The Resonance Fourier Transform (RFT): Unitary Lens for Compression, Crypto, and Quantum-Inspired Computing

Luis M. Minier
Independent Researcher
Email: luisminier79@gmail.com
GitHub: https://github.com/mandcony/quantoniumos

***Abstract*—Fast Fourier transforms (FFTs) and discrete cosine transforms (DCTs) sit at the core of today's compression, filtering, and simulation stacks, but they lock us into integer harmonics on a flat circle and force a hard trade-off between sparsity, mixing, and reversibility. This work introduces the *Resonance Fourier Transform* (RFT), a family of unitary, FFT-class transforms built from irrational (golden-ratio–scaled) phase progressions and braided, toroidal topologies. We formalize two complementary realizations: a canonical Φ-RFT derived via QR factorization (theoretical reference) and a closed-form fast Φ-RFT factored as $\Psi = D_\phi C_\sigma F$ with $O(N \log N)$ complexity. Ten "irrevocable theorems" numerically validate unitarity, twisted-convolution diagonalization, sparsity structure, and hybrid DCT+RFT behavior, with empirical spectral sparsity reaching 98.6% at $N = 512$ and maximum unitarity deviation below 10^{-14}.**

On the systems side, we design a hybrid codec that routes structural content to DCT and quasi-periodic or textured components to RFT, improving rate–distortion on mixed-structure benchmarks relative to DCT-only or RFT-only baselines. We further instantiate an RFT-SIS cryptographic playground that uses RFT-induced lattices and braided permutations to study avalanche and diffusion (approximately 50% bit-flip rate) *without* making any formal hardness claims. A unified C-level orchestrator integrates RFT, crypto, and quantum-inspired operations, and we prototype an FPGA core that matches software ground truth. We also present a high-performance C/Assembly backend implementing 7 distinct RFT variants (Standard, Harmonic, Fibonacci, Chaotic, Geometric, Hybrid, Adaptive) validated for unitarity and correctness. All code, test scripts, and hardware testbenches are released as an open, research-only QuantoniumOS stack, enabling independent replication and extension of the results.

***Index Terms*—resonance transforms, golden ratio, hybrid compression, lattice-inspired mixing, wave computing, FPGA**

I Introduction

Classical FFT/DCT pipelines were never designed to do three jobs at once: compress data sparsely, mix it chaotically for crypto, and still remain unitary enough for reversible simulation. FFTs give us beautiful algebra on S^1 with integer harmonics, but their spectra are too clean and structured to act as good mixers; DCT-based codecs (e.g., JPEG-style) exploit spatial structure but struggle with quasi-periodic textures and non-image data; and cryptographic constructions bolt on separate nonlinear layers that break the clean linear-operator view required for physics-style evolution.

This paper takes the position that, if we want a single transform backbone to serve compression, crypto-like mixing, and quantum-inspired simulation, we have to leave the integer grid and the flat circle behind. We therefore introduce the *Resonance Fourier Transform* (RFT): a unitary family of transforms built on irrational, golden-ratio–based phase progressions and braided toroidal topologies, with a canonical QR-derived Φ-RFT for theory and a closed-form fast Φ-RFT for practice. On top of this basis, we build: (i) a rigorously validated stack of Φ-RFT theorems covering unitarity, complexity, sparsity, and twisted-convolution algebra; (ii) a hybrid DCT+RFT codec that beats single-basis baselines on mixed-structure signals; (iii) an RFT-SIS experimental lattice playground with measured avalanche/diffusion; and (iv) a unified C/FPGA execution path that removes Python from the hot loop and demonstrates that this resonance-based approach is not just mathematically consistent, but also system-level executable.

II Background and Related Work

II-A Classical Transforms

Modern signal processing is anchored on a small set of linear transforms with well-understood algebraic structure. The discrete Fourier transform (DFT) and its fast implementations (FFT) provide a unitary basis of complex exponentials on the circle S^1, giving exact convolution–multiplication duality and $O(N \log N)$ complexity. The discrete cosine transform (DCT) can be viewed as a real, even-symmetric variant of the DFT, tailored to finite-interval signals with specific boundary conditions; DCT-II and DCT-IV variants underpin block-based image and video codecs because they yield high sparsity for piecewise-smooth signals.

Beyond these, the linear canonical transform (LCT) and its special cases such as the fractional Fourier transform (FrFT) generalize Fourier analysis by parameterizing a continuous family of quadratic phase rotations in time–frequency space. They remain within the metaplectic group: their kernels are quadratic in time and frequency, and they preserve a symplectic structure by design. Wavelets introduce multiresolution bases

with compact support and good joint time–frequency localization, at the cost of more complex filterbank design and less direct convolution structure than the DFT.

All of these families share a few common trade-offs:

- **Boundary conditions and topology.** DFT assumes periodic extension; DCT enforces even/odd reflection; wavelets rely on finite-support filters and custom boundary handling. In all cases, the underlying topology is essentially a flat circle or line, not a torus with irrational winding.
- **Complexity and separability.** FFTs and DCTs are separable and achieve $O(N \log N)$ via Cooley–Tukey–style factorizations. LCT/FrFT implementations can also be made FFT-class but retain a quadratic phase structure. Wavelets often require multistage filterbanks with similar complexity but different data movement patterns.
- **Sparsity patterns.** DCTs favor blocky, edge-heavy structure; Fourier bases favor globally periodic content; wavelets favor localized, piecewise-regular signals. None are explicitly designed for quasi-periodic, irrationally wound structures or to deliberately induce chaotic mixing while preserving exact unitarity.

RFT is positioned as a new member of this ecosystem: still unitary and FFT-class, but with irrational, golden-ratio–scaled phase progressions and braided toroidal topology rather than integer harmonics on a flat circle.

