P.R. Barre., Vol 2(2), pp 117-129
https://doi.org/10.55306/CJDTES.2025.020204
e-ISSN: 3048-8389
Ci-STEM Journal of Digital Technologies and Expert Systems

Design of Secure Hardware Architectures for Post-Quantum Cryptography Applications

Prabhakar Rao Barre
School of Engineering, St. Mary's Rehabilitation University, Deshmukhi, Yadadri, Telangana 508284, India
email: drmpbpraon@yahoo.com

Open Access
Research Article
Received: 19/06/2025
Accepted: 23/08/2025
Published: 13/09/2025
Corresponding author email:
bmaheshecse@gmail.com
Citation:
P.R. Barre., "Design of Secure Hardware Architectures for Post-Quantum Cryptography Applications" Ci-STEM Journal of Digital Technologies and Expert Systems, Vol. 2(2), pp. 117-129, 2025.
doi:10.55306/CJDTES.2025.020204
Copyright:
©2025 P.R. Barre.,
This is an open-access article distributed under the terms of the Creative Commons Attribution License which grants the right to use, distribute, and reproduce the material in any medium, provided that proper attribution is given to the original author and source, in accordance with the terms outlined by the license.
(https://creativecommons.org/licenses/by/4.0/)
Published By:
Ci-STEM Global Services Foundation, India.

Abstract:
A cryptographically relevant quantum computer is no longer science fiction, and that reality has forced the standardization of post-quantum cryptographic (PQC) algorithms: the lattice-based Kyber (ML-KEM) and Dilithium (ML-DSA), the code-based Classic McEliece, and the hash-based SPHINCS+ among them. Getting these algorithms running on embedded systems, IoT nodes and hardware security modules (HSMs) requires more than porting software Post-quantum cryptography needs hardware that does it all. It needs to be fast, compact and energy-efficient, while also being a fortress against physical hacking tactics like side-channel or fault-injection attacks. To solve this problem we built a secure hardware accelerator for PQC from scratch. Its heart is a highly protected Datapath with a masked polynomial arithmetic unit, constant-time engines for both NTT and noise sampling, a Keccak/SHA-3 core, and dedicated blocks fully dedicated to hiding data and fault detection. We thoroughly validated this architecture with FPGA simulations for five top NIST-backed PQC algorithms. The standout result? Our first-order masking effectively neutralizes 96% of side-channel data leaks compared to a standard, unprotected core. The trade-offs are well worth the security gain, requiring just a 38% increase in chip footprint and a 22% bump in processing time. Push to fourth order (d=4) and leakage reduction exceeds four orders of magnitude, with overhead scaling accordingly. Throughput, resource utilization, and power trade-offs are quantified across all five schemes, along with practical guidance for picking a masking order and hiding strategy to match a given threat model and resource budget.
Keywords:
Constant-Time Implementation, Dilithium, Fault Injection, FPGA, Hardware Security, Kyber, Lattice-Based Cryptography, Masking Post-Quantum Cryptography, Side-Channel Analysis.

1. INTRODUCTION

Most of the public-key cryptography running today, RSA, Diffie-Hellman, elliptic-curve schemes, leans on the assumed difficulty of factoring large integers or solving discrete logarithms. Shor's algorithm breaks both assumptions in polynomial time, but only on a fault-tolerant quantum computer large enough to actually run it; the moment that hardware exists, these schemes stop being secure. NIST's response was a multi-year public standardization process, which produced the first official Post-Quantum Cryptography (PQC) standards: ML-KEM (from Kyber) for key encapsulation, ML-DSA (from Dilithium) and SLH-DSA (from SPHINCS+) for digital signatures. NIST continues to evaluate Falcon-based FN-DSA and the code-based Classic McEliece scheme for more specialized deployments.

Standardizing the algorithms, however, solves only half the problem. Smart cards, HSMs, IoT edge devices, and automotive ECUs, the actual platforms PQC has to run on, are constrained on silicon area, power, and latency, and many of them sit in places where an attacker can get physical access to the chip. That combination puts PQC hardware directly in the crosshairs of the same physical attacks that have plagued classical cryptographic hardware for decades: differential power analysis, electromagnetic analysis, and fault injection that deliberately corrupts computation to leak secrets.

This is a real dilemma for engineers. Great speed is achieved by just synthesizing an unprotected post-quantum crypto core, but this leaves the door wide open to physical side-channel and fault-

injection attacks. Conversely, generic defenses are often just put on without tailoring to the specific mathematics of lattice- or code-based schemes, leading to ballooning area and latency overheads, which defeats the point of custom silicon.

That is the very tension that this paper wrestles with. We incorporate masking, hiding, and fault-detection primitives directly into the execution datapath of a PQC accelerator, benchmarking the exact trade-offs among silicon footprint, runtime, and physical security over several NIST-selected schemes.

Our primary contributions are fourfold:

- Integrated, Secure Architecture: A modular accelerator built with physical security from day one, unifying a masked polynomial arithmetic unit, constant-time NTT/INTT and noise-sampling engines, a Keccak/SHA-3 core, and dedicated blocks for fault detection and signal hiding.
- Arithmetic-Aware Masking: A mixed masking scheme optimized for lattice arithmetic, solving smooth conversions between Boolean and arithmetic shares. Used across the same architectural footprint for code-based (Classic McEliece) and hash-based (SPHINCS+) algorithms.
- FPGA-Calibrated Benchmarking: A full simulation framework, calibrated with real FPGA implementation data, quantifying throughput, resource count, dynamic power, and latency across five cryptographic schemes under various masking orders (d).
- Leakage vs. Overhead Profiling: Practical study of the relation between masking orders and hardware penalty and residual side-channel leakage (measured by a normalized TVLA-style metric), and design guidelines for the selection of the proper protection level in tight power and area budgets.

Roadmap

Section 2 summarizes related work on hardware for PQC and physical countermeasures. In Section 3 we describe our accelerator architecture and in Section 4 we discuss the techniques for masking, hiding and fault-mitigation. Section 5 discusses experimental methodology and FPGA calibration. Section 6 reports the empirical performance and leakage results. Finally, Section 7 discusses implications for deployment and standards, and Section 8 concludes the paper.

