IACR Communications in Cryptology
ISSN 3006-5496, Vol. 3, No. 2, 30 pages.

https://doi.org/10.62056/anxrdkmol

Check for updates

How to Benchmark Post-Quantum Accelerators

Bradley Morgan[1,2], Robert Dumitru[1], Rowan Fimmano[1], Valen Kostich[1], Ray Okamoto[1], Milan Sorak[1], Faisal Umar[1], Chitchanok Chuengsatiansup[3], Olaf Maennel[1], Paul Montague[2] and Yuval Yarom[4]

[1] Adelaide University, Australia
[2] Defence Science and Technology Group, Australia
[3] Hasso Plattner Institute and University of Potsdam, Germany
[4] Ruhr University Bochum, Germany

Abstract. With the recent standardisation of several post-quantum cryptographic (PQC) algorithms, there is an increased focus within the hardware research community to investigate efficient accelerator designs. However, it is difficult to determine whether the reported results are directly comparable due to differences in hardware prototyping platforms, integration contexts, and evaluation methodologies. This creates a gap between reported performance claims and their interpretability across works.
In this paper, we survey the reported evaluation metrics across 50 PQC accelerators. We identify a reporting asymmetry across the literature in the evaluations of cryptographic hardware, limiting fair comparison and obscuring system-level trade-offs. To resolve this, we design a homogeneous, FPGA-based benchmarking framework to appropriately evaluate cryptographic hardware. We replicate five recent accelerators as a case study in the same RISC-V system-on-chip environment to fairly compare their design approaches and compromises.

Keywords: post-quantum cryptography · hardware acceleration · RISC-V

1 Introduction

As post-quantum cryptography (PQC) transitions from standardisation into widespread deployment, implementers must make concrete engineering choices about which algorithms and implementations to adopt, and on what platforms [JMM+22]. These new algorithms defend against both classical and near-future quantum computer cryptanalysis [FRD+24]. The ongoing U.S. National Institute of Standards and Technology (NIST) PQC effort [Cen25, NIS25] and associated transition planning for industry adoption [MPR+24] create an impetus for integration of new public-key schemes into existing, and future protocols and products.

Implementation of PQC occurs in environments where constraints on latency, energy, and silicon area are of higher concern than previously. Despite significant progress in software-only implementations [KPR+24, SM16, KSSW22], leveraging processor features such as vectorisation and specialised instructions, these new schemes generally impose higher computational costs than classical public-key cryptography deployments [Atk21,

E-mail: bradley.morgan@adelaide.edu.au (Bradley Morgan), robert.dumitru@adelaide.edu.au (Robert Dumitru), rowan.fimmano@student.adelaide.edu.au (Rowan Fimmano), valen.kostich@student.adelaide.edu.au (Valen Kostich), ray@rayokamoto.com (Ray Okamoto), Milan.sorak@outlook.com (Milan Sorak), faisaumar115@gmail.com (Faisal Umar), chitchanok.chuengsatiansup@hpi.de (Chitchanok Chuengsatiansup), olaf.maennel@adelaide.edu.au (Olaf Maennel), paul.montague@defence.gov.au (Paul Montague), yuval.yarom@rub.de (Yuval Yarom)

Received: 2026-05-05 Accepted: 2026-06-30

PPPH24, LRJ24]. In response, the hardware community has proposed a growing body of cryptographic accelerators aimed at improving the runtime and energy costs of PQC operations, and reducing the system-level cost of quantum-secure key establishment and authentication [SBNK19, DTH+23, IAP20b, Des25, NDR+19]. Such accelerators may be exposed to software in several ways, for example as memory-mapped peripherals [DVMM24, YMÖS21], as coprocessors [BUC19, XHY+20], or via dedicated instruction-set extensions (ISEs) for the host processor [BNAMK21, LVC24].

Researchers compare accelerators to understand the state-of-the-art, identify design trade-offs, and demonstrate which approaches are best suited for their integration goals. To assist this, conferences and journals have introduced artefact evaluation into their submission workflows, enhancing visibility and reproducibility in the research [fCM20, Saa24]. In principle, such comparisons are driven by reported evaluation metrics such as (but not limited to) latency (cycles or wall-clock time), hardware resource cost, achievable frequency, energy consumption, and firmware/software footprints. Prototyping is commonly carried out using two fundamentally different platforms, field-programmable gate arrays (FPGAs) for their reconfigurability, and application-specific integrated circuits (ASICs) for real-world performance behaviour. There is no single standard for implementation of hardware logic, which is mapped to either platform with a synthesis tool of the designer's choice. Many are proprietary [Fah24].

Comparability across accelerators is not immediately possible across designs. It is not straightforward to compare FPGAs and ASICs, which contain a mixture of hardware building-block primitives that disproportionately affect their compared performance ratios [KR07]. Furthermore, two reported numbers may not be directly comparable. Works may target different hardware platforms or evaluate under different integration contexts and place asymmetric measurement boundaries around what is timed and what is counted. Moreover, authors write papers with varying goals, such that the evaluation of an accelerator is naturally shaped by what the authors consider relevant to demonstrate their contribution. These factors can make it difficult for a reader to determine when a comparison is well-founded versus when it rests on unstated assumptions from reported results alone.

To date, there is no shared, system-level methodology that defines measurement boundaries and metric definitions well enough to support consistent cross-work comparability. A notable gap remains in the cryptographic hardware literature, where a consolidated view of current evaluation practices is missing amongst diverse prototyping platforms, toolchains, and integration contexts. Even where platforms are recommended (e.g. NIST's suggested embedded/prototyping targets [Apo19]), there is no corresponding consensus on which metrics must be reported or how they should be measured. This leaves authors to make ad hoc evaluation and reporting choices that can unintentionally hinder later interpretation and cross-work comparison, forcing readers to infer comparability without a common frame of reference. This motivates a structured study of evaluation practices in PQC accelerator research. Hence, we pose the following questions:

***RQ*1.** Which evaluation metrics do PQC accelerator publications report to support their claims?

***RQ*2.** What does this reporting imply for the extent to which works can be compared, i.e. is there an evaluation reporting gap?

***RQ*3.** Which metrics are most important to constitute a fair, apples-to-apples comparison of cryptographic hardware?

1.1 Our Contribution

In this work, we systematise the diverse evaluation approaches adopted by authors for assessing their post-quantum hardware designs. Following this, we demonstrate how to

effectively conduct a comparability analysis across different accelerators using a targeted benchmarking approach. For that, we provide several main contributions.

We first conduct a evaluation metrics survey as reported by 50 FPGA and ASIC PQC accelerator works. The aim of our survey is to build a concrete inventory of how authors evaluate accelerators, focussing on the metrics they report rather than attempting to re-rank designs based on incompatible experimental setups. We then characterise the common metrics to demonstrate superiority over state-of-the-art into four *comparability classes*. Through this, we identify a systemic *asymmetric reporting* issue, an imbalance in reported metrics across many works. We find the majority of works report cycle and hardware resource metrics, meaning they can only compare as standalone hardware designs that is not necessarily apples-to-apples. This creates the aforementioned asymmetry, inherently limiting a fair evaluation and comparison of accelerators.

Second, we present a framework for benchmarking cryptographic accelerators, designed to evaluate PQC hardware within a common RISC-V system-on-chip (SoC) environment. This permits fairer comparisons between designs within the specific scope of our SoC's shared prototyping and execution context. Using this, we replicate five past works and benchmark them for ML-KEM [oST24b] and SLH-DSA [oST24c], effectively analysing and comparing their design trade-offs to demonstrate that closing the asymmetric reporting gap is indeed possible. We discover implementation drawbacks and functional correctness issues in the replicated works, highlighting the need for a homogeneous evaluation method in the community. Importantly, we do not aim to mandate our framework, but rather use it as an example of how fair evaluation can be performed.

Finally, our findings inform several key recommendations and guidelines for authors, reviewers and publication committees to observe with the goal of maintaining the comparability and reproducibility of future research. We propose a set of practical recommendations aimed at improving the consistency, transparency, and comparability of reported results through rigorous metric reporting, robust testing practices, and recommended open sourcing of code artefacts.

We summarise our contributions as follows.

- We survey the evaluation metrics reported across 50 PQC accelerator publications, highlighting where asymmetric reporting limits direct comparability between works. (Section 3)
- We analyse the impact of these reporting gaps through our classification of four comparability classes, discussing how this affects fair comparison across literature. (Section 4)
- We present our FPGA-based RISC-V benchmarking framework to enable reproducible and fair analysis of accelerators within the scope of our chosen SoC. (Section 5)
- We provide practical recommendations for authors and reviewers to improve the consistency, transparency, and comparability of accelerators. (Section 6)

Following the practices of open science, we make our source code and research artefacts publicly available at https://github.com/0xADE1A1DE/How-To-Benchmark-Post-Quantum-Accelerators.

2 Background

We now introduce the relevant background on the standardisation and transition towards PQC and its need for bespoke hardware acceleration.

2.1 Transition to Post-Quantum Cryptography

Post-quantum cryptography (PQC) refers to the set of public-key cryptographic algorithms designed to resist cryptanalysis by quantum and classical computers. Legacy public-key algorithms such as RSA [RSA78], and those in the elliptic curve family (e.g. ECDSA, ECDH)

are vulnerable to Shor's algorithm [Sho94], requiring the development of alternatives. The ongoing NIST PQC standardisation effort [Cen25] is underway, with its associated transition process [MPR+24] expected to deprecate and disallow legacy algorithms by the mid-2030s.

To date, NIST has selected five post-quantum algorithms for standardisation. These are ML-KEM [oST24b] and HQC [MAB+18] for key exchanges and ML-DSA [oST24a], SLH-DSA [oST24c] and FN-DSA [FHK+18] for digital signatures.

2.2 PQC Implementation on Hardware

One important aspect governing the selection of these algorithms for standardisation is their ease of implementation in embedded software [KPR+24, Atk21, PPPH24, LRJ24] and hardware accelerator contexts [SBNK19, DTH+23, PPPH24]. Observed performance decreases in terms of runtime and key size compared to legacy algorithms [OPowp19] necessitate a critical endeavour for the hardware research community to develop efficient software and hardware implementations for resource-constrained devices. Each of these algorithms commonly relies on the SHA-3 [oST15] family of hash functions for hashing and pseudorandom number generation, which takes up a considerable portion of their overall execution time [JO24, FVS19]. This provides incentive to the research community to accelerate SHA-3 and other inefficient subroutines with application-specific hardware.

Authors commonly utilise FPGAs, ASICs, or graphics cards to prototype post-quantum accelerators [BSNK19, IAP20a, DTH+23]. Hardware logic is written in a hardware description language (HDL) of choice, commonly Verilog or VHDL, which can then be synthesised into a gate-level netlist and subsequently implemented for the target platform. FPGA-based prototyping allows for flexible, rapid implementation and testing of a given design, albeit with limited run-time performance and a lower achievable clock frequency than custom integrated circuits. In comparison, ASIC synthesis gives a more accurate representation of the silicon performance in real-world contexts [KR07, AAE06], but requires a lengthier and more costly design cycle due to circuit printing and fabrication overhead. Graphics cards with their parallel architecture can accelerate certain algorithm subroutines such as polynomial multiplication in lattice-based cryptography, and SHA-3 hashing [LRJ24, NHNN25].