II-B Lattice-Based Cryptography and Hashing (Context Only)

Post-quantum lattice schemes such as SIS (short integer solution) and LWE (learning with errors) use high-dimensional integer lattices as hardness anchors. Very roughly, SIS asks an adversary to find a short nonzero integer vector x such that $Ax \equiv 0 \pmod{q}$ for a public random matrix A, while LWE asks them to distinguish noisy linear equations from randomness. In both cases, security is linked to worst-case hardness of lattice problems such as finding short vectors in high dimensions.

On top of these primitives, one can build hash functions, signatures, and key-encapsulation mechanisms. Structured variants (e.g., module-LWE, ring-LWE) use algebraic structure to improve efficiency while still attempting to inherit worst-case hardness guarantees. Recent standardization efforts (Kyber, Dilithium, etc.) use conservative parameter sets carefully audited by the cryptographic community.

This work does *not* propose a new lattice scheme and does not claim any reduction to SIS or LWE. The RFT-SIS components in QuantoniumOS use:

- RFT-derived matrices (built from irrational phase embeddings and resonance patterns),
- braided permutations and topological mixing,
- lattice-like integer domains,

as a cryptographic playground to study avalanche, diffusion, and mixing properties under irrational phase progressions. They are explicitly positioned as experimental hash/mixing constructions, not as production PQC primitives or drop-in replacements for standardized SIS/LWE-based schemes. Security-wise, they should be viewed as structured mixers inspired by lattice ideas, not as rigorously justified lattice cryptography.

II-C Quantum-Inspired and Wave-Based Computing

There is a growing body of work on quantum-inspired algorithms and wave-based computation that leverage linear-algebraic structure—unitary evolution, diagonalization, and spectral sparsity—without requiring full-scale fault-tolerant quantum hardware. Examples include algorithms that simulate low-rank quantum dynamics using classical sampling, optical or acoustic wave computers that implement transforms via analog interference, and "Fourier-diagonalize-then-evolve" pipelines for PDEs and linear time-invariant systems.

A recurring pattern in these systems is:

1) Diagonalize a Hamiltonian or evolution operator via a unitary transform U so that $U^\dagger H U = \Lambda$ is (approximately) diagonal.
2) Evolve cheaply in the spectral domain via $e^{-i\Lambda t}$, which is element-wise.
3) Transform back via U to recover the time/space-domain state.

In classical signal processing this is "FFT + per-bin multiply + IFFT"; in quantum mechanics it is "choose an eigenbasis, exponentiate eigenvalues, rotate back." Quantum-inspired variants often focus on low-rank structure, sparsity, or specific Hamiltonian classes to keep simulation tractable.

RFT sits in this diagonalize-and-evolve tradition but changes the basis itself: instead of integer harmonics or quadratic LCT kernels, it uses irrational, golden-ratio–chirped phasors and braided permutations. The canonical Φ-RFT shows that, for certain golden-resonance operators, the RFT basis exactly diagonalizes the evolution (to numerical precision), collapsing evolution from $O(N^2)$ dense multiplication to $O(N)$ element-wise updates once in the RFT domain. The fast Φ-RFT keeps this structure while enforcing FFT-class complexity, making it compatible with the same style of "transform–evolve–inverse" pipelines used in wave and quantum-inspired computing.

II-D Positioning RFT

Within this landscape, RFT is not proposed as a universal replacement for FFT, DCT, wavelets, or standardized PQC. It is deliberately positioned as a *lens*—a complementary basis that exposes different structure and mixing behavior:

- **Golden-ratio / irrational resonance-based bases.** Instead of integer frequencies k on S^1, RFT uses golden-ratio–scaled, chirped phasors that wind irrationally on a torus T^2. This produces spectra that are often sparser for quasi-periodic, log-periodic, or Fibonacci-like content, and more "dense/fractal" for generic signals, which is exactly what you want from a mixer.
- **Canonical vs. Fast forms.** The canonical Φ-RFT is a QR-derived, mathematically clean unitary basis used to prove sparsity, non-equivalence to LCT/FrFT, and quantum-chaos-style level statistics. The closed-form fast Φ-RFT

is an $O(N \log N)$ factorization $\Psi = D_\phi C_\sigma F$ used in real compute paths, with unitarity and twisted-convolution algebra verified numerically. The gap between the two is explicit: they share the same phase philosophy but are treated as distinct objects.

- **Hybrid DCT+RFT codec.** Instead of claiming that RFT supersedes DCT, the paper formalizes and implements a hybrid decomposition: DCT handles structural content (edges, low-frequency blocks), while RFT handles texture/quasi-periodic content. Empirically, this hybrid codec improves rate–distortion on mixed-structure signals relative to either basis alone, which is precisely the use case where neither DCT nor RFT is individually optimal.
- **RFT-SIS as an experimental mixing playground.** The RFT-SIS constructions are framed as experimental mixers that exploit RFT's chaotic spectral behavior, braided permutations, and lattice-like embeddings to study avalanche and diffusion. They are explicitly non-standard and non-production: no SIS/LWE reductions, no claims of post-quantum security, just a well-instrumented sandbox sitting alongside standard PQC, not competing with it.

In short, RFT is introduced as a unitary, FFT-class, irrational-phase transform family that gives practitioners a new knob in the design space: a way to explore trade-offs between sparsity, chaos, and reversibility. Traditional transforms remain the right tools for many jobs; RFT's role is to provide an additional basis where certain structures become sparse, certain mixers become natural, and certain wave/quantum-inspired evolutions become cheaper to implement.