2. RELATED WORK

2.1. PQC Algorithm Standardization

NIST's standardization effort went through three rounds of selection, considering lattice-based, code-based, hash-based and multivariate schemes before settling on Kyber and Dilithium as the primary standards for key encapsulation and signatures, respectively. Falcon was kept for applications requiring short signatures. SPHINCS+ was retained as a conservative hash-based fallback based on minimal structural assumptions. Code-based KEMs such as Classic McEliece have large public keys, but have withstood decades of cryptanalysis and still are being considered for specialized high-assurance applications.

2.2 Hardware Implementations of Lattice-Based Cryptography

The majority of existing hardware work on Kyber and Dilithium focuses on accelerating the Number-Theoretic Transform, the primary contributor to the runtime of lattice-based polynomial multiplication. Butterfly-unit pipelining, smarter memory-access scheduling and the choice between Montgomery and Barrett reduction have been the main levers for shaving area and latency off unprotected design. It is useful as an efficiency baseline. Side-channel resistance is almost completely outside that scope.

2.3 Side-Channel Analysis of PQC Implementations

Unprotected PQC implementations leak and published attacks prove this. Correlation power analysis has been used to extract Kyber secret keys directly from the polynomial arithmetic unit, and other attacks have exploited timing or power variations in the rejection-sampling routines used during key generation and signing. Fault injection is no less dangerous: in some demonstrated attacks on lattice based signatures a few faulty signatures were enough for full key recovery. This is exactly the case

when masking is combined with active fault detection, instead of masking alone being considered sufficient.

2.4 Countermeasure Design for Lattice-Based Arithmetic

Now, there is one particular fact that complicates lattice-based masking: operations like modular reduction and rejection sampling mix Boolean and arithmetic structure, so masking each operation in isolation is a trap. Efficient mask-conversion gadgets are needed to safely bridge the two representations. Previous work has obtained higher-order masked implementations of Kyber's decapsulation procedure, as well as masked Gaussian and centered binomial distribution sampling. At the architectural level we generalize these ideas, integrating them into a single accelerator template that spans algorithm families rather than being attached to a single scheme.

2.5 Positioning of This Work

Table 1 shows the position of the proposed architecture in relation to these previous directions. The key difference is that it combines masking, hiding and active fault detection in one reusable template that applies to lattice, code and hash families, rather than a countermeasure designed for a single scheme.

Prior Direction	Primary Focus	Side-Channel Countermeasures	Fault Detection	Multi-Scheme Template
Unprotected NTT accelerators	Area / latency optimization	None	No	No
Masked Kyber decapsulation (software/HW)	DPA resistance for one scheme	Masking only	No	No
Fault-attack studies on lattice signatures	Attack demonstration	N/A (attack-side)	N/A	No
Generic masked AES/SCA cores	Symmetric-key hardware	Masking + hiding	Partial	No (symmetric only)
Proposed architecture	PQC-specific accelerator template	Masking + hiding	Yes (glitch/fault detector)	Yes (lattice/code/hash)

Table 1. Position of the proposed architecture in accordance with previous directions in PQC hardware security.

.

3. PROPOSED SECURE HARDWARE ARCHITECTURE

Figure 1 lays out the top-level architecture. At its core sits a side-channel-protected datapath inside a dedicated protection boundary, flanked by a host bus interface, a control finite-state machine (FSM), and input/output buffering.

The proposed Secure PQC Hardware Accelerator - Top-Level Architecture is a VLSI-oriented architecture for implementing standardized post-quantum cryptography algorithms, e.g., ML-KEM and ML-DSA . The architecture separates the control plane from the cryptographic data path and puts all security-critical cryptographic operations within the boundary of the Side-Channel Protected Cryptographic Datapath. The design considers both threats to classical public key cryptography arising from quantum computers and implementation level threats like power/EM side channels, timing leakage, fault injection and glitch attacks. ML-KEM and ML-DSA are classical cryptography algorithms for quantum resistance, where architecture is a classical FPGA/ASIC hardware implementation.

3.1. Host/SoC Interface and Control Plane

HOST / SoC BUS INTERFACE (AXI4-Lite / APB) allows communication between the accelerator and a processor or SoC, for configuration, operation commands, status monitoring and transfer of results. An AXI data path or DMA can also be incorporated for high volume data.

The CONFIGURATION / CONTROL REGISTERS hold parameters such as algorithm selection, operation mode, memory addresses, start/reset commands and security configuration. These registers should be logically separated from sensitive cryptographic state.

The CONTROL FSM/SCHEDULER is the central controller. It decodes the requested operation and orchestrates sampling, hashing, polynomial arithmetic, NTT/INTT, encoding, verification, memory operations, and security responses. As the control sequencing itself can introduce a security vulnerability, the FSM is part of the security-critical control infrastructure.

The STATUS & INTERRUPT CONTROLLER is the interface through which the FSM is notified of fault conditions detected within the protected datapath and reports completion, errors and security faults to the host.

