Drop-In Masked Modular Reduction for ML-DSA: Cutting Side-Channel Cost in the Root-of-Trust

Merve Karabulut
Florida Atlantic University
Boca Raton, FL, USA
mkarabulut@fau.edu

Reza Azarderakhsh
Florida Atlantic University
PQSecure Technologies
Boca Raton, FL, USA
razarderakhsh@fau.edu

***Abstract*—Masking is an effective defense against side-channel attacks, yet it remains costly under hardware constraints. The Caliptra Root-of-Trust is a representative case, where its masked ML-DSA implementation incurs about $6\times$ area overhead. We propose a novel first-order masking solution that optimizes Caliptra, achieving significant improvements in area–delay efficiency. Compared to Caliptra's ML-DSA reduction, our design achieves a $12.1\times$ speedup, reducing LUTs by 86.7% and FFs by 94.5%, while improving area–delay efficiency by $91\times$. TVLA, with over $1{,}000{,}000$ traces, shows no first-order leakage, satisfies Caliptra's security requirements, and significantly improves implementation efficiency.**

***Index Terms*—ML-DSA, Modular Reduction, Masking, Side-Channel Resistance, Caliptra**

I. Introduction

The advent of scalable quantum computers poses a significant threat to modern asymmetric-key cryptography such as elliptic-curve cryptography and RSA, which rely on the hardness of the discrete logarithm and integer factorization problems [1], [2]. Shor's algorithm [3] demonstrates that these problems can be solved in polynomial time with quantum computing, thereby rendering these classical cryptography schemes insecure. To address this threat, the National Institute of Standards and Technology (NIST) initiated the Post-Quantum Cryptography (PQC) standardization process [4]. NIST selected two key encapsulation mechanisms as PQC standards: the Module-Lattice-based Key Encapsulation Mechanism (ML-KEM) [5] and the Hamming Quasi-Cyclic (HQC) mechanism [6]. It also standardized three digital signature algorithms: Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [7], Falcon [8], and SPHINCS+ [9].

PQC algorithms offer strong mathematical resilience against quantum attacks. Still, their physical implementations remain vulnerable to side-channel attacks [10]–[26]. These attacks exploit power, electromagnetic, and timing leakages to recover secret information. Although several masking and hiding techniques exist [27]–[29], they typically incur significant area and performance overheads, which is especially problematic for hardware systems operating under strict resource constraints. Motivated by the tension between side-channel security and implementation cost, we investigate the side-channel protected ML-DSA engine in Caliptra [30], an industry-grade Root-of-Trust (RoT) subsystem.

Caliptra [30] is an open-source hardware RoT platform collaboratively developed by major technology companies. It is designed for integration into diverse System-on-Chip (SoC) platforms and serves as a trusted security anchor. To mitigate both quantum computing threats and side-channel attacks, Caliptra introduces the Adam's Bridge IP [31]. This is a hardware implementation of ML-DSA and ML-KEM[1] with side-channel countermeasures.

Caliptra operates under strict area, power, and performance constraints due to its integration across diverse SoCs. Achieving strong side-channel resistance within these limits is a significant design challenge. Prior work has shown that the unprotected version of Caliptra's ML-DSA engine is vulnerable to Correlation Power Analysis (CPA) [17], motivating the integration of masking and hiding countermeasures in the current design. While these protections are effective, they introduce substantial implementation overhead, increasing area by up to $6\times$ for the Adam's Bridge IP [31], [32]. This overhead is especially significant in the modular reduction unit of the ML-DSA datapath, which is a key component in both the inverse Number Theoretic Transform (INTT) and pointwise multiplication operations. To address this, we propose a first-order masked modular reduction architecture that preserves leakage resistance while significantly improving efficiency. The proposed optimization accelerates both INTT and pointwise multiplication stages, reduces latency and hardware cost, and increases signing throughput without compromising first-order masking protection. The main contributions of this study are as follows:

- We performed a comprehensive analysis of Caliptra and identified the source of the substantial area and performance overhead in the Adam's Bridge masking block. We then propose a novel reduction decision unit that enables a more efficient modular reduction circuit, serving as a drop-in replacement.
- The proposed first-order masked reduction design features a fully pipelined architecture, achieves a $12.1\times$ speedup, and reduces hardware cost by 86.7% in LUTs and 94.5% in FFs. This results in $91\times$ improvement in area–delay efficiency. When integrated into Adam's

[1]ML-KEM development was ongoing when at the time work was completed.

Bridge, the new masking block further improves signature generation throughput by 1.32× and reduces overall hardware cost by 22.3% in LUTs and 38.1% in FFs.
- We empirically validate the proposed masked design with Test Vector Leakage Assessment (TVLA) over 1,000,000 traces and observe no first-order leakage, indicating that our solution satisfies Caliptra's security requirements while significantly improving implementation efficiency.

