Persistente modulare Phasenzustände für die GFSK-Wellenformerzeugung

Eine reproduzierbare WSJT-X-Fallstudie

Rudolf Stepan
Unabhängiger Forscher, Wien, Österreich
OE3SRO | ORCID: 0009-0004-2842-2579

21. Oktober 2026 | Journal-Forschungsentwurf, Version 0.4
Vollständige deutsche Fassung des englischen Journalentwurfs
Forschungsentwurf; nicht begutachtet.

Zusammenfassung

Ein digitaler Oszillator muss sich bereits durchlaufene Umdrehungen nicht merken. Es genügt, die Phase innerhalb einer Periode zu speichern und diesen Zustand nach jedem Frequenzschritt zu korrigieren. Diese Arbeit untersucht, wie sich dieses bekannte Prinzip auf die GFSK-Wellenformerzeugung eines WSJT-X-Forks auswirkt. Ausgangspunkt ist die zentrierte Zustandsformulierung aus Stepans REIST-Division-Manuskript. Zunächst wird hergeleitet, unter welchen Voraussetzungen in binary64-Arithmetik eine einzige Korrektur ausreicht. Anschließend werden zentrierte und positive Zustände mit allgemeiner Restbildung, maskierter Korrektur, Q48-Festkomma, unbegrenzter Akkumulation und zwei Varianten mit Block-Präfixsummen verglichen. Auf einem Intel Core i9-14900K benötigt die allgemeine Restbildung auf dem gewählten Performance-Kern etwa die 2,6-fache Zeit der zentrierten Korrektur; auf dem gewählten Efficiency-Kern ist es die 4,4- bis 5,2-fache Zeit. Positive und zentrierte Zustände sind annähernd gleich schnell. Die getesteten Festkomma- und Präfixvarianten sind langsamer. Numerische Prüfungen umfassen codierte Frames, lange Folgen, Intervallgrenzen und digitale Spektren. Der praktische Spielraum ist begrenzt: Die bereits optimierte Wellenformerzeugung beansprucht nur 2,13–3,73% der gemessenen Dateidecoder-Laufzeit. Selbst ihr vollständiger Wegfall würde die gesamte Verarbeitung nur um 2,17–3,87% beschleunigen, sofern alle übrigen Arbeiten gleich blieben. Der Beitrag liegt damit im reproduzierbaren Vergleich der Phasendarstellungen und ihres Aufwands innerhalb der Anwendung. Ein exklusiver Vorteil zentrierter Arithmetik oder eine bessere Empfangsempfindlichkeit ist dadurch nicht belegt. Quellhashes, Rohmessungen und Auswertungsskripte liegen beiden Sprachfassungen bei.

Schlüsselwörter: modulare Phasenakkumulation; GFSK; DDS; reproduzierbare Laufzeitmessung; Gleitkommaarithmetik; WSJT-X.

1 Einleitung

Um eine Schwingung zu erzeugen, schreibt Software einen Phasenwinkel fort und berechnet daraus einen Abtastwert. Bei der gaußgefilterten Frequenzumtastung (GFSK) geht die Frequenz stetig von einem Symbol zum nächsten über. Deshalb ändert sich auch der Phasenschritt von Abtastwert zu Abtastwert. Eine volle Umdrehung verändert die ideale Oszillatorausgabe nicht. Das Programm kann bereits durchlaufene Umdrehungen daher verwerfen und nur die Phase innerhalb einer Periode speichern.

Entscheidend ist, wie viel Rechenarbeit dafür nötig ist. Eine allgemeine Restfunktion kann einen großen Wertebereich auf eine Periode zurückführen. Sind die gespeicherte Phase und der nächste

Schritt bereits begrenzt, kann dagegen eine einzige Addition oder Subtraktion genügen. Stepans REIST-Division-Manuskript [1] beschreibt diesen fortlaufend erhaltenen zentrierten Zustand für ganzzahlige Reste und behandelt in Abschnitt 6 auch Phasenfenster. Hier wird die Formulierung auf Gleitkommaphasen angewandt. Die folgende Herleitung ist in sich geschlossen; frühere CPU- und FPGA-Beschleunigungen aus dem REIST-Manuskript werden nicht als Messwerte dieser Anwendung verwendet.

