

REIST Division als Zustandsprinzip

in den GFSK Rechenkernen von
FT8 FT4 und FT2

Rudolf Stepan

OE3SRO | Unabhängiger Forscher, Wien

ORCID: 0009-0004-2842-2579

23. Oktober 2026 | Technischer Forschungsbericht, Version 1.2

Arbeitsfassung ohne externe Begutachtung

Zusammenfassung

Diese Arbeit untersucht die Anwendung des in der REIST-Division beschriebenen persistent zentrierten Restzustands auf die Phasenakkumulation von GFSK-Signalen. Der untersuchte WSJT-X-Custom-Build erzeugt FT8-, FT4- und FT2-Wellenformen durch fortgesetzte Addition begrenzter Phasenschritte. Für den festen Modul 2π genügt nach jedem Schritt höchstens eine Addition oder Subtraktion von 2π . Die Arbeit leitet die Invariante her, dokumentiert die konkreten C++-Aufrufpfade und trennt diese Änderung von Sinustabellen, höherer Rechengenauigkeit, LDPC-Optimierungen, Audioübergabe und Mehrkernverarbeitung.

Eine neue faktorielle Ablation variiert ausschließlich die Phasenreduktion und die Sinusauswertung. Auf einem Intel Core i9-14900K sind bei unveränderter interpolierter Tabelle die Mediane der allgemeinen Restvariante um die Faktoren 5.55 (FT8), 5.24 (FT4) und 4.99 (FT2) größer als jene der begrenzten Korrektur. Für die drei geprüften deterministischen Tonfolgen stimmen die erzeugten Float-Samplewerte dieser beiden Varianten numerisch exakt überein. Die Messung umfasst je Variante 21 zufällig angeordnete Blöcke mit jeweils zehn Wellenformen. Sie belegt einen lokalen Vorteil der Zustandskorrektur gegenüber der gewählten Bibliotheksreferenz, jedoch keine allgemeine Überlegenheit jeder zentrierten Darstellung und keine Verbesserung der Funkempfindlichkeit. Vorhandene Differenz- und Aufnahmetests ergänzen die Messung. Rohdaten, Benchmarkquellen und SHA256-Quellstandnachweise liegen bei.

Schlüsselwörter: zentrierte Restarithmetik, Phasenakkumulator, GFSK, FT8, FT4, FT2, Ablationsstudie, C++, WSJT-X, OpenMP.

1 Fragestellung und Beitrag

Digitale Schmalbandverfahren benötigen sowohl eine korrekte Signalform als auch rechtzeitig abgeschlossene Berechnungen. Diese Anforderungen sind verschieden: Ein schneller erzeugtes Signal ist nicht automatisch leichter zu decodieren. Ziel dieser Arbeit ist deshalb ein nachprüfbarer Nachweis verminderter Rechenkosten bei dokumentierten Signal- und Integrationsprüfungen.

Die REIST-Arbeit von Stepan [1] beschreibt zentrierte Restdarstellungen als Implementierungsprinzip für additive modulare Zustände. Die vorliegende Fallstudie überträgt dieses Prinzip auf einen Gleitkomma-Phasenakkumulator. Untersucht werden drei Fragen: Unter welcher Voraussetzung ist eine einzelne Korrektur ausreichend? An welchen Stellen des Programms wird sie tatsächlich verwendet? Welcher Anteil eines beobachteten Laufzeitgewinns bleibt bestehen, wenn Sinusauswertung und Rechengenauigkeit konstant gehalten werden?

Der Beitrag besteht in dieser mathematischen Zuordnung, ihrer Integration in einen realen Signalpfad und einer kontrollierten lokalen Ablation. Zentrierte Reste, periodische Phasen und

Phasenakkumulatoren sind etablierte Konzepte. Ein digitaler Oszillator mit Phasenregister und Amplitudentabelle ist beispielsweise in der DDS-Literatur beschrieben [3]. Es wird weder eine neue Zahlentheorie noch die erstmalige Erfindung modularer Phasenakkumulation beansprucht.

Der Softwarestand basiert auf WSJT-X 3.0.2-devel im Fork github.com/rudolfstepan/wsjsx. Ausgangspunkt ist Commit `b527b02f9d7be92e26c5c6a59019df518f97725d`; die untersuchten nativen Kerne sind zum Messzeitpunkt lokale, noch nicht commitierte Änderungen. Der Commit allein identifiziert sie daher *nicht*. Das beigefügte `data/source_manifest.json` identifiziert die tatsächlich untersuchten Dateien über SHA256. Die Anwendung trägt weiterhin das Branding `build by OE3SR0`.

2 Signalmodell und Anwendungsgrenzen

FT8 und FT4 verbinden strukturierte Nachrichten, LDPC-Fehlerkorrektur und phasenstetige Frequenzumtastung mit Gaussian-Pulsformung [2]. Die lokale FT2-Integration verwendet den produktiven FT4-Codierpfad mit halbierten Symbolzeit; empfangene FT2-Samples werden für die FT4-Auswertung zeitlich gestreckt. Der historische experimentelle Codec unter `lib/ft2` ist nicht der hier untersuchte Decoderpfad.

Tabelle 1: Wellenformparameter der untersuchten lokalen Implementierung. Die zwei zusätzlichen FT4-/FT2-Symbole sind Rampensymbole.

Parameter	FT8	FT4	FT2
Tonanzahl	8	4	4
Nutz- und Synchronisationssymbole	79	103	103
Ausgegebene Symbole	79	105	105
Samples je Symbol bei 48 kHz	7680	2304	1152
Samples je Wellenform bei 48 kHz	606720	241920	120960
Wellenformdauer in Sekunden	12,64	5,04	2,52
Parameter BT im Generator	2	1	1
Nomineller Audiostart in Millisekunden	500	300	150

Die letzten Zeilen dokumentieren Programmparameter, keine gemessenen USB- oder Transceiverlatenzen. Standard-FT8 ist von Fox- und SuperFox-Sonderpfaden zu unterscheiden. Die hier beschriebene private Qt-Audioübergabe gilt für die lokale Audioausgabe; der separate TCI-Audiopfad behält seine bisherige Übergabe.

Für Tonindizes a_j , eine Symbolgröße L und den normalisierten Pulsparameter $\beta = BT$ wird der Gaussian-Puls in der Implementierung durch

$$p_\beta(t) = \frac{1}{2} [\operatorname{erf}(c\beta(t + 1/2)) - \operatorname{erf}(c\beta(t - 1/2))], \quad (1)$$

$$c = \pi\sqrt{2/\ln 2} \quad (2)$$

gebildet. Aus zeitverschobenen diskreten Pulsen entsteht ein geglätteter Phasenschritt $d[k]$. Mit der Trägerfrequenz f_0 und Abtastrate f_s gilt

$$d[k] = \frac{2\pi}{L} \sum_j a_j p_L[k - jL] + c_{\text{Rand}}[k], \quad (3)$$

$$\delta_n = \frac{2\pi f_0}{f_s} + d[\kappa + n], \quad x[n] = A[n] \sin \phi_n. \quad (4)$$

Hier bezeichnet p_L den diskret abgetasteten Puls. Die FT8-Randverlängerung liegt in c_{Rand} ; $\kappa = L$ lässt das erste Dummy-Symbol weg. FT4/FT2 verwenden $c_{\text{Rand}} = 0$, $\kappa = 0$ und vollständige Rampensymbole. $A[n]$ ist die unveränderte kosinusförmige Ein- und Ausblendung. Die Phase wird vor ihrer Aktualisierung in das Sample umgerechnet.

