Hardware Implementation of Signature Verification in the New Post-Quantum Signature Scheme MAYO

Kowsyap Pranay Kumar Musumuri *(Author)*
Master's in Computer Engineering
George Mason University
kmusmur@gmu.edu

***Abstract*—The MAYO signature scheme is a promising PQC method based on multivariate quadratic equations, offering smaller public keys compared to the Oil-and-Vinegar (OV) scheme. This paper focuses on the hardware implementation of the MAYO Verify algorithm, addressing challenges in polynomial arithmetic and GF arithmetic operations. The implementation results demonstrate efficient verification with reduced latency and enhanced scalability for various parameter sets.**

I. INTRODUCTION

The rise of quantum computing represents a paradigm shift in the field of computing and cryptography. Unlike classical computers, which rely on bits to process information as binary states (0 or 1), quantum computers utilize quantum bits or qubits. Qubits exploit the principles of superposition and entanglement, allowing quantum computers to process a vast number of computations simultaneously. While this capability holds the promise of revolutionizing fields like optimization, materials science, and artificial intelligence, it also poses a significant threat to traditional cryptographic systems.

Modern cryptography relies heavily on hard mathematical problems such as:

1. **Integer factorization**: The security of RSA encryption, a widely used cryptographic system, is based on the difficulty of factoring large composite numbers into their prime constituents. Quantum computers, equipped with Shor's algorithm, can perform this task exponentially faster than classical computers.
2. **Discrete logarithms**: This problem underpins the security of systems such as Diffie-Hellman key exchange and Elliptic Curve Cryptography (ECC). Again, Shor's algorithm makes discrete logarithms solvable in polynomial time on a quantum computer.

The problem? Both integer factorization and discrete logarithms are rendered insecure in the presence of a sufficiently powerful quantum computer. Consequently, all encrypted data, secure communications, and digital signatures based on these primitives become vulnerable to decryption or impersonation attacks.

The timeline for the realization of large-scale quantum computers capable of breaking current cryptographic systems is uncertain, but proactive measures are necessary to ensure the long-term security of sensitive information. Post-Quantum Cryptography (PQC) is the field dedicated to developing cryptographic systems that remain secure against adversaries equipped with quantum computers.

Multivariate Quadratic (MQ) cryptography is one of the promising approaches within PQC. MQ cryptography relies on the computational difficulty of solving systems of multivariate quadratic equations over finite fields. Unlike traditional problems like integer factorization, the MQ problem is resistant to quantum attacks, as there are no known efficient quantum algorithms to solve these systems.

Several post-quantum signature schemes have been proposed and evaluated under the NIST Post-Quantum Cryptography Standardization process. These include:

- **Lattice-based Schemes**: E.g., Falcon, Dilithium.
- **Code-based Schemes**: E.g., Classic McEliece.
- **Hash-based Schemes**: E.g., SPHINCS+.
- **Multivariate-based Schemes**: E.g., Rainbow, GeMSS, and MAYO.

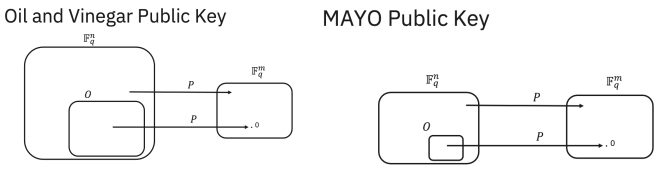
Each category has its trade-offs in terms of security, efficiency, and practicality. Among these, multivariate-based schemes stand out due to their strong theoretical foundation and the MQ problem's resistance to quantum attacks. This project focuses on the hardware implementation of a new post-quantum signature scheme called **MAYO** (Multivariate Algebraic-based Yields Optimized), a **variant** of the **Oil-and-Vinegar (O&V)** scheme[1]. The hardware implementation of signature verification operations within the MAYO scheme is critical for ensuring secure and efficient communication in quantum-resistant environments. I will be exploring how this verification can be optimized for time and implemented using VHDL in Vivado, aiming for minimal execution time.

II. MAYO OVERVIEW

The MAYO signature scheme is an optimized variant of the Oil-and-Vinegar (OV) cryptographic scheme, a well-established approach within multivariate cryptography. The OV scheme relies on the difficulty of solving multivariate quadratic equations over finite fields, a problem known to be computationally infeasible even for quantum computers. The variables in the OV scheme are divided into two categories: oil variables and vinegar variables. Oil variables have limited interaction with the system, making them essential for the structured solving process facilitated by the private key. Vinegar variables, on the other hand, can freely interact among

themselves and with oil variables, contributing randomness and complexity to the equations.

In the OV scheme, the key generation process involves creating a public key consisting of multivariate quadratic polynomials and a private key that acts as a trapdoor for efficient solving. Signing entails solving the equations using the private key to produce a valid signature, while verification evaluates the public polynomials at the signature and checks if the result matches a given target. Despite its strong security foundation, the OV scheme suffers from a significant drawback: its public keys are exceptionally large, often exceeding 44 KB. This poses a serious limitation for applications with constrained memory or bandwidth, such as IoT devices or embedded systems.