Ein früherer deutscher Bericht [2] verglich zentrierte Korrektur und allgemeine Restbildung bei gleicher trigonometrischer Auswertung. Die vorliegende Studie erweitert diesen Vergleich in drei Richtungen: Sie untersucht die Voraussetzungen und verbleibenden Fehler der Gleitkommarechnung, vergleicht weitere Phasendarstellungen unter gleichen Bedingungen und misst den Zeitanteil der Wellenformerzeugung im vollständigen Dateidecoder. Performance- und Efficiency-Kerne des Prozessors werden dabei getrennt vermessen.

Der Hauptbefund besteht aus zwei Teilen. Begrenzte Zustandsupdates senken den Aufwand gegenüber der allgemeinen Restbildung auf dem untersuchten Rechner deutlich. Die Zentrierung bietet dabei kaum einen Zeitvorteil gegenüber einem positiven Intervall. Zugleich macht die Wellenformerzeugung bereits nur einen kleinen Teil der Decoder-Laufzeit aus. Der Beitrag dieser Arbeit liegt somit in der geprüften Implementierung und reproduzierbaren Bewertung eines bekannten Zustandsprinzips, einschließlich seiner praktischen Grenze.

2 Verwandte Arbeiten und Bezug zu REIST

Die direkte digitale Synthese (DDS) verwendet üblicherweise einen Phasenakkumulator mit anschließender Umrechnung der Phase in eine Amplitude. Frequenzauflösung und die Begrenzung der Phasenbitzahl gehören zu den bekannten Entwurfsfragen [3]. Ein Gleitkommazustand in $[0, T)$ ist deshalb eine naheliegende Vergleichsvariante zum zentrierten Zustand. Die allgemeine Restfunktion erfüllt eine weitergehende Aufgabe: Sie akzeptiert einen größeren Eingabebereich und legt fest, wie Grenzfälle bei halben Perioden behandelt werden.

Zwei weitere Forschungsgebiete sind für den Vergleich relevant. Die Gleitkommaanalyse erklärt, warum eine Identität über reellen Zahlen nach Rundung nicht mehr gelten muss und wann eine Subtraktion exakt ist [4]. Präfixsummen erlauben es, mehrere Phasenversätze gemeinsam zu berechnen [5]. Sie verkürzen einen Teil der Abhängigkeiten, ändern aber die Reihenfolge der Additionen und benötigen zusätzliche Rechenoperationen und Speicher.

Die Entwickler von FT4 und FT8 beschreiben deren GFSK-Modulation, Synchronisation und Fehlerkorrektur in [6]. Hier wird der Wellenformgenerator dieser Modi und der FT2-Integration des Forks untersucht. Gesonderte Experimente zur LDPC-Decodierung und Synchronisationssuche stehen im beigefügten Decoder-Forschungsdossier.

Die DDS-Literatur zeigt außerdem, dass die Phasenakkumulation nur einen Teil des Oszillators ausmacht. Cordesses [9] behandelt die Frequenzauflösung sowie die Quantisierung von Phase und Amplitude. Vankkas Dissertation [10] untersucht Speicherkompression, Taylor-Näherungen, CORDIC und Ursachen von Störlinien. Seine Arbeiten zur Amplitudenabbildung und Störlinienreduktion behandeln diese Fragen getrennt [11, 12]. Im vorliegenden Vergleich bleibt die interpolierte Amplitudentabelle mit 65 536 Einträgen gleich, damit der Einfluss der Phasendarstellung erkennbar wird.

Andere Oszillatorentwürfe verdienen einen eigenen Vergleich. Palomäki und Nurmi [13] verwenden eine Taylor-Interpolation zweiter Ordnung in einer Quadratur-DDS auf einem FPGA mit komprimiertem Speicher. Deren Taktraten und Störlinienwerte lassen sich nicht als CPU-Laufzeiten oder IQ-Fehlerenergien lesen. Vicaneks technische Notiz [14] beschreibt einen rekursiven Quadraturoszillator und geht auf Frequenzmodulation ein. Eine solche Rekursion schreibt die

Amplitudenzustände direkt fort; bei veränderlicher GFSK-Frequenz müssen auch ihre Koeffizienten angepasst werden. Beide Ansätze werden hier nicht vermessen. Eine weiterführende Studie sollte sie mit denselben modulierten Eingaben und Genauigkeitsanforderungen implementieren.

Tabelle 1: Bekannte Verfahren und ihre tatsächliche Beziehung zu dieser Studie.

