OFHE: An Electro-Optical Accelerator for Discretized TFHE

Mengxin Zheng[†§] Cheng Chu[†] Qian Lou[§] Nathan Youngblood[*] Mo Li[‡] Sajjad Moazeni[‡] Lei Jiang[†]
[†] Indiana University Bloomington [§] University of Central Florida
[*] University of Pittsburgh [‡] University of Washington
[†] *{zhengme, chu6, jiang60}@iu.edu* [§] *qian.lou@ucf.edu*
[*] *nathan.youngblood@pitt.edu* [‡] *{moli96, smoazeni}@uw.edu*

ABSTRACT

This paper presents *OFHE*, an electro-optical accelerator designed to process Discretized TFHE (DTFHE) operations, which encrypt multi-bit messages and support homomorphic multiplications, lookup table operations and full-domain functional bootstrappings. While DTFHE is more efficient and versatile than other fully homomorphic encryption schemes, it requires 32-, 64-, and 128-bit polynomial multiplications, which can be time-consuming. Existing TFHE accelerators are not easily upgradable to support DTFHE operations due to limited datapaths, a lack of datapath bit-width reconfigurability, and power inefficiencies when processing FFT and inverse FFT (IFFT) kernels. Compared to prior TFHE accelerators, OFHE addresses these challenges by improving the DTFHE operation latency by 8.7%, the DTFHE operation throughput by 57%, and the DTFHE operation throughput per Watt by 94%.

CCS CONCEPTS

• **Hardware → Emerging optical and photonic technologies**;
• **Security and privacy → Cryptography**.

KEYWORDS

electro-optical accelerator, fully homomorphic encryption

ACM Reference Format:
Mengxin Zheng, Cheng Chu, Qian Lou, Nathan Youngblood, Mo Li, Sajjad Moazeni, and Lei Jiang. 2024. OFHE: An Electro-Optical Accelerator for Discretized TFHE. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '24), August 5–7, 2024, Newport Beach, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.xxxx/xxxxxxx.11111108

1 INTRODUCTION

The realm of cryptography has evolved significantly with the advent of *Fully Homomorphic Encryption* (FHE) [4]. Tailored for cloud computing, FHE allows users, like Alice, to send encrypted data as ciphertexts to a server. The server can directly compute on these ciphertexts, ensuring unparalleled data privacy. Once processed, results, still encrypted, are returned to Alice. The server never accesses actual data, underscoring FHE's end-to-end encryption strength. Only Alice, using her secret key, deciphers the results.

Table 1: The comparison between various FHE schemes.

Scheme	Operation Type	Data Type	Bootstrapping	Application
CKKS [5]	×/+	fixed-point	high latency	machine learning
TFHE [6]	logic	binary	low latency	general purpose
DTFHE [7]	×/+/logic/mod	integer	low latency	general purpose

In comparison to other FHE schemes, *Discretized TFHE* (DTFHE) [7] emerges as a proficient FHE scheme, as illustrated in Table 1.

- *DTFHE exhibits versatility in operational types.* Traditional FHE schemes, e.g., CKKS [5], which only support homomorphic additions and multiplications, are potent in machine learning but lack applicability in general-purpose scenarios [15]. While third-generation FHE schemes like TFHE [6] focus on Boolean logic operations, they do not support native homomorphic additions and multiplications. For example, TFHE requires a lengthy sequence of logic gates to implement a multi-bit multiplication. In contrast, DTFHE enhances its capabilities to support homomorphic operations including Boolean algebra, modular arithmetic, and native multi-bit additions and multiplications. Compared to TFHE running on the same CPU, DTFHE reduces the latency of a 4-bit homomorphic multiplication by 99.5% [9].
- *DTFHE can achieve fast bootstrapping operations.* In the context of FHE, operations performed on ciphertext inherently introduce noise. Over time, this accumulation of noise can impede accurate decryption. To mitigate this noise accumulation, FHE schemes invariably require periodic bootstrapping. Traditional FHE schemes like CKKS, unfortunately, suffer from prolonged bootstrapping durations, sometimes extending to several hundred seconds [5]. However, akin to TFHE, DTFHE boasts rapid bootstrapping for binary messages. Nonetheless, for multi-bit integer messages, DTFHE's bootstrapping may require $\sim 5s$ on a CPU. The expedited bootstrappings render DTFHE capable of implementing various general-purpose applications requiring substantial circuit depth.

Recent advancements in hardware accelerators [13, 18] have been instrumental in expediting the bootstrapping operations of TFHE. Nevertheless, the domain of accelerating DTFHE—distinctive for its encryption of multi-bit messages within a singular ciphertext—remains largely unexplored. The challenge of adapting pre-existing TFHE accelerators to be compatible with DTFHE arises from two fundamental issues.

- The initial concern is the static nature of the datapaths in earlier TFHE accelerators, either 32 [13] or 64 [18] bits, which are ill-equipped to accommodate the versatile computational demands intrinsic to DTFHE. As an illustration, DTFHE's encryption of a singular-bit message can utilize a 32-bit datapath. However, for the homomorphic LUT operations on ciphertexts that encrypt messages up to 4 bits, a 64-bit datapath becomes imperative.

Extending further, the homomorphic multiplications in DTFHE for ciphertexts with messages of 5 bits or more necessitate 128-bit polynomial multiplications, which subsequently calls for a 128-bit datapath. Regrettably, prior TFHE accelerators neither possess the infrastructure to sustain a 128-bit datapath nor the adaptability to alternate between diverse datapath bit-widths.