Evaluation Methodologies. To preface our survey on post-quantum hardware designs, it is important to define the common metrics authors use to characterise their work and facilitate their comparisons to state-of-the-art. Performance metrics common across both FPGA and ASIC designs include latency in cycles and/or time, and hardware resource utilisation in terms of FPGA resources or integrated circuit die area [SBN+21]. For FPGA-based designs, resources include lookup tables (LUTs), flip-flops (FFs), block RAM (BRAM), and digital signal processing (DSP) slices, which a synthesis and implementation tool maps to instantiate a given design. ASIC-based designs report area in terms of standard cells, gate equivalents (GE) or square millimetres (mm^2) for the target semiconductor technology node. The maximum achievable clock frequency is also reported by the synthesis tool alongside a worst negative slack (WNS) metric, with positive slack indicating successful timing closure and negative slack as a timing violation. Power consumption metrics are stated in terms of static (frequency independent), and dynamic (frequency dependent) switching activity per operation of an algorithm [BKG22]. Energy usage is then derived from power over execution time. These metrics for a design can be derived from either a simulated power analysis report, or more accurately using power measurement equipment connected directly to the hardware.

To permit fair benchmarking, NIST recommends the use of the ARM Cortex M4 microcontroller for embedded contexts, and the Xilinx Artix-7 FPGA for prototyping hardware designs [Apo19], however they do not specify concrete evaluation metrics. They do not recommend a specific technology node for ASIC designs.

Figure 1: Distribution of surveyed works presenting post-quantum hardware accelerators.

3 Asymmetric Reporting in PQC Accelerator Metrics

Works that propose new hardware implementations of cryptographic systems and claim improvements over the state-of-the-art must validate those claims, typically through experimental evaluation and comparison with prior work to demonstrate scientific merit and rigour. Such evaluation facilitates acceptance and potential adoption of the proposed technological improvements, as it concretely highlights the significance of the contribution and enables readers to assess advancements in the field. This involves authors typically evaluating their designs with a myriad of metrics to establish superiority over state-of-the-art in a given domain. Due to recent post-quantum standardisation efforts by academia and industry, both moving towards maturity and integration across a wider range of hardware platforms, it is crucial that the community can fairly evaluate and compare works. Ideally, this evaluation should occur in an equivalent manner to give a clear picture of trade-offs between designs.

In this section, we survey the evaluation metrics in PQC accelerator papers. We record and map the various metrics authors use to evaluate their designs and consequently compare with prior accelerators. Overall, our mapping reveals a stark asymmetry in evaluation methodologies, which greatly hinders fair comparison.

Survey Scope. Our survey focusses on 50 FPGA and ASIC PQC hardware implementations published since 2009, presented in Figure 1. Here, we classify observed metrics as either present (●) or partially observed (❍). Our objective now is to create a factual inventory of the metrics each paper reports, not to draw direct comparisons between works, given we do not distinguish between FPGA and ASIC designs, nor the absolute performance of each accelerator. We likewise recognise that some accelerators are designed for deployment scenarios where only a limited number of metrics are relevant, resulting in limited evaluation of the broader design trade-offs.

Given the abstract nature of describing hardware designs in academic works, we prefer works with open-source code, permitting more granular analysis of their contributions. The second preference is for publications in higher-tier venues to filter for more influential works. The third preference is for recently published works to keep the survey relevant with respect to current reporting methods.

Table 1: Reported metrics and comparability class membership for post-quantum hardware accelerators in our survey.

		Perf.			Hardware							Repr.		Sec.			Class			
Work	**Algorithm**	Cycles	Time	Throughput	HW Res.	A×T	F_{max}	Energy	Crit. Path	Mem O/H	Code Size	Version	Open Source	Const. Time	Side Ch.	Fault Inj.	HW Impl.	Sys. Int.	SW Fprint	Sec. Cons.
[EGHP09]	Classic McEliece	●	●	●	●		●		●	●		●		●	●		●	❍	❍	●
[GFS+12]	LWE			●	❍				●			●			●		❍	❍		●
[PG13]	Ring-LWE	●	●	❍	●		●		●				❍	●			●	❍		❍
[PDG14]	BLISS	●	●	●	●		●		❍				❍	●			●	❍		❍
[MBR15]	Classic McEliece	●	●		●	●	●			●			●	●			●		❍	❍
[KAMK16]	SIDH	●	●	●	●		●		❍					●			●	❍		❍
[WSN18]	Niederreiter	●	●		●	●	●						●	●			●			❍
[OG19]	NewHope-Simple	●	●		●		●		●	●			●	●			●	❍	❍	❍
[SBNK19]	qTESLA, ML-DSA	●			●		❍					●					❍			
[BSNK19]	Multiple	●	❍		❍	●	❍	❍	❍			●					❍	❍		
[BUC19]	Multiple	●			●	❍	❍	●	●			●	❍	●	●		❍	●		●
[KAK+19]	SIKE	●	●	●	●	●	●		●			●		●			●	❍		❍
[FSS20b]	LAC	●			●							●		●			❍			❍
[AEL+20]	ML-KEM, NewHope	●	●		●	●	●		❍	●	●		●				●	❍	●	
[ZYC+20]	NewHope-NIST	●	●	❍	●	●	●		●			●		●			●	❍		❍
[FSS20a]	Multiple	●			●		●	●	●		●	●	●	●			●	●	❍	❍
[XHY+20]	Multiple	●	●		●	●	●	●		●		●	●	●			●	❍	❍	❍
[SRB20]	Saber	●	●		●		●		●			●	●	●			●	❍		❍
[KRR+20]	Picnic, LowMC	●	●	●	●		●		●	●			●				●	❍	❍	
[ZZY+21]	Saber	●		●	●	●	●	●						●			●	❍		❍
[RMJ+21]	ML-DSA	●		●	●		●										●			
[BNAMK21]	ML-KEM	●	●		●	●	●		●			●		●			●	❍		❍
[TG21]	XMSS	●	●		●	●	●			●			●	●			●		❍	❍
[NDMZ+21]	ML-KEM, ML-DSA	●			●		❍	●	❍								❍	❍		
[YMÖS21]	ML-KEM	●			●		●						●	●			●			❍
[AMD+21]	Saber	●	●		●		●					●	●	●	●		●			●
[XL21]	ML-KEM	●	●		●		●			●		●	●	●			●		❍	❍
[BNG21]	ML-DSA, FN-DSA	●	●		●		●		●			●	●				●	❍		
[LSG22]	ML-DSA	●	●	●	●		●						●	●			●			❍
[HBTX22]	Saber	●	●		●	●	●	●	❍			●	❍	●			●	❍		❍
[CCD+22]	Classic McEliece	●	●		●	●	●		●	●		●	●	●			●	❍	❍	❍
[DMG23]	Multiple	●	●		●	●	●		❍			●		●			●	❍		❍
[FVBBR+21]	ML-KEM, Saber	●			●		●		❍		●		❍	●	●		●	❍	❍	●
[RBMG22]	BIKE	●	●		●	●	●					●	●	●			●			❍
[NKOL22]	SIKE	●	●		●	●	●		●			●	●	●			●	❍		❍
[RBCGG22]	BIKE	●	●		●	●	●		❍			●	●	●	●		●	❍		●
[ABC+23]	Saber	●			●		●					●	●		●		●			●
[DVMM24]	ML-KEM	●			●								●				❍			
[KSFS24]	ML-DSA, FN-DSA	●	●		●		●	●	●	●	●	●					●	●	●	
[GLM24]	ML-KEM	●		●	●			●	❍	●	●	●	●	●			●	❍	●	❍
[KA24]	FN-DSA	●		●	●		●					●			●		●			●
[LQYW24]	ML-KEM	●			●		●	●	❍			●	●				●	❍		
[LVC24]	SLH-DSA		●		●		❍						●				●			
[GOW+24]	SLH-DSA	●			●		●					●	●			●	●			●
[KRGF+24]	Classic McEliece		●		●		●			●		●	●				●		❍	
[THG24]	LMS, XMSS		●		❍	●	●			●			●			●	❍		❍	●
[OZZ+24]	FN-DSA	●	●	●	●	●	●	●	●			●	●				●	●		
[KAB+25]	CROSS	●	●		●	●	●					●		●			●			❍
[DLK+25]	SLH-DSA	●	●		●		●					●	●	●			●			❍
[AOHP+25]	ML-KEM, ML-DSA	●			●		●		●	●	●	●	●		●	●	●	❍	●	●
Total	**Present ●**	46	32	12	47	19	41	10	17	14	6	32	29	31	9	3	42	4	4	11
	Partial ❍		1	2	3	1	5	1	11				5				8	26	12	26

3.1 Methodology

We record the targeted PQC algorithms and provided evaluation metrics for each paper. For currently standardised algorithms, we use their new names for consistency, e.g. ML-KEM instead of CRYSTALS-Kyber. We group metrics by *design performance*, *resource efficiency*, *system reproducibility* and *security* aspects. In the case of partial or ambiguously reported metrics, we provide explanations below. Otherwise, we use a binary reported or not reported state. We now define each group of metrics in turn.

Design Performance. Running time is a fundamental metric for an accelerator as it expresses the performance of the design, critical for choosing an appropriate deployment scenario. This group includes direct latency measures such as *cycles* and *time*, alongside *throughput* as a complementary performance measure. It also includes timing closure metrics such as maximum frequency ($F_{\max}$) and critical path location in the design (*Crit. Path*). We mark partial results for ambiguous discussion of the critical path without explicitly stating its location, for example when described as being outside the accelerator. For maximum frequency, we mark results as partial when only a fixed frequency was tested rather than the highest achievable frequency. This is distinct from $F_{\max}$, obscuring peak performance and therefore limiting comparison between accelerators.

Hardware Resource Efficiency. Resource efficiency is a critical aspect of hardware design, as it influences the practical deployability and operational cost of cryptographic solutions. For this group we track the reported design costs in terms of generic hardware resource utilisation, e.g. LUT/FF/DSP/BRAM for FPGA, or cells/GE/mm^2 for ASICs (*HW Res.*) as well as area-time product efficiency ($A \times T$) and *energy*-related metrics. We also record whether the authors consider the accelerator's memory overhead from the point-of-view of system integration into an SoC either in terms of memory consumption or bus transfer limits (*Mem O/H*). We additionally include *code size*, as this reflects the implementation footprint on an embedded system. Partial results occur where hardware resources are not fully reported, (i.e. missing DSP or BRAM metrics) or only presented for submodules of the entire system.

System Reproducibility. Ensuring system reproducibility is paramount for validating research findings and building trust in cryptographic implementations. We include the listed specification *version* for the implemented algorithm(s) and whether the authors provide their *open-source* codebase. Partial reporting occurs when an open-source repository link is broken and unavailable on the Web Archive, or when the repository is incomplete, e.g. missing HDL files. We also mark as partial when the specific algorithm version cannot be determined due to ambiguous or conflicting references.

Security. Evaluation of cryptographic accelerators for their security and robustness against attacks pertinent to the hardware level is important for establishing and maintaining trust [HCS+21, GYCH18, Tea24]. We report whether authors discuss or evaluate their work for constant-time behaviour, side-channel and fault-injection resilience. For these, we record the presence or absence of any related discussion or experiments, hence the lack of partial results in the table.