Verfahren/Quelle	Prinzip	Unterschied und empirische Rolle
Persistenter zentrierter Zustand, Stepan [1]	Ganzzahliger Zustand mit begrenzter vorzeichenbehafteter Korrektur	Motiviert die Anwendung. Gleitkommainvarianten und GFSK-Zielkonflikte werden hier untersucht; frühere Benchmarkzahlen werden nicht übernommen.
DDS/NCO [3]	Modulares Phasenregister, Amplitudenabbildung	Positiver Gleitkommazustand und Q48-Ganzzahlzustand sind konventionelle konkurrierende Implementierungen.
Allgemeine Restfunktion	Reduktion eines umfassend gültigen Eingabewerts bei jedem Update	Dieselben Double-Inkremente und dieselbe LUT; gemessene Referenz mit breiterem Funktionsumfang.
Präfix-Scan [5]	Umordnung von Teilsummen	Serielle und baumförmige Block-64-Architekturen; zusätzliche Reduktionen und Rundungen sind eingeschlossen.
Komprimierte/Taylor-DDS [10, 13]	Andere Amplitudenabbildung	Literarisch verglichen, nicht vermessen; die Abbildung bleibt im Experiment fest.
Rekursive Quadratur [14]	Amplitudenzustände fortschreiben	Literarisch verglichen, nicht vermessen; Modulation und Amplitudenfehler erfordern eigene Kontrollen.

Der Bezug zu REIST stützt sich auf das bereitgestellte Manuskript [1]; dessen SHA256 steht im Audit. Die abgedruckte DOI und die Zenodo-Metadaten konnten nicht unabhängig abgerufen werden. Referenz [3] dieses Manuskripts enthält eine DOI zu einem fachfremden Verlagsartikel und wird hier nicht übernommen. Maßgeblich für die Anwendung sind die Zustandsdefinition und die additive Fortschreibung.

3 Mathematische Grundlagen

3.1 Der zentrierte Phasenzustand

Wir betrachten zunächst die exakte Rechnung mit reellen Zahlen. Sei $T > 0$ die Länge einer Periode. Zwei Phasen, die sich um ein ganzzahliges Vielfaches von T unterscheiden, beschreiben dieselbe Position innerhalb dieser Periode. Zur Speicherung der Phase wählen wir daher aus jeder solchen Klasse einen Repräsentanten.

Definition 3.1 (Zentrierter Repräsentant)**.** Für eine reelle Zahl x setzen wir

$$C_T(x) = x - T\left\lfloor x/T + \tfrac{1}{2}\right\rfloor \in [-T/2, T/2). \tag{1}$$

Wir nennen $C_T(x)$ den kanonischen zentrierten Repräsentanten von x.

Dabei bezeichnet $\lfloor\cdot\rfloor$ das Abrunden auf eine ganze Zahl. Die Formel zieht eine ganzzahlige Anzahl von Perioden ab. Der linke Rand von $[-T/2, T/2)$ gehört zum Intervall, der rechte nicht. Damit ist der Repräsentant auch bei einer halben Periode eindeutig festgelegt.

Die Definition gilt für jede reelle Eingabe. Bei einem Oszillator kennen wir jedoch bereits die vorhergehende Phase. Ist auch der nächste Schritt begrenzt, können wir diese Information nutzen, um die Rechnung zu vereinfachen.

Proposition 3.2 (Eine Korrektur genügt)**.** *Sei $r \in [-T/2, T/2)$ der aktuelle Zustand und $u \in [-T/2, T/2]$ der nächste Phasenschritt. Wir setzen $s = r + u$. Dann erfüllt das Update*

$$r^+ = \begin{cases} s - T & s \geq T/2, \\ s + T & s < -T/2, \\ s & \textit{sonst} \end{cases} \tag{2}$$

die Gleichung $r^+ = C_T(r + u)$. Insbesondere liegt der neue Zustand wieder in $[-T/2, T/2)$.

Beweis. Aus den Voraussetzungen folgt $-T \leq s < T$. Wir unterscheiden drei Fälle.

1. Ist $s \geq T/2$, so ergibt der Abzug einer Periode $-T/2 \leq s - T < 0$.
2. Ist $s < -T/2$, so ergibt die Addition einer Periode $0 \leq s + T < T/2$.
3. Andernfalls gilt $-T/2 \leq s < T/2$, und es ist keine Korrektur nötig.