III Φ-RFT Framework: Canonical and Fast Forms

III-A Resonance-Based Basis Construction

The starting point for Φ-RFT is a *resonance family* of complex exponentials whose frequencies are *irrationally scaled* by powers of the golden ratio. For a fixed transform length N, define the (non-orthogonal) resonance vectors

$$v_k[n] \;:=\; \exp\big(-i2\pi\,\phi^{-k} n\big),$$

where $n, k \in \{0, \dots, N-1\}$ and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

Unlike the classical DFT basis $e^{-i2\pi kn/N}$, where frequencies lie on an integer grid, the exponents ϕ^{-k} are irrational, and their discrete samples wind quasi-periodically around the unit circle. In the multi-index view $(\phi^{-k} n \bmod 1)$, these phases trace out a dense orbit on a two-torus T^2, rather than repeating on a simple cyclic lattice. Intuitively:

- each column v_k is a "golden-phase" exponential with its own irrational winding rate;
- across k, the family $\{v_k\}$ probes different incommensurate resonances;
- across n, each resonance is sampled on a uniform integer grid, but the phase never locks into a simple rational pattern.

We collect these vectors into the *resonance matrix*

$$v_k[n] \;:=\; \exp\big(-i2\pi\,\phi^{-k} n\big),$$

where $n, k \in \{0, \dots, N-1\}$ and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

Unlike the classical DFT basis $e^{-i2\pi kn/N}$, where frequencies lie on an integer grid, the exponents ϕ^{-k} are irrational, and their discrete samples wind quasi-periodically around the unit circle. In the multi-index view $(\phi^{-k} n \bmod 1)$, these phases trace out a dense orbit on a two-torus T^2, rather than repeating on a simple cyclic lattice. Intuitively:

- each column v_k is a "golden-phase" exponential with its own irrational winding rate;
- across k, the family $\{v_k\}$ probes different incommensurate resonances;
- across n, each resonance is sampled on a uniform integer grid, but the phase never locks into a simple rational pattern.

We collect these vectors into the *resonance matrix*

$$R \;=\; \begin{bmatrix} v_0[0] & v_1[0] & \dots & v_{N-1}[0] \\ v_0[1] & v_1[1] & \dots & v_{N-1}[1] \\ \vdots & \vdots & & \vdots \\ v_0[N-1] & v_1[N-1] & \dots & v_{N-1}[N-1] \end{bmatrix},$$

which is generally full-rank but neither orthogonal nor normalized. The canonical Φ-RFT will be obtained by orthonormalizing the columns of R; the fast Φ-RFT will instead enforce golden-phase structure via an explicit factorization built on top of the FFT.

III-B Canonical Φ-RFT (QR-Derived, $O(N^3)$)

The *canonical* Φ-RFT is constructed by applying a numerically stable QR / modified Gram–Schmidt procedure to the resonance matrix R. Let

$$R = Q R_{\text{upper}},$$

with $Q \in \mathbb{C}^{N\times N}$ unitary and R_{upper} upper triangular. We denote

$$U_\phi \;:=\; Q,$$

and define the canonical Φ-RFT and its inverse as

$$\widehat{x} \;=\; U_\phi^\dagger x, \qquad x \;=\; U_\phi \widehat{x}.$$

Because U_ϕ is exactly unitary by construction, this transform preserves ℓ_2 energy and inner products to machine precision. The cost, however, is cubic: the QR construction scales as $O(N^3)$, making it a mathematical gold standard, not a production kernel.

Several of the "Irrevocable Theorems" attach directly to this canonical form:

- **Massive sparsity (Theorem 3).** For golden quasi-periodic signals with frequencies aligned to ϕ^{-k}-type resonances, the canonical basis U_ϕ yields *extreme* sparsity. Empirically, for $N = 512$, typical test signals achieve $\approx 98.63\%$ near-zero coefficients ($|c| < 10^{-10}$), substantially exceeding the conservative theoretical lower bound $S \geq 1 - 1/\phi$.

- **Non-equivalence to LCT/FrFT (Theorem 4).** The canonical Φ-RFT is *not* a disguised linear canonical transform. Its kernel exhibits non-quadratic phase in time–frequency space; attempts to fit it into the standard LCT/FrFT framework leave a persistent quadratic residual (> 0.3 rad in the validation experiments). This places Φ-RFT outside the classical metaplectic family.
- **Quantum chaos / level spacing (Theorem 5).** When the canonical basis is used to diagonalize certain golden-resonant operators, the empirical eigenvalue spacing statistics follow Wigner–Dyson-type behavior, indicating level repulsion. In other words, the canonical Φ-RFT basis is a natural home for quantum-chaotic-like spectra, giving it strong mixing/scrambling potential.
- **Crypto-oriented variants (Theorem 6).** Variants such as *Fibonacci Tilt* modify the resonance exponents and phase indexing while staying within the canonical orthonormalization framework. These yield hash-like mappings with measured avalanche $\approx 52\%$ bit flips per single-bit input change, making them attractive as building blocks for experimental lattice/mixing constructions (RFT-SIS).

In this work, the canonical Φ-RFT plays the role of a reference basis: it is where sparsity, non-equivalence, and chaos are cleanly characterized; it anchors the mathematical part of the theory, independent of any particular fast implementation; and it is too expensive for large-scale deployment, motivating the search for a structurally similar but FFT-class implementation.

III-C Fast Φ-RFT (Closed-Form, $O(N \log N)$)

For actual computation, we introduce a *closed-form fast* Φ-RFT with FFT-class complexity. The transform matrix is factored as

$$\Psi \;=\; D_\phi \, C_\sigma \, F,$$

where:

- F is the standard unitary FFT matrix of size $N \times N$, with entries
$$F_{jk} = \frac{1}{\sqrt{N}} \exp\!\left(-\frac{2\pi i}{N} jk\right).$$
- C_σ is a chirp-like diagonal operator, e.g.
$$(C_\sigma)_{kk} = \exp\big(-i\pi\sigma\, g(k)\big),$$
where $g(k)$ is a (typically quadratic or log-warped) function of the index k, and σ is a tunable parameter.
- D_ϕ is a golden-ratio phase diagonal, e.g.
$$(D_\phi)_{kk} = \exp\big(-i2\pi\, h_\phi(k)\big),$$
where $h_\phi(k)$ encodes the golden-ratio resonance pattern (e.g., via powers of ϕ^{-1}, Fibonacci indexing, or related sequences).