To address these challenges, the MAYO scheme introduces several key innovations that drastically reduce public key size while retaining the advantages of the OV scheme. One of the most impactful changes is the reduction in the number of oil variables (o), which significantly decreases the dimensions of the public key. By optimizing o, MAYO maintains the security of the underlying system while achieving compactness. Another major innovation is the k-fold whipping of polynomials, a technique that expands the oil and vinegar equations into a larger space using a set of public matrices. These matrices introduce additional randomness and ensure that the equations remain solvable with the private key. This approach ensures that the system remains secure while enabling smaller parameters and more compact representations of the public key.

Another significant feature of MAYO is its efficient representation of public keys. Instead of storing the entire public key, MAYO uses a seed to generate most of the coefficients, drastically reducing storage requirements. At NIST Level 1 security, MAYO achieves a public key size of only 1.1 KB, compared to 44 KB for the OV scheme. The signature size is slightly larger, at 321 bytes, but this trade-off is acceptable given the dramatic reduction in key size.

MAYO is parameterized to balance security and efficiency across different applications. Its parameters include n (the total number of variables), m (the number of equations), o (the number of oil variables), k (the expansion factor for whipping), and q (the size of the finite field). For NIST Level 1 security, the parameters (n,m,o,k,q)=(66,64,8,9,16) yield a public key size of 1.1 KB and a signature size of 321 bytes. These

	Oil & Vinegar GF(16)	Oil & Vinegar GF(256)	MAYO 1 $o = 8$	MAYO2 $o = 18$
# Variables	160	112	66 x 9	66 x 16
# Equations	64	44	64	69
Finite Field	GF(16)	GF(256)	GF(16)	GF(16)
Pk size	67 KB	44 KB	1.1 KB	5.4 KB
Signature size	96 B	128 B	321 B	180 B

parameters ensure a high level of security while maintaining practical implementation requirements.

MAYO's compactness and efficiency make it particularly well-suited for resource-constrained environments such as IoT devices and embedded systems. In IoT applications, memory and bandwidth limitations often restrict the use of cryptographic systems with large keys. By reducing public key size, MAYO enables secure communication in these scenarios without overburdening system resources. Similarly, embedded systems benefit from MAYO's efficient verification process and minimal storage requirements, making it an ideal choice for secure firmware updates and device authentication. Beyond constrained environments, MAYO is also valuable in post-quantum networks, where fast and reliable verification processes are essential for maintaining secure communications.

The transition from OV to MAYO highlights an essential trade-off in cryptographic design: reducing public key size often comes at the cost of increased signature size or computational overhead. In MAYO, this trade-off is carefully balanced to ensure practicality without compromising security. For example, while the signature size in MAYO is three times larger than that of the compressed OV scheme, the reduction in public key size by a factor of 58 makes it far more practical for real-world deployment.

MAYO represents a significant advancement in post-quantum cryptography. By addressing the limitations of traditional multivariate schemes, it bridges the gap between theoretical constructs and practical implementations. Its ability to provide secure, compact, and efficient signatures positions it as a strong candidate for standardization in the post-quantum

Parameter set security level	$MAYO_1$ 1	$MAYO_2$ 1	$MAYO_3$ 3	$MAYO_5$ 5
n	66	78	99	133
m	64	64	96	128
o	8	18	10	12
k	9	4	11	12
q	16	16	16	16
salt_bytes	24	24	32	40
digest_bytes	32	32	48	64
pk_seed_bytes	16	16	16	16
$f(z)$	$f_{64}(z)$	$f_{64}(z)$	$f_{96}(z)$	$f_{128}(z)$
secret key size	24 B	24 B	32 B	40 B
public key size	1168 B	5488 B	2656 B	5008 B
signature size	321 B	180 B	577 B	838 B
expanded sk size	69 KB	92 KB	230 KB	553 KB
expanded pk size	70 KB	97 KB	233 KB	557 KB

era. As quantum computing continues to advance, schemes like MAYO will play a critical role in ensuring the security of digital communications in a quantum-resistant future.

III. MAYO VERIFICATION

The MAYO Verify algorithm is a vital component of the signature verification process within the MAYO signature scheme. Its primary purpose is to ensure the authenticity of a

given signature by evaluating the corresponding polynomial equations and performing arithmetic operations over finite fields. This process ensures that the signature satisfies the mathematical requirements imposed by the public key and matches the hash of the original message. By integrating efficient operations and rigorous mathematical principles, the

Algorithm 9 MAYO.Verify(epk, M, sig)

Input: Expanded public key epk $\in \mathcal{B}^{\text{epk\_bytes}}$
Input: Message M $\in \mathcal{B}^*$
Input: Signature sig $\in \mathcal{B}^{\text{sig\_bytes}}$
Constant: $\mathbf{E} \in \mathbb{F}_q^{m \times m}$ # Represents multiplication by z in $\mathbb{F}_q[z]/(f(z))$
Output: An integer result to indicate if sig is valid (result $= 0$) or invalid (result < 0).