In jedem Fall liegt das Ergebnis im gewählten Intervall und unterscheidet sich von s um ein ganzzahliges Vielfaches von T. Wegen der Eindeutigkeit des zentrierten Repräsentanten folgt die Behauptung. □

Die Proposition erklärt zugleich, warum wir das Verfahren wiederholen dürfen. Liegt der Anfangszustand im Intervall und erfüllt jeder Schritt die genannte Schranke, so liefert jedes Update wieder einen Zustand, auf den die Proposition anwendbar ist. Mit vollständiger Induktion folgt die Intervallzugehörigkeit nach jedem Update. Diese erhaltene Eigenschaft nennen wir *Zustandsinvariante*. Sie ist die additive Anwendung der REIST-Formulierung [1, Satz 3.6 und Korollar 3.7].

Beispiel 3.3. Es seien $T = 360$, $r = 170$ und $u = 30$. Dann ist $s = 200$. Wir ziehen also eine Periode ab und erhalten $r^+ = -160$. Beide Winkel beschreiben dieselbe Position. Die Gradangaben dienen der Veranschaulichung; die Implementierung verwendet das Bogenmaß.

Bemerkung 3.4 (Positiver Zustand)**.** Dieselbe Überlegung ist für $p \in [0, T)$ möglich. Unter derselben Schranke für den vorzeichenbehafteten Schritt liegt $p + u$ in $[-T/2, 3T/2)$. Bei einer negativen Summe addieren wir eine Periode, bei einer Summe von mindestens T ziehen wir eine Periode ab. Andernfalls bleibt sie unverändert. Der begrenzte Zustand und die begrenzten Schritte machen die allgemeine Division also entbehrlich. Die Zentrierung ist für diese Folgerung nicht erforderlich.

3.2 Das Update in Gleitkommaarithmetik

Wir gehen nun zu der Rechnung über, die das Programm tatsächlich ausführt. Mit $P = \mathrm{double}(\pi)$ bezeichnen wir die gespeicherte Näherung an Pi, mit $\widehat{T} = 2P$ die dargestellte Periode. Die Multiplikation mit zwei ist hier exakt. fl bezeichne die Rundung auf den nächsten binary64-Wert.

Die Proposition über reelle Zahlen ist noch kein Beweis für diese Rechnung. Wir müssen zusätzlich prüfen, ob die Rundung das Ergebnis aus dem gewählten Intervall herausführen kann.

Proposition 3.5 (Erhaltung des binary64-Intervalls)**.** *Es gelte binary64-Arithmetik mit Rundung auf den nächsten Wert und graduellem Unterlauf. Die gespeicherten Werte seien endlich und es gelte $r \in [-P, P)$ sowie $u \in [-P, P]$. Die angegebene Operationsreihenfolge werde eingehalten. Wir setzen $s = \mathrm{fl}(r + u)$ und wenden Gleichung (2) mit der Periode $\widehat{T}$ an. Dann liegt der korrigierte Zustand in $[-P, P)$. Jede dabei ausgeführte korrigierende Addition oder Subtraktion ist exakt.*

Beweis. Die Rundung erhält die Ordnung, also gilt $s \in [-2P, 2P]$. Anders als bei reeller Rechnung kann der obere Rand erreicht werden. Für $s \geq P$ unterscheiden sich die positiven Werte s und $2P$ höchstens um den Faktor zwei. Nach dem Sterbenz-Lemma ist daher $s - 2P$ exakt und liegt in $[-P, 0]$. Für $s < -P$ ist der Ausdruck $s + 2P$ die Differenz positiver Beträge mit derselben Faktor-zwei-Eigenschaft. Auch diese Rechnung ist exakt; ihr Ergebnis liegt in $[0, P)$. Im verbleibenden Fall gehört s bereits zu $[-P, P)$ und bleibt unverändert. Damit ist die Behauptung bewiesen, einschließlich des Randfalls $s = 2P$. □

Als Nächstes untersuchen wir den Fehler. Bei einer normalen Summe ist die erste Addition die einzige Operation dieses Updates, die rundet. Ihr absoluter Fehler beträgt höchstens $\epsilon_u \widehat{T}$, wobei $\epsilon_u = 2^{-53}$ die Rundungseinheit ist.

