2022 IEEE 40th International Conference on Computer Design (ICCD) | 978-1-6654-6186-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCD56317.2022.00083

HPMA-Saber: High-Performance Polynomial Multiplication Accelerator for KEM Saber

Pengzhou He, Tianyou Bao, Yazheng Tu, and Jiafeng Xie
Deparmtne of Electrical and Computer Engineering, Villanova University, Villanova PA 19085 USA
Email: {phe,tbao,ytu1,jiafeng.xie}@villanova.edu

***Abstract*—The recent research in post-quantum cryptography (PQC) field has gradually switched to efficient implementation of PQC algorithms on hardware platforms. As polynomial multiplication is typically one of the critical operations within lattice-based PQC, its hardware acceleration has drawn significant attention from the research community recently. We propose a high-speed processing strategy to construct a new High-performance Polynomial Multiplication Accelerator (HPMA) for key encapsulation mechanism (KEM) Saber. Firstly, we have given a detailed mathematical derivation to obtain a low-latency processing algorithm for Saber polynomial multiplication. Then, we have innovatively used the derived the proposed algorithm to construct a new structure HPMA for FPGA implementation. Lastly, we have demonstrated the superior performance of the proposed HPMA-Saber by comparing with state-of-the-art works. The proposed design strategy is highly efficient and the obtained results can be useful for the PQC research community.**

***Index Terms*—High-performance, key encapsulation mechanism (KEM) Saber, polynomial multiplication accelerator, post-quantum cryptography (PQC).**

I. Introduction

Along with the rapid progress in quantum computing, more attention has switched to post-quantum cryptography (PQC) research and development as the current public-key cryptosystems are proved to be vulnerable to the attacks launched from powerful quantum computers executing Shor's algorithm [1].

The learning-with-rounding (LWR) is a variant of the learning-with-errors (LWE) problem, which obtains the error term by a rounding operation rather than the random distribution [2]. The Module-LWR (MLWR) is a module variant of the LWR and has been used to build cryptosystems for PQC standardization, e.g., key encapsulation mechanism (KEM) Saber, one of the NIST 3rd round public-key finalists [2], [3].

Prior Works. As the major arithmetic operation of KEM Saber, polynomial multiplication plays an essential role in determining the overall performance of the Saber cryptoprocessor. Recent hardware implementation of polynomial multiplication for Saber includes (i) an early paper used the Toom-Cook method to obtain efficient hardware-software co-design [4]; (ii) a full-hardware polynomial multiplier for the Saber coprocessor [5]; (iii) optimized hardware polynomial multipliers were then proposed in [6]; (v) a new cyclic-row originated processing (CROP) technique based polynomial multiplier was then presented in [7]; (vi) high-performance polynomial multipliers for Saber were also presented in [8]; (vii) a dual-CROP based polynomial multiplier for Saber was introduced in [9].

Major Challenges. Major challenges include: (i) Few alternative efforts have been made besides the traditional schoolbook algorithm; (ii) the existing hardware structures do not demonstrate high operational frequency; (iii) some implementations do not consider the practical application setup, e.g., input/output bit-width needs to be set the same as the memory in/out port and the module scheme feature of Saber.

Major Contributions. In this paper, we propose to design a new **H**igh-performance **P**olynomial **M**ultiplication **A**ccelerator for KEM **Saber** (HPMA-Saber). Main contributions include:

- We have presented a mathematical derivation to obtain a new polynomial multiplication algorithm for Saber.
- We have designed the proposed polynomial multiplier accelerator with practical input/output setup.
- We have provided sufficient comparison to demonstrate the superior performance of the proposed accelerator.

II. Preliminaries

KEM Saber. Saber is an adaptive Chosen Ciphertext Attack (IND-CCA) secure KEM, which is based on the hardness of the MLWR problem to achieve both classical and quantum security [2]. Saber was constructed as a Chosen Plaintext Attack (CPA) secure public-key scheme. Then, CCA Saber KEM was built through the Fujisaki-Okamoto transformation [10]. Interested readers can go to the paper [2] for detailed information.

Polynomial Multiplication for Saber. The most complicated operation within each phase of Saber is the polynomial multiplication over the ring $\mathbb{Z}_l/(x^N+1)$ (l is either q or p). Without loss of generality, we can define that the polynomial multiplication of Saber involves one input polynomial with 13-bit coefficients and another polynomial has 4-bit coefficients (sampled secrets), while the output coefficient is 13-bit (the polynomial with 10-bit coefficients is also covered here).