II. Background

This section presents the background information relevant to our work. We first introduce masking as a countermeasure against side-channel attacks and describe the leakage models considered in this work. We then briefly review the ML-DSA algorithm. Finally, we give a short overview of the Caliptra RoT subsystem.

A. Masking and Leakage Model

Masking is an effective countermeasure against side-channel attacks, grounded in the principles of secure multi-party computation and secret sharing [33]. The d-th-order masking scheme randomizes sensitive variables by dividing them into $d+1$ shares so that no subset of up to d shares reveals exploitable information. In Boolean masking, the shares are defined over $\mathbb{F}_2$, and a sensitive variable x is represented as

$$x = x_0 \oplus x_1 \oplus \cdots \oplus x_d,$$

where $\oplus$ denotes the bitwise XOR operation. This representation is naturally suited for masking bitwise logical operations. In the arithmetic domain, masking is defined over a modulus $p \in \mathbb{Z}, p > 2$, and the same x variable is expressed as

$$x = (x_0 + x_1 + \cdots + x_d) \bmod p,$$

where p denotes the masking modulus.

Boolean and arithmetic masking domains can be converted into each other using dedicated gadgets. Arithmetic-to-Boolean (A2B) converters take arithmetic shares and produce Boolean shares of the same sensitive variable. Boolean-to-Arithmetic (B2A) converters perform the inverse transformation and recover arithmetic shares from Boolean ones. Several studies have proposed secure methods for such domain conversions [34], [35], which form the foundation of modern hardware masking practice. The adopted masking strategy adheres to standard hardware masking principles.

TVLA [33] is an effective and commonly used technique for leakage detection. It identifies statistically significant differences between the means of two groups of side-channel measurements corresponding to different inputs, using the Welch t-score defined as

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}},$$

where μ_i, σ_i, and N_i denote the mean, standard deviation, and number of traces in group $i \in \{1, 2\}$. In practice, a commonly adopted first-order TVLA criterion is that $|t| \geq 4.5$ at any sample point is interpreted as evidence of first-order leakage, whereas $|t| < 4.5$ for all samples indicates no detectable first-order leakage.

Algorithm 1 ML-DSA Key and Signature Generation [7]

KeyGen
$A \leftarrow R_q^{k \times \ell}$
$(s_1, s_2) \leftarrow S_\eta^\ell \times S_\eta^k$
$t := As_1 + s_2$
$(t_1, t_0) \leftarrow \text{Power2Round}_q(t, d)$
return $(pk = (A, t_1),\ sk = (A, t_0, s_1, s_2))$

Sign(sk, M)
$(z, h) := \perp$
while $(z, h) = \perp$ **do**
  $y \leftarrow S_{\gamma_1}^\ell$
  $w \leftarrow Ay$
  $w_1 \leftarrow \text{HighBits}(w)$
  $\tilde{c} \leftarrow H(M \| w_1)$
  $c \in B_\tau \leftarrow \text{SampleInBall}(\tilde{c})$
  $z \leftarrow y + cs_1$
  $r_0 \leftarrow \text{LowBits}(w - cs_2)$
  if $\|z\|_\infty \geq \gamma_1 - \beta$ **or** $\|r_0\|_\infty \geq \gamma_2 - \beta$ **then**
    $(z, h) \leftarrow \perp$
  else
    $h \leftarrow \text{MakeHint}(-ct_0, w - cs_2 + ct_0)$
    if $\|ct_0\|_\infty \geq \gamma_2$ **or** # of 1's in $h > \omega$ **then**
      $(z, h) \leftarrow \perp$
    end if
  end if
end while
return $\sigma = (\tilde{c}, z, h)$

B. ML-DSA Algorithm

ML-DSA is a post-quantum digital signature algorithm standardized by NIST as FIPS 204 [7]. The specification defines three security levels, ML-DSA-44, ML-DSA-65, and ML-DSA-87. All parameter sets operate over the ring R_q, where the scheme modulus is fixed to $q = 8{,}380{,}417$.

ML-DSA includes three routines: Key Generation, Signature Generation, and Verification [7]. Since the Verification routine does not use the secret key, it is not directly relevant for side-channel analysis. Algorithm 1 summarizes these Key and Signature Generation steps and highlights where the long-term secrets s_1 and s_2 are used. At a high level, Key Generation samples the secret polynomials s_1 and s_2 together with the public matrix A, and computes the public key component $t = As_1 + s_2$, which is decomposed into (t_1, t_0) for compression. The public key contains (A, t_1), while the secret key stores (A, t_0, s_1, s_2) for use during signing. During Signature Generation, the signer samples vector y and computes $w = Ay$. A challenge polynomial c is derived from the message M and a compressed view of w. The response values are then formed as $z = y + cs_1$ and $r_0 = \text{LowBits}(w - cs_2)$, and the procedure is repeated whenever the specified norm bounds are violated.