Zum Vergleich zweier Phasenzustände brauchen wir einen Abstand, der die Periodizität berücksichtigt. Nahe einer Intervallgrenze kann die gewöhnliche Differenz fast eine ganze Periode betragen, obwohl die Phasen dicht beieinanderliegen. Wir lassen deshalb vor der Differenzbildung eine Verschiebung um ganze Perioden zu.

Korollar 3.6 (Aufgelaufener Kreisfehler)**.** *Unter den vorstehenden Voraussetzungen seien alle Summen normal. Mit $\widehat{r}_n$ bezeichnen wir den berechneten Zustand nach n Updates, mit r_n den Zustand bei exakter Rechnung mit demselben Anfangswert, denselben dargestellten Schritten und derselben Periode. Dann gilt*

$$d_{\widehat{T}}(\widehat{r}_n, r_n) \leq n\epsilon_u \widehat{T}, \qquad d_T(a, b) = \min_{k \in \mathbb{Z}} |a - b - kT|, \tag{3}$$

wobei d_T den Kreisabstand zur Periode T bezeichnet.

Beweis. Die Addition desselben Schritts zu beiden Zuständen ändert ihren Kreisabstand nicht. Dasselbe gilt für das Hinzufügen oder Entfernen einer ganzen Periode. Nach der vorangehenden Proposition ist die Korrektur exakt. Jede gerundete Addition kann den Abstand somit höchstens um $\epsilon_u \widehat{T}$ vergrößern. Der Anfangsabstand ist null. Addieren wir diese Schranken über n Schritte, erhalten wir die behauptete Ungleichung. □

Ist eine Summe nicht normal, müssen bei graduellem Unterlauf subnormale Terme berücksichtigt werden. Die Schranke beschreibt den ungünstigsten Fall, keine vorhergesagte Drift. Nicht enthalten sind Fehler bei der Bildung der Schritte, der Unterschied zwischen $\widehat{T}$ und dem tatsächlichen 2π sowie der Restfunktionspfad für große Schritte. NaNs, andere Rundungsmodi und Umformungen der angegebenen Operationen liegen außerhalb der Begründung.

Bemerkung 3.7 (Randwerte anderer Darstellungen)**.** Die Funktion `std::remainder` rundet ihren Quotienten auf die nächste ganze Zahl, bei Gleichstand auf die gerade. Sie kann daher bei einer halben Periode $+P$ zurückgeben. Ein einfaches positives Update kann $\widehat{T}$ liefern, wenn die Addition von $\widehat{T}$ zu einem winzigen negativen subnormalen Wert aufgerundet wird. Diese Werte haben eine gültige periodische Bedeutung, liegen aber außerhalb des gewählten halboffenen Intervalls. Intervallzugehörigkeit und Wellenformfehler müssen deshalb getrennt geprüft werden.

3.3 Festkomma und blockweise Summen

Die Q48-Variante teilt eine Umdrehung in 2^{48} ganzzahlige Positionen. Ihr Zustand $q/2^{48}$ wird in Perioden gemessen. Um einen Phasenschritt zu addieren, rechnen wir ihn zunächst in diese ganzzahlige Skala um:

$$q^+ = (q + \mathrm{round}(2^{48} u/\widehat{T})) \bmod 2^{48}. \tag{4}$$

Die Modulo-Operation behält die niedrigsten 48 Bits. Vorzeichenlose Addition und Maskierung sind exakt. Bei exakter Skalierung trägt die Rundung eines Schritts höchstens einen halben

diskreten Phasenschritt bei. Die Kreisfehlerschranke beträgt deshalb $n\widehat{T}/2^{49}$ nach n Updates. Die Double-Skalierung verursacht weitere Rundung; hinreichend kleine Schritte werden auf null gerundet. Die getestete Implementierung ruft für jeden veränderlichen GFSK-Schritt `llround` auf. Die Zeit für diese Umrechnung ist in der Messung enthalten.

Die Blockvarianten gehen anders vor. Sie fassen $b = 64$ Abtastwerte zusammen und berechnen jeden Phasenversatz als Summe der vorhergehenden Schritte innerhalb des Blocks. Eine solche Summe heißt Präfixsumme. Sie kann mehrere Umdrehungen umfassen, sodass die Proposition zur einzelnen Korrektur nicht mehr anwendbar ist. Im Experiment werden die ausgegebenen Versätze durch eine Formel mit Abrundung reduziert. Für den Zustand, der an den nächsten Block übergeben wird, kommt eine Restfunktion zum Einsatz.

