Time-Optimal Design of Finite Field Arithmetic for SIKE on Cortex-M4

Mila Anastasova[1], Reza Azarderakhsh[1], and Mehran Mozaffari Kermani[2]

[1] Computer and Electrical Engineering and Computer Science Department and I-SENSE at Florida Atlantic University, Boca Raton, FL, USA
(manastasova2017, razarderakhsh)@fau.edu
[2] Computer Engineering and Science Department at University of South Florida, Tampa, FL, USA,
mehran2@usf.edu

Abstract. The advances in quantum technologies and the fast move toward quantum computing are threatening classical cryptography and urge the deployment of post-quantum (PQ) schemes. The only isogeny-based candidate forming part of the third round of the standardization, the Supersingular Isogeny Key Encapsulation (SIKE) mechanism, is a subject of constant latency optimizations given its attractive compact key size lengths and, thus, its limited bandwidth and memory requirements. In this work, we present a new speed record of the SIKE protocol by implementing novel low-level finite field arithmetics targeting ARMv7-M architecture. We develop a handcrafted assembly code for the modular multiplication and squaring functions where we obtain 8.71% and 5.38% of speedup, respectively, compared to the last best-reported assembly implementations for p434. After deploying the finite field optimized architecture to the SIKE protocol, we observe 5.63%, 3.93%, 3.48%, and 1.61% of latency reduction for SIKE p434, p503, p610, and p751, respectively, targeting the NIST recommended STM32F407VG discovery board for our experiments.
Key Words: Supersingular Isogeny Key Encapsulation (SIKE), Post-Quantum Cryptography (PQC), ARM Cortex-M4

1 Introduction

Public key cryptography is essential for the security and integrity of data conveyed across an unsecured channel. The classical cryptographic protocols, however, such as RSA and ECC relying on the difficulty of factoring large prime numbers and the Elliptic Curve Discrete Logarithm Problem (ECDLP), are vulnerable to quantum attacks. Thus, a transition to post-quantum robust protocols is required to offer secure data transmission in the era of large-scale quantum computers. This is the reason for the National Institute of Standards and Technology (NIST) [1] to initiate a standardization process for evaluation and optimization of post-quantum primitives. Forming part of the alternate group of candidates, the Supersingular Isogeny Key Encapsulation (SIKE) [2] mechanism is the candidate with the smallest public key and ciphertext sizes, thus it is suitable for deploying it in scenarios with limited bandwidth and memory resources.

The incorporation of Internet of Things (IoT) devices into daily life improves the quality of life. However, the restricted resources on low-end target systems

make it difficult to accommodate PQ candidates. Thus, researchers and engineers have prioritized the development of target-specific handcrafted assembly code to minimize their latency by optimal resource utilization. In this work, we focus on the NIST recommended platform for performance evaluation on embedded devices - the ARMv7-M Cortex-M4-based STM32F407VG microcontroller.

Contribution. In this paper, the time-optimal architecture for finite field arithmetic operations, base of SIKE, is proposed for the resource-constrained ARM Cortex-M4 processor. The breakdown of our contributions is as follows:

- We present the first multi-precision multiplication, squaring, and reduction architecture based on a hybrid method that integrates both product- and operand-scanning approaches in the inner multiplication loop design. By adopting a hybrid method to the inner loop's architecture, we are able to improve resource use, resulting in a record-setting execution time.

- We propose novel design for the multi-precision multiplication, squaring and reduction routines, achieving new speed record for the modular functions compared to the previous best-reported results [3]. We observe 8.71%, 5.99%, 4.46%, and 2.04% of latency reduction for the execution of modular multiplication based on prime lengths of 434-, 503-, 610-, and 751-bits, respectively. We achieve 5.38%, 6.43%, 14.64%, and 6.42% of speedup compared to the counterparts in [3] for the modular squaring routine.

- We integrate the suggested multi-precision multiplication, squaring and reduction routines in the SIKE implementation and we obtain more than 5.6% of speedup for SIKEp434. We report 3.93%, 3.48%, and 1.61% of latency reduction for SIKEp503, SIKEp610, and SIKEp751, respectively.

2 Related Work

SIKE is the PQ primitive requiring the smallest communication latency, allowing for rapid data transmission even when bandwidth is constrained. In addition, the identical operation components and finite field arithmetic of SIKE and the widely used classical ECC schemes enables a straightforward transition to hybrid schemes, which remain compliant with government and industry regulations and offer classical- and PQ- resilience to the users. However, the SIKE's computational delay is considerable compared to traditional cryptosystems and other PQ alternatives, hence, the primary objective for the isogeny-based PQ primitive has been the reduction of its latency.