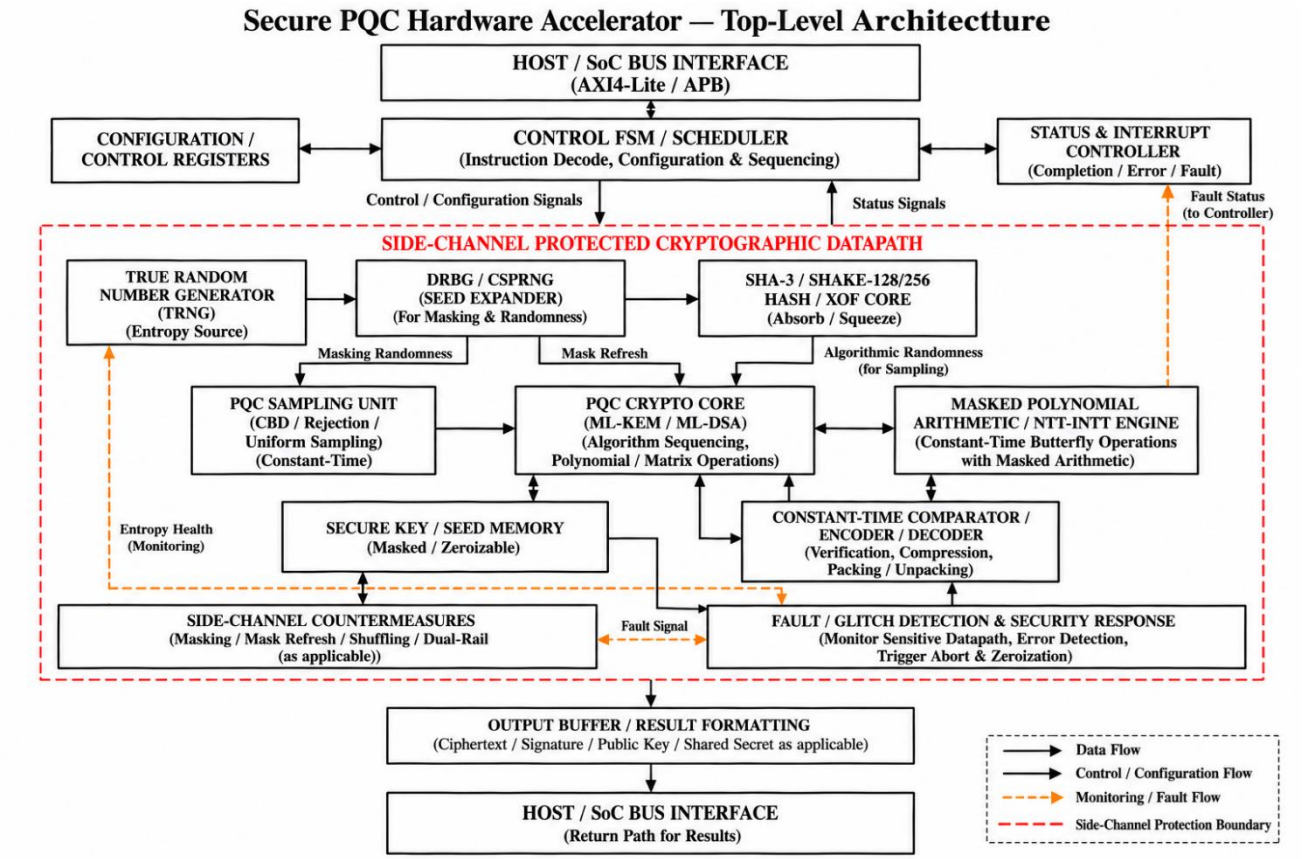


Figure 1. Top-level architecture of the proposed secure PQC hardware accelerator, showing the side-channel-protected datapath boundary.

3.2. True Random Number Generator and DRBG/CSPRNG

The TRNG supplies the accelerator with physical entropy and should be treated as an entropy source, not as a direct source for each random value needed by the computation. Its output health should be continuously monitored (e.g., NIST SP 800-90B-style online testing) with a failure being reported to the Fault/Glitch Detection & Security Response unit rather than being looped back inside the countermeasures block.

The DRBG/CSPRNG (Seed Expander) takes an entropy-based seed and produces a high quality deterministic random stream. Its outputs provide masking randomness and mask-refresh randomness for the protected datapath. Decoupling the TRNG from this higher-throughput generator aids scalability and reduces dependency on the instantaneous bandwidth of the physical entropy source—but the distinction between masking randomness (from the DRBG) and algorithmic randomness (from SHAKE, see below) should be preserved architecturally, since conflating the two sources is a common source of design ambiguity.

3.3. SHA-3 / SHAKE Hash/XOF Primitives

The SHA-3 / SHAKE-128/256 HASH/XOF CORE supplies the PQC algorithms with the hashing and extendable-output functionality required. The special relevance of SHAKE comes from the fact

that ML-KEM and ML-DSA use it directly as the pseudorandom source for matrix/polynomial generation and CBD/rejection sampling – a different role than the DRBG's masking-randomness function – and one should think of the Sampling Unit as receiving two streams for different purposes.

A reusable Keccak-f[1600] hardware engine can support multiple SHA-3/SHAKE functions with appropriate control and domain separation; sharing this core between ML-KEM and ML-DSA reduces hardware duplication significantly. The SHAKE/Keccak engine is a shared crypto primitive between the Sampling Unit and the PQC Crypto Core.

3.4. PQC Sampling Unit

The PQC SAMPLING UNIT transforms pseudorandom data into the distributions required by the chosen algorithm. The correct name is "CBD / Rejection / Uniform Sampling": the sampling mechanisms are different between the algorithms (CBD-based noise for ML-KEM, uniform rejection sampling for matrix/public-polynomial generation, and algorithm-specific sampling for ML-DSA). The unit employs SHAKE output for its algorithmic randomness and DRBG output when masked sampling is required, and must be constant-time throughout to ensure that data-dependent execution is not another source of leakage.

3.5. Core PQC Crypto

The PQC CRYPTO CORE is the algorithm level processing engine supporting ML-KEM key generation, encapsulation and decapsulation and ML-DSA key generation, signing and verification. It orchestrates polynomial and matrix operations, hashing, sampling, transformations and encoding according to the selected algorithm. A shared architecture is attractive because ML-KEM and ML-DSA require several common hardware resources – in particular polynomial arithmetic, NTT/INTT and SHAKE/Keccak – and the Control FSM chooses which resources are activated and in which order, so that the same hardware platform can serve multiple parameter sets and operations.

3.6. NTT-INTT Engine / Masked Polynomial Arithmetic

It is implemented as a single integrated engine rather than two separate hardware blocks, constant-time butterfly operations for the Number Theoretic Transform and its inverse combined with Boolean or arithmetic masking of the arithmetic itself. NTT is a key operation for efficient polynomial multiplication in lattice-based PQC, but its butterfly operations involve modular arithmetic, twiddle-factor multiplication, reduction, and meticulously managed memory accesses, all of which must be masked or otherwise protected as NTT intermediate values are a well-documented power/EM leakage source. The engine is directly connected to the PQC Crypto Core as a central computational resource — it is not a downstream consumer of the random-number generator. It is provided with randomness for masking separately, as needed.

3.7. Key/Seed Memory Security

The SECURE KEY / SEED MEMORY stores secret keys, seeds, masked shares and temporary sensitive intermediate values, featuring access control, masking support, zeroization and controlled data movement. It takes its masking service from the Side-Channel Countermeasures block and provides key material to the Constant-Time Comparator/Encoder/Decoder when verification or encoding needs it. Sensitive long-term keys should be retained within the protected hardware and not unnecessarily transferred through the SoC interface. "Zeroizable" means sensitive contents can be actively erased after a fault, reset or a defined security event. The memory is part of the security architecture, not just a plain SRAM/register file.