1: //Decode epk.
2: P1_bytestring $\leftarrow$ epk[0 : P1_bytes]
3: P2_bytestring $\leftarrow$ epk[P1_bytes : P1_bytes + P2_bytes]
4: P3_bytestring $\leftarrow$ epk[P1_bytes + P2_bytes : P1_bytes + P2_bytes + P3_bytes]
5: $\{\mathbf{P}_i^{(1)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(1)}}(\text{P1\_bytestring})$ // $\mathbf{P}_i^{(1)} \in \mathbb{F}_q^{(n-o)\times(n-o)}$ upper triangular
6: $\{\mathbf{P}_i^{(2)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(2)}}(\text{P2\_bytestring})$ // $\mathbf{P}_i^{(2)} \in \mathbb{F}_q^{(n-o)\times o}$
7: $\{\mathbf{P}_i^{(3)}\}_{i\in[m]} \leftarrow \text{Decode}_{\mathbf{P}^{(3)}}(\text{P3\_bytestring})$ // $\mathbf{P}_i^{(3)} \in \mathbb{F}_q^{o\times o}$ upper triangular
8:
9: //Decode sig.
10: salt $\leftarrow$ sig$[\lceil nk/2 \rceil : \lceil nk/2 \rceil + \text{salt_bytes}]$
11: $\mathbf{s} \leftarrow \text{Decode}_{vec}(kn, \text{sig})$
12: **for** i from 0 to $k-1$ **do**
13: $\quad \mathbf{s}_i \leftarrow \mathbf{s}[i*n : (i+1)*n]$
14:
15: //Hash message and derive $\mathbf{t}$.
16: M_digest $\leftarrow$ SHAKE256(M, digest_bytes) // M_digest $\in \mathcal{B}^{\text{digest\_bytes}}$
17: $\mathbf{t} \leftarrow \text{Decode}_{vec}(m, \text{SHAKE256}(\text{M\_digest} \,\|\, \text{salt}, \lceil m \log(q)/8 \rceil))$ // $\mathbf{t} \in \mathbb{F}_q^m$
18:
19: //Compute $\mathcal{P}^*(\mathbf{s})$.
20: $\mathbf{y} \leftarrow \mathbf{0}_m$ // $\mathbf{y} \in \mathbb{F}_q^m$
21: $\ell \leftarrow 0$
22: **for** i from 0 to $k-1$ **do**
23: $\quad$ **for** j from $k-1$ to i **do**
24: $\quad\quad \mathbf{u} \leftarrow \begin{cases} \left\{ \mathbf{s}_i^\intercal \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_i \right\}_{a\in[m]} & \text{if } i = j \\ \left\{ \mathbf{s}_i^\intercal \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_j + \mathbf{s}_j^\intercal \begin{pmatrix} \mathbf{P}_a^{(1)} & \mathbf{P}_a^{(2)} \\ \mathbf{0} & \mathbf{P}_a^{(3)} \end{pmatrix} \mathbf{s}_i \right\}_{a\in[m]} & \text{if } i \neq j \end{cases}$ // $\mathbf{u} \in \mathbb{F}_q^m$
25: $\quad\quad \mathbf{y} \leftarrow \mathbf{y} + \mathbf{E}^\ell \mathbf{u}$
26: $\quad\quad \ell \leftarrow \ell + 1$
27:
28: //Accept signature if $\mathbf{y} = \mathbf{t}$.
29: **if** $\mathbf{y} = \mathbf{t}$ **then**
30: $\quad$ **return** 0
31: **return** -1

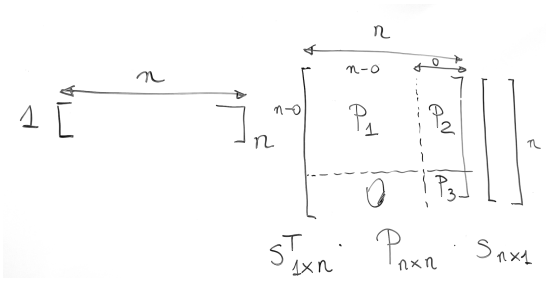
MAYO Verify algorithm plays a crucial role in establishing the validity of digital signatures while maintaining security and computational efficiency.