The pyramidal structure of the isogeny-based protocol permits optimization of its many levels. As a result, there have been several research and engineering groups devoted to optimizing the higher-level isogeny optimizations of the SIKE protocol in order to find an optimal solution for the calculation of the heavy isogeny maps [4,5,6,7]. The execution latency, however, of SIKE, remain relatively high and a target-specific implementation of the low-level computations is required. The authors in [8] base their work on the AVX-512 extended Advanced Vector Extensions SIMD instruction set for x86 instruction set architecture, where their optimized design can be deployed on the widely used Xeon Phi x200and Skylake-X CPUs. Another line of research demonstrates ongoing efforts to optimize SIKE on ARMv8 architecture deploying the NEON advanced

Key Generation	**Encapsulation**	**Decapsulation**
Input: -	*Input:* pk_A	*Input:* s, sk_B, pk_B, c
Output: s, sk_A, pk_A	*Output*: c, ss	*Output*: ss
1. $sk_A \in_R \mathbb{Z}/2^{e_A}\mathbb{Z}$	1. $m \in_R \{0,1\}^t$	1. $\phi'_A : E_B \to E_{BA}$ with
2. $\phi_A : E_0 \to E_A$ with	2. $r = H(m\|\|pk_A) mod 3^{e_B}$	$ker(\phi'_A) =$
$ker(\phi_A) = \langle P_A + [sk_A]Q_A \rangle$	3. $\phi_B : E_0 \to E_B$ with	$\langle \phi_B(P_A) + [sk_A]\phi_B(Q_A) \rangle$
3. $pk_A =$	$ker(\phi_B) = \langle P_B + [r]Q_B \rangle$	2. $m' = c_1 \oplus K(j(E_{BA}))$
$(E_A, \phi_A(P_B), \phi_A(Q_B))$	4. $pk_B =$	3. $r' = H(m'\|\|pk_A) mod 3^{e_B}$
4. $s \in_R \{0,1\}^t$	$\{E_B, \phi_B(P_A), \phi_B(Q_A)\}$	4. $\phi''_A : E_0 \to E_{B'}$ with
	5. $\phi'_B : E_A \to E_{AB}$ with	$ker(\phi''_A) = \langle P_B + [r']Q_B \rangle$
	$ker(\phi'_B) =$	5.
	$\langle \phi_A(P_B) + [r]\phi_A(Q_B) \rangle$	$pk'_B = \{E_{B'}, \phi''_A(P_A), \phi''_A(Q_A)\}$
	6. $c = (pk_B\|\|K(j(E_{AB})) \oplus m)$	6. IF $pk'_B = pk_B$
	7. $ss = (J(m\|\|c))$	$ss = (J(m'\|\|c))$
		ELSE $ss = (J(s\|\|c))$

Fig. 1: SIKE algorithm [2]. H, K and J denote hash functions. $p = 2^{e_A}3^{e_B} - 1$, $E_0/\mathbb{F}_{p^2}$, $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ are public parameters.

Single Instruction Multiple Data (SIMD) architecture extension for the A-profile and R-profile processors technology [9,10,11,12,13]. Additionally, there is an exhaustive work on SIKE targeting reconfigurable architecture [14,15,16,17].

In addition to the several high-end target-specific implementations, the focus has remained on low-end devices due to the difficulty of fitting such a computationally intensive protocol into devices with limited resources. The authors in [18,3,19] provide implementation solutions for the low-level finite filed arithmetic of SIKE and compressed SIKE and achieve a record speedup on the target platform, running the SIKE protocol in 139MCCs for security Level I. The nature of the SIKE protocol, based on supersingular elliptic curves defined over a quadratic extension of a finite field, benefits from the previous work on low-level arithmetic for classical protocols such as RSA and ECC. Thus, implementation strategies proposed for ECC optimizations [20,21,22,23,24] are applicable to the isogeny-based PQ protocol if not targeting a specific finite field size. Nonetheless, despite the low computational cost and extremely small key sizes, this classical crypto scheme will not ensure secure data transmission in the case of large quantum computers, necessitating the NIST post-quantum standardization effort and a substantial effort on implementing hybrid systems providing both classical and quantum security [25,26,27,28].

3 Preliminaries

In the next part, we provide a comprehensive overview of the SIKE protocol and the primary characteristics of the Cortex-M4 platform. We refer the readers to [2] and [29] for further details.

3.1 SIKE

The Supersingular Isogeny Key Encapsulation process provides post-quantum resilience since it relies on difficult mathematical issues that are thought to be

resistant to large-scale quantum computers. The SIKE protocol, which is based on pseudorandom walks on isomorphic graphs, assures that both communication parties reach a shared secret based on a curve $j-$invariant.