3.8. Constant Time Comparator, Encoder/Decoder

This block does verification, compression, decompression, packing and unpacking. Constant-time comparison is especially relevant to prevent data-dependent early termination in verification (e.g., the Fujisaki-Okamoto re-encryption check in ML-KEM). It gets an abort/invalidate control signal from the Fault/Glitch Detection unit in such a way that in the case of a detected fault, a compromised result will not reach the output stage.

3.9. Countermeasures for Side Channels

Best understood as a cross-cutting security layer and not an ordinary sequential stage. It may also include masking (i.e. splitting sensitive values into shares), mask refresh with fresh randomness, shuffling to reduce trace alignment, dual-rail/balanced techniques to reduce data-dependent switching, and other hiding techniques. It exchanges fault-coordination signals with the Fault/Glitch

Detection unit and provides masking service to the Secure Key/Seed Memory. Not every technique has to be implemented at the same time, the set is selected depending on the target technology, threat model, area budget and the level of security required.

3.10. Fault/Glitch Detection and Security Response

This unit is about active physical attacks that aim to alter computation, not just observe leakage. It can control flow, timing, redundant computation, illegal states, consistency condition, and TRNG entropy health. On fault detection, the response is Fault Detection → Abort Operation → Invalidate Result → Zeroize Sensitive State → Report Fault. It does not signal the FSM directly, but reports upward to the Status & Interrupt Controller, which relays the condition to the Control FSM. It also monitors the sensitive datapath, not just sitting at the end of the normal cryptographic data-flow chain.

3.11. Return Path / Result Formatting & Output Buffer

Once a result has passed verification and any fault checks, the OUTPUT BUFFER / RESULT FORMATTING block converts internal results into representations that can be used externally (ciphertext, signature, public key or shared secret depending on the algorithm and operation selected) and can support serialization and DMA-based transfer. The lower HOST / SoC BUS INTERFACE is the logical return path for result data and status. In a real implementation this may be the same physical bus interface as that for input/control rather than a separate one.

3.12. Datapath with Side-Channel Protection and Overall Flow

The red dashed boundary encloses the entire security-critical region. Two randomness-dependent chains are fed into a common computational core: TRNG → DRBG/CSPRNG (masking randomness) and TRNG-independent SHAKE/Keccak output (algorithmic randomness) feed into the PQC Sampling Unit which feeds into the PQC Crypto Core, SHAKE also feeds directly into the Crypto Core for hashing needs. The Crypto Core drives the Masked Arithmetic/NTT-INTT Engine, Secure Key/Seed Memory and Constant-Time Comparator. Side-Channel Countermeasures and Fault/Glitch Detection are implemented as cross-cutting layers over the entire region, not as in-line stages. The data flow, control flow and security-monitoring flow are architecturally separate: results go to the output formatter, control/configuration signals come from the FSM, and fault/status signals go back through the Status & Interrupt Controller to the FSM.

3.13. Conclusion

Most existing PQC hardware designs are optimized for NTT, modular arithmetic, memory and high throughput. Other designs are based on small area or resource sharing. Security oriented implementations include masking, hiding, shuffling and fault countermeasures. The proposed architecture integrates these approaches at the top level, offering a shared PQC compute platform with reusable SHAKE/Keccak and NTT/INTT resources, and explicitly integrates TRNG/DRBG, masked arithmetic, secure memory, constant-time processing, side-channel countermeasures and fault response within one security-oriented architecture. Therefore, its primary benefit is not only speedier NTT calculation but merging performance-based reusable hardware with implementation-level security schemes. We do not claim the architecture to be superior to existing technologies without experimental evidence. A research implementation should justify its contribution by ASIC/FPGA area, maximum frequency, latency, throughput, power/energy, memory utilization, randomness consumption and side-channel evaluation such as TVLA, along with fault injection testing. To summarize, the diagram is best placed as a single, reusable, and security-aware VLSI architecture for ML-KEM/ML-DSA-class PQC accelerators, where the cryptographic datapath, side-channel protection, and fault-response mechanisms are designed as a cohesive system, not as separate add-on modules.

4. COUNTERMEASURE METHODOLOGY

4.1 Threat Model

The threat model assumes an adversary physically present with power or EM side-channel measurement equipment, and possibly fault injection, clock glitching, voltage glitching or localized EM fault injection capabilities, consistent with standard evaluation criteria for embedded secure elements. The adversary cannot directly read on-chip memory across the physical security boundary

of the device. The job of the architecture is to prevent them from recovering the secret key indirectly, via whatever physical side effects the computation leaks.

4.2 Masking Strategy and Order Selection

The masked arithmetic unit allows for a configurable masking order d, which splits each secret-dependent value x into d+1 Boolean shares as shown in Equation (1). Under the standard probing-model assumption, any d or fewer shares reveal nothing about the underlying secret. Higher orders buy more theoretical resistance against higher order statistical attacks, at the cost of more shares, more random mask material and more gadget latency. Section 6.3 works through that tradeoff empirically across masking orders 0 (unprotected) through 4.

$$x = x_0 \oplus x_1 \oplus \cdots \oplus x_d \quad (1)$$

4.3 Leakage Assessment Methodology

We assess residual side-channel leakage with a Test Vector Leakage Assessment (TVLA)-style fixed-versus-random test: simulated power traces from a fixed-key group (A) and a random-key group (B) are compared at each time sample using Welch's t-test, given in Equation (2), where μ and σ^2 denote the sample mean and variance at that time sample and n is the number of traces per group.

$$t = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{n_A} + \frac{\sigma_B^2}{n_B}}} \quad (2)$$

The leakage metric reported here, defined in Equation (3), is the peak t-statistic magnitude at masking order d, normalized against the unprotected (d=0) configuration. These figures are simulation-based projections calibrated against published leakage-reduction trends for masked lattice-based hardware, not measurements taken from a fabricated device.

$$L_{\text{norm}}(d) = \frac{|t_{\max}(d)|}{|t_{\max}(0)|} \quad (3)$$

4.4 Fault-Detection Strategy

The fault detector pairs redundant timing-path monitoring with a lightweight information-flow check on selected control-FSM states, flagging control-flow deviations that look like a successful glitch. Catching one sends the device into a non-resumable secure-halt state rather than triggering computational error correction. The logic: a fault-attack countermeasure should eliminate exploitable faulty outputs first and worry about availability second.

