IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 17 August 2021; revised 15 December 2021; accepted 14 January 2022.
Date of publication 25 January 2022; date of current version 6 December 2022.

Digital Object Identifier 10.1109/TETC.2022.3144101

Ultra High-Speed Polynomial Multiplications for Lattice-Based Cryptography on FPGAs

DUR-E-SHAHWAR KUNDI, (Member, IEEE), YUQING ZHANG, CHENGHUA WANG, AYESHA KHALID, (Member, IEEE), MÁIRE O'NEILL, (Senior Member, IEEE), AND WEIQIANG LIU, (Senior Member, IEEE)

Dur-e-Shahwar Kundi is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China, and also with the Queen's University Belfast, BT7 1NN Belfast, U.K.

Yuqing Zhang, Chenghua Wang, and Weiqiang Liu are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China

Ayesha Khalid and Máire O'Neill are with the Centre for Secure Information Technologies (CSIT), Queen's University Belfast, BT7 1NN Belfast, U.K.

CORRESPONDING AUTHOR: WEIQIANG LIU (liuweiqiang@nuaa.edu.cn)

This work was supported in part by the grants from National Natural Science Foundation of China (NSFC) under Grants 62022041 and 62134002, and in part by Engineering and Physical Sciences Research Council (EPSRC) Quantum Communications Hub under Grant EP/T001011/1.

ABSTRACT Lattice-based cryptography (LBC) has emerged as the most viable substitutes to the classical cryptographic schemes as 5 out of 7 finalist schemes in the 3rd round of the NIST post-quantum cryptography (PQC) standardization process are lattice based in construction. This work explores novel architectural optimizations in the FPGA-based hardware implementation of polynomial multiplication, which is a bottleneck in every LBC construction. To target ultra-high throughput, both schoolbook polynomial multiplication (SPM) and number theoretic transform (NTT) are explored: a completely parallel architecture of an SPM is undertaken while for NTT, radix-2 and radix-2^2 multi-path delay commutator (MDC) based pipelined architectures are adopted. Our proposed high-speed SPM (HSPM) structure on latest Xilinx UltraScale+ FPGA is $5\times$ faster than the state-of-the-art LBC designs. Whereas, the proposed high-speed NTT (HNTT) structure (i.e., R2^2MDC) takes only 0.63μs for the encryption, hence achieving the highest throughput of 408 Mbps. Moreover, all of the proposed designs achieve highest design efficiencies (i.e., throughput per slice (TPS)) in comparison to available LBC designs.

INDEX TERMS Lattice-based cryptography (LBC), schoolbook polynomial multiplication (SPM), number theoretic transform (NTT), ring-Learning with errors (R-LWE)

I. INTRODUCTION

Advances in the field of quantum computing has rendered the security of the digital world today at the brink of a complete breakdown. All of the currently deployed public-key cryptography (PKC) primitives, including RSA, DSA, elliptic curve cryptography (ECC) etc. will be insecure after the practical development of a quantum computer by virtue of Shor's algorithm [1]. Reacting to the urgency, the National Institute of Standards and Technology (NIST) initiated a post-quantum cryptography (PQC) process in 2016 to standardize quantum-resilient PKC for the public-key encryption (PKE)/key-encapsulation mechanism (KEM) and digital signature schemes (DSS) [2].

Of the various different flavors of PQC proposed to date, lattice-based cryptography (LBC) stands out for various reasons. *First*, these schemes offer security proofs based on NP-hard problems with average-case to worst-case hardness. *Second*, the LBC implementations are notable for their efficiency, primarily due to their inherent simple linear algebra based matrix/vector operations on integers. This makes them a preferred choice for high-speed networks as well as internet-of-things (IoTs) applications. *Third*, LBC constructions offer extended functionality for advanced security problems such as identity-based encryption (IBE) [3], fully-homomorphic encryption (FHE) [4] etc. in addition to the basic ones as needed in a quantum age [5]. *Finally, the popularity of LBC PQC schemes can be gauged from their proportion in NIST PQC standardization for PKE/KEM and DSS schemes; 26 out of 69 submissions in round 1 and 12 out of 26 in round 2 were lattice-based proposals [2], [6]. In a latest round 3 finalist selection (July 2020), 5 out of 7 constitute the lattice-based; CRYSTALS-Kyber, Saber and NTRU are*

PKE/KEM schemes whereas CRYSTALS-Dilithim and FALCON are DSS [7].

A number of hard mathematical problems underpin the lattice-based cryptography (LBC) schemes, including a commonly used learning-with errors (LWE) problem. The LWE or *standard lattice based schemes* use standard lattices requiring costly on the fly matrix-vector multiplications that are memory hungry and have quadratic complexity. Alternative to standard lattices is the Ring-LWE (R-LWE) or *ideal lattice-based schemes*, representing the matrix of standard lattices by a single row in ring lattices. The remaining rows are generated by cyclic shifts of the first row. Hence, it requires simpler computations, since the main arithmetic operation is now the polynomial multiplication instead of matrix-vector multiplications. Another variant of standard lattices is the **module lattices-based schemes** or (M-LWE), that were introduced as a trade-off between the efficiency of ideal lattices and the trust in the security of standard lattices. Hence, in M-LWE the matrix has smaller dimensions than LWE and the coefficients of the matrix are no longer simple integers but entire polynomials like R-LWE.

The acceleration of PKC in general and LBC in particular is an active research area, as the world is observing a continual and rapid expansion of internet and wireless-based communications across open networks. The provision of security to these multi-giga bit data transmission networks requires ultra high-speed custom designed hardware accelerators for PKC on field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs). Yet another motivation of this research is the fact that most of the LBC schemes are more computationally intensive compared to their older alternates used today like RSA etc.

The *polynomial multiplication* is common to all LBC variants: LWE/M-LWE/R-LWE and is the *most computationally* intensive operation of the cryptosytem. Consequently, its acceleration alone will accelerate the entire cryptoystem. It can be carried out either using the schoolbook algorithm or the number-theoretic transform (NTT) algorithm. The schoolbook based polynomial multiplication (SPM) is inherently simple but has a quadratic complexity of $O(n^2)$ for matrix-vector multiplication, where n is the lattice dimension. The NTT in comparison offers a remarkable speedup with a quasi-linear time complexity of $O(n\log n)$. It however, requires complex operations including pre-calculation, sequential re-arrangement and post-processing. It is critical to study both the algorithms for polynomial multiplication, since NTT can only be used with the specific parameter-sets (explained in Section 2.1) while SPM has a wider applicability for any parameter-set. Hence, this work undertakes the acceleration of R-LWE processor via both SPM and NTT in order to easily deployed in LWE/M-LWE variants with little modifications.

The reported LBC acceleration efforts deploying SPM, primarily target the lightweight applications [8]–[11], with very few targeting high-speed network applications [12], [13]. The naive SPM based R-LWE designs in [9] resulted in a very high latency, i.e., greater than $2n^2$ cycles and thereby taken 457 μs for the complete encryption phase with lattice dimension $n=256$ [9]. An optimized R-LWE architecture was proposed in [10], [11], that utilized the signed Gaussian data and accommodated 2 multiplications per DSP block resulting in a reduced latency. Hence, lowering the R-LWE encryption time to 229 μs [10] and 128 μs [11] with 1 and 2 such multiply-and-accumulate (MAC) units. The SPM based high-speed designs proposed in [12], [13] support the Module-Learning with Rounding (M-LWR) LBC scheme (i.e., Saber). The architecture with 256 parallel MAC units for $n=256$ and matrix_dimension $l=2$ (i.e., 2× computations of R-LWE) takes maximum of 41 μs [12] while with 512 parallel MAC units takes 26.7 μs [13] for the complete decapsulation phase. Increasing further the matrix dimension will have little effect on the overall latency as the next data block will be processed in the pipeline, once it is filled [12].

TABLE 1. Implementation results of previous work.

Design	# of Units	# of Cycles[a]	Exec. Time (μs)	Thr. (Mbps)
SPM				
[9]	1	131604/65802	457/228	0.56/1.12
[10]	1	69654/34436	229/114	1.12/2.25
[11]	2	35478/17732	129/64	1.98/3.97
[12]	256	13846/13202	43/41	6.0/6.3
[13]	512	3135/4012	20.9/26.7	12.3/9.57
NTT				
[16]	1	6300/2800	20/9	12.72/28.62
[17]	1	7197[b]	52.92	4.84
[18]	–/16	–/917	–/3.72	–/69/
[19]	26/22	1194/644	5.14/2.72	50/98
[20]	128	440/220	1.87/0.94	137/273

[a]*figures for Enc/Dec,* [b]*polynomial multiplication only.*

The inherent lower complexity in the NTT based polynomial multiplication and architectural optimizations can offer remarkable speedups [14]. Some of the more compact designs resorted to a *single butterfly unit* in hardware [15]–[17] while others targeting high-performance, employed *multiple butterfly units* and pipelining approaches [18]–[21]. The compact NTT based R-LWE design (for $n=256$) having single radix-2 butterfly (BF2) unit and sequential strategy, so far achieved 20 μs per R-LWE encryption [16], whereas the NTT architecture reported in [17], targeting the M-LWE scheme, i.e., CRYSTALS-Kyber with matrix_dimension $l=2$ attained 52.92 μs for the polynomial multiplication part only. However, high-speed NTT designs managed to reduce the computational time many-fold, the NTT architecture with 2 parallel BF2 reduced the decryption time to 3.72 μs [18] whereas in [19] resulted in 2.72 μs by the use of multi-path delay commutator (MDC) pipelining strategy. In one of the recent works, a fully parallel multi-lane radix-2 NTT algorithm based on the Stockham NTT for $n=256$ achieved 0.935 μs for one polynomial multiplication [20].