III. Algorithmic Operation

Consideration. Existing polynomial multiplication algorithms include Toom-Cook [4], [11], KA [12], and schoolbook methods (or similar) [5]. Interestingly, based on reported results, the schoolbook algorithm probably is the most effective one since the polynomial multiplication for Saber has "unbalanced" bit-widths for input polynomials. For instance, the deploying of fast algorithm such as KA can actually increase the bit-width due to the addition related pre-processing operations, i.e., [-4,4] will increase to [-8,8], which is not favorable for hardware implementation. Therefore, we use schoolbook algorithm for the proposed polynomial multiplier in this paper to maintain efficient implementation complexity on the hardware platform, but with a different algorithmic operation/strategy.

2576-6996/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCD56317.2022.00083

Proposed Mathematical Derivation Strategy. The existing polynomial multiplications [5]–[7], deployed the traditional schoolbook algorithm that one polynomial is used as the operational operand (combined with modulo operation) while the other is fed into through serial format. This setup is not ideal for high-performance applications since it requires N cycles for accumulation-based computation and then also needs N cycles for output delivery [7]. In this work, we do computation and output delivery at the same time to save processing latency. Meanwhile, to obtain a high-speed operation, we made the computation of the polynomial multiplication to be able to executed through polynomial-wise based operations.

Following the above strategy, we define the polynomial multiplication for Saber as (the polynomial multiplication with an input polynomial of 10-bit coefficients is also included):

$$W = GD \bmod f(x), \tag{1}$$

where $f(x) = x^N + 1$, $W = \sum_{i=0}^{N-1} w_i x^i$, $G = \sum_{i=0}^{N-1} g_i x^i$, and $D = \sum_{i=0}^{N-1} d_i x^i$ (g_i, d_i, and w_i are 13-bit, 4-bit, and 13-bit integers over $\mathbb{Z}_q$, respectively). We can have

$$\begin{aligned} w_0 =& d_0 g_0 + (-d_{N-1}) g_1 + \cdots + (-d_1) g_{N-1}, \\ w_1 =& d_1 g_0 + (d_0) g_1 + \cdots + (-d_2) g_{N-1}, \\ & \cdots \ \cdots \ \cdots \\ w_{N-2} =& d_{N-2} g_0 + d_{n-3} g_1 + \cdots + (-d_{N-1}) g_{N-1}, \\ w_{N-1} =& d_{N-1} g_0 + d_{N-2} g_1 + \cdots + d_0 g_{N-1}, \end{aligned} \tag{2}$$

where each w_i (i from $N-1$ to 0) is addition result of circularly shifted coefficients of D (with one coefficient's sign inverted) multiplied with the corresponding coefficients of G, respectively. Then, we can derive the proposed algorithm for polynomial multiplication of Saber as:

Algorithm 1: Proposed polynomial multiplication algorithm for KEM Saber

Input : G and D are integer polynomials; // where g_i and d_i are 13-bit and 4-bit coefficients according to Saber setup.

Output: $W = GD \bmod (x^N + 1)$; // where w_i is 13-bit coefficient over $\mathbb{Z}_q$.

Initialization step

1 Make ready the inputs G and D.

Main step

2 **for** $i = N-1$ *to* 0 **do**

3 | **for** $j = 0$ *to* $N-1$ **do**

4 | | w_i=$\sum_{j=0}^{N-1} D_j^{(i)} g_j$.

5 | **end**

6 **end**

Final step

7 Serially deliver all the coefficients of output W;

which follows the proposed derivation strategy that the final output is calculated and delivered out once per cycle. Note that $D^{(i-1)}$ can be obtained from $D^{(i)}$ by circular shifting of all the coefficients with one sign inverted, and the detailed operation can be seen in the following hardware structure section.

IV. Accelerator: HPMA-Saber

Background Overview and Proposed Hardware Design Strategy. The existing hardware structures for the polynomial multiplier of Saber calculate the output values in a parallel format [6], [7], and have to go through a parallel-in serial-output buffer to deliver these coefficients into the RAM. We hence decide to propose a new polynomial multiplication where all the output results can be processed in a serial format to save the unnecessary delay cycles.