TABLE I: Hardware resource utilization of side-channel-protected modules in Adam's Bridge on a Xilinx Virtex UltraScale+ FPGA [31].

Module	LUTs	FFs	DSPs	BRAMs
Adam's Bridge	210,251	234,180	268	6
NTT	140,794	209,961	256	0
Modular Reduction*	54,246	98,958	0	0

*Aggregate resource utilization of all modular reduction units instantiated in the NTT engine.

C. Caliptra Root-of-Trust Subsystem

Caliptra is an open-source RoT subsystem designed for data center-class SoCs, and it is also considered to be used in IoT devices. It combines hardware IP and firmware to implement the RoT for Measurement inside the SoC. This RoT block provides device identity, measured boot, and attestation, and relies on a dedicated cryptographic subsystem to deliver these services [30]. The PQC subsystem, called Adam's Bridge [31].

Adam's Bridge contains an ML-DSA implementation. It is the first open-source hardware implementation that fully aligns with NIST's FIPS 204 specification, passes the NIST known-answer tests (KATs), and has undergone formal verification, making it the backbone of the Caliptra RoT. Adam's Bridge uses the highest ML-DSA parameter set, ML-DSA-87, to support post-quantum security. ML-DSA runs during secure boot, firmware verification, and key management, so it is one of the most performance and security-critical components in Caliptra. The open-source implementation includes side-channel countermeasures to protect its secret-dependent computations. These protections increase hardware cost and result in a $6\times$ area overhead for Adam's Bridge compared to the unprotected design [32], motivating a closer examination of the ML-DSA engine and its masked datapath.

III. Threat Model

In this study, we adopt the same threat model defined in the Adam's Bridge ML-DSA accelerator specification [31]. The attacker is assumed to observe physical side-channel leakages during ML-DSA key and signature generation, with full knowledge of the RTL (Register-Transfer Level) implementation. The model focuses on non-profiled attacks such as Differential Power Analysis (DPA), where the adversary collects many traces under a fixed secret key and correlates them with hypothesized intermediate values.

Under this model, all secret key components and all intermediates derived from them are treated as sensitive. Any operation where a public value interacts directly with a secret polynomial must be mitigated with side-channel countermeasures. We keep this protection boundary unchanged and do not modify which operations are considered sensitive. Instead, we focus on improving the efficiency of the masking implementation.

IV. Optimizing Side-Channel Protection in Adam's Bridge

In this section, we evaluate the ML-DSA implementation in Adam's Bridge to identify the main sources of performance and area overhead. Using these insights, we then compare alternative modular reduction schemes for ML-DSA and present our proposed architecture.

A. Adam's Bridge Side-channel Evaluation

During ML-DSA signing in Adam's Bridge, the challenge polynomial c and the secret polynomials s_1 and s_2 are processed by the NTT engine. The implementation first maps these polynomials to the NTT domain, second performs coefficient-wise multiplications, and finally applies the INTT to map the polynomials back to the non-NTT domain. Since multiplications make secret polynomials interact with public polynomials, Adam's Bridge protects these operations with masking and shuffling side-channel countermeasures [31].

The side-channel protection is applied to a single merged-layer radix-4 NTT engine. Adam's Bridge employs a resource-sharing architecture, reusing this engine to perform NTT, INTT, pointwise multiplication, pointwise addition, and pointwise subtraction. Adam's Bridge shuffling method randomizes the processing order of coefficient indices in the NTT, pointwise addition, pointwise subtraction, and the norm check operations. However, the masking countermeasure is only applied to the pointwise multiplication operation and the first stage of the INTT [31].

Table I reports our implementation-cost analysis. The NTT engine alone uses about 67% of the LUTs, 90% of the FFs, and 95% of the DSPs within Adam's Bridge. This shows that the masked NTT engine is the main cost driver of the ML-DSA accelerator. The masked NTT engine instantiates two masked reduction circuits for the first two BFUs in its radix-4 architecture. These reduction circuits are active for every pointwise multiplication and for the first stage of each INTT call. Because modular reduction is performed once per coefficient pair, the reduction unit is one of the most heavily exercised components in Adams Bridge during the signature generation flow (see lines 13, 14, and 18 in Algorithm 1). As shown in Table I, all modular reduction instances in the NTT engine together account for 54,246 LUTs and 98,958 FFs, which is about 38% of the NTT LUTs and 47% of its FFs. Within the arithmetic logic of the NTT, this makes masked modular reduction the most expensive block. Our implementation-cost analysis, therefore, identifies masked modular reduction as the dominant contributor to the overhead of side-channel protection in the NTT datapath.