3.2 Prevalence of Reported Metrics

Table 1 presents the mapping of reported metrics for our surveyed papers. Figure 2a visualises how frequently each individual metric is reported across the literature, and Figure 2b shows the spread of reported metrics per paper, split by whether a paper has any partially reported metrics. The majority of works include hardware resource usage alongside metrics relating to the latency or performance of the implementation. Conversely, we find that fault-injection, code-size, side-channel, and energy metrics are the least reported, each appearing in only 3–11 of the surveyed papers. Discussion of the critical path had the highest number of partial results (11) due to ambiguous descriptions in qualitative terms rather than explicit detail. In terms of coverage, authors report 7.5 ± 1.9 of our 15 recorded metrics on average, inclusive of partial, and 7.0 ± 2.0 excluding partial overall.

Discussion. Aside from reported metrics, we notice other circumstances that can obscure superiority over state-of-the-art. For example, if we were to compare a design implemented on a Xilinx Virtex-7 generation FPGA to one implemented on an Ultrascale+ generation, this could distort the hardware utilisation metric. The latter has more advanced features

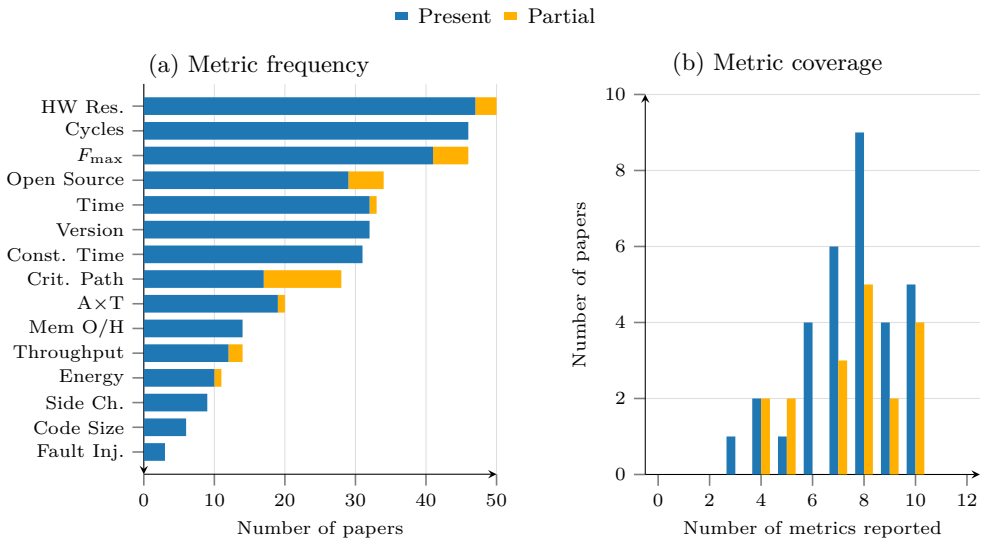

Figure 2: Metric reporting across the 50 surveyed papers.

and increased resources available that can positively influence the synthesis tool's ability to optimise the design during placement and routing of the logic blocks [MR98]. Similarly for ASICs, comparing designs fabricated (or simulated) using different technology nodes lacks uniformity in terms of power, area, and performance characteristics. Comparing across ASICs and FPGAs weakens the comparison further due to fundamental differences in their implementation and performance characteristics [KR07]. This highlights the importance of using a shared prototyping platform to compare hardware designs.

Even within a given implementation technology, fair comparisons become harder when accelerators are evaluated across CPU-centric deployments. Differences in host microarchitecture, cache hierarchy, and memory topology (e.g. interconnect and direct memory access (DMA) capabilities) influence the observed performance benefits of an accelerator. Furthermore, performance metrics are not always measured within the same context. Some works report accelerator-internal cycles, whereas others may include cycles measured by the host which would naturally include memory overhead and movement of data that are entirely host-dependent. Without an explicit measurement boundary, results can be misinterpreted.

Furthermore, we observe that security evaluations vary amongst works that discuss these properties. Side-channel and fault-injection analyses occurred rarely, ostensibly due to the need for advanced measurement equipment and the requisite expertise to evaluate hardware robustness against attacks in these domains. However, works that specifically target side-channel security, such as masked accelerator designs, typically include relevant experimental results. Constant-time behaviour was more commonly discussed qualitatively. We observed little widespread use of explicit formal verification [NOV+22, RBFSG22] or experimental validation (e.g. ChipWhisperer [OC14]) frameworks.

4 Impact on Apples-to-Apples Comparisons

Our survey identifies that metrics usually associated with integration into a common prototyping environment occur less often in the literature. We find many accelerators evaluated in isolation rather than within a more specific deployment scenario such as an embedded SoC or hardware root of trust. This means that even accelerators with

similar performance and resource results may remain difficult to compare for deployment scenarios. We also observe comparatively sparse security reporting, with side-channel and fault-injection metrics appearing minimally in our survey. It can be hard to compare works directly without introducing additional assumptions, which can lead to systemic inconsistency in the literature and obscure the state-of-the-art boundary. These choices may be reasonable on a per-case basis, as authors will prioritise the metrics most relevant to their work. Extra integration or analysis effort may be forgone when not central to the paper, but the resulting omissions still limit later comparisons.

We now discuss how the absence of certain groupings of metrics limits the extent to which papers in our survey can reasonably be compared. We define four groupings of metrics for accelerators that we refer to as *comparability classes*, and discuss the resulting impact of their asymmetric occurrence in the survey. This informs the approach of our benchmarking framework in Section 5 to increase comparability between works.

4.1 Accelerator Comparability Classes

We group the reported metrics identified in Section 3 into four separate comparability classes relating to the performance, functionality and security of the hardware. Each may exhibit multiple, or none of these classes, depending on what the authors decide to include in their evaluation. Naturally, each paper has its own intent as determined by the authors. Where the evaluation of an accelerator shares one or more of these classes, some comparisons can be made within the scope defined by the shared class. However, the strongest analyses become possible when an accelerator is evaluated using all four.

Class A: Hardware Implementation. The *hardware implementation* class pairs hardware resource utilisation with a primary performance metric to characterise intrinsic accelerator capability. Specifically, comparisons between hardware designs are nominally possible with one or more of cycles, wall-clock time, or throughput alongside a hardware cost-related metric specific to the prototyping platform (FPGA or ASIC). If only reporting cycles, the maximum frequency of the design must be present to normalise the performance.

Class B: System Integration. The *system integration* class captures hardware-level integration constraints posed by power and timing-closure considerations. This relates to the reporting of energy consumption and critical-path information. The critical path location can assist in determining whether an accelerator can feasibly be integrated within the designated target system. It indicates whether the accelerator can function at higher system clock frequencies without limiting overall system timing, or whether it would require crossing clock domains and its associated synchronisation mechanisms. Static and dynamic energy use must be likewise evaluated against the deployment's power budget.

Class C: Software Footprint. The *software footprint* class reflects the integration considerations at the software-level boundary through evaluation of code size and memory overhead metrics. This determines its practicality and benefit for certain scenarios with limited resources. Code size reflects the expected reduction in firmware footprint when functionality is offloaded from software into the accelerator. Memory overhead captures the additional buffering and data-movement (and associated RAM/ROM usage) cost for the target deployment. These metrics determine whether an accelerator can be practically supported by the software stack in resource-constrained environments.

Class D: Security Considerations. The *security considerations* class captures whether authors evaluate their implementation for robustness against attacks targeting cryptographic hardware. This includes discussion and evaluation of constant-time behaviour, together with side-channel analysis, fault-injection analysis, or both. Such metrics indicate whether an accelerator is compatible with deployment scenarios facing physical or local adversaries, as is common at the hardware level [HCS+21, GYCH18, Tea24].

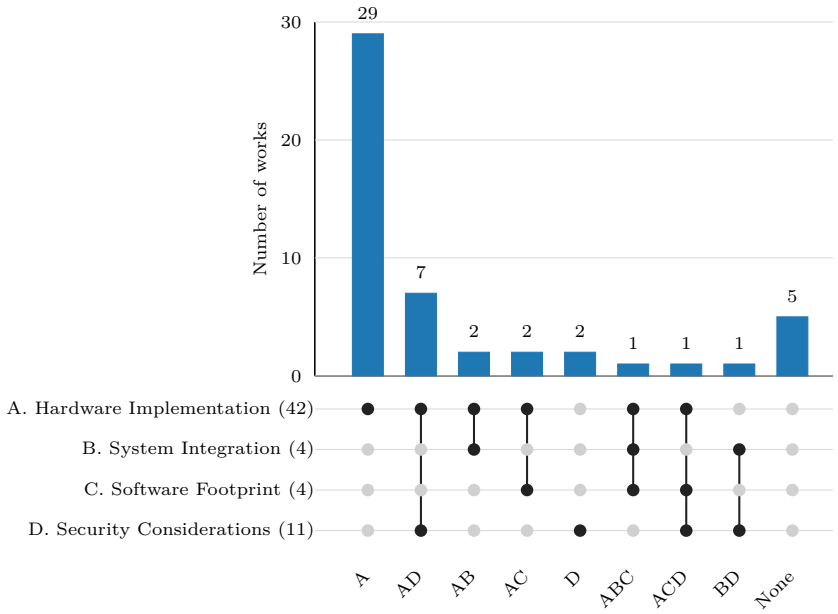

Figure 3: UpSet plot of comparability class coverage across the survey.

4.2 Implications for Comparability Studies

In Figure 3 we present an UpSet plot for the distribution of the comparability classes held by each paper in the survey, based on the counts in Table 1. The bars report the number of papers in each comparability-class combination. The filled dots in the matrix indicate which comparability classes are present in each combination, with connecting lines grouping classes which both occur in the same papers. Each class is defined independently by requiring fully reported metrics, and excludes partial reporting. The objective here is to evaluate how works could reasonably compare to each other.

No surveyed paper reports all four classes fully. The highest coverage is three classes, achieved by [KSFS24] (A, B, C) and [AOHP+25] (A, C, D). Otherwise, the majority (58%) only report hardware implementation statistics. Papers with two comparability classes made up 24% of the total, with 10% belonging to none. Even if all surveyed accelerators were evaluated on the same deployment platform, the absence of metrics in Classes B, C, and D would still prevent full comparison for 34 works.

Our classification highlights that the surveyed publications skew towards isolated hardware-centric evaluation, with comparatively little emphasis on system, software, or security-oriented metrics. Not every paper has the intent of integration within a larger system, however this nevertheless limits fair comparisons to a small subset of works. It is difficult to draw conclusions about relative accelerator quality without a common, realistic deployment scenario. This further motivates our need for a benchmarking framework that enforces more comprehensive and symmetric reporting of evaluation metrics.

5 PQC Accelerator Benchmarking Framework

Our findings in Section 3 show reporting of evaluation metrics to be asymmetric between works, making later comparisons between accelerators difficult. System-level integrations enable fairer apples-to-apples comparisons than single-number reporting of cycles, time, or derived metrics such as area–time product. When evaluated within the same system,

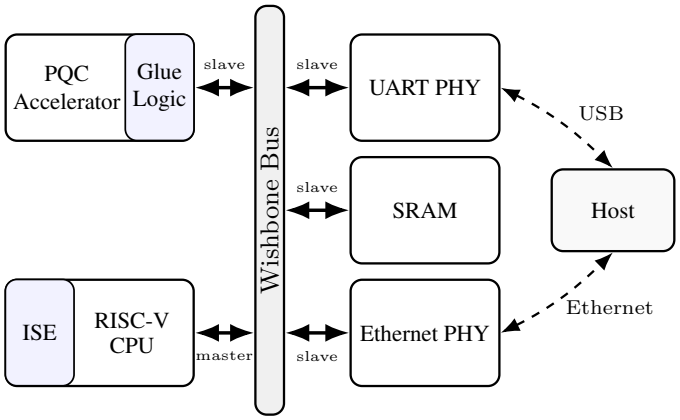

Figure 4: Structure of our RISC-V-based LiteX SoC with attached accelerators.

accelerators share an *execution context*, made up of the software stack, memory hierarchy, interconnect and clocking constraints, constraining the usual reported metrics.

