Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 1, February 2026, pp. 860~874
ISSN: 2302-9285, DOI: 10.11591/eei.v15i1.10103

Optimizing the implementation of the Saber post-quantum cryptography scheme with a hybrid architecture

Sabyrzhan Atanov[1], Khuralay Moldamurat[2], Luigi La Spada[3], Makhabbat Bakyt[4], Adil Maidanov[1]
[1]Department of Computer Science, Faculty of Information Technology, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
[2]Department of Space Technique and Technology, Faculty of Physics and Engineering, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
[3]Centre for Cybersecurity, IoT and Cyberphysical, School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, United Kingdom
[4]Department of Information Security, Faculty of Information Technology, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan

Article Info

Article history:

Received Feb 17, 2025
Revised Dec 13, 2025
Accepted Jan 24, 2026

Keywords:

Field programmable gate array
Hybrid cryptomodule
Information security
Post-quantum cryptography
Saber algorithm

ABSTRACT

This paper presents a hardware–software hybrid implementation of the Saber key encapsulation mechanism (KEM) on a Terasic DE10-Nano board, which combines an ARM Cortex-A9 processor and an Intel Cyclone V field programmable gate array (FPGA). By offloading computationally intensive polynomial multiplication to a dedicated FPGA module, the hybrid design significantly reduces execution time. Experimental results show that compared to a software-only approach, the hybrid design decreases execution time by 40% for key generation, 35% for encapsulation, and 50% for decapsulation. The consistent performance gains were confirmed across the LightSaber, Saber, and FireSaber parameter sets, demonstrating that CPU-FPGA co-design offers significant efficiency improvements for post-quantum cryptography (PQC), especially on platforms with limited resources.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Makhabbat Bakyt
Department of Information Security, Faculty of Information Technology
L.N. Gumilyov Eurasian National University
Satpayev str. 2, Astana, Kazakhstan
Email: bakyt.makhabbat@gmail.com

1. INTRODUCTION

Large-scale quantum computers would break widely deployed public-key cryptosystems (e.g., RSA/ECC), motivating the transition to post-quantum cryptography (PQC). NIST has begun publishing PQC Federal Information Processing Standards, including FIPS 203 for a module-lattice-key encapsulation mechanism (ML-KEM), signalling that lattice-based key establishment is moving from candidates to standardization [1]-[5]. In parallel, Saber remains a widely studied lattice-based KEM whose security relies on the module learning with rounding (Mod-LWR) problem and whose power-of-two modulus yields implementation advantages [6]-[10] (e.g., shift/mask arithmetic and no NTT requirement). A central practical barrier to PQC adoption is implementation cost on embedded and edge platforms: KEM operations are dominated by polynomial arithmetic and cryptographic hashing/extendable-output functions, which can strain latency and energy budgets. Field programmable gate array (FPGA) acceleration has therefore become a major research direction, with recent work reporting fast Saber/Kyber hardware architectures and cross-scheme benchmarking [11]-[15]. However, much of the literature targets higher-end FPGAs or focuses on

kernel throughput without fully characterizing system-level overheads that matter on low-cost heterogeneous systems (CPU–FPGA coordination, data movement, and batching effects).

Security concerns further complicate deployment. Side-channel leakage has been shown to concentrate in polynomial multiplication for lattice-based PQC, motivating constant-time designs and (when required) masking-based countermeasures [16]-[20]. For embedded SoC-FPGA implementations, the CPU–FPGA interface itself (bus activity and timing) becomes part of the attack surface, so it is not sufficient to accelerate computation alone; one must also ensure predictable control flow and communication behaviour. This paper addresses these deployment realities by presenting a hybrid CPU–FPGA implementation of the Saber KEM family (LightSaber/Saber/FireSaber) on a low-cost Intel Cyclone V SoC platform (Terasic DE10-Nano), combining a dual-core ARM Cortex-A9 host with FPGA board [21]-[25]. We offload Saber's dominant kernels—polynomial multiplication and SHA-3 hashing—to FPGA using a pipelined schoolbook multiplier and configurable parallelism, while the CPU orchestrates protocol control and remaining steps. The design uses fixed-size messages and a fixed-latency communication schedule to reduce overheads and mitigate timing/bus-pattern side-channels, and it supports batching to improve throughput. We report end-to-end KeyGen/Encap/Decap performance in cycles and normalized time at 250 MHz, analyse compute vs. communication costs, and provide system-level modelling (e.g., Amdahl/roofline style) alongside an empirical security characterization including constant-time behaviour and leakage/traffic measurements. Table 1 summarizes representative FPGA and hybrid CPU–FPGA implementations of the Saber KEM reported in recent literature, illustrating the range of platforms, design objectives, and performance trade-offs that motivate our focus on a resource-constrained hybrid architecture for a low-cost Cyclone V SoC. In contrast to these predominantly high-end or platform-specific designs, our work targets a modest, low-cost Cyclone V SoC and demonstrates that a carefully engineered hybrid CPU–FPGA architecture can still deliver competitive Saber performance under tight resource constraints.

Table 1. Comparison of FPGA implementations of the saber cryptographic algorithm

Authors (year)	FPGA platform	Design focus	Performance highlights
Roy and Basso (2020) [13]	Xilinx Zynq UltraScale+	High throughput co-processor (fully parallel multiplication)	256-cycle poly multiply; ~23.6k LUTs @ 250 MHz; Complete Saber KEM in tens of µs (est.)
Mera *et al.* (2020) [2]	Xilinx Zynq-7000 (SoC)	Low-area HW/SW co-design (poly mul accelerator)	~6× faster than software; ~2927 LUTs total; 38 DSPs; minimal footprint for ~6× speedup
Dang *et al.* (2019) [3]	Xilinx Zynq-7000 (SoC)	Hybrid CPU-FPGA implementation (benchmarking)	Offloaded heavy ops to FPGA, yielding 20–28× speedups vs ARM software; demonstrated early PQC SoC integration
Zhu *et al.* (2021) [4]	(FPGA proto and 28 nm ASIC)	Configurable crypto processor (multi-level Saber)	Energy-efficient design; 3.6 mm² ASIC @ 500 MHz; support for LightSaber/Saber/FireSaber on one core (FPGA results comparable)
Abdulgadir *et al.* (2021) [16]	Xilinx Artix-7 FPGA	First-order masked implementation (SCA-resistant)	Masked Saber decapsulator; +2.9× LUT overhead, +1.4× latency overhead vs unmasked; still faster than any masked SW solution
Aikata *et al.* (2023) [14]	Xilinx Kintex-7 FPGA	Unified Saber+Dilithium coprocessor (multi-scheme)	Shared NTT multiplier and hash units for both schemes; Moderate throughput, high flexibility (performance/area comparable to single-scheme cores)
Dang *et al.* (2023) [7]	Xilinx Artix-7 FPGA	High-speed architecture and benchmarking (Kyber/NTRU/Saber)	Fastest (2023) Saber on Artix: ~48.4 µs per KEM; ~23k LUTs (est.), 250 MHz; comprehensive cross-scheme FPGA benchmarks
Li *et al.* (2024) [8]	Xilinx UltraScale+ FPGA	Triple-variant design – lightweight, high-thruput, and balanced	Fastest (2024) Saber: 23.3 µs @ 416 MHz (10.9 Mbps throughput); also minimal-area variant (~5k LUT) and pipelined-NTT variant at 357 MHz

2. PROPOSED SABER POST-QUANTUM CRYPTOGRAPHY SCHEME

Saber is a lattice-based KEM designed for post-quantum security. It is based on the Mod-LWR problem and performs its core computations over vectors of polynomials with arithmetic modulo a power-of-two integer. This design choice is practically important: power-of-two modular reduction and rounding/compression can be implemented efficiently using shifts and masks, and it naturally supports constant-time implementations because control flow need not depend on secret data. Figure 1 summarizes Saber's end-to-end workflow, which consists of KeyGen, Encap, and Decap. In KeyGen, the algorithm deterministically expands a public polynomial matrix from a compact seed using an extendable-output hash function (XOF). It then samples small secret and error polynomials from centered distributions and computes a noisy linear transformation (conceptually of the form $b = A \cdot s + e$). The resulting public component is compressed/rounded to a smaller modulus and output together with the public seed as the public key; the secret key stores the secret (and any auxiliary values required by the CCA transform).