The performance figures of previous up-to-date implementations (as discussed above) are summarized in Table 1,

where the highest achievable throughput (Thr.) is 12.3 Mbps [13] with SPM while is 273 Mbps [20] with NTT.

This work presents two ultra high-speed hardware accelerators, targeting the core operation in every LBC constructions, i.e., polynomial multiplication. The major contributions of this work are summarized as follows:

1) A high-performance SPM (HSPM) core is proposed with several carefully worked out architectural optimizations: *First* an efficient data accessing technique, i.e., vector to matrix analogy (VMA) is devised to perform column-based multiplication, which also accommodates in-place coefficient reduction, resulting in encryption and decryption latency of $(5n+2)$ and $(3n+2)$ clock cycles, respectively (originally $(2n^2)$ and (n^2) in [9], ref. Section 4.1). *Second*, the parallelism offered by VMA is exploited by utilizing 128 built-in dedicated DSP blocks on FPGAs. *Third*, substantial hardware saving is achieved by signed modular multiplication and packing of twice per DSP block via an optimally pipelined signed double modular multiplication (SDMM) unit. *Finally*, the encryption and decryption operations are combined into a single efficient HSPM core, enabling reuse of major blocks.
2) Our proposed HSPM with better adaptability and scalability, surpasses all comparable previously reported FPGA implementations of SPM based R-LWE implementations for lattice dimension $n = 256$ in terms of throughput performance and design efficiency (i.e., throughput per slice (TPS)). On a Xilinx Virtex UltraScale+ our HSPM takes 4.30 μs with throughput of 60 Mbps for encryption and achieves 5$\times$ speed gain with highest efficiency compared to the best performing LBC implementation [13]. On a Xilinx Kintex-7, HSPM takes 4.93 μs with 52 Mbps for encryption and achieves a speed gain of $\approx$26$\times$ with a 14% better design efficiency in comparison to an efficient and parallel R-LWE cryptoprocessor on the same FPGA device [11].
3) A high-performance NTT cores (HNTT/HINTT) for lattice dimension $n = 256$, are proposed employing full pipelining in architecture via MDC methodology. First the simplistic radix-2 butterfly is utilized to present R2MDC NTT/INTT core. Further improvement in design is achieved by exploring the radix-2^2 (R2^2MDC) with an optimized BF4* butterfly unit, undertaken for the first time for R-LWE. The designed BF4* requires 29% less operations in comparison to a conventional BF4 W-matrix (Figure 11(a).)
4) Our proposed R2MDC and R2^2MDC perform remarkably faster and achieve better throughput efficiency (TPS) in comparison to all earlier reported NTT based R-LWE designs for a comparable lattice dimension of $n = 256$. Out of our two proposed designs, R2MDC results in a higher TPS efficiency and compared with the same architectural implementation [19] achieves $\approx 5\times$ speed gain. Whereas, our R2^2MDC design on Xilinx Virtex family of devices, offers the highest throughput of 408 Mbps with 0.63 μs execution time.

The rest of the paper is organized as follows. Section 2 introduces the theoretical background on LBC and its simplest variant, i.e., the R-LWE based PKE protocol. Section 3 describes the proposed high-performance LBC accelerators with the selected parameter sets. Section 4 discusses the detailed architecture of the proposed high-speed SPM (HSPM) based accelerator while Section 5 includes the architecture of two high-speed NTT/INTT (HNTT/HINTT) based accelerators. The hardware results on FPGA and comparisons with state-of-the-art implementations are provided in Section 6 . Finally, conclusions are presented in Section 7.

II. BACKGROUND

The LWE problem introduced by Regev in 2005 [22], served as a foundation for several modern LBC schemes. The classic LWE problem states that given a matrix a and samples of vector $b = a.s + e$ where e is a small (unknown) error vector then it is difficult to determine the secret vector s. Considering its application in a public-key encryption (PKE) protocol, the public parameter a is generated by a uniform (U) distribution, and the secret key s and the error vector e are produced by a discrete Gaussian (DG) distribution; χ with zero mean μ and standard deviation σ. But LWE requires computations with large matrices that need a lot of memory and also results in large key sizes.

Consequently, Lyubashevsky *et al.* in 2010 proposed for the first time, the R-LWE problem while maintaining the hardness of the original problem [23]. It is based on ideal lattices which correspond to ideals in the ring $\mathbb{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ where n is a power-of-two integer defining the lattice dimensions and q is a prime modulus. The generic R-LWE based PKE protocol is outlined by Algorithm 1, where the encryption and decryption processes are depicted as follows:

Algorithm 1. R-LWE Based PKE Protocol

Input: Public parameter: $\boldsymbol{a} \leftarrow U$; Secret key: $\boldsymbol{r_2} \leftarrow \chi$; Public key: $\boldsymbol{p}$;
Plaintext: $\boldsymbol{m} \leftarrow \{0, 1\}$.
Output: Ciphertext: $\boldsymbol{c_1}, \boldsymbol{c_2}$; Decrypted plaintext: $\boldsymbol{m'}$.
1: Sampling to generate $\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3} \leftarrow \chi$;
2: Encoding plaintext $\bar{\boldsymbol{m}} = \boldsymbol{encode}(\boldsymbol{m})$;
3: Encryption process: $\boldsymbol{c_1} = \boldsymbol{a} \times \boldsymbol{e_1} + \boldsymbol{e_2}$, $\boldsymbol{c_2} = \boldsymbol{p} \times \boldsymbol{e_1} + \boldsymbol{e_3} + \bar{m}$;
4: Decryption process: $\boldsymbol{c} = \boldsymbol{c_1} \times \boldsymbol{r_2} + \boldsymbol{c_2}$;
5: Decoding plaintext $\boldsymbol{m'} = \boldsymbol{decode}(\boldsymbol{c})$;
6: **return** $\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{m'}$.

All the operations are based on integer polynomials on the ring $\mathbb{R}_q$ and '$\times$', '$+$' represent the modular polynomial multiplication and polynomial addition operations, respectively. As evident from the protocol description, the DG sampler and the modular polynomial multiplication are the main

bottlenecks for the efficient hardware implementation of R-LWE. But the polynomial multiplication constitutes the major overhead; therefore the focus of this work is to design optimal architectures through the exploitation of maximum parallelism.

A. ALGORITHMS FOR POLYNOMIAL MULTIPLICATION

Generally, polynomial multiplication is computed using a SPM algorithm or NTT. The simplest method is the classical SPM and when used for standard/ideal lattices, the algorithm has a quadratic complexity of $O(n^2)$. It requires n^2 modular multiplications and $(n-1)^2$ modular additions/ subtractions. The product $a(x) \times b(x)$ can be computed by (1), as follows:

$$\begin{aligned} \boldsymbol{a(x) \cdot b(x)} &= \left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j x^{i+j}\right] \bmod (x^n+1) \\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (-1)^{\lfloor (i+j)/n \rfloor} a_i b_j x^{(i+j) \bmod n} \end{aligned} \tag{1}$$

The sign of each modular multiplication is determined by $(-1)^{\lfloor (i+j)/n \rfloor}$, as follows:

$$sign = \begin{cases} 0 & i+j < n \\ 1 & n \le i+j \le 2n-2 \end{cases} \tag{2}$$

Despite its quadratic complexity, the SPM is more generic than the NTT and is therefore applicable to any set of LBC parameter-set (with any lattice dimensions-n and modulus-q) and its variant, i.e., LWE/M-LWE/R-LWE etc.

On the other hand, NTT is deemed more attractive for polynomial multiplication due to its efficient execution having quasi-linear complexity of $O(n\log n)$. The NTT is primarily a variant of the fast Fourier transformation (FFT) defined over a finite field and does not require any floating point/complex arithmetic. As a consequence, all computations carried out are within the finite field or ring R_q, with the complex roots of unity in FFT replaced with integer roots of unity. However, NTT based polynomial multiplication only works with a particular set of parameters (i.e., parameters with circular convolution property [24] and *prime* modulus-q satisfying $q \equiv 1 \bmod 2n$).

For the NTT transformation, the coefficients of a polynomial $a(x) = (a_0, a_1, a_2, \ldots, a_{n-1})$ are expressed as $\hat{a}(x) = (\hat{a_0}, \hat{a_1}, \hat{a_2}, \ldots, \hat{a_{n-1}})$ and the transformation rules are as follows:

The forward NTT operation converts $a(x)$ to $\hat{a}(x)$, where,

$$\hat{a}(x) = \sum_{j=0}^{n-1} a_j \omega_n^{ij} \bmod q, i \in [0, n-1] \tag{3}$$

whereas, the inverse-NTT (INTT) operation converts $\hat{a}(x)$ back to $a(x)$, where,

$$a(x) = \frac{1}{n}\sum_{j=0}^{n-1} \hat{a}_j \omega_n^{-ij} \bmod q, i \in [0, n-1] \tag{4}$$

The product of two polynomials, $c = a \times b$ can be calculated as $c = INTT(NTT(a) * NTT(b))$. The iterative NTT algorithm as outlined by Algorithm 2 is similar to the FFT with the NTT inner loop involving butterfly computations.