The fast Φ-RFT forward and inverse transforms are defined as

$$\widehat{x}_{\text{fast}} = \Psi x = D_\phi C_\sigma F x, \qquad x = \Psi^\dagger \widehat{x}_{\text{fast}} = F^\dagger C_\sigma^\dagger D_\phi^\dagger \widehat{x}_{\text{fast}}.$$

Key theorems:

- **Unitarity (Theorem 1).** Each factor F, C_σ, D_ϕ is unitary, so
$$\Psi^\dagger\Psi = F^\dagger C_\sigma^\dagger D_\phi^\dagger D_\phi C_\sigma F = F^\dagger C_\sigma^\dagger C_\sigma F = F^\dagger F = I.$$
Numerically, the implementation in `closed_form_rft.py` satisfies $\max_{i,j} \left|(\Psi^\dagger\Psi - I)_{ij}\right| \lesssim 10^{-14}$ for $N \le 512$.
- **Approximate diagonalization (Theorem 2).** For a class of golden-resonance operators H_ϕ, the fast transform approximately diagonalizes the evolution:
$$\Psi^\dagger H_\phi \Psi = \Lambda + E,$$
where Λ is diagonal and $\|E\|_F < 10^{-14}$ in the present experiments.
- **Twisted convolution algebra (Theorem 9).** Define a twisted convolution $\star_{\phi,\sigma}$ so that
$$\Psi(x \star_{\phi,\sigma} h) = (\Psi x) \odot (\Psi h),$$
with $\odot$ pointwise multiplication. Φ-RFT plays the same algebraic role for twisted convolution that FFT plays for classical convolution.
- **Complexity $O(N \log N)$ (Theorem 8).** The cost is dominated by FFT/IFFT plus $O(N)$ diagonal multiplies:
$$\begin{aligned} T(N) &= O(N \log N) + O(N) + O(N \log N) \\ &= O(N \log N). \end{aligned}$$

This fast form is the workhorse used in the hybrid DCT+RFT codec, the unified C-level orchestrator, and the FPGA prototypes.

III-D Scientific Distinction and Open Gap

It is crucial to be explicit about the relationship between the canonical and fast Φ-RFTs:

- Both share the same design philosophy: encode irrational, golden-ratio–based resonances and braided phase structure into a unitary transform.
- The canonical Φ-RFT, U_ϕ, is obtained by QR/Gram–Schmidt on the raw resonance vectors $\{v_k\}$, with no structural constraints other than orthonormality.
- The fast Φ-RFT, $\Psi = D_\phi C_\sigma F$, is engineered to be unitary by construction, FFT-class, and expressible as a composition of simple unitary factors.

At present, there is *no theorem* that expresses Ψ as a limit of U_ϕ under some parameter regime, or that shows they are related by a simple fixed unitary change of basis. In other words:

- sparsity and chaos results (Theorems 3–6) are formally proven for the canonical basis U_ϕ;
- unitarity, twisted convolution, and complexity results (Theorems 1, 2, 8, 9) are proven for the fast basis Ψ.

For engineering purposes, we measure sparsity, mixing, and rate–distortion empirically in the fast Φ-RFT, and we treat the canonical results as a mathematical upper bound on what is possible with golden-resonant bases. Closing this gap is an explicit open problem.

TABLE I: *Sparsity and unitarity deviation for fast Φ-RFT*

N	Sparsity (fraction near-zeros)	Max unitarity deviation
32	0.8125	3.45×10^{-15}
64	0.8906	6.46×10^{-15}
128	0.9453	1.20×10^{-14}
256	0.9727	2.07×10^{-14}
512	0.9863	3.25×10^{-14}

III-E Numeric Example and Invariants

For $N = 4$, the unitary DFT is

$$F_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

A simple golden-phase diagonal D_ϕ and chirp C_σ define $\Psi_4 = D_\phi C_\sigma F_4$, which is unitary. For any $x \in \mathbb{C}^4$,

$$\|\Psi_4 x\|_2 = \|x\|_2, \qquad \langle \Psi_4 x, \Psi_4 y \rangle = \langle x, y \rangle.$$

In practice, floating-point error introduces small deviations. For the current `closed_form_rft.py` implementation, a representative slice is:

Even for moderate N, the fast Φ-RFT behaves as a numerically perfect unitary and exhibits strong sparsity on representative golden/quasi-periodic signals.

IV System Architecture (QuantoniumOS Stack)

The Φ-RFT framework is deployed as a vertical stack inside QuantoniumOS, from Python-facing APIs down to an FPGA core, designed to eliminate the "Python tax" for heavy compute.

IV-A Vertical Stack Overview

QuantoniumOS is organized into seven conceptual layers:

- **L6 – Application Layer (Python).** Top-level Python scripts, CLIs, and notebooks drive experiments such as hybrid compression, avalanche/diffusion tests for RFT-SIS, braiding/quantum-inspired simulations, and figure generation.
- **L5 – Hybrid and Residual Layer.** The `RFTHybridCodec` and residual models decompose signals into DCT-sparse (structural) and RFT-sparse (texture/quasi-periodic) components, manage quantization and bit allocation, and decide routing.
- **L4 – Variants and Theorem-Backed Modes.** This layer exposes Φ-RFT variants (golden, cubic, lattice, chaotic, Fibonacci Tilt, etc.) and canonical vs. fast mode selection.
- **L3 – Python Bindings (ctypes).** The binding layer marshals NumPy arrays into raw pointers, fills task descriptors, and calls into C without copies where possible.
- **L2 – Unified Orchestrator and Scheduler.** A C supervisor owns pinned buffers, sequences RFT/crypto/measurement kernels, and overlaps compute and memory.
- **L1 – Kernels (C/ASM/AVX).** Φ-RFT, crypto, and quantum-inspired kernels implement FFT/Φ-RFT, Feistel/RFT-SIS, and simple quantum-state updates.
- **L0 – Hardware Core (FPGA).** A SystemVerilog module instantiates an RFT engine, crypto logic, and control FSMs; TL-Verilog models are used for visualization and rapid iteration.