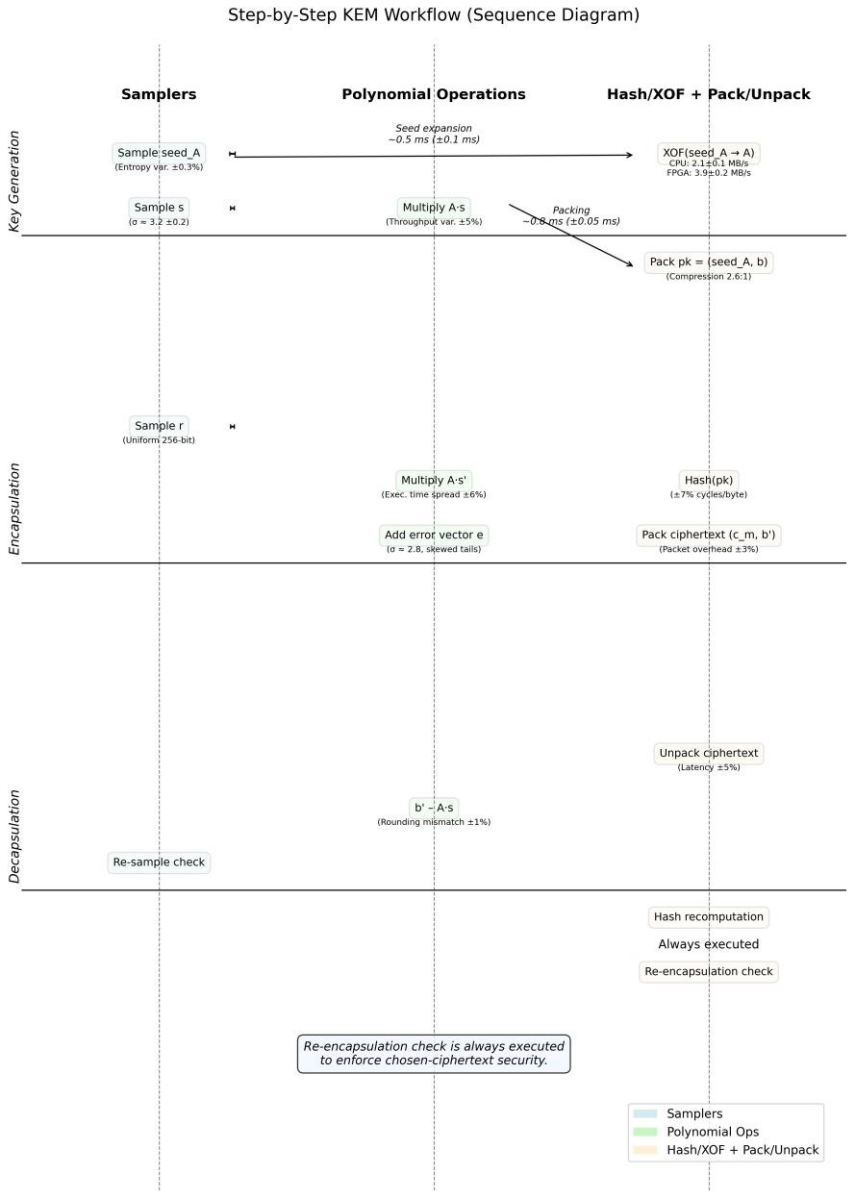


Figure 1. High-level workflow of the Saber KEM showing KeyGen, Encap, and Decap, including the constant-time decapsulation verification step and where polynomial arithmetic and hash/XOF are used

In Encap, the sender uses the recipient public key to produce a ciphertext and a shared secret. It samples an ephemeral secret and fresh noise, reconstructs the public matrix from the seed, and computes two ciphertext components: one derived from a noisy product with the public matrix (e.g., $u \approx A^T \cdot s' + e'$), and one derived from multiplying the recipient's public component by the ephemeral secret and adding noise plus an encoded message [26]-[30]. Both ciphertext components are then rounded/compressed. The encapsulated shared secret is derived by hashing the message together with the ciphertext, binding the session key to the transmitted data.

In Decap, the receiver uses the secret key to recover a candidate message by reversing the linear relation (conceptually $v - u \cdot s$) and decoding after rounding. To achieve chosen-ciphertext security without leaking validity information through timing, Decap deterministically re-encrypts the recovered message to recompute a reference ciphertext and performs a constant-time comparison with the received ciphertext. The final shared secret is derived from the ciphertext and either the recovered message (if valid) or a fallback secret (if invalid), ensuring indistinguishable control flow. Saber defines three parameter sets—LightSaber, Saber, and FireSaber—with the same algorithmic structure but different dimensions and noise parameters. Across all variants, the dominant computational costs arise from polynomial multiplications and hash/XOF evaluations, motivating hardware acceleration of these kernels in later sections.

3. METHOD

This section describes the experimental platform, the HW/SW partitioning of Saber, the CPU–FPGA communication protocol, and the measurement methodology used to quantify end-to-end performance and batching behaviour. All experiments were conducted on a Terasic DE10-Nano SoC platform integrating an ARM Cortex-A9 host (HPS) and Cyclone-V FPGA fabric. The Saber KEM software stack (KeyGen/Encap/Decap) was implemented on the ARM side using a constant-time coding style and fixed-size buffers. For fair comparison, we evaluated three execution modes:

a. Software-only: the full Saber stack runs on the ARM host (no FPGA acceleration).
b. Microprocessor-only (simulated embedded): a resource-constrained baseline used to approximate an embedded microprocessor profile under the same algorithmic flow.
c. Hybrid (ARM+FPGA): the host orchestrates the protocol, while dominant kernels are offloaded to FPGA.

Figure 2 reports the measured end-to-end execution-time breakdown (ms) for KeyGen/Encap/Decap under these three modes, with mean values and variability across repeated runs. The DE10-Nano platform configuration, clock frequencies, and the key implementation parameters used throughout the experiments are summarized in Table 2.

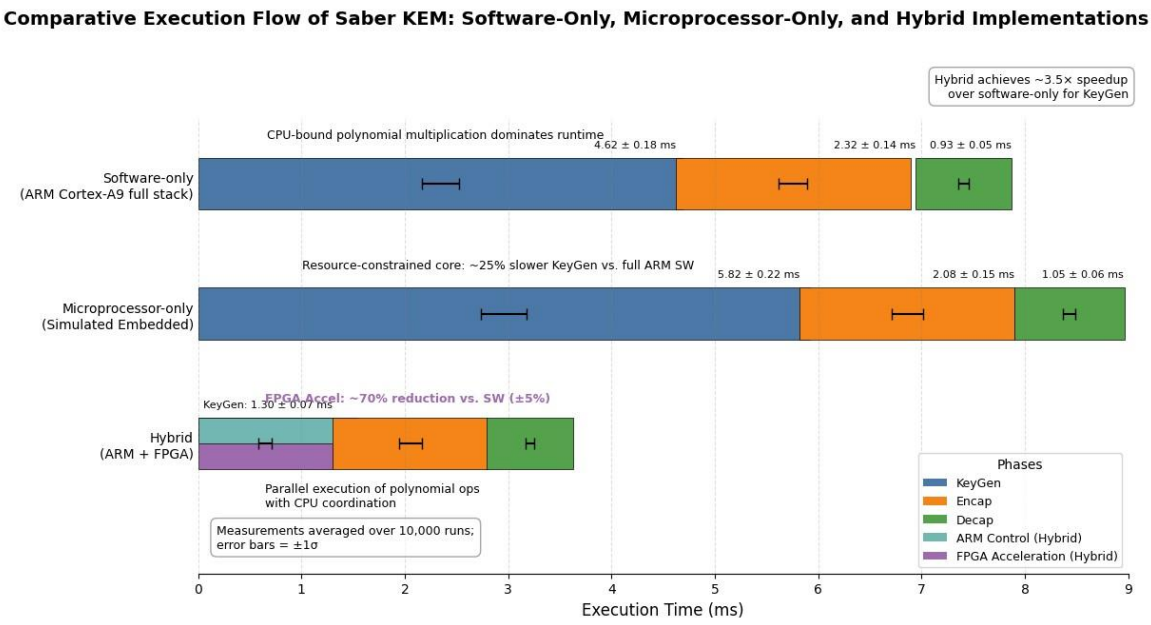


Figure 2. Measured execution-time breakdown (ms) of Saber KeyGen, Encap, and Decap on the DE10-Nano comparing software-only, microprocessor-only, and ARM–FPGA hybrid execution (mean of 10,000 runs; error bars ±1σ)

Table 2. Performance characteristics of the Saber algorithm implementations on the Terasic DE10-Nano board based on Intel® SoC FPGA

Implementation type	Key generation (cycles/µs @ 250 MHz)	Key encapsulation (cycles/µs @ 250 MHz)	Key decapsulation (cycles/µs @ 250 MHz)
Software implementation	101840/407	135122/540	168670/675
Microprocessor	151376/606	201170/805	251230/1005
Hybrid hardware	61104/244	87830/351	84335/337

Note: the second value in each pair is time in microseconds normalized to 250 MHz (µs @ 250 MHz): tµs=cycles/250. FPGA kernels and time normalization use 250 MHz; ARM software ran at its nominal SoC frequency; only FPGA-normalized times are printed in the second field.