- The second impediment revolves around the power efficiency—or lack thereof—in pre-existing TFHE accelerators. Their DSP [3] or CMOS ASIC [13, 18] platforms for FFT and IFFT ((I)FFT[1]) are notorious energy consumers. Indeed, FFT and IFFT kernel processing is responsible for a ∼ 55% [13, 18] to ∼ 80% [17] of the total energy expenditure in preceding TFHE accelerators.

This paper unveils *OFHE*, an electro-optical accelerator designed for DTFHE, with a focus on accelerating homomorphic multiplications and full-domain functional bootstrappings. The cornerstone of OFHE is a photonic FFT engine, engineered to speed up (I)FFT kernels across various scales and precisions integral to DTFHE operations, with high power efficiency. The other kernels integral to these operations are executed via CMOS modules.

- **A Chiplet Package Design**: At the heart of OFHE lies a photonic FFT engine adept at executing 64-point FFT functions. Leveraging passive photonic devices, the FFT engine's fabrication is optimized on dedicated chips. In parallel, CMOS modules find residence on a distinct CMOS chip. Each of these chips, whether photonic or CMOS, is organized as a individual chiplet, interconnected through photonic I/O links.
- **Adaptable DTFHE Parameter Support**: OFHE's hallmark is its adaptability, accommodating a plethora of DTFHE parameters. Avoiding the slow runtime photonic device reconfiguration, OFHE incorporates a forward FFT engine, enriched with conjugating operations, tailored for IFFT kernels. Furthermore, a bit-level pipeline crafts a versatile 32-, 64-, or 128-bit (I)FFT datapath. A novel electro-optical computing flow segments a 1K-, 2K-, or 4K-point FFT kernel into several 64-point FFT kernels, which can be natively supported by the photonic FFT engine.
- **Higher Throughput and Power Efficacy**: Experimental results show OFHE enhances the DTFHE operation latency by 8.7%, the DTFHE operation throughput by 57% and the throughput per Watt by 94%, compared to prior TFHE accelerators.

Algorithm 1: Various TFHE Bootstrappings.

```
   Input: a TLWE sample $c = (\mathbf{a}, b) \in \mathrm{TLWE}_s(\frac{m}{B})$, $m \in \mathbb{Z}_B$; a LUT
       $\mathrm{L} = [l_0, \ldots, l_{B-1}] \in \mathbb{Z}_B^B$; a bootstrap. key $BK_i \in \mathrm{TRGSW}_S(s_i)$, $i \in [\![1, n]\!]$
   Output: $c' \in \mathrm{TLWE}_S(\frac{\mathrm{L}[m]}{B})$, $S \in \mathbb{B}^N$
1  Function FullDomainBootstrap($(\mathbf{a}, b)$, L, BK):
2      $tv \leftarrow \sum_{i=0}^{\frac{B}{2}-1} \sum_{j=0}^{1} \sum_{k=0}^{\frac{N}{B}-1} \frac{1}{B} l_{\frac{jB}{2}+i} X^{(2i+j)\frac{N}{B}+k}$; $pa \leftarrow \frac{B+1}{4B}$
3      $c_{sign} \leftarrow$ FunctionalBootstrap($c$,[$pa, \ldots, pa$], BK) $-pa$
4      return FunctionalBootstrap($c + c_{sign}$, $tv$, BK)
5  Function FunctionalBootstrap($(\mathbf{a}, b)$, L, BK):
6      $b \leftarrow \lceil 2Nb \rfloor$ and $a_i \leftarrow 2Na_i \in \mathbb{Z}_{2N}$, $i \in [\![1, n]\!]$
7      $tv \leftarrow \sum_{i=0}^{N-1} \frac{1}{2B} \cdot l_{\lfloor \frac{iB}{N} \rfloor} X^i \in \mathbb{T}_N[X]$
8      $ACC \leftarrow$ BlindRotate($(0, tv)$, $(\mathbf{a}, b + \frac{1}{4B})$, BK))
9      return SampleExtract($ACC$)
10 Function BlindRotate($(\mathbf{a}, b)$, $tv$, BK):
11     $ACC \leftarrow X^{-\lceil b2N \rfloor} \cdot tv$
12     for $i = 1$ to $n$ do
13         $ACC \leftarrow BK_i \cdot (ACC - X^{\lceil a_i 2N \rfloor} \cdot ACC) + ACC$
14     return $ACC$
```

2 BACKGROUND

2.1 Discretized TFHE (DTFHE) Basics

Notations. In this paper, (1) boldface signifies vectors; (2) superscripts express the element count in vectors; (3) modulus is indicated by subscripts. (4) $\mathbb{S}_q^n$ denotes the set of n-element vectors in $\mathbb{S}$ modulo q. (5) $\mathbb{S}[X]$ represents the set of polynomials over variable X with $\mathbb{S}$ coefficients. (6) Power-of-2 cyclotomic polynomial's modulo is portrayed by its degree. (7) Vectors of polynomials over X with modulus $\Phi 2N(X) = X^N + 1$ and coefficients in $\mathbb{S}$ modulo q are represented by $\mathbb{S}_q[X]_N^n$. (8) $\mathbb{M}$ stands for the $\mathfrak{R}$-module.