Algorithm 2. Iterative NTT Algorithm [25]

Input: $A(x) \in \mathbb{Z}_q[x]/(x^n+1)$
Input: n-th primitive root of unity $\omega \in \mathbb{Z}_q, n = 2^l$
Output: $\hat{A}(x) = NTT(A(x))$
1: **for** $(i = 1; i \le l; i = i+1)$ **do**
2: $m = 2^{l-i}$
3: **for** $(j = 0; j \le 2^{i-1} - 1; j = j+1)$ **do**
4: **for** $(k = 0; k \le m-1; m = m+1)$ **do**
5: $U \leftarrow A[2 \cdot j \cdot m + k]$
6: $V \leftarrow A[2 \cdot j \cdot m + k + m]$
7: $A[2 \cdot j \cdot m + k] \leftarrow U + V$
8: $A[2 \cdot j \cdot m + k + m] \leftarrow (U - V) \cdot \omega^{2^{(i-1)} \cdot k}$
9: **end for**
10: **end for**
11: **end for**
12: return $\hat{A}$

A straightforward implementation of Eqs. (3) and (4) would not be any more efficient than a schoolbook approach having quadratic complexity, therefore, algorithms to enable fast computations of the NTT are required. The Cooley-Tukey (CT) radix-2 decimation-in-time (DIT) algorithm is the simplest implementation of the NTT that enables an $O(n\log n)$ complexity [26]. The algorithm recursively splits the problem into the even and the odd inputs. It requires the input polynomial values to be arranged in a bit-reversed order to enable in-place computation of the NTT. The algorithm computes the CT butterfly, calculating $c + \omega^N d$ and $c - \omega^N d$ for $N \in \{0, n/2 - 1\}$ and $\omega, c, d \in Z_q$. This fast algorithm can be used to compute both the NTT and the inverse NTT (INTT). The powers of the primitive roots of unity, so called twiddle factors or ω, can either be computed on-the-fly or pre-computed and stored to save the on-the-fly computational resources and the trade-off is ROM vs multipliers. Since, CT-DIT algorithm requires a bit-reversal step before starting the NTT computation and produces naturally ordered output. The Bit-reversal is required again before INTT computation if it is using the same CT-DIT butterfly. Removing bit-reversal is possible by using a fast decimation-in-frequency (DIF) technique, called gentleman-sande (GS) [25] for INTT computation. This DIF NTT algorithm also recursively splits the computation into even and odd sub-problems and expects the input array in bit reversed order.

The R-LWE based PKE protocol utilizing the frequency domain NTT is outlined in Algorithm 3, which is different from the ordinary R-LWE scheme based on SPM. Using the NTT technique, firstly we need to transform each input into the NTT domain (i.e., line 4) and then perform the computations as required by the R-LWE scheme (i.e., line 5). And for the final decrypted message, we need to apply the INTT to convert it back to the normal domain (i.e., line 7).

Algorithm 3. R-LWE Based PKE Using the NTT

Input: Public key: $\hat{a}$ and $\hat{p}$; Secret key: $\hat{r_2}$; Plaintext: $m \leftarrow \{0,1\}$.
Output: Ciphertext: $\hat{c_1}, \hat{c_2}$; Decrypted plaintext: m'.
1: Sampling to generate $e_1, e_2, e_3 \leftarrow D_\sigma$;
2: Encoding plaintext $\bar{m} = encode(m)$;
3: Computing $e_3 m = e_3 + \bar{m}$
4: Computing $\hat{e_1}$ = NTT(e_1), $\hat{e_2}$ = NTT(e_2), $e_3\hat{m}$ = NTT $(e_3 m)$;
5: Ciphertext is calculated $\hat{c_1} = \hat{a} * \hat{e_1} + \hat{e_2}$, $\hat{c_2} = \hat{p} * \hat{e_1} + e_3\hat{m}$;
6: Decrypting $\hat{c} = \hat{c_1} * \hat{r_2} + \hat{c_2}$;
7: Computing c = INTT($\hat{c}$)
8: Decoding plaintext $m' = decode(c)$;
9: **return** c_1, c_2, m'.

B. PARAMETER-SET

Generally, the R-LWE based PKE is parameterized by n (lattice-dimension), modulus-q and σ. In this work, we have taken up the well-known parameter-set of R-LWE, i.e., $n = 256$, $q = 7681$ and $\sigma = 4.51$ as proposed in [27], which was highly utilized by [9]–[11]. These chosen parameter-set has the advantage of being able to employ the NTT in addition to schoolbook method for the polynomial multiplication and provide medium security equivalent to NIST PQC security level-1 [15]. Moreover, the chosen lattice dimensions for the third round finalist schemes in NIST PQC, i.e., Kyber and Saber also have the same lattice dimensions [13], [21].

Algorithm 4. Barrett's Modular Reduction Algorithm

Input: x: 26-bit unsigned, q = 7681 modulus
Output: y: 13-bit unsigned
1: t = $\lfloor \frac{x}{q} \rfloor$;
2: tq = t × q;
3: y = x - tq;
4: **while** (y ≥ q) **do**
5: y = y - q;
6: **end while**
7: **return** (y);

C. MODULAR REDUCTION (MR) ALGORITHM

For the modular reduction (MR) operation in multiplication, a simplest Barrett's reduction algorithm [28] (as outlined in Algorithm 4) is considered rather than Montgomery's reduction algorithm [29] that requires numbers to be converted into and out of "Montgomery form". To accelerate Barrett's MR in hardware, the shift-addition-multiplication-subtraction-subtraction (SAMS2) technique is utilized, which is adopted in [9], [10], [19]. For example, for modulus $q = 7681$ the quotient t (line 1) is approximated to $(x \ll 13) + (x \ll 17) + (x \ll 21)$ while multiplication by modular q (line 2) is done by $(t \ll 13) - (t \ll 9) + t$. The resulting hardware for Barrett's MR [9] requires a large number of adders, subtractors and shift blocks to generate the 26-bit value in order to subtract it from the 26-bit input x (line 3) and further requires at most three subtractions of q to get the required 13-bit output. This may subjects to timing attacks [30], [31].

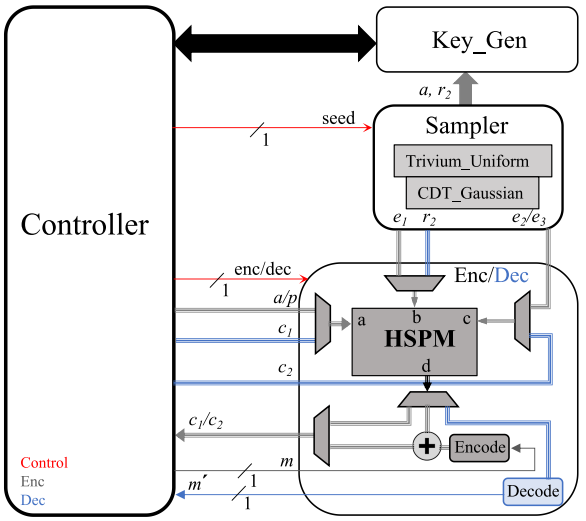

FIGURE 1. Block diagram of the proposed SPM based R-LWE accelerator.

III. THE PROPOSED HIGH-PERFORMANCE LBC ACCELERATORS

In this section, two high-performance LBC accelerators are proposed. One based on the SPM and the other based on NTT algorithm considering the R-LWE scheme with the aforementioned parameter-set. Since polynomial multiplication is the core module of every LBC scheme, its functionality is the same across the different variants, i.e., ranging from LWE to R-LWE. Therefore, the proposed hardware designs for R-LWE are equally feasible for LWE and M-LWE schemes but may require minor modification with respect to the modulus-q only. For example, the modulus $q \equiv 1 \bmod 2n$ can be used to support NTT whereas $q = 2^a$ does not require any MR operation etc. Moreover, the full R-LWE hardware needs to operate l times based on the matrix_dimension 'l' of LWE and M-LWE schemes.