An elliptic curve is defined as $E_{a,b}/\mathbb{F}_{p^2} : ay^2 = x^3 + bx^2 + x$. SIKE elliptic curve elements are defined by two coordinated x, y defined over the quadratic extension of a finite field such that $x, y = ai + b$ with $a, b \in \mathbb{F}_p$. The value of the prime p has the shape of $p = 2^{e_A}3^{e_B} \pm 1$ where the values of e_A and e_B vary depending on the security level of the deployed protocol.

Two elliptic curves E and E' are said to be isogenous if they are of the same order, thus $\#(E) = \#(E')$. An isogeny map φ is defined as $\varphi : E_0 \rightarrow E_0/\langle G \rangle$ with G the kernel of the isogeny. The isomorphism is a mapping function, which ensures that the original and projection curve feature the same $j-$invariant. After executing SIKE protocol, both communication parties reach the same isomorphic class, thus, deduce the same value for a $j-$invariant Fig. 1.

3.2 ARMv7-M Architecture

The fast expansion of the Internet of Things necessitates low-cost devices that can be integrated into real-time embedded systems. For resource-constrained devices, NIST has designated the STM32F407VG as the target microcontroller. The PQM4 library [30] was created as a consequence of the standardization effort low-end recommended target. The STM32F407VG microcontroller features 1MB of flash memory and 192KB of RAM.

The ARMv7-M Cortex-M4 platform has 16 32-bit GPRs and 32 32-bit Floating-Point Registers (FPRs). Two GPRs are designated for the Stack Pointer and the Program Counter. The whole of FPR is available to the user, and data transmission between GPRs and FPRs is instantaneous via `VMOV` instruction, unlike data accessing instructions like `LDR` and `STR` which might cause stall cycles if not correctly planned.

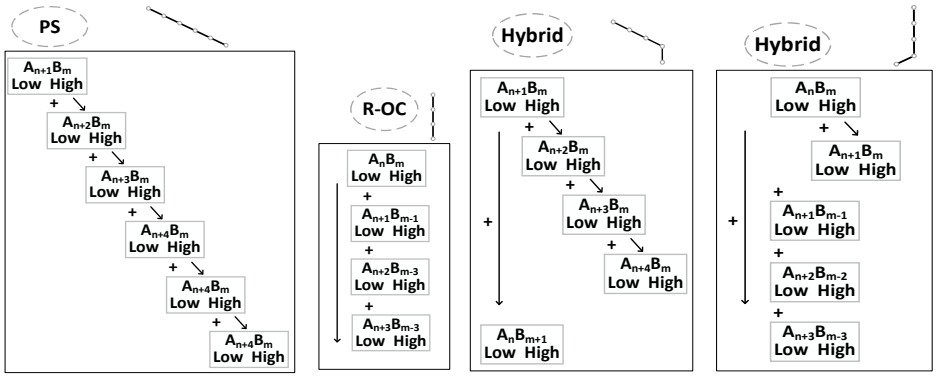
The powerful Multiply ACcumulate (MAC) instructions allow the execution of up to three operations in a single clock cycle; hence, we integrate the use of `UMULL` and `UMAAL` in our work.

4 Proposed Design for SIKE field arithmetic

In this section, we present the proposed the multi-precision multiplication, squaring, and reduction techniques and analyze their performance compared to the counterparts.

4.1 Notation

In this paper, a novel implementation technique for multi-precision multiplication, square, and reduction is proposed and deployed into the SIKE PQ protocol. To our knowledge, our multi-precision architecture is the first to integrate hybrid Operand Scanning (OS) and Product Scanning (PS) techniques in the inner multiplication loop. The authors of [18] present a hybrid multi-precision arithmetic approach in which the outer and inner multi-precision loops, respectively, implement non-hybrid OS and PS techniques. The work presented by [3] presents improved design based on the same independent PS inner and OS outer loops