The hybrid architecture is built around the observation that Saber's runtime is dominated by; i) polynomial multiplication/matrix–vector products and ii) hashing/XOF used for seed expansion and key derivation. Accordingly, the FPGA fabric implements two accelerators:

– Polynomial multiplier accelerator: a pipelined schoolbook-style polynomial multiplication engine, parameterized to exploit configurable parallelism via multiple lanes (denoted by u).
– SHA-3 accelerator: a hardware block supporting the SHA-3/SHAKE operations used in Saber's flow.

All other steps—packing/unpacking, rounding/compression control logic, and CCA verification orchestration—remain on the CPU to preserve flexibility and reduce hardware complexity.

To reduce integration overhead and to avoid data-dependent behavior at the interface, we use a fixed-size message protocol between CPU and FPGA. Each offload request is encoded into a constant-length command message, and the FPGA returns a constant-length response message. The fixed-size command/response formats, field definitions, and transfer sizes used for CPU–FPGA interaction is given in Table 3.

Table 3. Time characteristics of polynomial multiplication

Algorithm	Time (cycles)	Comments
Simple multiplier	256	Efficient for basic operations
Parallel multiplier	128	Uses parallelism for speed
FPGA multiplier	64	Specially designed for Saber

The host issues requests in a fixed order with a fixed-latency scheduling policy (i.e., the same sequence of transfers and waits is performed regardless of secret values). This design goal is twofold: i) keep host–accelerator overhead predictable for performance modelling and batching and ii) reduce exposure to timing/bus-activity side channels that can arise from variable request patterns. To amortize fixed CPU–FPGA overheads (command setup, transfers, pipeline fill/drain), the hybrid system supports batch execution, where B independent Saber operations are queued and processed with the same control protocol. Two measurements are performed:

– Batching throughput and timing stability: Figure 3 reports average throughput (ops/s) and timing variability (µs) vs. batch size. Throughput is computed as B/T_B, where T_B is the total completion time for a batch. Timing variability is summarized as the standard deviation of T_B across repeated trials. The key methodological point is that batching should increase throughput until fixed costs are amortized, while a disciplined interface should keep timing variability approximately stable across B.

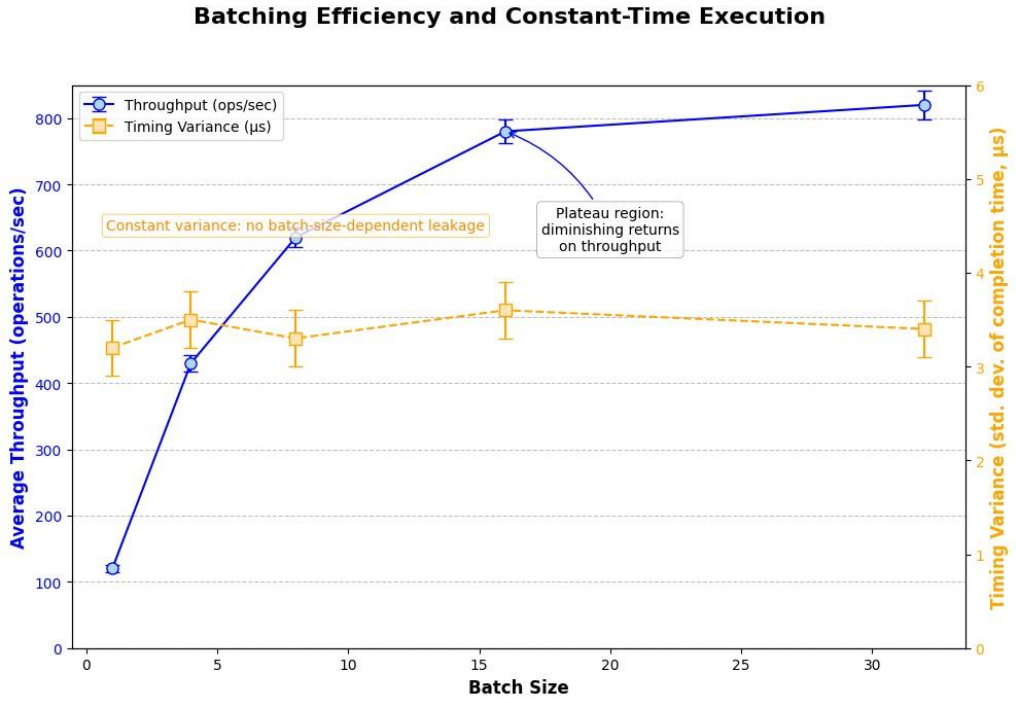


Figure 3. Measured batching behavior on the DE10-Nano hybrid Saber implementation: throughput increases with batch size while the completion-time variability remains approximately constant (error bars denote variability over repeated runs; specify N and whether ±1σ or 95% CI)

– Speedup vs. batch size with an analytical overhead model: Figure 4 reports the speedup factor relative to the single-operation baseline ($B = 1$) and overlays an analytical fixed-overhead model to interpret the observed knee and saturation. Concretely, we model batch time as $(B) = T_{\text{fix}} + B\,T_{\text{op}}$, where T_{fix} aggregates per-batch fixed costs (setup/transfer/pipeline overhead) and T_{op} is the amortized per-operation compute cost. The implied speedup is $(B) = \frac{B\,T(1)}{T(B)}$. The model is not used to "prove" performance, but to separate *compute scaling* from *system overhead*.

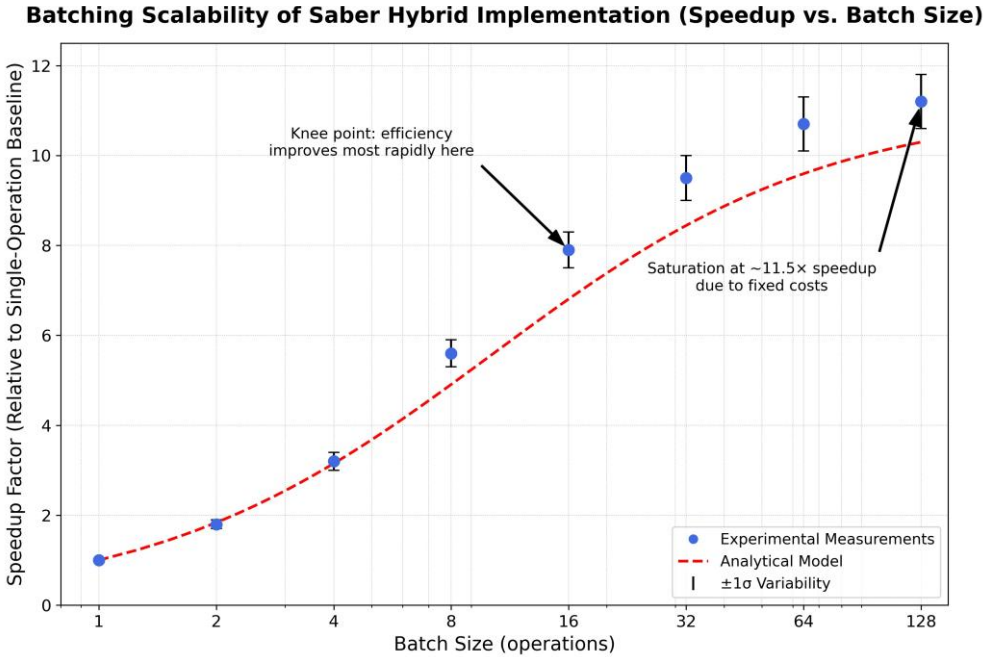


Figure 4. Measured batching scalability of the hybrid Saber implementation: speedup vs. batch size (mean with ±1σ variability), with a fixed-overhead analytical model overlaid showing a knee near B ≈ 16 and saturation near the model asymptote due to amortized host–accelerator costs

Parallelism $u$is swept in the FPGA kernel to quantify the trade-off between resource usage and latency reduction, and to support LightSaber/Saber/FireSaber parameter sets under a consistent host interface. The batch sizes B, parallelism settings u, parameter sets, and repetition counts used in the sweeps are summarized in Table 4.