Die serielle Variante addiert die Schritte nacheinander. Die Baumvariante gruppiert die Additionen, um unabhängige Arbeit zu ermöglichen, benötigt aber mehr Operationen und Speicher. Beide ändern gegenüber dem skalaren Oszillator die Rundungsreihenfolge. Die beiden Arrays der Baumimplementierung belegen zusätzlich zu den Wellenformpuffern 1024 Byte für Double-Werte.

4 Implementierung in WSJT-X

Der produktive Generator speichert die Phasenschritte und die zentrierte Phase als Double-Werte. Für jeden Abtastwert berechnet er zunächst aus der aktuellen Phase eine komplexe Amplitude und schreibt danach die Phase fort. Die Amplitude entsteht durch lineare Interpolation in einer komplexen Float-Tabelle mit 65 536 Intervallen. Für ideale Sinus- und Kosinuswerte ist der Interpolationsfehler durch $h^2/8$ begrenzt; $h = 2\pi/65536$ bezeichnet den Abstand der Tabelleneinträge. Das ergibt ungefähr $1{,}15 \cdot 10^{-9}$ je Komponente. Die Speicherung als Float und die Rundung bei der Interpolation verursachen weitere Fehler. Die Bildung des Gaußpulses sowie die Ein- und Ausblendrampen bleiben modusspezifisch.

Tabelle 2: Frameparameter bei 48 ksample/s in diesem Fork.

	FT8	FT4	FT2
Modulationstöne	8	4	4
Eingabesymbole	79	103	103
Samples/Symbol	7680	2304	1152
Ausgabesamples	606720	241920	120960
Dauer (s)	12,64	5,04	2,52
Gaussian-Parameter BT	2	1	1

Der vermessene produktive Quellstand ist durch Commit

`46ebbd78c663bf07eed054570df4888a6268e156`

identifiziert. Alle 45 Dateihashes im Manifest des früheren Berichts stimmen mit diesem Commit überein. Die Forschungsprogramme sind gesondert durch Dateihashes erfasst, einschließlich der tatsächlich für die Messungen verwendeten Quellversionen. Historische Manifeste behalten ihre ursprüngliche Beschreibung des Quellstands zum Messzeitpunkt.

In der Tabelle steht `journal/experiment` verkürzt für `docs/papers/reist-ftx/journal/experiment`. Jede experimentelle Kopie ersetzt nur den Oszillatorabschnitt. Pulserzeugung, Rampen und Ausgabepuffer bleiben gleich. Die produktiven Algorithmen werden durch die Studie nicht verändert.

Die ursprünglichen Fortran-Quellen der Wellenformerzeugung stehen für Referenztests weiterhin zur Verfügung. Sie unterscheiden sich in Genauigkeit, trigonometrischer Auswertung und

Tabelle 3: Implementierungsstellen; Pfade sind relativ zum Repository-Hauptordner.

Datei/Funktion	Aufgabe
`lib/ft8/native/ft8_wave.cpp`, `waveform`	Gemeinsame Phase, LUT, Pulse, Schutzpfade und Rampen für FT8/FT4/FT2.
Dieselbe Datei, `pulseFor`	Höchstens acht unveränderliche gemeinsam genutzte Pulsobjekte; eine Cache-Sperre pro Frame; Arbeitsdaten gehören jedem einzelnen Aufruf.
`journal/experiment/phase.hpp`, `oscillate`	Getrennte alternative Oszillatoren; generierte Kopien ersetzen ausschließlich den Oszillatorabschnitt.
`lib/ft4/native/ft4_sync.cpp`, `search`	Wiederverwendung frequenzkorrigierter Costas-Referenzen; unabhängige Zeilen können parallel laufen.
`lib/ft8/native/ft8_ldpc.cpp`	Produktiver nativer Encoder und zwei BP-Paritätsanordnungen; in dieser Studie unverändert.
`lib/ft8/decode174_91.f90`	Standard-Hybrid aus BP und OSD, auch von FT4/FT2 aufgerufen.

Zwischenspeicherung. Ein Laufzeitvergleich mit ihnen kann daher nicht allein den Einfluss der Programmiersprache bestimmen. Sowohl Fortran als auch C++ werden in nativen Maschinencode übersetzt. FT2 verwendet die FT4-basierte Integration des Forks und keinen eigenständigen LDPC-Code.