The block diagram of the SPM based R-LWE accelerator is shown in Figure 1. In a R-LWE based PKE protocol, two polynomial multiplications are needed during the encryption (line 3 Algorithm 1) while one polynomial multiplication is needed during decryption (line 4 Algorithm 1). Therefore, to achieved a balanced architecture a unified and fully paralleled core of SPM named as high-performance SPM (HSPM) is designed to perform the polynomial multiplication for both the encryption and decryption. The functionality of HSPM is defined through the controller via an enc/dec signal that in turn controls all the multiplexers. During encryption (shown by the gray datapath), first the error terms e_1, e_2/e_3 are sampled from the Gaussian distribution and in the meantime loaded into the HSPM along with the pre-calculated public-key p and a sampled public parameter a from the Key_Gen module. Afterwards, processed by the HSPM in a pipelined

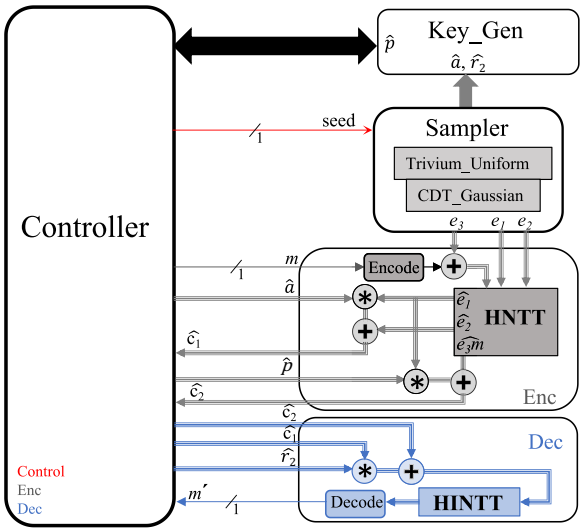

FIGURE 2. Block diagram of the proposed NTT based R-LWE accelerator.

fashion to first generate c_1 and then $c_2^{'}$, where m after encoding is added to it to get the final c_2. In the last, the output from the HSPM is obtained serially. Similarly, during decryption (shown by the blue datapath) the ciphertexts c_1, c_2 along with the secret key r_2 are loaded and then processed to calculate $c^{'}$ which after decoding reproduces the final output message $m^{'}$. Hence, the latency for encryption is almost twice than that for decryption. A detailed description of the HSPM hardware accelerator is provided in Section 4.

The block diagram for the NTT based R-LWE accelerator is shown in Figure 2. Here for simplicity the encryption and decryption processes are designed separately. During encryption, first the error-terms; e_1, e_2, and e_3m need to be transformed into the NTT domain as $\hat{e_1}$, $\hat{e_2}$, and $\hat{e_3m}$ (line 4 Algorithm 3). In order to achieve high-performance, all three terms are processed by the high performance NTT (HNTT) core in a fully parallel fashion and therefore, will approximately have the latency of a single NTT module. Then, after the point-wise modular multiplication and modular addition of $\hat{a}$ with $\hat{e_1}$ and $\hat{e_2}$ respectively, it produces ciphertext $\hat{c_1}$, while $\hat{p}$ with $\hat{e_1}$ and $\hat{e_3m}$ respectively, produces ciphertext $\hat{c_2}$. In decryption, we require only a single high performance INTT (HINTT) module to generate the final decrypted message $m^{'}$ after the decoding. The received $\hat{c_1}$ (already NTT transformed) undergoes a point-wise modular multiplication with $\hat{r_2}$ and then point-wise modular addition with $\hat{c_2}$ to produce $\hat{c^{'}}$, which is transformed back to the normal domain through the INTT transform. In this architecture, both encryption and decryption have the same latency. A detailed description of the HNTT hardware accelerator is provided in Section 5.

IV. DESIGN OF HIGH-PERFORMANCE SPM (HSPM) ACCELERATOR

In this section, we explain the internal architecture of our proposed HSPM accelerator. The HSPM hardware design is

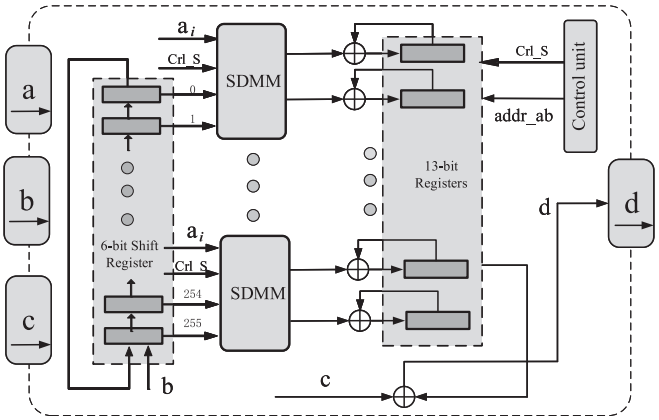

FIGURE 3. The proposed high-performance SPM (HSPM) structure.

fully parallelized, comprising of 128 MAC units for n=256. Subsequently, each MAC unit accommodates 2 parallel modular multiplications via a single DSP utilizing the signed data representation, hence named as signed double modular multiplication (SDMM) unit.

The overall architecture of HSPM accelerator is shown in Figure 3, that includes three pipelined stages i.e., the data loading phase, the modular polynomial multiplication unit, i.e., SDMM and final accumulation registers. The input data is applied to the HSPM hardware design in a serial-to-parallel fashion while the final result after the addition is obtained serially via an address signal *addr_ab*.

In R-LWE based PKE, the fundamental equation is d = a × b + c, where operand-a represents p, a, c_1 (from Algorithm 1), operand-b represents e_1, r_2 and operand-c represents e_2, e_3, c_2. During the loading phase, all the 256 coefficients of polynomial-b; $b_i = 0 \ldots 255$ are input serially into the 6-bit shift register while the first coefficient a_0 of polynomial-a is applied in parallel to all 128 SDMM units. Each SDMM is able to perform the modular multiplication of two coefficients of 'b' with a single a_i as shown in Figure 3. After the complete loading of polynomial-b coefficients, the SDMM units start to perform the multiplication (i.e., $a_i \times b$) of each i^{th} coefficient of a with 256 coefficients of b in parallel and then the procedure is repeated for $i = \{0 \text{ to } 255\}$ cycles. To access the corresponding coefficient of a as well as to perform the inevitable in-place reduction of b, an efficient sequential strategy is devised based on the vector to matrix analogy (VMA) (further explained in Section 4.1). Finally, the results of each SDMM calculation are accumulated in each cycle and the result of the final polynomial-product 'p' is sequentially read by the address signal *addr_ab* and correspondingly added to the coefficient of 'c' that will produce the final output 'd' (i.e., d = a × b + c), and transmitted serially in n clock cycles.

A. EFFICIENT SEQUENTIAL STRATEGY-VMA

To implement a high-speed SPM structure with full parallelism, an efficient sequential strategy is proposed that not only control the cyclic shifting of 'b' but at the same time perform

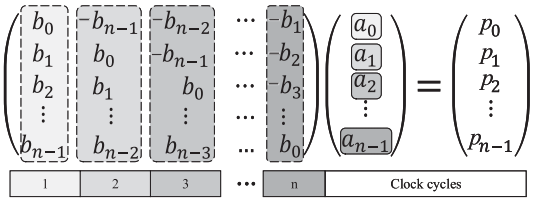

FIGURE 4. Devised vector to matrix analogy (VMA).

the in-place reduction of the partial-products (required due to the finite field operation). To accomplish this, the SPM vector operation (i.e., Eq. (1)) is expressed in the form of a matrix as shown in Figure 4, where one of the operands i.e., 'b' is transformed into an $n \times n$ matrix from $n \times 1$ vector.

The first column of the matrix represents the original 'b' vector i.e., b_0, b_1, ..., b_{255} that during the first clock cycle is accessed to perform the modular multiplication with the 1st coefficient of 'a' (i.e., a_0) to produce the first array of 256 partial-products simultaneously via 128 SDMM units (as detailed in Section 4.2). Then, the second column represents the circular shift by one position i.e., -b_{255}, b_0, ..., b_{254}, where -b_{255} represents the in-place reduction of the 256th coefficient as [q-b_{255}]. So, during the second clock cycle this column with the next coefficient of 'a' (i.e., a_1) generates the second array of 256 partial-products and at the same time is accumulated with the first array of partial-products and so on. Hence, the calculation of the entire polynomial multiplication requires n clock cycles instead of n^2. Further, it involves the in-place reduction of only one coefficient per cycle via a controller and this is done during the same operation. After the polynomial multiplication is completed, the final products are then added to the corresponding coefficients of the third operand-c to implement the entire HSPM structure.

B. SIGNED DOUBLE MODULAR MULTIPLICATION (SDMM) UNIT

To accommodate two multiplications per DSP Slice, we utilized signed Gaussian sampling (as proposed in [10]) for the error-vector e_1 term as well as the secret-key r_2. In a signed representation, the range of the Gaussian distribution varies from $[0, k\sigma]$ to $[q - k\sigma, 0]$ where $k = 1, 2, 3, \ldots$ and is symmetrically distributed across $\mu = 0$. In other words, there is no data in the middle of $[0, q-1]$ and the maximum value of the 13-bit samples (i.e., up to 7σ) can be easily represented by a 6-bit signed number. The modular multiplication using these signed numbers can be easily implemented using Eq. (5):

$$b \times -a \bmod q = q - [b \times a \bmod q] \tag{5}$$

where, the MSB signed-bit (as shown in Figure 5) will only define the subtraction of result from the modulus q.