3 Mathematische Übertragung des REIST Zustandsprinzips

3.1 Zentrierter Repräsentant und persistenter Zustand

Für einen festen Modul $B > 0$ kann ein reeller Wert durch

$$r_B(x) = x - B \left\lfloor \frac{x}{B} + \frac{1}{2} \right\rfloor, \quad r_B(x) \in [-B/2, B/2) \quad (5)$$

zentriert werden. Diese Formel ist eine mathematische Definition. Sie wäre als wiederholte Berechnung im inneren Akkumulator gerade die Arbeit, die vermieden werden soll. Entscheidend ist, den bereits zentrierten Rest als Zustand zu erhalten und nur kleine Änderungen anzuwenden.

Satz 1 (Eine Korrektur bei begrenztem Update). *Es seien $r_n \in [-B/2, B/2)$ und $u_n \in [-B/2, B/2]$. Für $s_n = r_n + u_n$ liefert*

$$r_{n+1} = \begin{cases} s_n - B, & s_n \geq B/2, \\ s_n + B, & s_n < -B/2, \\ s_n, & \text{sonst} \end{cases} \quad (6)$$

einen zu s_n kongruenten Zustand in $[-B/2, B/2)$. Höchstens eine Korrektur ist erforderlich.

Beweis. Die Voraussetzungen implizieren $s_n \in [-B, B)$. Falls $s_n \geq B/2$, liegt $s_n - B \in [-B/2, 0)$. Falls $s_n < -B/2$, liegt $s_n + B \in [0, B/2)$. In allen übrigen Fällen ist s_n bereits im Zielintervall. Addition und Subtraktion von B ändern die Restklasse nicht. Induktion erhält die Invariante für beliebig viele Schritte mit derselben Voraussetzung. $\square$

Dies ist die für den Signalpfad relevante Übertragung des additiven REIST-Zustandsprinzips [1, Abschnitte 3 und 5]. Die Ausgangsarbeit behandelt überwiegend ganzzahlige Arithmetik. Der obige Beweis ist ausdrücklich ein Beweis über reelle Zahlen; Aussagen über den konkreten Gleitkommacode erfordern zusätzlich numerische Prüfung.

3.2 Spezialisierung auf die Phase

Mit $B = T = 2\pi$, $r_n = \phi_n$ und $u_n = \delta_n$ wird

$$\phi_{n+1} = \phi_n + \delta_n - k_n T, \quad k_n \in \{-1, 0, 1\}. \quad (7)$$

Die Korrektur verändert weder $\sin \phi_n$ noch $\exp(i\phi_n)$ in exakter Arithmetik. Da der Generator keine fortlaufende Anzahl ganzer Umdrehungen benötigt, wird k_n nicht gespeichert. Bei Bedarf ließe sich dieser Quotientenanteil getrennt akkumulieren.

Der Produktionskern begrenzt einen ungewöhnlich großen Einzelschritt zuvor mit `std::remainder`. Danach folgt die einzelne Korrektur:

`std::remainder` kommt damit nicht prinzipiell aus dem gesamten Generator heraus: Der Schutzpfad bleibt erhalten. Für die geprüften Audiofrequenzen und Tonindizes liegen die Schritte bereits innerhalb der Grenze. Der feste Modul und die kleine Änderung erlauben hier die günstigere Aktualisierung.

Listing 1: Tatsächlicher Phasenpfad in `ft8_wave.cpp`, Zeilen 104–110.

```

104 double step = increments[(ft4 ? 0 : nsps) + k] + base;
105 // Handle arbitrary caller frequencies before entering the bounded update.
106 if (step > pi || step < -pi) step = std::remainder(step, twoPi);
107 phase += step;
108 if (phase >= pi) phase -= twoPi;
109 else if (phase < -pi) phase += twoPi;
110 }

```

Das Programm startet mit $\phi_0 = 0$, verwendet `double` für Phase und Schritte und gibt Float-Samples aus. Es verwendet eine bedingte Form mit `if/else if`. Eine verzweigungsfreie, konstantzeitliche oder innerhalb eines einzelnen Phasenstroms SIMD-parallele Implementierung wird nicht behauptet. Die Rekurrenz besitzt eine Abhängigkeit vom vorherigen Sample.

3.3 Numerische Fehler und periodischer Abstand

Ein kleiner Phasenfehler nahe einer Intervallgrenze kann als ungerichtete Differenz fast 2π erscheinen. Sinnvoll ist der periodische Abstand

$$d_T(a, b) = \min_{k \in \mathbb{Z}} |a - b - kT|. \quad (8)$$

Für den Sinus ist die resultierende Amplitudenabweichung höchstens dieser kleine periodische Phasenabstand. Der Beweis von Satz 1 beseitigt jedoch keine Gleitkommarundungen. In der Implementierung werden eine Addition, gegebenenfalls eine Korrektur und die Darstellung der Konstanten gerundet. Die Prüfung muss deshalb auch das resultierende Signal betrachten.

Die Tabelle enthält $N = 65536$ gleichmäßig verteilte komplexe Stützstellen und eine zusätzliche Endstützstelle. Lineare Interpolation liefert komponentenweise in exakter Tabellenarithmetik den bekannten Interpolationsfehler

$$|e_{\text{interp}}| \leq \frac{h^2}{8}, \quad h = \frac{2\pi}{N}, \quad \frac{h^2}{8} \approx 1,15 \cdot 10^{-9}. \quad (9)$$

Hinzu kommen Rundungen der Float-Tabelle und der Interpolation. Diese Schranke ist deshalb keine vollständige Fehlergarantie des Programms. Die neue Messung erfasst die tatsächlich ausgegebenen Samplewerte separat.

4 Implementierung im untersuchten Quellstand

4.1 Gemeinsamer Wellenformkern

Die Routine `ft8_native::waveform` in `lib/ft8/native/ft8_wave.cpp` erzeugt FT8 und über `FrameShape::Ft4` auch FT4/FT2. Die Auswahl bestimmt Ausgangslänge, Dummy-Randverlängerung, Zugriff auf die Schrittfolge und Rampenlänge. FT8 verwendet ungefähr $L/8$ Samples pro Rampe; FT4/FT2 verwenden L . Ein nach einer kurzen FT2-Wellenform verbleibender größerer Ausgangspuffer wird mit Nullen gefüllt. So bleiben keine Samples einer vorherigen FT4-Erzeugung übrig.

Die C-ABI-Einstiege `gen_ft8wave_`, `gen_ft8wavevar_` und `gen_ft4wave_` erhalten die Aufrufstruktur der bisherigen Routinen. Damit verwenden sowohl die normale TX-Erzeugung als auch Referenzsignale für Subtraktionspfade den gemeinsamen Kern. Die ABI wurde hier nur für den gemeinsamen MinGW-GCC/GFortran-Werkzeugsatz geprüft.

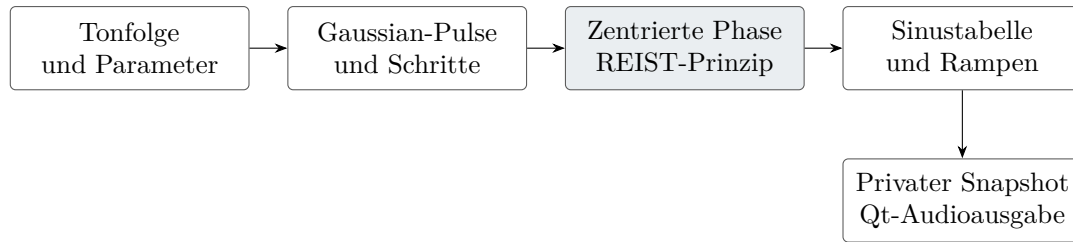


Abbildung 1: Lokaler TX-Datenpfad. Nur die hervorgehobene Zustandsaktualisierung wird in dieser Arbeit als unmittelbare REIST-Anwendung zugeordnet.