5 Versuchsaufbau und Auswertung

5.1 Laufzeitvergleich der Phasendarstellungen

Für die Messungen verwenden wir einen Intel Core i9-14900K mit 24 physischen Kernen und 32 logischen CPUs unter Windows 11 Enterprise, Build 26300. Die MSYS2-Werkzeugkette stellt GCC/GFortran 15.2.0 bereit. Die Wellenformprogramme verwenden C++11 und `-O3 -fno-fast-math -ffp-contract=off`, ohne LTO und ohne `-march=native`. Nur die gesonderte decimal-100-Prüfung mit Boost 1.90 erfordert C++14; sie gehört nicht zu den vermessenen Wellenformkernen.

Der Prozessor besitzt zwei Kerntypen. Eine Affinitätsbindung legt den Prozess auf eine logische CPU fest; CPUID-Blatt 1A bestätigt deren Typ [7]: 0x40 für den gewählten Performance-Kern (logische CPU 0), 0x20 für den gewählten Efficiency-Kern (CPU 16). Jede Auswahl wird in drei getrennten Prozessläufen vermessen. In drei weiteren Läufen darf das Betriebssystem frei zuweisen. SMT-Geschwister, Turbo, Energieverwaltung und Hintergrundlast sind nicht isoliert. Die Ergebnisse beschreiben daher diese ausgewählten Kerne unter den dokumentierten Bedingungen.

Jeder Prozesslauf umfasst neun Blöcke je Modus. Innerhalb eines Blocks laufen nach dem Aufwärmen alle zehn Implementierungen in zufälliger Reihenfolge und erzeugen jeweils drei vollständige Frames. Sie erhalten dasselbe Telemetriewort (Seed 991), eine Abtastrate von 48 ksample/s und eine Trägerfrequenz von 1855 Hz. Die bestehende LDPC-/CRC-Implementierung codiert das Wort mit den passenden Costas-, Gray- und FT4-Scrambling-Zuordnungen. Die Ausgabe besteht aus komplexen Float-IQ-Abtastwerten. Gemessen werden auch Pulserzeugung, Speicherbelegung, Cache-Zugriff und Rampen. Als Implementierungsprüfung muss die herausgelöste zentrierte Variante im numerischen Testkorpus dieselben Ausgabebytes wie der produktive Generator erzeugen.

Für das Laufzeitverhältnis bestimmen wir zunächst den Median aller Blockzeiten einer Variante. Diesen teilen wir durch den entsprechenden Median der zentrierten Variante. Zur Abschätzung der Unsicherheit werden zunächst ganze Prozessläufe erneut gezogen, anschließend gepaarte zyklische Blöcke der Länge drei innerhalb des jeweiligen Laufs (2000 Wiederholungen, Seed 991202610). Die Blocklängen eins und fünf prüfen, wie empfindlich das Ergebnis auf diese Wahl reagiert. Diese Gruppierung erhält einen Teil der zeitlichen Abhängigkeit; wiederholte Frames zählen nicht als unabhängige Messungen. Bei nur drei Prozessläufen und gemeinsamen Temperatur- und Betriebssystemeinflüssen gelten die Intervalle bedingt auf dieses Experiment. Sie belegen weder eine allgemeine statistische Überdeckungswahrscheinlichkeit noch die Streuung zwischen verschiedenen Rechnern.

Für die Zeitmessung verwenden wir `steady_clock`. Die Windows-Funktion `QueryThreadCycleTime` liefert ergänzend die dem Thread zugerechneten Zyklen aus Benutzer- und Kernelmodus je Abtastwert [8]. Diese Werte sind weder eine Zählung abgeschlossener Maschinenbefehle noch kalibrierte Hardware-Zählerdaten. Sprungfehlvorhersagen, Cache-Ereignisse, Energie und dynamische Befehlszahlen wurden nicht erfasst. Für kleine Aktualisierungsschleifen liegt Compiler-Assembler bei. Maskierte Ausdrücke im Quelltext allein garantieren keinen sprungfreien Maschinencode.

5.2 Referenzläufe, numerische Prüfungen und Spektren

Der frühere Benchmark mit reeller Ausgabe wird in getrennte Dateien erneut ausgeführt. Er verwendet die ursprünglichen 21 Blöcke, zehn Wiederholungen und Tonindizes mit Seed 991. Diese Indizes bezeichnen gültige Töne, müssen aber keine Nachricht codieren. Die Ergebnisse bleiben sowohl von den eingefrorenen Originaldaten als auch vom neuen Experiment mit komplexer Ausgabe getrennt.