Hence, the 13×13-bit modular multiplication is converted to a 13×5-bit modular multiplication. This not only enables us to reduce the hardware but also provides us the opportunity to perform two simultaneous multiplications via a single DSP slice. One DSP slice has a 27×18-bit

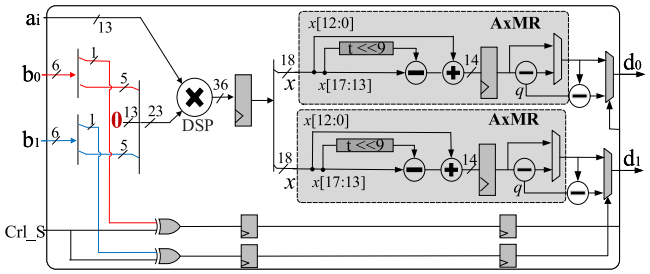

FIGURE 5. The proposed pipeline SDMM structure.

multiplier and the two 13×5-bit multiplications, i.e., $b_0 \times a$ and $b_1 \times a$ can be easily combined as $(b_1 + b_0) \times a$, where b_0 and b_1 are separated by at-most 13 '0' bits. Therefore, only one multiplication operation is required and the lowest 18-bit output of DSP represents the first product d_0 while the highest 18 bits denotes the second product d_1 corresponding to b_0 and b_1 respectively as shown in Figure 5.

After splitting, the two 18-bit products undergo modular reduction separately. We designed MR circuitry based on approximation as given in Eq. (6) [32]. The approximated MR (AxMR) unit performs the 18-bit reduction with 90% less hardware than the SAMS2 technique and is constant time as it requires only a single subtraction of q. The shaded portion in Figure 5 is the AxMR hardware. It comprises of a single shift block, a 14-bit subtractor and a 13-bit adder in comparison hardware of [9]. The AxMR in addition to the multiplier, produces the output in 2 clock cycles.

$$\begin{aligned} x \bmod 7681 &= 511(x_{17\ldots13}) + (x_{12\ldots0}) \\ &= (x_{17\ldots13} \ll 9) - (x_{17\ldots13}) + (x_{12\ldots0}) \end{aligned} \tag{6}$$

The pipelined HSPM accelerator for the processing of first vector yields $3 \times n + 2$ clock cycles; where the data loading phase takes n cycles (i.e., sampling phase), afterwards the SDMM operation requires $n + 2$ cycles while the final parallel-to-serial out takes n cycles as shown in Figure 6. Loading of the next vector for encryption can be done during the same output phase thereby saving n cycles and is then processed in the same manner, resulting in final latency of $5 \times n + 2$ cycles. Hence, HSPM can be easily extended to processes multiple vectors as in case of M-LWR scheme [12], [13].

The SPM in comparison to other efficient techniques; Karatsuba [33]/Toom-Cook [34], renders a very simple controller design for the multiplication as well as the in-place reduction

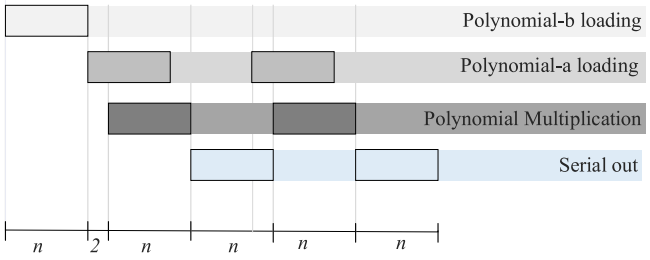

FIGURE 6. The HSPM processing timeline.

TABLE 2. NTT/INTT parameter.

Constant	n	q	φ	ω	ω^{-1}	φ^{-1}	n^{-1}
Value	256	7681	1704	198	1125	4431	7651

of product terms after the vector-vector multiplications is easily incorporated in the design process. Most importantly can be easily re-configured to support any modulus-q. Whereas, the Karatsuba/Toom-Cook based implementations require major modification when the q changes as it will not only effect the width of the multiplier but also demand MR operation (w.r.t q) to avoid the overflow by the multiplication/addition at each level.

V. DESIGN OF HIGH-PERFORMANCE NTT (HNTT) ACCELERATOR

In this section, we elaborate on the internal architecture of our proposed HNTT accelerators (as shown in Figure 2). Based on chosen n, i.e., 256, two n-point NTT cores are designed: a radix-2 NTT/INTT core and a radix-2^2 NTT/INTT core that involves an optimized version of a radix-4 algorithm. For the pipelined implementation of both the cores, MDC architecture of FFT algorithm is utilized [35] and the resulting NTT/INTT constants are listed in Table 2.

A. PIPELINED RADIX-2 MDC NTT/INTT CORE

The proposed radix-2 MDC (R2MDC) NTT/INTT core is shown in Figure 7, where the NTT includes the pre-processing (i.e., multiplication of input with φ) while INTT includes the post-processing (i.e., multiplication of final output with $\varphi^{-1} \times n^{-1}$), in addition to the main stages. For radix-2, the pipelined MDC architecture has 2-input and 2-output streams and $log_2 n$ processing elements (PE), i.e., 8. Each PE composed of a radix-2 butterfly (BF2) block, the data delay unit (DU), a two-channel commutator (C2) stage, and a ROM for storing the twiddle factor (ω/ω^{-1}).

The PE_i represents the i-th stage of the NTT transformation, where PE_1 and PE_8' are different from the other PEs in terms of the DU as well as the BF2 unit of PE_8', which does not require the multiplication by the twiddle factor. The DU is implemented by the 13-bit shift register to provide the required data delay before and after the BF2 unit in order to match the timing required for modular multiplication. The delay required at each stage ($i = 2, 3, \ldots, 7$) of the NTT core is different, therefore it is denoted as $DU0_i$ and $DU1_i$ for stream 0 & 1 to generate delays of $n/2^i$ and $n/2^{i+1}$, respectively.

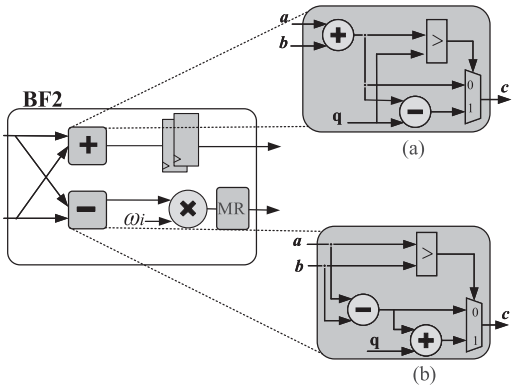

FIGURE 8. BF2 architecture (a) Modular Addition (b) Modular Subtraction.

A further design consideration is that we configured LUT as a distributed ROM (DROM) for the storage of the ω/ω^{-1} instead of dedicated Block-RAM (BRAM) because of the much smaller space requirements. This not only saves hardware resources but also boosts efficiency. Moreover, during the INTT the result needs to be multiplied by $\varphi^{-1} \times n^{-1}$ in the post-processing step, which is also stored in the DROM, as shown in Figure 7.

The internal architecture of the BF2 unit is shown in Figure 8. The BF2 unit consists of a 13-bit modular addition, a 13-bit modular subtraction and a 13-bit modular multiplication followed by the MR unit. The detailed designs for the modular addition and modular reduction are shown in Figures 8(a) and 8(b) respectively, which composed of an adder, a subtractor, a comparator and a MUX. In the modular adder, the additions of two numbers are performed first and then compared with the modulus q. If the result is greater than q, then the output c is the subtraction of q from the result otherwise it is $a + b$. Whereas, in the modular subtractor the size of a and b are compared first and then if $a > b$ the output c is $a - b$, otherwise $a - b + q$.

The entire R2MDC NTT/INTT pipelined architecture in a R-LWE encryption utilizes a total of 31 DSPs: 27 DSPs for 3 parallel NTT cores (i.e., e_1, e_2 and e_3) in addition to 4 DSPs (2×2-point) for the point-wise multiplication (as shown in Figure 2). Whereas, single NTT/INTT core costs 9 DSPs (i.e., 7 for PE_1 to PE_7 and 2 for the pre/post-processing stage). The

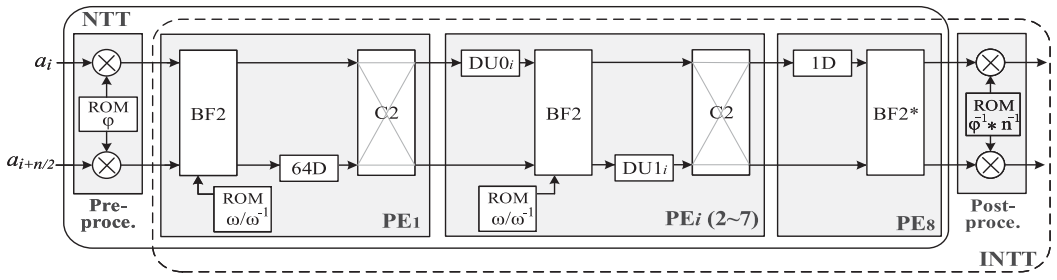

FIGURE 7. The proposed R2MDC NTT/INTT architecture.