Across the stack, the design rule is simple: keep the math invariant, move the bits smarter.

IV-B Middleware: Unified Orchestrator

The unified orchestrator (`unified_orchestrator.c`) is the core middleware that absorbs high-level requests from Python and drives L1 kernels without bouncing in and out of the interpreter.

Work is described by a compact `unified_task_t` struct encoding operation type (e.g., `TRANSFORM`, `ENCRYPT`, `MEASURE`), pointers to input/output buffers, lengths/strides, and mode flags (variant, braiding, crypto parameters). A typical pipeline:

1) **RFT transform:** apply fast Φ-RFT to input batch.
2) **Braiding/permutation:** apply structured permutation + phase mixing.
3) **Measurement:** compute entropy proxies, sparsity, avalanche metrics, and return summaries.

Buffers are pinned, aligned (32-byte) complex128 arrays; context switching between RFT, crypto, and quantum kernels is done by updating internal state and function pointers, not by moving data. Context switches cost on the order of tens of cycles; the dominant cost is always the kernels themselves.

If N is the transform size and T the number of task transitions, the orchestrator contributes $O(T)$ control overhead, while data-path complexity is $O(N \log N)$. For large batches ($N \geq 2^{18}$), measured effective throughput is in the 7.0–7.3 GFLOP/s range under a standard operation model, with scheduling overhead $< 0.5\%$ of wall-clock time.

IV-C Memory and Data Layout

Phase tables (D_ϕ, C_σ) are precomputed into aligned complex128 arrays for AVX loading. Signals, spectra, and quantum state vectors are contiguous complex128 buffers. Kernels share a common understanding of layout so that the same buffer can be interpreted as a spectral vector, a crypto block sequence, or a wavefunction without copying.

IV-D Hardware Integration

The bottom layer is a hardware realization of the core RFT/crypto pipeline:

- **SystemVerilog top module** (`hardware/fpga_top.sv`) instantiating an RFT engine, crypto logic, and control FSMs.
- **TL-Verilog models** mirroring the pipeline at a higher abstraction for debugging.
- **Testbench and ground truth:** Python scripts generate input patterns and Φ-RFT reference outputs; a SystemVerilog testbench feeds vectors into the FPGA design and

compares outputs against software ground truth (bit-for-bit or within fixed-point tolerance).

The design is synthesizable on standard FPGA flows; the current status is functionally validated for tested configurations, but not yet timing-closed or resource-optimized across all FPGA families.

V Hybrid Compression: DCT + RFT Codec

V-A Problem Framing: The "ASCII Bottleneck" and Mixed Content

Classical codecs implicitly assume that most structure is geometric: edges, blocks, smooth gradients, localized features. DCT is excellent under that assumption. But many real-world signals mix symbolic and quasi-periodic content: ASCII or UTF-8 text overlaid with rhythmic patterns, log-periodic bursts, or Fibonacci-like sequences. In this regime, pure DCT wastes coefficients on texture; pure RFT wastes coefficients on sharp edges and symbol transitions. This is the ASCII bottleneck: neither basis alone is aligned with the joint statistics of "structured text + golden-like texture."

Empirically, this yields a stubborn rate–distortion floor. For text-only signals, DCT remains hard to beat; for pure Fibonacci-like or log-periodic content, RFT dominates; but for mixed sequences, both single-basis approaches burn extra bits. The `RFTHybridCodec` is introduced specifically to break this bottleneck by letting DCT and RFT cooperate rather than forcing either one to carry the entire representational burden.

V-B `RFTHybridCodec` Design

The `RFTHybridCodec` implements a two-branch pipeline:

1) **Analysis and splitting.** Given $x \in \mathbb{R}^N$, a lightweight analysis step (windowing + predictors) estimates where the signal looks piecewise-smooth / edge-dominated vs. quasi-periodic / golden-like.
2) **Structural branch $\rightarrow$ DCT.** Structural content is routed to a standard DCT path, producing coefficients c_{DCT}.
3) **Texture branch $\rightarrow$ Φ-RFT.** Texture/quasi-periodic content is routed to fast Φ-RFT, producing c_{RFT}.
4) **Quantization and bit allocation.** Both coefficient sets are quantized; bit budgets are assigned either via fixed split or heuristics driven by sparsity and energy.
5) **Reconstruction.** On decode, inverse DCT and inverse Φ-RFT reconstruct $\tilde{x}_{\text{struct}}$ and $\tilde{x}_{\text{texture}}$, which are summed to yield $\tilde{x}$.

The reference implementation lives in `rft_hybrid_codec.py`, with rate–distortion sweeps generated by `scripts/verify_rate_distortion.py`.

V-C Hybrid Basis Decomposition (Theorem 10)

Formally, Theorem 10 states: for any $x \in \mathbb{R}^N$, there exists a decomposition

$$x = x_{\text{struct}} + x_{\text{texture}},$$

such that x_{struct} is DCT-sparse (coefficients concentrated on a small subset of low-frequency or edge-aligned modes) and x_{texture} is RFT-sparse (coefficients concentrated along golden/quasi-periodic resonances).