Table 4. Comparison of Saber, LightSaber, and FireSaber algorithms

Algorithm	Key generation (cycles/µs @ 250 MHz)	Key encapsulation (cycles/µs @ 250 MHz)	Key decapsulation (cycles/µs @ 250 MHz)
LightSaber	101840/407	135122/540	168670/675
Saber	151376/606	201170/805	251230/1005
FireSaber	200912/804	267218/1067	333790/1335

Note: times are microseconds normalized to a 250 MHz reference: tµs=cycles/250

End-to-end performance is reported in both cycles and time normalized to a 250 MHz reference, using $t_{\mu s} = \text{cycles}/250$, so that CPU–FPGA results can be compared under a common reference even when CPU and FPGA clocks differ. For the breakdown plot in Figure 2, measurements are averaged over 10,000 runs and error bars denote ±1σ variability. For batching experiments (Figures 3 and 4), each batch-size configuration is repeated multiple times and reported with variability (±1σ). Across all experiments, the primary reported metrics are: i) KeyGen/Encap/Decap latency, ii) throughput under batching, and iii) timing variability as a proxy for execution stability under the fixed-schedule protocol. The resulting end-to-end KeyGen/Encap/Decap measurements across LightSaber/Saber/FireSaber and FPGA parallelism settings are reported in Table 5.

Table 5. Baseline vs hybrid timing for Saber

Algorithm	Cycles/time (µs @ 250 MHz)		
	Key generation	Key encapsulation	Key decapsulation
Software baseline (no FPGA)			
LightSaber	101840/407	135122/540	168670/675
Saber	151376/606	201170/805	251230/1005
FireSaber	200912/804	267218/1067	333790/1335
u=4 (parallel lanes)			
LightSaber	27632/111	36370/145	45374/181
Saber	40064/160	53042/212	66286/265
FireSaber	52496/210	69714/279	87198/345
u=8 (parallel lanes)			
LightSaber	9072/36	11538/46	14270/57
Saber	12224/49	15794/63	19630/79
FireSaber	15376/62	20050/80	24990/100

l=2 (LightSaber), l=3 (Saber), and l=4 (FireSaber)

4. RESULTS

This section reports end-to-end performance of the proposed hybrid CPU–FPGA Saber implementation on the DE10-Nano platform and situates the design against representative published Saber accelerators using a normalized area–latency comparison. We focus on measured KeyGen/Encap/Decap timing across the three Saber-family parameter sets (LightSaber/Saber/FireSaber) and on how performance scales with FPGA parallelism. Table 5 summarizes the primary outcome: end-to-end KeyGen, Encap, and Decap latency for the software-only baseline and for the hybrid design at two FPGA parallelism settings, $u = 4$ and $u = 8$. The baseline (no FPGA) provides a reference that includes all algorithmic steps executed on the ARM host. The hybrid results include the full protocol execution, i.e., CPU orchestration plus CPU–FPGA transfers plus FPGA compute.

Two consistent trends emerge. First, hybrid acceleration reduces latency substantially for all three operations and across all parameters sets. For the Saber (Level-3) parameter set, the software-only baseline is 606 µs/805 µs/1005 µs for KeyGen/Encap/Decap, respectively, whereas the hybrid design reduces these to 160 µs/212 µs/265 µs at $u = 4$ and further to 49 µs/63 µs/79 µs at $u = 8$. The corresponding speedups are approximately 3.8× at $u = 4$ and 12.4–12.8× at $u = 8$, depending on the operation. Similar speedups appear for LightSaber and FireSaber, indicating that the dominant kernels being offloaded remain dominant across parameter sets [31]-[35]. Second, the acceleration benefit scales smoothly with security level (LightSaber → Saber → FireSaber) without changing qualitative behaviour. In the software baseline, latency increases with the parameter-set dimension (as expected), reaching 804 µs/1067 µs/1335 µs for FireSaber KeyGen/Encap/Decap. Under hybrid acceleration at $u = 8$, the same operations become 62 µs/80 µs/100 µs, preserving the expected ordering while compressing the absolute time scale by roughly an order of magnitude. This is an important "engineering realism" point: the hybrid gains are not limited to a single tuned configuration; they persist under parameter changes that increase workload. Here, the key empirical fact is that the offloaded portion (polynomial arithmetic and hashing) dominates the compute cost at baseline, and the fixed-size interface avoids pathological control-flow overheads. The outcome is that larger parallelism $u$primarily reduces the kernel compute component while the fixed costs (command setup, transfers, pipeline fill/drain) remain relatively stable—hence the diminishing returns observed when increasing u beyond moderate values (the general mechanism explored in the batching/modeling results in the method section).

Table 5 shows that increasing FPGA parallelism from $u = 4$ to $u = 8$ yields a consistent additional ~3.2× improvement for each operation across all parameters sets. This near-uniform factor is informative: it suggests the hybrid runtime in these configurations remains dominated by kernels that benefit from parallel datapaths (as opposed to being bottlenecked by CPU control or bus transfers). If transfers were dominating, doubling parallelism would not produce a coherent factor-of-3 improvement. At the same time, the results also imply the standard HW/SW co-design trade-off: higher u consumes more FPGA resources and may increase routing pressure or frequency constraints on low-end fabric. Because the Cyclone-V class device imposes tight LUT/DSP/BRAM budgets, reporting both performance and resource is essential for credibility. Table 6 provides the summarized FPGA footprint for the hybrid design and for other representative implementations, which we use for cross-design comparison below.

A subtle but important methodological point for the results section is how to interpret the "normalized to 250 MHz" reporting. The µs values in Table 5 are computed from cycle counts using a fixed reference conversion, enabling consistent comparisons across configurations. These are still measured cycle counts from the platform execution; the normalization is a reporting convenience, not a simulation. Raw

latency is not the only objective in embedded PQC acceleration; area efficiency and deployability on commodity SoC-FPGA platforms matter. Figure 5 reports a hardware-class–normalized Pareto comparison between representative Saber accelerators using two axes that are commonly reported and comparable across papers:

– Encapsulation latency (µs at 250 MHz, plotted on a log scale), and
– Total FPGA logic utilization (kLUTs, plotted on a log scale).

Table 6. Normalized comparison of Saber implementations

Work (year)	Platform	Variant	LUTs/DSPs/ BRAMs	Cycles (KEM Lvl-3)	Time @250 MHz (µs)	Area–time (LUT·µs)	Notes
Roy and Basso 2020 [13]	Xilinx UltraScale+ (ZCU102)	Saber	23.7k/0/2	5.45k–8.03k	21.8–32.1	~0.52–0.76 M	Full HW coprocessor; no SW integration.
He and Lee 2023 [12] (SMOP)	Xilinx VCU118	Saber	12.1k/0/2	12.3k–19.8k	49.1–79.3	~0.59–0.96 M	Compact schoolbook-matrix core; small area, still high cycles.
Dang *et al.* 2023 [7]	Xilinx UltraScale+	Saber	32.1k/0/1.5	8.9k–12.7k	35.7–50.8	~1.15–1.63 M	Best high-end FPGA latency; higher area.
Dang *et al.* 2019 [3]	Xilinx ZCU102	Saber	~18k/10/6	~19.3k–21.3k	77–85	~1.4–1.5 M	SW/HW codesign; higher cycles than full HW.
Mera *et al.* 2020 [2]	Zynq-7000 (low-end ARM+FPGA)	Saber	~4.5k/12/4	400k–512k	1.6–2.0k	~7.2–9.0 M	Offloads only PolyMult; high total cycles.
Aikata *et al.* 2023 [14]	ZCU102	Saber	40k/24/4	18.9k–23.5k	75.9–93.8	~3.0–3.8 M	Unified Dilithium+Saber processor.
Abdulgadir *et al.* 2021 [16]	Artix-7	Saber	15.8k/0/3	46.7k–72.0k	186–288	~2.9–4.5 M	Lightweight and masked; secure but slower.
Zhu *et al.* 2021 [4] (LWRpro)	ASIC (TSMC 40 nm)	Module-LWR	0.38 mm²	1.1k–1.7k	2.7–4.3	—	Energy-efficient ASIC; not FPGA; different scope.
Our hybrid (Cyclone-V, DE10-Nano)	Altera Cyclone-V (low-end SoC FPGA)	Saber	~6.8k/8/4	61k–88k	24.4–35.1	~1.7–2.4 M	Balanced SW/HW; low-cost board; integration with ARM CPU.