The algorithm operates on three main inputs: the expanded public key (*epk*), the signature (*sig*), and the hash target (*t*). The public key, initially stored in a compressed form to minimize

$$\begin{aligned} f_{64}(z) &= z^{64} + x^3z^3 + xz^2 + x^3 \\ f_{96}(z) &= z^{96} + xz^3 + xz + x \\ f_{128}(z) &= z^{128} + xz^4 + x^2z^3 + x^3z + x^2 \end{aligned}$$

storage requirements, is expanded into three matrices, *P1, P2*, and *P3*, during verification. These matrices represent the coefficients of the whipped multivariate polynomials and serve as the foundation for the polynomial evaluations. The signature, on the other hand, consists of a tuple (*s1 ,s2 ,...,sk*) of elements in $F_q{}^n$, generated by solving the whipped polynomial equations using the private key during the signing process. It acts as proof that the signer possesses the private key corresponding to the public key. Finally, the hash target t is derived using the SHAKE256 cryptographic hash function, which hashes the original message along with a randomly chosen salt. The hash target serves as the reference value that the polynomial evaluations must match to validate the signature.

During the evaluation, the algorithm substitutes each component of the signature into the expanded polynomials and combines the results using matrix operations. These operations, including addition and multiplication over the finite field, are designed to handle the complexity of multivariate quadratic equations efficiently. The modular arithmetic ensures that all calculations remain within the bounds of Fq , maintaining the integrity of the results. Once the evaluation is complete, the output of the polynomial P∗ is compared against the hash target t. If the evaluated value matches t, the signature is



declared valid, indicating that it was generated using the correct private key. Conversely, a mismatch signifies an invalid signature.

The mathematical framework underlying the verification process is both rigorous and efficient. The algorithm leverages the inherent difficulty of solving random multivariate quadratic equations, ensuring its security even against adversaries with quantum computing capabilities. Key operations include polynomial addition and multiplication, matrix multiplications with the E-matrices, and modular reduction. These operations are computationally intensive but are optimized for performance in hardware implementations.

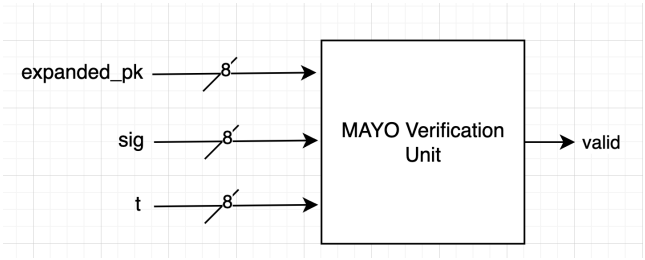
To enhance the efficiency of the MAYO Verify algorithm, it is often implemented in h`

The significance of the MAYO Verify algorithm lies in its ability to establish trust and authenticity in digital communication. Its compact representation of public keys and efficient verification process make it an ideal candidate for constrained environments such as IoT devices and embedded systems. Furthermore, its compatibility with hardware acceleration allows it to scale for applications in post-quantum networks, where secure and efficient verification is essential for maintaining data integrity. The algorithm's reliance on multivariate quadratic equations, a problem known for its computational hardness, ensures that it remains secure even in the face of quantum adversaries. In doing so, the MAYO Verify algorithm bridges the gap between theoretical cryptographic constructs and practical implementations, paving the way for secure communication in a post-quantum world.

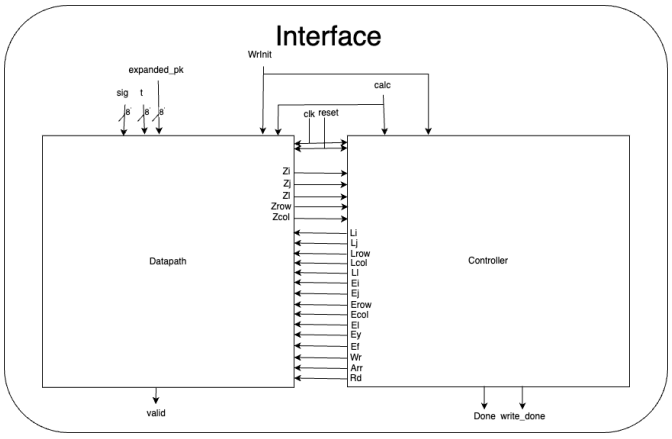
IV. HARDWARE IMPLEMENTATION

The hardware implementation of the MAYO Verify[2][3] algorithm represents a significant step toward realizing practical post-quantum cryptographic systems. While the algorithm is computationally efficient in software, its hardware implementation offers distinct advantages, including reduced latency, improved throughput, and scalability for resource-

constrained environments. Translating the abstract mathematical operations of the MAYO Verify algorithm into hardware-friendly modules requires careful consideration of datapath design, control logic, and memory management. This section delves into the goals, components, and challenges of implementing the MAYO Verify algorithm in hardware, highlighting its practicality and efficiency.



