REED: Chiplet-based Accelerator for Fully Homomorphic Encryption

Aikata Aikata[1], Ahmet Can Mert[1], Sunmin Kwon[2], Maxim Deryabin[2], Sujoy Sinha Roy[1]
[1]*IAIK, Graz University of Technology,* [2]*Samsung Advanced Institute of Technology, Suwon, Republic of Korea*

{*aikata,ahmet.mert,sujoy.sinharoy*}*@iaik.tugraz.at,*{*sunmin7.kwon,max.deriabin*}*@samsung.com*

***Abstract*—Fully Homomorphic Encryption (FHE) enables privacy-preserving computation and has many applications. However, its practical implementation faces massive computation and memory overheads. To address this bottleneck, several Application-Specific Integrated Circuit (ASIC) FHE accelerators have been proposed. All these prior works put every component needed for FHE onto one chip (monolithic), hence offering high performance. However, they suffer from practical problems associated with large-scale chip design, such as inflexibility, low yield, and high manufacturing cost.**

In this paper, we present the *first-of-its-kind* multi-chiplet-based FHE accelerator 'REED' for overcoming the limitations of prior monolithic designs. To utilize the advantages of multi-chiplet structures while matching the performance of larger monolithic systems, we propose and implement several novel strategies in the context of FHE. These include a scalable chiplet design approach, an effective framework for workload distribution, a custom inter-chiplet communication strategy, and advanced pipelined Number Theoretic Transform and automorphism design to enhance performance.

Experimental results demonstrate that REED 2.5D microprocessor consumes 96.7 mm^2 chip area, 49.4 W average power in 7nm technology. It could achieve a remarkable speedup of up to 2,991$\times$ compared to a CPU (24-core 2$\times$Intel X5690) and offer 1.9$\times$ better performance, along with a 50% reduction in development costs when compared to state-of-the-art ASIC FHE accelerators. Furthermore, our work presents the *first* instance of benchmarking an encrypted deep neural network (DNN) training. Overall, the REED architecture design offers a highly effective solution for accelerating FHE, thereby significantly advancing the practicality and deployability of FHE in real-world applications.

***Index Terms*—Fully Homomorphic Encryption, Hardware Accelerator, Chiplets, CKKS, privacy-preserving DNN training**

1. Introduction

Data breaches can put millions of private accounts at risk because data is often stored or processed without encryption, making it vulnerable to attacks [28], [42], [50]. Fully Homomorphic Encryption (FHE) is a solution that allows secure, private computations, communications, and storage. It enables servers to compute on homomorphically encrypted data and return encrypted outputs. FHE has a wide range of applications, including cloud computing [37], [45], data processing [7], and machine learning [55]. The concept of FHE was introduced in 1978 by Rivest, Adleman, and Dertouzos [62], and the first FHE scheme was constructed in 2009 by Gentry [24]. Since then, many FHE schemes have emerged- BGV [9], FV [19], CGGI [16], and CKKS [14], [15], [38]. These schemes allow computations to be outsourced without the need to trust the service provider, providing a functional and dependable privacy layer.

Despite significant progress in the mathematical aspects of FHE, state-of-the-art FHE schemes typically introduce 10,000$\times$ to 100,000$\times$ slowdown [32] compared to plaintext calculations. This overhead can be attributed to plaintext expanding into large polynomials when encrypted using an FHE scheme. Subsequently, simple operations, like plaintext multiplication, translate into complex polynomial operations. FHE's massive computation and data overhead hinders its deployment in real-life applications. To bridge this performance gap, researchers have proposed acceleration techniques on various platforms, including GPU, FPGA, and ASIC [5], [6], [20], [21], [23], [31], [34]–[36], [47], [53], [59], [61], [64]–[66], [71], [73], [77]–[79]. Software implementations offer flexibility but poor performance. Attempts have been made to provide GPU [5], [31] and FPGA-based solutions [47], [61], [71]. However, the performance gap is still 2-3 orders compared to plain computation.

Currently, the fastest hardware acceleration results for FHE have been reported using ASIC modeling [21], [23], [34]–[36], [66]. The works propose utilizing large chip architecture designs with all FHE building blocks onto a single chip to maximize performance, hence monolithic. While simulations of these architectures show that they can achieve high performance for FHE workloads, the limitations of the current manufacturing capabilities, such as inflexibility, low yield, and higher manufacturing costs [25], hinder their real-world deployment. For instance, the large architectures [34]–[36] with area-consumption of approximately 400mm^2, result in a manufacturing yield of only 67% [44], chip fabrication cost of over 25M$ [52], and long time-to-market (>3 years).

Additionally, several of these proposals overlook the crucial need for communication-computation parallelism as the off-chip to on-chip communication is slower than the

chip's computation speed. Our analysis shows that this feature is important in an FHE accelerator for achieving good performance when running complex tasks like neural network training.

In summary, the key problem we identify is the large and complex monolithic FHE architectures proposed in prior works, as they are difficult to realize in practice due to expensive manufacturing costs, low yields, and long time-to-market. As a result, achieving the desired acceleration is challenging, necessitating the exploration of alternative approaches. One such approach is chiplet-based architecture design, where a large chip is realized by utilizing multiple smaller chiplets instead of one large monolithic chip. Chiplets are modular building blocks that are combined to create more complex integrated circuits, such as CPUs, GPUs, Systems-on-Chip (SoCs), or System-in-Package (SiPs).

The transition to chiplet integrated systems represents both the present and future of architectural designs [25], [26], [44], [46], [81], [82]. In the DATE2024 keynote talk [63], the speaker remarks how chiplet-based designs help 'push the performance boundaries, with maximum efficiency, while managing costs associated with manufacturing and yield'. Although chiplet-based architectures enjoy the aforementioned advantages, they also face a trade-off between performance and yield. Multiple smaller chiplets offer high yields and reduced manufacturing costs but, at the same time, suffer performance overhead due to slow chiplet-to-chiplet communication.

In this work, we present- REED a multi-chiplet architecture designed for FHE acceleration. Thus, we first propose a scalable design methodology for one chiplet. This ensures full utilization of chiplets for varying amounts of available off-chip data bandwidths. After finalizing an efficient design of one chiplet, we move to a data and task distribution study for multiple chiplets in the context of CKKS [15] routines. Towards this, we contribute novel strategies that offer long-term computation and communication parallelism. Finally, we synthesize the proposed design methodology for ASIC and report application benchmarks.

Contributions

To the extent of our knowledge, this is the *first chiplet-based architecture for accelerating FHE*. Throughout this work, we have followed Occam's razor, seeking the simplest solutions for the best results. We unfold our major contributions as follows:

- **Chiplet-based FHE accelerator:** We present a novel and cost-effective chiplet-based FHE implementation approach, which is inherently scalable. The chiplets are homogeneous (i.e., identical), which reduces testing and integration costs. REED with 2.5D packaging surpasses state-of-the-art work SHARP_{64} [34] with 1.9× better performance and 2× less development cost.
- **Workload division strategy:** The first step to realizing a multi-chiplet architecture is to develop an efficient disintegration strategy that helps us divide the workloads among multiple chiplets and reduces memory consumption. Hence, we propose an interleaved data and workload distribution technique for all FHE routines.
- **Efficient C2C communication:** Chiplet-based architectures suffer from slow C2C (chiplet-to-chiplet) communication bottleneck. We address this by proposing the *first non-blocking ring-based inter-chiplet communication* strategy tailored to FHE. This mitigates data exchange overhead during the KeySwitch macro-routine, accelerating Bootstrapping (the most expensive FHE routine).
- **Scalable design:** To attain scalability by design, we propose a configuration-based design methodology such that the memory read/write and computational throughput are the same. By changing the configuration parameters, the architecture can be adapted to the desired area and throughput requirements. This also offers inherent communication-computation parallelism in the design of every chiplet.
- **Novel compute acceleration:** Furthermore, we present new design techniques for the micro-procedures of FHE- the number-theoretic transform (NTT) and automorphism (AUT). Our approach introduces Hybrid NTT, eliminating the need for expensive transpose operation and scratchpad memory. It is easily scalable for higher or lower polynomial degrees. Hence, other applications, such as zero-knowledge proofs, can also benefit from this, where transposition is expensive due to high polynomial degrees. Additionally, we have prototyped these building blocks on FPGA- AlveoU250.
- **Application benchmark:** Finally, we choose parameters offering high precision and good performance. REED is the *first work to benchmark an encrypted deep neural network training*, showcasing practical and real-world impact. While CPU (24-core, 2×Intel Xeon CPU X5690 @ 3.47GHz) requires 29 days to finish it, REED 2.5D would take only 15.4 minutes, a realistic time for an NN training. We also use DNN training to run accuracy/precision experiments and validate our parameter choice.

2. Background

Let $\mathbb{Z}_Q$ represent the ring of integers in the $[0, Q-1]$ range. $\mathcal{R}_{Q,N} = \mathbb{Z}_Q[x]/(x^N+1)$ refers to polynomial ring containing polynomials of degree at most $N-1$ and coefficients in $\mathbb{Z}_Q$. In the Residue Number System (RNS) [22] representation, Q is a composite modulus comprising co-prime moduli, $Q = \prod_{i=0}^{L-1} q_i$. The RNS representation is used to divide a big computation modulo Q into much smaller computations modulo q_i such that the small computations can be carried out in parallel. With the application of RNS, a polynomial $a \in \mathcal{R}_{Q,N}$ becomes a vector, say $\boldsymbol{a}$, of residue polynomials. Let the i-th residue polynomial within

TABLE 1. CKKS PARAMETERS

Parameter	Definition
$N, n\ (\leq \frac{N}{2})$	Polynomial size, maximum slots packed
Q, q_i	Coefficient modulus, RNS bases $Q = \prod_{i=0}^{L} q_i$
L, l	Multiplicative depth (#RNS bases - 1) $l < L$
$dnum$	Number of digits in the switching key
P, p_i	Special modulus and its RNS base
$K\ (= \lceil \frac{L+1}{dnum} \rceil)$	Number of RNS bases for $P = \prod_{i=0}^{K-1} p_i$
w	Word size ($\log p_i, \log q_i$)
$L_{boot}, L_{\mathit{eff}}$	Multiplicative depth of/after bootstrapping

$\boldsymbol{a}$ be denoted as $a^i \in \mathcal{R}_{q_i,N}$. We use the 'mathtt' font (c/sk) to represent ciphertexts/keys. Operators $\cdot$ and $\langle , \rangle$ denote the multiplication and dot-product between two ring elements. Noise (e) is refreshed for every computation.

2.1. FHE schemes and CKKS routines

Different FHE schemes exist in literature, such as, BFV [19], BGV [9], CGGI [16], CKKS [14], [15]. These schemes use polynomial arithmetic but differ primarily in the data types they can encrypt. For instance, BGV and BFV encrypt integers, while CKKS encrypts fixed-point numbers. Due to the support for fixed-point arithmetic, CKKS is widely adopted for benchmarking machine learning applications [27], [33]. Therefore, this work targets the RNS (Residue Number System) CKKS [14].

In the following, we briefly describe the main procedures within the RNS CKKS [14] for ciphertexts at level l (multiplicative depth is $l-1$) where $l < L$, $Q_l = \prod_{i=0}^{l-1} q_i$, and L is the maximum level. The residue polynomial associated with each modulus q_i in the RNS representation is commonly called the RNS limb. A CKKS ciphertext consists of components, e.g., $\mathtt{c} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$, where $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ are vectors of limbs. Table 1 describes the CKKS parameters.

