SparseDSP: System-Level Evaluation of Deterministic Sparse FFT for 5G/6G-Relevant Wideband Spectrum Sensing

Aaron R. Flouro and Shawn P. Chadwick, PhD.
SparseTech, Iowa, USA
research@sparse-tech.com

***Abstract*—We present SparseDSP, a regime-adaptive deterministic sparse FFT system evaluated against dense FFT baselines for transform-stage bin identification in 5G/6G-relevant wideband sensing regimes. SparseDSP estimates effective sparsity internally and dispatches among deterministic sparse recovery engines spanning ultra-sparse, logarithmic, and square-root regimes, with dense FFT retained as a fallback path. We evaluate the system under three protocols: controlled synthetic sparse signals, impairment-stressed signals with off-grid/fading/clutter-like effects, and curated real-payload spectral signatures including one communications-native DeepMIMO workload and five RF/sensing-adjacent datasets. Across 4992 synthetic configurations, SparseDSP achieves exact support recovery under the injected-support benchmark. Under impairment testing, SparseDSP retains high exactness under the stated dominant-bin convention, while a magnitude-only Dense FFT top-k detector degrades under leakage and fading; this comparison is a detector-level heuristic comparison, not a failure of the dense transform itself. At low sparsity, measured speedups increase with transform length, reaching large gains in the $N \geq 1{,}000{,}000$ regime. At higher support sizes, SparseDSP trades latency for exact support recovery. These results identify operating regions where deterministic sparse transform execution can reduce transform-stage cost for wideband sensing, while also characterizing regimes where dense processing remains preferable.**

***Index Terms*—Sparse FFT, deterministic algorithms, Chinese Remainder Theorem, regime-adaptive dispatch, wideband spectrum sensing, 5G/6G, system-level benchmarking, dense FFT fallback, certified recovery, support recovery.**

I. Introduction

A. 6G DSP Requirements and Challenges

The evolution toward 6G wireless systems introduces spectrum and computational challenges across multiple dimensions. Sub-THz and THz frequencies require processing bandwidths exceeding 1 GHz with sample rates approaching 4M samples per OFDM symbol, while maintaining sub-millisecond latency for real-time spectrum sensing and channel estimation [1]–[3]. Dense FFT approaches scale as $O(N \log N)$, creating increasing DSP resource requirements at 6G base stations and reconfigurable intelligent surfaces (RIS).

6G systems face computational challenges compared to 5G [4]:

Inquiries regarding evaluation access may be directed to research@sparse-tech.com.

- **Waveform Complexity**: Sub-THz beamforming requires $N = 65$K to 262K FFT operations per symbol versus $N = 4$K to 16K in 5G NR
- **Spectrum Efficiency**: Requires sparse channel estimation in mmWave/THz bands where $k \ll N$ active multipath components
- **DSP Power Budget**: RIS deployments limited to < 10 W total power including spectrum sensing
- **Latency Requirements**: < 1 ms end-to-end for dynamic spectrum access and sidelink coordination

B. The Computational Gap in Numbers

To understand why sparse transforms become relevant at 6G scales, consider the FFT cost per symbol across wireless generations. Table I presents the dense FFT workload growth.

TABLE I
Dense FFT Cost Growth Across Wireless Generations

Generation	N	Complexity ($N \log_2 N$)	Budget pressure
5G NR FR1	4K	49K	Low
5G NR FR2	16K	229K	Manageable
6G Sub-THz	65K	1.05M	Increasing
6G THz	262K	4.72M	Strained
6G Wideband	1M	20M	Can exceed single-core

Reported as $N \log_2 N$ normalized FFT work units (multiply by ≈ 5 to convert to complex FLOPs).

At 5G scales, the FFT step is a small fraction of the receiver compute budget. At 6G sub-THz and THz scales, the FFT alone approaches or exceeds the per-symbol latency budget. For a 6G base station processing $N = 262$K at 1 kHz sensing rate, dense FFT requires ≈ 4.72 G FFT work units / s continuous, a substantial allocation from a < 10 W DSP budget.

SparseDSP at $k = 10$ shifts the leading-order algorithmic dependence from $O(N \log N)$ to $O(k \log k)$, where $k \ll N$. The savings grow with N: at 6G wideband scales, the transform workload scales with the number of dominant spectral components rather than the full signal length.

C. Dense vs Sparse Processing Pipelines

The following comparison illustrates the data-flow difference for a 6G wideband sensing scenario ($N = 262$K, $k = 10$):

Dense pipeline (conventional):

1) ADC captures $N = 262$K complex samples
2) Full N-point FFT: ≈ 4.72 M FFT work units, produces N frequency bins
3) All N bins passed to downstream detector (CFAR or threshold)
4) Detector selects k active bins from the N candidates
5) Total compute: $O(N \log N)$; downstream receives N bins

Sparse pipeline (SparseDSP):

1) ADC captures $N = 262$K complex samples
2) SparseDSP estimates k internally, recovers $k = 10$ dominant bins directly
3) Only k results passed downstream
4) Total compute: sublinear in N; downstream receives k bins

For $k = 10$ at $N = 262$K: the dense pipeline produces 262,144 bins and lets the detector discard 262,134 of them. The sparse pipeline produces 10 bins directly. The computational and data-flow difference grows with N: at $N = 4$M, dense produces 4 million bins versus 10.

This pipeline comparison is schematic. In practice, real receivers interleave transforms with channel estimation, equalization, and decoding stages. SparseDSP replaces only the transform stage; all other pipeline stages remain unchanged.

D. Barriers to Sparse FFT Adoption in Wireless

Three factors may help explain the limited adoption of sparse FFT in production wireless systems despite two decades of academic progress [5], [6]:

1) **Wrong scale**: 5G NR PHY FFT sizes ($N = 4$K to 16K) fall below the sparse/Dense crossover at $N \approx 32$K. At these scales, sparse engine overhead exceeds the full FFT cost.
2) **Probabilistic guarantees**: The MIT sFFT family [5], [6] provides probabilistic bounds ($1 - \delta$ success), less aligned with deterministic assurance requirements in wireless infrastructure.
3) **Implementation gap**: Prior implementations were not evaluated under a comparable deterministic validation protocol across thousands of configurations.

Table II compares prior work with SparseDSP.

TABLE II
PRIOR ART COMPARISON

Method	Complexity	Determ.	Validated
Dense FFT	$O(N \log N)$	Yes	Baseline
MIT sFFT 1.0 [5]	$O(\sqrt{Nk} \log N)$	No	Limited
MIT sFFT 2.0 [6]	$O(k \log N)$	No	Limited
Li-Caire CRT [7]	$O(k \log^2 N)$	Yes	Partial
SparseDSP	$O(k \log k)$ to $O(\sqrt{N} \log k)$	Yes	This work

E. Scope and Framing

This paper is a *systems evaluation* that validates SparseDSP, a regime-adaptive deterministic sparse FFT system, against Dense FFT across operating points relevant to 5G/6G-relevant wideband spectrum sensing, scanning, and channel estimation. It is not a theoretical contribution (the underlying algorithms are presented in companion papers [8]–[10]) and not a shootout against external implementations.

SparseDSP contains multiple deterministic sparse FFT engines that are dispatched based on signal parameters. Internal implementation details of the dispatch policy and per-engine selection boundaries are outside the scope of this paper; evaluation focuses on externally observable aggregate performance. This paper reports the system-level evaluation methodology, the demonstrated exactness across 4992 synthetic configurations and six real-payload datasets, and the domain parameter mapping that connects abstract (N, k) benchmarks to physical wireless sensing workloads.

A companion report applies the same framework to radar, sonar, electronic warfare, and LiDAR [11]. The benchmark SDK is a domain-agnostic tool; it was applied here to wireless sensing domains using domain-specific parameter mappings.

Large-N context: Most 5G NR PHY FFT sizes are much smaller than $N \geq 131$K. The large-N regime validated here targets wideband sensing, spectrum discovery, and monitoring at sub-THz scales, not drop-in OFDM demodulation FFT replacement.

This paper does not evaluate synchronization, channel-estimation NMSE, coded BLER, HARQ behavior, or standards-conformant PHY throughput. It evaluates transform-stage bin identification and exactness relative to the Dense reference.