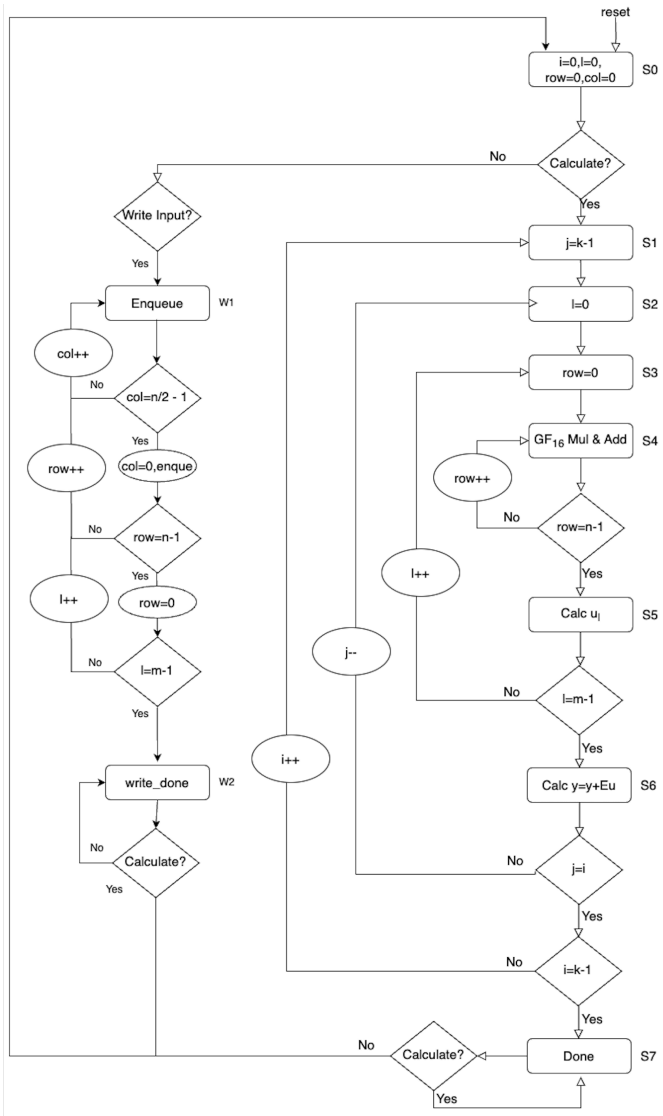
The primary goal of the hardware implementation is to achieve a balance between performance and resource utilization. Post-quantum cryptographic schemes, including MAYO, rely heavily on operations over finite fields, such as modular addition, multiplication, and reduction. Implementing these operations efficiently in hardware is critical to ensuring low latency and high throughput. Another key goal is scalability. The hardware must accommodate different parameter sets of the MAYO scheme, such as variations in the number of variables (n), equations (m), and field size (q), without requiring significant redesigns. Additionally, the implementation should minimize power consumption and area requirements, particularly for deployment in embedded systems and IoT devices.