4.2 Cache und Besitz der Arbeitsdaten

Der Pulscache enthält unveränderliche Objekte für höchstens acht Kombinationen aus L und BT . Ein Mutex schützt die Suche und das einmalige Einfügen; `shared_ptr` erhält die Lebensdauer bei Verdrängung. Der Mutex liegt außerhalb der Sample-Schleife. Phase und Schrittarrray gehören jedem Aufruf. Auch die Sinustabelle ist nach ihrer einmaligen Initialisierung unveränderlich. Dies ist eine Maßnahme zur Wiederverwendung und Thread-Sicherheit, kein weiterer mathematischer REIST-Nachweis.

Eine erste lokale Variante mit veränderlichem `thread_local`-Pulscache zeigte einen sporadischen Heapfehler im Parallel-Stresstest und wurde ersetzt. Aus diesem Befund wird keine allgemeine Fehlerhaftigkeit von `thread_local` oder eines bestimmten Compilers abgeleitet. Dokumentiert und ausgeliefert ist die anschließend wiederholt geprüfte unveränderliche Cacheform.

4.3 Quelltextzuordnung

Tabelle 2 nennt überprüfbare Einstiegspunkte. Die Zeilen beziehen sich auf den per Manifest fixierten Stand und können nach weiteren Änderungen wandern. Funktionsnamen und Hashes sind deshalb die wichtigeren Anker.

5 Weitere Optimierungen und ihre getrennte Attribution

5.1 LDPC und Nachrichtenkodierung

Die native Kodierung packt 91 Informations- und CRC-Bits in zwei 64-Bit-Wörter. Jede der 83 Paritätszeilen wird durch Bitmasken, XOR und Parität ausgewertet. Die Generator- und Prüfmatrizen stammen aus den bestehenden WSJT-X-Tabellen; `tools/Generate-Ft8NativeTables.py` erzeugt die C++-Konstanten. Formatierte interne Fortran-Ein-/Ausgabe entfällt im Encoder. Die BP-Arbeitsarrays sind pro Aufruf getrennt.

LDPC arbeitet bereits über $GF(2)$. Eine zusätzliche zentrierte ganzzahlige Restdarstellung bietet hier keinen nachgewiesenen Vorteil. Die beobachteten Encoder- und BP-Gewinne werden folglich nicht REIST zugerechnet. Der produktive Hybrid-BP/OSD-Pfad enthält weiterhin Fortran; der Austausch einzelner BP-Routinen ist keine vollständige C++-Portierung aller Decodierverfahren.

5.2 FT4 und FT2 Synchronisationssuche

Der Suchkern bewertet Frequenz- und Zeitlagen über vier komplexe Costas-Korrelationen. Die frequenzkorrigierte Referenz wird einmal je Frequenzhypothese berechnet. Große Raster mit mindestens 256 Punkten können unabhängige Frequenzzeilen verteilen. Das Workerbudget ist

Tabelle 2: Implementierungsstellen und Zuordnung der Änderung.

Ort	Inhalt und Einordnung
<code>lib/ft8/native/ft8_wave.cpp</code> , <code>waveform</code> , Z. 67–125 Dieselbe Datei, <code>pulseFor</code> , Z. 50–64 Dieselbe Datei, Z. 128–152	Gemeinsamer GFSK-Kern; REIST-bezogene Schrittbegrenzung und Phasenkorrektur Z. 104–110. Unveränderlicher Pulscache; reine Implementierungs- und Parallelitätsmaßnahme. Bestehende Fortran-Aufrufpfade auf die C++-ABI abbilden.
<code>widgets/mainwindow.cpp</code> , Z. 8063 und 8099	FT8-/FT4-/FT2-Snapshot nach Erzeugung an den lokalen Audio-Thread senden.
<code>Modulator/Modulator.cpp</code> , Z. 91–93, 184–185 und 302–356	Auswahl des privaten Snapshots, Vermeiden gemeinsamer TX-Lesezugriffe und Begrenzung am Frameende.
<code>lib/ft4/native/ft4_sync.cpp</code> , <code>search</code> , Z. 75–114 <code>lib/ft4_decode.f90</code> , Z. 272–273	Native Synchronisationssuche, größere Raster mit bis zu acht Workern; eigenständige SMP-Optimierung. Geordnete Decodersteuerung ruft die neue native Grob-/Feinsuche auf.
<code>lib/ft8/native/ft8_ldpc.cpp</code> , <code>encode</code> und <code>bp</code>	Gemeinsamer LDPC-Encoder und Standard-/SMP-BP-Kerne; keine als REIST beanspruchte Änderung.
<code>lib/ft8/ft8_thread_budget.f90</code> ; <code>lib/decoder.f90</code> , Z. 261–263 <code>Audio/soundout.cpp</code> ; <code>widgets/TxQualityDialog.hpp</code>	FT8-Workerzahl auf CPU- und OpenMP-Laufzeitgrenzen begrenzen. Audiofehlerbehandlung und Diagnose für Audiostart, Puffer und optionalen CAT-ALC-Wert.

höchstens acht, berücksichtigt zwei logische CPUs als Reserve und beachtet OpenMP-Grenzen. Bereits aktive Parallelbereiche erhalten kein zusätzliches verschachteltes Team.

Jede Frequenzzeile speichert ihr eigenes bestes Ergebnis. Die anschließende serielle Reduktion verwendet die ursprüngliche Frequenzreihenfolge und strikte Größer-Vergleiche. Bei Gleichstand bleibt dadurch der zuerst besuchte Punkt erhalten. Kleine Feinsuchen bleiben seriell. Kandidaten-abhängige Subtraktionsdurchläufe werden nicht ungeordnet parallel ausgeführt. Diese Maßnahmen beruhen auf Wiederverwendung und unabhängigen Hypothesen, nicht auf REIST-Arithmetik.

5.3 Audio und FT8 Mehrkernsteuerung

Die lokale Qt-Audioausgabe erhält einen `QVector<float>`-Snapshot über die Audio-Thread-Warteschlange. Eine Änderung des GUI-seitigen Puffers verändert den vorhandenen Snapshot durch Copy-on-write nicht. Der Modulator liest kein Sample hinter dem Frameende. Ein ungültiges Audioformat darf keinen alten Stream weiterverwenden; Audiofehler stoppen TX vor dem modalen Fehlerdialog.

Die vorhandene FT8-SMP-Architektur behält maximal zwölf Worker. Deren Auswahl berücksichtigt `OMP_NUM_THREADS`, `OMP_THREAD_LIMIT` und deaktivierte aktive Ebenen. Dies verhindert den geprüften Hängerfall bei einem kleineren erlaubten Team als angeforderten Abschnittsworkern. `OMP_WAIT_POLICY=PASSIVE` wird ohne Benutzervorgabe als Standard gesetzt. Die Spezifikation beschreibt diese Variable als Hinweis an die Laufzeit; das konkrete Warteverhalten ist implementierungsabhängig [4].

Der gemeinsame TX-Diagnosedialog im Menü **View** zeigt Softwaretiming, Audiofrequenz, Dämpfung und optional CAT-ALC auf backendabhängiger relativer Skala. Er misst keine abgestrahlte Signalqualität und regelt keinen Transceiverpegel automatisch. Er ist über den Eintrag **TX quality** für FT8, FT4 und FT2 erreichbar.