In this section, we present our unified, FPGA-based benchmarking framework for PQC accelerators to enable such a shared execution context. We design our framework to allow testing of different accelerator types within the specific scope of a shared RISC-V-based benchmarking platform. This permits meaningful comparisons between these works to identify performance characteristics and trade-offs through explicit analysis of the comparability classes we introduced previously in Section 4.

In Section 5.2 we reimplement five recent (2024–2025), open-source hardware accelerators [LVC24, DVMM24, low24, Tea25b, Tea25a]. To demonstrate the utility of our approach, we target acceleration of the PQClean [KSSW22] ML-KEM and SLH-DSA reference implementations[1] as a case study for these works in Section 5.3. We focus on ML-KEM due to its prominence in the literature and standardisation as the primary key exchange algorithm by NIST. We additionally include SLH-DSA due to its NIST standardisation as a non-lattice-based digital signature scheme whose performance and memory demands can be challenging for embedded deployments.

We present a comparison of each accelerator's performance, hardware resource and power consumption relative to a baseline RISC-V software implementation. We find that all accelerators improve the performance of their compatible algorithms, with only the NTT and SHA-3 accelerators falling behind the ML-KEM optimisation curve of performance and resource consumption. Finally, we contribute several guidelines for authors and reviewers based on our findings in the survey and replication of accelerators to inform future work.

5.1 Scope and Design

To support our case study, the benchmarking process utilises an FPGA-based soft-core RISC-V SoC as our target platform due to its reconfigurability. We instantiate a stripped-down SoC using LiteX [KBLB20] on the Digilent Nexys Video Arty A7 FPGA board[2] for synthesis and implementation, the same FPGA die recommended by NIST [Apo19]. LiteX is a Python-based SoC framework built on top of the Migen HDL [Bou17], allowing for parameterised SoCs managed through the Python environment. It includes 29 RISC-V, ARM and OpenRISC cores plus individual configurations, with the ability to hot-swap

[1] At commit: https://github.com/PQClean/PQClean/tree/2cc64716044832eea747234ddbffc06746ab815d/

[2] https://digilent.com/shop/nexys-video-amd-artix-7-fpga-trainer-board-for-multimedia-applications/

between them using CPU to memory bus converters. LiteX supports both Wishbone and AXI-Lite memory buses, with accelerators integrated as MMIO peripherals. Figure 4 shows the architecture of our SoC, consisting of a user-specified RISC-V core, Wishbone memory bus, FPGA-based SRAM, UART/Ethernet interfaces for communication with the host PC, and room for add-on accelerators.

We use Vivado 2024.2 for synthesis of the FPGA designs, and Verilator 5.042 for cycle-accurate simulation. Our benchmarking system executes code at the bare-metal level, letting us directly interact with accelerators while avoiding OS-related noise in our measurements. LiteX uses PicoLibC[3] for C standard library access. With this setup, we can programmatically initialise SoCs with not only various accelerator peripherals but also different RISC-V cores, allowing for a wide range of testing scenarios. The host controls programming of the FPGA, as well as transfer of the benchmark program to execute on the instantiated SoC. The framework also supports generic targets to allow testing on various microcontrollers and x86-based machines.

For testing PQC algorithms, we tightly integrate the PQClean library into the compilation step to support benchmarking its reference C implementations. It also supports generic binary execution targets. Use of PQClean is not a strict requirement to initiate a test. To focus on measuring the performance of the cryptographic algorithms themselves, we do not include a true random number generator in our SoC. Instead, we generate random inputs for each benchmark using the `/dev/urandom` interface on the host, injecting this into the source code prior to compilation. We also support input of NIST known-answer test vectors (KATs) for functional correctness validation which bypasses the random input generation when selected.

The framework also supports simulation-based power measurement using Verilator to generate waveform traces, output in the switching activity interchange format (SAIF). We use this to estimate power consumption with the Xilinx power estimator tool[4] for different accelerators without requiring a complex physical measurement setup. The processor controls a control-status register (CSR) to start and stop the SAIF tracing around specific code regions, allowing for fine-grained power measurements.

The benchmarking system is available at: https://github.com/0xADE1A1DE/How-To-Benchmark-Post-Quantum-Accelerators.

5.1.1 Benchmarking Workflow

Our benchmarking system works in a three-step process to execute a test, as described in Figure 5. First, the user configures the SoC in terms of selected processor, integrated accelerators and size of the on-chip SRAM. Then they select the target to execute the binary on, either an FPGA device or simulation using Verilator.

With these configurations, the system compiles a benchmark program with a selected number of iterations, and patches the PQClean library to utilise the chosen accelerator instead of the reference C code. It then flashes the FPGA bitstream over JTAG to ensure a clean environment and starts a TFTP server for the LiteX BIOS to download the binary for immediate execution. Finally, the system deploys the binary to the target device via TFTP and initiates execution. The instantiated SoC then sends performance metrics back to the host PC via the Ethernet interface for reporting of test results.

In our case study for ML-KEM and SLH-DSA, we trace the individual timeframes for key generation, encapsulation, and decapsulation to get total power consumption over the full cryptographic scheme and compare this to a software-only baseline. This method does have drawbacks due to inherent limitations of simulation-based power estimation resulting in differences from the true observed power consumption [BKG22]. As we use the

[3] https://github.com/picolibc/picolibc

[4] https://www.amd.com/en/products/adaptive-socs-and-fpgas/technologies/power-efficiency/power-estimator.html

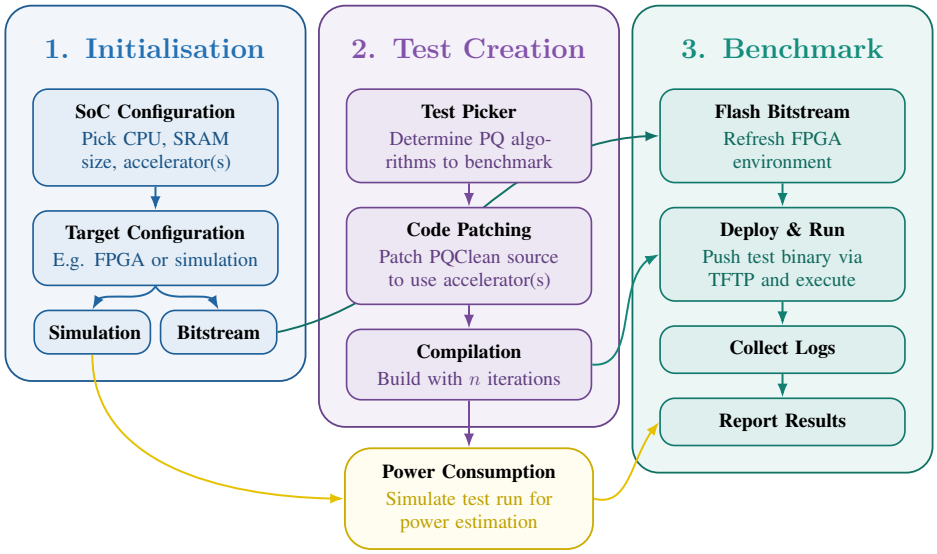

Figure 5: End-to-end flow of the benchmarking process.

same method across all tested accelerators, our aim is to only make relative comparisons between them.

5.2 Replication of Accelerators

We now detail the work required to replicate each chosen accelerator into our SoC for our comparability analysis. We select the works described in Table 2 to suit different accelerator contexts, ISE or memory-mapped. These accelerators are relevant to ML-KEM and, where applicable, SLH-DSA, either by implementing the complete scheme or by accelerating key primitives. For ML-KEM, this includes primitives such as the number-theoretic transform (NTT) [AB74], while hash-oriented accelerators targeting the `Keccak-f1600` permutation [BDPVA13] or SHA-3 hash family [oST15] are relevant to the hash-based primitives of ML-KEM and SLH-DSA.

For each, we adapt the published source code to interface with the Wishbone Bus of our LiteX SoC, adding glue logic to handle data transfers and control signals, and repurpose available C driver code to work with our setup. We also add an internal cycle counter within the glue logic to measure the exact number of cycles taken by just the accelerator, allowing us to isolate its performance from memory bus overhead. We simplify the interface for each accelerator to use sets of control or data registers in the glue logic, interacting with these via pointers to memory addresses. During periods of accelerator operation, the processor busy-waits for completion.

Lopez-Valdivieso and Cumplido [LVC24] present a tightly coupled ISE for the in-order 32-bit RISC-V RVX CPU.[5] They implement a single round of the `Keccak-f1600` permutation as a hardware instruction to accelerate SLH-DSA. Executing this instruction 24 times executes the full permutation. We update their implementation to the latest RVX version 3.1, with the unmodified CPU as the basis of our SoC. We modify the instruction decoder to include the new instruction opcode, and connect the existing `Keccak-f1600` module to the current version's instruction pipeline as in the original work.

Dolmeta et al. [DVMM24] propose an NTT accelerator specific to ML-KEM, integrating this with the X-HEEP RISC-V core with its customisable hardware extension interface [SMPQ+23], hence its moderate coupling. Two internal parameters required

[5] https://github.com/rafaelcalcada/rvx

tweaking to increase the time spent doing NTT and inverse NTT (INTT) by ten and eighteen cycles respectively to output the correct results as the original published test cases. The glue logic for this interface includes two separate finite state machines for inbound and outbound data between the Wishbone Bus and accelerator, which themselves handle control and status signals for the NTT operation during state transitions. For straightforward compatibility with the original design's use of X-HEEP's extension interface (which uses DMA), our glue logic instantiates an intermediate buffer to hold the polynomial data for the processor to read from.

LowRISC [low24] provide an exemplar `Keccak-f1600` permutation module for the OpenTitan project, which we adapt as a loosely coupled accelerator for comparison to Lopez-Valdivieso and Cumplido [LVC24]. Our glue logic hosts the full 1600-bit `Keccak-f1600` internal state for access by the Wishbone Bus, directly plumbed into the module, as well as enable and status registers to control and monitor the operation. This includes an internal round counter loop to indicate to the module which round of the permutation to execute in sequence while the CPU busy waits for completion.

Camacho-Ruiz et al. [Tea25b] implement a SHA-3 hash family accelerator as part of their post-quantum secure element (SE) for the QUBIP Project [CRNTMRB25]. They also contribute an accelerator for the full ML-KEM system [Tea25a]. Although the SE is designed to use either AXI-Lite or I2C protocols, the internal acceleration modules of interest use a custom register-based control and data interface that we adapt for our SoC instead. The glue logic for both accelerators uses direct plumbing to connect the Wishbone Bus reads and writes to the accelerators' register interfaces for controlling their respective operations.

Table 2: Chosen accelerators for replication with their original implementation scope.

Work	Function	Coupling	Scope	Note
[LVC24]	`Keccak-f1600`	ISE	Primitive	Single round
[DVMM24]	NTT/INTT	DMA	Primitive	ML-KEM only
[low24]	`Keccak-f1600`	Memory-mapped	Primitive	All 24 rounds
[Tea25b]	SHA-3	Memory-mapped	Primitive	Entire hash family
[Tea25a]	ML-KEM	Memory-mapped	Full System	ML-KEM only

5.3 Accelerator Benchmarking

To accurately evaluate and compare each of the accelerators integrated into our SoC, we first establish a reference software-only baseline for ML-KEM and SLH-DSA. From there, we benchmark each accelerated implementation with the same process to permit meaningful comparisons.