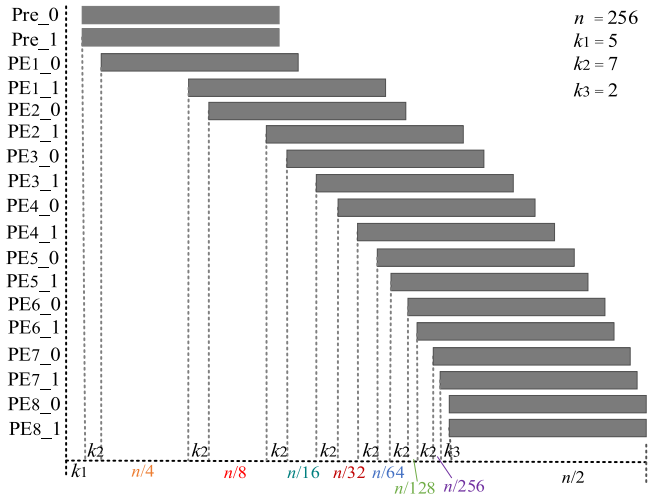

FIGURE 9. The R2MDC NTT processing timeline.

complete processing of data through the different stages of our proposed pipelined R2MDC based HNTT core is shown in Figure 9. The pre-processing stage takes k_1 clock cycles to generate the first output that goes into PE_1 which in turn yields k_2 and $k_2 + n/4$ clock cycles for the respective data streams 0 and 1. Both the data streams follow the same execution pattern from PE_2 to PE_7 by utilizing k_2 and $k_2 + n/2^{i+1}$ clock cycles while the last PE takes only k_3 clock cycles and then the final $n/2$ clock cycles for the processing of n=256 data. Hence, the latency of this design is calculated as $(k_1 + 7 \times k_2 + k_3) + (16log_2 n - 1) + (n/2)$ that in addition to final point-wise multiplication & addition costs total of 316 cycles.

B. PIPELINED RADIX-2^2 MDC NTT/INTT CORE

To further enhance the performance of NTT based accelerator, n=256 also provides us with the opportunity to utilize a higher radix, i.e., radix-4 or radix-2^2 other than the radix-2. We designed a pipelined HNTT core targeting radix-2^2 MDC architecture, that was not undertaken by any of the previous up-to-date implementations. However, the butterfly unit of a high-radix NTT consumes more hardware resources but provides far better speed than a radix-2 NTT. The procedure to infer a radix-4 or radix-2^2 NTT from Eq. (3) for the normal NTT operation is to divide it into four groups as shown in Eq. (7). The $0 \leq l \leq 3$ defines the four groups, each with $n/4$ outputs: $0 \leq i \leq (n/4 - 1)$.

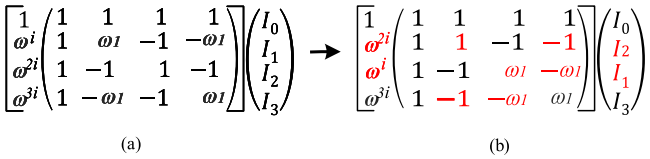

FIGURE 11. W-matrix (a) BF4 (b) BF4*.

$$\hat{a}_{4i+l} = \sum_{j=0}^{\frac{n}{4}-1} \left[\sum_{k=0}^{3} a_{j+\frac{kn}{4}} (\omega_n^{\frac{n}{4}})^{kl} \right] \cdot \omega_{\frac{n}{4}}^{ij} \cdot \omega_n^{lj} \bmod q$$
$$\text{where } l = 0, \ldots, 3 \text{ and } : i = 0, \ldots, (n/4 - 1). \tag{7}$$

The proposed radix-2^2 MDC (R2^2MDC) NTT/INTT core is shown in Figure 10. The pipelined MDC architecture for radix-2^2 has 4-input and 4-output streams and $log_4 n$ PE i.e., 4. Each PE is further composed of an optimized version of a radix-4 butterfly block, i.e., BF4*. For the NTT, the input data first undergoes the pre-calculation and is then processed by the 4 stages while for INTT, the input data first passes through the 4 transformations and then the post-calculation. The pre/post-calculation is similar to that for the radix-2 NTT/INTT.

The optimized BF4* structure is designed using W-matrix presented in Figure 11(b), which is formed by the re-adjustment of an original matrix of Figure 11(a) in such a way that the hardware resources for modular multiplication, modular addition/subtraction can be shared. As a result, the proposed BF4* architecture for radix-2^2 now comprises of 8 modular additions/subtractions and 4 modular multiplications in comparison to 12 modular additions/subtractions and 5 modular multiplications as needed for conventional radix-4 BF4. By comparing the total number of operations, optimized BF4* results in saving of 29% (i.e., $(\frac{17-12}{17}) \times 100$) of the reduced operations than the original BF4 W-matrix. The BF4* architecture is the same for the first three PE stages, but in the last PE_4' we have only one multiplication with a constant, i.e., w_1.

The proposed pipelined BF4* architecture is shown in Figure 12, where, the value of w_1 is $w_n^{n/2}$ for the NTT while the value of w_1 is $w_n^{-n/2}$ for the INTT.

The entire R2^2MDC NTT/INTT pipelined architecture in a R-LWE encryption utilizes a total of 59 DSPs: 51 DSPs for the 3 parallel NTT cores (i.e., e_1, e_2 and e_3) in addition to 8

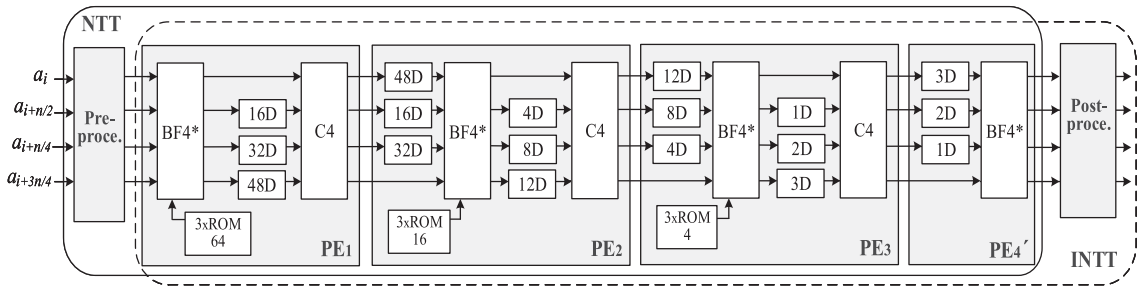

FIGURE 10. The proposed R2^2MDC NTT/INTT architecture.

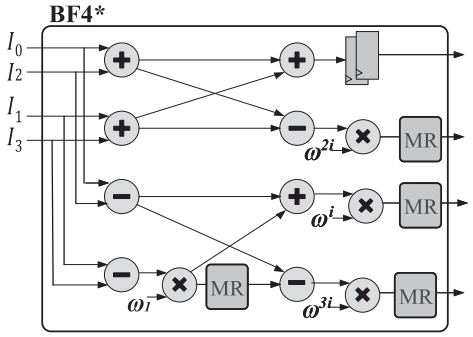

FIGURE 12. **The proposed pipelined BF4* architecture.**

DSPs (2×4-point) for the point-wise multiplication. Whereas, single NTT/INTT core costs 17 DSPs (i.e., 12 for PE_1 to PE_3) + 1 for PE_4' + 4 for the pre/post-processing stage). The complete processing of data through the different stages of our proposed pipelined R2^{2}MDC based HNTT core is shown in Figure 13. Here, we have four streams, i.e., 0 to 3 that undergo the same execution pattern as that of R2MDC architecture. The latency of this design is $(k_1 + 3 \times k_2 + k_3) + (16log_4 n - 1) + (n/4)$ that in addition to final point-wise multiplication & addition results in 181 clock cycles.

Both the proposed R2MDC and R2^{2}MDC HNTT accelerators are fully pipelined and afterwards will take only 64 cycles for the processing of next input data. Hence, these designs can be easily employed for the LWE and M-LWE [21] scheme with 'l' matrix_dimension.

VI. SCALABILITY

Current HSPM architecture with 128 SDMMs units carry out 256 multiplications/cycle, requiring a total of 256+2 cycles for $n = 256$. This architecture can be easily scaled up or down to achieve different area/latency trade-offs w.r.t final complexity of O(n^2). Moreover, HSPM is highly versatile and can be easily extended to accommodate different values of n, i.e., 512 without any major change in the circuit. Whereas, the HNTT architecture being designed for $n = 256$ may not support other lattice dimensions depending upon the radix selection; 2, 4, 8 etc. The R2MDC works with lattice dimension that is powers of two (i.e., $n = 2^8 = 256$ or $n = 2^9 = 512$ etc.) while R4MDC only works with powers of 4 (i.e., $n = 256 = 4^4$), hence will not supports $n = 512$. Moreover, the HNTT will require major modification in the architecture (i.e., PEs, DU, C2 etc.) with change in lattice dimensions.

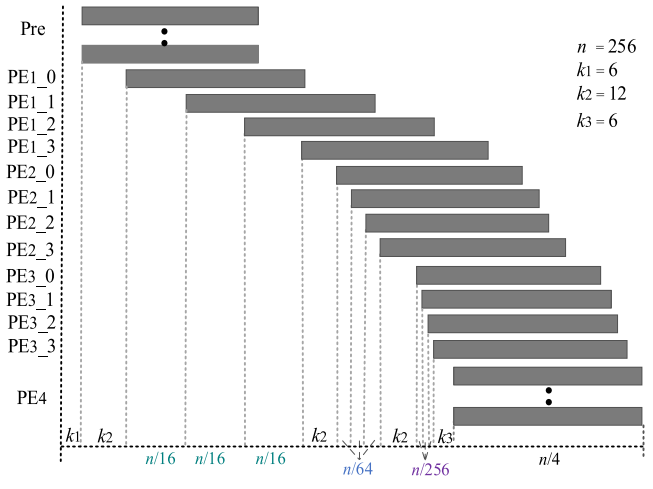

FIGURE 13. **The R2^{2}MDC NTT/INTT processing timeline.**

VII. HARDWARE RESULTS AND COMPARISON

The proposed polynomial multiplication accelerators are synthesized and implemented using the Xilinx Vivado 2018.3 suite targeting the latest Virtex UltraScale+ (VU9P) devices. Results on Kintex−7 (K480T) and Virtex−7 (VX690T) devices are also given for comparison purposes. Tables 3 and 4 list the detailed resource consumption (in terms of LUTs/FFs/Slices, DSPs, BRAMs, Equivalent Slices (Equ. Slices)) and performance results (operating frequency, execution cycles, execution time, throughput (Thr.) and Thr. per Equ. Slices (TPS) as a design efficiency metric [10]) for our proposed HSPM and HNTT accelerators. For clarity, we take our cores throughput and TPS as reference (denoted by **1** in tables) and compare with all other reported cores throughput and TPS, respectively, in terms of ratios or rates (theirs/ours). (Greater than 1 represents the improvement and less than 1 represents the deterioration with respect to our proposed designs.)