6 Experimenteller Aufbau

6.1 Messsystem und kontrollierte Varianten

Die neue Ablation wurde am 15. Oktober 2026 auf einem Intel Core i9-14900K mit 24 physischen Kernen und 32 logischen Prozessoren unter Windows 11 Enterprise, Build 26300, ausgeführt. Der Werkzeugsatz ist MinGW GCC/GFortran 15.2.0. Der eigenständige Harness verwendet `-O3 -fno-fast-math -ffp-contract=off`, keine Linkzeitoptimierung und kein `-march=native`. Er misst die Wellenformerzeugung seriell. OpenMP-Suchzeiten sind ein separater Validierungslauf und gehen nicht in die Ablation ein.

Die CPUs wurden nicht auf feste Kerne gebunden; Turbo, Energieregulierung und Betriebssystemhintergrundlast wurden nicht kontrolliert. Insbesondere auf der heterogenen CPU kann der Scheduler die absoluten Zeiten beeinflussen. Gemeldet werden daher deskriptive Verteilungen, keine allgemeingültigen Prozessorfaktoren oder inferenzstatistischen Signifikanzen.

Die Varianten bilden ein 2×2 -Design:

Tabelle 3: Ablationsfaktoren. C bezeichnet die begrenzte Korrektur, G die allgemeine Restberechnung, L die interpolierte Tabelle und S die direkte Sinusfunktion.

Variante	Phasenaktualisierung	Amplitudenauswertung
C-L	Produktionskorrektur	Interpolierte Tabelle
G-L	<code>std::remainder(phase+step,T)</code>	Dieselbe Tabelle
C-S	Produktionskorrektur	<code>std::sin(double)</code>
G-S	<code>std::remainder(phase+step,T)</code>	<code>std::sin(double)</code>

Die drei nicht produktiven Varianten entstehen mechanisch aus der aktuellen Datei `ft8_wave.cpp`. Der Generator prüft erwartete Quelltextblöcke und bricht bei abweichender Vorlage ab. Für G-L ändert er ausschließlich die normale Phasenaktualisierung; Schutzpfad, Datentypen, Pulse, Cache, Interpolation und Rampen bleiben gleich. Die C++-Sprache selbst wird somit für den entscheidenden Vergleich konstant gehalten.

Als zusätzlicher Kontext wird die ursprüngliche Fortran-Routine im selben eigenständigen Harness gemessen. Diese Referenz verwendet wie bisher einfache Phasengenauigkeit und ihre ursprüngliche Sinusauswertung. Der Harness übernimmt nicht alle Zusatzflags des gesamten WSJT-X-Projekts, etwa `-funroll-all-loops` und `-fbounds-check`. Der Fortran-Vergleich ist deshalb weder ein Sprachbenchmark noch ein identischer Nachbau des Produktions-Buildbenchmarks. Für die REIST-Zuordnung ist der kontrollierte Vergleich G-L gegen C-L maßgeblich.

6.2 Arbeitslast und Stichprobeneinheit

Alle drei Modi laufen bei 48 kHz und $f_0 = 1855$ Hz mit den Parametern aus Tabelle 1. Der Zufallsgenerator `std::mt19937` mit Seed 991 erzeugt je Modus eine deterministische Folge gültiger Tonindizes. Diese Folge dient der Kernmessung; sie muss keine gültige codierte FT-Nachricht bilden. Gültige Nachrichten und Aufnahmen sind Gegenstand der getrennten Integrationstests.

Nach drei ungemessenen Aufwärmufrufen je Variante werden 21 Blöcke gemessen. Innerhalb jedes Blocks wird die Reihenfolge der fünf Varianten mit Seed 20261015 zufällig permutiert. Ein Messwert ist die mit `std::chrono::steady_clock` gemessene Dauer von zehn vollständigen Erzeugungen, geteilt durch zehn. Speicher für Ausgaben wird vorher reserviert; interne Arbeitsallokationen des Generators bleiben Teil der Messung. Eine beobachtbare Prüfsumme verhindert ein folgenloses Entfernen der Aufrufe.

Damit entstehen $3 \cdot 5 \cdot 21 = 315$ gespeicherte Messwerte für 3150 gemessene Wellenformen. Die 21 Blockmittelwerte sind die Auswertungseinheiten, nicht 3150 unabhängige Versuche. Ausgegeben werden Median und empirisches Interquartilsintervall mit linear interpolierten Quartilen. Die Blöcke stammen aus einem Prozess und einer Plattform; ihre Unabhängigkeit wird nicht behauptet.

7 Ergebnisse der neuen Ablation

7.1 Laufzeiten und zuordenbarer Phaseneffekt

Tabelle 4 enthält ausschließlich die neu erhobene Messreihe. Sie ersetzt nicht die früheren explorativen Kernelzeiten aus der Implementierungsarbeit; unterschiedliche Flags, Last und Messhüllen können andere absolute Werte erzeugen.

Tabelle 4: Millisekunden je vollständiger reeller TX-Wellenform. Je Zeile 21 Blockmittelwerte; die Klammer enthält Q1 und Q3, kein Konfidenzintervall.

Modus	Variante	Median in ms	Interquartilsintervall in ms
FT8	Fortran	40.137	[39.591; 41.087]
FT8	G-S	42.997	[42.194; 44.406]
FT8	C-S	27.058	[26.536; 27.290]
FT8	G-L	18.419	[18.119; 18.670]
FT8	C-L	3.321	[3.229; 3.472]
FT4	Fortran	15.661	[15.491; 15.758]
FT4	G-S	16.837	[16.549; 17.090]
FT4	C-S	10.816	[10.725; 10.968]
FT4	G-L	7.440	[7.345; 7.628]
FT4	C-L	1.421	[1.406; 1.455]
FT2	Fortran	7.816	[7.762; 8.045]
FT2	G-S	8.297	[8.219; 8.386]
FT2	C-S	5.395	[5.361; 5.583]
FT2	G-L	3.735	[3.582; 3.794]
FT2	C-L	0.748	[0.624; 0.764]

Für die Zuordnung wird der Faktor

$$S_{\text{Phase,L}} = \frac{\text{Median}(t_{\text{G-L}})}{\text{Median}(t_{\text{C-L}})} \quad (10)$$

verwendet. Er beträgt 5.55, 5.24 und 4.99 für FT8, FT4 und FT2. Dies sind Verhältnisse der Mediane, nicht der Median gepaarter Verhältnisse. Der Vergleich hält Sinustabelle, Pulsrechnung, Gleitkommapräzision und Ausgaberrampen konstant.

Tabelle 5: Laufzeitverhältnisse im neuen Harness. Die letzte Spalte enthält mehrere gleichzeitig veränderte Faktoren und ist kein reiner REIST-Effekt.

