Exploring a Resource-Efficient NTT FPGA Accelerator for Fully Homomorphic Encryption

Valentino Guerrini, Giuseppe Sorrentino, Davide Conficconi
Dipartimento di Elettronica Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy
{10743695, giuseppe.sorrentino, davide.conficconi}@polimi.it

***Abstract*—CKKS encryption scheme stands as one of the most valuable solutions for Fully Homomorphic Encryption (FHE), enabling privacy-preserving computation on encrypted data, at the cost of high computational bottlenecks. In such a scheme, the Number Theoretic Transform (NTT) consumes most of the computational resources due to irregular memory access patterns. Thus, literature accelerates this step on specific hardware devices, such as FPGAs, often exhausting device resources while gaining performance and energy efficiency improvements. However, this prevents further utilization of the FPGA to accelerate other compute-intensive stages. As an alternative, we perform the HW/SW co-design of resource-efficient solutions by integrating them into well-known software libraries implementing CKKS encryption scheme. In particular, we deploy on a Kria KV260 SoC a resource-efficient NTT accelerator with state-of-the-art security parameters ($logN \in 12, ..., 16$ and $logQ \in [32, 64]$), and integrate it into the full-RNS HEANN library – the reference implementation for CKKS scheme. By doing so, we obtain up to $4.47\times$ and $3.63\times$ speedup in the encoding and encryption steps, respectively, while minimizing hardware consumption. These results show the end-to-end improvements achievable without fully utilizing the FPGA resources, leaving headroom for accelerating additional stages of the encryption pipeline.**

***Index Terms*—Fully Homomorphic Encryption, CKKS, Number Theoretic Transform, FPGA acceleration, resource-efficient design, HEAAN integration, embedded SoCs.**

I. Introduction

Fully Homomorphic Encryption (FHE) enables a client to outsource computations to an untrusted server (e.g., the cloud) while preserving the privacy of inputs, intermediate values, and outputs [1]–[3]. This paradigm is essential in scenarios where sensitive data cannot be disclosed, even at the cost of significant computational overhead. A representative example is distributed learning, where partially trained models are exchanged among peers and must therefore be encrypted to prevent information leakage.

This encryption step is particularly critical in post-quantum schemes such as CKKS [3], which operates over $R_Q = \mathbb{Z}_Q[x]/(x^N + 1)$, with $N = 2^n$ and Q composed of multiple 50–61 bit prime moduli. In this setting, polynomial multiplications represent the main computational bottleneck, introducing delays that can hinder the practical adoption of FHE schemes. To mitigate this issue, the literature adopts the Number Theoretic Transform (NTT), reducing polynomial multiplication complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ by transforming convolutions into point-wise multiplications in $\mathbb{Z}_Q$. However, FHE schemes invoke the NTT repeatedly during encryption, causing it to account for up to 54% of the total execution time even in highly optimized libraries such as Microsoft SEAL [4], [5].

Given the NTT's pivotal role, literature proposes multiple optimized solutions on both general and specialized hardware [4], [6]–[19]. However, such solutions typically treat the NTT as an isolated kernel, aggressively exploiting hardware resources to maximize performance. While effective in isolation, this limits the integration into complete FHE pipelines and leaves other computationally intensive stages unaccelerated. Moreover, to mitigate the numerical error inherent to approximate schemes such as CKKS, many works target large FHE parameter sets, resulting in accelerators tailored exclusively to HPC platforms. In contrast, embedded solutions often rely on smaller parameter sets to meet resource constraints, at the cost of increased information loss during encrypted computations.

In contrast to performance-driven designs, we explore resource-efficient NTT accelerators that support state-of-the-art FHE parameter sets typically adopted for HPC platforms. We deploy such a design on a Kria KV260 SoC, **trading peak performance for improved resource efficiency** while preserving support for large parameters. As our **main contribution**, we integrate the accelerator into Full-RNS HEANN [20], the reference CKKS implementation, and evaluate whether fully saturating FPGA resources is necessary to achieve meaningful performance gains.

II. Background

Homomorphic encryption (HE) enables computation directly on encrypted data. Given a ciphertext $c = \mathsf{Enc}(m)$ of a message m, an evaluator can compute $\mathsf{Enc}(f(m))$ without access to the secret key, enabling privacy-preserving outsourcing. Modern schemes rely on hard lattice problems such as Ring Learning With Errors (RLWE) [21] and operate over polynomial rings $R_Q = \mathbb{Z}_Q[x]/(x^N + 1)$, where N is a power of two and Q is a large modulus. Ciphertexts are vectors of polynomials in R_Q, and homomorphic operations reduce to polynomial arithmetic modulo $(x^N + 1, Q)$.

A. The CKKS Scheme

CKKS [22] is widely used because it supports approximate arithmetic over complex numbers, which suits applications such as machine learning inference and data analytics. However, operating directly modulo a large Q requires expensive multi-precision arithmetic.

TABLE I
EXECUTION TIME OF THE NTT KERNEL AND HEAAN INTEGRATION ON THE KRIA KV260 (ZU3EG). RESULTS ARE AVERAGED OVER 10 RUNS.