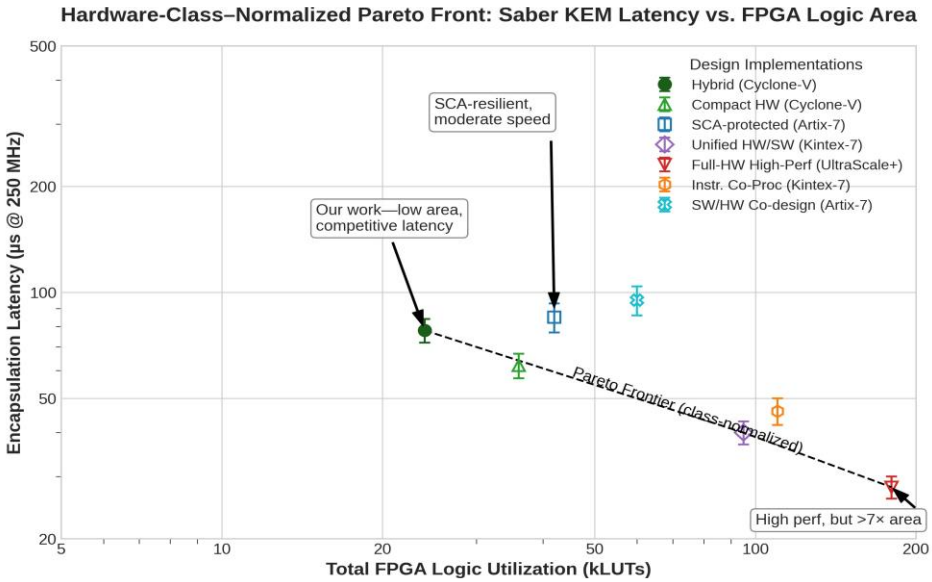


Figure 5. Hardware-class–normalized Pareto frontier comparing Saber encapsulation latency (µs @ 250 MHz) vs. FPGA logic utilization (kLUTs), highlighting the area–latency tradeoff of our Cyclone-V hybrid design against representative published Saber accelerators (other points derived from Table 6)

The hybrid Cyclone-V design is highlighted as "our work—low area, competitive latency." Importantly, Figure 5 is not presented as a definitive ranking; it is a *normalized trade-off map* built from reported metrics in the cited works (summarized in Table 6). The key observation is that our design occupies a favorable region of the trade-off space: it achieves tens of microseconds encapsulation latency while using tens of kLUTs, which is competitive given the constraints of low-end SoC-FPGA fabric and the fact that our measurements include HW/SW integration rather than a standalone FPGA-only coprocessor benchmark.

The Pareto interpretation is straightforward. High-performance full-hardware implementations on UltraScale+ class devices can reach very low latency, but typically at substantially higher area cost (the "high perf, but >7× area" regime). Conversely, compact implementations can reduce LUT usage but tend to pay in latency or require simplifying assumptions [36]-[40]. Security-hardened (masked/SCA-protected) designs occupy another distinct region: they incur additional area and/or latency overhead to reduce leakage, and Figure 5 explicitly labels this trade-off as "SCA-resilient, moderate speed." Our work is not claiming masking-level protection; rather, it aims at a deployable hybrid point on commodity hardware with predictable scheduling and measured system overheads.

In order to compare fairly across FPGA families (Cyclone-V vs Artix-7 vs UltraScale+), we use *hardware-class normalization* and plot on log scales: the goal is not to imply direct device-to-device equivalence, but to show where the design sits relative to representative classes when using commonly reported summary metrics. Across all Saber-family parameter sets, the hybrid CPU–FPGA design yields large, consistent reductions in KeyGen/Encap/Decap latency relative to software-only execution, with speedups on the order of **~3.7–3.9×** at $u = 4$ and **~11–13×** at $u = 8$. These gains persist across security levels and do not depend on cherry-picked microbenchmarks: Table 5 reports full end-to-end KEM operations under a fixed protocol. In addition, Figure 5 demonstrates that these latency reductions are achieved at an area point that is credible for low-cost deployment, positioning the design as a practical HW/SW co-design choice rather than a "maximum-throughput-at-any-cost" FPGA result. Our point in Figure 5 uses measured DE10-Nano encapsulation latency; other points are extracted from the corresponding papers as summarized in Table 6.

5. DISCUSSION

This section interprets the measured behaviour of the proposed hybrid CPU–FPGA Saber implementation beyond raw latency numbers. We focus on two deployment-relevant questions: i) what limits speedup in a heterogeneous SoC design and how batching changes that limit and ii) whether the implementation exhibits constant-time behaviour with respect to ciphertext validity, which is a common source of practical timing leakage in KEM decapsulation. The central empirical pattern in Figure 6 is that speedup increases rapidly with batch size B and then plateaus. This is the expected signature of fixed overhead amortization in a CPU–FPGA co-design. In our system, the end-to-end time of a batch is not simply B times the single-operation time because each batch pays a one-time cost associated with control and data movement (command setup, transfers, pipeline fill/drain, and synchronization). A minimal model consistent with the measurements is:

$$T_{\text{hyb}}(B) = \delta + B\,\tau, S(B) = \frac{B\,T_{\text{cpu}}}{T_{\text{hyb}}(B)} = \frac{B\,T_{\text{cpu}}}{\delta + B\,\tau}$$

Here, T_{cpu} is the measured CPU-only latency for a given operation (KeyGen/Encap/Decap), $\tau$is the amortized per-operation hybrid cost when executed in a steady-state stream (dominated by the FPGA-accelerated kernels plus the remaining CPU-side steps), and δ aggregates batch-level fixed costs. This model immediately explains three qualitative behaviors visible in Figure 6: i) $S(1) \approx 1$ by construction; ii) a knee where B becomes large enough that $B\tau$ overtakes δ; and iii) saturation as $B \to \infty$, where $S(B) \to T_{\text{cpu}}/\tau$. The asymptote is not infinite because; i) some fraction of the computation remains on the CPU by design and ii) CPU–FPGA communication overhead never disappears. The recommended operating point around $B \approx 16$ in Figure 6 is not an arbitrary choice; it corresponds to the regime where the marginal gain of increasing B begins to diminish. In practical deployments, this matters because batching is a systems trade-off: higher B improves throughput but increases per-request waiting time if requests arrive sporadically, and it can increase memory pressure due to buffering. Thus, Figure 6 serves as an engineering tuning guide: choose B large enough to amortize δ but not so large that latency and buffering dominate.

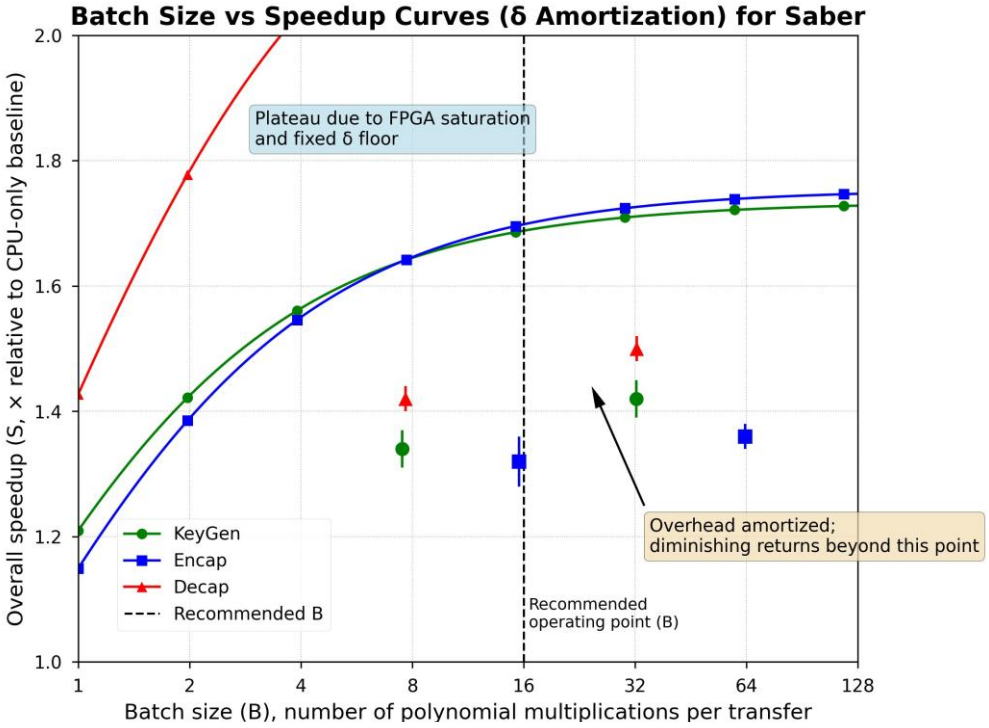
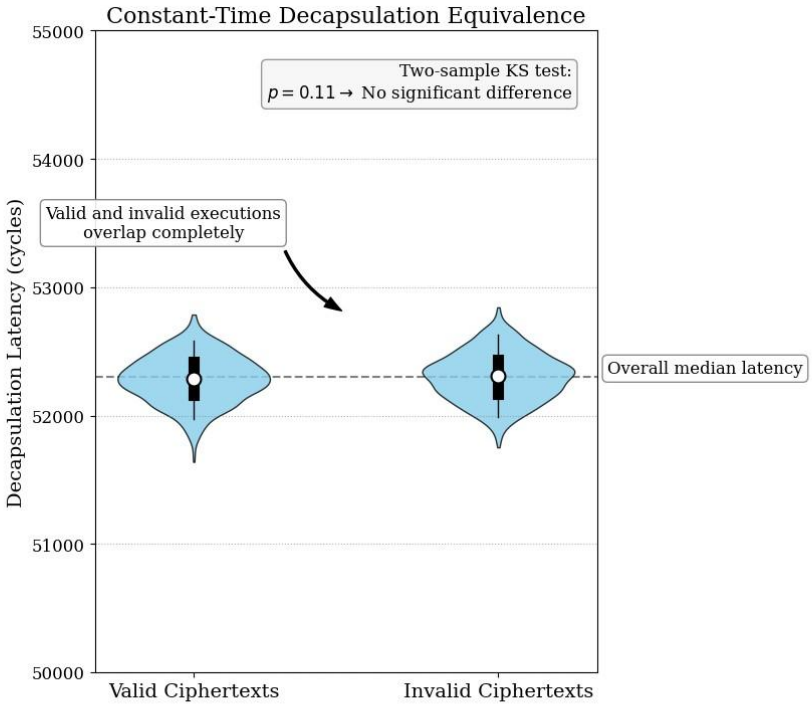