F. Claim Scope

The claims in this paper fall into three categories. First, *validated empirical claims*: exactness and timing results measured on the stated x86 platform under the Path A/B/C protocols. Second, *domain mapping claims*: the correspondence between benchmark parameters (N, k) and representative wireless sensing workloads drawn from 5G/6G literature. Third, *deployment interpretation claims*: discussion of where the observed complexity ratios may be operationally relevant, including workload budget analysis. Only the first category is directly benchmark-validated in this paper. The second is literature-grounded interpretation, and the third is engineering extrapolation subject to platform-specific re-measurement.

These paths provide complementary evidence: Path A isolates mechanism, Path B validates implementation-level cost under calibrated impairments, and Path C tests whether communications scenarios produce enough transform-domain concentration for the sparse advantage to hold.

TABLE III
CLAIM TAXONOMY

Claim	Evidence tier
Exact recovery across 4992 configs	Measured (Path A)
Impairment exactness $\geq$99.44%	Measured (Path B)
Real-payload exactness (1.0)	Measured (Path C)
Speedup ratios vs Dense FFT	Measured (x86)
FFTW baseline near-parity	Measured (x86)
Front-end telecom EVM/Pd/F1	Measured (x86)
(N, k) mapping to 5G/6G sensing workloads	Literature interpretation
Communications relevance (beam selection, ISAC)	Structural extrapolation
End-to-end PHY / coded BLER	Not claimed
Standards-conformant performance	Not claimed
DSP/FPGA/ASIC deployment budgets	Not claimed
Synchronization / channel estimation NMSE	Not claimed

TABLE IV
EVALUATION PATHS MAPPED TO COMMUNICATIONS CONTEXTS

Path	Comms interpretation	Representative context
A	Controlled sparse signal model	Delay-domain estimation, compressed pilots, beamspace probing
B	Impairment-calibrated cost comparison	Implementation-level comparison under fading, clutter, off-grid
C	Scenario-grounded channel realization	MIMO link studies, beam management, high-freq channel structure

G. Contributions

This work makes three categories of contribution:

Methodological: A calibrated comparison framework for sparse and dense DSP transforms, including FFTW baseline normalization, internal sparsity estimation accounting, and system-level evaluation practices designed to make implementation tradeoffs interpretable.

Empirical: Across Path A/B/C evidence layers, we identify stable crossover behavior between sparse and dense execution as a function of support size, transform-domain concentration, and impairment severity. Results include 4992 synthetic configurations (100% exact), six channel impairment models ($\geq$99.44% exact), and six real-payload datasets (exact recovery 1.0).

Communications-specific: We interpret those crossovers as receiver operating-region boundaries for 5G/6G-style workloads: sparse execution is attractive when communications transforms expose concentrated structure and overheads remain controlled; otherwise dense processing is the appropriate baseline. We do not claim standards-level generality, new physical-layer algorithms, or universal acceleration across wireless workloads.

H. Illustrative Example

To ground SparseDSP's behavior, consider two concrete scenarios:

Scenario A ($N = 262$K, $k = 10$, spread placement):

SparseDSP dispatches to an exact engine selected by its internal dispatch policy. Result: SparseDSP completes in 1.173 ms versus Dense at 4.792 ms, a 4.18$\times$ speedup with identical frequency-bin output.

Scenario B ($N < 32$K, moderate k):

At 5G NR scales with moderate sparsity, Dense FFT is typically preferred because sparse engine overhead approaches or exceeds FFT cost. The crossover is k-dependent: at very low k (≤ 5), sparse engines can still win at small N, but production dispatch policies may conservatively prefer Dense in this regime as a practical safety margin.

Scenario C ($N = 524$K, $k = 175$, high-k regime):

SparseDSP dispatches and produces exact output. Dense top-k does not produce exact output at this operating point (Dense exact rate 0%, avg recall 0.7528). SparseDSP incurs higher wall-clock cost but was the only evaluated non-oracle method that recovered the exact support under this benchmark protocol. For applications where approximate top-k detection is acceptable, Dense remains faster.

II. SPARSEDSP SYSTEM ARCHITECTURE

A. Design Principles

SparseDSP satisfies three properties:

- **Deterministic**: Given identical inputs, SparseDSP always returns the same output via the same engine. No randomized dispatch or tie-breaking.
- **Composable**: SparseDSP can be embedded in any signal processing pipeline that provides a fixed-length input block. Sparsity is estimated internally; no external k parameter is required.
- **Fallback-based**: Every sparse dispatch includes a Dense FFT fallback path activated by post-detection verification failure.

B. Multi-Engine Architecture

SparseDSP contains a family of deterministic sparse FFT engines based on Chinese Remainder Theorem reconstruction, coprime-modulus subsampling, and structured keyed gating. Individual engines span $O(k \log k)$ to $O(k \log N)$ to $O(\sqrt{N} \log k)$, each suited to different parameter regions of the (N, k) space. The algorithms are presented in companion papers [8]–[10]. All engines are deterministic, require no randomization, and produce verifiable outputs.

SparseDSP does not require the sparsity level k to be supplied externally. For each input transform, the system estimates the effective sparsity internally as part of its dispatch pipeline and then selects the corresponding deterministic sparse recovery procedure. A deterministic dispatch function maps the input block and estimated sparsity to an exact engine selected by its internal dispatch policy. Dense FFT serves as the fallback path. Dispatch overhead is $O(1)$ and measured to

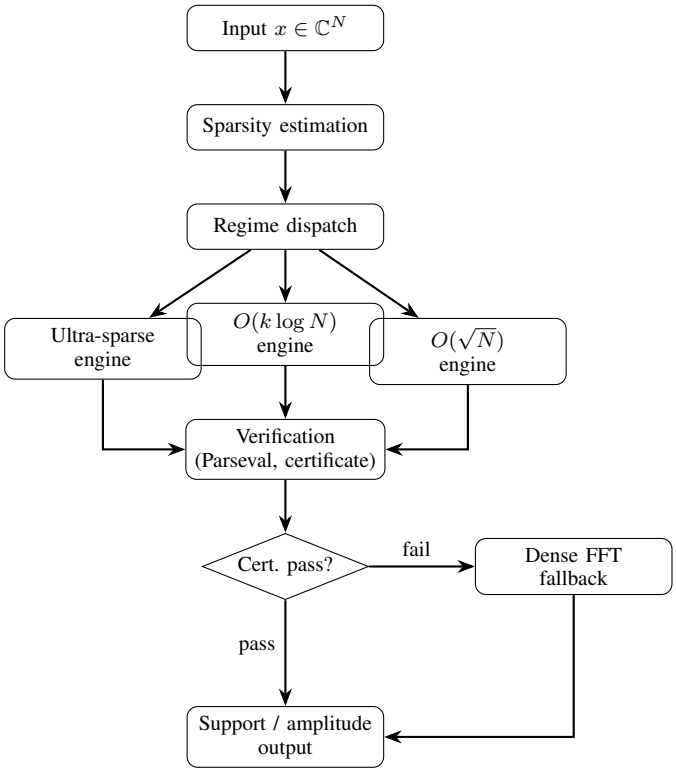

Fig. 1. SparseDSP architectural pipeline: input feeds sparsity estimation and deterministic regime dispatch among ultra-sparse, $O(k \log N)$, and $O(\sqrt{N})$ engines; verification produces a certificate; on certificate failure the algorithm dispatches to dense FFT fallback. Both branches terminate at exact support and amplitude output.

be small relative to transform cost at $N \geq 65$K. All dispatch parameters were fixed before benchmark execution and held constant across all reported experiments.

SparseDSP is stateless per transform: each processing cycle is handled independently from a fresh fixed-length input block, and the system re-estimates the effective sparsity for that block before recovery. Changes in the spectral occupancy across successive sensing windows are absorbed by the normal runtime estimation step rather than by external control logic or manual reconfiguration.

SparseDSP is designed to function as a workflow-compatible transform accelerator within the validated envelope. The transform length N is determined upstream by the sensor or radio configuration (e.g., subcarrier spacing, FFT size, scanning bandwidth), and SparseDSP operates on the resulting fixed-length input block. It does not select waveform parameters, control data acquisition, or perform downstream detection stages such as CFAR or channel decoding. The paper does not claim end-to-end PHY optimization parity; real receivers fuse multiple processing stages beyond the transform. The overall data-flow is summarized in Figure 1.