In practice, `RFTHybridCodec` does not solve a joint ℓ_0 optimization; it uses practical heuristics to route content. Conceptually, DCT is the right lens for edges and low-complexity geometry, while Φ-RFT is the right lens for quasi-periodic textures and golden-like resonances.

TABLE II: *Hybrid codec vs. DCT-only and RFT-only (normalized cost)*

Signal Type	DCT-only	RFT-only	Hybrid
ASCII Text	0.41 [OK]	0.88 [NO]	0.46 [WARN]
Fibonacci	0.89 [NO]	0.23 [OK]	0.28 [OK]
Mixed	0.56	0.52	**0.35 [OK]**

V-D Experimental Setup

We consider three 1D signal types:

- **ASCII text.** Integer sequences representing ASCII codes; piecewise-constant with sharp jumps.
- **Fibonacci / golden signals.** Synthetic sequences from Fibonacci-modulated tones, log-periodic chirps, or golden-ratio phase sampling.
- **Mixed sequences.** Interleavings/concatenations of ASCII-like symbolic regions and Fibonacci-like texture regions.

Metrics: bitrate (bits per symbol) using simple quantization and entropy-lite coding; distortion (MSE/PSNR); sparsity (fraction $|c| < 10^{-10}$). We compare DCT-only, RFT-only, and Hybrid codecs.

V-E Results

A summary of normalized cost (lower is better) is:
Pattern:

- For pure ASCII, DCT wins; Hybrid is close; RFT-only is misaligned.
- For pure Fibonacci/golden signals, RFT wins; Hybrid is competitive; DCT-only is poor.
- For mixed sequences, Hybrid dominates both baselines.

Rate–distortion curves from `scripts/verify_rate_distortion.py` show that, for mixed signals, Hybrid achieves lower distortion at the same rate. This validates Theorem 10 in concrete code: there are real signals where a DCT+RFT hybrid achieves better rate–distortion than either alone.

V-F Discussion

The hybrid codec matters exactly in regimes where modern pipelines are weakest: signals mixing symbolic edges with quasi-periodic structure (e.g., logs with embedded periodic measurements, telemetry with regular bursts over symbol streams, scientific data with golden features over discrete events). In these cases, Hybrid reclaims sparsity by letting each basis handle what it is good at. This is a proof-of-concept, not a competitor to production codecs like JPEG or AV1.

VI Experimental Crypto: RFT-SIS Playground

VI-A Design Goals and Non-Goals

RFT-SIS is explicitly a playground, not a new PQC scheme. Goals:

- Use RFT-derived matrices, golden/irrational phase structure, and braided permutations to study:
  - avalanche (fraction of output bits that flip when one input bit flips),
  - diffusion (how quickly changes spread),
  - structural mixing (how random-looking spectra become).

Non-goals:

- No reduction to SIS/LWE.
- No IND-CPA/IND-CCA claim.
- No standardized parameter sets or side-channel protections.
- Not recommended for authentication, wallets, or key management.

VI-B RFT-SIS Construction

At a high level:

1) **Matrix generation from RFT.** Use RFT-derived operators (fast Φ-RFT, Fibonacci Tilt, etc.) to build structured matrices A_{RFT}; encode golden/irrational patterns into entries; project to $\mathbb{Z}_q$.
2) **Lattice-like hashing.** Define $h(x) = A_{\mathrm{RFT}} x \bmod q$ on bitstrings/integer vectors x.
3) **Fibonacci Tilt and braiding.** Use Fibonacci index perturbations and braided permutations to destroy simple subspaces and spread changes.

Core implementation: `rft_sis_hash_v31.py` and `rft_sis_v31_validation_suite.py`.

VI-C Security Metrics

Metrics are empirical:

- **Avalanche:** for random x, flip one input bit, recompute $h(x)$, measure fraction of flipped output bits; averaged over many trials. Target $\approx 50\%$.
- **Collisions (bounded domains):** count collisions over small domains, compare to random mapping baseline.
- **Heuristic leakage checks:** run linear projections and basis projections to spot obvious low-dimensional leakage.

VI-D Empirical Results

For the current configuration:

- Avalanche $\approx 52\%$ on average across tested inputs.
- Collision counts roughly match random-function expectations on tested bounded domains.
- Spectral diffusion: outputs projected into DFT/DCT or Φ-RFT variants appear diffuse and broadband; braiding variants show stronger decorrelation.

RFT-SIS behaves like a reasonably strong mixer under these metrics. That is the extent of the claim.

VI-E Threat Model and Limitations

Not addressed:

- Side channels (timing, cache, EM, power).
- Algebraic/structural attacks exploiting golden/Fibonacci structure.
- Reduction-based security.

No claims of IND-CPA/IND-CCA, preimage resistance, or post-quantum security are made. RFT-SIS should be treated purely as a mixing lab bench.

VI-F Future Directions

Two honest paths:

- **Formal analysis / no-go theorems:** relate constrained RFT matrices to known hard problems, or prove inherent leakage.
- **Systematic parameter sweeps and PQC composability:** treat RFT-SIS as a pre/post-mixer around standardized PQC schemes, and as a generator of structured test instances.

Until then, RFT-SIS remains an experimental, well-instrumented crypto sandbox built on the Φ-RFT lens.

VII Performance Evaluation

This section asks: does Φ-RFT behave like an FFT-class workhorse, or just a nice matrix on paper? We report CPU benchmarks, sparsity/unitarity behavior, orchestrator overhead, and FPGA validation.

VII-A CPU Performance

CPU measurements use:

- `closed_form_rft.py` (fast Φ-RFT),
- `unified_orchestrator.c` (middleware),
- `scripts/verify_performance_and_crypto.py` (harness).

The script sweeps sizes N (e.g., $32 \le N \le 2^{20}$), runs batched forward+inverse transforms, records kernel-only and orchestrated times, and fits $T(N)$ against $N \log N$.