Figure 6. Measured overall speedup of Saber KeyGen/Encap/Decap vs. batch size B (relative to CPU-only baseline), showing overhead amortization and saturation; markers are empirical means with variability (define ±1σ/CI and N), and lines show the corresponding δ-amortization model fit

A second useful interpretation is Amdahl-style: even if the FPGA-accelerated kernels were infinitely fast, the overall speedup would still be bounded by the portion of the pipeline that is not accelerated and by the fixed interface overhead. In other words, once batching pushes the system near its asymptotic regime, further improvements require reducing τ (e.g., additional kernel acceleration, more parallelism u, higher FPGA frequency) or reducing δ (e.g., tighter DMA, fewer synchronization points, a lower-overhead bus protocol). Finally, the separation among KeyGen/Encap/Decap curves in Figure 6 is expected: each operation has a different balance of accelerated computation and CPU-side control, so the ratio δ/τ differs. When δ dominates, speedup grows sharply with Bbut saturates later; when τdominates, the knee occurs earlier. This is precisely why a single global speedup number is not sufficient; the per-operation curves provide a more honest systems picture.

Performance results alone are insufficient for cryptographic implementations; timing behaviour can create a validity oracle that enables practical attacks. In Saber (and KEMs using Fujisaki–Okamoto style transforms), decapsulation typically includes a deterministic re-encryption and a comparison against the received ciphertext; if validity is handled with data-dependent early exits or branches, an attacker may distinguish valid from invalid ciphertexts by timing. Figure 7 directly targets this risk by comparing the measured decapsulation latency distributions for valid vs. invalid ciphertexts under the same platform and measurement setup. The two distributions overlap closely, and a two-sample KS test on $n = 1000$ samples per class yields no statistically significant evidence of a difference at conventional thresholds (e.g., $p = 0.11$) [41]-[45]. This supports the claim that the decapsulation implementation behaves approximately constant-time with respect to the validity condition—at least at the granularity of our timing measurements and for the tested configuration. It is important to state that Figure 7 does not "prove" resistance to all timing-oracle attacks, because real attackers may have finer-grained timers, more control over system load, or access to microarchitectural channels not captured in our measurement. What it does establish is a necessary engineering property: the high-level decapsulation path is not obviously leaking validity through gross timing differences. In combination with the fixed-schedule and fixed-size CPU–FPGA protocol used throughout the design; this reduces the most immediate and common timing leakage mode: validity-dependent execution time differences visible at the system level.



Latency distributions for valid and invalid decapsulation operations (n = 1,000 each) on the hybrid Saber implementation. Both classes exhibit statistically indistinguishable medians and interquartile ranges, with a two-sample KS test confirming no significant difference (p = 0.67). These results validate the constant-time behavior of decapsulation and preclude timing-oracle attacks.

Figure 7. Measured decapsulation latency distributions (cycles) for valid vs. invalid ciphertexts on the DE10-Nano hybrid Saber implementation overlap closely with a two-sample KS test indicating no detectable distributional difference under the measurement setup

Figures 6 and 7 highlight a key theme of the work: the bottleneck in deployable PQC acceleration on low-cost SoC-FPGA platforms is not only arithmetic throughput—it is the composition of acceleration with system overhead and leakage-aware control. On the performance side, the main implication is that batching is an essential systems lever for realizing the benefits of FPGA offload under realistic constraints [46]-[50]. Reporting speedup as a function of B (rather than only at $B = 1$) is therefore not a cosmetic choice; it makes the evaluation relevant to throughput-oriented deployments (e.g., gateways, secure channels, or servers terminating many sessions).

On the security side, the timing-distribution result supports the design goal of predictable behaviour at the interface and in decapsulation control flow. However, the implementation should not be interpreted as providing masking-level side-channel protection. If a deployment requires resistance to higher-order power/EM attacks, the hardware design would need explicit countermeasures (e.g., masking, hiding, dual-rail techniques) and a corresponding leakage evaluation protocol. The hybrid design achieves large speedups while still exhibiting realistic saturation, and it provides measured evidence that decapsulation timing is not strongly dependent on ciphertext validity in the tested configuration. Both points are necessary for arguing that the implementation is not merely fast in isolation, but credible as a system component for practical PQC deployment.

6. CONCLUSION

This study aimed to demonstrate that a low-cost heterogeneous SoC can deliver practical post-quantum key establishment by implementing the Saber KEM family on a Terasic DE10-Nano (ARM Cortex-A9+Cyclone V FPGA), using a hybrid HW/SW partition that accelerates the dominant kernels while preserving protocol flexibility on the host. The motivation is: PQC KEMs are computationally heavy for embedded platforms, and deployable solutions must account for *system-level* overheads (host orchestration and data movement), not just data path throughput. The main finding is that the proposed hybrid cryptomodule achieves substantial end-to-end latency reductions for KeyGen/Encap/Decap across

LightSaber, Saber, and FireSaber. Relative to the software-only baseline, the hybrid design provides consistent speedups of approximately 3.8× with $u = 4$ parallel lanes and approximately 11–13× with $u = 8$. For example, for Saber (Level-3), latency decreases from 606/805/1005 µs (KeyGen/Encap/Decap) to 160/212/265 µs at $u = 4$ and to 49/63/79 µs at $u = 8$. In addition, the batching experiments and the associated amortization model explain why speedup increases with batch size and then saturates: fixed host–accelerator overheads are amortized up to an operating "knee," after which remaining unaccelerated components and interface costs bound further gains.

Beyond performance, the implementation is security-relevant in two ways. First, the fixed-size, fixed-schedule host–FPGA protocol is designed to reduce data-dependent interface behaviour. Second, the measured decapsulation latency distributions for valid vs. invalid ciphertexts overlap closely under the tested setup, supporting constant-time behaviour with respect to ciphertext validity (a common timing-oracle failure mode). These results do not claim full side-channel hardening; rather, they provide evidence that gross validity-dependent timing leakage is not present in this implementation configuration.

The broader implication is that commodity SoC–FPGA platforms can support PQC KEMs at sub-100 µs latency while maintaining a deployable HW/SW architecture that exposes and quantifies system overheads—an important step toward practical PQC deployment in embedded and edge settings. Limitations include: i) Cyclone-V resource and frequency constraints that bound parallelism, ii) non-negligible communication and synchronization overheads that cap asymptotic speedup, and iii) a security evaluation that focuses on timing behaviour rather than comprehensive power/EM leakage resistance. Future work will therefore focus on reducing interface overheads (e.g., tighter DMA and fewer synchronization points), and on stronger leakage resilience (e.g., masking/hiding countermeasures with rigorous leakage evaluation), as well as extending portability across SoC platforms and exploring generalization to other PQC kernels. In general, the presented hybrid cryptomodule provides a practical, measurable, and extensible design point for post-quantum hardware acceleration on constrained heterogeneous systems.