	G-L / C-L	G-S / C-S	Fortran / C-L
FT8	5.55	1.59	12.09
FT4	5.24	1.56	11.02
FT2	4.99	1.54	10.44

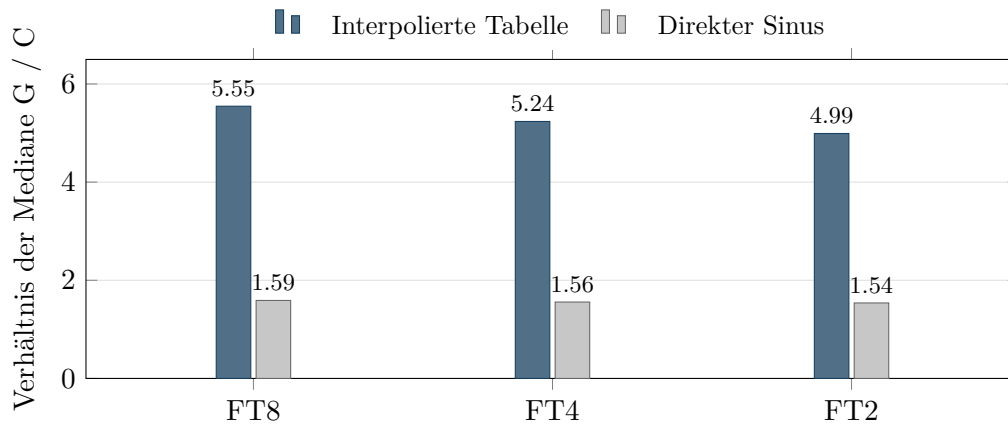


Abbildung 2: Der Phasenvorteil hängt vom übrigen Rechenkern ab. Mit teurer direkter Sinusauswertung ist der relative Einfluss der Reduktion kleiner. Gezeigt sind deskriptive Verhältnisse ohne Unsicherheitsbalken.

Der Unterschied zwischen den beiden mittleren Spalten zeigt, warum eine ungeprüfte Multiplikation einzelner Speedups unzulässig ist. Reduktion und Sinusauswertung teilen sich eine Gesamtlaufzeit mit weiteren Kosten. Wenn der direkte Sinus dominiert, wird eine gleichartige Einsparung der Phasenreduktion relativ weniger sichtbar. Ein allgemeiner, vom Kontext unabhängiger Faktor für REIST lässt sich aus dieser Fallstudie nicht ableiten.

7.2 Numerische Vergleichswerte

Für G-L gegen C-L wurde bei allen drei geprüften Tonfolgen eine maximale absolute Sampedifferenz von null gemessen. Diese Aussage betrifft die numerischen Float-Werte der aufgezeichneten Fälle, nicht eine formale Bitidentitätsgarantie für alle Eingaben oder Randfälle der Bibliotheksfunktion.

Die direkte Sinusauswertung unterscheidet sich von C-L maximal um $5,9605 \cdot 10^{-8}$, bei RMS-Abweichungen um $2,2 \cdot 10^{-8}$. Die ursprüngliche Fortran-Wellenform hat gegenüber C-L größere maximale Abweichungen: $7,6442 \cdot 10^{-3}$ (FT8), $5,0338 \cdot 10^{-3}$ (FT4) und $1,1280 \cdot 10^{-3}$ (FT2). Hier ändern sich unter anderem Phasenpräzision und Pulsrechnung; der alte Generator ist kein analytischer Genauigkeitsstandard. Diese Abweichungen werden deshalb weder als Kompatibilitätsfehler noch als Beleg besserer RF-Signalqualität interpretiert. Die vollständigen Werte stehen in `data/numerical.csv`.

8 Korrektheitsprüfung und Integrationsnachweise

Die Daten zum lokalen Implementierungsstand [5] enthalten drei Evidenzebenen. Erstens prüft die neue Ablation die kontrollierten Varianten. Zweitens vergleichen die bestehenden nativen Tests Wellenformen und Fehlerkorrektur mit den aufrufbar erhaltenen Originalroutinen. Drittens prüfen WAV-Aufnahmen, synthetische Nachrichten und ein Speicherausgang die Integration. Die ursprünglichen Fortran-Dateien bleiben für Testziele erhalten und werden für die ersetzten Routinen nicht mehr in die Produktionsbibliothek aufgenommen.

Die FT4-/FT2-Aufnahmen bestehen mit einem Worker, acht Workern, acht angeforderten bei zwei erlaubten Threads sowie deaktivierten aktiven Teams. FT8 besitzt zusätzliche synthetische Tests für 1/2/4/Auto/8/12 Worker. Der Unterschied von 20 zu 19 Nachrichten zwischen den beiden FT8-Pfaden bestand bereits in den Referenzen; er ist kein durch die neue Implementierung

Tabelle 6: Umfang der vorhandenen und erneut geprüften Regressionen. Ein bestandenes endliches Testset ersetzt keinen universellen Korrektheitsbeweis.

Bereich	Prüfung und Ergebnis
Phasen- und GFSK-Kerne	Reelle FT8-Wellenformen bei 12/48 kHz; FT4/FT2 mit 288/576/1152/2304 Samples je Symbol, vier Frequenzen und reeller/komplexer Ausgabe; analytischer Träger als unabhängige Kontrolle.
Numerische Grenzwerte	Endliche Samples, Amplitudenbegrenzung, Rampen, Nulltail und Trägerabweichung höchstens $2 \cdot 10^{-6}$ im geprüften Innenbereich. Letzteres ist eine Testtoleranz, kein gemessener Maximalwert dieser Tabelle.
LDPC	Je 1000 Zufallsnachrichten für beide Encoder; 2000 verrauschte BP-Vergleiche mit und ohne AP-Maske; in den Tests gleiche Codewörter, Erfolgszustände und Iterationen.
Nebenläufigkeit	Zwölf gleichzeitige FT8-Wellenformaufrufe, gemischte FT8-/FT4-/FT2-Cachewechsel und vier gleichzeitige native Suchen mit getrennten Eingaben.
FT4-/FT2-Synchronisation	30 deterministische Rauschpuffer, Positionen an Puffergrenzen, Grob-/Feinsuche, Maximum und Gleichstände gegen Original.
Lokaler Audiopfad	13 QtTest-Ergebnisse einschließlich Initialisierung/Ende: Snapshot-Besitz, exakte Samplezahl und modusspezifischer verspäteter Start für FT8/FT4/FT2, Nullgerät, ALC-Zustand und Dialogsteuerung.

Tabelle 7: Erhaltene Nachrichten in den festgelegten Aufnahmen. Verglichen wird die Nachrichtenmenge je Decoderpfad, nicht nur die Anzahl.

Modus/Pfad	Aufnahme	Original	Neuer Stand
FT8 Standard	210703_133430.wav	20	20
FT8 SMP	dieselbe Aufnahme	19	19
FT4	000000_000002.wav	19	19
FT2	260301_170123.wav	6	6
FT2	260301_085945.wav	2	2

erzeugter Verlust.

Ein aktueller Aufruf der nativen Tests ist in `data/native_validation.log` gespeichert. Er meldet unter anderem 956 erfolgreiche Fälle innerhalb der 2000 absichtlich unterschiedlich stark verrauschten BP-Eingaben; die übrigen Fälle stimmen im Fehlschlag mit der Referenz überein. Die dabei zusätzlich ausgegebenen Kernelzeiten sind explorative Einzelmessungen und werden nicht mit den 21-Block-Ablationsdaten zu einer künstlich größeren Stichprobe zusammengeführt.

9 Diskussion und Grenzen der Aussage

Die kontrollierte Beobachtung lautet: Das Erhalten eines begrenzten Phasenzustands ist in diesem C++-Kern gegenüber der gewählten allgemeinen Restfunktion günstiger. Dies passt zur erwarteten Verkürzung des Rechenwegs. Die Messung weist jedoch keine konkrete Anzahl eingesparter CPU-Instruktionen, Branch-Mispredictions oder Divisionsinstruktionen nach; eine solche mikroarchitektonische Zuordnung würde Disassemblierung und Hardwarezähler benötigen. Der Ausdruck *divisionfrei* meint hier den normalen Aktualisierungspfad auf Quelltextebene, nicht einen nachgewiesenen bestimmten Maschinencode. Auch der Schutzpfad und andere Generatoroperationen bleiben bestehen.