Our proposed HSPM accelerator on a Virtex UltraScale+ consumes 3.3k Slices and 128 DSPs for a unified Enc/Dec core and takes just 4.30 μs/2.58 μs for the complete encryption/decryption process, respectively, with high design efficiency, i.e., a TPS of 6.00/9.90 compared with the state-of-the-art LBC designs as well as other PQC candidate, i.e., SIKE [36] as shown in Table 3. Our proposed design achieved the highest performance, i.e., 60 Mpbs (for the encryption with n=256) in contrast to up-to-date SPM based high-throughput LBC designs [12], [13]. Moreover, comparing to the efficient R-LWE design [11] on the same Kintex-7 platform, our proposed HSPM design during encryption achieves a speed gain of 26× (i.e., 129/4.93) while reducing the number of cycles by a factor of 28, i.e., from 35478 to 1282. The same design offers 14% (i.e., $\frac{2.46-2.16}{2.16} \times 100$) better design efficiency for encryption over [11] as it is saving n clock cycles each time while processing multiple vectors.

In Table 4, our proposed high-performance R2MDC based HNTT/HINTT accelerator for R-LWE on a Virtex-7 takes around 1.09 μs for both the Enc/Dec cores and attains 236 Mbps throughput while consuming around 3× the decryption hardware for the Enc, i.e., 2678 Slices and 31 DSPs (total 5.85k Equ. Slices) because of the deployment of the 3 parallel HNTT cores as shown in Figure 2. Our proposed R2^{2}MDC achieves ≈ 2× (i.e., 408/236) better throughput performance than our proposed R2MDC architecture. It takes on average 0.62 μs for both the Enc/Dec and attains the highest throughput figure of 408 Mbps while consuming only 44% (i.e., $\frac{10.5k-5.85k}{10.5k} \times 100$) more hardware resources for the Enc, i.e., 4494 Slices and 59 DSPs (total 10.5k Equ. Slices). Comparing our proposed R2MDC HNTT architecture with

TABLE 3. Results comparison of the proposed HSPM accelerator.

Implementation	Device	Type	LUTs/FFs/Slices	DSPs	BRAMs	Equ.[a] Slices	Freq. (MHz)	# of Cycles	Exec. Time (μs)	Thr. (Mbps)	TPS (Kbps/Equ. Slices)
SIKEp503[36]	Virtex-7	Enc /Dec	21.2k/13.6k/≈8.2k[b]	162	38	29k	142	1973k /2101k	13.9k /14.8k	—	0.66(0.26)
R-LWE [10]	Kintex-7	Enc	898/815/303	1	3	754	304	69654	229	1.12(0.021)	1.75 (0.42)
		Dec	635/190/194	1	1	412	303	34436	114	2.25(0.026)	5.44 (1.32)
R-LWE [11]	Kintex-7	Enc /Dec	1381/1179/479	2	2	916	275	35478 /17732	129 /64.5	1.98(0.038) 3.97(0.046)	2.16(0.878) /4.33 (1.05)
This work (HSPM)	Kintex-7	Enc /Dec	20k/18k/8k	128	0	21k	260	1282 /770	4.93 /2.96	52(**1**) 86(**1**)	2.46 (**1**) /4.10 (**1**)
M-LWR [13]	UltraScale+	Encap	24.9k/10.7k/≈4.2k[b]	0	2	4.3k	150	3135	20.9	12.3 (0.205)	2.86 (0.476)
M-LWR [12] (LightSaber)	UltraScale+	Encap /Decap	12.3k/11.2k/2k	256	3.5	15k	322	13846 /13202	43 /41	6.0(0.100) /6.3(0.063)	0.40 (0.066) /0.42 (0.042)
This work (HSPM)	UltraScale+	Enc /Dec	19k/18k/3.3k	128	0	10k	298	1282 /770	4.30 /2.58	60(**1**) 99(**1**)	6.00 (**1**) /9.90(**1**)

[a]*Equ. Slices: 1 DSP ≈ 102.4 Slices(7 series)/51.2 Slices (UltraScale+) while 18K BRAM ≈ 116.2 Slices (7 series)/58.1 Slices (UltraScale+) [10],* [b]*Estimates while comparing LUTs with implementations on same FPGA device, UltraScale+ has 8 LUTs in one Slice/CLB.*

TABLE 4. Results comparison of the proposed HNTT based accelerators.

Implementation	Device	Type	LUTs/FFs/Slices	DSPs	BRAMs	Equ.[a] Slices	Freq. (MHz)	# of Cycles	Exec. Time (μs)	Thr. (Mbps)	TPS (Kbps/Equ. Slices)
M-LWE [17] Kyber512	Artix-7	Enc /Dec	442/237/≈147	1	3[b]	598	136	7197	52.92	4.84 (0.020)	8.09 (0.200/0.059)
R-LWE [16]	Virtex-6	Enc /Dec	1349/860/410	1	2	744	313	6300 /2800	20 /9	12.72 (0.053) /28.62 (0.116)	17 (0.421) /38.47 (0.281)
R-LWE [18]	Spartan-6	Enc	–/–/953	16	14.5	3.32k	247	917	3.72	69 (0.292)	21 (0.520)
R-LWE [19] (R2MDC)	Stratix-4	Enc	28105/28358/—	26	220[c]	—	232	1194	5.08	50 (0.211)	—
		Dec	4122/6178/—	22	15[c]	—	237	644	2.72	94 (0.383)	—
NTT [20]	Spartan-6	Enc /Dec	—/—/14000	128	1	27.2k	235	440 /220	1.87 /0.94	137 (0.580) /273 (1.114)	5.00 (0.123) /10.00 (0.073)
This Work (R2MDC)	Virtex-7	Enc	7334/5644/2678	31	0	5.85k	291	316	1.09	236 (**1**)	40.34 (**1**)
		Dec	2549/1884/847	11	0	1.97k	302	315	1.04	245 (**1**)	136.87 (**1**)
This Work ($R2^2$MDC)	Virtex-7	Enc	13358/9762/4494	59	0	10.5k	290	182	0.63	408 (1.720)	38.86 (0.963)
		Dec	4632/3401/1540	21	0	3.70k	299	181	0.61	422 (1.722)	114.05 (0.833)

[a]*Equ. Slices,* [b]*one 36K BRAM = two 18K BRAMs,* [c] *M9K in Startix FPGA ≈ 8K BRAM ≈ 56 Slices [10], (**1**) represents the reference design and values in () are the rate of improvement/deterioration.*

the same architectural implementation of [19], we achieve a 74% (i.e., $\frac{1194-316}{1194} \times 100$) reduction in the number of cycles for encryption and a 51% (i.e., $\frac{644-315}{644} \times 100$) reduction for decryption at a cost of comparable DSP resources. Furthermore, our design attains a throughput of 236 Mbps which is 5× higher than that of [19] (calculated 50 Mbps). Whereas, in comparison to the proposed fully parallel implementation of NTT in [20], both of our HNTT based R-LWE accelerators achieve higher speeds and also consume less area resources than their individual NTT cores. The [20] consumed 27k Equ. Slices (14k Slices+128 DSPs×102.4) and takes 1.87 μs (worst case) for only the polynomial multiplication while our proposed designs for the full R-LWE Enc/Dec takes 1.09 μs (R2MDC) and 0.63 μs ($R2^2$MDC) with just 5.85k and 10.5k Equ. Slices respectively. Moreover, our proposed HNTT/HINTT accelerators offer higher design efficiencies, i.e., TPS compared with the state-of-the-art NTT designs for R-LWE [15], [16], [19], [20] and M-LWE [17] with $n = 256$.

Furthermore, it is evident from Tables 3 and 4 that for the same parameter set, the fully paralleled HSPM implementation with 128 DSPs achieves only 60 Mbps throughput (Enc) on the Virtex UltraScale+ whereas the HNTT/HINTT accelerators with two parallel paths, i.e., R2MDC and with four parallel paths, i.e., $R2^2$MDC achieve performance gains of 3.91× (i.e., 236 Mbps) and 6.81× (i.e., 408Mbps) on a Virtex-7 device while consuming less number of DSPs, i.e., 31 DSPs and 59 DSPs, respectively. Hence, it is clear that LBC schemes that can support the NTT (i.e., CRYSTALS-Kyber) will achieve higher speeds as well as better design efficiencies than the schemes supporting the SPM only (i.e., Saber).