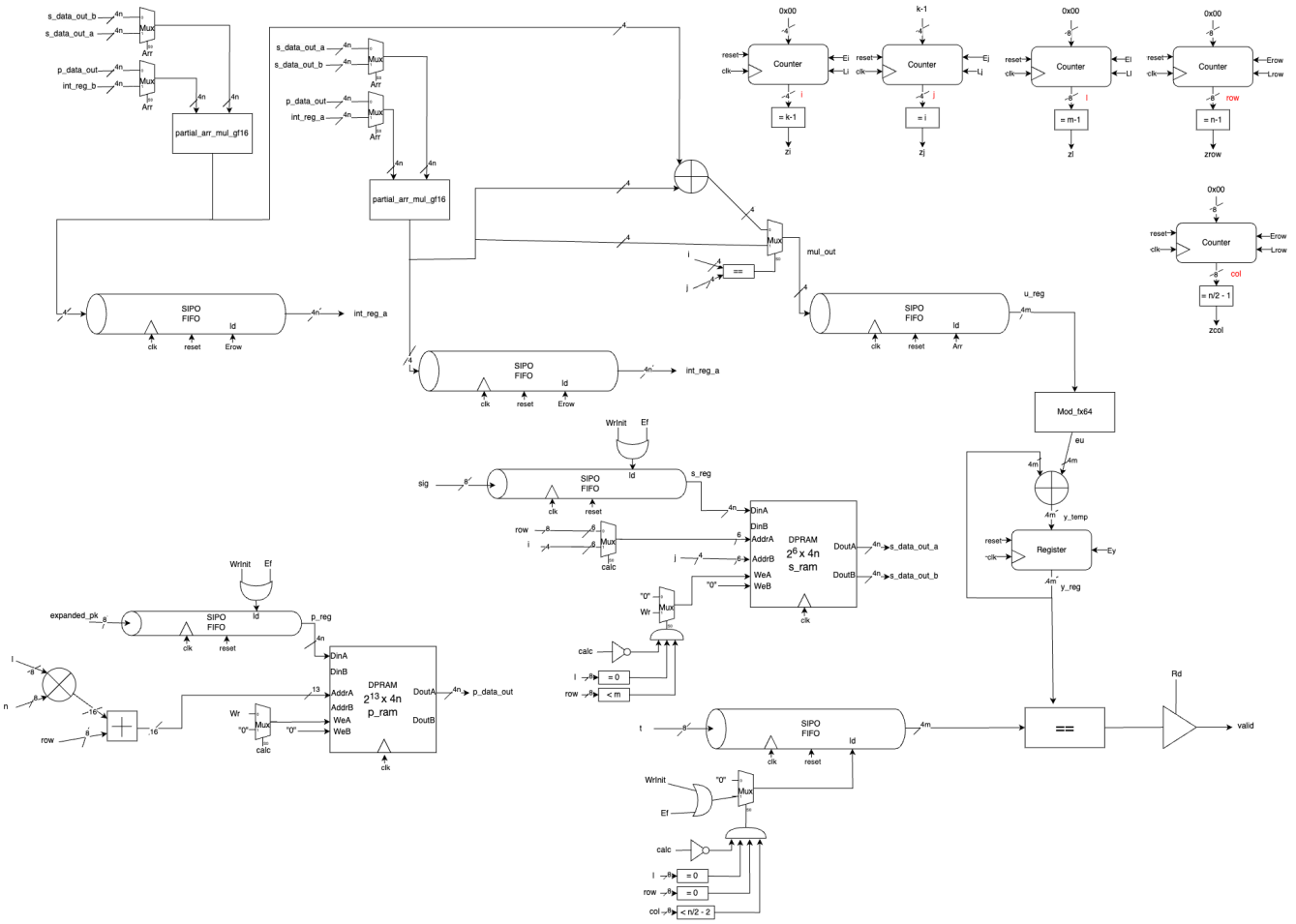


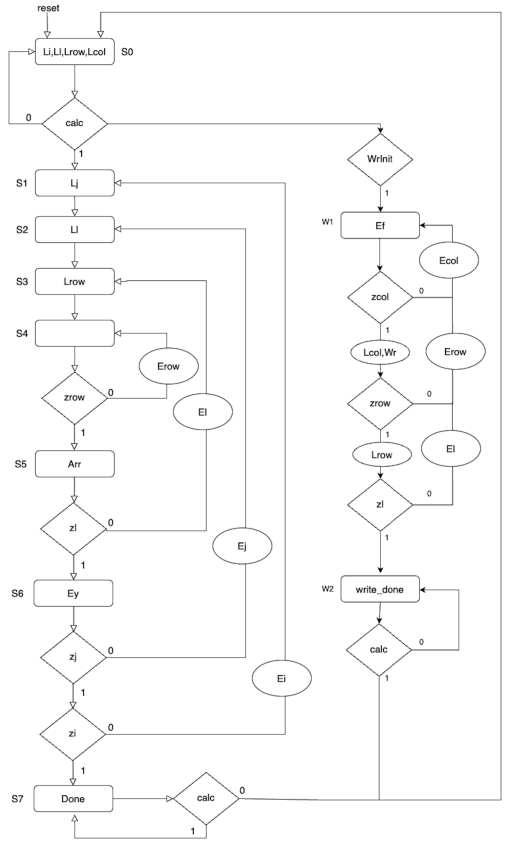
The hardware design is divided into three main components: the datapath, the controller, and the interface. The datapath is the computational core of the system, responsible for performing arithmetic operations and polynomial evaluations. It includes dedicated modules for finite field addition, multiplication, and modular reduction, which are optimized for high-speed operation. These modules are organized into a pipeline architecture, allowing multiple operations to be processed simultaneously. The datapath also handles the evaluation of the whipped polynomials, ensuring that the computations are performed efficiently and accurately.

The controller acts as the brain of the hardware implementation, orchestrating the sequence of operations in the datapath. It ensures that the inputs are processed in the correct order and that intermediate results are stored and retrieved as needed. The controller also manages interactions with external memory, such as loading the expanded public key matrices and the signature components into the datapath. The data path contain two 2^{13} x 4n DPRAM modules and five SIPO FIFO modules to store and use the input and intermediate data. By implementing an efficient control flow, the controller minimizes idle time and ensures seamless execution of the verification process.

The interface connects the hardware implementation to external inputs and outputs. It manages the transfer of the expanded public key (epk), signature (sig), and hash target (t) into the system and provides the verification result (valid) as the output. The interface is designed to handle the compressed format of the public key, decoding it into the expanded matrices P1, P2, and P3 before passing them to the datapath.







This decoding step is essential for leveraging the compact representation of the public key while ensuring accurate polynomial evaluations.