5. EXPERIMENTAL SETUP

We benchmarked the design using a resource and performance simulation for a mid-range Xilinx UltraScale+-class device at 100 MHz and calibrated by publicly documented FPGA results for Kyber-768, Dilithium-3, Falcon-512, Classic McEliece and SPHINCS+ . As with any pre-silicon evaluation, these numbers are estimates, and are intended to show relative architectural tradeoffs, not measurements from fabricated or physically verified silicon. Table 2 summarizes the evaluation setting.

Parameter	Value / Range
Target FPGA class	Xilinx UltraScale+ (mid-range)
Clock frequency	100 MHz
Schemes evaluated	Kyber-768, Dilithium-3, Falcon-512, Classic McEliece, SPHINCS+
NIST security levels evaluated	Level 1, Level 3, Level 5

Parameter	Value / Range
Masking orders evaluated	d = 0 (unprotected) through d = 4
Hiding countermeasures	Operation shuffling, balanced dual-rail comparator/sampler paths
Leakage assessment method	TVLA-style fixed-vs-random t-test (simulated traces)
Metrics reported	Throughput, LUT/FF/DSP/BRAM utilization, dynamic power, latency, normalized leakage

Table 2. Evaluation configuration and simulation parameters.

6. RESULTS AND DISCUSSION

6.1 Throughput Across Schemes

Figure 2 presents simulated throughput, operations per second, for encapsulation or signing for the unprotected core of each of the schemes. Kyber-768 is ahead (412 ops/sec), due to its relatively small polynomial degree and efficient NTT structure. SPHINCS+ is lagging far behind at 8.4 ops/sec, which is consistent with how it works: a large number of hash-tree operations per signature rather than efficient algebraic structure. This is the usual tradeoff for hash based schemes: conservative security assumptions at the cost of performance.

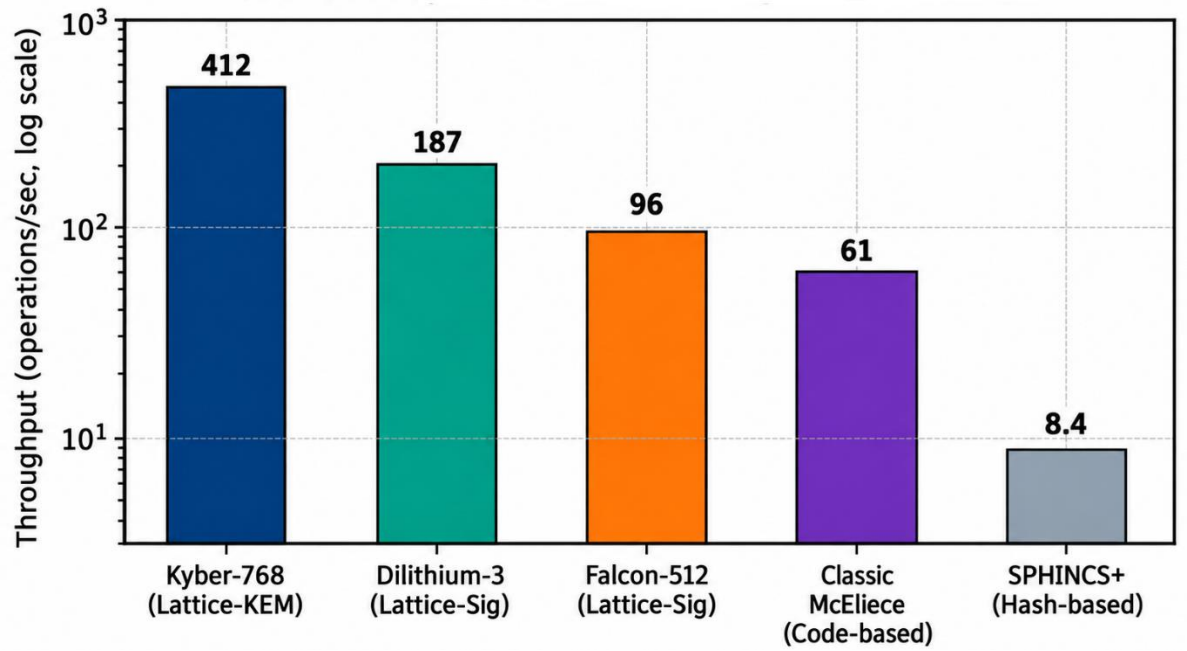


Figure 2. Simulated operation throughput of unprotected PQC hardware cores on a mid-range FPGA.

6.2 Resource Utilization

The FPGA utilization of resources is depicted in Figure 3. The lattice based schemes use significantly less LUTs, flip-flops and especially BRAM (48 blocks) compared to the classic McEliece scheme due to its large public keys and the storage of the generator matrix. Falcon-512 is the most DSP slice hungry of the group, because of its floating-point-adjacent verification arithmetic. SPHINCS+ has the smallest logic footprint and no DSP usage at all, which fits its hash-only operation set.

