	-	-
Neil et al. [24] v2	(512,4096,-)	Artix-7	1509/ 2969	4086	50	21	-	-
Matteo et al. [25]	(1024,32,-)	Xilinx ZCU106	1440/ -	3168	42	20	-	-
Bian et al. [44]	(1024,32,-)	Xilinx ZCU106	1440/ -	3168	42	20	-	-

*Power data not available in articles [7], [8], [17], [18], [29], [24] and [25]

14 components of RLWE with voltage range of $V_{ccint_{min}} = 0.95$ volt to $V_{ccint_{max}} = 1.05$ volt for both *Vivado* and *VTR*. Table 5 shows *Vivado* 28nm Artix-7, *VTR* 22nm, *VTR* 45nm and *VTR* 130nm can save 4.34%, 1.19%, 1.18% and 1% power respectively.

B. VTR: VOLTAGE REGION: $V_{CCINT_{MIN}} = 0.5$ VOLT TO $V_{CCINT_{MAX}} = 1.3$ VOLT

Unlike *VTR*, *Vivado* does not allow any V_{ccint} below 0.95 and above 1.05. As shown in Table 6, different numbers of clusters generated from K-Means, Mean-Shift, DBScan and Hierarchical algorithms consider voltage region from $V_{ccint_{min}} = 0.5$ volt to $V_{ccint_{max}} = 1.3$ volt for 22nm, 45nm and 130nm in *VTR* flow.

As an example, the K-Means algorithm generates five partitions. Algorithm 1 calculates average critical path CP_k of the k^{th} partition where $A_{Critical}$ is the total critical path the k^{th} partition. Then algorithm 2 first computes the voltage per unit critical path $V_{r/c}$, after that required biasing voltage VP_k for k^{th} partition is calculated in line 7 of algorithm 2. The five V_{ccint}s calculated by algorithm 2 using K-Means are: $V_{ccint_1} = 0.96$ $V_{ccint_2} = 0.995 \approx 1$, $V_{ccint_3} = 0.965 \approx 0.97$ $V_{ccint_4} = 0.975 \approx 0.98$, $V_{ccint_5} = 0.955 \approx 0.95$. In Table 5, our voltage scaling method in the same voltage ranges, reduces the dynamic power consumption 3.73% to 5.78% in *Vivado* environment and reduces 0.93% to 1.24% in *VTR* environment. Table 6 shows adoption of voltage scaling technology reduces dynamic power consumption 10.19% to 11.15%, 9.83% to 10.82% and 4.82% to 6.38% for 22nm, 45nm and 130nm *VTR* flow respectively. Table 7 shows a comparison of resource and throughput of our RLWE hardware accelerator with existing literature. The existing literature [7], [8], [17], [18], [29] did not report the power consumption data in their manuscript. Though our paper does not claim any architectural contribution, Table 7 reports that resource usage and throughput are significantly better compared to existing RLWE. It is to be noted that the resource utilization and throughput of our RLWE may vary compared to existing RLWE implementations due to differences in FPGA technologies. Fig. 8 shows the number of timing errors increases with the clock frequency of the proposed RLWE design.

VIII. CONCLUSION

This paper presents a novel approach to implementing a low-power RLWE hardware accelerator across various FPGA platforms: Artix-7 28nm commercial FPGA, as well as 22nm, 45nm, and 130nm academic FPGAs using Vivado and VTR tools. The methodology involves clustering the subcomponents of RLWE based on their critical paths and partitioning the FPGA floor to allocate different clusters to separate FPGA partitions. By adjusting the biasing voltage based on the average critical path of each cluster, the proposed technique mitigates the possibilities of timing failures and optimizes power consumption. An innovative voltage calibration algorithm is introduced to determine the precise biasing voltage required for a given average critical path, considering factors such as available FPGA technology parameters and the number of partitions. The integration of Razor flip-flops in each subcomponent enables the detection of timing errors resulting from near-threshold computation. Results demonstrate significant power savings of approximately 6% and 11% in *Vivado* and VTR platforms, respectively, compared to traditional approaches. Furthermore, The resource usage and throughput of the implemented RLWE hardware accelerator are comparatively better than the existing literature.

IX. FUTURE SCOPE

In the future, to reduce the power consumption of high-processing algorithms, we will explore the possibilities of developing an automated tool flow for near-threshold computation of other high-processing designs. This tool will investigate two primary concerns: (i) It has been observed that the delay depends on the input patterns. Input patterns with similar delays can be grouped together. The sequence of inputs from a group may experience similar bit flips, potentially resulting in comparable delays. This process may anticipate timing errors in the design by employing heuristics to identify families of input sequences. (ii) The tuning of the clock frequency for each subcomponent may play a major role in controlling timing errors. However, adopting different clock domains may significantly impact the design's performance.

ACKNOWLEDGMENT

(Paresh Baidya and Rourab Paul contributed equally to this work.)