1) CKKS.Add($\mathtt{c}, \mathtt{c}'$): It takes two input ciphertexts c and c′ and computes $\mathtt{c}_{\text{add}} = (\boldsymbol{d}_0, \boldsymbol{d}_1) = (\boldsymbol{c}_0 + \boldsymbol{c}'_0, \boldsymbol{c}_1 + \boldsymbol{c}'_1)$.
2) CKKS.Mult($\mathtt{c}, \mathtt{c}'$): It multiplies the two input ciphertexts c and c′, and computes the non-linear ciphertext $\mathtt{d} = (\boldsymbol{d}_0, \boldsymbol{d}_1, \boldsymbol{d}_2) = (\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0, \boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}'_0, \boldsymbol{c}_1 \cdot \boldsymbol{c}'_1)$. Subsequently, CKKS.KeySwitch transforms d into a linear ciphertext.
3) CKKS.KeySwitch(d, ksk): It uses a KeySwitch or 'evaluation key ksk to homomorphically transform a ciphertext decryptable under one key into a new ciphertext decryptable under another key. It computes c″ where $\boldsymbol{c}''_0 = \sum_{i=0}^{l-1} d_2^i \cdot ksk_0^i \in \mathcal{R}_{PQ_l,N}$ and $\boldsymbol{c}''_1 = \sum_{i=0}^{l-1} d_2^i \cdot ksk_1^i \in \mathcal{R}_{PQ_l,N}$. This is followed by $\mathtt{c} = \big((d_0, d_1) + (\text{CKKS.ModDown}(\mathtt{c}'')\big) \in \mathcal{R}^2_{Q_l,N}$. CKKS.ModDown() scales down the modulus (PQ_l to Q_l).
4) CKKS.Rotate(c, *rot*, $\mathtt{ksk}_{\mathtt{rot}}$): It rotates the plaintext slots within c by *rot*. First, a permutation ρ is applied to the ciphertext polynomial coefficients. This permutation is called automorphism and is determined by the Galoi element $gle = 5^{rot} \bmod 2N$. Finally, the permuted ciphertext is processed by CKKS.Keyswitch using the rotation key $\mathtt{ksk}_{\mathtt{rot}}$.
5) CKKS.Bootstrap: It refreshes a noisy ciphertext [8], [11], [13] by producing a new ciphertext with a higher depth or lower noise. As bootstrapping itself consumes a certain number of depths, the depth of a bootstrapped ciphertext, say L_{eff}, is smaller than the initial depth L after fresh encryption. Bootstrapping is required in complex applications, e.g., DNN, to refreshing the processed ciphertexts.

2.2. FHE Hardware design goals

A tiered structure exists in the CKKS scheme routines. The high-level or *macro* rotuines are CKKS.Add, CKKS.Mult, CKKS.Rotate, and CKKS.KeySwitch. These macro procedures apply micro procedures, such as forward and inverse Number Theoretic Transforms (NTT/INTT), dyadic Multiplication/Addition/Subtraction (MAS), and Automorphism (AUT). The NTT is used for multiplying two N coefficients long polynomials in $\mathcal{O}(N \log N)$ time complexity, which is the asymptotically fastest one.

The special CKKS.Bootstrap procedure uses these macro procedures in a specific sequence to refresh noisy ciphertexts. Note that contrary to schemes like FHEW, TFHE [16] where bootstrapping is a standalone procedure, CKKS-Bootstrapping is a high-level routine which utilizes KeySwitches, Automorphisms, and MACs. Therefore, while TFHE/FHEW accelerators (e.g., [6]) focus on optimizing the programmable bootstrapping, acceleration for CKKS relies on optimized KeySwitching, Automorphisms, and MACs. Among these operations, MAC is a straightforward linear operation. Automorphism involves permutation, followed by KeySwitch. This permutation, if naively implemented, can become complex and expensive as the input polynomial has N=2^{16} coefficients and offers N/2 different permutations. In this work, we show how we design the permutation unit that is not only cheap in terms of area but also has linear time complexity for all permutations. The final operation-KeySwitch, is the most expensive due to the expensive base-conversion step (Figure 1). Since KeySwitch is the most expensive operation, the task and data distribution approach aims to optimize this particular operation. For simplicity, KeySwitch for $dnum = L+1$ ($K = 1$) is utilized throughout the paper.

2.3. Monolithic vs Chiplet packaging

In the context of large Integrated Circuits, authors in [25], [44], [81] discuss the advantages of chiplet-based designs over monolithic designs. The problem with monolithic designs stems from the fact that to keep up with the increasing demand for high performance and functionality, chips need to be scaled up, and advanced technology nodes must be utilized. Manufacturing such big chips reduces the wafer yield as more surface area is exposed to defects per chip and increases the development cost. Such huge designs

take a long time-to-market, and it is not straightforward to test and verify them. More factors, such as size limitation and sub-optimal die performance due to overload, contribute to a shift to SiP [25].

In SiP, multiple heterogeneous smaller chiplets can be manufactured separately and later integrated together using various packaging techniques. This promotes chiplet-reuse, lowering the development costs. The chiplet-packaging techniques can be broadly classified into three main categories: 2D, 2.5D, and 3D [25], [44]. In 2D packaging, different dies are mounted on a substrate, known as a multi-chip module. Due to substrate limitations, this results in slow die-to-die communication and high power consumption.

To address these limitations, silicon interposers are used, and this technique is known as 2.5D integration [1], [57]. In this approach, an interposer is placed between the die and the substrate, enabling die-to-die connections on the interposer itself. The use of an interposer significantly enhances interconnectivity, leading to improved performance. Taking the integration capabilities a step further, 3D packaging involves stacking different dies on top of each other, akin to a skyscraper. In 3D packaging, the dies are interconnected using through-silicon vias (TSVs). 3DIC is gaining significant popularity and serves as the foundation for advancements like High Bandwidth Memory (HBM/HBM2/HBM3) [10], [56], [74], [76]. This approach significantly reduces the critical path and area, resulting in higher performance, lower power consumption, and increased bandwidth. The slowdown of Moore's law finds hope in 2.5D and 3D IC.

3. FHE-tailored multi-chiplet design

A widely adopted approach for realising a chiplet-based architecture is to distribute the components of one large monolithic design across multiple chiplets connected in a mesh [12], [39], [60]. While such a disintegration approach has found utilities in applications like Machine Learning [69], they fall short in leveraging the inherent algorithmic intricacies of FHE, thereby hindering efficient workload distribution among chiplets. In the following, we investigate the trade-offs of various chiplet design possibilities for FHE and consolidate on an optimal solution. For this, let us consider the most performance-heavy macro routine-KeySwitch. Its data flow is illustrated in Figure 1 for $l = 3$.
① The naive approach for chiplet decomposition would be to closely follow the data flow of Figure 1 and allocate one chiplet per square-box in the figure (I=INTT, F=NTT, K=Key Multiplication). This approach will lead to (i) uneven allocation of chiplet resources; for example, the chiplets for NTT or INTT will be much larger than those for MAS, (ii) a massive increase in C2C communication overhead due to the continuous data exchange between the components, and (iii) increase in required on-chip memory for data duplication. Thus, this approach significantly inhibits the performance of the design. A deeper disintegration strategy would imply breaking the individual square-boxes into several chiplets, for example computing one NTT using several chiplets. Although this approach can lead to smaller

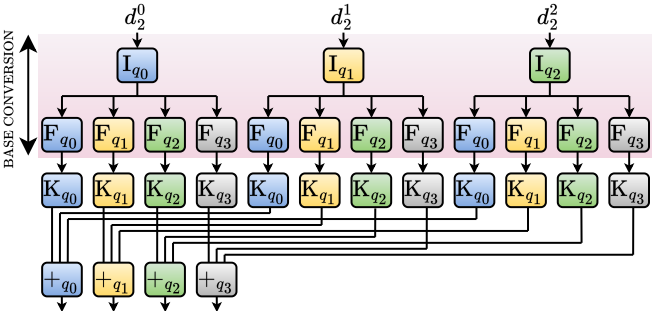

Figure 1. KeySwitch operation for $l = 3$, where I, F, and K represent INTT, NTT, and key multiplication operations using MAS, respectively.

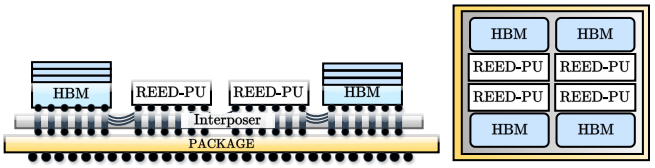

Figure 2. Side and top view of proposed four chiplet-based REED 2.5D.

chiplets, it will lead to additional C2C communication overhead.
② The second approach involves assigning each RNS limb computation, as depicted in Figure 1, to a single chiplet encompassing NTT, AUT, and MAS components. Computations within boxes of the same colour are performed within the same chiplet. This approach also faces slow C2C communication overhead due to data dependencies between the processing of RNS limbs in the KeySwitch routine. Furthermore, as the depth of the FHE application decreases over time, reducing the number of RNS limbs (l), their respective chiplets become idle.
③ A refined third approach involves enabling each chiplet to support multiple RNS limbs to reduce C2C communication overhead. For instance, the first chiplet can initiate NTT_{q_1} after executing INTT_{q_0} without sending it to the second chiplet, as depicted in Figure 1. This strategy minimizes communication overhead and intermediate storage requirements by storing copies of input RNS limbs d_2^0, d_2^1, d_2^2 in Figure 1 on each chiplet. Hence, INTT_{q_0}, INTT_{q_1}, NTT_{q_0}, etc., in Figure 1 are computed within the same chiplet. Although it reduces C2C communication, more memory is required due to duplicate RNS limbs.

Thus, all the available approaches offer certain trade-offs, and our aim is to determine the best and most practical solution. With this aim, we develop a chiplet-based design approach for REED.

3.1. REED 2.5D Architecture

In a chiplet-oriented design process, there are two critical choices: the size of chiplets and the number of chiplets. The manufacturing cost is reduced, and yield increases when

the chiplets are small in area. However, having many small chiplets reduces performance as the complexity of slow C2C communication increases. In this work, we will develop a chiplet design strategy and integration topology, considering the data flow of FHE. This approach provides a balance between yield, manufacturing cost, and C2C communication overhead.

As an example, Figure 2 shows a four chiplet-based REED 2.5D architecture, where the chiplets are connected in a ring formation and have exclusive read/write access to HBM in its proximity. Later, we will show that this architecture scales well with an increasing number of chiplets (Section A). To overcome C2C communication overhead and memory storage issues, we propose an RNS polynomial (limb)-oriented task and data distribution strategy, which is built on top of approach ③. Specifically, chiplets are assigned certain RNS limbs and all tasks related to these limbs without requiring data duplication (detailed in Section 5). The proposed ring formation (Section 5.3) allows us to increase the number of chiplets at the cost of only a linear increase in the number of interconnects. Hence, we can scale it to eight or sixteen chiplets as well. This ring formation for connecting the FHE chiplets is specifically tailored to the data-flow of performance-critical FHE workloads. With this formation, not all dies need to communicate with every other die simultaneously, which is crucial for minimizing C2C communication requirements.

Furthermore, our communication protocol ensures (Section 5.3) that no HBM-to-HBM communication is required. Hence, HBMs are positioned on the outer side. We also avoid sharing one HBM among multiple chiplets, ensuring that each HBM is located only in proximity to the one chiplet it serves. Notably, our placement strategy aligns well with [57], [82], where authors design chiplet-based general-purpose processors with an actual tapeout, demonstrating practical viability. Finally, we also ensure a homogeneous design where all chiplets are identical, simplifying post-silicon realization.

Disintegration Granularity: Chiplet systems face a trade-off between development cost and performance degradation, depending on the disintegration granularity. Existing works on chiplet-based architectures, such as [26], [44], [74], [76], [81], [82], show that disintegration improves yield, but it introduces challenges such as floorplanning and post-silicon testing overhead. Hence, the question:

How much disintegration is too much disintegration?

Considering a maximum die area of 800mm^2, dividing it into four chiplets offers an $\approx 80\%$ yield, while eight chiplets provide a yield of $\approx 90\%$. The yield numbers are obtained from [44].

While the eight-chiplet option shows promise for achieving high yield, it faces the challenge of underutilization over time as the number of RNS limbs decreases after rescaling. Specifically, when l becomes smaller than 8, certain REED chiplets remain idle (Detailed in Section 5.2 and Section A). On the contrary, the instantiation of four chiplets strikes an optimal balance between manufacturing cost and utilization. We want to remark that the number of REED chiplets is flexible and can be changed as per user requirements, depending on the technology and computation constraints.

4. Architecture Design of One Chiplet

The need for scalability and high throughput drives our design methodology. We introduce the REED design configuration- (N_1, N_2) for polynomial degree N, where $N_1 \cdot N_2 = N$. For the clock frequency f, this configuration provides a throughput of $\frac{f}{N_1}$ operations per second and can process N_2 coefficients in parallel. A configuration-flexible design approach will help obtain computation-communication parallelism within every chiplet by ensuring that the memory read/write throughput is the same as the computational throughput. Now, let us explore how we design the ingredients of REED Processing Unit (PU) to ensure flexibility.