(a) Product Scanning (PS). (b) R-OC, identical to OS. (c) Hybrid (4 PS&1 OS). (d) Hybrid (1 PS&4 OS).

```
VMOV R11,S12 R12                VMOV R1,S12 R12             VMOV R0,S12 R12             VMOV R0,S12 R12
VMOV R7,S30 a12                 LDR R6,[R0,#(4*12)] a12
UMAAL R0,R11,R1,R7 a0a12        UMAAL R14,R10,R5,R7 a0a12   UMAAL R0,R10,R2,R6 a6b6 UMAAL R0,R12,R4,R6 a9b3
UMAAL R10,R11,R2,R7 a1a12       UMAAL R14,R11,R4,R8 a1a11   UMAAL R11,R10,R3,R6 a7b6 UMAAL R0,R14,R3,R7 a8b4
UMAAL R12,R11,R3,R7 a2a12       UMAAL R14,R12,R3,R9 a2a10   UMAAL R12,R10,R4,R6 a8b6 UMAAL R12,R14,R5,R6 a10b3
UMAAL R9,R11,R4,R7 a3a12        UMAAL R1,R14,R2,R6 a3a9     UMAAL R14,R10,R5 a9b6       VMOV R6,S6 b6
UMAAL R14,R11,R5,R7 a4a12                                   LDR R7,[R8,#4*7] b7         UMAAL R0,R10,R2, R8 a7b5
UMAAL R8,R11,R6,R7 a5a12                                    UMAAL R0,R9,R1,R7 a5b7      UMAAL R0,R9,R1,R6 a6b6
VMOV S12,R0 R12                 VMOV S12,R1 R12             VMOV S12,R0 R12             VMOV S12,R0 R12
```

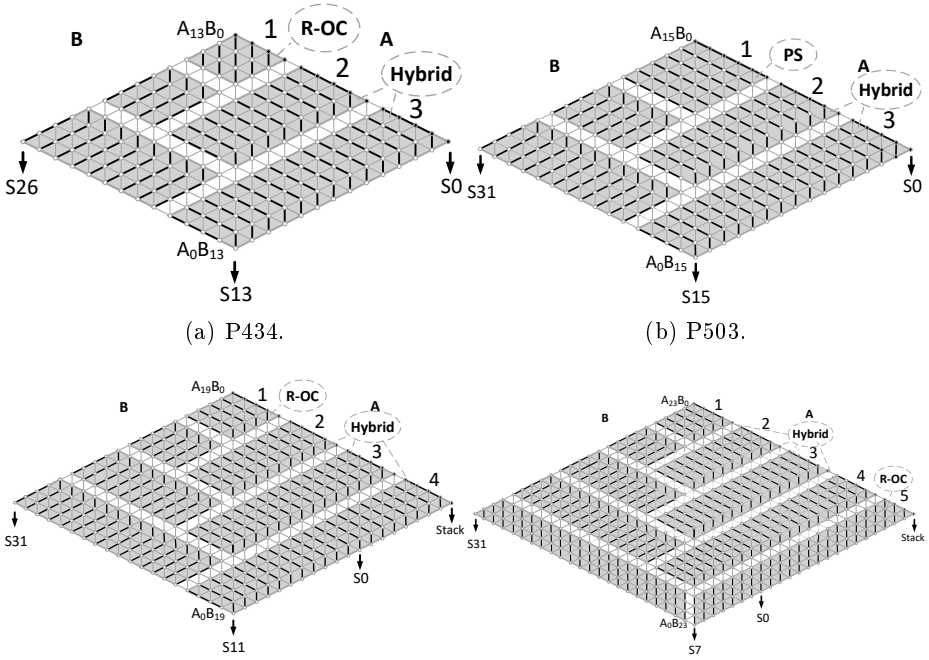
Fig. 2: Deployed list of inner multi-precision loop execution flows along with the associated assembly instruction set.

strategies. In this paper, we present the first design that combines SO and PS techniques within the implementation of the inner multi-precision loop and OS techniques within the outer loop.

We employ rhombus representation to visualize our arithmetic design flow. The varied inner loop designs are shown in Fig. 2 along with the respective assembly code design. While the PS and Refiend-Operand Caching (R-OC) strategies are not novel, we combine them to create a more optimum and time-efficient hybrid inner loop execution phase.

The PS Fig. 2a requires larger number of memory accesses for accessing the previously calculated partial results, however, allow to compute larger set of partial values per iteration. It facilitates the register set use by loading a single word from one operand and large set of words from the second operand; hence, it allows to increase the size of the inner loop iteration. The R-OC Fig. 2b uses an Operand Scanning technique in the inner multiplication loop, which allows to reuse the values of the operands when modifying the increasing and decreasing indices of operand words (i.e., when changing the direction of the flow).

In this work, we benefit from both techniques and propose a hybrid design Fig. 2c and Fig. 2d where we increase the size of the loop iteration and still keep the partial results values in the register set without need of moving them into



(a) P434. (b) P503.

(c) P610. (d) P751.

Fig. 3: Proposed architecture for the implementation of multi-precision multiplication for all SIKE primes.