REFERENCES

[1] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying grover's algorithm to AES: Quantum resource estimates," in *Post-Quantum Cryptography*. Cham, Switzerland: Springer, 2016, pp. 29–43.

[2] X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher, "Quantum security analysis of AES," *IACR Trans. Symmetric Cryptol.*, pp. 55–93, Jun. 2019.

[3] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2011, pp. 19–34.

[4] M. E. Sabani, I. K. Savvas, D. Poulakis, G. Garani, and G. C. Makris, "Evaluation and comparison of lattice-based cryptosystems for a secure quantum computing era," *Electronics*, vol. 12, no. 12, p. 2643, Jun. 2023.

[5] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. 37th Annu. ACM Symp. Theory Comput.*, May 2005, pp. 84–93.

[6] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 147–191.

[7] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Proc. Int. Conf. Sel. Areas Cryptogr.* Berlin, Germany: Springer, 2013, pp. 68–85.

[8] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2012, pp. 512–529.

[9] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2796–2799.

[10] S. S. Roy, F. Vercauteren, and I. Verbauwhede, "High precision discrete Gaussian sampling on FPGAs," in *Proc. Int. Conf. Sel. Areas Cryptography*. Berlin, Germany: Springer, 2013, pp. 383–401.

[11] A. Aysu, C. Patterson, and P. Schaumont, "Low-cost and area-efficient FPGA implementations of lattice-based cryptography," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust (HOST)*, Jun. 2013, pp. 81–86.

[12] A. C. Mert, E. Öztürk, and E. Savas, "Design and implementation of a fast and scalable NTT-based polynomial multiplier architecture," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2019, pp. 253–260.

[13] A. C. Mert, E. Öztürk, and E. Savaş, "FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware," *Microprocessors Microsystems*, vol. 78, Oct. 2020, Art. no. 103219.

[14] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel, "A hybrid ASIC and FPGA architecture," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2002, pp. 187–194.

[15] E. Kusse and J. Rabaey, "Low-energy embedded FPGA structures," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 1998, pp. 155–160.

[16] S. Borkar, *Extreme Energy Efficiency by Near Threshold Voltage Operation*. Cham, Switzerland: Springer, 2016, pp. 3–18, doi: 10.1007/978-3-319-23389-5_1.

[17] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[18] S. Roy, F. Vercauteren, N. Mentens, D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2014, pp. 371–391.

[19] T. Pöppelmann, L. Ducas, and T. Güneysu, "Enhanced lattice-based signatures on reconfigurable hardware," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2014, pp. 353–370.

[20] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in *Proc. 2nd Int. Conf. Cryptology Inf. Security*, 2012, pp. 139–158.

[21] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Proc. Cryptographers' Track at RSA Conf.* Berlin, Germany: Springer, 2011, pp. 319–339.

[22] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2011, pp. 27–47.

[23] A. Sarker, M. M. Kermani, and R. Azarderakhsh, "Fault detection architectures for inverted binary ring-LWE construction benchmarked on FPGA," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 4, pp. 1403–1407, Apr. 2021.

[24] N. Thanawala, H. Nejatollahi, and N. Dutt, "Accelerating polynomial multiplication for RLWE using pipelined FFT," Cryptol. ePrint Arch., Univ. California, Irvine, Tech. Paper 2023/1815, 2023. [Online]. Available: https://eprint.iacr.org/2023/1815

[25] S. Di Matteo, M. L. Gerfo, and S. Saponara, "VLSI design and FPGA implementation of an NTT hardware accelerator for homomorphic SEAL-embedded library," *IEEE Access*, vol. 11, pp. 72498–72508, 2023.

[26] C. Xu, H. Yu, W. Xi, J. Zhu, C. Chen, and X. Jiang, "A polynomial multiplication accelerator for faster lattice cipher algorithm in security chip," *Electronics*, vol. 12, no. 4, p. 951, Feb. 2023.

[27] K. Rohloff and D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on NTRU," in *Financial Cryptography and Data Security*, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Berlin, Germany: Springer, 2014, pp. 221–234.

[28] W. Dai, Y. Doröz, and B. Sunar, "Accelerating NTRU based homomorphic encryption using GPUs," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2014, pp. 1–6.

[29] T. Zijlstra, K. Bigou, and A. Tisserand, "FPGA implementation and comparison of protections against SCAs for RLWE," in *Proc. Int. Conf. Cryptol. India*. Cham, Switzerland: Springer, 2019, pp. 535–555.

[30] R. Mukherjee and S. O. Memik, "Realizing low power FPGAs: A design partitioning algorithm for voltage scaling and a comparative evaluation of voltage scaling techniques for FPGAs," Northwestern Univ., Tech. Rep. TR-CMPE-05-0001, 2005.

[31] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Annu. Int. Conf. theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2010, pp. 1–23.

[32] T. Kaya, "Memristor and Trivium-based true random number generator," *Phys. A, Stat. Mech. Appl.*, vol. 542, p. 124071, Mar. 2020.