Zudem ist Zentrierung nicht die einzige Möglichkeit für eine beschränkte Korrektur. Bei rein positiven begrenzten Schritten kann auch ein persistenter Zustand in $[0, T)$ mit einer einzelnen Subtraktion arbeiten. Ein optimierter Festkomma-DDS kann den natürlichen Wortüberlauf nutzen. Diese Alternativen sind hier nicht vermessen. Die Referenz G-L ist eine allgemeine Bibliotheksrestberechnung, kein bestmöglicher Vertreter aller anderen Oszillatorarchitekturen. Die nützliche REIST-Zuordnung besteht im bewusst erhaltenen, durch eine Invariante abgesicherten Zustand, nicht in einer beanspruchten Exklusivität zentrierter Intervalle.

Die vollständigen Wellenformgewinne gegenüber dem alten Code umfassen mehrere Eingriffe. C++ ersetzt auch keine zuvor interpretierte Sprache: Fortran wurde bereits zu nativem Maschinencode übersetzt. Die Aussage *C++ ist um einen bestimmten Faktor schneller als Fortran* wäre aus diesen Versuchen nicht ableitbar. Ebenso wenig lassen sich aus der GF(2)-Kodierung oder der SMP-Suche eigenständige REIST-Gewinne ableiten.

Ein Kernelgewinn kann im Gesamtprogramm klein bleiben. Ist der ursprüngliche Anteil des beschleunigten Kerns p , dann ergibt sich unter sonst gleichen Bedingungen das einfache Kostenmodell

$$S_{\text{gesamt}} = \frac{1}{(1 - p) + p/S_{\text{Kern}}}. \quad (11)$$

Der Anteil p wurde hier nicht profiliert. Insbesondere wird die TX-Wellenform nur zu bestimmten Anlässen erzeugt; eine große relative Kerneinsparung bedeutet keine entsprechend große Erhöhung des gesamten Decoderdurchsatzes. Bei SMP kommen Scheduling- und Wartekosten hinzu.

Die Studie ist auf eine Plattform, einen Compilerstand und die beschriebenen Eingaben begrenzt. Es fehlen Wiederholungen auf weiteren Prozessoren, fest gebundene Kerne, Energie- und Zyklusmessungen, eine umfassende Monte-Carlo-Empfindlichkeitskurve sowie Spektral- und Transceiverprüfungen. Die analytische Trägerkontrolle prüft einen Teilaspekt der Genauigkeit, kein vollständiges Emissionsspektrum. Die vorhandenen Aufnahmen weisen keinen Verlust innerhalb dieses Materials nach; sie beweisen keine unveränderte Sensitivität für sämtliche Kanäle.

Es wurde kein RF-Sendetest am FT-991A durchgeführt. Aussagen über SNR-Gewinn, Reichweite, Nebenwellen oder bessere Empfangbarkeit bei Gegenstationen wären mit diesen Daten unbelegt. Die praktische Verbesserung ist zunächst weniger Rechenaufwand sowie überprüfte Programm- und Audiozustände.

10 Schlussfolgerung

Die REIST-bezogene Zustandsinvariante lässt sich unmittelbar auf den GFSK-Phasenakkumulator übertragen: Ein fester Modul und begrenzte Schritte ersetzen im normalen Samplepfad die allgemeine Restberechnung durch höchstens eine Korrektur. Der untersuchte Quellstand verwendet denselben C++-Kern für FT8, FT4 und FT2 und erhält deren unterschiedliche Rahmen- und Rampenformen.

Die kontrollierte lokale Ablation zeigt bei gleicher Tabellenstrategie Faktoren von 5.55, 5.24 und 4.99 gegenüber der gewählten allgemeinen Restfunktion. Genau diese Zuordnung ist der wissenschaftlich belegte Leistungsbefund dieser Fallstudie. Weitere Gewinne durch Tabelle, Cache, LDPC, Audioübergabe und SMP sind getrennte Implementierungsbeiträge. Ein umfassender Vorteil gegenüber allen Phasenakkumulatoren oder eine bessere Funkempfangbarkeit wird nicht beansprucht.

A Reproduktion und Artefaktstruktur

Die beigelegte Quell- und Datenfassung enthält das Manuskript, die 315 rohen Blockmessungen, numerische Vergleichswerte, Auswertungscode und die konkreten Kernquellen. Private Konfigurationen, Logbücher und die externe REIST-Original-PDF werden nicht in das Reproduktionspaket aufgenommen.

Listing 2: Messung und Satz aus dem Repository-Hauptordner.

```
1 cmake -S docs/papers/reist-fts/experiment ^
2   -B .local/reist-paper-build -G Ninja ^
3   -DCMAKE_BUILD_TYPE=Release
4 cmake --build .local/reist-paper-build --parallel 4
5 .local\reist-paper-build\reist_paper_benchmark.exe ^
6   docs/papers/reist-fts/data/timings.csv ^
7   docs/papers/reist-fts/data/numerical.csv
8 python docs/papers/reist-fts/experiment/analyze.py
9 powershell -File docs/papers/reist-fts/Build-Paper.ps1
```

Die Fortsetzungszeichen ^ im Listing gehören zur Windows-cmd-Syntax. In PowerShell können die Befehle jeweils in einer Zeile eingegeben werden. Alternativ führt `Build-Paper.ps1 -Remeasure` die Messung und den Satz zusammen aus. Eine neue Messung verändert naturgemäß die Datentabellen; die beigelegte Messreihe ist als historische Originalfassung zu erhalten.

Listing 3: Zusätzliche Integrationstests im vollständigen Custom-Repository.

```
1 .\tools\Build-Custom.ps1
2 .\tools\Test-TxQuality.ps1
3 .\tools\Test-Ft2.ps1 -SamplesDirectory .local\ft2-reference\samples
```

Die beiden FT2-Aufnahmen sind separat bereitzustellen. Der Hashvergleich ist im Testskript festgelegt. FT8- und FT4-Beispielaufnahmen sind im vollständigen Repository unter `samples` vorhanden. Die eigenständige Ablation braucht weder Qt noch Hamlib noch ein Funkgerät; benötigt werden CMake, Ninja, Python und kompatible GNU-C++-/Fortran-Compiler. Für den Satz sind LaTeX und die im Manuskript verwendeten Pakete erforderlich.

Tabelle 8: Zentrale Dateien des Reproduktionspakets relativ zum Paperordner.

Datei	Bedeutung
<code>experiment/benchmark.cpp</code>	Messhülle, Aufwärmen, randomisierte Blöcke und Fehlervergleich.
<code>experiment/generate_variants.py</code>	Kontrollierte, mechanisch erzeugte Ablationen.
<code>experiment/analyze.py</code>	Mediane, Quartile, Tabellen und Quellmanifest.
<code>data/timings.csv</code>	315 Rohwerte, einschließlich Block und Wiederholungszahl.
<code>data/numerical.csv</code>	Maximale und RMS-Sampleabweichung pro Variante.
<code>data/summary.json</code>	Deskriptive Statistiken ohne inferenzstatistische Zusätze.
<code>data/source_manifest.json</code>	SHA256 des untersuchten Implementierungsstands.
<code>data/environment.json</code>	Rechner- und Compilerangaben sowie Messbedingungen.
<code>data/native_validation.log</code>	Erneuter nativer Differenz- und Korrektheitslauf.