4.1. The Hybrid NTT (Frankenstein's Approach)

The NTT/INTT unit plays a vital role in converting polynomials from slot to coefficient representation and vice versa. It is the most computationally expensive micro building-block and occupies over 50% architectural area. Therefore, designing an efficient NTT/INTT unit is crucial as it directly impacts the overall throughput and area consumption of REED.

Prior works: There are various approaches in the literature to implement NTT in hardware for large-degree polynomials, such as iterative [47], [67], pipelined [80], [83] and hierarchical [21]. The implementation complexity of the plain iterative approach increases significantly with the number of processing elements. The pipelined approach (also known as single-path delay feedback (SDF)) provides a bandwidth-efficient solution but a diminished performance. The hierarchical approach (also referred to as four-step NTT), utilized in [21], treats a polynomial of size N as an $N = N_1 \times N_2$ matrix and divides a large NTT into smaller parts. It involves performing N_1-point NTTs on the N_2 columns of the matrix, then multiplying each coefficient by $\omega^{i \cdot j}$ (where i and j are matrix row and column indices), transposing the matrix, and finally performing N_2-point NTTs on the N_1 columns.

Transposing a matrix of size $N_1 \times N_2$ requires N_1 separate memories and large data re-ordering units. Hence, in [21], the transpose unit consumes 14% of the area per compute cluster. Moreover, it also requires additional N_2 cycles for writing data to the transpose memory and N_1 cycles for reading it. Therefore, although the hierarchical approach simplifies the NTT implementation, we observe that it has the following limitations: (i) the costly transpose operation, (ii) fixed N_1 and N_2 such that $N_1 = N_2$ [21], [23], and (iii) the reliance on scratchpad in some works leads to large memory fan-in and fan-out, causing routing inefficiencies.

Our Technique: We aim to design an NTT that is routing-friendly, throughput-oriented, and does not necessitate costly

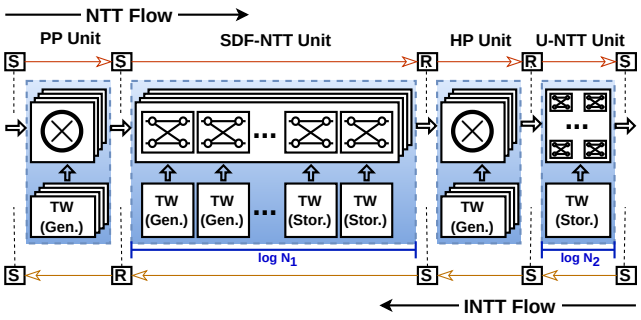

Figure 3. Novel routing-friendly Hybrid NTT/INTT design flow for $N = N_1 \times N_2$.

Algorithm 1 Hybrid NTT with NWC

In: a (a matrix of size $N_1 \times N_2$ in row-major order)
In: ω (N-th root of unity), ψ ($2N$-th root of unity)
Out: $a = \text{NTT}(a)$ (a matrix of size $N_1 \times N_2$ in column-major order)

1: **for** $(i = 0; i < N_1; i = i + 1)$ **do**
2: **for** $(j = 0; j < N_2; j = j + 1)$ **do**
3: $a[i][j] \leftarrow a[i][j] \cdot \psi^{i \cdot N_2 + j} \pmod q$ ▷ PP:Pre-processing
4: **end for**
5: **end for**
Apply N_1-pt NTT to the columns of a ▷ using SDF-NTT
6: **for** $(i = 0; i < N_1; i = i + 1)$ **do**
7: **for** $(j = 0; j < N_2; j = j + 1)$ **do**
8: $a[i][j] \leftarrow a[i][j] \cdot \omega^{i \cdot j} \pmod q$ ▷ HP:Hadamard prod.
9: **end for**
10: **end for**
Apply N_2-pt NTT to the rows of a ▷ Unrolled(U)-NTT
11: **return** a

transposition. To achieve this, we utilize parts of hierarchical, iterative, pipelined, and plain unrolled NTTs (Frankenstein's approach) and introduce a novel Hybrid NTT. It is fully pipelined, and its flow is shown in Algorithm 1 and Figure 3.

During the NTT operation, we first perform preprocessing (Step 3 of Algorithm 1) using N_2 modular multipliers (PP). The resulting coefficients are sent to N_2 pipelined NTT units (N_1-pt SDF-NTT). The output coefficients of SDF-NTT units are processed via the Hadamard Product unit (HP) that multiplies the coefficients with powers of roots of unity (ω) using N_2 modular multipliers. Finally, we employ a N_2-pt unrolled NTT (U-NTT) unit. The properties and advantages of the proposed unit are as follows.

❶ **Transpose elimination:** The Hybrid NTT eliminates transpose by using two orthogonal NTT approaches, pipelined (SDF) approach for N_1-sized NTTs and unrolled (U-NTT) approach for N_2-sized NTTs. As shown in Figure 4 (a), the output coefficients of SDF-NTT are processed directly by U-NTT, providing a seamless, natural transpose operation.

❷ **Bi-directional workflow:** The above method of transpose elimination also helps make our NTT unit bi-directional (for INTT), as illustrated in Figure 4 (b). The additional routing complexity is balanced with efficient pipelining.

❸ **Low-level optimizations:** For modular multiplication and reduction unit, we adopted the word-level Montgomery [48], [49] modular reduction algorithm. and optimized it for our special prime form, $2^{w-1}+q_H \cdot 2^m+1$, where $m = 18$ is Montgomery reduction size, and $\lceil \log_2 q_H \rceil = 10$ is small. A total of $(N_2+1)\log_2(N1)+\frac{N_2}{2}(\log_2(\frac{N_2}{2})+5)-7$ modular multipliers are utilized.

❹ **On-the-fly twiddle generation:** To reduce the on-chip twiddle factor memory, we employ on-the-fly generation [47] using a small constant memory that stores a few initial roots of unity. This helps reduce the on-chip constant storage by up to 98.3%.

Algorithm 2 Automorphism

In: $a[N_1][N_2], gle$
Out: $\hat{a} = \rho(a)$

1: $index \leftarrow gle$
2: **for** $(l_0 = 0; l_0 < N_1; l_0 = l_0 + 1)$ **do**
3: $l_1 \leftarrow index \pmod{\log(N_1)}$
4: $start \leftarrow index \gg \log(N_1)$
5: $addr[j] \leftarrow (start + j \cdot gle) \pmod{\log(N_2)} \ \forall \ j \in [0, N_2)$
6: $\hat{a}[l_1] \leftarrow \texttt{shuffle\_tree\_}N_2 \times N_2(addr, a[l_0])$
7: $index \leftarrow index + gle$
8: **end for**
9: **return** $\hat{a}[N_1][N_2]$

4.2. Multiply-Add-Subtract (MAS) and Automorphism/Conjugation (AUT)

MAS is elementarily designed as a *triadic* unit for computing point-wise multiplication, addition, subtraction, or multiply-and-accumulate operations. It utilizes the modular multiplier proposed for the Hybrid NTT unit. On the other hand, designing an efficient AUT unit is challenging [21], [66]. It permutes ciphertexts using the Galois element (gle) to achieve rotation or conjugation. A polynomial is stored as a matrix $N_1 \times N_2$ in N_2 memories.

For AUT, we make a key observation that when we load N_2 coefficients from memory address l_0 across all N_2 memories, they are shuffled based on the desired rotation offset (ρ_{rot}), and then written to address l_1 across all N_2 memories. Hence, even though the coefficient order is shuffled, they all go to the same address of N_2 distinct memories. We utilize this property to permute all N_2 coefficients in parallel. This out-of-place automorphism is presented in Algorithm 2. The in-place permutation techniques proposed in previous

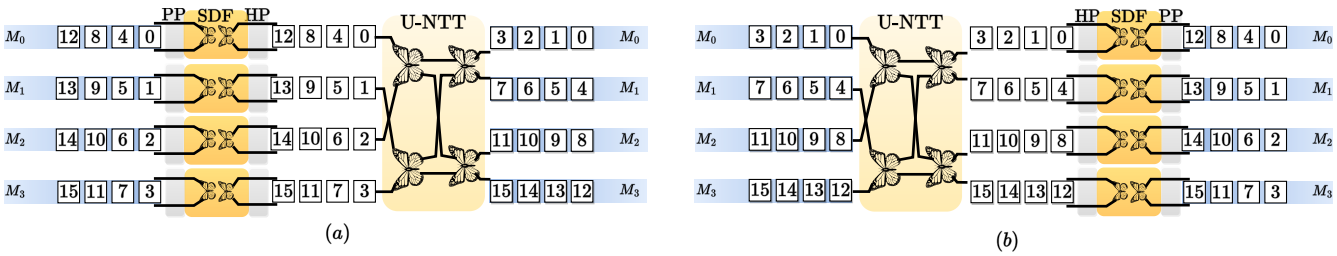

Figure 4. The proposed novel Hybrid NTT/INTT design flow with Memory access for (a) NTT and (b) INTT with $N_1 = 4$, $N_2 = 4$, and $N = 16$. The butterflies represent the Gentleman-Sande butterfly [67] operation employed in our design.

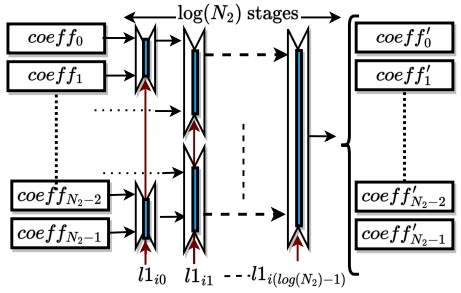

Figure 5. An example of a `shuffle_tree_`$N_2 \times N_2$ workflow. Every stage has sufficient registers to hold N_2 coefficients.

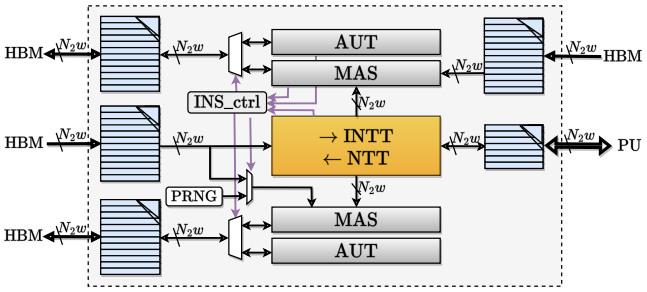

Figure 6. The REED-PU design. Every data communication (memory to building blocks and off-chip to on-chip) here has a bandwidth of N_2w bits/clock-cycle.

works [21], [66] increase routing complexity due to memory transposition requirements.

At the end of this, we are still left with a quadratically complex and expensive shuffle $\mathcal{O}(N_2^2)$ among the coefficients. However, we analyzed that all the shuffles could be performed pairwise on the coefficient batches, as shown in Figure 5. After each stage, two batches of coefficients are merged to form a new batch. With this, we replace the naive and expensive operation with a pipelined binary-tree-like shuffle. Its number of pipeline stages adjusts with N_2, making the pipelining scalable and efficient for higher configurations. Moreover, the unit can handle any arbitrary rotation. This concludes the design of micro procedures.

4.3. Programmable Instruction-Set Architecture

In this section, we discuss how to program the micro procedures (NTT, AUT, MAS) for high-level FHE routines. This is crucial for determining their placement in the architecture, which will be discussed in the subsequent section.

Prior works define a strict operation flow. This prevents adaptations to future changes in the FHE algorithms or routine flow. Noting this, we utilize an instruction-based architecture design technique [71], wherein a relatively small instruction controller manages the multiplexers and collects 'done' signals from these units. Two types of instructions are: *micro* and *macro*. Micro-instructions are low-level arithmetic procedures like NTT, INTT, point-wise modular addition, subtraction, multiplication, multiplication-and-accumulation, and automorphism. They are used to compose microcodes for realizing macro-instructions for homomorphic addition/multiplication, KeySwitch, rotation, and moddown. A bootstrapping is performed using these macro instructions.

4.4. REED Processing Unit (PU)

We initiate the PU design, as shown in Figure 6. A PRNG is deployed to generate half the key components on the fly [47]. We uncover two important design decisions. The first is regarding the placement of NTT/INTT and MAS/AUT units. The second deals with the problems associated with large on-chip memories utilised in prior works [34], [35] for storing keys.