SoC Configuration. We use the unmodified version of the RVX CPU for the reference and memory-mapped accelerator benchmarks, given our interest in benchmarking an ISE for the same processor [LVC24]. We configure the SoC with 48 KB of ROM for BIOS storage and 256 KB of SRAM for benchmark program space. To determine $F_{\max}$, we synthesise the design three times at a given frequency in Vivado to determine if we achieve a stable SoC implementation. If this fails, we decrement the target frequency to the nearest lower frequency supported by LiteX's SoC clock manager until we find a configuration that meets timing and successfully runs our benchmarks. We use default synthesis and implementation settings in Vivado for all designs to ensure consistency.

Methodology. For each of the reference and accelerated implementations, we follow the same benchmarking procedure to ensure consistency as discussed in Figure 5. We compile and execute a running time benchmark with the FPGA implementation for each algorithm's security levels, measuring the cycle count for each cryptographic operation with a ten-iteration warm-up phase. The benchmark runs 1,000 measured iterations with

the random seed generated on the host, reporting whether the shared secret outputs match, or signature verification succeeds. Then, we run the benchmark with SAIF tracing enabled on the simulated design for power estimation of each operation. To determine an accurate increase or decrease in code size for the PQClean implementation, we use GCC flags `-ffunction-sections` and `-fdata-sections` to allow the linker to remove unused functions and data. We additionally functionally verify each FPGA implementation against the latest version of the NIST cryptographic validation test system[6] for ML-KEM to determine correctness. PQClean does not currently implement a FIPS 205 compliant version of SLH-DSA.

Table 3: FPGA implementation results for replicated ML-KEM and SLH-DSA accelerators on the Digilent Nexys Video.

Work	Function	SoC Coupling	Freq. (MHz)	FPGA Resource LUT	FF	BRAM	DSP
				2 959	2 655	88	0
Ref.	–	–	48.85	0	0	0	0
				0	0	0	0
				6 875	3 689	88	0
[LVC24][1]	`Keccak-f1600`	ISE	48.85	0	0	0	0
				3 916	1 034	0	0
				5 219	7 747	91	10
[DVMM24]	NTT	Memory-mapped	36.92	1 565	4 363	0	0
				695	729	3	10
				6 724	4 438	88	0
[low24]	`Keccak-f1600`	Memory-mapped	48.85	1 594	1 783	0	0
				2 171	0	0	0
				8 113	6 170	88	0
[Tea25b]	SHA-3	Memory-mapped	48.85	202	255	0	0
				4 952	3 260	0	0
				13 435	7 689	124	15
[Tea25a]	ML-KEM	Memory-mapped	44.94	293	243	0	0
				10 183	4 791	36	15

[1] ISE resources included in the total SoC resources.
Resource row triplets report total SoC, glue logic, and accelerator resources, respectively.

5.4 Comparability Analysis

Facilitating comparability in Class A (Hardware Implementation), Table 3 presents the FPGA implementation results of the replicated accelerators in our framework. Listed beneath each SoC's FPGA resource consumption is the additional overhead introduced by the glue logic and accelerator where applicable. [Tea25a] uses the most resources due to its highly targeted approach towards ML-KEM, implementing the entire algorithm in hardware. The [DVMM24] and [low24] designs require increased logic in comparison to the others to handle the entire NTT and Keccak intermediate state respectively. This could be replaced with a streaming interface at the cost of increased cycle duration.

Table 4 details each accelerator's correctness, performance, and patched code size to execute ML-KEM and SLH-DSA. All accelerators improve the execution time of both ML-KEM, and where relevant, SLH-DSA, at the cost of significantly more FPGA resources. Figure 6a shows a comparison of the performance in seconds against combined LUT and FF utilisation for each accelerator across the three ML-KEM security levels with pareto fronts. Each presents a different trade-off between performance and resource utilisation,

[6] https://github.com/usnistgov/ACVP-Server v1.1.0.41

however [DVMM24] and [Tea25b] (SHA-3) fall short of the ML-KEM front consistently across all security levels. The full-system accelerator [Tea25a] achieves a 97% reduction in running time across all security levels, scaled by frequency. In contrast, [DVMM24] shows a scaled running time approximately ranging from a 1% increase to a 19% reduction across ML-KEM primitives and security levels due to its lower operating frequency. We note that the original accelerator employs DMA, whereas our integration does not. This could be handled using multiple clocks, however this would result in further integration costs in terms of complexity and resource overhead.

We observe a similar hierarchy of results to ML-KEM with [LVC24], [low24] and [Tea25b] accelerators for SLH-DSA. Here, we see in Figure 6b that [LVC24] sits at the forefront of A×T performance due to its tightly coupled design for optimising Keccak. It achieves a 89% reduction in running time across all security levels at the same frequency as the reference SoC. [Tea25b] is the slowest, yet still achieves on average 81% reduction in running time, albeit with a higher resource cost due to implementing the SHA-3 family.

Table 4: Performance and code size results for the replicated ML-KEM and SLH-DSA accelerators. Each row triplet corresponds to increasing algorithm security levels.

	ML-KEM-512/768/1024				SLH-DSA-SHAKE-128f/192f/256f			
Work	**KeyGen**	**Encap**	**Decap**	**Code**	**KeyGen**	**Sign**	**Verify**	**Code**
Ref.	5 867 ±21	8 032 ±24	9 932 ±26	80.84	482 962 ±1	11 308 921 ±0	669 305 ±15 483	110.73
	9 673 ±32	12 902 ±32	15 456 ±36	80.31	708 132 ±1	18 327 966 ±0	980 636 ±15 390	147.49
	14 973 ±39	19 063 ±41	22 287 ±45	82.02	1 875 264 ±1	37 752 374 ±0	999 624 ±16 642	176.84
[LVC24][1]	3 289 ± 14	5 548 ± 16	7 447 ± 21	73.50	51 028 ± 19	1 202 214 ± 0	68 812 ± 1 824	103.02
	5 566 ± 19	8 698 ± 21	11 252 ± 26	72.98	77 041 ± 1	2 025 216 ± 0	105 207 ± 1 652	140.26
	8 379 ± 22	12 373 ± 24	15 597 ± 29	74.70	211 530 ± 1	4 339 000 ± 0	111 071 ± 1 950	169.61
[DVMM24]	4 066 ±19	5 470 ±19	6 091 ±20	83.27	–	–	–	–
	7 008 ±28	9 356 ±27	10 267 ±28	84.35	–	–	–	–
	11 421 ±38	14 524 ±38	15 743 ±39	87.64	–	–	–	–
[low24]	3 315 ± 14	5 578 ± 16	7 477 ± 20	73.98	55 518 ± 1	1 306 341 ± 0	75 290 ± 1 717	103.47
	5 608 ± 18	8 747 ± 20	11 300 ± 25	73.23	83 812 ± 1	2 199 344 ± 0	114 155 ± 1 711	140.15
	8 443 ± 22	12 451 ± 23	15 675 ± 28	75.09	228 920 ± 1	4 687 992 ± 0	120 308 ± 2 072	169.37
[Tea25b]	4 521 ±23	6 792 ±24	9 223 ±27	82.50	87 681 ±1	2 043 601 ±0	123 322 ±2 800	111.37
	8 273 ±34	11 418 ±34	14 771 ±37	81.73	132 680 ±1	3 403 937 ±0	186 030 ±2 916	148.09
	13 132 ±43	17 141 ±44	21 432 ±46	83.45	361 647 ±1	7 223 904 ±0	194 856 ±3 393	177.62
[Tea25a]	232 ± 0	203 ± 0*	247 ± 0*	60.03	–	–	–	–
	309 ± 0	282 ± 0*	359 ± 0*	60.09	–	–	–	–
	404 ± 0	378 ± 0*	489 ± 0*	60.11	–	–	–	–

Operation timings are kilocycles ± σ, $n = 1000$. Code is reported in KB.
[1] ∗ marks NIST ACVP non-compliance for the cryptographic operation.

5.4.1 Critical Path Analysis

Table 5 forms part of our Class B (System Integration) analysis, highlighting the difference between system-level and accelerator timing slack and critical path behaviour across the replicated works. In several cases, the integrated critical path is dominated by generic CPU logic despite the presence of one of the accelerators. This masks the true timing characteristics of the accelerator itself, visible only when we constrain Vivado's timing analysis to the accelerator hierarchy. Accelerator bottlenecks may not limit system frequency in our instance, but they do still have a limit that may apply in other scenarios.

Rather than comparing ($F_{\max}$), dependent on a specific Vivado implementation run, we compare the target frequency specified during LiteX SoC generation. The NTT design has the lowest of all tested accelerators at 36.92 MHz, with the butterfly unit being the critical path of the design, handling modular arithmetic on the polynomial coefficients. This constitutes a 24.4% reduction in frequency compared to the reference of 48.85 MHz.

The full-system ML-KEM accelerator [Tea25a] has the second lowest frequency at 44.94 MHz, an 8% reduction from the reference, with the critical path being within its Keccak module. Given the rest of the designs do not influence the critical path of the

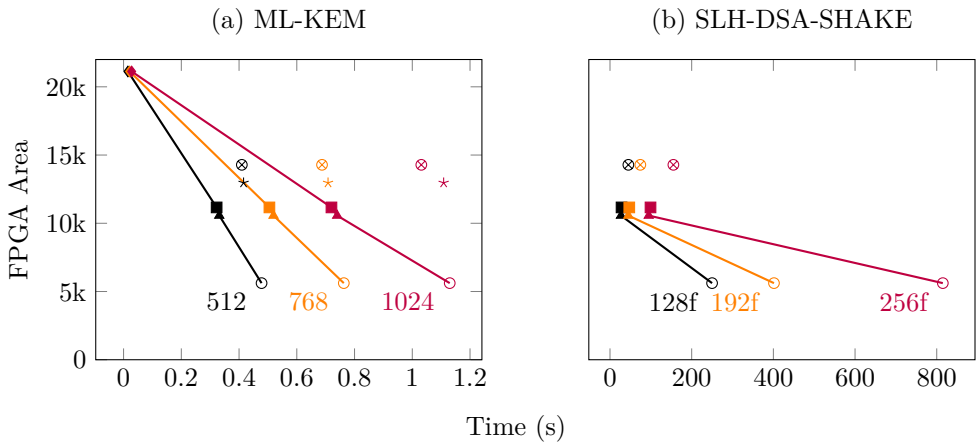

Figure 6: A×T Pareto fronts across each compared SoC. Ref.=○, [LVC24]=▲, [DVMM24]=⋆, [low24]=■, [Tea25b]=⊗, [Tea25a]=⧫.

RVX, multi-domain clocking could be used to further increase overall performance at the cost of glue logic complexity.

Table 5: Critical path information for each accelerator integration.