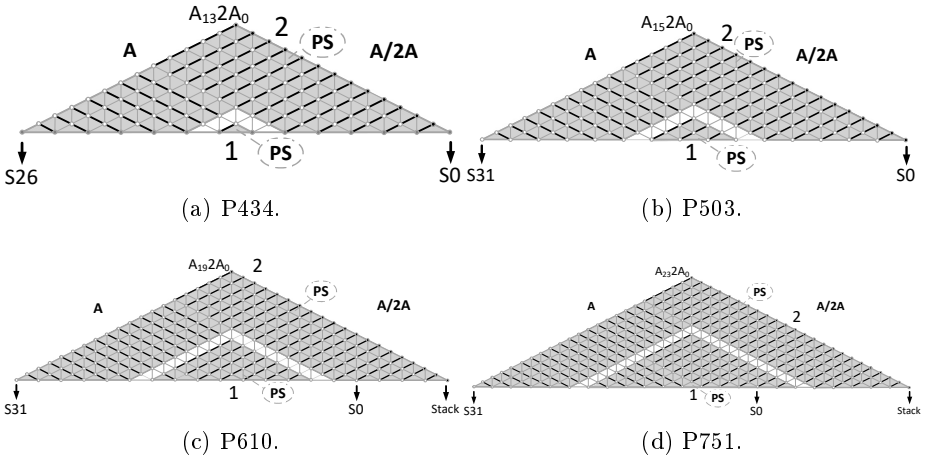
the memory. In Fig. 2c, four partial results are generated based on PS and one based on OS. In Fig. 2d, four R-OC and one PS steps are used. The former hybrid design uses requires a single register for storing the second operand, minimizing the register use. The latter hybrid design reserved three register to the second operand, which benefits the R-OC principal in the mid-point of each row.

4.2 Multi-precision Multiplication

Optimizing multi-precision multiplication is critical for cryptographic protocols, especially when costly calculations like isogeny maps among supersingular elliptic curves are required.

We propose efficient multi-precision multiplication for all four SIKE security levels, based on prime lengths of 434, 503, 610, and 751 bits. We used a novel hybrid method in the inner multiplication loop to improve row size and hence reduce memory accesses for partial values. Depending on the prime number, we apply a combination of the three inner iteration designs. The purpose of this effort is to increase row size, hence the number of operand words and the maximum applicable row size determines which iteration strategy is applied.

In Fig. 3 we show the four prime multiplication processes in a rhombus, with the row sizes highlighted in grey. For the execution of SIKEp434 multiplication routine Fig. 3a we combine the new hybrid iteration design and R-OC. By de-



(a) P434. (b) P503.

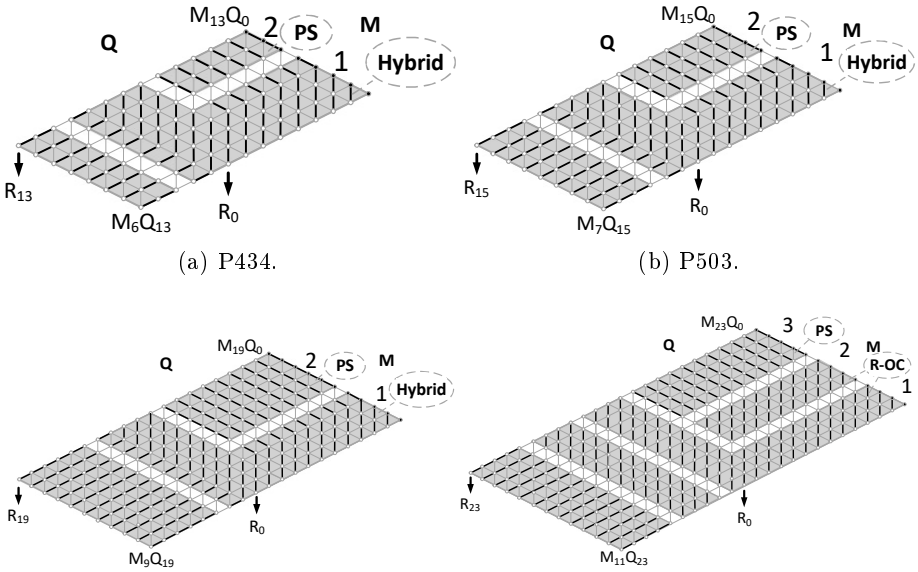
(c) P610. (d) P751.

Fig. 4: Proposed architecture for the implementation of multi-precision squaring for all SIKE primes.

ploying the hybrid iteration we increase the row sizes and thus decrease the number of rows by one compared to the counterparts [3]. SIKEp503 multiplication design featured optimized row sizes already, proposed in [3], however, basing our work on the hybrid iteration step Fig. 3b, we avoid multiple word reloads, thus, obtain better results. We also implement the PS strategy for the first row of the implementation, which additionally allows to save operand memory accesses. SIKEp610 is the only prime with number of words divisible by the size of the hybrid row design, thus, it features hybrid rows only Fig. 3c. Similarly, SIKEp751 Fig. 3d, uses a hybrid design for all rows except the last one. This paper provides optimum multi-precision multiplication techniques for all four SIKE prime numbers, based on the new hybrid inner iteration pattern. Table 1 presents the reduce nuber of instructions and clock cycles for all primes multi-precision multiplication routines compared to our counterparts.