Binary-Secret Scale-Invariant LWE. FHE foundations lie in the Learning With Errors (LWE) problem. An LWE sample encompasses a duo $(\mathbf{a}, b) \in \mathbb{M}^{n+1}$, with $\mathbf{a}$ uniformly drawn from $\mathbb{M}^n$, $b = \langle \mathbf{a}, \mathbf{s} \rangle + e \in \mathbb{M}$, and $n \geq 1 \in \mathbb{Z}$. The binary secret key $\mathbf{s}$ is uniformly selected from $\mathfrak{B}^n$. Meanwhile, error e arises from a Gaussian distribution over $\mathbb{M}$, centralized at 0 with standard deviation σ.

Encryption and Decryption. The idea of a LWE-based cryptosystem is to encrypt a message by adding the message to the b part of an LWE sample, since it is indistinguishable from a vector sampled from the uniform distribution. TFHE encrypts 1-bit messages in both TLWE and TRLWE samples [7]. Both are a type of LWE samples, differing by the definition of $\mathbb{M}$ and $\mathfrak{B}$.

DTFHE. DTFHE [7] is proposed to encrypt a multi-bit message in a T(R)LWE sample. $c = (\mathbf{a}, b) \in \mathrm{TLWE}(\frac{m}{2B})$ is a TLWE sample, where m indicates a multi-bit message, B denotes a discretization parameter, and only a half of the torus is used. DTFHE encodes the message by $m = \sum_{i=0}^{\lfloor \log_2 m \rfloor} 2^i \tilde{m}_i$, where $\tilde{\mathbf{m}}$ is the binary vector representation of m. Messages are mapped to integers by discretizing the torus. B specifies the base in which messages are decomposed when working with messages encrypted in multiple samples.

2.2 DTFHE Operations and Implementations

Gate Bootstrapping. At the end of a two single-bit inputs TFHE gate, a gate bootstrapping is required to remove the noises accumulated in the ciphertext. A gate bootstrapping can be summarized in three steps: (1) setting the accumulator vector, *ACC*, to be $\sum_{i=0}^{N} \frac{1}{4} X^i \in \mathbb{T}_N[X]$; (2) using *BlindRotate* to compute $ACC \cdot X^{-\phi(c)2N} \bmod \Phi_{2N}$; and (3) using *SampleExtract* to extract the constant term of the rotated *ACC*. Particularly, the most important step, *BlindRotate*, in a gate bootstrapping is shown in Line 10 of Algorithm 1, where a blind rotation of *ACC* by $-\phi(c)$ is performed.

Functional Bootstrapping. Besides removing noises, a functional bootstrapping [18] also performs a homomorphic lookup table (LUT) through replacing the regular test vector by an encoded LUT, as shown in Line 5 of Algorithm 1. A functional bootstrapping discretizes the domain of a function, evaluates the function in all discretized points, and then stores the results in a LUT. The LUT is encoded as a polynomial (Line 7, where B is a discretization parameter). Due to the negacyclic property of *BlindRotate*, the function supported by functional bootstrapping has to be anti-symmetric, i.e., $f(x + N) = -f(x)$. For an arbitrary function, the negacyclic property can be avoided using only the first half of the torus.

[1]Henceforth, we will use (I)FFT to refer to both FFT and IFFT.

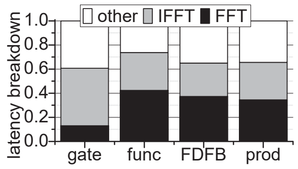

Figure 1: The latency breakdown of DTFHE operations (i.e., gate, functional, full-domain bootstrappings, and TRLWE tensor product).

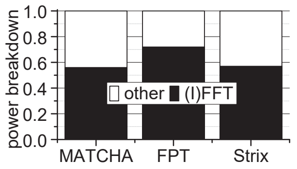

Figure 2: The power breakdown of prior TFHE hardware accelerators including MATCHA [13], FPT [3] and Strix [18].

Full-Domain Functional Bootstrapping (FDFB). Besides using only the first half of the torus, a functional bootstrapping cannot perform homomorphic modular arithmetic. FDFB [9] shown in Line 1 of Algorithm 1 is created to work over the entire torus and to support homomorphic modular arithmetic. A FDFB combines two functional bootstrappings in a chaining way, which guarantees the lowest output error variance.

TRLWE Tensor Product and TLWE multiplication. DTFHE supports BFV-like TRLWE tensor product operations [7] by native TFHE parameters. A integer parameter q controls the precision of the TRLWE result. A TRLWE tensor product also requires a relinearization key to reduce the number of terms in the TRLWE result to 2. Via a TRLWE tensor product, DTFHE can support a homomorphic multiplication between two TLWE samples. When implementing an m-bit $\times$ n-bit integer homomorphic multiplication, a DTFHE TRLWE tensor product is faster than $O(mn)$ TFHE gates.

Torus Implementation. In order to map and rescale all elements from $\mathbb{T}$ to $\mathbb{Z}_q$, an integer precision parameter $q \in \mathbb{N}$ is used when implementing a torus. For TFHE, which encrypts 1-bit messages and works with two single-bit inputs logic gates, $q = 2^{32}$ [7] is precise enough. However, DTFHE requires a larger plaintext space to encrypt multiple bits, perform homomorphic multiplications, and support full-domain functional bootstrappings. So DTFHE uses $q = 2^{64}$ [7] to map all torus elements to 64-bit long integers.