Ⓘ The polynomial processed by NTT unit is multiplied with two polynomials of KeySwitch keys and accumulated. Hence, we instantiate a pair of MAS/AUT units capable of simultaneously processing both key components. Thus, the design has the ability to run NTT and MAS units concurrently (shown in Figure 7), which improves the KeySwitch performance by 66.7%, as explained in Figure A. Moreover, since the AUT and MAS units are relatively cheaper, this design decision also does not add significant area overhead, as presented later in Table 2.

Ⓘⓘ In hardware accelerators, on-chip memory causes significant area overhead. Chiplets with large on-chip memory are not power, area, and manufacturing cost efficient [70]. In the context of FHE, each KeySwitch key demands $\approx$ 820MB or 91MB storage for $dnum = L + 1$ or 3 respectively. Given the limited on-chip memory capacity of small chiplets, accommodating even a single KeySwitch key

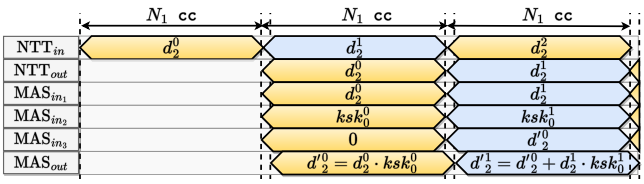

Figure 7. Timeline of parallel and pipelined operation flow.

becomes rather challenging. Consequently, reliance on off-chip memory access becomes essential when a KeySwitch operation necessitates a different key. In our architecture, we store the keys in large off-chip HBM. To reduce the overhead of off-chip memory access, we develop an efficient prefetch unit that streamlines data movements in parallel to computation, as described next.

4.5. Streamlined Prefetch for On-Chip Storage

As mentioned in Section 4.1, REED's design methodology mitigates the need for scratchpad-like on-chip memory, allowing us to use memory units solely as prefetch units.

Each memory unit in Figure 6 exhibits balanced fan-in and fan-out, and among the five memory units depicted, four are fed by off-chip memory. The small memory is responsible for storing and communicating the INTT result to the other PUs (or Chiplets) (elaborated in Section 5.3). Only two of the four memories communicating with off-chip memory need to write back the results, as illustrated by bi-directional arrows in Figure 6.

Summarily, three memories perform off-chip read/write communication. These memories are physically divided into two parts. When one is utilized for on-chip computation, the other performs off-chip prefetch (similar to ping-pong caching).

5. Techniques for exploiting Comm-Comp Parallelism

In the previous section, we discussed how a configuration-based design methodology enables off-chip and on-chip communication-computation parallelism. However, when distributing FHE workloads among multiple chiplets, we also have to consider the C2C communication overhead. This is important as state-of-the-art HBM3 [58] features a bandwidth of 1.2TB/s, while the state-of-the-art C2C interconnect UCIe [1], [2] only offers a bandwidth of 0.63 TB/s. Consequently, a chiplet system optimized for HBM bandwidth would face bottleneck due to slow C2C communication.

The routine that necessitates most data exchange is KeySwitch. It switches the modulus (base-conversion) of each L residue polynomial (INTT$(d_{2q_i})_{q_i}$ $\forall i \in [0, L)$) to $(L+1)$ residues polynomials (NTT$(d_{2q_i})_{q_j}$ $\forall j \in [0, L]$), followed by key multiplication.

5.1. Communication cost-analysis

In a multi-PU work [47], the authors briefly discuss limb-based decomposition and propose distributing computation across the RNS polynomials (limbs) by employing one PU per limb. While this approach enables highly parallel computations, as the multiplicative depth decreases, many PUs become idle, causing underutilization.

In [34]–[36], the authors utilize both the limb-based and coefficient-based task distribution during KeySwitch. They [34], [35] propose using limb-wise distribution for INTT and NTT steps and coefficient-wise distribution for base-conversion. [35] utilizes four PUs, and since base conversion is needed between the INTT and NTT steps, all-to-all broadcasts are done across PUs to switch from one distribution to another. In Figure A, we analyze how both techniques attain the same communication overhead.

Additionally, all-to-all C2C broadcast between the chiplets is slow and increases by $\mathcal{O}(\mathtt{r}^2)$ with the number of chiplets $\mathtt{r}$. In the context of FHE, switching between limb- and coefficient-wise task distribution becomes expensive as it demands all-to-all C2C data movements. For example, [35] proposes utilizing four PUs in a single monolithic chip. However, when extended to a chiplet setting, where each PU occupies a separate chiplet, the lack of an all-to-all broadcast capability makes it difficult to send data across all chiplets instantly. Using bi-directional C2C communication ability, the polynomial would reach all four chiplets via at least two serial C2C communication interfaces. The on-chip bandwidth used in the prior works is (20TB/s [35], 36TB/s [34]) is much less than the state-of-the-art C2C communication bandwidth (0.63TB/s [1], [2]). As a result, chiplets would have to wait longer for data to arrive before computing, and this C2C communication overhead will significantly inhibit the performance. Hence, there is a need to devise a schedule that can couple most of the communication with computation.

5.2. Data Distribution across Multiple Chiplets

The above analysis establishes that limb-based decomposition is indeed the best technique for task and data distribution across multiple chiplets. Within one chiplet, the computation is coefficient-wise distributed as N_2 coefficients are processed in parallel (discussed in Section 4). However, as discussed in Section 3 ③, the limb-based data and task technique also has limitations. Therefore, we adapt it to offer long-term high-performance benefits.

This adaptation stems from the two key observations from the data flow of the KeySwitch operation in Figure 1. Firstly, the INTT results that need to be shared and duplicated are ephemeral, and thus, they do not require any long-term storage – subsequent operations immediately consume them. To improve the efficiency, instead of duplicating the INTT data or sharing memory across chiplets, we leverage a *ring-based C2C communication* as shown in Figure 8. The chiplets are connected in a ring formation where each chiplet processes one INTT result and then sends it to the

next chiplet. The ring-based data movement minimizes the number of C2C interconnects and ensures that each chiplet operates on independent memory, simplifying placement and routing constraints.

The second observation is related to the underutilization of chiplets with a reduction in the number of levels or depths in the ciphertext after homomorphic rescaling. If we distribute the limbs across the chiplets in an interleaved manner, then as the multiplication depth decreases, the number of limbs per ciphertext in each chiplet also uniformly decreases. To explain the benefits of the interleaved distribution, we will use the analogy of a card game.

The FHE card game: In this game, each player represents a chiplet, while the residue polynomials or limbs act as the cards. The cards dealt out are collected in a LIFO (last-in-first-out) manner [1], imitating the loss of multiplicative depth during computation. All players engage in the game (FHE routine computation) until they exhaust their cards. The start of a new FHE computation game mirrors Bootstrapping, which starts when only one player retains a single card. Until this point, players without cards must wait until the next game to participate. With these rules, the dealer (user or compiler [75]) has two choices: deal out all the cards to one player before moving to the next or do alternate distributions such that every other card goes to different players. Let us say there are $L = 32$ cards and $\mathtt{r} = 4$ players. In the former case, the first player gets the cards drawn at instances $\{0, 1, 2, \cdots, 7\}$, and in the latter case, at instances $\{0, 4, 8, \cdots, 28\}$, and so on for the other players.

In both scenarios, each player receives 8 cards. In the first option, the last player exhausts their cards first, followed by the preceding player, and so on. Consequently, until the first player runs out of cards, others remain inactive. Conversely, with alternating distribution, each player loses one card in turn. Thus, at any point in the game, players either possess the same number of cards or one less, ensuring active involvement throughout.

The goal of FHE architecture design is to ensure the full utilization of chiplets in the long term and, thus, deliver high performance for the available computation resources. It translates to maximizing the player interaction in the FHE card game. Thus, the latter technique of interleaved alternate distribution offers maximum chiplet participation in the "card game" of computations. Now, if there are too many players, then the number of players becoming idle will increase no matter which technique is used. The latter technique will only minimize the idle time. This is the problem with using more chiplets. Thus, next, we will discuss our final key technique for reducing communication overhead and then derive a good upper-bound on the number of chiplets.

5.3. Efficient Non-Blocking C2C Communication

The proposed ring-based communication still faces overhead as the chiplets have to initially wait for every chiplet before it in the ring to send the INTT result. Hence, we propose a communication strategy, illustrated in Figure 8, to overcome this remaining problem.

The key idea is that the chiplets concurrently operate on different limbs instead of waiting for one limb and then processing it, as shown in Algorithm 3. Each chiplet starts with the assigned limb, computes INTT, and then performs multiple NTTs on it. While performing NTT, it starts sending/receiving the INTT result. For example, the REED_0 sends its INTT result to REED_3 and receives the INTT result from REED_1. This is a uni-directional *ring-*

[1] It can also be FIFO (first-in-first-out) without any loss of generality.

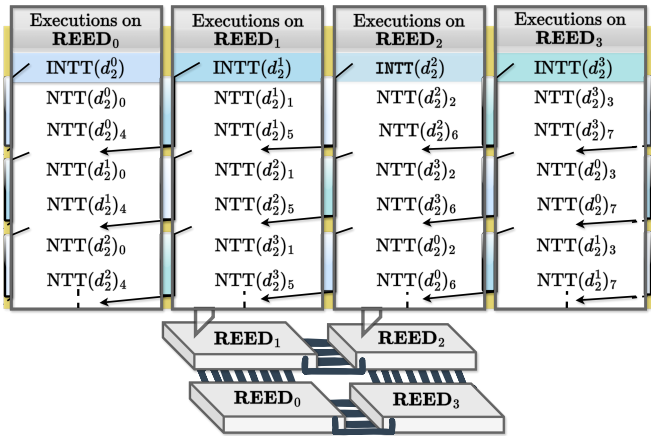

Figure 8. Non-blocking ring-based communication for four REED chiplets when $L = 7$. The blocks between executions represent the long communication window to make up for slow inter-chiplet (C2C) communication.

Algorithm 3 KeySwitch

In: $\boldsymbol{d_2}$ (the ciphertext component to be linearized)
Out: BUF $= KeySwitch(\boldsymbol{d_2})$

Following tasks are executed by REED_i $\forall\ i \in [0, r)$
▷ All REED_i operate **in parallel** as shown in Figure 8
for $(j = 0; j < \frac{l}{r}; j = j + 1)$ **do**
  $\mathtt{I_i^{rcv}} \leftarrow \mathtt{INTT}(\boldsymbol{d}_2^{j \cdot r + i})$
  ▷ *Initiate communication with* $\text{REED}_{(i+1)\%4}$, $\text{REED}_{(i-1)\%4}$
  for $(m = 0; m < r; m = m + 1)$ **do**
    $\mathtt{I_i^{proc}} \leftarrow \mathtt{I_i^{rcv}}$
    ▷▷ *Long Communication window opens now* ◁◁
    ▷ *Start receiving* $\mathtt{I}^{\mathtt{rcv}}_{(\mathtt{i}+1)\%4}$ *from* $\text{REED}_{(i+1)\%4}$
    ▷ *Start sending* $\mathtt{I_i^{rcv}}$ *to* $\text{REED}_{(i-1)\%4}$
    for $(t = 0; t < \frac{l+1}{r}; t = t + 1)$ **do**
      $\text{BUF}_{t*r+i} \mathrel{+}= (\mathtt{NTT}(\mathtt{I_i^{proc}}) \cdot \mathtt{KSK}^{j \cdot r + (i+m)\%4})_{q_{t*r+i}}$
    end for
    ▷▷ *Ensure* $\mathtt{I}^{\mathtt{j \cdot r + (i+m+1)\%4}}$ *has been received*
    ▷▷ *Communication window closes* ◁◁
    $\mathtt{I_i^{rcv}} \leftarrow \mathtt{I^{rcv}_{(i+1)\%4}}$
  end for
end for
return BUF

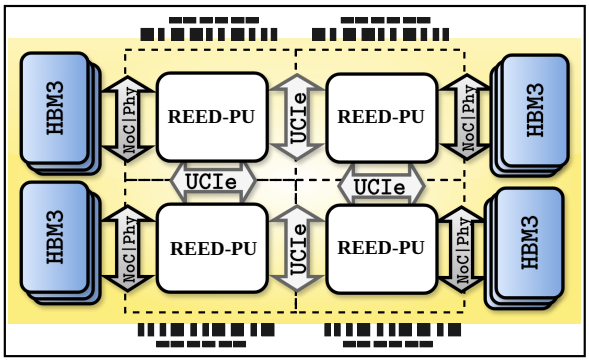

Figure 9. The complete architecture diagram of 4-chiplet REED 2.5D for 1024×64 configuration. The multiple small black blocks denote I/O interconnects along the edges.

based communication. Since only one INTT result needs to be sent for $\frac{l+1}{\mathtt{r}}$ parallel NTT computation, we have a larger C2C communication window compared to computation. Consequently, *non-blocking communication* is achieved as data computation can proceed concurrently with relatively slower communication.