Table 1: Instruction count and clock cycles for memory access and register move operations.

Design	Memory accesses											
	SIKEp434			SIKEp503			SIKEp610			SIKEp751		
	Memory	VMOV	CC	Memory	VMOV	CC	Memory	VMOV	CC	Memory	VMOV	CC
Seo et al. [31]	170	-	340	220	-	440	335	-	670	474	-	948
Anastasova et al. [3]	70	100	240	100	140	310	139	190	468	230	240	700
This work	58	88	**204**	68+2×(16+1)	112	**214**	118	168	**404**	148	212	**508**



(a) P434. (b) P503.

(c) P610. (d) P751.

Fig. 5: Proposed architecture for the implementation of multi-precision reduction for all SIKE primes.

4.3 Multi-Precision Squaring

The multi-precision squaring procedure was earlier improved in [3] by proposing new row architecture. Using their approach, we reduce the number of rows while increasing their size using a PS only inner loop iteration pattern.

The use of PS allows to reserve one and six registers for the operands and reach the maximum row size on the target platform. Although we do not benefit from the R-OC operand caching when the increasing index is changed (i.e., in the mid-point of each row), this adds simply five additional operand accesses per row, which when kept in the FP set, is insignificant. However, it reduces the partial results kept into the stack.

Fig. 4 shows all SIKE primes multi-precision squaring procedures. Our design uses the double operand technique, where we double the operand when the indexes are different (e.g., $A_i \cdot A_j$ with $i \neq j$) and use the original value when the indexes are the same (e.g., $A_i \cdot A_j$ with $i = j$).

The design for SIKEp434, SIKEp503, SIKEp610, and SIKEp751 are shown in Fig. 4a, Fig. 4b, Fig. 4c, and Fig. 4d, respectively. The use of PS iteration minimizes memory access for all primes since it reduces the size of the first row and hence the number of partial values stored in memory for subsequent accumulation. Additionally, the new squaring design reduces the number of rows for p610 and p751 by one in comparison to the results in [3]. By using the PS approach for squaring, we reduce partial results and consequently memory accesses.

Table 2: SIKE finite field arithmetic latency targeting STM32F407VG

Implementation	Latency [CC]							
	$\mathbb{F}_p$mul	$\mathbb{F}_p$sqr	$\mathbb{F}_p$mul	$\mathbb{F}_p$sqr	$\mathbb{F}_p$mul	$\mathbb{F}_p$sqr	$\mathbb{F}_p$mul	$\mathbb{F}_p$sqr
	SIKEp434		SIKEp503		SIKEp610		SIKEp751	
SIDH v3.3 [32]	17,964	17,964	23,364	23,364	35,047	35,047	49,722	49,722
Seo et al. [31]	1,110	981	1,333	1,139	-	-	2,744	2,242
Seo et al. [31]	1,011	889	1,221	1,024	1,869	1,535	2,577	2,066
Anastasova et al. [3]	769	594	952	734	1,506	1,171	2,103	1,543
This work	**702**	**563**	**895**	**684**	**1,435**	**997**	**2,062**	**1,444**

4.4 Multi-precision Reduction

A new row pattern (i.e., outer loop execution flow) has been proposed in [3], ensuring minimal number of rows. In this study, we use our hybrid inner loop architecture to maximize row sizes and therefore delay.

In Fig. 5 we present the optimized multi-precision reduction routine. We have deployed the hybrid inner loop pattern to the SIKEp434 Fig. 5a, SIKEp503 Fig. 5b, and SIKEp610 Fig. 5c architecture. Their designs include one hybrid and one PS row, with the latter allowing for p610 row expansion to six. Thus, we reduce the number of rows by one compared to the counterparts. Due to the 2-piece row shape, the hybrid inner loop does not require extra cycles for reloading the operand, so the R-OC and PS have the same cost. As for the implementation of p751 Fig. 5d, we employ two R-OC rows and one PS, the former ensuring lowest cost row 1, owing to reuse of cached operand values, and the latter ensuring row size of five with extra free register saved for address destination. Based on the modulus length, multiple inner loop iteration patterns were used to enhance the multi-precision reduction.

4.5 Performance Evaluation

The latency of the finite field arithmetic operations developed in this work are reported in CCs in Table 2. Since our approach is constant-time, we are protected from side channel attacks such as Simple and Differential Power Analysis (SPA & DPA). Our results are based on the STM32F407 Discovery Board @24MHz for exact latency.