(I)FFT. Homomorphic multiplications and various types of bootstrappings in DTFHE require numerous polynomial multiplications [7]. To reduce the computational complexity of polynomial multiplication from $O(N^2)$ to $O(N \log N)$, where N is the degree of the polynomials, DTFHE uses (I)FFT. As shown in Figure 1, (I)FFT kernels consume 64% ∼ 75% [13] of the latency of various DTFHE operations running on a CPU. Our experimental methodology is explained in Section4. When the integer precision parameter q is 2^{32}, 64-bit double-precision floating-point (I)FFT kernels can be directly used without introducing significant errors. However, if $q = 2^{64}$, particularly when the numeric base B in Algorithm 1 is large, polynomial multiplications of DTFHE require up to 128-bit precision. Therefore, double-precision floating-point (I)FFT kernels are not precise enough for various DTFHE operations [9].

2.3 Related Work and Motivation

Related Work. FHE is renowned for its security but often criticized for its high computational overheads. To address these challenges, ASIC-based accelerators [14, 20] have been developed for FHE schemes such as CKKS and BGV, which support only homomorphic multiplications and additions. Additionally, several accelerators [13, 18] have emerged, focusing on bootstrapping functions within single-bit TFHE ciphertexts. However, no accelerator adequately supports DTFHE encrypting multi-bit messages. Although prior work [20] leverages photonic microdisks to design 512-bit electro-optical adders and multipliers, enhancing BGV's number-theoretic transform (NTT) processes, no existing work except OFHE uses analog photonic signals to accelerate (I)FFT kernels for DTFHE.

Motivation. Existing TFHE accelerators have limitations when processing DTFHE operations. Two constraints are observed:

- **Limited Datapath Scalability**: Efforts to retrofit current TFHE accelerators [13, 18] for DTFHE encounter several barriers. Intrinsic operations of DTFHE, like full-domain functional bootstrappings, TRLWE tensor products, and TLWE multiplications, necessitate 128-bit polynomial computations. Most prior accelerators offer datapaths confined to 32 bits, thus remaining incompatible with highly precise (I)FFT kernels or 128-bit polynomial computations. While DTFHE's single-bit message encryption parallels TFHE, demanding a 32-bit datapath, multi-bit (e.g., 4-bit) encryption in DTFHE necessitates a 64-bit one. To adeptly manage these variable requirements, accelerators must be equipped with flexible data paths of 32, 64, and 128 bits, a feature conspicuously absent in current TFHE accelerators.
- **Power Efficiency Concerns**. Predominantly prior TFHE accelerators [3, 13, 18] are notorious power guzzlers, with consumption often breaching 40 ∼ 100 Watts. In these accelerators, (I)FFT kernels notably surge the power demands, accounting for nearly 56% to 80% of their total consumption, as depicted in Figure 2.

3 OFHE

Architecture. This paper presents OFHE, a photonic accelerator designed for efficient DTFHE operations. OFHE comprises multiple photonic FFT chips and a CMOS chip, as depicted in Figure 3(a). The components are structured as chiplets interconnected through photonic I/O links. The CMOS chip initializes and calibrates all photonic components. When an FFT or IFFT kernel is invoked, the CMOS chip transmits input data to a photonic chip using an I/O link. Digital-to-Analog Converters (DACs) transform this data into optical signals, which are processed by the photonic chip. The resultant signals are then sampled and digitized into final outputs through Analog-to-Digital Converters (ADCs).

Photonic FFT. Unitary FFT operations can be performed using optical devices [11]: $\mathbf{X}_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathbf{x}_n e^{-i\frac{2\pi kn}{N}}$, where $k \in [\![0, N-1]\!]$, $\mathbf{x}_n$ represents an input sampling point, and $\mathbf{X}_k$ denotes an output point. A 4-point Photonic FFT Engine (PFFTE) is shown in Figure 3(b), where a butterfly unit is a 2-point unitary FFT substrate. The butterfly of the 2-point FFT $\begin{pmatrix} out_0 \\ out_1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} in_0 + in_1 \\ in_0 - in_1 \end{pmatrix}$ can be described by $\underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -j \end{pmatrix}}_{\text{output phase shifter}} \underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & j \\ j & 1 \end{pmatrix}}_{\text{coupler}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -j \end{pmatrix}}_{\text{input phase shifter}} \begin{pmatrix} in_0 \\ in_1 \end{pmatrix}$, where in_0 and in_1 are inputs, and out_0 and out_1 are outputs. The first and third matrices are implemented by input and output phase shifters, respectively, while the second matrix is performed by a 2×2 directional coupler. All components of the 2-point PFFTE are passive optical devices, and therefore consume little power.