VIII. CONCLUSION

In order to support high-speed applications, this work proposes high-performance LBC accelerators (targeting the R-LWE scheme) by exploring both SPM and NTT structures suitable for different parameter sets/LBC schemes, i.e., M-

LWE, M-LWR or LWE. We explore the computational characteristics of SPM to achieve a fully-parallelised and pipelined high-speed structure, which achieves a 5× speed increase while maintaining better design's metrics. In addition, a multi-path FFT structure, i.e., MDC is investigated to design a fully-pipelined radix-2 and radix-2^2 based NTT/INTT architectures, which not only obtain the fastest speeds up to 0.63 μs but also result in highest throughput of 408 Mbps with excellent design's metrics, hence demonstrating their suitability for high-speed applications.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.
[2] D. Moody, "Post-Quantum Cryptography: NIST's plan for the future," in *Proc. Talk Given PQCrypto 16th Conf.*, 2016. [Online]. Available: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf
[3] T. Güneysu and T. Oder, "Towards lightweight identity-based encryption for the post-quantum-secure internet of things," in *Proc. 18th Int. Symp. Qual. Electron. Des.*, 2017, pp. 319–324.
[4] T. Pöppelmann, M. Naehrig, A. Putnam, and A. Macías, "Accelerating homomorphic evaluation on reconfigurable hardware," in *Proc. Cryptographic Hardware Embedded Syst.*, 2015, pp. 143–163.
[5] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, and T. Güneysu, "Practical lattice-based digital signature schemes," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 3, 2015, Art. no. 41.
[6] NIST, "Status report on the first round of the NIST post-quantum cryptography standardization process," 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf
[7] NIST, "Status report on the second round of the NIST post-quantum cryptography standardization process," 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/nistir/8309/final
[8] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2014, pp. 2796–2799.
[9] S. Fan, W. Liu, J. Howe, A. Khalid, and M. O'Neill, "Lightweight hardware implementation of R-LWE lattice-based cryptography," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 2018, pp. 403–406.
[10] W. Liu, S. Fan, A. Khalid, C. Rafferty, and M. O'Neill, "Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 10, pp. 2459–2463, Oct. 2019.
[11] Y. Zhang, C. Wang, D.-S. Kundi, A. Khalid, M. O'Neill, and W. Liu, "An efficient and parallel R-LWE cryptoprocessor," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 67, no. 5, pp. 886–890, May 2020.
[12] V. B. Dang, F. Farahmand, M. Andrzejczak, and K. Gaj, "Implementing and benchmarking three lattice-based post-quantum cryptography algorithms using software/hardware codesign," in *Proc. Int. Conf. Field-Programmable Technol.*, 2019, pp. 206–214.
[13] S. S. Roy and A. Basso, "High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware," *Int. Assoc. Cryptol. Res. Trans. Cryptographic Hardware Embedded Syst.*, vol. 4, pp. 443–466, 2020.
[14] F. Valencia, A. Khalid, E. O'Sullivan, and F. Regazzoni, "The design space of the number theoretic transform: A survey," in *Proc. Int. Conf. Embedded Comput. Syst. Archit. Model. Simul.*, 2017, pp. 273–277.
[15] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2013, pp. 68–85.
[16] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact Ring-LWE cryptoprocessor," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2014, pp. 371–391.
[17] Z. Chen, Y. Ma, T. Chen, J. Lin, and J. Jing, "Towards efficient Kyber on FPGAs: A processor for vector of polynomials," in *Proc. 25th Asia South Pacific Des. Automat. Conf.*, 2020, pp. 247–252.
[18] C. Du, G. Bai, and X. Wu, "High-speed polynomial multiplier architecture for ring-LWE based public key cryptosystems," in *Proc. Int. Great Lakes Symp. Very Large Scale Integr.*, 2016, pp. 9–14.
[19] C. P. Rentería-Mejía and J. Velasco-Medina , "High-throughput Ring-LWE cryptoprocessors," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, pp. 2332–2345, Aug. 2017.
[20] X. Feng, S. Li, and S. Xu, "R-LWE-oriented high-speed polynomial multiplier utilizing multi-lane stockham NTT algorithm," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 67, pp. 556–559, Mar. 2020.
[21] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse," *IEICE Electron. Exp.*, vol. 17, 2020, Art. no. 20200234.
[22] O. Regev, "On lattices, Learning with Errors, random linear codes, and cryptography," in *Proc. 37th Annu. ACM Symp. Theory Comput.*, 2005, pp. 84–93.
[23] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and Learning with Errors over rings," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 1–23.
[24] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Berlin, Germany: Springer, 1981, pp. 80–111.
[25] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2016, pp. 124–139.
[26] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.
[27] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the design of hardware building blocks for modern Lattice-Based Encryption schemes," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2012, pp. 512–529.
[28] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, 1987, pp. 311–323.
[29] P. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, pp. 519–521, 1985.
[30] N. V. Vizev, "Side channel attacks on NTRUEncrypt," Bachelor thesis, Dept. Comput. Sci., Univ. Technol. Darmstadt, Germany, 2007.
[31] J. H. Silverman and W. Whyte, "Timing attacks on NTRUEncrypt via variation in the number of hash calls," in *Proc. Cryptographers' Track RSA Conf.*, 2007, pp. 208–224.
[32] D.-S. Kundi, S. Bian, A. Khalid, C. Wang, M. ONeill, and W. Liu, "AxMM: Area and power efficient approximate modular multiplier for R-LWE cryptosystem," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.
[33] A. Karatsuba, "Multiplication of multidigit numbers on automata," *Sov. Phys. Doklady*, vol. 7, pp. 595–596, 1963.
[34] S. A. Cook, "On the minimum computation time of functions," Ph.D. thesis, Mathematics Dept., Harvard Univ., Cambridge, MA, USA, pp. 51–77, 1966.
[35] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *Proc. Int. Union Radio Sci. Int. Symp. Signals Syst. Electron.*, 1998, pp. 257–262.
[36] P. Massolino, P. Longa, J. Renes, and L. Batina, "A compact and scalable hardware/software co-design of SIKE," *Int. Assoc. Cryptol. Res. Trans. Cryptographic Hardware Embedded Syst.*, vol. 2020, no. 2, pp. 245–271, 2020.

DUR-E-SHAHWAR KUNDI (Member, IEEE) received the MSc and PhD degrees in electrical engineering from the National University of Sciences and Technology (NUST), Karachi, Pakistan, in 2010 and 2016 respectively. She currently joined Queen's University Belfast (QUB) as a postdoctoral research fellow. This work was completed during the postdoctoral fellowship with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), China under the scheme of China Postdoctoral Science Foundation 2019. She is a member of CASCOM Technical Committee of IEEE CAS Society. Her research interests include hardware security, cryptographic engineering and reconfigurable computing.

YUQING ZHANG received the BS degree in applied physics from the Xuzhou Institute of Technology, XuZhou, China, in 2018 and the MS degree in circuits and systems from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, 2021. His research interests include hardware security, cryptographic engineering.

CHENGHUA WANG received the BSc and MSc degrees from Southeast University, Nanjing, China, in 1984 and 1987, respectively. In 1987, he joined the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), where he became a full professor in 2001. He has published six books and more than 100 technical papers in journals and conference proceedings. He is the recipient of more than ten teaching and research awards at the provincial and ministerial level. His current research interests include testing of integrated circuits and systems for communications.

AYESHA KHALID (Member, IEEE) received the BE and MS degrees from the National University of Sciences and Technology (NUST, Pakistan) and the University of Engineering and Technology, (UET-Taxila, Pakistan), respectively. She is currently a lecturer with Queen's University Belfast, NI, U.K. She won a DAAD Scholarship Award for her PhD('15) from RWTH Aachen, Germany. She is a member of VAS Technical Committee of IEEE ISCAS. Her research interests include lattice-based cryptography, embedded systems security, side channel attacks, and cryptographic hardware.

MÁIRE O'NEILL (Senior Member, IEEE) received the MEng and PhD degrees in electrical and electronic engineering from Queen's University Belfast (QUB), UK in 1999 and 2002, respectively. She is currently a regius professor with Electronics and Computer Engineering, acting director of ECIT and director of UKRI in Secure Hardware and Embedded Systems. She has been awarded U.K. Royal Academy of Engineering Silver Medal-2014, IET Young Woman Engineer of Year-2006 and British Female Inventor of Year-2007. She has authored two-research books and has more than 190 leading conference and journal publications. She is a fellow of RAEng and Irish Academy of Engineering and member of Royal Irish Academy.

WEIQIANG LIU (Senior Member, IEEE) received BSc degree in information engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), China and the PhD degree in electronic engineering from Queen's University Belfast (QUB), UK. in 2006 and 2012, respectively. He joined NUAA, since 2013 where he is currently a professor and vice dean with the College of Electronics and Information Engineering as well as College of Integrated Circuits. He has published a research book by Artech House and more than 160 leading journal and conference papers. He has been awarded an Excellent Young Scholar Award by NSFC in 2020. His research interests include approximate computing, hardware security and VLSI design for DSP, cryptography and mixed signal IC design.