Work	Integrated Crit. Path	WNS (ns)	Accel. Crit. Path	WNS (ns)
[LVC24]	RVX Load/Store Logic	0.11	Memory	0.30
[DVMM24]	NTT Butterfly Module	0.44	NTT Butterfly Module	0.44
[low24]	RVX Load/Store Logic	0.78	Keccak Module	18.19
[Tea25b]	RVX ALU Data Path	0.52	Keccak Submodule	4.37
[Tea25a]	Keccak Submodule	0.18	Keccak Submodule	0.18

5.4.2 Power and Energy Consumption

Furthering our Class B analysis, we report the energy and power consumption in terms of static and dynamic switching activity [BKG22]. Each SoC has very similar static power use, as shown in Table 6. The slight increase observed in [Tea25a] is attributed clock gating of enabled BRAMs [Sub17]. Dynamic power varies across the SoCs more substantially, with the full-system ML-KEM accelerator [Tea25a] contributing a greater proportion of dynamic power with its larger FPGA area in comparison to the others. Despite this, we find the energy expended per key exchange significantly lower than the other accelerators due to its heavily reduced execution time.

SLH-DSA consumes significantly more energy across the SoCs than ML-KEM due to its expected longer runtime, despite the use of the algorithm's fast parameter set. This demonstrates the comparative benefits of optimising the internal SHAKE hashing operations, with the accelerators reducing energy consumption up to 9× compared to reference.

5.4.3 Code Size

Our initial Class C (Software Footprint) analysis involves determining any code size reduction benefits of each accelerator. Given the smaller size of the driver code in comparison to the PQClean library, [Tea25a] achieves a 26% reduction on average for each ML-KEM security level compared to the reference software implementation. [DVMM24]

Table 6: Per-SoC power and energy estimation for ML-KEM and SLH-DSA.

Work	Static	ML-KEM-512				SLH-DSA-SHAKE-128f			
		Dynamic (mW)			Energy	Dynamic (mW)			Energy
	(mW)	SoC[1]	RVX	Acc.	(J)	SoC[1]	RVX	Acc.	(J)
Ref.	141	367	13	–	0.25	368	12	–	129.98
[LVC24]	141	376	21	1	0.17	375	20	1	13.98
[DVMM24]	141	385	9	3	0.22	–	–	–	–
[low24]	141	369	12	0	0.17	369	12	0	15.02
[Tea25b]	141	352	2	4	0.21	371	12	5	23.66
[Tea25a]	143	410	8	39	0.01	–	–	–	–

[1] Inclusive of CPU and accelerator dynamic power consumption.

reports an increase due to overwriting the NTT and INTT source functions. The additional source code introduced by the accelerator driver code offsets larger gains.

Replacing the `Keccak-f1600` software function, the [LVC24] and [low24] accelerators shrink the firmware by approximately 9% and 5% for ML-KEM and SHAKE-based SLH-DSA respectively. In contrast, for the SHA-3 hash module [Tea25b], the driver code increases the overall code size for both algorithms, as they rely on specific functions from PQClean's FIPS 202 [oST15] implementation that remain implemented in software.

5.4.4 Memory Bus Overhead

For Class C, we present the memory buffering and data-movement costs for deployment in Table 7. We report overhead as the proportion of total execution time that is spent by the RVX transferring data. We find that the memory-mapped accelerators suffer from significant communication overhead. This highlights the performance benefits of tight coupling and suggests that future optimisation should target general memory-access patterns rather than cryptography-specific improvements. The least affected design is [Tea25a], which only has to transfer keys and shared secret to and from the accelerator, instead of repeatedly transferring data for each primitive operation.

Table 7: Memory bus overheads for each accelerator function in cycles.

Work	Function	RVX	Accelerator	Overhead (%)
[LVC24]	`Keccak-f1600`	828	24	97.18
[DVMM24]	NTT	6 642	923	87.80
	INTT	8 110	931	89.70
[low24]	`Keccak-f1600`	1 857	24	98.72
[Tea25b]	SHA3-256	21 210	25	99.88
	SHA3-512	17 910	25	99.86
	SHAKE128	23 803	25	99.90
	SHAKE256	21 635	25	99.88
[Tea25a]	KeyGen (512)	134 868	98 169	57.87
	Encap (512)	77 966	126 127	38.20
	Decap (512)	51 113	196 237	20.66

5.4.5 Functional Correctness

We test each implementation with the NIST KAT system [NIS24] for ML-KEM, and against randomly generated seed inputs on the FPGA target to see if any operations fail for both algorithms. The tested SLH-DSA accelerators pass the randomly generated tests at all security levels, confirmed with successful signature verification. All ML-KEM accelerators except [Tea25a] pass the KATs at all security levels. Specifically, their computed ciphertext

during encapsulation differs by 1–3 bytes on some of the tests, however the shared secret is correct. For decapsulation, there is no implicit rejection for an invalid ciphertext, which does not seem to propagate to the success or fail signal, but does include an internal check for this in the hardware design. Therefore, it passes decapsulation tests with a valid ciphertext, but not with an invalid ciphertext.

5.5 Future Improvements

There is room for expansion in the evaluation methodology, particularly in extending comparisons across additional platforms and security properties. Comparing designs across other CPUs for non-ISE accelerators could provide further insights into the trade-offs of different coupling levels between the accelerator and the processor.

Beyond performance, future work should assess the security properties of PQC accelerators in greater depth as part of Class D (Security Considerations). One direction is the analysis of constant-time behaviour at the hardware level [vGKSJ19, vGKSJ21], such that operations on secret data are independent of execution latency to prevent timing attacks. Evaluating works for more focussed side-channel security, such as employing a physical fault-injection setup [DO22, OC14], to verification of the provided HDL [FRBSG22, NOV+22], would further improve the trustworthiness of published hardware.

Finally, future work could incorporate physical FPGA power and energy measurements to complement on-chip estimations made by synthesis tools, making Class B (System Integration) analyses more realistic. Using a physical measurement setup, such as current shunt resistors or dedicated side-channel analysis platforms [OC14] which permit power analysis would allow direct observation of instantaneous power consumption during cryptographic operations. This would enable validation of reported energy figures and support more detailed side-channel evaluations, such as capture and analysis of power variations for secret data-dependent behaviour.

6 Recommendations

Stricter practices by authors and reviewers can reduce the discussed asymmetric reporting issue in Section 3 with some extra effort. We understand that some reporting limitations (e.g. platform heterogeneity) are more difficult to fix than others, yet some are quite simple. Our suggestions begin with straightforward solutions, followed by those that are more challenging for both parties.

6.1 For Authors

Thorough Metric Reporting. Performance metric reporting should be thorough and non-conflicting, with all hardware resources stated clearly and separately from any meta-analyses. Cycle or time metrics should be averaged with statistical significance of results stated. This increases the accuracy of future comparisons between works.

Open-source Artefacts. We recommend that authors should open source all design code where possible, inclusive of HDL testbenches, helper scripts, working bitstreams and driver code as appropriate. This supports its reproducibility in the future. Any provided documentation should clearly state the hardware design's 'top' file, how to use the repository's information and also the versions of associated software to remove barriers for replication. In the case that this is not possible due to intellectual property restrictions, as common with certain privately funded projects, authors could provide only the bitstream instead.

Rigorous Testing. Hardware designs require both KATs (where available) and random inputs to increase trust in their correctness. This should be run on the hardware platform

(e.g. on an FPGA), not in simulation, and cross-checked with a reference software version's outputs.

Relative Testing. Where appropriate, the design should be tested on a readily available prototyping platform to permit straightforward comparisons between works. Additionally, the authors should evaluate in a context suitable to their research goals e.g. a RISC-V SoC if aiming for acceleration in an embedded scenario. This provides better grounding for the paper's contributions and brings superiority claims closer to real-world performance.

Implementation Security Consideration. Authors should consider their implementation in context of physical attacks, e.g. side-channel leakage and fault injection. Due to the susceptibility of hardware to physical access and intrusion, these evaluations are meaningful and relevant for cryptographic hardware. Such discussion additionally identifies avenues for future hardware security research efforts.

6.2 For Reviewers

Identify Inappropriate Performance Evaluations. Determining whether the evaluation undertaken by the authors was fair, reasonable and complete helps reviewers reason whether claims of superiority are true or not.

Open-sourced Artefacts. Reviewers should consider whether the amount of provided detail is enough to accurately replicate the research, and therefore encourage the eventual availability of code artefacts alongside published works where possible. Combined with recommendations for authors, the community stands to benefit from increased access to cryptographic hardware and spur offshoot works.

7 Conclusion

Following NIST's recent standardisation efforts for post-quantum algorithms, we survey 50 PQC accelerator publications to characterise how the community reports results, and how fairer comparisons can be made between hardware designs. We identify an asymmetry in reported accelerator evaluations, where key metrics are unevenly available across works. This makes apples-to-apples comparisons difficult without introducing additional assumptions. We find that most works compare designs in the hardware implementation domain, while discussion of system integration, software footprint, and security remains uncommon.

In response, we present a benchmarking framework for PQC accelerators that helps address these gaps by demonstrating fair comparison and analysis of cryptographic hardware. We replicate five recent accelerators to accurately highlight their design trade-offs when integrated and evaluated within the scope of our chosen RISC-V-based SoC. Finally, our observations inform several guidelines for both authors and reviewers to improve cryptographic benchmarking comparability and reproducibility in future accelerator research.

Acknowledgments

This work was supported by an ARC Discovery Project number DP210102670; Defence Science and Technology Group (DSTG), Australia under Agreement No. 11965; and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and through project number 560392681.

References

[AAE06] Amara Amara, Frédéric Amiel, and Thomas Ea. FPGA vs. ASIC for low power applications. *Microelectronics Journal*, 2006. URL: https://www.sciencedirect.com/science/article/pii/S0026269205003927, doi:10.1016/j.mejo.2005.11.003.

[AB74] R. Agarwal and C. Burrus. Fast convolution using Fermat number transforms with applications to digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1974. doi:10.1109/TASSP.1974.1162555.

[ABC+23] Aikata Aikata, Andrea Basso, Gaetan Cassiers, Ahmet Can Mert, and Sujoy Sinha Roy. Kavach: Lightweight masking techniques for polynomial arithmetic in lattice-based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023. doi:10.46586/tches.v2023.i3.366-390.

[AEL+20] Erdem Alkim, Hülya Evkan, Norman Lahr, Ruben Niederhagen, and Richard Petri. ISA extensions for finite field arithmetic: Accelerating Kyber and NewHope on RISC-V. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020. URL: https://tches.iacr.org/index.php/TCHES/article/view/8589, doi:10.13154/tches.v2020.i3.219-242.

[AMD+21] Abubakr Abdulgadir, Kamyar Mohajerani, Viet Ba Dang, Jens-Peter Kaps, and Kris Gaj. A lightweight implementation of Saber resistant against side-channel attacks. In *Progress in Cryptology – INDOCRYPT 2021*, 2021. doi:10.1007/978-3-030-92518-5_11.

[AOHP+25] Amin Abdulrahman, Felix Oberhansl, Hoang Nguyen Hien Pham, Jade Philipoom, Peter Schwabe, Tobias Stelzer, and Andreas Zankl. Towards ML-KEM & ML-DSA on OpenTitan. In *2025 IEEE Symposium on Security and Privacy (SP)*, 2025. doi:10.1109/SP61157.2025.00220.

[Apo19] Daniel C. Apon. On recommended hardware. NIST PQC Forum, February 2019. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/cJxMq0_90gU/m/qbGEs3TXGwAJ.

[Atk21] Derek Atkins. Requirements for post-quantum cryptography on embedded devices in the IoT. In *Third PQC Standardization Conference*, 2021. URL: https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/atkins-requirements-pqc-iot-pqc2021.pdf.

[BDPVA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Advances in Cryptology – EUROCRYPT 2013*, 2013. doi:10.1007/978-3-642-38348-9_19.