Photonic Chip. On a photonic chip, we use ITO plasmonic MZ modulators [1] as amplitude and phase modulators, and Michelson

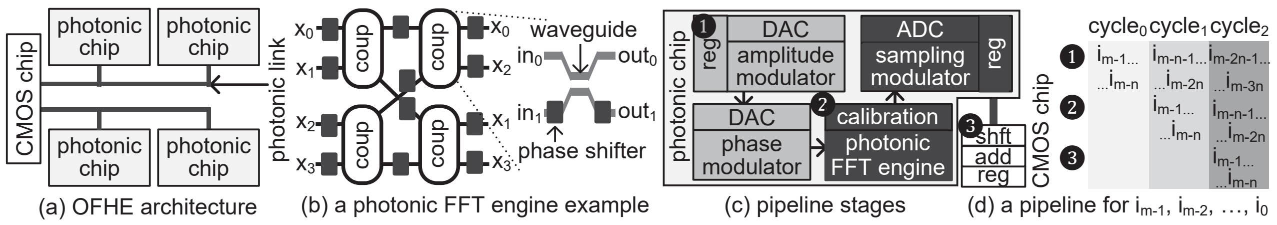

Figure 3: The components and pipeline of OFHE, when processing m-bit inputs $i_{m-1}, i_{m-2}, \ldots, i_0$.

modulators as sampling modulators. To construct a PFFTE, various compact and low-loss photonic components are employed, including 2×2 directional couplers [19], phase shifters [10], Y-splitters and combiners, and straight and spiral [12] waveguides. The input/output interfaces of the PFFTE are implemented using grating couplers. The length of the spiral waveguide of a phase shifter in the i_{th} FFT stage is calculated as $T/2^i$, where $1 \le i \le \log_2(N)$, N is the number of FFT inputs, and T denotes the physical length of time delay. T can be computed as $c/(n_{eff} * freq)$, where c is the speed of light, n_{eff} is the effective index, and $freq$ is the operating frequency of the PFFTE. We set $freq$ to 12GHz [11], resulting in $T = 10$ mm.

The Input Scale of a PFFTE. As the integration of additional photonic components into a PFFTE escalates, there is an inevitable rise in total optical loss. This uptick demands compensation via heightened input power [11]. Consequently, to avert disproportionate power usage, moderating the input count (N) becomes paramount. High values of N accentuate the optical losses associated with spiral waveguides [11], making them the principal contributors to the PFFTE's optical loss. The extinction ratio of a PFFTE, expressed as the ratio P_{max}/P_{min} (with P_{max} and P_{min} being the peak and nadir of PFFTE output powers respectively), diminishes significantly with ascending input numbers, as depicted in Figure 4, due to the increased necessity for spiral waveguides. For precise sampling by modulators, maintaining a robust PFFTE output power is vital, hence a restricted input count for the PFFTE is advocated. Our design focuses on a 64-point PFFTE.

Fixed Point Representation. In TFHE and DTFHE, elements from $\mathbb{T}$ are mapped to $\mathbb{Z}_q$, with q taking values 2^{32} or 2^{64}. Previous TFHE accelerators, when $q = 2^{32}$, leveraged 38- [13] and 30-bit fixed-point results within (I)FFT cores, deviating from the conventional double floating-point computations. The 30-bit depiction as per Figure 5 divides into 19-, 14-, and 6-bit fractions for respective bootstrapping-key, FFT, and IFFT computations, thereby ensuring consistent gate bootstrapping activities. Contrarily, earlier DTFHE CPU executions employed the Karatsuba algorithm [9] since achieving quadruple-precision floating-point (I)FFTs on CPUs posed challenges. OFHE, aiming for minimized hardware expenditure, translates the intermediary outcomes and twiddle factors of (I)FFT cores in different DTFHE operations into fixed-point representations. Adopting this scheme, OFHE cautiously utilizes 64-bit and 128-bit fixed-point representations for $q = 2^{32}$ and $q = 2^{64}$, respectively, dividing them into 29-, 26-, and 17-bit fractions.

FFT Pipeline with a Reconfigurable Datapath. DACs and ADCs are limited by their precision, making it infeasible for a PFFTE to process integer inputs of m-bit in a singular operation, with m being 32, 64, or 128 bits. Thus, OFHE segments the FFT operation into a tri-stage pipeline, as depicted in Figure 3(c). In stage ❶,

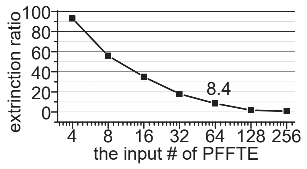

Figure 4: PFFTE extinction ratio with varying input #.

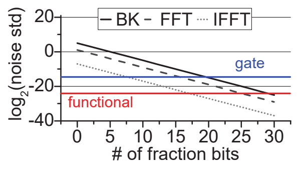

Figure 5: Fraction bit # vs output approx. noise.

DACs extract n_{in} bits from the FFT input register, converting these to photonic signals transmitted to the PFFTE. Here, n_{in} specifies the DAC's bit-width. In stage ❷, the PFFTE's output signals are relayed to sampling modulators. ADCs then transform these voltage outputs into n_{out}-bit digital FFT outcomes, subsequently stored in a register, where n_{out} denotes the ADC's bit-width. In stage ❸, a shifting and addition procedure on the CMOS chip compiles each n_{out}-bit FFT outcome into an m-bit conclusive result. An example pipeline, initialized from input's most significant bits, is portrayed in Figure 3(d). Achieving a stable pipeline kernel necessitates three cycles from the PFFTE. The FFT pipeline's bit-width adaptability is facilitated by varying m values, with its operational frequency tethered to the ADCs' and DACs' speeds.

Fast IFFT. Although it is possible for the PFFTE to perform an IFFT operation through reconfiguring phase shifters, this approach has a long latency and requires a large number of DACs. Reconfiguring a phase shifter can take several milliseconds [11], and adjusting the parameters of all necessary phase shifters concurrently for an IFFT operation requires hundreds of DACs. To avoid the expensive run-time phase shifter reconfiguration, OFHE employs the conversion [8] between IFFT and FFT. The conversion is described as $\text{IFFT}(\mathbf{X}_k) = \frac{1}{N}\text{conj}(\text{FFT}(\text{conj}(\mathbf{X}_k)))$, where $k \in [\![0, N-1]\!]$, and conj represents a conjugate operation. In this way, OFHE can perform conjugate operations on the input and output points of an IFFT operation on the CMOS chip and use the same PFFTEs to complete the IFFT operation.