Parameters		NTT [ms]		Encode [ms]			Encrypt [ms]		
N	$\log Q$	Kernel	E2E	SW	HW	SU	SW	HW	SU
2^{12}	1×61-bit	0.215	0.217	32.31	7.51	4.30×	109.3	32.36	3.38×
2^{13}	2×61-bit	0.463	0.466	70.21	16.30	4.31×	237.8	69.09	3.44×
2^{14}	4×61-bit	1.005	1.014	149.1	34.34	4.34×	503.9	145.2	3.47×
2^{15}	7×61-bit	2.149	2.165	318.7	73.57	4.33×	1078.7	301.9	3.57×
2^{16}	10×61-bit	4.576	4.598	681.4	152.6	4.47×	2282.0	629.3	3.63×

Cheon *et al.* proposed the Residue Number System (RNS) variant of CKKS [3], where $Q = q_0 q_1 \cdots q_{L-1}$, and ciphertexts are represented as residue polynomials modulo the q_i. This enables word-size modular arithmetic in each residue channel, avoiding large-integer operations and improving practicality.

III. INTEGRATION AND EVALUATION

Our **primary objective** is to accelerate CKKS client-side routines by *integrating* a resource-efficient NTT into an existing FHE software stack, rather than optimizing an isolated kernel. In particular, we target Full-RNS HEAAN [20], the reference implementation of CKKS [3], where forward NTTs are executed repeatedly during `encode()` and `encrypt()` (once per RNS residue polynomial). Figure 1 depicts the hardware/software co-design, showcasing the NTT modules integrated into the software library. Notably, we completely hide the burden of hardware management by allowing the accelerated module to be used with the same API offered by the software HEANN library. After a one-time initialization, required to load the bitstream and allocate the device buffers, any NTT call issued inside `encode()` and `encrypt()` is automatically redirected to the FPGA through XRT.

Experimental setup - We evaluate on a Xilinx Kria KV260 (ZU3EG) using Vitis HLS and Vivado 2023.2. All kernels run at 100 MHz. We evaluate FHE-relevant configurations with $\log N \in \{12, \ldots, 16\}$ and primes up to 64 bits [23]. Table II shows that the design versions integrated in HEANN.

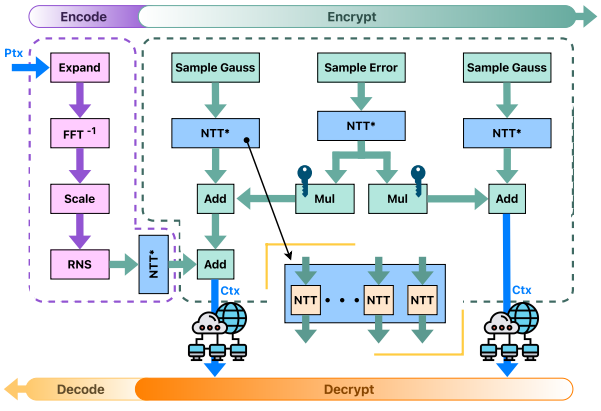


Fig. 1. Encoding and encryption flow of CKKS in HEAAN. Blue boxes (`NTT*`) mark forward NTT calls, executed once per RNS residue and transparently offloaded to the FPGA accelerator.

Notably, each design uses only a fraction of KV260 resources across all $\log N$. Importantly, LUT/FF/DSP usage remains well below device limits and no URAM is used, leaving substantial headroom to integrate and accelerate additional CKKS operators on the same FPGA.

HEANN Integration Table I reports kernel latency and end-to-end improvements in HEAAN. Quantitatively, the integration delivers consistent application-level speedups: `encode()` improves by 4.30–4.47× and `encrypt()` by 3.38–3.63× across supported parameters. At $N = 2^{16}$, `encode()` reduces from 681ms to 153ms and `encrypt()` from 2.28s to 0.63s. We also measure the energy efficiency, measured in joules, using power measurements collected through on-board `hwmon` sensors. From this analysis, our solution attains up to 2.8× energy efficiency improvements against the software reference for $N = 2^{16}$. These results demonstrate that meaningful end-to-end gains can be achieved *without* utilizing the entire FPGA, supporting our design choice of resource efficiency to enable future full-pipeline acceleration.

TABLE II
RESOURCE UTILIZATION ON KV260 ($\log q = 64$, 100MHz).
PERCENTAGES ARE W.R.T. DEVICE CAPACITIES.

$\log N$	kLUT	kFF	BRAM	DSP
12	18.6 (15.8%)	16.2 (6.9%)	9.5 (6.6 %)	536 (42%)
13	18.1 (15.4%)	17.0 (7.2%)	18 (12.5 %)	536 (42%)
14	19.2 (16.3%)	17.5 (7.4%)	34 (23.6 %)	594 (47%)
15	21.2 (18.1%)	18.0 (7.7%)	66 (45.8 %)	594 (47%)
16	24.2 (20.6%)	18.5 (7.8%)	130 (90.3 %)	652 (52%)
Average power consumption: 3.64 W				

IV. CONCLUSION & ACKNOWLEDGEMENT

This research explores the integration of a hardware/software co-designed accelerator into the client-side encryption pipeline of CKKS. In particular, we integrate an NTT accelerator into the Full-RNS HEAAN, delivering consistent end-to-end speedups of up to 4.47× for encoding and 3.63× for encryption on an embedded SoC FPGA. These gains require only a fraction of the available device resources, leaving substantial headroom to accelerate additional CKKS operators.