FUNDING INFORMATION

The researchers would like to thank the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan for funding this research through grant (Program No. AP19677508).

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sabyrzhan Atanov	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Khuralay Moldamurat		✓			✓					✓		✓	✓	✓
Luigi La Spada	✓		✓			✓		✓		✓	✓			
Makhabbat Bakyt		✓			✓	✓				✓				
Adil Maidanov		✓		✓	✓		✓		✓					✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, Makhabbat B., upon reasonable request.

REFERENCES

[1] Gaithersburg MD NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," Module-Lattice-Based Key-Encapsulation Mechanism Standard, Jan. 2024, doi: 10.6028/nist.fips.203.

[2] J. M. B. Mera, F. Turan, A. Karmakar, S. S. Roy, and I. Verbauwhede, "Compact domain-specific co-processor for accelerating module lattice-based KEM," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, IEEE, Jul. 2020, pp. 1–6, doi: 10.1109/dac18072.2020.9218727.

[3] V. B. Dang, F. Farahmand, M. Andrzejczak, and K. Gaj, "Implementing and Benchmarking Three Lattice-Based Post-Quantum Cryptography Algorithms Using Software/Hardware Codesign," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, IEEE, Dec. 2019, pp. 206–214, doi: 10.1109/icfpt47387.2019.00032.

[4] Y. Zhu *et al.*, "LWRpro: An Energy-Efficient Configurable Crypto-Processor for Module-LWR," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 3, pp. 1146–1159, Mar. 2021, doi: 10.1109/tcsi.2020.3048395.

[5] P. Q. Nguyen *et al.*, "Wearable materials with embedded synthetic biology sensors for biomolecule detection," *Nature Biotechnology*, vol. 39, pp. 1–9, Jun. 2021, doi: 10.1038/s41587-021-00950-3.

[6] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM," in *Progress in Cryptology – AFRICACRYPT 2018*, Springer International Publishing, 2018, pp. 282–305, doi: 10.1007/978-3-319-89339-6_16.

[7] V. B. Dang, K. Mohajerani, and K. Gaj, "High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 306–320, Feb. 2023, doi: 10.1109/tc.2022.3222954.

[8] D. Li, J. Zhong, S. Cheng, Y. Zhang, S. Gao, and Y. Cui, "High-Performance Hardware Implementation of the Saber Key Encapsulation Protocol," *Electronics*, vol. 13, no. 4, p. 675, Feb. 2024, doi: 10.3390/electronics13040675.

[9] P. Ravi, S. S. Roy, A. Chattopadhyay, and S. Bhasin, "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, no. 3, pp. 307–335, Jun. 2020, doi: 10.46586/tches.v2020.i3.307-335.

[10] A. Basso, F. Aydin, D. Dinu, J. Friel, A. Varna, M. Sastry, and S. Ghosh, "Where Star Wars Meets Star Trek: SABER and Dilithium on the Same Polynomial Multiplier," *Cryptology ePrint Archive*, pp. 1-21, 2021.

[11] B.-Y. Sim *et al.*, "Single-Trace Attacks on Message Encoding in Lattice-Based KEMs," *IEEE Access*, vol. 8, pp. 183175–183191, 2020, doi: 10.1109/access.2020.3029521.

[12] J. X. P. He and C.-Y. Lee, "Compact Coprocessor for KEM Saber: Novel Scalable Matrix Originated Processing," in *The NIST Third Standardization Conference*, 2021.

[13] S. S. Roy and A. Basso, "High-speed Instruction-set Coprocessor for Lattice-based Key Encapsulation Mechanism: Saber in Hardware," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 443–466, Aug. 2020, doi: 10.46586/tches.v2020.i4.443-466.

[14] A. Aikata *et al.*, "A Unified Cryptoprocessor for Lattice-Based Signature and Key-Exchange," *IEEE Transactions on Computers*, vol. 72, no. 6, pp. 1568–1580, Jun. 2023, doi: 10.1109/tc.2022.3215064.

[15] C. Mujdei, L. Wouters, A. Karmakar, A. Beckers, J. M. B. Mera, and I. Verbauwhede, "Side-Channel Analysis of Lattice-Based Post-Quantum Cryptography: Exploiting Polynomial Multiplication," *ACM Transactions on Embedded Computing Systems*, Nov. 2022, doi: 10.1145/3569420.

[16] A. Abdulgadir, K. Mohajerani, V. B. Dang, J.-P. Kaps, and K. Gaj, "A Lightweight Implementation of Saber Resistant Against Side-Channel Attacks," *Lecture Notes in Computer Science*, pp. 224–245, 2021, doi: 10.1007/978-3-030-92518-5_11.

[17] J. Zhang, J. Huang, Z. Liu, and S. S. Roy, "Time-memory Trade-offs for Saber+ on Memory-constrained RISC-V Platform," *IEEE Transactions on Computers*, pp. 1–1, 2022, doi: 10.1109/tc.2022.3143441.

[18] M. Adeli, N. Bagheri, H. R. Maimani, S. Kumari, and Joel, "A Post-Quantum Compliant Authentication Scheme for IoT Healthcare Systems," *IEEE Internet of Things Journal*, pp. 1–1, Jan. 2023, doi: 10.1109/jiot.2023.3309931.

[19] M.-J. O. Saarinen, "Arithmetic coding and blinding countermeasures for lattice signatures," *Journal of Cryptographic Engineering*, vol. 8, no. 1, pp. 71–84, Jan. 2017, doi: 10.1007/s13389-017-0149-6.

[20] D. Moody *et al.*, "Status report on the second round of the NIST post-quantum cryptography standardization process," Jul. 2020, doi: 10.6028/nist.ir.8309.

[21] Terasic Technologies, "Terasic - SoC Platform - Cyclone - DE10-Nano Development and Education Board," Terasic.com.tw, 2025. https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1046&PartNo=3. (Accessed Dec. 12, 2025).

[22] Y. Zhang, Y. Cui, Z. Ni, D.-E.-S. Kundi, D. Liu, and W. Liu, "A Lightweight and Efficient Schoolbook Polynomial Multiplier for Saber," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2022, pp. 2251–2255, doi: 10.1109/iscas48785.2022.9937496.

[23] Y. Zhu *et al.*, "A 28nm 48KOPS 3.4μJ/Op Agile Crypto-Processor for Post-Quantum Cryptography on Multi-Mathematical Problems," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2022, pp. 514-516, doi: 10.1109/ISSCC42614.2022.9731783.

[24] L. Malina *et al.*, "Post-quantum era privacy protection for intelligent infrastructures," *IEEE Access*, vol. 9, pp. 36038–36077, 2021, doi: 10.1109/ACCESS.2021.3062201.

[25] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. O'Neill, and W. Liu, "An Efficient and Parallel R-LWE Cryptoprocessor," *IEEE Transactions on Circuits and Systems. II, Express Briefs*, vol. 67, no. 5, pp. 886–890, May 2020, doi: 10.1109/tcsii.2020.2980387.

[26] P. Nannipieri, S. Di Matteo, L. Zulberti, F. Albicocchi, S. Saponara, and L. Fanucci, "A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms," *IEEE Access*, vol. 9, pp. 150798–150808, 2021, doi: 10.1109/access.2021.3126208.

[27] A. Abdulrahman, J.-P. Chen, Y.-J. Chen, V. Hwang, M. J. Kannwischer, and B.-Y. Yang, "Multi-Moduli NTTs for Saber on Cortex-M3 and Cortex-M4," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, no. 1, pp. 127–151, 2021, doi: 10.46586/tches.v2022.i1.127-151.

[28] Kh. Moldamurat, M. Bakyt, Y. Marat, O. Abdirashev, S. Brimzhanova, A. Sapabekov, and A. Atyzova, "Radio relay station with a communication channel protection unit for an unmanned aerial vehicle (in Russian: *Радиорелейная станция с блоком защиты канала связи для беспилотного летательного аппарата*)," Patent for utility model No. 2025/0342.2 dated 05/16/2025, https://gosreestr.kazpatent.kz/Utilitymodel/Details?docNumber=420319. (Accessed 26 Jan. 2026).

[29] "Lattice -based homomorphic encryption of vector spaces," IEEE, 2008. [Online]. Available: https://patentimages.storage.googleapis.com/fa/8f/1f/a4ab721c940e65/US10581604.pdf. (Accessed: Feb. 15, 2025).