B. Modular Reduction Techniques for ML-DSA

The choice of the modular reduction scheme plays a central role in overall efficiency. We first examine the custom modular reduction used in Adam's Bridge and then discuss alternative methods that are compatible with the ML-DSA datapath.

a) Custom Reduction: Adam's Bridge employs a custom reduction circuit optimized for the scheme's modulus $q = 2^{23} - 2^{13} + 1$ [31]. This circuit exploits the congruence $2^{23} \equiv 2^{13} - 1 \bmod q$ to realize modular reduction through shift–add operations without explicit division. However, the circuit was

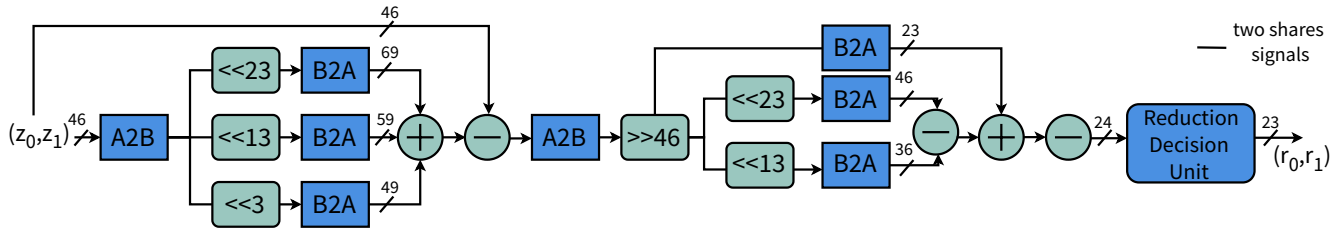

Fig. 1: Proposed first-order masked shift-based Barrett reduction architecture, where each line represents two shares processed in parallel. Two 46-bit masked shares (z_0, z_1) are processed through a pipelined datapath alternating between arithmetic and Boolean domains via A2B and B2A conversions. Shift–add operations ($\ll 23, \ll 13, \ll 3$) approximate the Barrett quotient without multipliers.

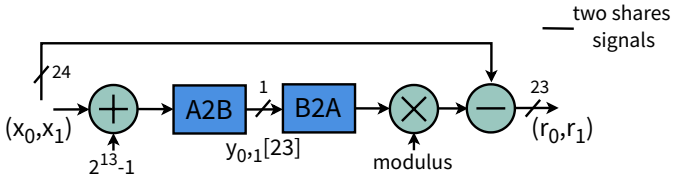

Fig. 2: The Reduction decision unit performs the final correction without combining the shares. It checks the 23rd bit of the accumulated value after adding $2^{13} - 1$ to detect overflow. This bit is converted back to the arithmetic domain and used to conditionally subtract q.

not originally designed for masking-based side-channel protection. A CPA attack [17] demonstrated exploitable leakage in the unprotected reduction unit. Later revisions introduced masking and shuffling countermeasures; however, these were not co-optimized with the arithmetic datapath, resulting in increased latency and costly A2B/B2A conversions due to the heavy interaction between Boolean and arithmetic operations.

a) Montgomery Reduction: Montgomery reduction [36] replaces modular division with shifts and multiplications by operating in a transformed residue domain. The method computes $r = (x \cdot y \cdot R^{-1}) \bmod q$, where $R = 2^n > q$, thereby avoiding explicit division by q. This approach is efficient for general modular arithmetic and software implementations. However, it requires additional conversions between the standard and Montgomery domains, which increases the complexity and adds performance overhead compared to other reduction mechanisms. Therefore, it is also not preferred in the baseline Adam's Bridge [31].

a) Barrett Reduction: Barrett reduction [37] is a generic technique that removes trial division by replacing it with a multiplication by a precomputed reciprocal μ. Given an integer x such that $0 \le x < q^2$, the method precomputes

$$\mu = \left\lfloor \frac{2^{2n}}{q} \right\rfloor,$$

where $n = \lceil \log_2 q \rceil$ is the bit length of the modulus. An approximation of the quotient $\lfloor x/q \rfloor$ is then obtained as

$$m = \left\lfloor \frac{x \cdot \mu}{2^{2n}} \right\rfloor,$$

and the reduced value is computed as $r = x - m \cdot q$. This approach works in the standard residue domain and does not require domain conversions.