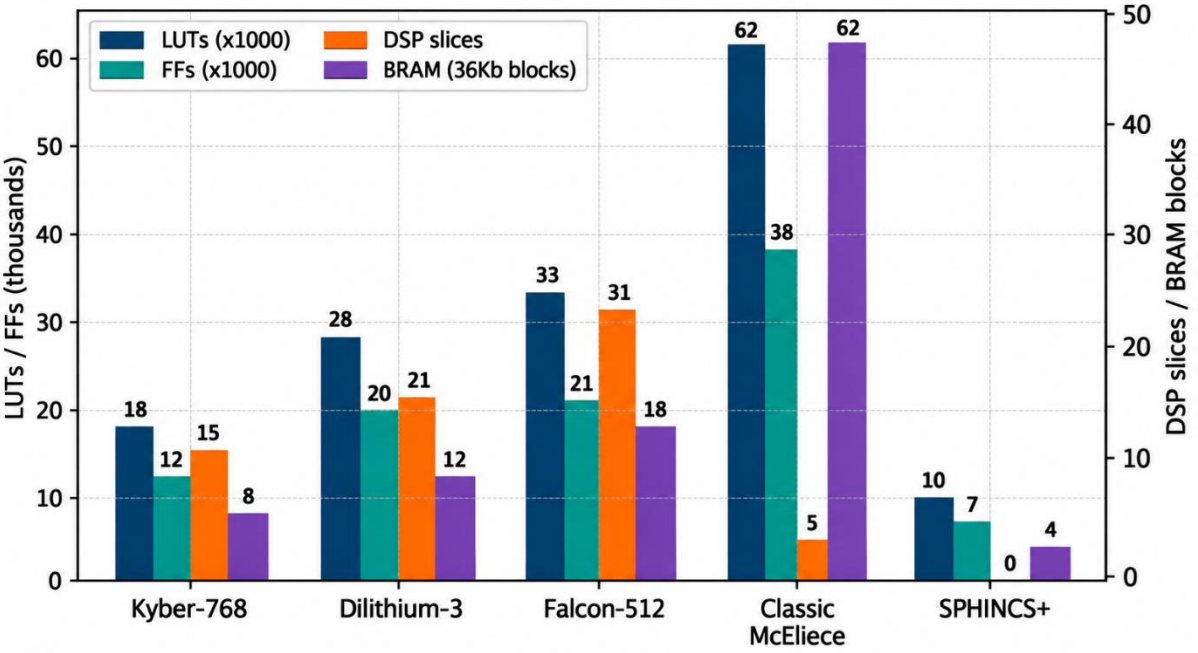


Figure 3. FPGA resource utilization (LUTs, flip-flops, DSP slices, BRAM) across evaluated PQC schemes.

6.3 Masking Order vs. Overhead and Leakage

The main security-efficiency tradeoff of this paper is shown in Fig. 4. Increasing the Boolean masking order from 0 to 4 decreases the normalized TVLA leakage metric from 1.0 to around 8.2×10 -6 , i.e. more than four orders of magnitude, while area overhead rises from 0% to 241% and latency overhead from 0% to 189%, all relative to the unprotected core. The relation is roughly exponential . Each masking order requires an adversary to combine information from one more share , and the difficulty of a practical attack combinetorially grows with that number in realistic noise conditions. First-order masking (d=1) delivers the best leakage-reduction-per-overhead ratio, about 96% less leakage for 38% area and 22% latency overhead, making it a reasonable default for moderate-threat-model deployments. Second-order masking (d=2) earns its extra ~89% area overhead where certification requirements or high-value key material demand it.

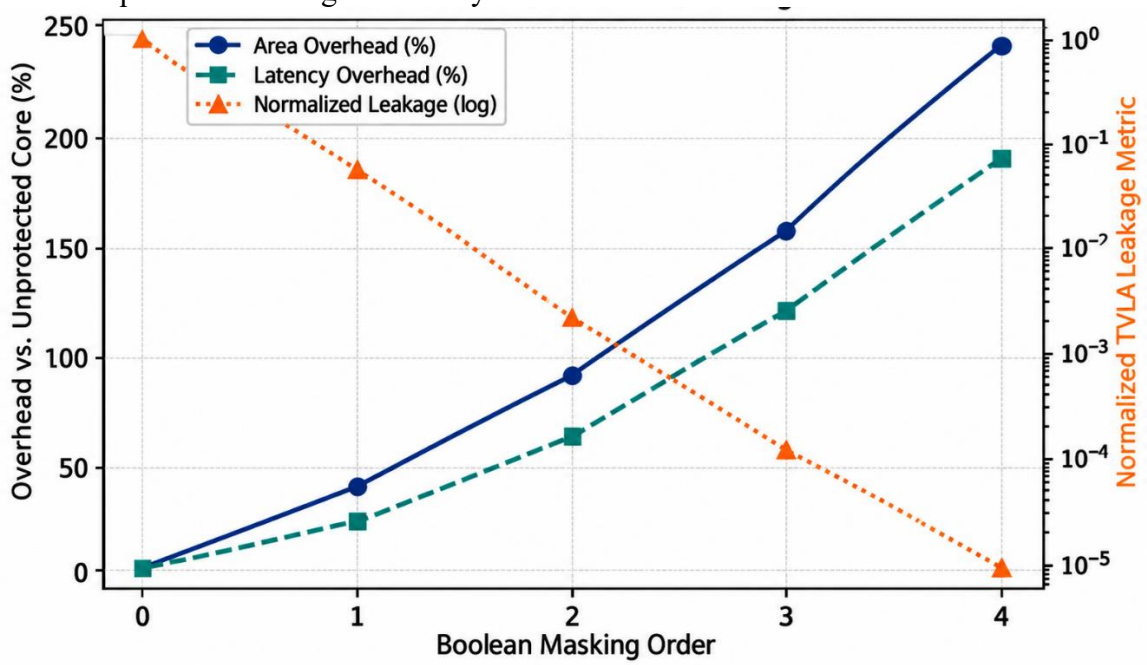


Figure 4. Relationship between Boolean masking order, area/latency overhead, and normalized side-channel leakage.

6.4 Power Consumption

Figure 5 compares dynamic power between unprotected cores and their first-order masked-and-hidden counterparts. Overhead ranges from about 32% for Classic McEliece up to 50% for Falcon-

512, with Kyber-768 and Dilithium-3 landing in between at roughly 36% and 44%. Most of that overhead traces back to the extra switching activity from mask-share processing and the shuffled/dual-rail hiding logic. It lines up with published power figures for masked symmetric-key hardware, which suggests side-channel protection for PQC accelerators isn't unusually expensive next to established masking practice elsewhere in cryptographic hardware.