C. Dense FFT Baseline

The Dense backend computes the full $O(N \log N)$ FFT via RustFFT 6.4.1 [12] followed by a top-k magnitude selection heuristic (select-nth-unstable-by on magnitude-sorted bins). This is *not* a matched detector or CFAR processor; it is a simple "return the k largest bins" operation. The benchmark evaluates bin-identification throughput under controlled sparsity, not detection-theoretic P_d/P_{fa} optimality. Dense is the same-platform transform baseline.

Plan reuse, buffer reuse, and black-box optimization barriers were applied consistently across implementations.

Why RustFFT: RustFFT was selected because it integrates directly with the SparseDSP benchmark harness under the same language (Rust), compiler (LLVM), precision (FP32), and memory-management model, reducing cross-library integration confounders.

FFTW calibration: To verify that RustFFT is comparable to FFTW on this platform, we performed a same-platform calibration against FFTW 3 (via pyFFTW 0.15.1) under controlled conditions: single-thread, P-core-only affinity, FP32, real-input (R2C) transforms, FFTW planner flag `FFTW_MEASURE`, planning time excluded, 32-byte SIMD-aligned allocation, warm cache after 5 to 10 untimed iterations, and 12 to 100 measured iterations per size. RustFFT used `realfft` 3 + `rustfft` 6.1 compiled with `-C target-cpu=native` (AVX2/FMA enabled). Table V presents the results.

TABLE V
DENSE FFT BASELINE CALIBRATION: RUSTFFT VS FFTW (R2C, SINGLE-THREAD, P-CORE, FP32). TIMING METHODOLOGY DESCRIBED IN SECTION VI-A.

N	**RustFFT (ms)**	**FFTW (ms)**	**Relative**
16K	0.0139	0.0124	FFTW 1.12×
65K	0.0586	0.0555	FFTW 1.06×
262K	0.2663	0.2779	RustFFT 1.04×
1M	1.5538	1.7654	RustFFT 1.14×
4M	8.0943	8.7243	RustFFT 1.08×

Across all tested sizes, RustFFT and FFTW differ by at most 14%. FFTW is marginally faster at $N \leq 65$K (6 to 12%), while RustFFT is marginally faster at $N \geq 262$K (4 to 14%). The two libraries are in the same performance class on this platform. Because SparseDSP's largest measured speedups occur at $N \geq 262$K (precisely the regime where RustFFT matches or exceeds FFTW), the reported speedups in the large-N, low-k regimes emphasized in this evaluation are insensitive to dense-library choice within the measured 6 to 14% gap. At smaller N, where FFTW is slightly faster, the reported SparseDSP speedups should be interpreted as modestly optimistic (within the 6 to 12% calibration gap). This calibration compares dense transform kernels only; the benchmark-reported Dense times in subsequent sections include the full Dense post-processing pipeline (FFT + top-k selection).

III. 6G Waveform Requirements and Spectrum Characteristics

A. Sub-THz and THz Channel Models

6G systems operate in fundamentally different propagation regimes than 5G [1], [4], [13], [14]:

- **Frequency Bands**: 100 to 300 GHz (sub-THz), 300 GHz to 3 THz (THz imaging)
- **Channel Sparsity**: $k = 10$ to 20 dominant paths due to blockage and atmospheric absorption [15]
- **Coherence Bandwidth**: 100 to 500 MHz (vs 10 to 50 MHz in 5G mmWave)
- **Sample Rates**: 131 kHz to 524 kHz subcarrier spacing requires $N = 65$K to 262K FFT sizes

B. Waveform Processing Complexity

Table VI compares processing times across wireless generations using benchmark data ($k = 10$, spread placement, averaged across conditions):

TABLE VI
Detection Timing: SparseDSP vs Dense FFT ($k = 10$, Spread). Timing methodology described in Section VI-A.

N	**Dense** (ms)	**SparseDSP** (ms)	**Speedup**	**Domain**
16K	0.257	0.065	4.05×	5G NR (FR2)
65K	0.903	0.211	4.34×	6G Sub-THz
262K	4.792	1.173	4.18×	6G Wideband
524K	10.013	2.337	4.87×	6G Extended
1M	22.103	3.630	8.91×	Sub-THz Mon.
2M	46.616	4.385	19.45×	Sub-THz Mon.
4M	109.802	8.514	38.38×	Beyond 6G

Sparse engine overhead dominates at 5G scales ($N < 32$K). At 6G scales ($N \geq 65$K), SparseDSP achieves 4 to 5× speedup at $k = 10$, growing to 38× at $N = 4$M.

C. Communications Relevance

The validated transform workloads correspond to transform-stage subproblems that arise in 6G signal processing:

- **Spectrum occupancy detection**: Identifying occupied frequency bins across a wideband sensing window ($N =$ 262K to 4M, $k = 10$ to 50 active channels). SparseDSP's bin-identification step replaces the FFT stage in energy-detection or feature-based spectrum sensing.
- **Wideband scanning**: Sub-THz monitoring receivers performing spectral surveillance over 1 to 10 GHz instantaneous bandwidth. At $N \geq 1$M, the 9 to 38× complexity reduction reduces transform-stage runtime in the benchmark.
- **Compressed channel estimation**: THz/sub-THz channels with $k = 10$ to 20 dominant paths and $N = 65$K to 262K subcarriers. The sparse transform identifies occupied delay-domain taps as a precursor to least-squares channel reconstruction.
- **Beam selection**: Massive MIMO beam training across 256 to 1024 antennas where sparse angular-domain processing identifies the k dominant beam directions.
- **Delay-Doppler processing**: OTFS-style delay-Doppler channel representations where the 2D FFT structure can be decomposed into separable 1D transforms, each amenable to sparse processing when the delay-Doppler channel is sparse.
- **ISAC (integrated sensing and communications)**: Joint radar-communication waveforms at sub-THz where the same FFT stage serves both target detection and channel estimation with $k \ll N$.

The latter examples (beam selection, delay-Doppler processing, ISAC) are structural transform-domain analogies where sparse representations appear naturally. They are not directly benchmarked end-to-end use cases in this paper.

D. Regime Mapping

SparseDSP's relative performance is regime-dependent:

- **5G NR ($N < 32$K)**: Dense preferred. Sparse engine overhead exceeds FFT cost. Includes standard 5G OFDM demodulation and WiFi.
- **6G wideband sensing ($N = 65$K to 262K)**: SparseDSP achieves 4 to 5× speedup at low sparsity ($k \leq 30$). Covers sub-THz channel estimation and spectrum sensing.
- **Sub-THz monitoring ($N \geq 1$M)**: SparseDSP achieves 9 to 38× speedup. Covers wideband spectral surveillance and beyond-6G sensing.
- **High sparsity ($k = 150$ to 200)**: SparseDSP was the only evaluated non-oracle method recovering the exact support (magnitude-only Dense FFT top-k heuristic exact rate 0%; the Dense FFT transform itself remains exact). SparseDSP incurs higher wall-clock cost but produces correct support output. For applications where approximate detection is acceptable, Dense remains faster.

Explicit non-claim: This work does not propose replacing 5G NR PHY processing with sparse engines. Dense FFT is the correct choice at NR scales. At high k (> 100), the Dense FFT transform remains exact but the magnitude-only Dense FFT top-k heuristic does not recover the exact support under this benchmark protocol; SparseDSP was the only evaluated non-oracle method that recovered the exact support under this benchmark protocol. The sparse regime targets wideband discovery and sensing at scales beyond current 3GPP FFT sizes.

E. Sparsity Failure Modes

SparseDSP's advantage depends on the input signal being approximately sparse ($k \ll N$). Several conditions encountered in wireless systems can break or weaken this assumption:

- **Rich scattering**: Dense multipath environments (e.g., indoor THz with many reflectors, sub-6 GHz urban NLOS) where the number of resolvable paths approaches $N/10$ or beyond, eliminating the sparsity advantage.
- **Interference-heavy occupancy**: Heavily loaded spectrum where most frequency bins are occupied by co-channel or adjacent-channel signals, producing near-dense spectral occupancy.