Overall Architecture. The proposed hardware accelerator (HPMA-Saber) for polynomial multiplication of Saber is shown in Fig. 1 (based on Algorithm 1). The values sampled from the sampler are represented by the two's complement form (refer to G here), which is different from the ones in the existing designs [5]–[7]. This setup, however, provides more generality of the proposed HPMA-Saber for deploying in different application environments. As seen from Fig. 1, the proposed HPMA-Saber has three components, namely the input loading component (highlighted with brown), the main computational component (highlighted with blue), and the control component (highlighted with red). The details of these components are described below.

Input Loading Component. The input loading component consists of two circular shift-registers (CSRs) for inputs D and G, respectively. As indicated by Algorithm 1 that $D^{(i-1)}$ can be obtained from $D^{(i)}$ by circular shifting of all the coefficients with one sign inverted. Note that the input D is originally represented in two's complement format and is transferred into the sign-magnitude form to facilitate following processing. Apart from the regular N number of registers, one 2-to-1 MUX and one sign cell are also included in the proposed CSR, where the MUX functions to load the input coefficients into the corresponding registers and then switches to another channel to circularly shift the positions of the values of D in CSR. The CSR for G does not need further circular shifting after all the coefficients are loaded into the CSR, i.e., the N parallel output values of the CSR remain stable.

Main Computational Component. The main computational component consists of N parallel point-wise multipliers, where each point-wise multiplier (Fig. 2) executes the multiplication between related output from CSR (for G) and corresponding output from CSR (for D) according to Step 4 of Algorithm 1. We have used the MUX-based strategy to design the multiplier following similar designs in the literature [5], [6].An adder tree is used to produce the final output in a serial format. To facilitate the high-performance operation, we can insert layers of registers into the adder tree, as indicated in Fig. 1.

Control Component. The control component (control unit) mainly consists of a finite state machine (FSM) to coordinate the overall operation of the polynomial multiplier, i.e., the operation can be split into two stages, i.e., loading and computation. The loading stage ends when all the coefficients are loaded into two CSRs; while the computation stage starts to produce the output values after necessary cycles (determined by the adder tree). Note that the control unit will need to be updated to match the practical operation, i.e., assumed to be deployed inside the Saber cryptoprocessor.

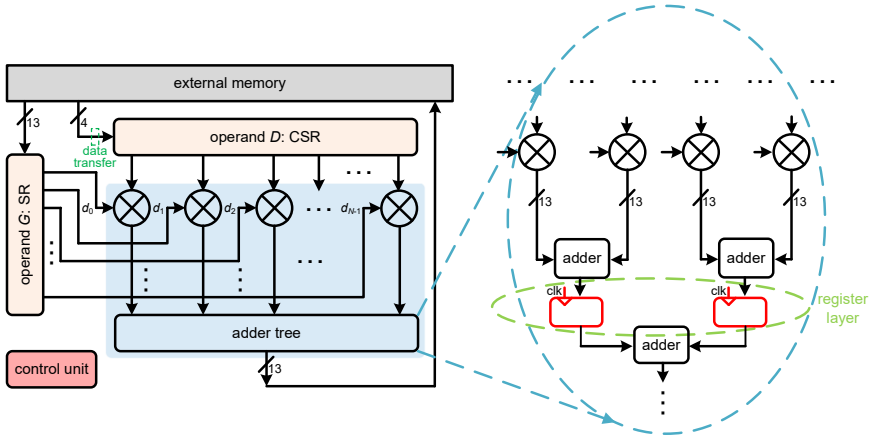

Fig. 1. The architectural details of the proposed HPMA for KEM Saber. CSR: circular shift-register. Data transfer refers to the transferring of two's complement to the sign-magnitude format.

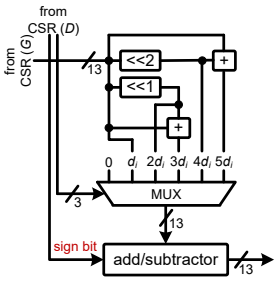

Fig. 2. The structural details of the point-wise multiplier.

Overall Operation and Discussion of Advantage. The accelerator presented in Fig. 1 can be very efficient as it can calculate and deliver the results almost at the same time. For instance, for $N = 256$ and every layer is inserted with registers, the overall computation and output delivery need only $256+8 = 264$ cycles, while the existing polynomial multipliers under the ideal setup like the ones proposed in [6] all require 512 cycles. Meanwhile, as the adder tree is fully pipelined, the proposed polynomial multiplier will have a very small critical-path (high frequency), which indicates that the proposed accelerator is suitable for high-performance applications.