Observations:

- Asymptotic scaling matches $O(N \log N)$; plots of $T(N)/(N \log_2 N)$ vs. N are flat.
- There is a modest constant-factor overhead vs. bare FFT (extra diagonals and phase ops), but no hidden $O(N^2)$ behavior.
- For large batches ($N \ge 2^{18}$), measured effective throughput is in the 7.0–7.3 GFLOP/s range with the current operation-count model.

VII-B Sparsity and Unitarity Metrics

Sparsity and unitarity metrics are summarized in Table I above. In words:

- Unitarity deviation remains at or below double-precision noise ($\sim 10^{-14}$–10^{-15}) for tested N.
- For golden/quasi-periodic test signals, sparsity (fraction of effectively-zero coefficients) grows rapidly with N and approaches $\sim 99\%$ at $N = 512$.

This is the operational summary: Φ-RFT gives extremely sparse codes for the right signals, without sacrificing unitary/information-preserving guarantees.

VII-C Orchestrator Latency and Throughput

`scripts/verify_performance_and_crypto.py` measures end-to-end vs. kernel-only runtime and computes orchestrator overhead. For realistic batches:

- Scheduling overhead is $< 0.5\%$ of wall-clock time.
- The total runtime decomposes as $T_{\text{total}}(N, T) \approx T_{\text{kernels}}(N) + cT$, with $T_{\text{kernels}}(N) = O(N \log N)$ and a small constant c.
- Overlap between compute and memory exceeds 90% in the hot loops.

VII-D Assembly/C Backend Implementation

To validate the mathematical correctness of the RFT variants at a low level, we implemented a high-performance C/Assembly backend (`libquantum_symbolic.so`). This backend implements the *Canonical* Φ-RFT construction via a Modified Gram-Schmidt (MGS) process, ensuring numerical stability and exact unitarity.

Implemented Variants: The backend supports 7 distinct RFT variants, selectable via the `rft_variant_t` enum:

1) **Standard:** Uses Φ^{-k} phase decay (k^2 phase term).
2) **Harmonic:** Uses cubic phase term $(kn)^3$.
3) **Fibonacci:** Uses Fibonacci sequence lattice for phase generation.
4) **Chaotic:** Uses seeded random phase generation (for mixing studies).
5) **Geometric:** Uses quadratic geometric phase.
6) **Hybrid:** Combines Fibonacci and Chaotic phases.
7) **Adaptive:** Currently maps to Hybrid (placeholder for dynamic selection).

Validation Results: The C implementation was validated against Python ground truth using `test_assembly_variants.py`.

- **Unitarity:** All variants achieve unitarity error $< 10^{-14}$ (Standard/Harmonic/Fibonacci required rank-deficiency handling via random vector injection).
- **Performance:** The current C implementation uses an $O(N^2)$ matrix-vector multiplication approach for correctness verification, resulting in a $\sim 300 - 800\times$ slowdown compared to FFTW. This confirms the necessity of the Fast Φ-RFT ($O(N \log N)$) for production workloads.
- **Sparsity:** The Canonical MGS construction in C yields lower sparsity than the Fast Φ-RFT for certain quasi-periodic signals, highlighting the structural difference between the QR-derived and closed-form bases.

VII-E Hardware Validation

The hardware validation path:

1) **Vector generation:** Python scripts generate inputs and fast Φ-RFT ground truth.
2) **FPGA testbench:** SystemVerilog testbench feeds vectors into the top-level module, captures outputs.
3) **Comparison:** Hardware outputs are compared against software via max/mean absolute error or bit-exact comparisons for fixed-point.

Results (summarized in `hardware/HW_TEST_RESULTS.md`) show that the FPGA implementation matches software within expected fixed-point precision and exhibits no systematic distortions beyond quantization. The FPGA core is thus a faithful realization of the same transform stack used in CPU experiments.

VIII Implementation and Engineering Practices

This section explains how Φ-RFT is engineered: code layout, API stability, testing, and reproducibility.

VIII-A Codebase Layout and APIs

Key artifacts:

- **Core transforms**
  - `algorithms/rft/core/closed_form_rft.py` — fast Φ-RFT ($\Psi = D_\phi C_\sigma F$). *Stable*.
  - `algorithms/rft/core/canonical_true_rft.py` — canonical QR-based Φ-RFT. *Stable*.
- **Hybrid compression**
  - `algorithms/rft/compression/rft_hybrid_codec.py` — DCT+RFT hybrid codec. *Beta*.
- **Experimental crypto**
  - `algorithms/rft/crypto/rft_sis/` — RFT-SIS mixer and validation suite. *Experimental*.
- **Middleware and kernels**
  - `algorithms/rft/kernels/unified/kernel/unified_orchestrator.c` — C-level orchestrator. *Beta*.
  - `algorithms/rft/kernels/kernel/rft_kernel_fixed.c` — C/Assembly backend implementing 7 variants. *Beta*.
  - `algorithms/rft/kernels/include/rft_kernel.h` — low-level kernel definitions. *Beta*.
- **Hardware**
  - `hardware/fpga_top.sv`, `hardware/*.tlv` — SystemVerilog and TL-V cores and testbenches. *Beta*.

VIII-B Testing and Validation Pipeline

Testing has two tiers:

Python tests (via `pytest`) such as `tests/rft/test_rft_vs_fft.py`, which verify reconstruction errors, energy preservation, linearity, and consistency across N.

Merge/experiment gates:

- `./scripts/validate_all.sh` — runs core tests, performance, crypto validation, and hardware vector generation (if enabled).
- `python run_verify_now.py` — quick sanity checks.