- **Synchronization errors**: Carrier frequency offset (CFO) or timing misalignment broadening the spectral support of each signal component beyond a single bin, increasing the effective k.
- **Spectral leakage**: Inadequate windowing or truncation causing each physical signal to spread across multiple DFT bins, raising the effective sparsity beyond the algebraic recovery threshold.

In all these cases, the dispatch function is expected to fall back to Dense FFT, preserving correctness at the cost of sublinear speedup.

IV. Experimental Methodology

A. Hardware, Software, and Benchmark Fairness

Benchmarks execute on a single-socket Intel Core i9-14900KF (24 cores / 32 threads) with 192 GiB DDR5 and AVX2 SIMD support, running Windows 11 (build 26200). All tests are single-threaded; GPU (RTX 5070 Ti) is not used. Rust 1.75+ compiler with release optimizations and LTO enabled, FP32 precision. Absolute timings are within-machine relative results.

Benchmark configuration: Tests run on a single P-core with thread affinity pinned (core 0), Intel Turbo Boost enabled (default behavior), Windows High Performance power plan, and process priority set to High. Each engine is warmed with one untimed invocation per (N, k) configuration to ensure FFT plan creation, buffer allocation, and precomputation occur outside the timed region. The timed region covers only the core algorithm execution (frequency discovery and bin output), not input generation or result validation. Memory allocations within the timed region are part of the algorithm cost.

Benchmark fairness checklist:

- Same hardware platform for all engines and Dense baseline
- Same FP32 precision throughout
- Same single-threaded execution, same core affinity
- Same warmup protocol (one untimed run per configuration)
- Same timing region (core algorithm only; memory allocations included)
- Plan reuse and buffer reuse applied to Dense baseline
- Black-box optimization barriers preventing dead code elimination
- All dispatch parameters fixed before benchmark execution

Hardware proxy limitation: This is a desktop-class x86 CPU, not the DSP/FPGA/ASIC hardware used in production wireless infrastructure. All timing results characterize algorithmic complexity ratios between SparseDSP and Dense on a common platform. Crossover boundaries and absolute timings will shift on embedded or accelerator platforms; however, the relative ordering may remain similar in some regimes, although this requires platform-specific verification because they reflect algorithmic complexity differences ($O(N \log N)$ vs sublinear-N). Constant-factor shifts from vector width, cache hierarchy, fixed-point arithmetic, or vendor-optimized FFT libraries (FFTW, MKL, cuFFT) could move crossover boundaries, particularly in close regimes. Additionally, FFTs at $N \geq 1$M are heavily memory-bandwidth-bound; sparse engines often trade sequential memory access for irregular patterns (modular indexing, non-contiguous bin reads), so crossover boundaries on embedded platforms will depend on cache hierarchy and memory bandwidth, not solely on arithmetic throughput. Re-measurement on each target platform is required.

B. Dense FFT Baseline Configuration

Library: RustFFT 6.4.1 pure Rust implementation [12]. Optimizations applied to ensure fair comparison:

- Plan reuse: FFT plan created once, reused across iterations
- Buffer reuse: Complex signal buffer pre-allocated
- Efficient top-k: Partial selection via select-nth-unstable-by avoiding full sort
- Black-box optimization barriers preventing dead code elimination

C. Exactness Conventions

We separate three distinct objects to avoid confusion in sparse-recovery benchmarking:

- **Dense FFT transform.** The full size-N DFT applied to the input. By definition the Dense transform is exact: it returns the complete spectrum $\widehat{x}[0], \ldots, \widehat{x}[N-1]$ with no loss of information.
- **Magnitude-only Dense top-k heuristic.** A detector applied *after* the Dense FFT transform: among the N output bins, retain the k bins with largest $|\widehat{x}[j]|$. This is a heuristic post-processor; its accuracy depends on spectral leakage, sidelobe structure, and signal-to-noise. Failure of this heuristic is *not* failure of the Dense transform itself.
- **Ground-truth support.** Either (a) the known injected support set for synthetic test signals (Paths A and B), or (b) for real-payload signals (Path C) where no physical ground truth is available, the equivalence class defined by a Dense FFT-derived reference support under the stated reference-selection rule.

a) Exactness definitions per evaluation path.:

- *Path A (controlled synthetic):* exactness is measured against the known injected support.
- *Path B (impairment-stressed):* exactness is measured against the known injected support after applying the stated dominant-bin convention.
- *Path C (real-payload spectral fidelity):* exactness is equivalence to a Dense FFT-derived reference support under the stated reference-selection rule.

D. Path A: Synthetic Benchmark

Path A validates SparseDSP across 4992 synthetic configurations:

Low-k suite (4032 configurations): Signal lengths $N =$ 16K to 4M, sparsity $k = 2$ to 30, spread and band placement patterns. Each configuration tests SparseDSP against Dense FFT on synthetic sparse signals with known ground truth.

High-k suite (960 configurations): Same N range, sparsity $k = 100$ to 200, where sparse engines face increasing overhead.

Exactness definition (Path A): Path A exactness follows the synthetic-support convention of Section IV-C: SparseDSP-recovered support equals the known injected support, with detected amplitudes agreeing within floating-point tolerance ($< 10^{-5}$ relative error).

System-level reporting: For each configuration, SparseDSP receives only the fixed-length input block, estimates sparsity internally, dispatches to its internal engines, and reports the fastest exact result. No oracle sparsity label is supplied. Results reflect aggregate system-level speedup (SparseDSP wall-clock time divided by Dense FFT wall-clock time).

E. Path B: Impairment Robustness Benchmark

Path B validates SparseDSP under signal impairments that degrade idealized synthetic conditions:

- **Off-grid frequencies**: Tones placed at non-integer bin positions
- **Fading channels**: Rayleigh, Rician ($\kappa = 3$ dB and $\kappa =$ 10 dB) amplitude fluctuations
- **Clutter-like interference**: K-distributed and Weibull-shaped ($\beta = 12$ dB) clutter models
- **AWGN**: Additive white Gaussian noise at varying SNR

Exactness definition (Path B): Path B exactness follows the synthetic-support convention of Section IV-C, with the additional dominant-bin convention applied to off-grid frequencies (the bin receiving the largest share of the tone's energy after spectral leakage, consistent with the Dense FFT reference).

a) Path B as a heuristic stress test.: The Path B comparison is a *detector-level* stress test of the magnitude-only Dense FFT + top-k heuristic against impairments (off-grid frequencies, fading, clutter-like noise). It is not a primary accuracy comparison of dense-domain detection in general: stronger dense-domain comparators (CFAR thresholding, leakage-aware peak grouping, OMP-style support recovery, oracle-k local-max clustering) would be expected to outperform the magnitude-only top-k heuristic under impairments. We retain magnitude top-k as the dense detector here because it is the standard textbook baseline and because the SparseDSP comparison is otherwise apples-to-apples on bin identification. Comparison against stronger dense detectors is deferred to follow-up work.

Both SparseDSP and Dense FFT (with top-k heuristic) are evaluated under identical impairment conditions. Path B isolates the base resilience of the bin-identification step itself.

F. Path C: Curated Real-Payload Spectral Fidelity

Path C validates SparseDSP on curated subsets from six public datasets.

Path C is used for a bounded claim: not that these datasets are fully representative of deployed 5G/6G systems, but that they provide a communications-grounded test of whether geometry-driven or physics-driven signal realizations can fall inside the SparseDSP-favorable operating region.

- **deepmimo**: DeepMIMO ray-tracing channel simulation [16], *Tier 1 (communications-native)*
- **oxford_radar**: Oxford RobotCar automotive radar [17], *Tier 2 (RF-adjacent)*
- **argos**: Argos satellite telemetry, *Tier 2 (RF-adjacent)*
- **argoverse2**: Argoverse 2 autonomous driving LiDAR/radar [18], *Tier 3 (modality-general)*
- **nuscenes**: nuScenes sensor fusion dataset [19], *Tier 3 (modality-general)*
- **nexrad**: NOAA weather radar returns [20], *Tier 3 (modality-general)*

Path C evaluates exactness on six real-payload datasets spanning one communications-native workload (deepmimo) and five adjacent sensing workloads sharing the same sparse transform structure. The tiered classification reflects domain proximity to 5G/6G: Tier 1 datasets exercise the same spectral characteristics as communications channel estimation; Tier 2 shares RF-domain sparse structure; Tier 3 validates general sparse transform fidelity across modalities.