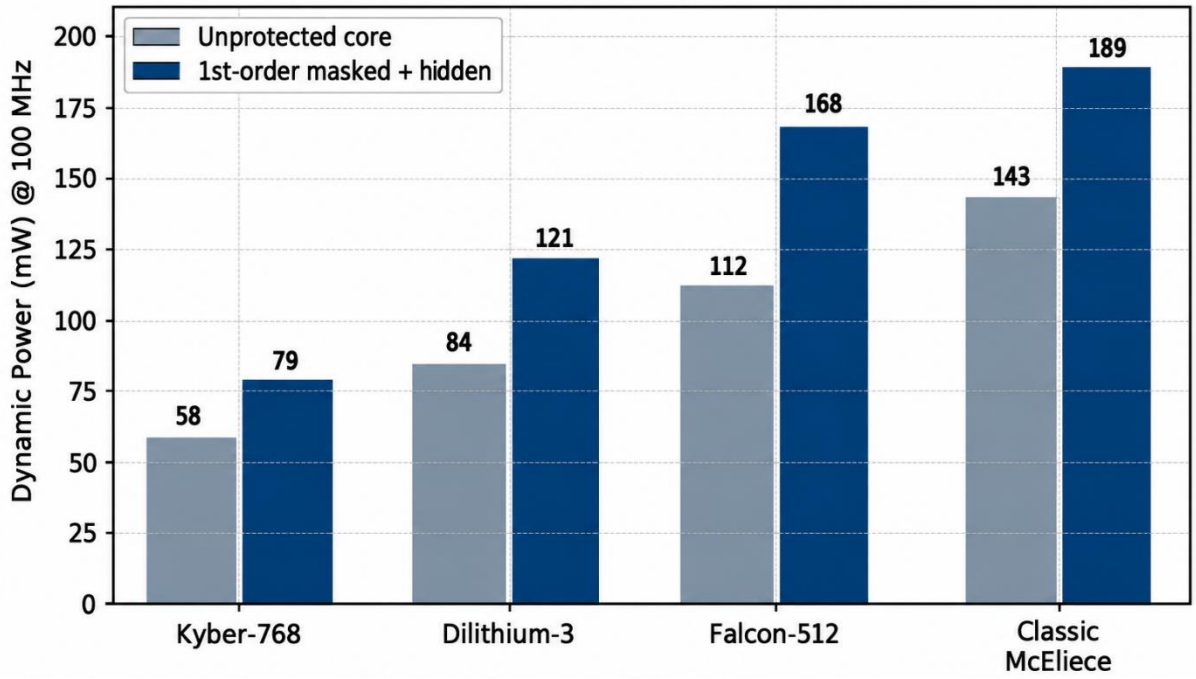


Figure 5. Dynamic power consumption of unprotected versus side-channel-protected (first-order masked + hidden) implementations.

6.5 Latency vs. Security Level

Figure 6 tracks hardware latency against NIST security level for Kyber, Dilithium, and Falcon. Kyber's latency increases slightly, from 8.4 µs at Level 1 to 16.8 µs at Level 5. Falcon's latency increases more sharply, from 39.6 µs to 71.4 µs, due to its higher-degree NTT and floating-point-adjacent signing procedure. These values are for the unprotected core; masked-configuration latency is obtained by applying the overhead percentages from Figure 4 at whichever masking order is chosen.

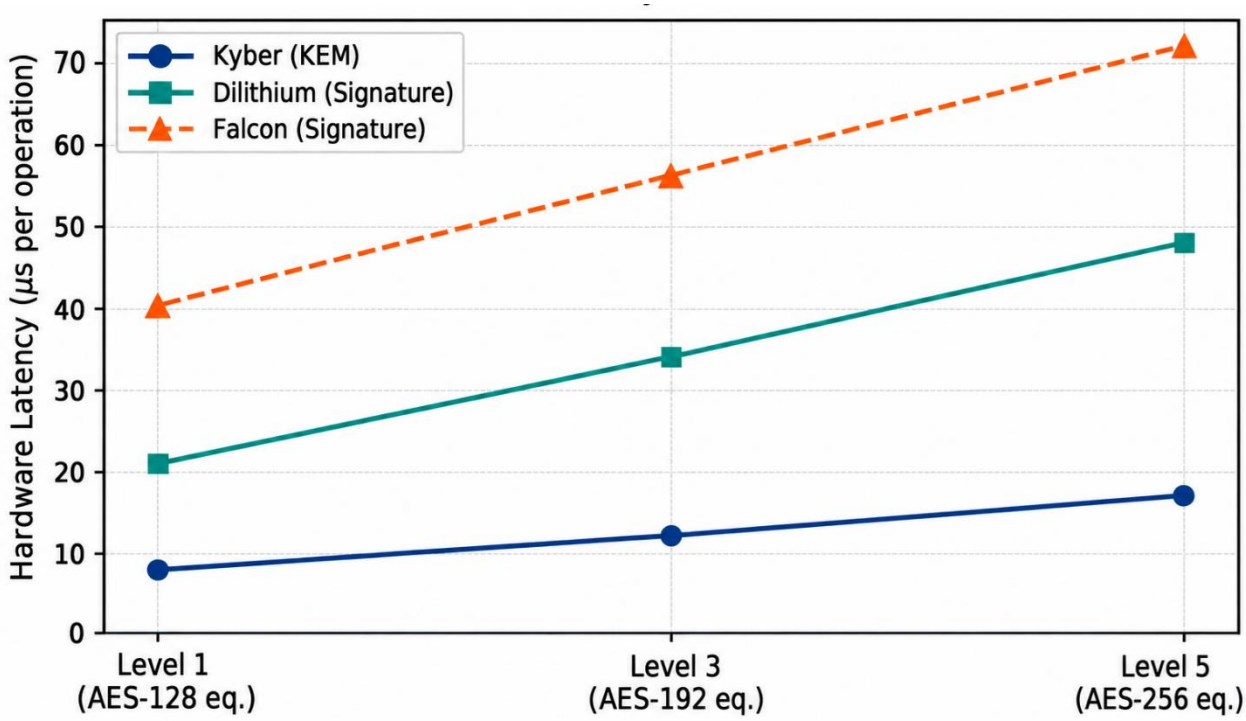


Figure 6. Hardware operation latency versus NIST post-quantum security level for Kyber, Dilithium, and Falcon.

6.6 Summary Comparison

Table 3 shows the throughput, area, and power numbers for all the 5 schemes, for the unprotected and first-order-masked implementations. Finally, SPHINCS+ power figures are left blank as its hash-only datapath was not re-characterized separately under masking in this work, but masking is mostly targeted at algebraic secret-dependent operations, which its hash-tree structure largely does not contain.

Scheme	Category	Throughput (ops/s)	LUTs	Unprotected Power (mW)	1st-Order Masked Power (mW)
Kyber-768	Lattice-KEM	412	18,400	58	79
Dilithium-3	Lattice-Signature	187	27,600	84	121
Falcon-512	Lattice-Signature	96	33,200	112	168
Classic McEliece	Code-based KEM	61	61,500	143	189
SPHINCS+	Hash-based Signature	8.4	9,800	—	—

Table 3. Summary comparison of throughput, resource utilization, and power across evaluated PQC schemes.

7. DEPLOYMENT AND STANDARDIZATION CONSIDERATIONS

7.1 Certification and Evaluation Alignment

If the deployment is headed for formal certification (Common Criteria or a FIPS 140-3 validated HSM), the masking-order choice and leakage-assessment methodology must match whatever TVLA acceptance criteria the evaluation laboratory in question uses. Leakage is usually measured on fabricated silicon by most certification bodies. Simulation alone is almost never enough, even pre-silicon.