Acknowledgement– This work has financial support from ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU and in part by AMD under the Fund for Academic Research (FAR). The authors are grateful for the anonymous reviewers' feedback.

REFERENCES

[1] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," Cryptology ePrint Archive, Paper 2011/277, 2011. [Online]. Available: https://eprint.iacr.org/2011/277

[2] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint Archive, Paper 2012/144, 2012. [Online]. Available: https://eprint.iacr.org/2012/144

[3] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," Cryptology ePrint Archive, Paper 2018/931, 2018. [Online]. Available: https://eprint.iacr.org/2018/931

[4] D. Kumarathunga, Q. Hu, and Z. Fang, "Autontt: Automatic architecture design and exploration for number theoretic transform acceleration on fpgas," in *2025 IEEE 33rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2025, pp. 1–9.

[5] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[6] 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. 72 498–72 508, 2023.

[7] P. Duong-Ngoc, S. Kwon, D. Yoo, and H. Lee, "Area-efficient number theoretic transform architecture for homomorphic encryption," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 3, pp. 1270–1283, 2023.

[8] Z. Guan, Y. Zhu, Y. Huang, L. Lei, X. Wang, H. Jia, Y. Chen, B. Zhang, J. Dong, and S. Bian, "Esc-ntt: An elastic, seamless and compact architecture for multi-parameter ntt acceleration," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[9] J. Huang, C. Kuo, S. Liu, and T. Su, "An area-efficient and configurable number theoretic transform accelerator for homomorphic encryption," *Electronics*, vol. 13, no. 17, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/17/3382

[10] K. Kawamura, M. Yanagisawa, and N. Togawa, "A loop structure optimization targeting high-level synthesis of fast number theoretic transform," in *2018 19th International Symposium on Quality Electronic Design (ISQED)*, 2018, pp. 106–111.

[11] S. Kim, K. Lee, W. Cho, Y. Nam, J. H. Cheon, and R. A. Rutenbar, "Hardware architecture of a number theoretic transform for a bootstrappable rns-based homomorphic encryption scheme," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2020, pp. 56–64.

[12] E. Kocer, S. Kirbiyik, T. Tosun, E. Alaybeyoglu, and E. Savas, "Io-optimized design-time configurable negacyclic seven-step ntt architecture for fhe applications," in *Proceedings of the Great Lakes Symposium on VLSI 2025*, ser. GLSVLSI '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 14–21. [Online]. Available: https://doi.org/10.1145/3716368.3735514

[13] Z. Li, J. Ren, G. Du, Z. Tu, X. Wang, Y. Yin, and Y. Ouyang, "An area-efficient large integer ntt-multiplier using discrete twiddle factor approach," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 751–755, 2023.

[14] R. Mareta, A. Satriawan, P. N. Duong, and H. Lee, "A bootstrapping-capable configurable ntt architecture for fully homomorphic encryption," *IEEE Access*, vol. 12, pp. 52 911–52 921, 2024.

[15] X. Meng, Z. Jiang, and Y. Lyu, "Hf-ntt: Hazard-free dataflow accelerator for number theoretic transform," 2024. [Online]. Available: https://arxiv.org/abs/2410.04805

[16] A. C. Mert, E. Karabulut, E. Öztürk, E. Savaş, and A. Aysu, "An extensive study of flexible design methods for the number theoretic transform," *IEEE Transactions on Computers*, vol. 71, no. 11, pp. 2829–2843, 2022.

[17] F. Hirner, A. C. Mert, and S. S. Roy, "Proteus: A pipelined NTT architecture generator," Cryptology ePrint Archive, Paper 2023/267, 2023. [Online]. Available: https://eprint.iacr.org/2023/267

[18] C. Wang and M. Gao, "Sam: A scalable accelerator for number theoretic transform using multi-dimensional decomposition," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[19] Y. Zhang, S. R. Sathi, Z. Kou, S. Sinha, and W. Zhang, "Tensor-product-based accelerator for area-efficient and scalable number theoretic transform," in *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2023, pp. 174–183.

[20] "Fullrns-heaan," https://github.com/KyoohyungHan/FullRNS-HEAAN, Oct. 2018, cryptoLab inc.

[21] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," Cryptology ePrint Archive, Paper 2012/230, 2012. [Online]. Available: https://eprint.iacr.org/2012/230

[22] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," Cryptology ePrint Archive, Paper 2019/524, 2019. [Online]. Available: https://eprint.iacr.org/2019/524

[23] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption standard," Cryptology ePrint Archive, Paper 2019/939, 2019. [Online]. Available: https://eprint.iacr.org/2019/939