[BKG22] Luke Beckwith, J Kaps, and Kris Gaj. FPGA energy consumption of post-quantum cryptography. In *Fourth PQC Standardization Conference*, 2022. URL: https://csrc.nist.gov/csrc/media/Events/2022/fourth-pqc-standardization-conference/documents/papers/fpga-energy-consumption-of-pqc-pqc2022.pdf.

[BNAMK21] Mojtaba Bisheh-Niasar, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Instruction-set accelerated implementation of CRYSTALS-Kyber. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021. doi:10.1109/TCSI.2021.3106639.

[BNG21] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. High-performance hardware implementation of CRYSTALS-Dilithium, 2021. doi:10.1109/ICFPT52863.2021.9609917.

[Bou17] Sebastien Bourdeauducq. Migen manual. https://app.readthedocs.org/projects/mithro-migen/downloads/pdf/latest/, October 2017.

[BSNK19] Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri. NIST post-quantum cryptography - a hardware evaluation study. Cryptology ePrint Archive, Paper 2019/047, 2019. URL: https://eprint.iacr.org/2019/047.

[BUC19] Utsav Banerjee, Tenzin S. Ukyab, and Anantha P. Chandrakasan. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019. URL: https://tches.iacr.org/index.php/TCHES/article/view/8344, doi:10.13154/tches.v2019.i4.17-61.

[CCD+22] Po-Jen Chen, Tung Chou, Sanjay Deshpande, Norman Lahr, Ruben Niederhagen, Jakub Szefer, and Wen Wang. Complete and improved FPGA implementation of Classic McEliece. Cryptology ePrint Archive, Paper 2022/412, 2022. URL: https://eprint.iacr.org/2022/412, doi:10.46586/tches.v2022.i3.71-113.

[Cen25] NIST Computer Security Resource Center. Post-quantum cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography, December 2025.

[CRNTMRB25] Eros Camacho-Ruiz, Pablo Navarro-Torrero, M. C. Martínez-Rodríguez, and P. Brox. Final implementation of the secure element. https://qubip.eu/final-implementation-of-the-secure-element/, 2025. QUBIP Project Blog, 3 October 2025.

[Des25] Sanjay Deshpande. *Hardware Accelerators for Post-Quantum Cryptography–A Comprehensive Study of PQC Hardware Accelerators Belonging to Different Families*. PhD thesis, Yale University, 2025.

[DLK+25] Sanjay Deshpande, Yongseok Lee, Cansu Karakuzu, Jakub Szefer, and Yunheung Paek. SPHINCSLET: An area-efficient accelerator for the full SPHINCS+ digital signature algorithm. *ACM Transactions on Embedded Computing Systems*, 2025. doi:10.1145/3728469.

[DMG23] Viet Ba Dang, Kamyar Mohajerani, and Kris Gaj. High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NTRU, and Saber. *IEEE Transactions on Computers*, 2023. doi:10.1109/TC.2022.3222954.

[DO22] Shaked Delarea and Yossi Oren. Practical, low-cost fault injection attacks on personal smart devices. *Applied Sciences*, 12(1):417, 2022. doi:10.3390/app12010417.

[DTH+23] Duc-Thuan Dam, Thai-Ha Tran, Van-Phuc Hoang, Cong-Kha Pham, and Trong-Thuc Hoang. A survey of post-quantum cryptography: Start of a new race. *Cryptography*, 2023. URL: https://www.mdpi.com/2410-387X/7/3/40, doi:10.3390/cryptography7030040.

[DVMM24] Alessandra Dolmeta, Emanuele Valpreda, Maurizio Martina, and Guido Masera. Implementation and integration of NTT/INTT accelerator on RISC-V for CRYSTALS-Kyber. In *Proceedings of the 21st ACM International Conference on Computing Frontiers: Workshops and Special Sessions*, 2024. doi:10.1145/3637543.3652872.

[EGHP09] Thomas Eisenbarth, Tim Güneysu, Stefan Heyse, and Christof Paar. MicroEliece: McEliece for embedded devices. In *Cryptographic Hardware and Embedded Systems – CHES 2009*, 2009. doi:10.1007/978-3-642-04138-9_4.

[Fah24] Suhaib Fahmy. Artifact evaluation in the FPGA community. https://tcfpga.org/books/artifacts-and-open-source/page/artifact-evaluation-in-the-fpga-community, 2024.

[fCM20] Association for Computing Machinery. Artifact review and badging version 1.1. https://www.acm.org/publications/policies/artifact-review-and-badging-current, August 2020.

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to NIST's post-quantum cryptography standardization process*, 2018. URL: https://falcon-sign.info/falcon.pdf.

[FRBSG22] Jakob Feldtkeller, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. CINI MINIS: Domain isolation for fault and combined security. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. doi:10.1145/3548606.3560614.

[FRD+24] Frank Wilhelm, Rainer Steinwandt, Daniel Zeuch, Paul Lageyre, and Susanna Kirchhoff. Status of quantum computer development, August 2024. URL: https://www.bsi.bund.de/dok/study_status_quantum_computer.

[FSS20a] Tim Fritzmann, Georg Sigl, and J. Sepulveda. RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020. doi:10.46586/tches.v2020.i4.239-280.

[FSS20b] Tim Fritzmann, Georg Sigl, and Johanna Sepúlveda. Extending the RISC-V instruction set for hardware acceleration of the post-quantum scheme LAC. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020. doi:10.23919/DATE48585.2020.9116567.

[FVBBR+21] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021. URL: https://tches.iacr.org/index.php/TCHES/article/view/9303, doi:10.46586/tches.v2022.i1.414-460.

[FVS19] Tim Fritzmann, Jonas Vith, and Johanna Sepúlveda. Post-quantum key exchange mechanism for safety critical systems, 2019. URL: https://hss-opus.ub.ruhr-uni-bochum.de/opus4/6653, doi:10.13154/294-6653.

[GFS+12] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, 2012. doi:10.1007/978-3-642-33027-8_30.

[GLM24] Carlos Gewehr, Lucas Luza, and Fernando Gehm Moraes. Hardware acceleration of Crystals-Kyber in low-complexity embedded systems with RISC-V instruction set extensions. *IEEE Access*, 2024. doi:10.1109/ACCESS.2024.3416812.

[GOW+24] Ivan Gavrilan, Felix Oberhansl, Alexander Wagner, Emanuele Strieder, and Andreas Zankl. Impeccable keccak: Towards fault resilient SPHINCS+ implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024, 2024. URL: https://tches.iacr.org/index.php/TCHES/article/view/11424, doi:10.46586/tches.v2024.i2.154-189.

[GYCH18] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 2018. doi:10.1007/s13389-016-0141-6.

[HBTX22] Pengzhou He, Tianyou Bao, Yazheng Tu, and Jiafeng Xie. HPMA-Saber: High-performance polynomial multiplication accelerator for KEM Saber. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, 2022. doi:10.1109/ICCD56317.2022.00083.

[HCS+21] Wei Hu, Chip-Hong Chang, Anirban Sengupta, Swarup Bhunia, Ryan Kastner, and Hai Li. An overview of hardware security and trust: Threats, countermeasures, and design tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021. doi:10.1109/TCAD.2020.3047976.

[IAP20a] Malik Imran, Zain Ul Abideen, and Samuel Pagliarini. An experimental study of building blocks of lattice-based NIST post-quantum cryptographic algorithms. *Electronics*, 2020. URL: https://www.mdpi.com/2079-9292/9/11/1953, doi:10.3390/electronics9111953.

[IAP20b] Malik Imran, Zain Ul Abideen, and Samuel Pagliarini. A systematic study of lattice-based nist pqc algorithms: From reference implementations to hardware accelerators. *arXiv preprint arXiv:2009.07091*, 2020. URL: https://arxiv.org/abs/2009.07091.

[JMM+22] David Joseph, Rafael Misoczki, Marc Manzano, Joe Tricot, Fernando Dominguez Pinuaga, Olivier Lacombe, Stefan Leichenauer, Jack Hidary, Phil Venables, and Royal Hansen. Transitioning organizations to post-quantum cryptography. *Nature*, 2022. doi:10.1038/s41586-022-04623-2.

[JO24] Heonhui Jung and Hyunyoung Oh. Designing a scalable and area-efficient hardware accelerator supporting multiple PQC schemes. *Electronics*,

2024. URL: https://www.mdpi.com/2079-9292/13/17/3360, doi:10.3390/electronics13173360.

[KA24] Emre Karabulut and Aydin Aysu. Masking FALCON's floating-point multiplication in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024. URL: https://tches.iacr.org/index.php/TCHES/article/view/11800, doi:10.46586/tches.v2024.i4.483-508.

[KAB+25] Patrick Karl, Francesco Antognazza, Alessandro Barenghi, Gerardo Pelosi, and Georg Sigl. High-performance FPGA accelerator for the post-quantum signature scheme CROSS. Cryptology ePrint Archive, Paper 2025/1161, 2025. URL: https://eprint.iacr.org/2025/1161.

[KAK+19] Brian Koziel, A-Bon Ackie, Rami El Khatib, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. SIKE'd up: Fast and secure hardware architectures for supersingular isogeny key encapsulation. Cryptology ePrint Archive, Paper 2019/711, 2019. URL: https://eprint.iacr.org/2019/711.

[KAMK16] Brian Koziel, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Fast hardware architectures for supersingular isogeny Diffie-Hellman key exchange on FPGA. In *Progress in Cryptology – INDOCRYPT 2016*, 2016. doi:10.1007/978-3-319-49890-4_11.

[KBLB20] Florent Kermarrec, Sébastien Bourdeauducq, Jean-Christophe Le Lann, and Hannah Badier. LiteX: an open-source SoC builder and library based on migen python DSL, 2020. URL: https://arxiv.org/abs/2005.02506, arXiv:2005.02506.

[KPR+24] Matthias J. Kannwischer, Richard Petri, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4, 2024. https://github.com/mupq/pqm4.

[KR07] Ian Kuon and Jonathan Rose. Measuring the gap between FPGAs and asics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007. doi:10.1109/TCAD.2006.884574.

[KRGF+24] Vatistas Kostalabros, Jordi Ribes-González, Oriol Farràs, Miquel Moretó, and Carles Hernandez. A safety-critical, RISC-V SoC integrated and ASIC-ready Classic McEliece accelerator. In *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, 2024. doi:10.1007/978-3-031-55673-9_20.

[KRR+20] Daniel Kales, Sebastian Ramacher, Christian Rechberger, Roman Walch, and Mario Werner. Efficient FPGA implementations of LowMC and Picnic. In *Topics in Cryptology – CT-RSA 2020*, 2020. doi:10.1007/978-3-030-40186-3_18.

[KSFS24] Patrick Karl, Jonas Schupp, Tim Fritzmann, and Georg Sigl. Post-quantum signatures on RISC-V with hardware acceleration. *ACM Transactions on Embedded Computing Systems*, 2024. doi:10.1145/3579092.

[KSSW22] Matthias J Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. Improving software quality in cryptography standardization projects. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2022. doi:10.1109/EuroSPW55150.2022.00010.

[low24] lowRISC. prim_keccak.sv. https://github.com/lowRISC/opentitan/blob/master/hw/ip/prim/rtl/prim_keccak.sv, December 2024.

[LQYW24] Lu Li, Guofeng Qin, Yang Yu, and Weijia Wang. Compact instruction set extensions for Kyber. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024. doi:10.1109/TCAD.2023.3327104.