Algorithm 2 Barrett Reduction for ML-DSA [25]

Input: $0 \le z < q^2$
Output: $r = z \bmod q$
$t \leftarrow (z \ll 23) + (z \ll 13) + (z \ll 3) - z$
$t \leftarrow t \gg 46$
$t \leftarrow (t \ll 23) - (t \ll 13) + t$
$r \leftarrow z - t$
if $r \ge q$ **then**
    $r \leftarrow r - q$
end if
return r

For ML-DSA's modulus q, the reciprocal constant can be chosen as

$$\mu = 2^{23} + 2^{13} + 2^3 - 1.$$

This special form enables the multiplication by μ to be implemented as a sequence of bit shifts and additions, eliminating the need for large multipliers. It leads to a regular shift-based data path that fits well into a pipelined masking architecture. Shifts are handled in the Boolean domain and additions are handled in the arithmetic domain, so the interaction between the two domains is structured and limited. As a result, the number of Boolean-to-arithmetic conversions is reduced compared to other reductions, which lowers the cost of A2B and B2A gadgets. The resulting shift-based Barrett variant, presented in [25] and summarized in Algorithm 2, forms the basis for our first-order masked reduction.

C. First-Order Masked Barrett Reduction Architecture

Figure 1 shows the proposed first-order masked, shift-based Barrett reduction architecture, which serves as a drop-in replacement for the modular reduction unit in Adam's Bridge [31]. The circuit reduces 46-bit operands modulo q while preserving first-order masking across all pipeline stages. It provides a masked hardware implementation of the shift-based Barrett algorithm summarized in Algorithm 2.

The module receives two 46-bit arithmetic shares (z_0, z_1) that represent the intermediate product z. These shares are first converted to the Boolean domain by an A2B block. A2B and B2A converters maintain masking when switching between Boolean and arithmetic domains, and each conversion utilizes fresh randomness, ensuring that the two shares remain

TABLE II: Resource and performance comparison between the side-channel protected ML-DSA design in Adam's Bridge [31] and this work.

Design	LUT	FF	BRAM	DSP	Freq (MHz)†	KeyGen (Cycles / μs)	Sign (Cycles / μs)	Verify (Cycles / μs)
[31]	210,251	234,180	6	268	400	15,528 / 38.82	145,286 / 363.22	18,720 / 46.80
This Work	**163,332**	**144,917**	6	276	400	15,528 / 38.82	**110,196 / 262.99**	18720 / 46.80

†The frequency remains unchanged from the baseline design because the NTT is not on the critical path. The proposed optimization targets the modular reduction unit in NTT and therefore does not affect the overall timing of the ML-DSA accelerator.

TABLE III: Hardware cost and latency of the side-channel-protected modular reduction and NTT modules in Adam's Bridge [31] and in this work.

Design	Module*	LUTs	FFs	DSPs	Cycles	Freq. (MHz)
[31]	Reduction	9,041	16,493	0	157	610
	NTT	140,794	209,961	256	595	408
This Work	Reduction	1,201	911	2	13	604
	NTT	93,875	120,698	264	401	402

*The reported cycle count for NTT corresponds to the INTT operation, since the masked modular reduction is utilized in the INTT rather than the NTT mode.

statistically independent throughout the pipeline. The next stages create shifted versions of z using the $\ll 23$, $\ll 13$, and $\ll 3$ blocks and combine them through additions and subtractions to form an intermediate value. This value is then right-shifted so that only its high-order bits remain, and these bits serve as an approximate quotient for the Barrett reduction. Subsequent stages use this quotient together with another sequence of shifts and add or subtract operations to reconstruct the correction term. The circuit subtracts this reconstructed term from the original input shares to produce a masked candidate result, which is passed to the reduction decision unit for final correction.

In a standard Barrett implementation, the final step performs a conditional subtraction of q. If the candidate result x is at least q, the circuit must subtract q once to bring the value back into $[0, q)$. A straightforward hardware design would use a comparator and a multiplexer, or an explicit branch in the control logic. Under masking, such structures are costly. A masked comparator requires wide logic and additional randomness, which complicates pipelining. Our goal is to solve this reduction decision problem without using wide masked comparators or control-flow decisions.

The reduction decision unit, shown in Figure 2, implements this idea and performs the conditional subtraction without explicit comparators or multiplexers. It starts from the masked intermediate value (x_0, x_1) in the arithmetic domain and first adds the constant $2^{13} - 1$ to the shares. This offset exploits the special form of the modulus $q = 2^{23} - 2^{13} + 1$, so that the most significant carry bit of the sum encodes whether the intermediate value has reached or exceeded q. The resulting 24-bit value is then converted to the Boolean domain through an A2B converter. In this domain, the carry bit at position 23, denoted by $y_{0,1}[23]$, serves as a single-bit predicate indicating whether the accumulated sum exceeds q.

This predicate is converted back to the arithmetic domain via a B2A converter and multiplied by the modulus q to generate the conditional subtraction term. If the predicate is zero, the product is zero. If it is one, the product equals q. Finally, it subtracts this product from the input shares to yield the reduced output (r_0, r_1). All computations remain in their respective masked domains, and only a single predicate bit is exposed in the Boolean domain. This design avoids wide masked comparisons, fits naturally into our pipeline, and provides an efficient masked correction step tailored to the ML-DSA modulus q.