One of the primary challenges in hardware implementation is translating the mathematical complexity of the MAYO Verify algorithm into a hardware-friendly format. The whipped polynomial evaluations involve multiple matrix operations and finite field computations, which can be resource-intensive. To address this, the hardware design incorporates lookup tables and parallel processing units to accelerate computations. Another challenge is memory management. The expanded public key matrices can be large, requiring efficient storage and retrieval mechanisms to avoid bottlenecks. The hardware must also handle varying parameter sets gracefully, ensuring compatibility with different security levels and use cases.

The implementation results demonstrate the effectiveness of the hardware design. Timing simulations show that the hardware achieves significant reductions in latency compared to software implementations. For example, using the Mayo1 parameter set, the hardware achieves a latency of 3.86 milliseconds and a throughput of 259 verifications per second. With the Mayo2 parameter set, the latency improves to 1.01 milliseconds, resulting in a throughput of 990 verifications per second. These results highlight the scalability and efficiency of the hardware implementation, making it suitable for high-performance applications.

The significance of hardware implementation extends beyond performance metrics. By integrating the MAYO Verify algorithm into hardware, the system becomes more robust and resistant to side-channel attacks, which are a common concern in cryptographic systems. Additionally, the compactness and energy efficiency of the hardware design make it ideal for deployment in constrained environments, such as IoT devices and embedded systems. The hardware implementation of MAYO Verify thus bridges the gap between theoretical cryptography and practical applications, paving the way for secure communication in the post-quantum era.

In conclusion, the hardware implementation of the MAYO Verify algorithm is a critical step toward realizing practical post-quantum cryptographic systems. By addressing challenges in arithmetic operations, memory management, and control flow, the hardware design achieves a balance between performance, resource utilization, and scalability. Its ability to handle varying parameter sets and its robustness against side-channel attacks make it a promising solution for secure and efficient verification in the quantum era.

V. Implementation Results

The hardware implementation of the MAYO Verify algorithm demonstrates significant improvements in performance and scalability, making it suitable for practical deployment in post-quantum cryptographic systems. Timing simulations were conducted using various parameter sets, including Mayo1 and Mayo2, which revealed the efficiency of the hardware design. For Mayo1 parameters, the design achieved a latency of **1.96 milliseconds** per verification, corresponding to a throughput of **510 verifications per second**. With the optimized Mayo2 parameters, the latency reduced to **0.51 milliseconds**, enabling a throughput of **1960 verifications per second**. These results validate the scalability and efficiency of the implementation.

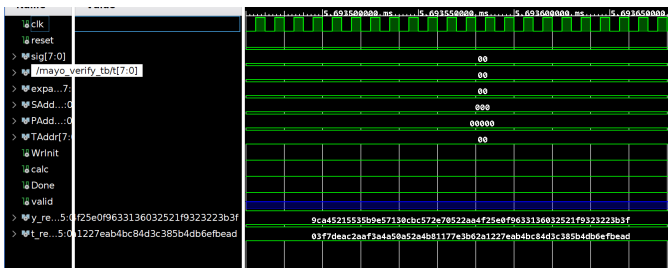
Mayo1

ET(cycles) = k[(k+1)(((n/2)+1)m+1)+1]+1
= 9[10 * 2177 +1]+1
= 195940 cycles
ET(ns) = ET(cycles) * T_{clk} = 195940 cycles * 10 ns
= 1959400 ns = 1.96 ms
Throughput = 1/ET(ns) = 1000/1.96
= 510 (operations/sec)

Mayo2

ET(cycles) = k[(k+1)(((n/2)+1)m+1)+1]+1
= 4[5 * 2561 +1]+1
= 51225 cycles
ET(ns) = ET(cycles) * T_{clk} = 51225 cycles * 10 ns
= 512250 ns = 0.51 ms
Throughput = 1/ET(ns) = 1000/0.51
= 1960 (operations/sec)

Latency and throughput, two critical performance metrics, were analyzed in detail. Latency reflects the time required to complete one verification, while throughput measures the system's ability to handle multiple requests per second. The pipeline architecture and efficient memory management were instrumental in achieving high throughput without sacrificing latency. These features ensure that the design can handle demanding real-time applications, such as IoT authentication and blockchain signature verification.




```
[
  9, 12, 10,  4, 5,  2, 1,  5,  5,  3,  5, 11,
  9, 14,  5,  7, 1,  3, 0, 12, 11, 12,  5,  7,
  2, 14,  7,  0, 5,  2, 2, 10, 10,  4, 15,  2,
  5, 14,  0, 15, 9,  6, 3,  3,  1,  3,  6,  0,
  3,  2,  5,  2, 1, 15, 9,  3,  2,  3,  2,  2,
  3, 11,  3, 15
]
```

The following above snapshots show the comparison of results between my hardware implementation in VHDL and the software implementation in JavaScript of MAYO.Verify algorithm for $MAYO_1$ parameter set.