We report 702, 895, 1435, and 2062 CC for the execution of modular multiplication for p434, p503, p610, and p751, respectively. We achieve optimized latency by 8.7%, 6.0%, 4.5%, and 2.0% for the aforementioned prime sizes, compared to the best-known findings in [3] where the improvement is significant based on the previous effort of assembly implementation of low-level arithmetic. We obtain modular squaring latency of between 5% and 14% better timing for all the primes, with particularly outstanding results for p610 due to the reduced row count.

Table 3: Report of SIKE timing results in terms of clock cycles and speedup percentage on STM32F407 running @24MHz

Implementation	Timing [cc×10^6]							
	KeyGen	**Encaps**	**Decaps**	**Total**	**KeyGen**	**Encaps**	**Decaps**	**Total**
	SIKEp434				SIKEp503			
SIDH v3.3 [32]	650	1,065	1,136	2,202	985	1,623	1,726	3,350
Seo et al. [31]	74	122	130	252	104	172	183	355
Seo et al. [31]	54	87	94	181	74	121	129	250
Anastasova et al. [3]	41	67	72	139	58	96	102	197
This work	**39.0**	**63.6**	**68.0**	**131.6**	**55.9**	**91.8**	**97.7**	**189.5**
	SIKEp610				SIKEp751			
SIDH v3.3 [32]	1,819	3,348	3,368	6,716	3,296	5,347	5,742	11,089
Seo et al. [31]	-	-	-	-	282	455	491	946
Seo et al. [31]	131	241	243	484	225	365	392	757
Anastasova et al. [3]	106	195	196	391	182	295	317	613
This work	**102.5**	**188.1**	**189.3**	**377.4**	**179.4**	**290.5**	**312.1**	**602.7**

5 SIKE Performance Speedup

We present broad findings for every Cortex-M4-based platform, which may vary depending on the target microcontroller's specifications.

We report the results in MCCs in Table 3. We note that we achieve Key Generation, Encapsulation and Decapsulation in 39.0, 63.6, and 68.0 MCCs for SIKEp434, respectively. The total execution timing of SIKEp434 becomes 131.6 MCC which is more than 5% than the previous best reported results. We report 189.5, 377.4, and 602.7 MCC for SIKE security Levels II, III and V, achieving a speedup of up to 3.9%.

In Table 3, we describe the timing of the quantum-safe isogeny based primitive. We highlight our final results in green color. This result is reached after two complete NIST standardized optimization rounds and three handmade assembly code SIKE protocol optimizations on Cortex-M4. Thus, our results show a considerable reduction in protocol execution speed and a unique multi-precision strategy that may be applied to other cryptographic protocols.

6 Conclusions

In this work, we presented the first hybrid architecture for the inner loop of the long-integer multi-precision multiplication, squaring, and reduction routines. We provide a handcrafted assembly implementation of the finite field routines for all SIKE security levels, reaching a new speed record, while keeping our design

constant-time, thus, the architecture remains robust against side-channel attacks such as SPA and DPA. We report 5.63%, 3.93%, 3.48%, and 1.61% improved execution time for SIKE p434, p503, p610, and p751, respectively.

We aim at continuing the progress of SIKE finite field arithmetic on low-end devices and push it further in the NIST standardization process.

Acknowledgments

This works has been supported in parts by an award from NSF CNS-2101085.

References

1. The National Institute of Standards and Technology (NIST)., "Post-quantum cryptography standardization, 2017-2018." https://csrc.nist.gov/Projects/post-quantum-cryptography.
2. D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, V. Soukharev, and D. Urbanik, "Supersingular Isogeny Key Encapsulation," Submission to the NIST Post-Quantum Standardization Project, 2017, https://sike.org/.
3. M. Anastasova, R. Azarderakhsh, and M. M. Kermani, "Fast strategies for the implementation of SIKE round 3 on ARM Cortex-M4," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4129–4141, 2021.
4. C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny Diffie-Hellman," in *Annual International Cryptology Conference*. Springer, 2016, pp. 572–601.
5. C. Costello and H. Hisil, "A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies," in *Advances in Cryptology – ASIACRYPT 2017 - 23rd International Conference on the Theory and Application of Cryptology and Information Security*, 2017, pp. 303–329.
6. C. Costello, P. Longa, M. Naehrig, J. Renes, and F. Virdia, "Improved Classical Cryptanalysis of the Computational Supersingular Isogeny Problem," Cryptology ePrint Archive, Report 2019/298, https://eprint.iacr.org/2019/298.
7. J. Tian, P. Wang, Z. Liu, J. Lin, Z. Wang, and J. Groszschaedl, "Efficient software implementation of the SIKE protocol using new data representation," *IEEE Transactions on Computers*, 2021.
8. H. Cheng, G. Fotiadis, J. Großschädl, and P. Y. Ryan, "Highly vectorized SIKE for AVX-512," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 41–68, 2022.
9. B. Koziel, A. Jalali, R. Azarderakhsh, D. Jao, and M. Mozaffari-Kermani, "NEON-SIDH: efficient implementation of supersingular isogeny Diffie-Hellman key exchange protocol on ARM," in *International Conference on Cryptology and Network Security*. Springer, 2016, pp. 88–103.
10. A. Jalali, R. Azarderakhsh, and M. M. Kermani, "NEON SIKE: Supersingular isogeny key encapsulation on ARMv7," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2018, pp. 37–51.
11. H. Seo, Z. Liu, P. Longa, and Z. Hu, "SIDH on ARM: faster modular multiplications for faster post-quantum supersingular isogeny key exchange," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–20, 2018.
12. A. Jalali, R. Azarderakhsh, M. M. Kermani, M. Campagna, and D. Jao, "ARMv8 SIKE: Optimized supersingular isogeny key encapsulation on ARMv8 processors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4209–4218, 2019.