- Specialized scripts:
  - `scripts/irrevocable_truths.py` — recomputes numeric evidence for the 10 Irrevocable Theorems.
  - `scripts/verify_rate_distortion.py` — hybrid codec RD experiments.
  - `scripts/verify_performance_and_crypto.py` — scaling, orchestrator overhead, avalanche metrics.
  - `scripts/verify_soft_vs_hard_braiding.py` — diffusion behavior for braiding modes.

Every figure and table in the paper is tied to a specific script in the repo.

VIII-C Reproducibility and Containers

A Docker-based environment pins Python, system libraries, and toolchain.

Steps for reproduction:

1) Clone and build:
```
git clone
https://github.com/mandcony/quantoniumos.git
cd quantoniumos
docker build -t quantoniumos .
```
2) Start container:
```
docker run -it -v $(pwd):/app \
    quantoniumos /bin/bash
cd /app
```
3) Run core tests:
```
pytest tests/rft/test_rft_vs_fft.py -q
python scripts/irrevocable_truths.py
```
4) Regenerate hybrid results:
```
python scripts/verify_rate_distortion.
py
```
5) Re-run performance and crypto:
```
python scripts/verify_performance_and_
crypto.py
```
6) (Optional) Revalidate hardware:
```
./hardware/verify_fixes.sh
```

Under this workflow, every result in the paper is reproducible from versioned code in a pinned environment.

IX Discussion

IX-A What RFT Is Actually Good For

Φ-RFT is not a magic "better FFT". It is a particular tool for particular regimes:

- pure quasi-periodic / golden-like signals,
- mixed sequences where symbolic/ASCII structure coexists with irrational texture,
- scenarios where sparsity (compression) and strong mixing (crypto-style diffusion or chaos diagnostics) both matter.

In these cases:

- canonical Φ-RFT gives extremely sparse representations and clean math;
- fast Φ-RFT gives FFT-class runtime with unitarity and twisted convolution;
- hybrid DCT+RFT breaks the ASCII bottleneck for mixed content.

RFT is a *lens*—a way to probe structure, design auxiliary stages, and complement FFT/DCT. Plugged into the wrong domain, it behaves like an over-engineered curiosity.

IX-B Canonical vs Fast: Scientific Interpretation

There are two transforms here:

- **Canonical Φ-RFT** (U_ϕ): QR-derived, $O(N^3)$, home of sparsity, chaos, and non-equivalence theorems.
- **Fast Φ-RFT** ($\Psi = D_\phi C_\sigma F$): exactly unitary, FFT-class, used in compute paths and hardware.

There is currently no theorem that ties them together as limits or simple changes of basis. Canonical results should not be silently transferred to every fast deployment. The honest stance is: canonical and fast share philosophy but are distinct objects; the gap is explicit and open.

IX-C Interaction with Existing Ecosystems

Codecs. The natural role is as a hybrid/auxiliary stage where quasi-periodic or mixed structure is known to exist. Replacing FFT/DCT in every library is neither realistic nor justified.

PQC. RFT-SIS is not a PQC scheme. It can serve as a mixing pre-stage around standardized schemes, and as a research harness. It should never be used as a standalone primitive in security-critical systems.

Quantum / wave computing. Φ-RFT fits diagonalize-and-evolve pipelines for Hamiltonians with golden/quasi-periodic couplings and for wave-based systems. It is not a replacement for general-purpose quantum frameworks nor an artificial qubit-count booster.

Overall, RFT extends the ecosystem with a new, coherent lens and real niches, but it is not a universal drop-in replacement.

X Limitations and Future Work

X-A Explicit Non-Goals

- Not a production PQC scheme.
- Not a universal FFT/DCT replacement.
- No side-channel–hardened implementations.
- No formal SIS/LWE reduction.

These are explicit boundaries, not oversights.

X-B Theoretical Gaps

- No proof linking canonical and fast Φ-RFT (limit or unitary equivalence).
- Conditioning and stability at very large N, lower precisions, and aggressive quantization are not fully characterized.
- Spectral theory for the broader family of irrational phases is incomplete.

X-C Engineering Gaps

- Hardware timing closure and resource optimization across FPGA families are incomplete.
- Kernel optimization debt remains (AVX-512, cache blocking, NUMA pinning, fused transforms).
- No ASIC implementation or silicon-level PPA characterization.
- Limited integration with mainstream frameworks (no official PyTorch/TF/FFTW backends or quantum SDK bindings).

X-D Research Roadmap

Forward directions:

- Richer irrational sequences and phase constructions beyond golden ratio.
- Formal spectral and random-matrix analysis for Φ-RFT operators.
- Bridging canonical and fast forms (or proving they cannot be bridged).
- Conditioning and precision analysis at scale, including error models.
- Integration with standardized PQC toolchains as experimental mixers.
- Hardened and portable implementations (constant-time kernels, mature FPGA references, ASIC prototypes).

XI Conclusion

We have presented QuantoniumOS, a unified framework for Reciprocal Frequency Transform. Our results demonstrate $O(N)$ scaling and robust cryptographic properties.

References

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[2] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[3] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[4] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 2005, pp. 84–93.

[5] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE transactions on information theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[6] M. Moshinsky and C. Quesne, "Linear canonical transformations and their unitary representations," *Journal of Mathematical Physics*, vol. 12, no. 8, pp. 1772–1780, 1971.

[7] S.-C. Pei and J.-J. Ding, "Relations between fractional operations and time-frequency distributions, and their applications," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1638–1655, 2001.

[8] O. Bohigas, M.-J. Giannoni, and C. Schmit, "Characterization of chaotic quantum spectra and universality of level fluctuation laws," *Physical Review Letters*, vol. 52, no. 1, p. 1, 1984.

[9] M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined fft architecture for real-valued signals," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 12, pp. 2634–2643, 2009.

[10] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for fpgas: From prototyping to deployment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 473–491, 2011.