This technique necessitates just one read/write port per chiplet, in contrast to the requirement for $(\mathtt{r}-1)$ ports in a star-like (i.e., all-to-all) C2C communication network.

The Adapted Distribution Technique: C2C communication bandwidth plays a major role in the performance versus the number of chiplets trade-off. Let us assume the C2C communication bandwidth is k× slower than the HBM to Chiplet communication bandwidth. Each chiplet operates on $\frac{L+1}{\mathtt{r}}$ polynomials. The total computation time to process all $L+1$ polynomials for the KeySwitch should be close to k. Otherwise, we will not be able to decouple the communication from computation as discussed above. This offers us a loose upper bound $\mathtt{r} < \frac{L+1}{k}$. To ensure $u\times$ higher utilization, this bound must be made tighter. u can take any value $\leq \frac{L+1}{k}$ ($u = \frac{L+1}{k}$ for monolithic chip). We take u as 4. The adapted limb-based task/data distribution technique has following properties:

- The number of chiplets is constrained by $\mathtt{r} \leq \frac{L+1}{4k}$.
- An interleaved data/task distribution approach is utilized such that Chiplet$_i$ gets data and task corresponding to limbs $\mathtt{r}j+i \; \forall \; 0 \leq j < \frac{L+1}{\mathtt{r}}$, instead of sequential allotment (Chiplet$_i \leftarrow \mathtt{r}i+j$).
- The technique outlined in Algorithm 3 is to be followed by all Chiplets to minimize data exchange overhead and costly C2C interconnects.

6. Implementation Results

Based on the precision-loss study done for $dnum = L$ (shown in Section 7.1), we choose the overall parameters for synthesizing and benchmarking our design as $N = 2^{16}, L = 30, K = 1, L_{boot} = 15, w = 54$. Upon implementation (silicon realisation), only the parameters N, w are fixed, and the other parameters (e.g., $dnum$) can be changed as per application requirements

TABLE 2. TOTAL AREA CONSUMPTION OF 4-CHIPLET REED 2.5D FOR DIFFERENT CONFIGURATIONS ON 28NM AND 7NM.

Components	28nm (mm^2)		7nm (mm^2)	
	1024×64	**512×128**	**1024×64**	**512×128**
REED	74.9	115	24	43.9
REED-PU	58.0	81.0	7.01	9.9
⌊ NTT/INTT	38.2	56.8	5.61	7.9
⌊ 2×MAS	3.1	6.6	0.42	0.76
⌊ PRNG	0.15	0.28	0.02	0.04
⌊ 2×AUT	0.14	0.32	0.02	0.04
⌊ Memory	16.1	16.1	1.2	1.2
HBM PHY/NoC	16.9	33.8	16.9	33.8
4×REED	299.6	392.4	96	175.6
C2C	12.32	14.64	0.8	1.6
Total Area	**311.9**	**461.4**	**96.7**	**177**

We synthesize REED 2.5D for configurations 1024×64 and 512×128 using TSMC 28nm and ASAP7 [17] 7nm ASIC libraries with Cadence Genus 2019.11, and SRAMs are used for on-chip memories. We simulated our design using Vivado 2022.2. Moreover, we take a step further by *prototyping* the building blocks on Xilinx Alveo U250 to verify functional correctness, which has not been done by prior ASIC FHE accelerators. Our primary objective is to achieve high performance while optimizing area and power consumption. To this end, we set our clock frequency target to 1.5 GHz, use High-vt cells (hvt) configuration for low leakage power, enable clock-gating, and set the optimization efforts to high. We set the input/output delays to 20% of the target clock period and leverage incremental synthesis optimization features.

As off-chip storage, we leverage the state-of-the-art HBM3 [30], [41], [56], [58] memory, offering improved performance and reduced power. It is already deployed in commercial GPUs and CPUs [18]. HBM3 with 8/12 stacks of 32Gb DRAMs has 32/48 GB storage capacity [30], [51]. The ciphertexts provided by the client can be transferred to REED using 32 lanes PCIe5 offering a bandwidth of 128 GB/s [72]. In our work, we present results for HBM3 PHY and HBM3 NoC, based on [10], [56], [58] with reported bandwidth of 1.2TB/s [58]. For C2C communication, UCIe advanced interconnect can offer a bandwidth of 0.63 TB/s [1], [2] for 2.5D integration.

Table 2 presents the area results for the REED 2.5D architecture, featuring a 4-chiplet configuration as illustrated in Figure 9. This design conforms to the fabricated chiplets systems [4], [29]. The inner REED-PU, NoC, and HBM (shown in Figure 9) constitute one chiplet (similar to [57], [82]). In Table 3, we present the performance of FHE routines for both configurations (512×128 and 1024×64) with the achieved target clock frequency of 1.5 GHz. Figure A explains how the throughput of KeySwitch is obtained.

6.1. Power and Performance Modelling

We obtain the performance and power consumption estimates for REED by using a cycle-accurate model. REED's communication and computation are decoupled by design and do not need application-specific schedules to reduce

TABLE 3. PERFORMANCE MICRO-BENCHMARKS FOR 28NM AND 7NM.

Micro-Benchmarks ↓ **Configuration →**	**Level** **1**	**Time (ms)** **1024×64**	**512×128**
AUT/MAS (pt-ct)	30	0.005	0.003
MAS (ct-ct)	30	0.01	0.005
KeySwitch	30→31	0.19	0.08
MULT & Relin.	30→29	0.22	0.11
Bootstrapping	1→30→15	14.2	7.1

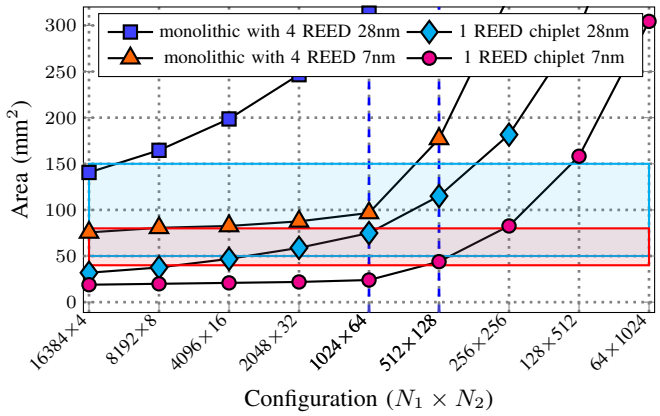

Figure 10. Increase in area with REED configurations, put in the order of increasing throughput [26].

data load/store stalls. REED also has a modular architecture design; all chiplets are identical, and none of the elementary building blocks is distributed across chiplets.

In Section 4.3, we elaborated on how our instruction-set architecture handles the micro (NTT, AUT, MAS) and macro (e.g. rotation, KeySwitch) instructions. Thus, a user does not need to handle micro-instructions due to the provided macro-instruction level abstraction. Predefined microcode for static macro-instructions ensures optimized data flow and memory management. Data exchange across chiplets occurs solely during KeySwitch, which is incorporated into the microcode, along with task distribution.

The simulator takes into account the bandwidth of C2C and HBM-chiplet communication along with data communication and distribution strategies (Section 5.2, Section 5.3). The run-time of macro-instructions is obtained using the known static schedule of micro-instructions. Finally, the macro-instructions are scheduled using OpenFHE [3], and runtime is obtained for higher-level operations (bootstrapping, DNN). Similarly, power consumption for elementary building blocks is determined using the Cadence toolchain. The model utilizes it to estimate overall power consumption based on utilization.

6.2. What to expect from higher-throughput configurations?

Until now, we have examined two configurations (1024×64 and 512×128) that only partially demonstrate the advantages of our proposed scalable design methodology. As we double the throughput (by doubling the value of N_2), the area of PU only increases by approximately 1.5×. This trade-off arises because the chip area comprises two components— (i) the computation logic area, which scales linearly with throughput, and (ii) on-chip storage that remains fixed to a number of polynomials. When we opt for a higher configuration, the polynomial-size remains the same while the number of coefficients to be processed in parallel increases.

However, an important question remains: *what configuration strikes the best balance between throughput and manufacturing cost?* To address this, we turn to [26]. The authors report that the best manufacturing size for high yield ranges from 40 to 80 mm^2 for 7nm technology, while for 40nm, it ranges from 50 to 150 mm^2. In Figure 10, we present two sets of area consumption results for 28nm and 7nm technologies. The first set corresponds to four REED cores produced as a single monolithic chip, while the second set represents one REED chiplet. The best area ranges are highlighted in blue and pink. As we can see, for both 7nm and 28nm, the configuration 512×128 falls within the best development area range and offers high throughput. The configuration 1024×64 is within the optimum range for 28nm and is close to it for 7nm. Monolithic designs, within the best range, offer 4× to 8× less throughput.

6.3. Comparison with Related Works

The realization of privacy-preserving computation through FHE holds great potential for the entire community, resulting in various acceleration works. Among these, the ASIC designs [21], [34]–[36], [66] have achieved the most promising acceleration results. However, a direct comparison with these works would be unfair as the benchmarks are provided for different parameters $(dnum, L, w, L_{boot})$. Hence, to ensure fairness, we provide results for bootstrapping for $dnum = 3$ in Table 4 utilized in [34]. Next, since we cannot change the word size ($w = 54$) chosen for high precision, we select $L = 23, K = 8$, and $L_{boot} = 17$ accordingly.

We use the amortized bootstrapping time $T_{A.S.}$ [35], [36] metric that calculates the bootstrapping time divided by L_{eff} and packing n. This metric overlooks factors such as area, power, and precision. Higher precision necessitates a larger word size, w (or expensive composite scaling). Thus, we use the EDAP (Energy-Delay-Area product) metric [43] and modify it (EDAP$_w$) to incorporate a linear increase due to word size (discussed in Section 7.1).

Table 4 compares our design's area consumption, performance, and power consumption for the packed bootstrapping operation with existing monolithic works, F1 [21], BTS [36], ARK [35], CraterLake (CLake) [66], and SHARP (SH) [34]. REED achieves 1.9× better performance than the state-of-the-art (SH$_{36}$) while consuming 1.8× less area. Our area consumption is less as the prior works utilize at least half of chip area for on-chip memory. In our case, on-chip memory is not significant, as we utilize HBMs for major storage. We obtain better performance results due to the high throughput

TABLE 4. COMPARISON OF REED 2.5D WITH STATE-OF-THE-ART

Work	Area (mm^2)	$T_{A.S.}$ (ns)	P_{Avg} (W)	$EDAP_w$ (/M)	Parameters ($N/L/dnum$)
$F1_{32}$	71.02†	470	28.5†	754.5	$2^{16}/23/24$
BTS_{64}	373.6	45.4	163.2	106.0	$2^{17}/39/2$
ARK_{64}	418.3	14.3	135	9.74	$2^{16}/23/4$
$CLake_{28}$	222.7†	17.6	124†	16.5	$2^{16}/60/1-3$
SH_{36}	178.8	12.8	94.7	4.1	$2^{16}/35/3$
SH_{64}	325.4	11.7	187	7.0	$2^{16}/23/3$
$REED^{‡(1)}_{54}$	**96.7**	**6.6**	**48.2**	**0.20**	$2^{16}/23/3$
		28.8	**49.4**	**3.96**	$2^{16}/30/31$
$REED^{‡(2)}_{54}$	**177**	**14.4**	**83.5**	**3.10**	$2^{16}/30/31$

† Area/power are normalized [34], [54] (14nm/12nm to 7nm).
‡ Result for configuration (1) 1024×64 with one HBM per chiplet, and (2) 512×128 with two HBM per chiplet.