V. Experimental Result

This section presents the hardware implementation results of the proposed masked modular reduction and its integration within the ML-DSA accelerator, along with a performance analysis and a side-channel evaluation. All experiments are conducted under identical synthesis and testing conditions to ensure a fair comparison with the open-source baseline design.

A. Implementation Result

All implementation results are obtained using Xilinx Vivado 2024.1 targeting the XCU26-VSVA1365-2L-E FPGA under identical synthesis and place-and-route constraints. Table III summarizes the synthesis results of the modular reduction and NTT modules. Compared to the custom modular reduction used in Adam's Bridge, the proposed masked Barrett reduction delivers substantial efficiency gains. It reduces LUT and FF utilization by 86.7% and 94.5%, respectively, and lowers the reduction latency from 157 cycles to 13. This corresponds to a $12.1\times$ speedup and a $91\times$ improvement in area–delay efficiency compared to the baseline design. DSP utilization increases by two units due to the multiplier used in the final correction stage, which represents a negligible overhead compared to the overall design. For the NTT module, integrating the proposed masked reduction path reduces LUT and FF usage by 33.3% and 42.5%, respectively, compared to the baseline NTT implementation. It also shortens the INTT operation from 595 cycles to 401 cycles. Overall, these results confirm that the proposed masked reduction achieves significant savings and that its integration improves the NTT module's area and latency while preserving first-order masking across all modes.

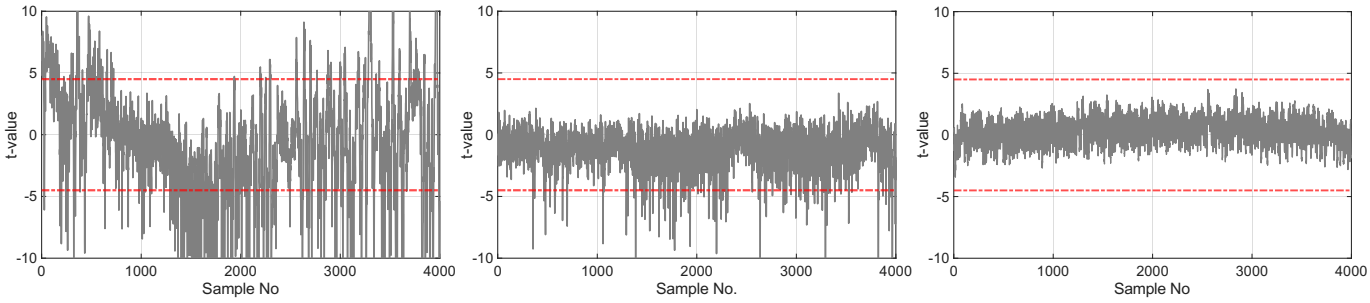

Fig. 3: Fixed-vs-random TVLA results for the pointwise multiplication operation under different countermeasure configurations. The three plots correspond to the following settings: (a) randomness is turned off with 3,000 traces, (b) shuffling is enabled, and masking is disabled with 15,000 traces, and (c) both masking and shuffling are enabled with 1,000,000 traces.

Table II shows the impact of integrating our masked reduction unit into the ML-DSA accelerator [31]. Both designs utilize the same synthesis setup and operate at 400 MHz. The clock frequency remains unchanged, since the optimized NTT is not on the critical path. The proposed masked design reduces LUT usage by 22.3% and FF usage by 38.1% compared to the baseline reduction implementation. The LUT count drops from 210,251 to 163,332, and the FF count drops from 234,180 to 144,917. DSP usage increases from 268 to 276, but this is a minor trade-off given the overall improvement. Signing latency decreases from 145,286 cycles to 110,196 cycles, giving a $1.32\times$ throughput gain with no changes to key generation or verification. We improve the area, reduce latency, and increase signing throughput, all while maintaining Caliptra's protection boundary and system architecture.

B. Side-channel Evaluation

We performed the side-channel evaluation using the CW305 ChipWhisperer FPGA platform and a PicoScope 6404D oscilloscope. Because the complete ML-DSA accelerator exceeds the resource capacity of available evaluation boards, we performed our testing on a single merged-layer radix-4 NTT engine, which is the only side-channel protected module in Adam's Bridge. During acquisition, the pointwise multiplication was tested under three protection settings, and all measurements were analyzed using the fixed-vs-random TVLA procedure with a threshold of ± 4.5. When randomness is turned off, clear first-order leakage appears within about 3,000 traces, as shown in Figure 3a. With shuffling enabled but masking still disabled, the t-values cross the threshold after 15,000 traces, as seen in Figure 3b. When both masking and shuffling are enabled, no t-value excursions beyond the threshold are observed up to 1,000,000 traces, as illustrated in Figure 3c, indicating no detectable first-order leakage. This matches the evaluation methodology used in the reference Adam's Bridge design, which also reports no first-order leakage up to one million traces [31]. Our results, therefore, meet the same leakage expectations as the baseline system with a significant area-delay improvement.