13. H. Seo, P. Sanal, A. Jalali, and R. Azarderakhsh, "Optimized implementation of SIKE round 2 on 64-bit ARM Cortex-A processors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 8, pp. 2659–2671, 2020.
14. J. Tian, B. Wu, and Z. Wang, "High-Speed FPGA Implementation of SIKE Based on an Ultra-Low-Latency Modular Multiplier," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 9, pp. 3719–3731, 2021.
15. R. Elkhatib, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-Performance FPGA Accelerator for SIKE," *IEEE Transactions on Computers*, 2021.
16. R. Elkhatib, B. Koziel, and R. Azarderakhsh, "Faster Isogenies for Post-quantum Cryptography: SIKE," in *Cryptographers Track at the RSA Conference*. Springer, 2022, pp. 49–72.
17. Z. Ni, M. O'Neill, W. Liu *et al.*, "A High-Performance SIKE Hardware Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.
18. H. Seo, M. Anastasova, A. Jalali, and R. Azarderakhsh, "Supersingular isogeny key encapsulation (SIKE) round 2 on ARM Cortex-M4," *IEEE Transactions on Computers*, vol. 70, no. 10, pp. 1705–1718, 2020.
19. M. Anastasova, M. Bisheh-Niasar, R. Azarderakhsh, and M. M. Kermani, "Compressed SIKE Round 3 on ARM Cortex-M4," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2021, pp. 441–457.
20. M. Hutter and E. Wenger, "Fast multi-precision multiplication for public-key cryptography on embedded microprocessors," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 459–474.
21. H. Seo, Z. Liu, J. Choi, and H. Kim, "Multi-precision squaring for public-key cryptography on embedded microprocessors," in *International Conference on Cryptology in India*. Springer, 2013, pp. 227–243.
22. H. Seo and H. Kim, "Consecutive operand-caching method for multiprecision multiplication," *Journal of information and communication convergence engineering*, vol. 13, no. 1, pp. 27–35, 2015.
23. M. Hutter and P. Schwabe, "Multiprecision multiplication on AVR revisited," *Journal of Cryptographic Engineering*, vol. 5, no. 3, pp. 201–214, 2015.
24. H. Fujii and D. F. Aranha, "Curve25519 for the Cortex-M4 and beyond," in *International Conference on Cryptology and Information Security in Latin America*. Springer, 2017, pp. 109–127.
25. E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH," *Cryptology ePrint Archive*, 2019.
26. M. Campagna and E. Crockett, "Hybrid post-quantum key encapsulation methods (PQ KEM) for transport layer security 1.2 (TLS)," *Internet Engineering Task Force, Internet-Draft draft-campagna-tls-bike-sike-hybrid*, vol. 1, 2019.
27. C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in tls," in *International Conference on Post-Quantum Cryptography*. Springer, 2020, pp. 72–91.
28. M. Anastasova, P. Kampanakis, and J. Massimo, "PQ-HPKE: Post-Quantum Hybrid Public Key Encryption," *Cryptology ePrint Archive*, 2022.
29. ARM, "Cortex-M4 ISA," https://developer.arm.com/documentation/100166/0001.
30. M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4," 2019.
31. H. Seo, A. Jalali, and R. Azarderakhsh, "SIKE round 2 speed record on ARM Cortex-M4," in *International Conference on Cryptology and Network Security*. Springer, 2019, pp. 39–60.
32. Microsoft Team, "Sidh library," https://github.com/Microsoft/PQCrypto-SIDH.