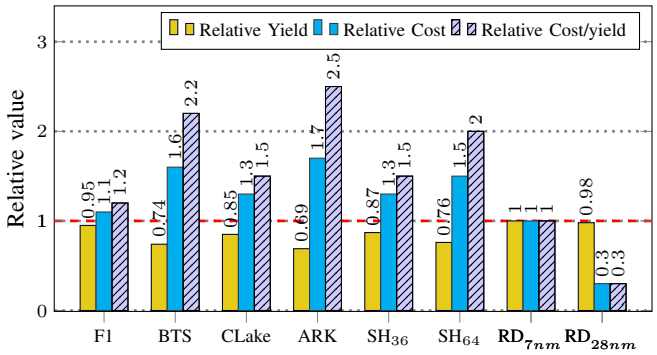

Figure 11. Relative a) yield of existing monolithic designs versus the proposed 7nm chiplet-based architecture [44], b) development cost (including Interposer cost) [26], [47], [52], and c) cost of SiP development (cost/yield). RD refers to our work REED 2.5D.

and $4.8\times$ higher off-chip communication bandwidth offered by four HBM3 blocks – where each chiplet is exclusively connected to one HBM3. Therefore, if the bandwidth and area of the four chiplet-based system is scaled down to that of a monolithic design, it only suffers $1.4\times$ performance loss. This is a trade-off that we face for utilizing a chiplet-based design over a monolithic design, and overall our design offers better performance for low chiplet-area.

We also assess the yield and manufacturing cost in Figure 11, by utilizing the results reported in [26], [44], [52]. For a fair comparison, we use the original area and not the word-size scaled area for prior works. As illustrated, we achieve the highest yield and lowest manufacturing cost on 7nm, resulting in the least overall cost (manufacturing cost/yield), 50% less than state-of-the-art monolithic design $SHARP_{64}$. On 28nm technology, we achieve 85% cheaper design compared to $SHARP_{64}$.

7. Application benchmarks

We benchmark three machine learning applications: linear regression, logistic regression, and a Deep Neural Network (DNN). Each application is evaluated for *encrypted* training and inference. In this setting, the server provides

TABLE 5. APPLICATION BENCHMARK AND THE SPEEDUP ACHIEVED BY REED 2.5D. THE CPU SPEED IS REPORTED ON A 24-CORE, 2×INTEL XEON CPU X5690 @ 3.47GHZ WITH 192GB DDR3 RAM.

Appl.	Accuracy	Op	Time		Speedup
			CPU	HW	
Lin.Reg.	78.12%	Inf.	0.86 s	0.31 ms	2,873×
		Trn.	13.82 s	4.6 ms	2,991×
Log.Reg.	61.8%	Inf.	1.27 s	0.46 ms	2,785×
		Trn.	11.18 s	3.8 ms	2,865×
DNN	95.2%	Inf.	128.7 s	48.6 ms	2,646×
		Trn.	29 days	920 s	2,725×

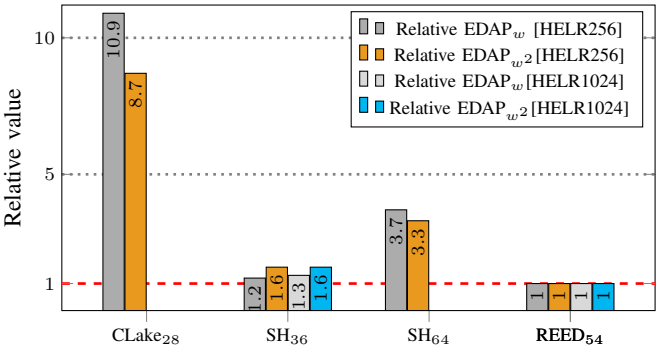

Figure 12. Relative metrics comparison for the HELR [27] application with batch sizes 256 and 1024. Under these metrics, the lower the value, the better.

computational support without knowledge of the data or model parameters, ensuring complete blind computation. Most applications benchmarked in the previous works [21] are partially blind; the server does not see the data but knows the model parameters to evaluate it. To the best of our knowledge, none of the previous works benchmark an encrypted neural network training. The speedup results are presented in Table 5 using the most area conservative design (1024×64). Higher configuration (512×128) will improve the performance by $2\times$.

1) Linear Regression: We employ the Kaggle Insurance dataset [68] to benchmark linear regression. The model uses a batch size of 1204 and 1338 input feature vectors (each containing six features) for training and inference and achieves an accuracy of 78.1% (same as plain model [68]).

2) Logistic Regression: It is a supervised machine learning model that utilizes the log function, evaluated using function approximations in homomorphic context. Its accuracy depends on the degree of approximation function expansion and precision. Existing works, such as [34], [66], utilize the HELR [27] to benchmark encrypted training on MNIST [40] data, with batch sizes (256, 1024). In Figure 12, we illustrate the performance advantage of REED 2.5D. We furthermore evaluate logistic regression on the iDASH2017 cancer dataset (similar to [33]) to predict cancer probability. Here we achieve a training accuracy of 62% in single iteration. This dataset comprises 18 features per input, with batch sizes of 1422 and 1579 used for training and inference.

3) Deep Neural Network : The DNN serves as a pow-

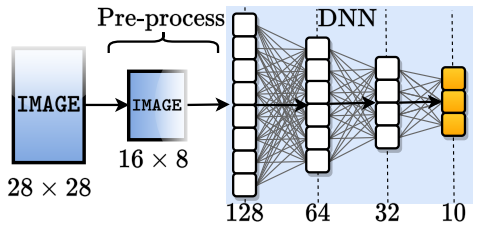

Figure 13. A DNN for MNIST [40] with two hidden and one output layers.

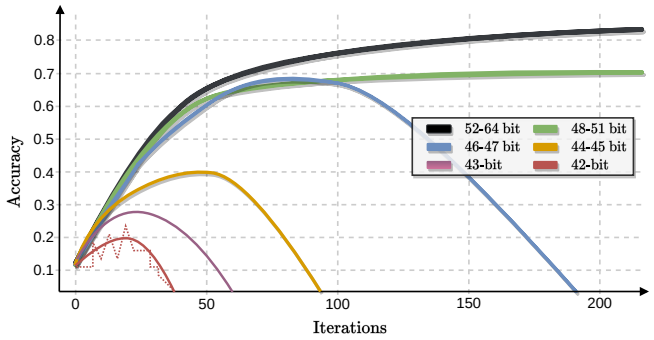

Figure 14. Accuracy plot of different word sizes for the DNN. The lines are smoothened and the red dotted zig-zag line resembles the original form.

erful tool for Deep Learning. In our study, we employ a DNN for the MNIST dataset [40], with two hidden and one output layer (shown in Figure 13). We pack four pre-processed images per batch to prevent overflow during matrix multiplication. DNN training requires 12,500 batches. Thus, all the existing works [34]–[36], [66] not providing computation-communication parallelism will suffer as their on-chip memory is insufficient. The DNN is trained for ≈7000 (≈5.8 Bootstrappings per iteration) iterations and achieves 95.2% accuracy in 29 days using OpenFHE [3]. REED 2.5D could finish this in only 15.4 minutes. This is where our computation-communication parallelism shines, as a huge amount of ciphertexts are required for such an application. None of the works in literature offers this and is bound to suffer for memory-intensive applications.

7.1. Precision-loss Experimental Study

Another facet of privacy-preserving computation is precision loss. Since the server cannot see the intermediate or final results, the best it can do is to ensure that the parameters it operates on support higher precision. To validate our parameter sets, we ran experiments for the DNN training. In Figure 14, we can see how quickly the training accuracy drops as the word size is reduced. Thus, precision plays a vital role in providing privacy-preserving computation on the cloud. Our choice of 54-bit word size strikes the perfect balance between precision and performance. Works offering a smaller word-size [21], [34], [66] require in-depth study to mitigate the accuracy loss due to low precision.

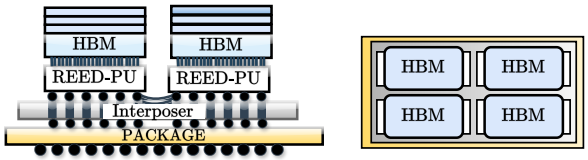

Figure 15. The side and top view of futuristic RE3D has four REED 3DIC chiplets.

8. Future Scope: Journey from 2.5D to 3

The extension of REED 2.5D to a complete 3D IC holds immense potential for future computing. To achieve this transition, we have two options: connecting the PU with the HBM controller via TSV (as shown in Figure 15) or merging the PU unit with the lower HBM controller die. Since HBM is sold as an IP, the latter approach relies on the IP vendors to integrate the PU. By adopting either of these approaches, we can significantly reduce the reliance on the Network-on-Chip (NoC), leading to a compact chip design with lower power consumption. Each chiplet will be a full 3D IC package (PU and Memory) and will need a C2C link via interposer for connecting to other chiplets. A reduction in the area is expected due to fewer HBM stacks on the lateral area and the integration of the REED-PU unit with the HBM controller. Additionally, decreased critical paths would further enhance the design's performance. Thus, the REED's 3D IC integration promises a huge reduction in overall chip area and power consumption.

9. Conclusion

FHE has garnered considerable interest due to its capability to preserve computation privacy. Consequently, numerous efforts have been dedicated to accelerating FHE; however, many of these attempts tend to focus excessively on acceleration at the expense of practicality. Our proposed accelerator design, REED, effectively addresses this limitation. We propose a scalable design methodology that can be easily extended to larger configurations while adapting to constrained environments. It is implemented using a chiplet-based technique, which enables easy practical realization.

Chiplet-based designs face several inherent disadvantages, such as increased latency costs due to slow C2C communication. REED not only mitigates these but also shows advantages over the monolithic designs in terms of performance, area, as well as energy consumption. REED achieves this feat by utilizing non-trivial yet uncomplicated design decisions and a modular design approach. Since all the chiplets are small and identical, we could also prototype the building blocks using FPGA and validate the functionality, which is not done by prior works. This paves the way for interesting future prospects such as formal verification. Overall, the advancements presented in this work hold the promise of advancing privacy-preserving computations and wider adoption of fully homomorphic encryption.

Acknowledgement

This work was supported in part by Samsung Electronics co. ltd., Samsung Advanced Institute of Technology and the State Government of Styria, Austria – Department Zukunftsfonds Steiermark. We also extend our gratitude to Ian Khodachenko for his assistance in conducting the application benchmarking process.

References

[1] "For the first time, ucie shares bandwidth speeds between chiplets," 2023. [Online]. Available: https://www.hpcwire.com/2023/06/07/for-the-first-time-ucie-shares-bandwidth-speeds-between-chiplets/

[2] "How universal chiplet interconnect express changes soc design," 2023.