Scaling FFT. To work with different DTFHE parameters with assorted security levels, OFHE needs to execute FFT operations on inputs spanning 1K, 2K, and 4K points. Given that PFFTE inherently supports only 64-point FFT operations, OFHE employs the 4-step FFT algorithm [2]. This algorithm compartmentalizes extensive FFT operations into numerous 64-point FFT tasks. Specifically, for an N-point FFT (where N encompasses 1K, 2K, or 4K), OFHE conducts n_0 64-point FFT operations on the input points structured as an $n_0 \times 64$ matrix, leading to a resultant matrix R_{pq}. Subsequently, this matrix undergoes multiplication by the exponential factor $e^{-2\pi ipq/N}$ and is transposed. Ultimately, OFHE executes 64 n_0-point FFT operations on the resulting matrix via the PFFTE, with necessary padding introduced when n_0 is less than 64.

Table 2: The area and power consumption of OFHE

Name	Component	Spec	Power (mW)	Area (mm^2)
64-point photonic FFT chip	D. coupler	[19], ×192	0	0.0013
	P. shifter	[10], ×448	0	0.014
	spiral	[12], ×192	0	0.06
	splt/comb	×484	0	0.028
	G. coupler	×128	0	0.003
	MZ mod.	[1], ×64	0.121	0.0069
	samp. mod.	×64	4.096	0.0076
	holder	12G, ×64	0.001	0.0001
	1-bit DAC	12G, ×128	2.5	0.0001
	6-bit ADC	[16], 12G, ×64	1280	0.768
	I/O buff.	32KB, ×1	9.92	0.017
	laser	×1	130	0.235
	chplt PHY	1KB	19.2	0.864
sub-total			1.45 Watt	1.99
32nm CMOS chip	I/O buff.	4KB, ×8	9.92	0.017
	core	1.2G, ×8	15,920	13.76
	NoC	8 × 32, 256b	2110	0.44
	SPM	2MB, 32 banks	1510	1.64
	chplt PHY	1KB	19.2	1.728
	mem ctrl	DDR5 PHY	6,400	7.5
sub-total			26.03 Watt	30.66
OFHE	OFFT chip	×8	11.6 Watt	15.92
	CMOS chip	×2	52.06 Watt	61.32
total			**63.66** Watt	**77.2**

CMOS Chip. The CMOS chip, interfacing with PFFTE chips via photonic I/O links, comprises 8 cores that manage computations for DTFHE operations, excluding (I)FFT kernels. A crossbar network connects these cores with a scratchpad memory. One core uses serial adders to collate intermediate results derived from the PFFTEs. Each core is calibrated to support datapaths of 32, 64, and 128 bits, leveraging serial adders and multipliers for polynomial additions and element-wise multiplications. Conjugating units modify IFFT inputs for direct PFFTE processing. Furthermore, a transposition and shuffling mechanism optimizes the data structure for inputs and intermediate (I)FFT kernel results. 1KB input and output registers serve all computational units within a core.

Design Overhead. Table 2 presents the power and area overhead of OFHE. The system comprises 8 PFFTE chips, operating at 12 GHz, and two CMOS chips running at 1.2 GHz.

- **PFFTE chip**. Each PFFTE chip is constructed using $\sim 1.5K$ passive photonic components. The light for a PFFTE is generated by an on-chip laser. The PFFTE guarantees only 6-bit accurate FFT intermediate results, due to the relative intensity noise of our laser. These components are modeled using Lumerical FDTD and INTERCONNECT software tools.
- **CMOS chip**. Each CMOS chip is modeled using Cadence Virtuoso with 32nm PTM technology. Memory parts are modeled using CACTI. Each core has 1K serial adders, 1K serial multipliers, 1K conjugate units, four transpose and shuffling units.

4 EXPERIMENTAL METHODOLOGY

Schemes. OFHE is compared against four prior accelerators as shown in Table 3. The comparison includes FPT, an FPGA-based accelerator [3], as well as two ASIC-based accelerators, MATCHA (MATC) [13] and Strix [18]. Additionally, CryptoLight (Cryp) [20], an electro-optical accelerator originally designed for processing BGV by photonic NTT units, was selected. Since its detailed configurations are not reported in [20], we adapted Cryp to accelerate DTFHE by replacing the CMOS adders and multipliers of Strix with the electro-optical counterparts of Cryp while following the same configuration of Strix. It is noted that all prior TFHE accelerators

Table 3: The comparison of TFHE hardware accelerators.

Schemes	Description	Power (Watt)
CryptoLight [20]	8 photonic units, 21MB on-chip SPM	68.5
MATCHA [13]	ASIC, 8-core 4MB SPM, 8GB HBM2	37
FPT [3]	Alveo U280, 8GB HBM, 16GB DDR4	99
Strix [18]	8-core, 21MB on-chip SPM, HBMe	77.14

support only gate-level functional bootstrappings (GFBs) and are unable to perform BFV-like TLWE multiplications or full-domain functional bootstrappings (FDFBs).