Each dataset is evaluated on three tasks: detection (identify occupied frequency bins), reconstruction (recover amplitudes), and clutter suppression (remove non-target spectral components). All tasks compare SparseDSP output against Dense FFT reference output.

Exactness definition (Path C): Path C exactness follows the Dense-reference equivalence convention of Section IV-C since no physical ground-truth support exists for the real-payload signals: detected frequency bins exactly match the Dense FFT reference output.

Sparsity estimation: All Path C experiments evaluated SparseDSP with its internal sparsity estimation enabled. Sparsity k was *not* supplied as an external oracle parameter; the system estimated the effective sparsity from each input block at runtime before recovery.

G. Definition of Exactness

Across all three paths, "exact" means equality to the path-specific reference defined in Section IV-C. For Paths A and B, the reference is the known injected support, with the dominant-bin convention for off-grid tones in Path B. For Path C, where no physical ground-truth support is available, the reference is the Dense FFT-derived support under the stated reference-selection rule. Amplitude tolerance and deterministic dispatch behavior are described below.

1) **Support match**: The set of frequency bin indices returned by SparseDSP equals the path-specific reference support defined above.

2) **Amplitude match**: For each matched bin, $|A_{\text{sparse}} - A_{\text{dense}}|/|A_{\text{dense}}| < 10^{-5}$.
3) **Determinism**: Repeated invocations on identical input produce identical output.

This provides a binary pass/fail criterion aligned with the determinism expectations of infrastructure-grade processing chains, and is distinct from physical-target ground-truth detection optimality.

The benchmark task evaluates support and amplitude equivalence to the Dense DFT-plus-selection reference output. This is distinct from packet detection, synchronization, channel estimation NMSE, or BLER metrics used in end-to-end receiver evaluation.

H. System-Level Reproducibility Envelope

This paper specifies the evaluated system-level benchmark conditions, consisting of: (1) benchmark paths and exactness criteria (Sections IV-D to IV-F); (2) tested N and k regimes; (3) impairment families and off-grid interpretation rules; (4) Dense baseline configuration including FFTW calibration (Table V); (5) hardware, software, and execution conditions; and (6) observable outputs in the form of exactness, timing, and regime-dependent behavior. The internal engine-selection mechanism and per-engine implementation details are not part of this envelope. The evaluation protocol and the expected form of system-level behavior under that protocol are fully specified.

V. Application Context: 6G Spectrum Sensing

The following application examples are deployment interpretations derived from the measured Path A/B/C evidence and the literature-grounded parameter mappings. They are not validated end-to-end PHY studies, standards-compliance tests, or post-decoding performance evaluations.

A. Channel Sparsity in Sub-THz and THz Bands

6G channel characteristics at 100 to 300 GHz exhibit natural sparsity due to atmospheric absorption, oxygen and water vapor resonances, and blockage effects [4]. Typical outdoor scenarios show $k = 10$ to 20 dominant paths. Indoor THz imaging exhibits even higher sparsity with $k = 5$ to 10 specular reflections.

This sparse channel structure places the typical sub-THz operating point well within the low-k regime validated by Path A.

B. RIS Beamforming and Spectrum Sensing

Reconfigurable intelligent surfaces require real-time spectrum sensing across 100 to 1000 elements [21], [22]. At $N = 262$K, $k = 10$: SparseDSP at 1.173 ms versus Dense at 4.792 ms (4.18× speedup), indicating lower per-element transform-stage compute cost in the benchmark. Whether this complexity reduction translates to meeting production RIS sensing budgets depends on the target hardware architecture.

C. Massive MIMO Channel Sounding

6G massive MIMO systems with 256 to 1024 antennas require channel sounding across large subcarrier counts [23], [24]. When the channel's delay-domain representation is sparse ($k = 20$ dominant taps across $N = 131$K subcarriers), the transform-stage cost per antenna can be reduced. Whether this translates to end-to-end channel sounding acceleration depends on the estimation algorithm and pilot structure, which are outside the scope of this benchmark.

D. Worked Example: 6G Wideband Spectrum Monitoring

Consider a 6G base station performing continuous wideband spectrum monitoring at sub-THz frequencies:

- Signal length: $N = 262$K samples per sensing window
- Dominant paths: $k = 10$ (typical outdoor sub-THz [4])
- Sensing rate: 1 kHz (1000 windows/second for dynamic spectrum access)
- Latency budget: < 1 ms per window

Dense FFT processing:

- Per-window cost: $N \log_2 N \approx 262\text{K} \times 18 \approx 4.72\text{M}$ FFT work units ($\log_2 262\text{K} \approx 18$)
- Continuous load: 4.72M FFT work units $\times\, 1000$ windows/s ≈ 4.72G FFT work units / s
- On x86 (i9-14900KF): 4.79 ms per window (Table VI), exceeding the 1 ms budget

SparseDSP processing:

- Per-window cost: sublinear in N; measured 1.17 ms on x86 (Table VI)
- Continuous load: ≈ 10 M FFT work units / s (estimated from measured timing ratio; equivalently ~ 50 complex MFLOP/s under the $5N \log_2 N$ convention)
- Exceeds 1 ms budget by 17% on x86 (1.17 ms); coming within range of the target on general-purpose hardware

Result: On the x86 benchmark platform, Dense FFT exceeds the 1 ms budget by 4.8× while SparseDSP approaches it. Whether this translates to meeting production 6G budgets depends on target DSP/FPGA hardware; this example illustrates the algorithmic complexity ratio, not a deployment claim.

This worked example uses measured timing from Table VI. The FLOPs estimates are approximate; actual instruction counts depend on implementation and platform.

VI. Validated Performance Results

Table VII summarizes the scope and coverage of the three validation paths.

A. Timing Methodology

a) Timing methodology.: Each timing point is the median of $n_{\text{iter}} = 100$ wall-clock runs on the platform described in Section IV-A (single-socket Intel Core i9-14900KF, single P-core, FP32, plan and buffer reuse, black-box optimization barriers). Median and 95% bootstrap confidence intervals (1000 resamples) accompany each headline number. Confidence intervals were computed alongside each median value; for table readability, the main-paper timing tables report median values

TABLE VII
BENCHMARK ACCOUNTING SUMMARY

Path	Configs	N range	k range	Conditions
A low-k	4032	16K to 4M	2 to 30	4 placements × 24 channel/SNR/offset
A high-k	960	262K to 4M	150 to 200	4 placements × 24 conditions
B	2160	16K to 1M	16 to 100	6 channels, off-grid, fading
C	102	16K to 1M	4 to 50	6 datasets × 3 tasks

only, while per-iteration data and 95% bootstrap confidence intervals are available as supplementary material. Headline speedups are geometric means across the reported transform-length configurations; raw per-iteration data is available as supplementary material.

B. Path A System-Level Results

SparseDSP achieves 100% exact recovery across all 4992 Path A configurations (4032 low-k + 960 high-k).

Low-k speedup distribution (4032 configurations, $k = 2$ to 30):

- Median speedup: 3.93×
- 10th percentile (P10): 3.03×
- 90th percentile (P90): 24.75×

High-k exactness regime (960 configurations, $k = 100$ to 200):

- SparseDSP achieves 100% exact recovery (960/960) under this exact-support benchmark. A magnitude-only Dense FFT top-k heuristic, applied to the (exact) Dense FFT output under the same benchmark protocol, achieved 0% exact support recovery at these operating points (avg recall 0.7528 against the planted support).
- SparseDSP incurs higher wall-clock cost (median 497.9 ms vs Dense 24.1 ms) but produces correct output where Dense does not.
- This is not a loss region; it is a correctness-for-latency tradeoff.

The Dense FFT transform computes all N bins exactly. The failure at high k is in the top-k magnitude selection heuristic: with $k = 150$ to 200 targets, many bins have similar energy levels, and magnitude-based ranking frequently selects incorrect bins (Dense avg recall 0.7528). SparseDSP's algebraic reconstruction identifies bins through residue structure rather than magnitude ranking, maintaining exact recovery regardless of amplitude distribution.

In production deployment, the dispatch function can prefer Dense FFT at high k when approximate detection is acceptable, or retain SparseDSP when exactness is required.