Practical Architectural Setup. The architecture shown in Fig. 1 is an ideal setup, e.g., the two inputs are set as 4-bit and 13-bit, respectively, while the output is set as 13-bit and can be smoothly delivered out along with the computation process. We need an extra setup on the input and output processing units for practical applications where the external memory usually has single 64-bit in/out ports. As shown in Fig. 3, we have updated the input and output word-length to 64-bit, and the original CSR and SR have been updated to the new buffers as well as the output buffer (the data transfer for D, from two's complement to sign-magnitude, is also updated to 64-bit).

The proposed buffer contains $4 \times N - 1$ register cells, where each cell is a 1-bit register combined with a 2-to-1 MUX and each cell is connected with the correct signals for correct operation. The input data of the first 13 register cells are from the input or the output of the sign cell (sign inverting according to Algorithm 1). The input data of the first 14th-64th register

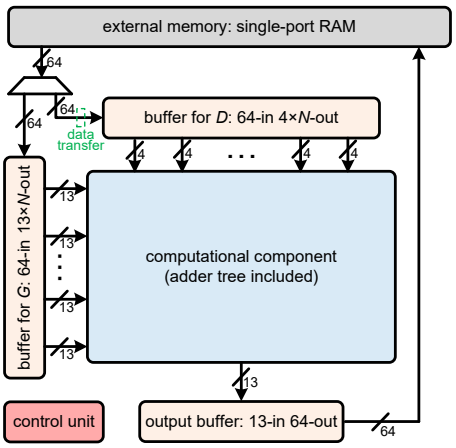

Fig. 3. Practical setup of the proposed HPMA-Saber (based on the single-port external memory).

cells are from the input or from the previous 13 register cells. The rest register cells only take data from the previous 13 register cells or the previous 64 register cells. Once all the input data are loaded into the registers, the values contained in the registers will be circularly shifted once per cycle based on the bit-width of 13. The same design strategy applies to the buffer for G and the output buffer.

Finally, the control unit follows traditional FSM setup to update the control unit. There are in total five stages involved within this redesigned FSM, i.e., "reset", "load", "calculation", "output", and "done". Note that the clock cycles are calculated based on the assumption that each layer of the adder tree is inserted with registers.

V. Implementation & Comparison

In this section, analysis on the impact of inserting register layers into the adder tree is provided first. Then, we will give a detailed implementation process for the proposed HPMA as well as the comparison with the state-of-the-art designs. Finally, some discussions and future work will be given.

Implementation on the FPGA Platform. We have also implemented the HPMA-Saber on the FPGA platform with

TABLE I
FPGA IMPLEMENTATION RESULTS (ULTRASCALE+ DEVICE)

design	LUT	FF	CLB	Fmax	DSPs	latency[!]	power	delay	ADP[0]
[5][1]	17,406	5,069	2712	250	0	256	-	1.02	17,823.74
[12][2]	13,735	4,486	-	160	85	83	-	0.52	*
[8] (FIR) [3]	16,971	8,755	-	250	0	511	-	2.04	34,688.72
[8] (Fast.2)[3]	25,831	12,850	-	250	0	255	-	1.02	26,347.62
[8] (Fast.4)[3]	35,306	19,143	-	250	64	127	-	0.51	*
HPMA-Saber	26,884	14,524	4,419	441	0	264	807	0.60	16,093.82

Unit for Fmax (maximum frequency): MHz; unit for power: mW.
[!]: latency refers to the computation time. Due to the architectural setup, the adder tree delay time is also included in our proposed HPMA-Saber.
[0]: ADP refers to area-delay product, which is ADP=#LUT×delay (since some of the existing designs do not report the CLB usage).
[1]: the CLB is obtained from the released source code, the output delivery cycle is not listed here.
[2]: based on Karatsuba algorithm (involves smaller latency but with larger area usage).
[3]: based on filtering based fast algorithm (Fast.4 has smaller latency but with larger resource usage).
*: as these designs use large number of DSPs, we do not calculate their ADP here. For a reference, however, one DSP typically can be seen as equivalent to 102.4 slices [13], which indicates that the designs of [8], [12] have significantly larger CLBs than the proposed one as they need at least 8,704 and 6,554 equivalent CLBs for DSPs.

practical setup. We have coded the final architectural with VHDL ($N = 256$) [1]. The coded design covers all three security ranks of KEM Saber, i.e., the values of D are set as [-5,+5]. The target device was set to Xilinx UltraScale+ XCZU9EG-FFVB1156-2 FPGA. To obtain a high frequency, we have inserted the registers into the adder tree at every layer, i.e., for $N = 256$, we have inserted $\log_2 N = 8$ layers of registers.