VI. Discussion

A. Choice of Leakage Evaluation Method

We use TVLA as our main method for detecting side-channel leakage. TVLA is widely used in literature for the leakage assesment [27], [28], and the Adam's Bridge reference design also relies on TVLA to validate its countermeasures [31]. Using the same procedure ensures that our results are directly comparable to the baseline and demonstrate that we meet the same leakage expectations.

We do not claim that TVLA is perfect or the best option in all cases. Other leakage detection methods exist and can also be applied, but they often need a detailed power model or controlled changes to secret data [38], [39]. We chose TVLA because it avoids these extra assumptions and is practical for our hardware setting. At the same time, we are aware that its moment-based test can miss some forms of leakage or raise false alarms [33], [40]. More advanced analyses or higher-order evaluations are interesting directions for future work.

B. Applicability Beyond ML-DSA

Our implementation and evaluation focus on the ML-DSA engine in Adam's Bridge, as this block is fully integrated in the current Caliptra release [30]. The proposed masked Barrett reduction is instantiated for the ML-DSA modulus q, but its shift-based structure and masking strategy are generic. In particular, the same construction can be retargeted to other prime moduli, such as those used in ML-KEM, by adapting the precomputed constant μ and the shift pattern in the unmasked Barrett core. A detailed ML-KEM instantiation following this approach is beyond the scope of this work and is left for future work.

VII. Conclusion

In this work, we addressed the cost of side-channel countermeasures in Caliptra's ML-DSA accelerator [31]. We identified modular reduction as the dominant source of overhead and introduced a first-order masked, shift-based Barrett reduction as a drop-in replacement. The resulting design reduces the overhead of the reduction unit and improves the overall efficiency of the ML-DSA datapath. At the same time, it continues

to meet the first-order leakage requirements expected in Caliptra. Our results show that carefully designed masked arithmetic can make side-channel-resistant post-quantum signatures more practical in real-world hardware security modules.

VIII. Acknowledgment

This work is supported in part by NSF-2147196.

References

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 124–134, 1994.

[4] D. Moody, "Post-quantum cryptography standardization: Announcement and outline of NIST's call for submissions," in *International Conference on Post-Quantum Cryptography (PQCrypto)*, 2016.

[5] National Institute of Standards and Technology, "FIPS 203: Module-lattice-based key-encapsulation mechanism standard," tech. rep., U.S. Department of Commerce, Aug. 2024. Accessed: 2025-06-08.

[6] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. Bos, J.-C. Deneuville, A. Dion, P. Gaborit, J. Lacan, E. Persichetti, J.-M. Robert, P. Véron, and G. Zémor, "Hamming quasi-cyclic (hqc): Fourth round specification," tech. rep., NIST Post-Quantum Cryptography Standardization Project, February 2025. Fourth Round Version, updated on 19 February 2025.

[7] National Institute of Standards and Technology, "FIPS 204: Module-lattice-based digital signature standard," tech. rep., U.S. Department of Commerce, Aug. 2024. Accessed: 2024-11-16.

[8] P. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "FALCON: Fast-Fourier lattice-based compact signatures over NTRU, specification v1.2," Technical Report v1.2, École normale supérieure, Paris, October 2020.

[9] National Institute of Standards and Technology, "FIPS 205: Stateless hash-based digital signature standard," tech. rep., U.S. Department of Commerce, Aug. 2024. Accessed: 2025-06-08.

[10] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay, and S. Bhasin, "Side-channel assisted existential forgery attack on dilithium - a NIST PQC candidate." Cryptology ePrint Archive, Paper 2018/821, 2018.

[11] A. Fournaris, C. Dimopoulos, and O. Koufopavlou, *Profiling Dilithium Digital Signature Traces for Correlation Differential Side-Channel Attacks*, pp. 281–294. Oct. 2020.

[12] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing, "An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 583–590, 2021.

[13] Z. Qiao, Y. Liu, Y. Zhou, J. Ming, C. Jin, and H. Li, "Practical public template attacks on crystals-dilithium with randomness leakages," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1–14, 2023.

[14] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," in *Progress in Cryptology–LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security, Proceedings 6*, pp. 130–149, Springer, 2019.

[15] C. Mujdei, A. Beckers, J. M. B. Mera, A. Karmakar, L. Wouters, and I. Verbauwhede, "Side-channel analysis of lattice-based post-quantum cryptography: Exploiting polynomial multiplication." Cryptology ePrint Archive, Paper 2022/474, 2022.