Speedup scaling with N: Table VIII presents system-level speedup at $k = 10$ across the tested signal-length range.

TABLE VIII
SPARSEDSP SPEEDUP VS DENSE FFT ($k = 10$, SPREAD PLACEMENT). TIMING METHODOLOGY DESCRIBED IN SECTION VI-A.

N	Dense (ms)	SparseDSP (ms)	Speedup
16K	0.257	0.065	4.05×
65K	0.903	0.211	4.34×
262K	4.792	1.173	4.18×
524K	10.013	2.337	4.87×
1M	22.103	3.630	8.91×
2M	46.616	4.385	19.45×
4M	109.802	8.514	38.38×

SparseDSP speedup grows with N because Dense FFT scales as $O(N \log N)$ while SparseDSP's internal engines scale sublinearly in N. The 38.38× speedup at $N = 4$M illustrates the measured speedup at this tested operating point. Note that the 4.05× speedup at $N = 16$K reflects benchmark-lane behavior at $k = 10$; production dispatch policies may conservatively prefer Dense at small N as a practical safety margin, since sparse engine overhead is closer to FFT cost in this regime.

C. Path B System-Level Impairment Robustness

Table IX presents system-level exactness under six channel impairment models:

TABLE IX
PATH B BIN-IDENTIFICATION EXACTNESS: ALGEBRAIC RECOVERY VS DENSE TOP-k HEURISTIC

Channel Model	SparseDSP	Dense Top-k
AWGN	≥99.44%	0.28%
K-distributed clutter	≥99.44%	0.00%
Rayleigh fading	≥99.44%	0.00%
Rician ($\kappa = 3$ dB)	≥99.44%	0.00%
Rician ($\kappa = 10$ dB)	≥99.44%	0.00%
Weibull ($\beta = 12$ dB)	≥99.44%	0.00%

"Dense Top-k" column is the exact-support score of the magnitude-only Dense FFT top-k heuristic; the underlying Dense FFT transform is exact in all rows. Path B is a heuristic stress test, not a head-to-head accuracy comparison.

SparseDSP achieves ≥99.44% exact recovery across all six channel models. Across 2160 total impaired trials (360 per channel model), the remaining failures ($\leq$ 12 cases) concentrate in Rician-κ=3 dB fading at high off-grid offsets (0.5 bin), where deep fading occasionally pushes a target tone below the reconstruction noise floor; these cases would be candidates for the Dense fallback path in the evaluated design.

Interpretation: The 0.00% figure is the exact-support score of the magnitude-only Dense FFT top-k heuristic; the Dense FFT transform itself is exact, and that is not in dispute. The magnitude top-k post-processor fails because channel impairments spread signal energy beyond the k strongest bins. A matched filter or CFAR detector on Dense FFT output would perform differently. The comparison isolates the bin-identification quality of SparseDSP's algebraic approach versus the magnitude-ranking heuristic.

D. Path C Real-Payload Results

Table X presents SparseDSP timing and exactness on six public datasets:

All six datasets achieve exact recovery (1.0) across all three tasks. Each task operates at a fixed (N, k) operating point: detection at $N = 65$K, $k = 25$; reconstruction at $N = 262$K, $k = 51$; clutter suppression at $N = 1$M, $k = 102$. Detection timing ranges from 0.25 ms (deepmimo) to 1.99 ms (argos). Timing variation across datasets at the same (N, k) reflects differences in signal structure, not in transform configuration. The Tier 1 communications-native dataset (deepmimo) achieves the fastest detection and clutter suppression times, consistent with its favorable spectral structure.

Path C results demonstrate transform-stage spectral fidelity on real-world signal structures, not end-to-end wireless receiver performance.

Dispatch reliability: Across all 36 Path C sparse-engine evaluations (6 datasets $\times$ 3 tasks $\times$ sparse engine), SparseDSP exhibited a 0% fallback rate and zero dispatch or runtime errors.

Diagnostic metrics per dataset are presented in Table XI:

Collision metrics are low across all datasets (CM ≤ 0.130), consistent with low measured bin interference during reconstruction. SparseDSP achieves exact recovery (1.0) on all six datasets and all three tasks; the API ratio reflects only task-specific timing variation at each dataset's (N, k) operating point. API > 1 (argos, nuscenes) indicates SparseDSP completed the task faster than Dense; API < 1 (argoverse2, deepmimo) indicates Dense completed faster for that task. These are timing differences, not correctness differences: SparseDSP produces exact output in every case.

Because the internal engines rely on coprime subsampling and algebraic reconstruction, system performance is sensitive to bin collisions and residue uniformity; the following metrics characterize these properties.

Let L denote the number of internal sub-transforms (views), m_ℓ the number of bins in view ℓ, S the recovered support with $|S| = k$, $c_\ell(f, g) = \mathbf{1}[h_\ell(f) = h_\ell(g)]$ the collision indicator for frequencies $f \neq g$ under view ℓ, and $r_\ell[b]$ the number of support elements mapped to bin b in view ℓ:

- **CM** (collision metric): The fraction of frequency pairs that collide across views:

$$\text{CM} = \frac{1}{L\binom{k}{2}} \sum_{\ell=1}^{L} \sum_{\substack{f,g \in S \\ f<g}} c_\ell(f, g) \tag{1}$$

- **U** (residue uniformity): Normalized chi-squared statistic of bin-occupancy deviation:

$$\text{U} = \frac{1}{L} \sum_{\ell=1}^{L} \frac{m_\ell}{k} \sum_{b=0}^{m_\ell - 1} \left(r_\ell[b] - \frac{k}{m_\ell} \right)^2 \tag{2}$$

- **SAT** (saturation): Peak-to-average bin occupancy ratio:

$$\text{SAT} = \frac{1}{L} \sum_{\ell=1}^{L} \frac{\max_b \, r_\ell[b]}{k/m_\ell} \tag{3}$$

- **API** (aggregate performance indicator): Wall-clock speedup ratio:

$$\text{API} = \frac{t_{\text{Dense}}}{t_{\text{SparseDSP}}} \tag{4}$$

  API > 1 indicates SparseDSP completed the task faster; API < 1 indicates Dense completed faster. In both cases, SparseDSP achieves exact recovery (1.0); the API ratio reflects task-specific timing at the operating point's (N, k), not a correctness difference.

The thresholds (CM < 0.15, U < 2.0, SAT < 5.0) are empirical internal acceptance criteria derived from successful reconstruction regions in the benchmark corpus, not universal theoretical bounds.

Path C results demonstrate transform-stage spectral fidelity on real-world signal structures. They support the existence of communications-relevant sparse regimes, but do not establish their prevalence across bands, deployments, mobility conditions, pilot designs, or receiver architectures.

E. Regime Applicability Summary

Table XII maps measured performance to representative wireless application contexts:

F. Front-End Telecom Validation

To assess compatibility with 5G/6G signal processing requirements, we performed a front-end telecom validation using the SparseDSP benchmark harness configured for telecom-grade evaluation. Both SparseDSP and Dense baselines were evaluated under identical conditions:

- Detection probability: $P_d = 1.0$ for both SparseDSP and Dense
- False alarm probability: $P_{fa} = 0.0$ for both
- Tone quality (F1): 1.0 for both
- Average EVM RMS: 0.00139 to 0.00204 for SparseDSP (matching Dense within measurement tolerance)
- Uncoded block error proxy: 0.0 for both

These results represent *pre-FEC front-end integrity evidence*, not post-LDPC coded BLER. The benchmark harness supports an LDPC bridge integration path for coded BLER validation, but decoder-backed BLER evidence has not yet been promoted into the current canonical evidence set. Accordingly, the paper does not claim post-LDPC coded BLER performance.

VII. Production Deployment Considerations

A. Dispatch Policy

SparseDSP selects an engine according to its fixed dispatch policy based on signal parameters. The dispatch function is deterministic: identical inputs produce identical engine selection. The evaluated dispatch behavior has the following observed characteristics:

- **Scale awareness**: In production-oriented dispatch, Dense may be preferred below approximately $N < 32{,}000$ unless the estimated sparsity is extremely low. The benchmark lane reports SparseDSP sparse-engine timing at

TABLE X
PATH C REAL-PAYLOAD RESULTS (ALL TASKS: EXACT = 1.0). TIMING METHODOLOGY DESCRIBED IN SECTION VI-A.