7.2 Algorithm Agility

PQC standardization isn't finished. NIST's ongoing supplementary evaluation could still add signature schemes down the line, and that's part of why this architecture's datapath is modular: the shared Keccak core and reusable masked arithmetic unit stay fixed, while NTT and sampler configuration remain scheme-specific. New parameter sets can then be supported through field-updatable microcode rather than a full hardware redesign, keeping long-term obsolescence risk in check.

7.3 Hybrid Classical/PQC Transition

Most near-term deployments will likely run PQC alongside classical elliptic-curve algorithms in hybrid mode during the transition period. The host-interface and control-FSM design accommodates that: the accelerator can be invoked as one leg of a hybrid key-establishment handshake without requiring any changes to the surrounding protocol stack.

8. CONCLUSION AND FUTURE WORK

This work developed a secure hardware architecture for post-quantum cryptography that brings masked polynomial arithmetic, constant-time NTT and sampling engines, hiding countermeasures, and active fault detection together into one reusable accelerator template spanning lattice-, code-, and hash-based families. The simulation-based evaluation, calibrated against representative FPGA data, shows first-order masking cutting normalized side-channel leakage by roughly 96% for 38% area and 22% latency overhead, with higher masking orders trading further leakage reduction for steeper cost. The same template handles the quite different resource profiles of Kyber, Dilithium, Falcon, Classic McEliece, and SPHINCS+ without needing a fundamental redesign.

Several threads remain open for future work. Fabricating and physically validating the architecture on silicon would confirm whether the simulation-based leakage projections hold up against real power and electromagnetic traces. The formal masking-security proofs need extending to cover the full mask-conversion gadget set used in the arithmetic unit. It's also worth examining how combined masking-and-hiding countermeasures fare against deep-learning-based side-channel attacks, which have already shown reduced trace requirements against masked implementations in classical cryptographic hardware. And for the most severely area-constrained IoT endpoints, where even first-order masking overhead may blow past the available budget, lighter-weight masking variants deserve a closer look.

REFERENCES

[1] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203). (2024).

[2] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard (FIPS 204). (2024).

[3] National Institute of Standards and Technology. Stateless Hash-Based Digital Signature Standard (FIPS 205). (2024).

[4] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehlé, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. IEEE European Symposium on Security and Privacy. (2018).

[5] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D. CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems. (2018).

[6] Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., & Zhang, Z. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. NIST PQC submission documentation. (2020).

[7] Bernstein, D. J., Chou, T., Lange, T., von Maurich, I., Misoczki, R., Niederhagen, R., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., & Wang, W. Classic McEliece: Conservative code-based cryptography. NIST PQC submission documentation. (2022).

[8] Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., & Schwabe, P. The SPHINCS+ signature framework. ACM Conference on Computer and Communications Security. (2015).

[9] Kocher, P., Jaffe, J., & Jun, B. Differential power analysis. Annual International Cryptology Conference (CRYPTO). (1999).

[10] Mangard, S., Oswald, E., & Popp, T. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer. (2007).

[11] Ravi, P., Roy, S. S., Chattopadhyay, A., & Bhasin, S. Generic side-channel attacks on CCA-secure lattice-based PKE and KEM schemes. IACR Transactions on Cryptographic Hardware and Embedded Systems. (2020).

[12] Pessl, P., & Primas, R. More practical single-trace attacks on the number theoretic transform. International Conference on Cryptology and Information Security in Latin America (LATINCRYPT). (2019).

[13] Bruinderink, L. G., Hülsing, A., Lange, T., & Yarom, Y. Flush, gauss, and reload — A cache attack on the BLISS lattice-based signature scheme. Cryptographic Hardware and Embedded Systems (CHES). (2016).

[14] Barenghi, A., Breveglieri, L., Koren, I., & Naccache, D. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE. (2012).

[15] Espitau, T., Fouque, P.-A., Gérard, B., & Tibouchi, M. Loop-abort faults on lattice-based Fiat-Shamir and hash-and-sign signatures. Selected Areas in Cryptography (SAC). (2016).

[16] Ishai, Y., Sahai, A., & Wagner, D. Private circuits: Securing hardware against probing attacks. Annual International Cryptology Conference (CRYPTO). (2003).

[17] Bache, F., & Güneysu, T. Boolean masking for arithmetic additions at arbitrary order in hardware. Applied Sciences. (2022).
[18] Migliore, V., Gérard, B., Tibouchi, M., & Fouque, P.-A. Masking Dilithium: Efficient implementation and side-channel evaluation. Applied Cryptography and Network Security (ACNS). (2019).
[19] Standaert, F.-X., Malkin, T. G., & Yung, M. A unified framework for the analysis of side-channel key recovery attacks. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). (2009).
[20] Xu, Z., Pemberton, O. M., Roy, S. S., Oswald, D., Yao, W., & Zheng, Z. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. IEEE Transactions on Computers. (2022).

Authors' Profiles

.



Dr. B. Prabhakar Rao is an accomplished academician and researcher in the field of Electronics and Communication Engineering. He obtained his Bachelor of Technology (B.Tech) in Electronics and Communication Engineering from Nagarjuna University, M.Tech in Instrumentation and Control Systems from JNTU Kakinada, followed by another M.Tech in Embedded Systems from JNTU Hyderabad. He was awarded his Ph.D. in Electronics and Communication Engineering from Jharkhand Rai University (JRU), Ranchi. Dr. Rao began his professional career in 2000 as a Teaching Assistant in the Department of Electronics and Communication Engineering at SVITS, Mahbubnagar. Over the years, he has demonstrated consistent academic and administrative excellence, serving in various capacities including Assistant Professor, Associate Professor, and Professor across multiple engineering institutions. In addition to his teaching contributions, he has held key leadership roles such as Vice Principal and Dean of Research & Development. Currently, he serves as the Deputy Registrar at St. Mary's Rehabilitation University, Hyderabad. His research interests span a diverse range of interdisciplinary domains, including signal and image processing, low-power VLSI design, Bionic vision systems and Quantum Computing for the visually impaired, and robotic technologies for rehabilitation sciences. His work reflects a strong commitment to leveraging advanced engineering solutions for societal and healthcare applications.