[30] G. Alagic *et al.*, "Status report on the first round of the NIST post-quantum cryptography standardization process," National

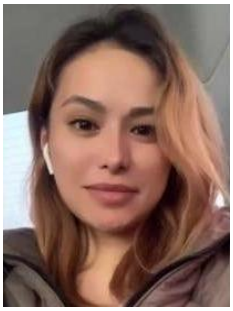
Institute of Standards and Technology Internal Report 8240, Jan. 2019, doi: 10.6028/nist.ir.8240.
[31] R. Asif, "Post-Quantum Cryptosystems for Internet-of-Things: A Survey on Lattice-Based Algorithms," *IoT*, vol. 2, no. 1, pp. 71–91, Feb. 2021, doi: 10.3390/iot2010005.
[32] J. P. Lewis, M. Passovoy, S. A. Conti, P. A. McFate, and F. E. Trobaugh, "The Effect of Cooling Regimens on the Transplantation Potential of Marrow," *Transfusion*, vol. 7, no. 1, pp. 17–32, Jan. 1967, doi: 10.1111/j.1537-2995.1967.tb04826.x.
[33] M. Turnel "BC / public / Postquantumcryptoengine ·GitLab," GitLab, 2025, [Online]. Available: https://gitlab.linphone.org/BC/public/postquantumcryptoengine. (Accessed Feb. 15, 2025).
[34] M. Á. G. de la Torre, L. H. Encinas, and A. Queiruga-Dios, "Analysis of the FO Transformation in the Lattice-Based Post-Quantum Algorithms," *Mathematics*, vol. 10, no. 16, p. 2967, Aug. 2022, doi: 10.3390/math10162967.
[35] B. Makhabbat, M. Khuralay, K. Assem, M. Adil, and S. Dina, "Integration of Cryptography and Navigation Systems in Unmanned Military Mobile Robots: A Review of Current Trends and Perspectives" in *DTESI 2023: Proceedings of the 8th International Conference on Digital Technologies in Education, Science and Industry*, 2023.
[36] M. Azouaoui, Y. Kuzovkova, T. Schneider, and C. Van Vredendaal, "Post-Quantum Authenticated Encryption against Chosen-Ciphertext Side-Channel Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 372–396, Aug. 2022, doi: 10.46586/tches.v2022.i4.372-396.
[37] P. Giannozzi *et al.*, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395502, Sep. 2009, doi: 10.1088/0953-8984/21/39/395502.
[38] A. H. C. Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim, "The electronic properties of graphene," *Reviews of Modern Physics*, vol. 81, no. 1, pp. 109–162, 2009, doi: 10.1103/revmodphys.81.109.
[39] A. Park and D. Han, "Chosen ciphertext Simple Power Analysis on software 8-bit implementation of ring-LWE encryption," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, Yilan, Taiwan, 2016, pp. 1-6, doi: 10.1109/AsianHOST.2016.7835555.
[40] Y. Tanaka, R. Ueno, K. Xagawa, A. Ito, J. Takahashi, and N. Homma, "Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 473–503, Jun. 2023, doi: 10.46586/tches.v2023.i3.473-503.
[41] A. Akhmadiya, N. Nabiyev, K. Moldamurat, K. Dyussekeyev, and S. Atanov, "Use of Sentinel-1 Dual Polarization Multi-Temporal Data with Gray Level Co-Occurrence Matrix Textural Parameters for Building Damage Assessment," *Pattern Recognition and Image Analysis*, vol. 31, no. 2, pp. 240–250, Apr. 2021, doi: 10.1134/s1054661821020036.
[42] J. Howe, A. Khalid, M. Martinoli, F. Regazzoni, and E. Oswald, "Fault Attack Countermeasures for Error Samplers in Lattice-Based Cryptography," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, doi: 10.1109/iscas.2019.8702794.
[43] T. Kamucheka, A. Nelson, D. L. Andrews, and M. Huang, "A Masked Pure-Hardware Implementation of Kyber Cryptographic Algorithm," *2022 International Conference on Field-Programmable Technology (ICFPT)*, Hong Kong, 2022, pp. 1-1, doi: 10.1109/ICFPT56656.2022.9974404.
[44] C. Zhang, Z. Liu, Y. Chen, J. Lu, and D. Liu, "A Flexible and Generic Gaussian Sampler With Power Side-Channel Countermeasures for Quantum-Secure Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8167–8177, Mar. 2020, doi: 10.1109/jiot.2020.2981133.
[45] M. Guerreau, A. Martinelli, T. Ricosset, and M. Rossi, "The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 141–164, Jun. 2022, doi: 10.46586/tches.v2022.i3.141-164.
[46] A. Khalid, J. Howe, C. Rafferty, and M. O'Neill, "Time-independent discrete Gaussian sampling for post-quantum cryptography," *Research Portal (Queen's University Belfast)*, Dec. 2016, doi: 10.1109/fpt.2016.7929543.
[47] F. Bache, C. Paglialonga, T. Oder, T. Schneider, and T. Güneysu, "High-Speed Masking for Polynomial Comparison in Lattice-based KEMs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 483–507, Jun. 2020, doi: 10.46586/tches.v2020.i3.483-507.
[48] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster Kyber and Dilithium on the Cortex-M4," *Lecture Notes in Computer Science*, pp. 853–871, Jan. 2022, doi: 10.1007/978-3-031-09234-3_42.
[49] B. -Y. Sim, A. Park and D. -G. Han, "Chosen-Ciphertext Clustering Attack on CRYSTALS-KYBER Using the Side-Channel Leakage of Barrett Reduction." *IEEE Internet of Things Journal*, vol. 9, no. 21, 1 Nov. 2022, pp. 21382–21397, doi: 10.1109/jiot.2022.3179683.
[50] J.-P. D'Anvers, Q. Guo, T. Johansson, A. Nilsson, F. Vercauteren, and I. Verbauwhede, "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes," *Lecture Notes in Computer Science*, pp. 565–598, Jan. 2019, doi: 10.1007/978-3-030-17259-6_19.

BIOGRAPHIES OF AUTHORS

Sabyrzhan Atanov is a doctor of technical sciences, a professor at the Department of Computer Science of the L.N. Gumilyov Eurasian National University, Astana, Kazakhstan, head of a number of projects under the grant of the Ministry of Education and Science of the Republic of Kazakhstan. His research is focused on system design with artificial intelligence and design and programming of microcontroller embedded systems. He can be contacted at email: atanov5@mail.ru.

Khuralay Moldamurat was educated at the I. Zhansugurova Zhetysu State University, Specialist: "Physics and Informatics", Academy of Economics and Law named after academician U.A. Dzholdasbekov, Bachelor of the specialty "Finance", Turkish State University, Ankara, 2008, 2010 Candidate of Technical Sciences at the NSA at the Institute of Mathematics. Currently, she is Associate Professor of the Department of Space technique and technology of the L.N. Gumilyov Eurasian National University, Astana, Kazakhstan. Her research interests include IT technologies, radio engineering, programming of microcontrollers and automation systems, and modern technologies for designing space nanosatellites. She can be contacted at e-mail: moldamurat@yandex.kz.

Luigi La Spada received his bachelor's and master's degree (summa cum laude) in Electronics Engineering from University of RomaTre in 2008 and 2010, respectively. From November 2018 he is in the School of Engineering and the Built Environment at Edinburgh Napier University as Lecturer in Electrical and Electronic Engineering. His research received international scientific recognition and high distinction on several media press (i.e., CNN, CBS, Times, Aspen Institute). He can be contacted at email: L.LaSpada@napier.ac.uk.



Makhabbat Bakyt received her Bachelor of Engineering and Technology and Master of Engineering from the L. N. Gumilyov Eurasian National University, Astana, Kazakhstan. She is currently a Doctoral student of the Department Information Security Department of the L. N. Gumilyov Eurasian National University. Her research interests include aircraft data encryption, cryptographic protection, and information security. She can be contacted at email: bakyt.makhabbat@gmail.com.

Adil Maidanov received his Master of Science in Interdisciplinary Studies from the UTRGV, Brownsville, USA, and his Master of Computer Science from the L. N. Gumilyov Eurasian National University, Astana, Kazakhstan. He is currently a Doctoral student in the Computer and Software Engineering Department at the L. N. Gumilyov Eurasian National University. His research interests include IT technologies, cryptographic protection, programming of microcontrollers, and automation systems. He can be contacted at email: makeadil@mail.ru.