It is clear that the proposed HPMA-Saber obtains the best area-time performance among all the reported designs. For instance, when compared with the existing design of [5], the proposed accelerator has at least 9.71% less ADP than the one of [5]. Meanwhile, when compared with the designs of [8], [12], the proposed accelerator involves significantly less resource usage, especially on the equivalent number of CLBs, as indicated in Table I. Lastly, one has to mention that due to the use of pipelining register layers in the adder tree, the proposed HPMA-Saber achieves highest operational frequency among all the reported designs, which indicates that the proposed design suits well high-performance applications.

Discussion. While the primary goal of this paper is to design HPMA-Saber, other aspects of research, such as side-channel attacks, are out of the scope of this paper. Nevertheless, the proposed accelerator has a stable critical-path and hence is resistant to timing attacks. Future work may focus on the building of high-performance Saber cryptoprocessors.

VI. CONCLUSION

This paper has presented a new HPMA for KEM Saber on the FPGA platform. We have conducted three layers of efforts to obtain the proposed work. First, an algorithmic derivation process is provided to formulate the polynomial multiplication into the desired form. Then, a detailed architectural design process has been presented to introduce the proposed HPMA. Finally, a thorough implementation based analysis and comparison have been given to demonstrate the effectiveness of the proposed HPMA. Future work will focus on extending the presented HPMA into high-speed LWR-based cryptoprocessors, and it is anticipated that the proposed work will be useful for PQC research and development.

[1]The source code is available at [14]

ACKNOWLEDGEMENT

The work of J. Xie was supported by NIST-60NANB20D203 and in part by NSF SaTC-2020625.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[2] J.-P. D'Anvers, A. Karmakar, S. S. Roy, F. Vercauteren, J. Mera, M. Beirendonck, and A. Basso, "SABER: Mod-LWR based KEM (round 3 submission),"

[3] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM," in *International Conference on Cryptology in Africa*, pp. 282–305, Springer, 2018.

[4] J. M. B. Mera, F. Turan, A. Karmakar, S. S. Roy, and I. Verbauwhede, "Compact domain-specific co-processor for accelerating module lattice-based KEM," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[5] S. S. Roy and A. Basso, "High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 443–466, 2020.

[6] A. Basso and S. S. Roy, "Optimized polynomial multiplier architectures for post-quantum KEM Saber," *Design Automation Conference (DAC)*, pp. 1–6, 2021.

[7] J. Xie, P. He, and C.-Y. Lee, "Crop: Fpga implementation of high-performance polynomial multiplication in saber kem based on novel cyclic-row oriented processing strategy," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 130–137, IEEE, 2021.

[8] W. Tan, A. Wang, Y. Lao, X. Zhang, and K. K. Parhi, "Low-latency vlsi architectures for modular polynomial multiplication via fast filtering and applications to lattice-based cryptography," *arXiv preprint arXiv:2110.12127*, 2021.

[9] E. Carter, P. He, and J. Xie, "High-performance polynomial multiplication hardware accelerators for kem saber and ntru," *Cryptology ePrint Archive*, 2022.

[10] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki-Okamoto transformation," in *Theory of Cryptography Conference*, pp. 341–371, Springer, 2017.

[11] A. Karmakar, I. Verbauwhede, *et al.*, "Saber on ARM. CCA-secure module lattice-based key encapsulation on ARM," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 243–266, 2018.

[12] Y. Zhu, M. Zhu, B. Yang, W. Zhu, C. Deng, C. Chen, S. Wei, and L. Liu, "LWRpro: An energy-efficient configurable crypto-processor for Module-LWR," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 3, pp. 1146–1159, 2021.

[13] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. O'Neill, and W. Liu, "An efficient and parallel r-lwe cryptoprocessor," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 886–890, 2020.

[14] J. Xie, "Security and Cryptography (SAC) Lab." https://www.ece.villanova.edu/~jxie02/lab/.