[3] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "OpenFHE: Open-Source Fully Homomorphic Encryption Library," in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, ser. WAHC'22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 53–63. [Online]. Available: https://doi.org/10.1145/3560827.3563379

[4] AMD, "Amd instinct™ mi300 series accelerators," Tech. Rep., 2023. [Online]. Available: https://www.amd.com/en/products/accelerators/instinct/mi300.html

[5] A. A. Badawi, L. Hoang, C. F. Mun, K. Laine, and K. M. M. Aung, "Privft: Private and fast text classification with homomorphic encryption," *IEEE Access*, vol. 8, p. 226544–226556, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.3045465

[6] M. V. Beirendonck, J. D'Anvers, F. Turan, and I. Verbauwhede, "FPT: A fixed-point accelerator for torus fully homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 741–755. [Online]. Available: https://doi.org/10.1145/3576915.3623159

[7] S. Bian, Z. Zhang, H. Pan, R. Mao, Z. Zhao, Y. Jin, and Z. Guan, "HE3DB: an efficient and elastic encrypted database via arithmetic-and-logic fully homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 2930–2944. [Online]. Available: https://doi.org/10.1145/3576915.3616608

[8] J. Bossuat, C. Mouchet, J. R. Troncoso-Pastoriza, and J. Hubaux, "Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys," in *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Canteaut and F. Standaert, Eds., vol. 12696. Springer, 2021, pp. 587–617. [Online]. Available: https://doi.org/10.1007/978-3-030-77870-5_21

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *Electron. Colloquium Comput. Complex.*, p. 111, 2011. [Online]. Available: https://eccc.weizmann.ac.il/report/2011/111

[10] K. Chae, J. Park, J. Song, B. Koo, J. Oh, S. Yi, W. Lee, D. Kim, T. Yeo, K. Kang, S. Park, E. Kim, S. Jung, S. Park, S. Park, M. Noh, H. Rhew, and J. Shin, "A 4nm 1.15TB/s HBM3 Interface with Resistor-Tuned Offset-Calibration and In-Situ Margin-Detection," in *IEEE International Solid- State Circuits Conference, ISSCC 2023, San Francisco, CA, USA, February 19-23, 2023*. IEEE, 2023, pp. 406–407. [Online]. Available: https://doi.org/10.1109/ISSCC42615.2023.10067736

[11] H. Chen, I. Chillotti, and Y. Song, "Improved Bootstrapping for Approximate Homomorphic Encryption," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, Y. Ishai and V. Rijmen, Eds., vol. 11477. Springer, 2019, pp. 34–54. [Online]. Available: https://doi.org/10.1007/978-3-030-17656-3_2

[12] K. Chen, S. Lin, W. Shen, and A. Wu, "A scalable built-in self-recovery (BISR) VLSI architecture and design methodology for 2d-mesh based on-chip networks," *Des. Autom. Embed. Syst.*, vol. 15, no. 2, pp. 111–132, 2011. [Online]. Available: https://doi.org/10.1007/s10617-011-9074-6

[13] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for Approximate Homomorphic Encryption," in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, ser. Lecture Notes in Computer Science, J. B. Nielsen and V. Rijmen, Eds., vol. 10820. Springer, 2018, pp. 360–384. [Online]. Available: https://doi.org/10.1007/978-3-319-78381-9_14

[14] ——, "A full RNS variant of approximate homomorphic encryption," in *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, ser. Lecture Notes in Computer Science, C. Cid and M. J. J. Jr., Eds., vol. 11349. Springer, 2018, pp. 347–368. [Online]. Available: https://doi.org/10.1007/978-3-030-10970-7_16

[15] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, ser. Lecture Notes in Computer Science, T. Takagi and T. Peyrin, Eds., vol. 10624. Springer, 2017, pp. 409–437. [Online]. Available: https://doi.org/10.1007/978-3-319-70694-8_15

[16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[17] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002626921630026X

[18] A. C. Elster and T. A. Haugdahl, "Nvidia Hopper GPU and Grace CPU Highlights," *Computing in Science & Engineering*, vol. 24, no. 2, pp. 95–100, 2022.

[19] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, p. 144, 2012. [Online]. Available: http://eprint.iacr.org/2012/144

[20] S. Fan, Z. Wang, W. Xu, R. Hou, D. Meng, and M. Zhang, "Tensorfhe: Achieving practical computation on encrypted data using GPGPU," in *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2023, Montreal, QC, Canada, February 25 - March 1, 2023*. IEEE, 2023, pp. 922–934. [Online]. Available: https://doi.org/10.1109/HPCA56546.2023.10071017

[21] A. Feldmann, N. Samardzic, A. Krastev, S. Devadas, R. Dreslinski, K. Eldefrawy, N. Genise, C. Peikert, and D. Sanchez, "F1: A fast and programmable accelerator for fully homomorphic encryption (extended version)," 2021.

[22] H. L. Garner, "The Residue Number System," *IRE Trans. Electron. Comput.*, vol. 8, no. 2, pp. 140–147, 1959. [Online]. Available: https://doi.org/10.1109/TEC.1959.5219515

[23] R. Geelen, M. V. Beirendonck, H. V. L. Pereira, B. Huffman, T. McAuley, B. Selfridge, D. Wagner, G. D. Dimou, I. Verbauwhede, F. Vercauteren, and D. W. Archer, "BASALISC: programmable hardware accelerator for BGV fully homomorphic encryption," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 4, pp. 32–57, 2023. [Online]. Available: https://doi.org/10.46586/tches.v2023.i4.32-57

[24] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, USA, 2009. [Online]. Available: https://searchworks.stanford.edu/view/8493082

[25] J. L. Gonzalez, "Heterogeneous Integration of Chiplets Using Socketed Platforms, Off-Chip Flexible Interconnects, and Self-Alignment Technologies," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, USA, 2021. [Online]. Available: https://hdl.handle.net/1853/64749

[26] A. Graening, S. Pal, and P. Gupta, "Chiplets: How Small is too Small?" *ACM/IEEE Design Automation Conference (DAC),*, 2023.

[27] K. Han, S. Hong, J. H. Cheon, and D. Park, "Logistic regression on homomorphic encrypted data at scale," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 9466–9471. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33019466

[28] IBM, "IBM Cost of a Data Breach 2022 – Highlights for Cloud Security Professionals," *Technical Report*, 2020.

[29] Intel, "Intel xeon cpu max 9400," Tech. Rep., 2023. [Online]. Available: https://geizhals.at/intel-xeon-cpu-max-9400-sockel-4677-v125548.html

[30] JEDEC, "High Bandwidth Memory DRAM (HBM3)," *Tech. Rep. JESD238*, 2022.

[31] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, p. 114–148, Aug. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9062

[32] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, K. Lee, C. Min, J. H. Cheon, and J. H. Ahn, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," *IEEE Access*, vol. 9, pp. 98 772–98 789, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3096189

[33] A. Kim, Y. Song, M. Kim, K. Lee, and J. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Medical Genomics*, vol. 11, 10 2018.

[34] J. Kim, S. Kim, J. Choi, J. Park, D. Kim, and J. H. Ahn, "SHARP: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption," in *Proceedings of the 50th Annual International Symposium on Computer Architecture, ISCA 2023, Orlando, FL, USA, June 17-21, 2023*, Y. Solihin and M. A. Heinrich, Eds. ACM, 2023, pp. 18:1–18:15. [Online]. Available: https://doi.org/10.1145/3579371.3589053

[35] J. Kim, G. Lee, S. Kim, G. Sohn, J. Kim, M. Rhu, and J. H. Ahn, "ARK: Fully Homomorphic Encryption Accelerator with Runtime Data Generation and Inter-Operation Key Reuse," 2022. [Online]. Available: https://arxiv.org/abs/2205.00922

[36] S. Kim, J. Kim, M. J. Kim, W. Jung, J. Kim, M. Rhu, and J. H. Ahn, "BTS: An Accelerator for Bootstrappable Fully Homomorphic Encryption," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 711–725. [Online]. Available: https://doi.org/10.1145/3470496.3527415

[37] T. Kim, H. Kwak, D. Lee, J. Seo, and Y. Song, "Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 726–740. [Online]. Available: https://doi.org/10.1145/3576915.3623176

[38] ——, "Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 726–740. [Online]. Available: https://doi.org/10.1145/3576915.3623176

[39] G. Krishnan, S. K. Mandal, M. Pannala, C. Chakrabarti, J. Seo, Ü. Y. Ogras, and Y. Cao, "SIAM: chiplet-based scalable in-memory acceleration with mesh for deep neural networks," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, pp. 68:1–68:24, 2021. [Online]. Available: https://doi.org/10.1145/3476999

[40] Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[41] D. Lee, K. S. Lee, Y. Lee, K. W. Kim, J. Kang, J. Lee, and J. H. Chun, "Design considerations of HBM stacked DRAM and the memory architecture extension," in *2015 IEEE Custom Integrated Circuits Conference, CICC 2015, San Jose, CA, USA, September 28-30, 2015*. IEEE, 2015, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CICC.2015.7338357

[42] J. A. Lewis, Z. L. M. Smith, and E. Lostri, "The Hidden Costs of Cybercrime," *Technical Report*, 2020.

[43] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42 2009), December 12-16, 2009, New York, New York, USA*, D. H. Albonesi, M. Martonosi, D. I. August, and J. F. Martínez, Eds. ACM, 2009, pp. 469–480. [Online]. Available: https://doi.org/10.1145/1669112.1669172

[44] X. Ma, Y. Wang, Y. Wang, X. Cai, and Y. Han, "Survey on chiplets: interface, interconnect and integration methodology," *CCF Trans. High Perform. Comput.*, vol. 4, no. 1, pp. 43–52, 2022. [Online]. Available: https://doi.org/10.1007/s42514-022-00093-0

[45] R. A. Mahdavi, H. Ni, D. Linkov, and F. Kerschbaum, "Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 2945–2958. [Online]. Available: https://doi.org/10.1145/3576915.3623095

[46] T. Mann, "Amd was right about chiplets, intel's gelsinger all but says," Tech. Rep., 2022. [Online]. Available: https://www.theregister.com/2022/09/28/intel_chiplets_advanced_packaging/

[47] A. C. Mert, Aikata, S. Kwon, Y. Shin, D. Yoo, Y. Lee, and S. S. Roy, "Medha: Microcoded Hardware Accelerator for computing on Encrypted data," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 1, pp. 463–500, 2023. [Online]. Available: https://doi.org/10.46586/tches.v2023.i1.463-500

[48] A. C. Mert, E. Öztürk, and E. Savaş, "Design and implementation of a fast and scalable ntt-based polynomial multiplier architecture," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 253–260.

[49] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[50] S. Morgan, "McAfee Vastly Underestimates The Cost Of Cybercrime," *Cybersecurity Report*, 2020.

[51] B. Murdock, "What Designers Need to Know About HBM3," *Synopsys*, Accessed on July 11, 2023. [Online]. Available: https://www.synopsys.com/designware-ip/technical-bulletin/hbm3-ip-dwtb.html

[52] MUSE Semiconductor, "TSMC UNIVERSITY FINFET PROGRAM," https://www.musesemi.com/university-finfet-program. Accessed July 27th 2023.

[53] M. Nabeel, D. Soni, M. Ashraf, M. A. Gebremichael, H. Gamil, E. Chielle, R. Karri, M. Sanduleanu, and M. Maniatakos, "CoFHEE: A Co-processor for Fully Homomorphic Encryption Execution," 2022. [Online]. Available: https://arxiv.org/abs/2204.08742

[54] S. Narasimha, B. Jagannathan, A. Ogino, D. Jaeger, B. Greene, C. Sheraw, K. Zhao, B. Haran, U. Kwon, A. K. M. Mahalingam, B. Kannan, B. Morganfeld, J. Dechene, C. Radens, A. Tessier, A. Hassan, H. Narisetty, I. Ahsan, M. Aminpur, C. An, M. Aquilino, A. Arya, R. Augur, N. Baliga, R. Bhelkar, G. Biery, A. Blauberg, N. Borjemscaia, A. Bryant, L. Cao, V. Chauhan, M. Chen, L. Cheng, J. Choo, C. Christiansen, T. Chu, B. Cohen, R. Coleman, D. Conklin, S. Crown, A. da Silva, D. Dechene, G. Derderian, S. Deshpande, G. Dilliway, K. Donegan, M. Eller, Y. Fan, Q. Fang, A. Gassaria, R. Gauthier, S. Ghosh, G. Gifford, T. Gordon, M. Gribelyuk, G. Han, J. Han, K. Han, M. Hasan, J. Higman, J. Holt, L. Hu, L. Huang, C. Huang, T. Hung, Y. Jin, J. Johnson, S. Johnson, V. Joshi, M. Joshi, P. Justison, S. Kalaga, T. Kim, W. Kim, R. Krishnan, B. Krishnan, K. Anil, M. Kumar, J. Lee, R. Lee, J. Lemon, S. Liew, P. Lindo, M. Lingalugari, M. Lipinski, P. Liu, J. Liu, S. Lucarini, W. Ma, E. Maciejewski, S. Madisetti, A. Malinowski, J. Mehta, C. Meng, S. Mitra, C. Montgomery, H. Nayfeh, T. Nigam, G. Northrop, K. Onishi, C. Ordonio, M. Ozbek, R. Pal, S. Parihar, O. Patterson, E. Ramanathan, I. Ramirez, R. Ranjan, J. Sarad, V. Sardesai, S. Saudari, C. Schiller, B. Senapati, C. Serrau, N. Shah, T. Shen, H. Sheng, J. Shepard, Y. Shi, M. Silvestre, D. Singh, Z. Song, J. Sporre, P. Srinivasan, Z. Sun, A. Sutton, R. Sweeney, K. Tabakman, M. Tan, X. Wang, E. Woodard, G. Xu, D. Xu, T. Xuan, Y. Yan, J. Yang, K. Yeap, M. Yu, A. Zainuddin, J. Zeng, K. Zhang, M. Zhao, Y. Zhong, R. Carter, C.-H. Lin, S. Grunow, C. Child, M. Lagus, R. Fox, E. Kaste, G. Gomba, S. Samavedam, P. Agnello, and D. K. Sohn, "A 7nm cmos technology platform for mobile and high performance compute application," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 29.5.1–29.5.4.