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	40484	0	0	134600	30.08
LUT as Logic	6164	0	0	134600	4.58
LUT as Memory	34320	0	0	46200	74.29
LUT as Distributed RAM	34320	0			
LUT as Shift Register	0	0			
Slice Registers	1869	0	0	269200	0.69
Register as Flip Flop	1869	0	0	269200	0.69
Register as Latch	0	0	0	269200	0.00
F7 Muxes	17952	0	0	67300	26.67
F8 Muxes	8860	0	0	33650	26.33

The table depicts the resource utilization report after synthesizing and implementing the design on the **Xilinx Artix-7 FPGA** with the part number **xc7a200tffg1156-3**. The table provides details about the usage of Slice Look-Up Tables (LUTs), Slice Registers, and Muxes. A total of **134,600 LUTs** are available on the device, out of which **40,484** are used, resulting in a utilization of **30.08%**. Most of the LUTs are used as Distributed RAM (34,320), with a high memory utilization of **74.29%**, while only **6,164 LUTs** are used for logic, which is a smaller fraction (**4.58%**). No LUTs are used as shift registers. The Slice Registers utilization is minimal, with only **1,869 registers** used as Flip Flops out of the available **269,200**, which is just **0.69%**. The design also uses **F7 Muxes** and **F8 Muxes**, with utilization rates of **26.67%** and **26.33%**, respectively. This data highlights the heavy use of Distributed RAM in the design, showing that memory resources are a key part of the implementation. Meanwhile, the low register usage and moderate utilization of logic and Muxes indicate a design optimized for specific functions. This information is useful for understanding how the FPGA resources are distributed and if the design can fit within the constraints of the selected FPGA device.

Compared to software implementations, the hardware design offers significantly lower latency and higher throughput. While software-based verification may require several milliseconds per operation, the hardware implementation processes thousands of verifications per second, bridging the gap between theoretical post-quantum cryptography and real-world applications. These results underscore the importance of hardware acceleration in achieving secure, efficient, and scalable verification systems for the quantum-resistant future.

The success of this implementation demonstrates the feasibility of integrating MAYO Verify into practical cryptographic frameworks, paving the way for its adoption in next-generation secure communication systems.

VI. Future Work

I could explore **optimization of the hardware architecture** for enhanced performance and reduced resource usage. For instance, further pipeline stages could be introduced to increase throughput, or lookup tables for finite field arithmetic could be optimized for faster access and lower memory requirements. Extending the design to support varying parameter sets dynamically, without reconfiguration, is another area that can significantly improve flexibility and adaptability.

Another potential area of research involves **power efficiency**. As the design targets embedded systems and IoT devices, reducing energy consumption without compromising performance is crucial. Investigating low-power design techniques, such as clock gating or approximate arithmetic, could make the hardware more practical for constrained environments.

Lastly, the security of the hardware implementation against **side-channel attacks** needs thorough analysis. Techniques such as power analysis or fault injection could potentially exploit weaknesses in the implementation. Countermeasures, such as masking or redundant computations, can be integrated into the design to enhance its robustness against such attacks.

VII. Conclusion

This project demonstrates the feasibility and practicality of implementing the MAYO Verify algorithm in hardware, an essential step toward real-world deployment of post-quantum cryptographic systems. By leveraging efficient datapath and control logic, the design achieved low latency and high throughput, making it suitable for applications in IoT, embedded systems, and blockchain technologies. Timing simulations validated the design, with $Mayo_2$ parameters achieving a latency of **0.51 milliseconds** and a throughput of **1960 verifications per second**.

The compactness and scalability of the hardware design make it a promising solution for resource-constrained environments. Furthermore, the ability to process thousands of verifications per second ensures its relevance for large-scale applications, such as secure communications in post-quantum networks. While the project achieved significant milestones, areas such as FPGA synthesis, power optimization, and security against side-channel attacks remain as future challenges.

The importance of hardware acceleration in post-quantum cryptography cannot be overstated. As quantum computing continues to evolve, the development of efficient, secure, and scalable cryptographic solutions like MAYO Verify will play a pivotal role in protecting digital communications. This work provides a foundation for advancing these efforts and bridging the gap between theoretical cryptography and practical implementation.

References

1. W. Beullens, MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps, PQC Seminar, September 24, 2024, https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/pqc-seminars/presentations/20-mayo-09242024.pdf
2. O. Sayari, S. Marzougui, T. Aulbach, J. Kramer, and J. P . Seifert, “HaMAYO: A Fault-Tolerant Reconfigurable Hardware Implementation of the MAYO Signature Scheme, ” in Constructive Side-Channel Analysis and Secure Design, COSADE 2024, pp. 240–259, Springer 2024, https://link.springer.com/chapter/10.1007/978-3-031-57543-313
3. F. Hirner, M. Streibl, F. Krieger, A. Can Mert, S.S. Roy, “Whipping the Multivariate-based MAYO Signature Scheme using Hardware Platforms, ”ACM CCS 2024 and Cryptology ePrint Archive, paper 2023-1267, https://eprint.iacr.org/2023/1267