[33] A. Satriawan, I. Syafalni, R. Mareta, I. Anshori, W. Shalannanda, and A. Barra, "Conceptual review on number theoretic transform and comprehensive review on its implementations," *IEEE Access*, vol. 11, pp. 70288–70316, 2023.

[34] D. E. Knuth and A. C. Yao, "The complexity of nonuniform random number generation," in *Proc. Algorithms Complex., New Directions Recent Results Symp.*, Pittsburgh, PA, USA. New York, NY, USA: Academic, 1976, pp. 357–428.

[35] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. Eldafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. P. Walker, H. Zeng, P. Patros, J. Luu, K. B. Kent, and V. Betz, "VTR 8: High-performance CAD and customizable FPGA architecture modelling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, no. 2, pp. 1–55, Jun. 2020.

[36] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, "Odin II–An open-source verilog HDL synthesis tool for CAD research," in *Proc. 18th IEEE Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, May 2010, pp. 149–156.

[37] Berkley Logic Synthesis and Verification Group, "ABC: A system for sequential synthesis and verification," in *Proc. Revison*, 2018.

[38] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, K. Kent, and J. Rose, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 1–23, Dec. 2011.

[39] E. Vansteenkiste, A. Kaviani, and H. Fraisse, "Analyzing the divide between FPGA academic and commercial results," in *Proc. Int. Conf. Field Program. Technol. (FPT)*, Dec. 2015, pp. 96–103.

[40] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. Stanford, 2006.

[41] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[42] D. Ernst, N. Sung Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 22nd Digit. Avionics Syst. Conf.*, 2003, pp. 7–18.

[43] *User Guide: Ac701 Evaluation Board for the Artix-7 FPGA*, AMD-Xilinx, San Jose, CA, USA, 2013.

[44] S. Bian, D. E. S. Kundi, K. Hirozawa, W. Liu, and T. Sato, "APAS: Application-specific accelerators for RLWE-based homomorphic linear transformations," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4663–4678, 2021.

PARESH BAIDYA received the M.Sc. degree in mathematics from Indian Institute of Technology Madras (IITM), in 2015, and the M.Tech. degree in computer science and data processing from Indian Institute of Technology Kharagpur (IIT-KGP), in 2017. Currently, he is a Ph.D. student with the National Institute of Technology Jamshedpur, Jamshedpur, India, and an Assistant Professor at Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar, Odisha. His research interests include post-quantum cryptography and hardware cryptographic designs for homomorphic encryption.

SWAGATA MANDAL received the Bachelor of Technology (B.Tech.) degree in electronics and communication from West Bengal University of Technology (WBUT) in 2011, and the M.E. degree in electronics & telecommunication communication engineering from Bengal Engineering and Science University (Deemed University), now Indian Institute of Engineering, Science and Technology (IIEST), Shibpur, Howrah, India in 2013. He has done his Ph.D. research on Development of FPGA based Error Resilient Self-Triggered Readout Chain for CBM Experiment at Variable Energy Cyclotron Centre Kolkata, from 2013 to 2017. He was a Post-Doctoral Fellow at the School of Computer Science Engineering, Nanyang Technological University, Singapore, during 2017 to 2018. He is at present an Assistant Professor with Jalpaiguri Government Engineering College (Autonomous), West Bengal, India. He is the recipient of DGFS-Ph.D. fellowship for pursuing Ph.D. in the area of Engineering Science from the Department of Atomic Energy, Government of India, in 2013. He had received fellowships from different prestigious conferences like VLSID, ATS, ISVLSI, etc. for presenting the paper. His research interests are: In memory computation, VLSI design, embedded system design, fault tolerant system design, and hardware cryptography.

ROURAB PAUL received the B.Sc. and M.Sc. degrees in electronic science from the University of Calcutta, India, in 2008 and 2010, respectively. He is currently a Postdoctoral Fellow with the University of Pisa, Italy. Previously, he was an Assistant Professor with Siksha 'O' Anusandhan University, Bhubaneswar, India. He also held the position of a Postdoctoral Fellow with the Computer Science and Engineering Department, Indian Institute of Technology Kanpur, India. His Ph.D. research was completed with the University of Calcutta, from 2012 to 2017, with a focus on cryptography and related VLSI design. He also held a visiting research fellow position with the Electronics and Communication Engineering Department, National University of Singapore (NUS), from September 2013 to January 2014. From 2015 to 2016, he was with European Organization for Nuclear Research (CERN), Geneva, Switzerland, contributing to the Large Ion Collider Experiment (ALICE). Additionally, he was a Visiting Lecturer with the Acharya Prafulla Chandra College, Madhyamgram, Kolkata, and Techno India, Salt Lake, Kolkata. He has contributed as a Senior Academic Consultant with Convergent Solutions, Salt Lake, and participated in an internship program at I-CEE Design Technology, Kolkata, India.

Open Access funding provided by 'Università di Pisa' within the CRUI CARE Agreement