[16] H. Steffen, G. Land, L. Kogelheide, and T. Güneysu, "Breaking and protecting the crystal: Side-channel analysis of dilithium in hardware." Cryptology ePrint Archive, Paper 2022/1410, 2022.

[17] M. Karabulut and R. Azarderakhsh, "Efficient cpa attack on hardware implementation of ml-dsa in post-quantum root of trust," in *2025 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 392–402, 2025.

[18] H. Wang, Y. Gao, Y. Liu, *et al.*, "In-depth correlation power analysis attacks on a hardware implementation of crystals-dilithium," *Cybersecurity*, vol. 7, no. 21, 2024.

[19] M. Azouaoui, O. Bronchain, G. Cassiers, C. Hoffmann, Y. Kuzovkova, J. Renes, M. Schönauer, T. Schneider, F.-X. Standaert, and C. van Vredendaal, "Protecting dilithium against leakage: Revisited sensitivity analysis and improved implementations." Cryptology ePrint Archive, Paper 2022/1406, 2022.

[20] J.-S. Coron, F. Gérard, M. Trannoy, and R. Zeitoun, "Improved gadgets for the high-order masking of dilithium." Cryptology ePrint Archive, Paper 2023/896, 2023.

[21] J.-S. Coron, F. Gérard, T. Lepoint, M. Trannoy, and R. Zeitoun, "Improved high-order masked generation of masking vector and rejection sampling in dilithium." Cryptology ePrint Archive, Paper 2024/1149, 2024.

[22] J. Lee, J. Han, S. Lee, J. Kwon, K.-H. Choi, J.-W. Huh, J. Cho, and D.-G. Han, "Systematization of shuffling countermeasures: With an application to crystals-dilithium," *IEEE Access*, vol. 11, pp. 142862–142873, 2023.

[23] L. Beckwith, D. T. Nguyen, and K. Gaj, "High-performance hardware implementation of CRYSTALS-dilithium." Cryptology ePrint Archive, Paper 2021/1451, 2021.

[24] Z. Wu, R. Chen, Y. Wang, Q. Wang, and W. Peng, *An Efficient Hardware Implementation of Crystal-Dilithium on FPGA*, pp. 64–83. 07 2024.

[25] K. Miteloudi, J. W. Bos, O. Bronchain, B. Fay, and J. Renes, "Pq.v.alu.e: Post-quantum risc-v custom alu extensions onnbsp;dilithium andnbsp;kyber," p. 190–209, 2023.

[26] Q. D. Truong, P. N. Duong, and H. Lee, "Efficient low-latency hardware architecture for module-lattice-based digital signature standard," *IEEE Access*, vol. 12, pp. 32395–32407, 2024.

[27] T. Kamucheka, A. Nelson, D. Andrews, and M. Huang, "A masked pure-hardware implementation of kyber cryptographic algorithm." Cryptology ePrint Archive, Paper 2022/1547, 2022.

[28] K. Raj, P. Ravi, T. K. Chia, and A. Chattopadhyay, "Improved ML-DSA hardware implementation with first order masking countermeasure." Cryptology ePrint Archive, Paper 2024/1817, 2024.

[29] D. Xu, K. Wang, and J. Tian, "A hardware-friendly shuffling countermeasure against side-channel attacks for kyber," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 72, p. 504–508, Mar. 2025.

[30] C. A. Project, "Caliptra: A datacenter system-on-a-chip (soc) root-of-trust (rot)," 2025. Revision 2.0, Version 0.5. Available: https://github.com/chipsalliance/Caliptra, Accessed: 2025-10-25.

[31] C. Alliance, "Adams-bridge," 2024. Available: https://github.com/chipsalliance/adams-bridge, Accessed: 2025-10-20.

[32] C. Swenson, J. Andersen, and M. Bisheh-Niasar, "Cryptography in caliptra," 2025. Technical presentation.

[33] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003*, pp. 463–481, 2003.

[34] O. Bronchain and G. Cassiers, "Bitslicing arithmetic/boolean masking conversions for fun and profit: with application to lattice-based kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, pp. 553–588, 08 2022.

[35] J. Liu, C. Zhao, S. Peng, B. Yang, H. Zhao, X. Han, M. Zhu, S. Wei, and L. Liu, "A low-latency high-order arithmetic to boolean masking conversion," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, pp. 630–653, 03 2024.

[36] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, pp. 519–521, 1985.

[37] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *Advances in Cryptology — CRYPTO' 86*, pp. 311–323, 1987.

[38] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, pp. 426–442, 2008.

[39] G. Hospodar, B. Gierlichs, E. D. Mulder, I. M. R. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, pp. 293–302, 2011.

[40] F.-X. Standaert, *How (Not) to Use Welch's T-Test in Side-Channel Security Evaluations: Studies on Socio-Ecological Systems' Vulnerability, Resilience and Governance*, pp. 65–79. 01 2019.