[LRJ24] Tao Liu, Gowri Ramachandran, and Raja Jurdak. Post-quantum cryptography for internet of things: A survey on performance and optimization, 2024. URL: https://arxiv.org/abs/2401.17538, arXiv:2401.17538.

[LSG22] Georg Land, Pascal Sasdrich, and Tim Güneysu. A Hard Crystal - implementing Dilithium on reconfigurable hardware. In *Smart Card Research and Advanced Applications*, 2022. doi:10.1007/978-3-030-97348-3_12.

[LVC24] Jonathan Lopez-Valdivieso and Rene Cumplido. Design and implementation of hardware-software architecture based on hashes for SPHINCS+. *ACM Transactions on Reconfigurable Technology and Systems*, 2024. doi:10.1145/3653459.

[MAB+18] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and IC Bourges. Hamming quasi-cyclic (HQC). *Submission to NIST's post-quantum cryptography standardization process*, 2018. URL: https://pqc-hqc.org/.

[MBR15] Pedro Maat C. Massolino, Paulo S. L. M. Barreto, and Wilson V. Ruggiero. Optimized and scalable co-processor for McEliece with binary Goppa codes. *ACM Transactions on Embedded Computing Systems*, 2015. doi:10.1145/2736284.

[MPR+24] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. Technical report, National Institute of Standards and Technology, 2024. URL: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf, doi:10.6028/NIST.IR.8547.

[MR98] H.-G. Martin and W. Rosenstiel. A comparing study of technology mapping for fpga. In *Proceedings Design, Automation and Test in Europe*, 1998. doi:10.1109/DATE.1998.655979.

[NDMZ+21] Pietro Nannipieri, Stefano Di Matteo, Luca Zulberti, Francesco Albicocchi, Sergio Saponara, and Luca Fanucci. A RISC-V post quantum cryptography instruction set extension for number theoretic transform to speed-up CRYSTALS algorithms. *IEEE Access*, 2021. doi:10.1109/ACCESS.2021.3126208.

[NDR+19] Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys*, 2019. doi:10.1145/3292548.

[NHNN25] Hien Nguyen, Samsul Huda, Yasuyuki Nogami, and Tuy Tan Nguyen. Security in post-quantum era: A comprehensive survey on lattice-based algorithms. *IEEE Access*, 2025. doi:10.1109/ACCESS.2025.3571307.

[NIS24] NIST. ACVP-Server. https://github.com/usnistgov/ACVP-Server, November 2024.

[NIS25] NIST Computer Security Resource Center. Post-quantum cryptography: Selected algorithms. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms, 2025.

[NKOL22] Ziying Ni, Dur-e-Shahwar Kundi, Máire O'Neill, and Weiqiang Liu. A high-performance SIKE hardware accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022. doi:10.1109/TVLSI.2022.3152011.

[NOV+22] Pascal Nasahl, Miguel Osorio, Pirmin Vogel, Michael Schaffner, Timothy Trippel, Dominic Rizzo, and Stefan Mangard. Synfi: Pre-silicon fault analysis of an open-source secure element. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022. doi:10.46586/tches.v2022.i4.56-87.

[OC14] Colin O'Flynn and Zhizhang Chen. ChipWhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer, 2014. doi:10.1007/978-3-319-10175-0_17.

[OG19] Tobias Oder and Tim Güneysu. Implementing the NewHope-Simple key exchange on low-cost FPGAs. In *Progress in Cryptology – LATINCRYPT 2017*, 2019. doi:10.1007/978-3-030-25283-0_7.

[OPowp19] David Ott, Christopher Peikert, and other workshop participants. Identifying research challenges in post quantum cryptography migration and cryptographic agility, 2019. URL: https://arxiv.org/abs/1909.07353, arXiv:1909.07353.

[oST15] National Institute of Standards and Technology. *SHA-3 standard: permutation-based hash and extendable-output functions*. 2015. URL: http://dx.doi.org/10.6028/NIST.FIPS.202, doi:10.6028/nist.fips.202.

[oST24a] National Institute of Standards and Technology. *Module-lattice-based digital signature standard*. August 2024. URL: http://dx.doi.org/10.6028/nist.fips.204, doi:10.6028/nist.fips.204.

[oST24b] National Institute of Standards and Technology. *Module-lattice-based key-encapsulation mechanism standard*. August 2024. URL: http://dx.doi.org/10.6028/nist.fips.203, doi:10.6028/nist.fips.203.

[oST24c] National Institute of Standards and Technology. *Stateless hash-based digital signature standard*. August 2024. URL: http://dx.doi.org/10.6028/nist.fips.205, doi:10.6028/nist.fips.205.

[OZZ+24] Yi Ouyang, Yihong Zhu, Wenping Zhu, Bohan Yang, Zirui Zhang, Hanning Wang, Qichao Tao, Min Zhu, Shaojun Wei, and Leibo Liu. FalconSign: An efficient and high-throughput hardware architecture for Falcon signature generation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024. URL: https://tches.iacr.org/index.php/TCHES/article/view/11927, doi:10.46586/tches.v2025.i1.203-226.

[PDG14] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In *Cryptographic Hardware and Embedded Systems – CHES 2014*, 2014. doi:10.1007/978-3-662-44709-3_20.

[PG13] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *Selected Areas in Cryptography*, 2013. doi:10.1007/978-3-662-43414-7_4.

[PPPH24] Maximilian Pursche, Nikolai Puch, Sebastian N. Peters, and Michael P. Heinl. SoK: The engineer's guide to post-quantum cryptography for embedded devices. Cryptology ePrint Archive, Paper 2024/1345, 2024. URL: https://eprint.iacr.org/2024/1345.

[RBCGG22] Jan Richter-Brockmann, Ming-Shing Chen, Santosh Ghosh, and Tim Güneysu. Racing BIKE: Improved polynomial multiplication and inversion in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022. doi:10.46586/tches.v2022.i1.557-588.

[RBFSG22] Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. Verica - verification of combined attacks: Automated formal verification of security against simultaneous information leakage and tampering. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022. URL: https://tches.iacr.org/index.php/TCHES/article/view/9820, doi:10.46586/tches.v2022.i4.255-284.

[RBMG22] Jan Richter-Brockmann, Johannes Mono, and Tim Güneysu. Folding BIKE: Scalable hardware implementation for reconfigurable devices. *IEEE Transactions on Computers*, 2022. doi:10.1109/TC.2021.3078294.

[RMJ+21] Sara Ricci, Lukas Malina, Petr Jedlicka, David Smékal, Jan Hajny, Peter Cibik, Petr Dzurenda, and Patrik Dobias. Implementing CRYSTALS-Dilithium signature scheme on FPGAs. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES '21, 2021. doi:10.1145/3465481.3465756.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, February 1978. URL: https://dl.acm.org/doi/10.1145/359340.359342, doi:10.1145/359340.359342.

[Saa24] Markku-Juhani O. Saarinen. Artifact evaluation and reproducibility (at CHES). https://mjos.fi/doc/20240904-optimist-artifact.pdf, September 2024. OPTIMIST Workshop.

[SBN+21] Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, Marc Manzano, and Ramesh Karri. *Hardware Architectures for Post-Quantum Digital Signature Schemes*. 2021. URL: http://dx.doi.org/10.1007/978-3-030-57682-0, doi:10.1007/978-3-030-57682-0.

[SBNK19] Deepraj Soni, Kanad Basu, Mohammed Nabeel, and Ramesh Karri. A hardware evaluation study of NIST post-quantum cryptographic signature schemes. Second PQC Standardization Conference, NIST, 2019. URL: https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/soni-hardware-evaluation.pdf.

[Sho94] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994. URL: http://ieeexplore.ieee.org/document/365700/, doi:10.1109/SFCS.1994.365700.

[SM16] Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the Open Quantum Safe project. In *Selected Areas in Cryptography*, 2016. doi:10.1007/978-3-319-69453-5_2.

[SMPQ+23] Pasquale Davide Schiavone, Simone Machetti, Miguel Peón-Quirós, Jose Miranda, Benoît Denkinger, Thomas Christoph Müller, Ruben Rodríguez, Saverio Nasturzio, and David Atienza Alonso. X-HEEP: An open-source, configurable and extendible RISC-V microcontroller. In *Proceedings of the 20th ACM International Conference on Computing Frontiers*, CF '23, 2023. doi:10.1145/3587135.3591431.

[SRB20] Sujoy Sinha Roy and Andrea Basso. High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020. doi:10.46586/tches.v2020.i4.443-466.

[Sub17] Karthikeyan Subramaniyam. Total power advantage using spartan-7 fpgas. White Paper WP488, Xilinx, 2017. URL: https://docs.amd.com/v/u/en-US/wp488-spartan7-power.

[Tea24] OpenTitan Team. OpenTitan, 2024. URL: https://opentitan.org/book/doc/introduction.html#introduction-to-opentitan.

[Tea25a] CSIC-IMSE Team. mlkem. https://github.com/HWSec-CSIC/se-pq/tree/main/se-qubip/rtl/mlkem, March 2025.

[Tea25b] CSIC-IMSE Team. sha3. https://github.com/HWSec-CSIC/se-pq/tree/main/se-qubip/rtl/sha3, March 2025.

[TG21] Jan Philipp Thoma and Tim Güneysu. A configurable hardware implementation of XMSS. Cryptology ePrint Archive, Paper 2021/352, 2021. URL: https://eprint.iacr.org/2021/352.

[THG24] Jan Philipp Thoma, Darius Hartlief, and Tim Güneysu. Agile acceleration of stateful hash-based signatures in hardware. *ACM Transactions on Embedded Computing Systems*, 2024. doi:10.1145/3567426.

[vGKSJ19] Klaus v. Gleissenthall, Rami Gökhan Kıcı, Deian Stefan, and Ranjit Jhala. IODINE: Verifying Constant-Time execution of hardware. In *USENIX Sec.*, 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/von-gleissenthall.

[vGKSJ21] Klaus v. Gleissenthall, Rami Gökhan Kıcı, Deian Stefan, and Ranjit Jhala. Solver-aided constant-time hardware verification. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021. doi:10.1145/3460120.3484810.

[WSN18] Wen Wang, Jakub Szefer, and Ruben Niederhagen. FPGA-based Niederreiter cryptosystem using binary Goppa codes. In *Post-Quantum Cryptography*, 2018. doi:10.1007/978-3-319-79063-3_4.

[XHY+20] Guozhu Xin, Jun Han, Tianyu Yin, Yuchao Zhou, Jianwei Yang, Xu Cheng, and Xiaoyang Zeng. VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020. doi:10.1109/TCSI.2020.2983185.

[XL21] Yufei Xing and Shuguo Li. A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021. doi:10.46586/tches.v2021.i2.328-356.

[YMÖS21] Ferhat Yaman, Ahmet Can Mert, Erdinç Öztürk, and Erkay Savaş. A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021. doi:10.23919/DATE51398.2021.9474139.

[ZYC+20] Neng Zhang, Bohan Yang, Chen Chen, Shouyi Yin, Shaojun Wei, and Leibo Liu. Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020. URL: https://tches.iacr.org/index.php/TCHES/article/view/8544, doi:10.13154/tches.v2020.i2.49-72.

[ZZY+21] Yihong Zhu, Min Zhu, Bohan Yang, Wenping Zhu, Chenchen Deng, Chen Chen, Shaojun Wei, and Leibo Liu. LWRpro: An energy-efficient configurable crypto-processor for Module-LWR. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021. doi:10.1109/TCSI.2020.3048395.