B Ergänzung vom 21. Oktober 2026: experimentelles REIST16

Die bisherigen Phasenexperimente betreffen die Wellenformerzeugung in FT8, FT4 und FT2. Am 21. Oktober 2026 wurde zusätzlich REIST16 als experimenteller Modus für kurze strukturierte Funkgespräche integriert. Diese Erweiterung liefert ein Anwendungsbeispiel, keine neue Zeitmessreihe der vorstehenden Ablation. Weder deren Beschleunigungsfaktoren noch die Empfangseigenschaften von FT8 werden auf dieses Protokoll übertragen.

Das ursprüngliche QSO/1-Profil verwendet die 77-Bit-Standardnachrichtenpackung von WSJT-X, beschränkt auf Typen 1 und 2, und ein von Weak16/CFP2 aus SDR-suite abgeleitetes Kanalformat. Jeweils einer von 16 Tönen ist aktiv, bei stetiger Phase, 25 Hz Tonabstand und 80 ms Symbolzeit. Die äußeren Tonmitten liegen 375 Hz auseinander; die belegte Bandbreite wurde nicht gemessen. Kopf und Nutzlast verwenden einen terminierten K7-Faltungscode der Rate 1/2 mit oktalen Generatoren (171,133), Interleaving und CRC32-Prüfung. Die zehn Nutzbytes enthalten die 77 Bits und Null-Padding. Das Profil RSQ1, hexadezimal `0x31515352`, kennzeichnet genau diesen Vertrag. Er ist weder mit FT8 noch mit dem vorhandenen WTX1-Flow-Textformat kompatibel.

Auf 16 Trainingssymbole folgen 43 Kopfsymbole, 59 Nutzlast-/CRC-Symbole und zwei stumme Symbole. Der Rahmen mit 120 Symbolen dauert somit 9,60 s, davon 9,44 s aktive Töne. Die abwechselnden UTC-Slots dauern 15 s; die Wellenform beginnt nominell 0,500 s nach der Slotgrenze. Der Sender verwendet 48 kHz Audio, der native Empfänger verarbeitet die 12-kHz-Slotsamples von WSJT-X. Der Modus besitzt eine eigene Menüauswahl, Decoderansteuerung, Standardnachrichten, Doppelklickauswahl und Auto Seq. Vollständige Rufzeichen einschließlich /P und /R unterscheiden Stationen mit gleichem Stammrufzeichen. Bitreihenfolge, CRC, Trainingsfolge und Startfristen sind in `docs/REIST16.txt` festgelegt.

B.1 QSO/2 und WebSDR-Empfang: Stand vom 23. Oktober 2026

Am 22. Oktober wurde QSO/2 als neues Sendeprofil eingeführt. Es verwendet 77 Informationsbits, CRC14 und LDPC(174,91), 16-GFSK mit 200 ms pro Symbol, 5 Hz Tonabstand und BT=4. Die 64 Symbole dauern 12,8 s; 20 davon dienen der Synchronisation. Die äußeren Tonmitten liegen 75 Hz auseinander, was keine gemessene belegte Bandbreite bezeichnet. QSO/2 ersetzt beim Senden den vorstehend beschriebenen QSO/1-Rahmen; der Empfänger unterstützt beide Profile. Die Historie und der Hardwarebeleg vom 21. Oktober betreffen QSO/1. Der aktuelle Bit- und Zeitvertrag steht in `docs/REIST16-QS02.md`.

Für WebSDR lässt sich Browseraudio direkt über WASAPI-Loopback empfangen, während dieselbe Instanz über den Yaesu sendet. Ein gesonderter Selbsttest verwendet zusätzlich eine reine Empfangsinstanz. In 31 nach erfolgreicher Dekodierung gespeicherten WebSDR-Aufnahmen vom 23. Oktober reproduzierte der native Decoder mit Tiefe 3 alle protokollierten Nachrichten, mit Tiefe 1 26 davon. Dies ist eine Auswahl erfolgreicher Aufnahmen und keine allgemeine Dekodierquote. Die Dateien enthalten 12-kHz-PCM16-Monoaudio; digitales Clipping wurde nicht gefunden. Die unveränderten Audiooriginale bleiben lokal; Messwerte und SHA256-Identitäten sind im Belegpaket dokumentiert.

Der 15-s-Dateirahmen enthielt nur bis 13,824 s aufgenommenes Audio, danach Nullauffüllung. Für zusätzliche WebSDR-Verzögerung blieb dadurch kaum Reserve. Ein unabhängiger Regressionstest erzeugt sechs verrauschte CQ-TEST-Rahmen mit Startzeiten 1,1, 1,3 und 1,5 s, jeweils zwei Fälle bei nominell -18 dB SNR bezogen auf 2500 Hz. Bis 13,824 s aufgenommen dekodierte keiner; bis 14,400 s aufgenommen dekodierten alle sechs. Das belegt in diesen Fällen bessere Verzögerungstoleranz, keinen zusätzlichen dB-Gewinn für vollständig aufgenommene Signale.

Die ausgelieferte Änderung verlängert ausschließlich den expliziten reinen Empfangsmodus `-receive-only`: 50 statt 48 Halbsymbole zu je 3456 Samples. Der kombinierte Sende-/Empfangsbetrieb bleibt bei 13,824 s. Der vollständige lokale Tiefe-3-Decoder benötigte im Prozess-Test etwa 0,98–1,34 s. Ein allgemeiner Start erst bei 14,400 s ließe jedoch nur etwa 0,9 s bis zum Ende des nächsten PTT-Anforderungsfensters bei 15,300 s. Die Einschränkung bewahrt deshalb das bisherige Antwortbudget; sie ist keine Laufzeitgarantie. Funkformat und Sendesperren gegen verspätete Starts bleiben unverändert. Tests über den echten Shared-Memory-Worker prüfen verzögerte Rahmen, einen leeren Slot und die Rückkehr zum normalen Aufnahme Fenster.

B.2 Pilotversuche zur gemeinsamen Auswertung wiederholter CQ-Rufe

Ein separater Offline-Prototyp mittelt Suchspektren zweier identischer Wiederholungen und addiert ihre Soft-Metriken vor LDPC/OSD. In synthetischen Paaren sind Trägerphase und Rauschen unabhängig, Startzeit und Frequenz dagegen gleich; die Hälfte der Fälle enthält zusätzlich 0,02 Hz/s Drift. Der Decoder erhält keine vorgegebenen Rufzeichen oder Locator. Erwartete Nutzbits dienen allein der nachträglichen Bewertung. Verglichen wird gegen mindestens einen erfolgreichen Einzeldecode aus beiden Slots:

SNR, 2500 Hz	Einzeln erfolgreich	Gemeinsam erfolgreich
-22 dB	4/8	7/8
-23 dB	0/8	1/8
-24 dB	0/8	0/8

Es entstanden vier zusätzliche exakte Ergebnisse und kein Verlust eines zuvor erfolgreichen Paares. An drei echten CQ-Aufnahmepaaren mit unabhängig zugesetztem Rauschen ergab sich hingegen kein zusätzlicher Erfolg: beide Verfahren erreichten bei um 3 dB erhöhter geschätzter lokaler Rauschleistungsdichte 1/3, bei 6 dB jeweils 0/3. Diese Zusatzrauschwerte sind keine kalibrierte absolute Empfangs-SNR. In 16 negativen Rausch-/Dauertonpaaren wurde keine falsche Nachricht ausgegeben; die kleine Stichprobe begründet keine Fehlalarmrate. CRC-, LDPC-, Synchronisations- und Hardfehlerprüfungen wurden nicht gelockert.