Dataset	Tier	Detection (ms)	Reconstruction (ms)	Clutter (ms)	Exact	Domain Connection
deepmimo	1	0.25	4.39	11.43	1.0	MIMO channel estimation
oxford_radar	2	0.48	6.18	14.24	1.0	Automotive radar (RF-adjacent)
argos	2	1.99	6.18	13.26	1.0	Satellite telemetry (RF-adjacent)
argoverse2	3	0.27	5.28	11.55	1.0	Autonomous driving
nuscenes	3	0.41	6.46	44.37	1.0	Sensor fusion
nexrad	3	0.28	6.13	49.39	1.0	Weather radar

TABLE XI
PATH C DIAGNOSTIC METRICS

Dataset	CM	U	SAT	API
argos	0.130	1.93	4.61	4.01
argoverse2	0.004	0.49	1.24	0.49
deepmimo	0.006	0.56	1.38	0.55
nexrad	0.008	0.72	1.87	0.88
nuscenes	0.012	0.83	2.14	1.12
oxford_radar	0.009	0.61	1.56	0.72

CM = collision metric, U = residue uniformity, SAT = saturation, API = dense-to-sparse performance ratio.

TABLE XII
SPARSEDSP REGIME APPLICABILITY

N Range	Speedup	Application
$\leq$ 16K	k-dependent†	5G NR, WiFi
16 to 65K	4.0 to 4.3×	5G FR2, transition
65 to 262K	4.2 to 4.9×	6G Sub-THz sensing
262K to 1M	4.9 to 8.9×	6G Wideband
1 to 2M	8.9 to 19.5×	Sub-THz monitoring
2 to 4M	19.5 to 38.4×	Beyond 6G
High-k correctness regime:		
Any N, $k > 100$	SparseDSP exact under protocol; magnitude-only Dense top-k inexact	

†SparseDSP can win at low k even at small N; production dispatch may conservatively prefer Dense in this regime.

$(N = 16{,}000, k = 10)$, where sparse execution can still win under the measured configuration; this benchmark behavior is distinct from the conservative production-dispatch heuristic.

- **Sparsity awareness**: At high k relative to N, Dense is preferred. SparseDSP does not force sparse processing when Dense is faster.
- **Fallback guarantee**: If the dispatched sparse engine fails post-detection verification, the system falls back to Dense FFT.

All dispatch parameters were fixed before benchmark execution.

B. 6G Requirement Context

Projected 6G spectrum sensing budgets include < 1 ms end-to-end latency and < 10 W for spectrum processing [1]. On the x86 benchmark platform, SparseDSP at $N = 262$K achieves 1.173 ms (4.18× reduction versus Dense). Whether this complexity reduction translates to meeting production 6G budgets depends on the target hardware architecture (DSP/FPGA/ASIC); this report establishes the algorithmic complexity ratio, not a hardware-specific deployment claim.

C. Workload Budget Analysis: Formal Deployment Interpretation

Section V-D presented a tutorial-level worked example for intuition. This section provides the formal deployment interpretation under stated assumptions. It does not establish that SparseDSP is deployable in real 6G networks; only that the measured complexity ratios produce specific outcomes under the defined workload model.

Scaling across N: The complexity gap widens with signal length. At $N = 262$K: Dense 4.792 ms vs SparseDSP 1.173 ms (4.18×). At $N = 1$M: Dense 22.103 ms vs SparseDSP 3.630 ms (8.91×). At $N = 4$M: Dense 109.802 ms vs SparseDSP 8.514 ms (38.38×). All on x86; absolute feasibility depends on target hardware.

D. Production Caveats

- **Path C staged datasets**: Real-payload datasets are curated subsets at known sparsity, not raw sensor data. They validate spectral signature fidelity, not end-to-end pipeline integration with unknown sparsity estimation.
- **Dispatch scope**: The dispatch function is deterministic and validated at the system level.
- **High-k correctness regime**: At $k = 150$ to 200, SparseDSP achieves exact support recovery but at higher wall-clock cost than the Dense FFT pipeline. The magnitude-only Dense FFT top-k heuristic does not recover the planted support in this regime; the Dense FFT transform itself is exact. Production systems should select based on whether exact support recovery or latency is the priority.
- **Baseline fairness**: Dense is the same-platform transform baseline. This paper does not claim end-to-end PHY optimization parity; real receivers fuse multiple processing stages beyond the transform.

E. System-Level ROI Translation

Benchmark speedup ratios translate to infrastructure-level resource implications as follows. All figures are analytical projections from measured Phase A timing data, not direct infrastructure measurements, and are labeled as deployment interpretations (Table IV).

- **DSP power per cell site**: If the FFT stage consumes a fraction of the baseband DSP budget, a 4× speedup at $N = 262$K frees proportional compute capacity for other processing stages (channel estimation, beam selection,

control signaling), subject to platform-specific power characteristics.

- **Latency margin**: A 4.18× speedup at $N = 262$K reduces the transform stage from 4.79 ms to 1.17 ms on x86, freeing 3.6 ms per sensing window for other receiver processing under the same latency envelope.
- **Spectrum coverage**: The same DSP budget can process proportionally more sensing windows per second, potentially enabling wider bandwidth monitoring at a given sensing rate.
- **RIS panel scaling**: If each RIS element requires one spectrum sensing operation, transform-stage speedup enables more elements to be processed within a fixed latency envelope, subject to controller architecture and scheduling.

These projections illustrate how transform-stage complexity reductions could affect system-level resource allocation. They are not validated infrastructure measurements and are subject to platform-specific re-measurement on production DSP/FPGA hardware.

These projections concern transform-stage complexity reductions only. End-to-end receiver or infrastructure impact would require system-level integration studies beyond the scope of this benchmark.

VIII. Conclusion

This work validates SparseDSP, a regime-adaptive deterministic sparse FFT system, against Dense FFT across operating points relevant to 5G/6G-relevant wideband spectrum sensing. The communications contribution is an operating-region characterization: when channel or signal representations remain sufficiently concentrated after standard receiver transforms, sparse execution is computationally favorable; when that structure degrades, dense execution remains the appropriate default.

In these experiments, SparseDSP estimates sparsity internally; in all reported experiments, including the six real-payload datasets in Path C, the system estimated sparsity from the input signal rather than receiving an oracle parameter. SparseDSP achieves 100% exact recovery across 4992 Path A configurations, ≥99.44% exact across six Path B channel impairment models (versus 0.00 to 0.28% for Dense top-k), and exact recovery (1.0) across six Path C real-payload datasets. At representative low-sparsity operating points ($k = 10$): 4.05× speedup at $N = 16$K, 4.87× at $N = 524$K, and 38.38× at $N = 4$M. At high support sizes ($k = 150$ to 200), SparseDSP achieves 100% exact support recovery where the magnitude-only Dense FFT top-k heuristic does not (heuristic exact rate 0%, avg recall 0.7528 against the planted support; the Dense FFT transform itself remains exact in this regime). SparseDSP incurs higher wall-clock cost but was the only evaluated non-oracle method recovering the exact support under this benchmark protocol.

Dense FFT is the correct choice at 5G NR scales ($N < 32$K) and when approximate top-k detection is acceptable at high k. The evaluated results are most relevant to regimes where wideband receivers produce signals with $k \ll N$: sub-THz spectrum sensing, wideband scanning, compressed channel estimation, and beyond-6G monitoring.

All measurements are detection-only (frequency bin identification), pre-FEC, and within-machine relative on x86. The principal output of this study is the measured system-level behavior under the stated protocol. A companion report validates the same framework for radar, sonar, electronic warfare, and LiDAR [11].

A. Implications for Receiver Design

For receiver designers, the actionable implication is a conditional deployment rule rather than a blanket replacement claim:

1) Use sparse execution only after transforms or estimation stages that already concentrate energy or support (e.g., delay, angle, beam, or other structured domains).
2) Treat sparsity estimation overhead as part of the receiver compute budget, not as a free preprocessing step.
3) Retain a dense fallback when channel spread, interference structure, or estimation uncertainty pushes the workload outside the favorable regime.
4) Expect the largest measured benefit in path-concentrated or bursty subproblems, not necessarily in end-to-end baseband pipelines.