DTFHE Operations and Parameters. To compare OFHE against prior TFHE accelerators, we studied GFBs operating on 32-bit datapaths. We also studied DTFHE-specific FHE operations including BFV-like TLWE multiplications, and FDFBs, which may require 64- or 128-bit datapaths. As Table 4 shows, we studied 4 sets of DTFHE parameters, i.e., I and II typically for GFBs, while III and IV [9] often for TLWE multiplications and FDFBs.

Table 4: The parameters of DTFHE.

Parameters	I	II	III	IV
q of $\mathbb{Z}_q$	2^{32}	2^{32}	2^{64}	2^{128}
Security Level (λ)	128-bit	110-bit	128-bit	128-bit
TLWE Dim. (n)	586	500	632	829
TGLWE Dim. (k)	2	1	1	1
Polynom. Size (N)	512	1024	2048	4096
Decomp. Base (β)	8	10	9	2
Decomp. Level (l)	2	2	4	1

Applications. Besides DTFHE operations, we also study encrypted general-purpose applications implemented by TFHE and DTFHE. We selected four circuit designs from the ISCAS'85 benchmark suite. These circuits cannot be realized using traditional FHE schemes like CKKS, as they necessitate linear operations such as additions, as well as Boolean Algebra operations like XOR. Although TFHE can implement these circuits bit by bit, we demonstrate that DTFHE can efficiently and inherently support both linear and Boolean operations within the circuits.

Simulation. We simulated the OFHE's cycle-level performance by our in-house CGRA modeling framework, and validated it against Strix [18]. Underpinning the architectural description of OFHE, the framework compiles DTFHE operations into data flow graphs, orchestrating and mapping each to an OFHE hardware unit. This methodology informs the derivation of both latency and energy metrics for each DTFHE operation.

5 RESULTS AND ANALYSIS

5.1 Comparison Against Prior Accelerators

GFB Latency. The bottleneck in TFHE gates with dual 1-bit inputs is the GFB, typically positioned at the end of a logic gate, operating on a 32-bit datapath. Figure 6 illustrates the latency differences between prior accelerators and OFHE when executing a GFB. FPT exhibits the longest GFB latency for parameter sets I and II, while OFHE achieves the most significant latency reduction, trimming Strix GFB latency by 12.5% and 8.7% for parameter sets I and II, respectively. Although Cryp uses faster photonic adders and multipliers, its latency for a GFB is determined by its on-chip CMOS memory speed. During an (I)FFT kernel, Cryp needs to write every intermediate result into the on-chip memory, so it has almost the same latency as Strix.

GFB Throughput. As depicted in Figure 7, the GFB throughput varies significantly among accelerators. MATCHA and FPT face constraints in supporting numerous simultaneous GFB operations

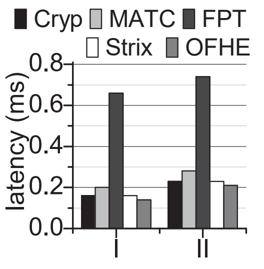

Figure 6: GFB Lat.

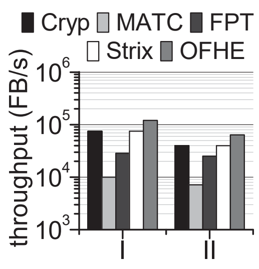

Figure 7: GFB thr.

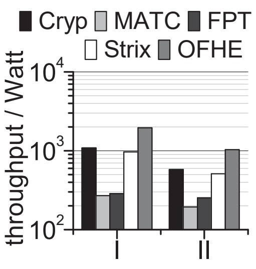

Figure 8: GFB thr/W.

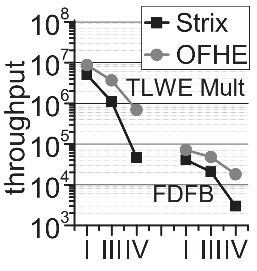

Figure 9: DTFHE Ops.

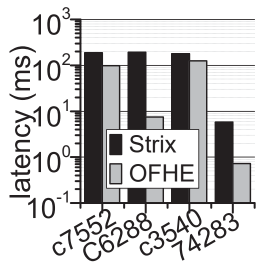

Figure 10: DTFHE Apps.

due to hardware limitations or lack of a fully-pipelined design. OFHE's PFFTE chips outperform FPT in GFB throughput, with enhancements of 61% & 57% over Strix for parameter sets I and II.

Throughput per Watt. Figure 8 depicts the energy efficiency of the accelerators, measured as GFB throughput per Watt. Apart from Cryp, the energy efficiency of the other platforms closely follows their throughput results. Cryp utilizes low-power photonic computing units, resulting in slightly superior energy efficiency compared to Strix. FPT, operating on FPGA substrate, incurs substantial power overhead, significantly reducing its energy efficiency. OFHE's PFFTE chips consume minimal power, resulting in a 2× and 94% improvement in GFB throughput per Watt compared to Strix for parameter sets I and II, respectively.