Mehr Suchkandidaten, ein Synchronisationsnenner aus 15 statt zwei falschen Tönen und eine einfache generische CQ-Vorannahme lieferten im Pilotkorpus keinen Zusatzgewinn. Die gemeinsame Auswertung bleibt Forschung und ist nicht Bestandteil des ausgelieferten Decoders. Robuste Ausrichtung, Nachrichtenzuordnung, Rücksetzen und größere unabhängige Negativtests fehlen noch. Rohresultate, Prototypquellen und Prüfprotokolle liegen unter `docs/papers/reist-wsjtx/evidence/websdr-20261023`; sie ergänzen die historischen Phasenmessungen, ohne deren Daten

oder Aussagen zu ändern.

B.3 Beobachteter Yaesu-/Pluto-Dialog

Der Benutzer stellte den Original-Screenshot `docs/images/REIST16_QS0.png` bereit. Er zeigt links Yaesu FT-991A / OE3SRO und rechts PlutoSDR / OE3SRO/P, beide mit der Dialfrequenz 432,500 MHz. Die Yaesu verwendet Soundkarten-Audio; Pluto eine lokale TCI-Bridge für Frequenzsteuerung, PTT und Audio mit USB-I/Q-Wandlung bei 3,072 MS/s. Am 21. Oktober 2026 ist folgende Folge sichtbar:

UTC	Sender	Nachricht
19:26:45	Yaesu	CQ OE3SRO JN88
19:27:30	Pluto	OE3SRO OE3SRO/P JN88
19:27:45	Yaesu	OE3SRO/P OE3SRO -02
19:28:00	Pluto	OE3SRO OE3SRO/P R+13
19:28:15	Yaesu	OE3SRO/P OE3SRO RR73
19:28:30	Pluto	OE3SRO OE3SRO/P 73

Die ersten fünf Nachrichten werden an der Gegenstelle dekodiert. Die letzte 73 ist als vom Pluto gesendet dargestellt; ihr Empfang an der Yaesu sowie ein Logbucheintrag sind nicht sichtbar. Die Yaesu zeigt SNR-Werte von -2 und -1 dB, der Pluto +13 und +14 dB, bei DT 0,1 s. Das sind Decoderschätzungen dieses Aufbaus. Der vom Benutzer als funktionierend gemeldete TxGain von -89 dB ist eine Pluto-Einstellung, keine kalibrierte HF-Leistung, kein Kanal-SNR und kein Ergebnis einer Empfindlichkeitsmessung. Beide Dialanzeigen stimmen überein, während die empfangenen Audiofrequenzen abweichen; der Ursprung dieses relativen Versatzes wurde nicht isoliert.

B.4 Streamingdiagnose und Aussagegrenzen

Vor der Pufferänderung wurden zwei vom Benutzer ausgelöste Pluto-CQs nur lesend aus dem Yaesu-Empfangspuffer erfasst. Ihre PCM-Spitzen lagen bei 1216 und 1176 auf der vorzeichenbehafteten 16-Bit-Skala, ohne geclippte Samples. Beide Originale scheiterten am Decoder; nach nachträglichem Resampling mit 994/1000 dekodierten beide als CQ OE3SRO/P JN88. Das Resampling ändert neben der Zeit auch die Tonhöhe; der native Decoder sucht die Frequenz unabhängig. Der Vergleich mit der exakt kodierten Symbolfolge ergab gegen Ende der ersten Aufnahme etwa 80–100 ms aufgelaufene Verzögerung und kurze Amplitudeneinbrüche. Das identifiziert eine Zeitverzerrung im Gesamtweg, trennt jedoch die Anteile von Host, Transport und DAC nicht voneinander.

Die Bridge bündelt jetzt aufeinanderfolgende 10-ms-Blöcke zu 40-ms-TX- Transfers und verwendet vier Kernpuffer; RX behält 10-ms-Blöcke. Ein stummer Nullblock geht dem Einschalten des TX-LOs voraus. Halt Tx verwirft ausstehende Samples; verkürzte IIO-Transfers führen zum Abbruch statt zum stillen Sampleverlust. Ein virtueller DAC-Test prüft geordnete I/Q-Samples und eine ununterbrochene 12-s-Zeitachse bei wiederholten 25-ms-Produzentenpausen. Native Modemtests, negative Kontrollen, Produktions-Audio-, Rufzeichen- und Shared-Memory-Tests ergänzen die Prüfung. Der anschließende Screenshot belegt beidseitige funktionierende Dekodierung und automatischen Fortschritt bis RR73 mit anschließendem 73-Senden. Er quantifiziert weder verbleibende Zeitfehler noch Empfindlichkeit, Fadingtoleranz, Falschdekodiertrate oder Langzeit-zuverlässigkeit. Eine unabhängige Wiederholung oder ein Vergleich unter gleichen Bedingungen mit FT8 wird nicht behauptet. Der Modus bleibt experimentell.

C Beitragszuordnung und Offenlegung

Die REIST-Formulierung wird der bereitgestellten Arbeit von Rudolf Stepan zugeordnet. Die ursprünglichen FT8-/FT4-Protokolle und Routinen stammen aus WSJT-X und bleiben entsprechend referenziert; die adaptierte FT2-Integration beruht auf WSJT-X Improved von DG2YCB, wie im Repository dokumentiert. Die Codegrundlage unterliegt der GPLv3. Die vorliegende Studie ersetzt keine Lizenz- oder Autorenhinweise der ursprünglichen Implementierung.

Implementierung, lokale Messautomatisierung und Ausarbeitung dieses Manuskripts entstanden mit Unterstützung von OpenAI Codex. Eine unabhängige Replikation und externe fachliche Begutachtung liegen für diese Fallstudie noch nicht vor. Messbeobachtung, mathematische Herleitung und offene Validierungsfragen sind deshalb ausdrücklich getrennt dargestellt.

Literatur

- [1] R. Stepan, *REIST-Division: Eine implementierungsorientierte Formulierung zentrierter Restarithmetik. Theorie, Architekturanalyse und empirische Evaluation auf ARMv8-A, x86-64 und FPGA*, bereitgestelltes Manuskript, 2026. DOI laut Manuskript: 10.5281/zenodo.21206471. Verwendet insbesondere Abschnitte 3, 5 und 11. Die verwendete PDF wird im Artefaktmanifest über SHA256 bezeichnet.
- [2] S. Franke, B. Somerville und J. Taylor, *The FT4 and FT8 Communication Protocols*, QEX, Juli/August 2020, S. 7–17. https://wsjt.sourceforge.io/FT4_FT8_QEX.pdf.
- [3] Analog Devices, *MT-085: Fundamentals of Direct Digital Synthesis (DDS)*, Tutorial, insbesondere S. 2 zur Phasenakkumulation und Amplitudentabelle. <https://www.analog.com/media/en/training-seminars/tutorials/mt-085.pdf>.
- [4] OpenMP Architecture Review Board, *OpenMP Application Programming Interface*, Version 5.1, Abschnitt 6.7, OMP_WAIT_POLICY. <https://www.openmp.org/spec-html/5.1/openmpse64.html>.
- [5] WSJT-X Custom, *Lokaler Implementierungs- und Teststand vom 15. Oktober 2026*, Basiscommit b527b02f9d7be92e26c5c6a59019df518f97725d, ergänzt durch die in `data/source_manifest.json` bezeichneten Dateien. Repository: <https://github.com/rudolfstepan/wsجتx>.