Under this reading, SparseDSP is a regime-selective acceleration strategy for certain 5G/6G receiver blocks rather than a wholesale architectural replacement for dense communication signal processing.

IX. Limitations

1) **Regime-dependent**: SparseDSP does not outperform Dense FFT at all operating points. At high sparsity ($k = 150$ to 200), SparseDSP achieves exact support recovery but at higher wall-clock cost than the Dense FFT pipeline. The magnitude-only Dense FFT top-k heuristic does not match the planted support in this regime (heuristic exact rate 0%); the Dense FFT transform itself is exact. The tradeoff is correctness for latency. At small N (< 32K), Dense is preferred. The dispatch function encodes these boundaries.
2) **Pre-FEC only**: All results are detection-only (frequency bin identification). No amplitude recovery optimization, no post-LDPC 3GPP BLER evaluation, no end-to-end PHY throughput measurement.
3) **Front-end telecom validation scope**: Front-end telecom validation (EVM, tone quality, pre-FEC block-error proxy) is evidenced and matches Dense baseline. Post-LDPC coded BLER validation is structurally supported through the LDPC bridge mechanism but not yet promoted as decoder-backed evidence.
4) **NR PHY non-claim**: This work targets wideband sensing and discovery, not OFDM demodulation replacement at 5G NR scales.
5) **Timing relativity**: All timing comparisons are within-machine relative on a desktop-class x86 CPU (i9-14900KF). Crossover boundaries and absolute timings

will shift on embedded, FPGA, or ASIC platforms; complexity differences may preserve similar trends, although this requires platform-specific measurement.

6) **Signal model**: All Path A and Path B tests use synthetic sparse tones with controlled impairments. No evaluation on realistic OFDM spectra, multipath channels with simultaneous Doppler and delay spread, colored noise, quantization, or adjacent-channel interference. Path C partially addresses this gap with real-world spectral signatures but at curated sparsity levels.
7) **Robustness gaps**: Carrier frequency offset (CFO), phase noise, timing offset, quantization effects, nonstationary channel dynamics, and adjacent-channel interference are out of scope and not evaluated.
8) **Hardware platform**: All results are from a desktop-class x86 CPU. No DSP (Xilinx RFSoC, TI Sitara), wireless baseband processor, or FPGA/ASIC validation.
9) **No external baselines**: This report benchmarks SparseDSP against the Dense FFT baseline. No comparison against external sparse FFT implementations (MIT sFFT, FFAST, etc.) or conventional wireless sparse recovery methods (OMP, compressed sensing) is included; such comparison is deferred to companion algorithm papers [8]–[10].
10) **Dispatch abstraction**: Validation assesses end-to-end system behavior, abstracting implementation-dependent engine selection boundaries.
11) **Sparsity estimation**: While SparseDSP estimates sparsity internally at runtime (and all reported results reflect this deployed behavior), this paper does not separately ablate the accuracy of the internal sparsity-estimation stage versus channel conditions or compare against alternative sparsity-selection heuristics.
12) **Stateless processing**: Because SparseDSP operates independently on each transform, this work does not exploit temporal tracking or cross-frame priors that could improve performance for slowly varying channel conditions.
13) **Path C curated data**: Real-payload datasets are curated subsets chosen for realistic spectral signatures; they do not represent raw unprocessed wireless data streams with arbitrary interference or unknown scene composition.
14) **Benchmark scope**: 4992 Path A configurations, six Path B channel models, and six Path C datasets represent a broad but not exhaustive coverage of the (N, k) parameter space and real-world wireless signal diversity.
15) **No real channel estimation evaluation**: SparseDSP is evaluated as a bin-identification transform, not as an end-to-end channel estimator. Integration with pilot structures, interpolation, and equalization is not evaluated.

ACKNOWLEDGMENT

The authors gratefully acknowledge the collaborative environment at SparseTech that made this research possible.

The theoretical and computational developments presented in this paper are part of the authors' broader research program on deterministic sparse FFT algorithms.

REFERENCES

[1] Ericsson Research, "6g – connecting a cyber-physical world," Ericsson AB, Stockholm, Sweden, Tech. Rep., Feb. 2023. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/co-creating-a-cyber-physical-world

[2] ITU-R, "Framework and overall objectives of the future development of imt for 2030 and beyond," International Telecommunication Union, Geneva, Switzerland, Recommendation ITU-R M.2160-0, Nov. 2023.

[3] Hexa-X Project, "Expanded performance indicators and initial target values for 6g," Hexa-X Consortium, European Commission, Deliverable D1.2, Jun. 2021. [Online]. Available: https://hexa-x.eu/

[4] Nokia Bell Labs, "6g technology components and requirements," Nokia Corporation, Espoo, Finland, Tech. Rep., Apr. 2022. [Online]. Available: https://www.nokia.com/bell-labs/insights/white-papers/

[5] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse fourier transform," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Kyoto, Japan, Jan. 2012, pp. 1183–1194.

[6] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse fourier transform," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, New York, NY, USA, May 2012, pp. 563–578.

[7] X. Li and G. Caire, "A new class of deterministic sparse fft algorithms based on chinese remainder theorem," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4050–4065, Jul. 2020.

[8] A. R. Flouro and S. P. Chadwick, "Crt-based deterministic reconstruction for sparse fft: First fully deterministic $o(k \log k)$ algorithm via gated chinese remainder theorem," *arXiv preprint*, 2026, arXiv ID pending author confirmation.

[9] A. R. Flouro and S. P. Chadwick, "Deterministic sparse fft via coprime decimation and iterative peeling: An $o(k \log n)$ algorithm with exact recovery guarantees," *arXiv preprint*, 2026, arXiv ID pending author confirmation.

[10] A. R. Flouro and S. P. Chadwick, "Keyed-gating sparse fft via structured residue views: Deterministic $o(\sqrt{N} \log k)$ frequency discovery," *arXiv preprint*, 2026, arXiv ID pending author confirmation.

[11] A. R. Flouro and S. P. Chadwick, "Sparsedsp: System-level evaluation of deterministic sparse fft for radar, sonar, and lidar," 2025, companion report.

[12] RustFFT Contributors, "Rustfft: High-performance fft library written in pure rust," 2024, version 6.4.1. [Online]. Available: https://github.com/ejmahler/RustFFT

[13] T. S. Rappaport, Y. Xing, O. Kanhere, S. Ju, A. Madanayake, S. Mandal, A. Alkhateeb, and G. C. Trichopoulos, "Wireless communications and applications above 100 ghz: Opportunities and challenges for 6g and beyond," *IEEE Access*, vol. 7, pp. 78 729–78 757, 2019.

[14] IEEE, "Ieee standard for high data rate wireless multi-media networks – amendment 2: 100 gb/s wireless switched point-to-point physical layer," IEEE, IEEE Std 802.15.3d-2017, Oct. 2017.

[15] I. F. Akyildiz, J. M. Jornet, and C. Han, "Terahertz band: Next frontier for wireless communications," *Physical Communication*, vol. 12, pp. 16–32, Sep. 2014.

[16] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. Information Theory and Applications Workshop (ITA)*, 2019, pp. 1–8.

[17] D. Barnes, M. Gadd, P. Pielniak, P. Newman, and I. Posner, "The Oxford radar RobotCar dataset: A radar extension to the Oxford RobotCar dataset," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[18] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proc. NeurIPS Datasets and Benchmarks Track*, 2023.

[19] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] National Oceanic and Atmospheric Administration (NOAA), National Weather Service, "NEXRAD (next-generation radar) network documentation and data archive," https://www.ncei.noaa.gov/products/radar/next-generation-weather-radar, accessed 2026.

[21] M. Di Renzo *et al.*, "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2450–2525, Nov. 2020.

[22] E. Basar, M. Di Renzo, J. De Rosny, M. Debbah, M.-S. Alouini, and R. Zhang, "Wireless communications through reconfigurable intelligent surfaces," *IEEE Access*, vol. 7, pp. 116 753–116 773, 2019.

[23] A. Alkhateeb, O. E. Ayach, G. Leus, and R. W. H. Jr., "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[24] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive mimo networks: Spectral, energy, and hardware efficiency," *Foundations and Trends in Signal Processing*, vol. 11, no. 3–4, pp. 154–655, 2017.