5.2 Performance of 64- & 128-bit Datapaths

Figure 9 compares the throughput of GFBs, TLWE Mults, and FDFBs between OFHE and Strix. GFBs, TLWE Mults, and FDFBs utilize parameter sets I, III, and IV, respectively, to encrypt different sizes of plaintexts. For parameter sets III and IV, DTFHE requires 64- and 128-bit datapaths. Prior TFHE accelerators face a limitation as none support datapaths exceeding 32-bits. Consequently, Strix only natively supports parameter set I. We employ the Karatsuba algorithm [9], with a time complexity of $O(3^{\log_2 n})$ on Strix to process 64- and 128-bit operations using native 32-bit operations, where n can be 64 or 128. This leads to an exponential decrease in throughput on Strix when transitioning from a 32-bit to a 64-bit to a 128-bit datapath, due to the high complexity of the Karatsuba algorithm. In contrast, OFHE maintains a linear latency evolution across its datapaths. Consequently, a significant throughput disparity arises between Strix and OFHE, ranging from 71.5% to 35.6×.

5.3 General Purpose Application Performance

We selected four circuits from the ISCAS'85 benchmark: c7552 (32-bit adder/comparator), c6288 (16x16 multiplier), c3540 (8-bit ALU), and 74283 (4-bit adder). Figure 10 compares the latency of these circuits on Strix and OFHE. While TFHE processes the circuits gate by gate on Strix, DTFHE utilizes multi-bit additions and multiplications on OFHE to realize the same circuits. OFHE demonstrates a latency reduction ranging from 29.4% to 96.1% compared to Strix. This improvement is particularly significant for c6288 and 74283, where most Boolean logic gates can be combined into multi-bit multiplication or addition operations. The superiority of DTFHE over TFHE in processing general-purpose applications is evident. For c7552 and c3540, the latency reduction benefits from both DTFHE advantages and OFHE's support for TLWE Mults and FDFBs. On average, compared to Strix, OFHE decreases the latency of these applications by 58.6%.

6 CONCLUSION

This study introduces OFHE, an electro-optical accelerator designed for DTFHE operations on multi-bit plaintexts, particularly BFV-like TLWE multiplications and FDFBs. Leveraging PFFTE chips, OFHE efficiently addresses the FFT and IFFT kernel bottlenecks, achieving enhanced performance with reduced power. When benchmarked against existing TFHE accelerators, OFHE showcases a 8.7% latency reduction, 57% throughput improvement, and an 94% boost in throughput per watt.

ACKNOWLEDGMENTS

This work was supported in part by NSF CCF-2105972.

REFERENCES

[1] Rubab Amin et al. 2020. Broadband Sub-λ GHz ITO Plasmonic MZM in Silicon Photonics. In *Integrated Photonics Research, Silicon and Nanophotonics*.
[2] D. H. Bailey. 1989. FFTs in external or hierarchical memory. In *ACM/IEEE Conference on Supercomputing*. 234–242.
[3] Michiel Van Beirendonck et al. 2022. FPT: a Fixed-Point Accelerator for Torus Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2022/1635.
[4] Zvika Brakerski et al. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Transactions on Computation Theory* 6, 3 (2014).
[5] Jung Hee Cheon et al. 2020. Remark on the Security of CKKS Scheme in Practice. Cryptology ePrint Archive, Report 2020/1581.
[6] Ilaria Chillotti et al. 2020. TFHE: Fast Fully Homomorphic Encryption Over The Torus. *Journal of Cryptology* 33, 1 (2020).
[7] Ilaria Chillotti et al. 2021. Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE. In *ASIACRYPT*.
[8] P. Duhamel, B. Piron, and J.M. Etcheto. 1988. On computing the inverse DFT. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36, 2 (1988), 285–286.
[9] Antonio Guimaraes et al. 2022. MOSFHET: Optimized Software for FHE over the Torus. Cryptology ePrint Archive, Report 2022/515.
[10] Nicholas C. Harris et al. 2014. Efficient, compact and low loss thermo-optic phase shifter in silicon. *Optics Express* 22, 9 (May 2014).
[11] D Hillerkuss et al. 2010. Simple all-optical FFT scheme enabling Tbit/s real-time signal processing. *Optics express* 18, 9 (2010).
[12] Shihan Hong et al. 2022. Ultralow-loss compact silicon photonic waveguide spirals and delay lines. *Photonics Research* 10, 1 (Jan 2022).
[13] Lei Jiang et al. 2022. MATCHA: A Fast and Energy-Efficient Accelerator for Fully Homomorphic Encryption over the Torus. In *Design Automation Conference*.
[14] Sangpyo Kim et al. 2022. BTS: An Accelerator for Bootstrappable Fully Homomorphic Encryption. In *ACM International Symposium on Computer Architecture*.
[15] Kotaro Matsuoka et al. 2021. Virtual Secure Platform: A Five-Stage Pipeline Processor over TFHE. In *USENIX Security Symposium*.
[16] Boris Murmann. [n. d.]. ADC Performance Survey 1997-2022. https://github.com/bmurmann/ADC-survey.
[17] Kevin Nam et al. 2022. Accelerating N-Bit Operations over TFHE on Commodity CPU-FPGA. In *IEEE/ACM International Conference on Computer-Aided Design*.
[18] Adiwena Putra et al. 2023. Strix: An End-to-End Streaming Architecture with Two-Level Ciphertext Batching for Fully Homomorphic Encryption with Programmable Bootstrapping. In *IEEE/ACM International Symposium on Microarchitecture*.
[19] Chenran Ye et al. 2015. A compact plasmonic MOS-based 2× 2 electro-optic switch. *Nanophotonics* (2015).
[20] Mengxin Zheng et al. 2022. CryptoLight: An Electro-Optical Accelerator for Fully Homomorphic Encryption. In *ACM NanoArch*.