[55] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 2018, pp. 399–414. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00035

[56] M. Park, J. Lee, K. Cho, J. H. Park, J. Moon, S. Lee, T. Kim, S. Oh, S. Choi, Y. Choi, H. S. Cho, T. Yun, Y. J. Koo, J. Lee, B. K. Yoon, Y. J. Park, S. Oh, C. K. Lee, S. Lee, H. Kim, Y. Ju, S. Lim, K. Y. Lee, S. Lee, W. S. We, S. Kim, S. M. Yang, K. Lee, I. Kim, Y. Jeon, J. Park, J. C. Yun, S. Kim, D. Lee, S. Oh, J. Shin, Y. Lee, J. Jang, and J. Cho, "A 192-Gb 12-High 896-GB/s HBM3 DRAM With a TSV Auto-Calibration Scheme and Machine-Learning-Based Layout Optimization," *IEEE J. Solid State Circuits*, vol. 58, no. 1, pp. 256–269, 2023. [Online]. Available: https://doi.org/10.1109/JSSC.2022.3193354

[57] G. Paulin, F. Zaruba, S. Mach, M. Eggimann, M. Cavalcante, P. Scheffler, Y. Zhang, T. Fischer, N. Wistoff, L. Bertaccini, T. Benz, L. Colagrande, A. D. Mauro, A. Kurth, S. Riedel, N. Huetter, G. Ottavi, Z. Jiang, B. Muheim, F. K. Gurkaynak, D. Rossi, and L. Benini, " Occamy: A 432-core, Multi-TFLOPs RISC-V-Based 2.5D Chiplet System for Ultra-Efficient (Mini-)Floating-Point Computation ," *PULP Platform, ETH Zurich*, 2023. [Online]. Available: https://pulp-platform.org/occamy/

[58] Rambus, "HBM3 Memory: Break Through to Greater Bandwidth." [Online]. Available: https://go.rambus.com/hbm3-memory-breakthrough-to-greater-bandwidth

[59] B. Reagen, W. Choi, Y. Ko, V. Lee, G.-Y. Wei, H.-H. S. Lee, and D. Brooks, "Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference," 2020. [Online]. Available: https://arxiv.org/abs/2006.00505

[60] M. Reshadi, A. Khademzadeh, A. Reza, and M. Bahmani, "A novel mesh architecture for on-chip networks." [Online]. Available: https://www.design-reuse.com/articles/23347/on-chip-network.html

[61] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "HEAX: an architecture for computing on encrypted data," in *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, J. R. Larus, L. Ceze, and K. Strauss, Eds. ACM, 2020, pp. 1295–1309. [Online]. Available: https://doi.org/10.1145/3373376.3378523

[62] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[63] U. K. Robert Dimond, System Architect ARM, "Keynote: Chiplet standards: A new route to arm-based custom silicon," Tech. Rep., 2024. [Online]. Available: https://www.date-conference.com/node/1750

[64] S. S. Roy, K. Järvinen, F. Vercauteren, V. Dimitrov, and I. Verbauwhede, "Modular hardware architecture for somewhat homomorphic function evaluation," in *Cryptographic Hardware and Embedded Systems - CHES*, 2015.

[65] S. S. Roy, K. Järvinen, J. Vliegen, F. Vercauteren, and I. Verbauwhede, "HEPCloud: An FPGA-based multicore processor for FV somewhat homomorphic function evaluation," *IEEE Transactions on Computers*, 2018.

[66] N. Samardzic, A. Feldmann, A. Krastev, N. Manohar, N. Genise, S. Devadas, K. Eldefrawy, C. Peikert, and D. Sanchez, "CraterLake: A hardware accelerator for efficient unbounded computation on encrypted data," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 173–187. [Online]. Available: https://doi.org/10.1145/3470496.3527393

[67] M. Scott, "A note on the implementation of the number theoretic transform," in *Cryptography and Coding - 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings*. Springer, 2017, pp. 247–258.

[68] ——, "Linear regression - insurance dataset," 2020. [Online]. Available: https://www.kaggle.com/code/kianwee/linear-regression-insurance-dataset

[69] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. R. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, B. Khailany, and S. W. Keckler, "Simba: scaling deep-learning inference with chiplet-based architecture," *Commun. ACM*, vol. 64, no. 6, pp. 107–116, 2021. [Online]. Available: https://doi.org/10.1145/3460227

[70] M. Siegesmund, "Chip memory," Tech. Rep., 2014. [Online]. Available: https://www.sciencedirect.com/topics/engineering/chip-memory

[71] S. Sinha Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, "Fpga-based high-performance parallel architecture for homomorphic computing on encrypted data," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 387–398.

[72] Y. Sun, Y. Yuan, Z. Yu, R. Kuper, I. Jeong, R. Wang, and N. S. Kim, "Demystifying CXL memory with genuine cxl-ready systems and devices," *CoRR*, vol. abs/2303.15375, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.15375

[73] J. Takeshita, D. Reis, T. Gong, M. Niemier, X. S. Hu, and T. Jung, "Algorithmic acceleration of b/fv-like somewhat homomorphic encryption for compute-enabled ram," Cryptology ePrint Archive, Report 2020/1223, 2020, https://ia.cr/2020/1223.

[74] T. Thorolfsson, K. Gonsalves, and P. D. Franzon, "Design automation for a 3DIC FFT processor for synthetic aperture radar: a case study," in *Proceedings of the 46th Design Automation Conference, DAC 2009, San Francisco, CA, USA, July 26-31, 2009*. ACM, 2009, pp. 51–56. [Online]. Available: https://doi.org/10.1145/1629911.1629928

[75] A. Viand, P. Jattke, M. Haller, and A. Hithnawi, "HECO: fully homomorphic encryption compiler," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 4715–4732. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/viand

[76] P. Vivet, E. Guthmuller, Y. Thonnart, G. Pillonnet, G. Moritz, I. Miro-Panades, C. F. Tortolero, J. Durupt, C. Bernard, D. Varreau, J. J. H. Pontes, S. Thuries, D. Coriat, M. Harrand, D. Dutoit, D. Lattard, L. Arnaud, J. Charbonnier, P. Coudrain, A. Garnier, F. Berger, A. Gueugnot, A. Greiner, Q. L. Meunier, A. Farcy, A. Arriordaz, S. Cheramy, and F. Clermidy, "2.3 A 220gops 96-core processor with 6 chiplets 3d-stacked on an active interposer offering 0.6ns/mm latency, 3tb/s/mm^2 inter-chiplet interconnects and 156mw/mm^2@ 82%-peak-efficiency DC-DC converters," in *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*. IEEE, 2020, pp. 46–48. [Online]. Available: https://doi.org/10.1109/ISSCC19947.2020.9062927

[77] W. Wang and X. Huang, "Fpga implementation of a large-number multiplier for fully homomorphic encryption," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 2589–2592.

[78] W. Wang, X. Huang, N. Emmart, and C. Weems, "Vlsi design of a large-number multiplier for fully homomorphic encryption," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1879–1887, 2014.

[79] G. Xin, Y. Zhao, and J. Han, "A multi-layer parallel hardware architecture for homomorphic computation in machine learning," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[80] Z. Ye, R. C. C. Cheung, and K. Huang, "Pipentt: A pipelined number theoretic transform architecture," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, no. 10, pp. 4068–4072, 2022. [Online]. Available: https://doi.org/10.1109/TCSII.2022.3184703

[81] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. D. E. Jerger, and G. H. Loh, "Modular Routing Design for Chiplet-Based Systems," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018*, M. Annavaram, T. M. Pinkston, and B. Falsafi, Eds. IEEE Computer Society, 2018, pp. 726–738. [Online]. Available: https://doi.org/10.1109/ISCA.2018.00066

[82] F. Zaruba, F. Schuiki, and L. Benini, "Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing," *IEEE Micro*, vol. 41, no. 2, pp. 36–42, 2021.

[83] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, and G. Sun, "Pipezk: Accelerating zero-knowledge proof with a pipelined architecture," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 416–428.

Appendix

In Section 3, we examined multiple chiplet configurations and selected four ($\mathtt{r} = 4$) based on long-term utilization, lower power dissipation, and low integration costs. Our design methodology and data/task distribution approach remain adaptable to any desired chiplet configuration and the number of chiplets. In fully connected

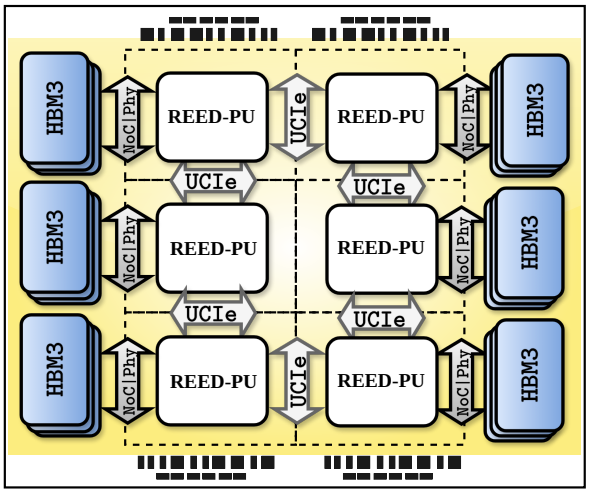

Figure 16. The complete architecture diagram of 6-chiplet REED 2.5D for 1024×64 configuration.

chiplet nodes interconnect length between chiplets can significantly impact energy consumption and latency. Therefore, our proposed non-blocking ring-based communication technique for KeySwitching shows a better advantage for higher chiplet integration density.

Figure 16 illustrates a chiplet-based architecture for six chiplets, which can be expanded to accommodate more chiplets. Additionally, energy-efficient lower-bandwidth memories, such as DDR, can be integrated, with the appropriate (N_1, N_2) configuration based on memory throughput. While increasing the number of chiplets enhances performance, it comes at the cost of area and underutilization when the current multiplicative depth (l) falls below the number of chiplets ($l < r$). However, in the long term, more chiplets can lead to better performance, albeit with the additional area, power, and integration overhead. Also, note that a higher number of Chiplet interconnects implies more additional points of failure, making testability and reliability more involved.

Among all the routines, the KeySwitch is the most expensive operation. For $dnum = L$, we transform all L residue polynomials from slot to coefficient representation (INTT), and then each of these is transformed to $L + 1$ NTTs, multiplied with two key components, and accumulated. This requires L INTTs, $L(L + 1)$ NTTs, and $2L(L + 1)$ MAS, making the naive throughput of this operation: $\frac{f}{L(1+3(L+1))\cdot N_1}$. By utilizing REED's parallel processing capability to perform all MAS operations concurrent to the NTT operations (shown in Figure 7), we save $2L(L+1)$ clock cycles and increase the throughput to $\frac{f}{L(L+3)\cdot N_1}$, resulting in a 66.7% improvement.

[35] utilizes four PUs and states that for the base-conversion step, the number of polynomials that need to be transferred during limb-based only decomposition is $2 \cdot dnum \cdot (L + K + 1)$, while for coefficient-wise it is $(dnum + 2) \cdot (L + K + 1)$.

This assumes the limb-wise distribution is done after

multiplication with KeySwitch keys, and all the results are sent to every PU via an all-to-all broadcast. However, we remark that one PU does not need to send all the polynomials and instead only needs to send $\frac{2\cdot(dnum-1)\cdot(L+K+1)}{dnum}$ polynomials so that every PU holds the results for the supported bases. Therefore, the total cost becomes $2\cdot(dnum-1)\cdot(L+K+1)$, which is less than the cost of coefficient-wise distribution for $dnum = 3$. Furthermore, we would like to note that polynomial distribution after KeySwitch is expensive as the data doubles in size after multiplication with the two key components. Hence, this distribution should be done immediately after NTT computation, which further reduces the cost to only $(dnum-1)\cdot(L+K+1)$. This is much less compared to coefficient-wise distribution $(dnum+2)\cdot(L+K+1)$. Hence, we reassert that limb-based distribution will attain less communication overhead than coefficient-wise without any extra computation overhead.