Der numerische Testkorpus umfasst drei Modi, zwei Abtastraten (12 und 48 ksample/s), zwei codierte Telemetrienachrichten und fünf Trägerfrequenzen je Abtastrate: -1855, 0.001, 1855, $0.49 f_s$ und 10^9 Hz. Dabei bezeichnet f_s die Abtastrate. Daraus ergeben sich 60 Framekonfigurationen für jede der zehn Implementierungen. Der extrem hohe Trägerwert prüft den Restfunktionspfad für große Schritte; er beschreibt keinen RF-Betriebsfall.

Die Long-Double-Referenz akkumuliert dieselben Double-Phasenschritte mit derselben dargestellten Periode und berechnet anschließend Sinus und Kosinus in Long Double. Ihr Signifikand hat auf diesem System 64 Bits. Damit liefert sie eine genauere arithmetische Vergleichsgröße, behält aber die Fehler der Pulsbildung und Periodendarstellung bei. Gesonderte decimal-100-Rechnungen prüfen 4096 lokale Updates; eine Million zufälliger Zustands-Schritt-Paare prüfen die Intervallzugehörigkeit.

Langzeittests umfassen jeweils eine Million Schritte: konstante positive Inkremente, zufällige vorzeichenbehaftete Inkremente und winzige alternierende Inkremente. Die Ausgabe unterscheidet den Kreisfehler der Phase, den maximalen und den RMS-Fehler des komplexen IQ-Signals sowie die exakte Byteübereinstimmung.

Für den Spektralvergleich transformiert eine deterministische Radix-2-FFT jeweils den vollständigen Frame einschließlich Rampen. Das Beobachtungsfenster ist rechteckig; bis zur nächsten Zweierpotenz wird mit Nullen aufgefüllt. Ein Test mit bekannter Frequenzposition und eine Energieprüfung nach Parseval kontrollieren die Transformation. Das gewählte Analyseband reicht jeweils zwei Tonabstände über den niedrigsten und höchsten nominellen Ton hinaus. Es ist eine Festlegung für die Auswertung, keine regulatorische Emissionsmaske. Die Fehlerenergie wird durch die Energie des Referenzframes geteilt. Alle Spektren beschreiben in Software erzeugte digitale IQ-Signale.

5.3 Zeitanteil der Wellenformerzeugung im Decoder

Ein gesondertes, vom normalen Build ausgeschlossenes Programm misst die Aufrufe der unveränderten produktiven Wellenformquelle innerhalb des vollständigen `jt9`-Dateidecoders. Es verarbeitet aufgezeichnete FT8-, FT4- und optional lokal vorhandene FT2-Samples mit Decodiertiefe drei und einem OpenMP-Thread. Auf einen Aufwärmlauf folgen drei gemessene Prozessläufe. Die Mengen der decodierten Nachrichten müssen dabei übereinstimmen.

Um den Komponentenanteil zu bestimmen, addieren wir die Zeiten aller Wellenformaufrufe. Die Summe teilen wir durch die verstrichene Zeit vom Prozessstart bis zum Ende. Der Programmstart ist somit enthalten. Das Profil beschreibt die gewählten Aufnahmen und Einstellungen; es vermisst weder die Bedienoberfläche noch einen laufenden Datenstrom oder einen Transceiver.

6 Laufzeitergebnisse

Tabelle 4: Wellenformzeit mit komplexer Ausgabe auf dem ausgewählten P-Kern: Median von 27 Blockmittelwerten, Millisekunden pro vollständigem Frame. Die gesamte Puls- und Rampenarbeit ist eingeschlossen.

Architektur	FT8 (ms)	FT4 (ms)	FT2 (ms)
Zentriert	2,774	1,088	0,550
Restfunktion	7,201	2,857	1,431
Positiv	2,860	1,089	0,601
Maskenform	2,876	1,134	0,557
Q48-DDS	6,482	2,672	1,332
Unbeschränkt	10,857	4,347	2,227
Serielles Präfix-64	4,758	1,890	1,024
Baumpräfix-64	5,864	2,403	1,260
Long double	27,437	10,588	5,386
Produktion	2,719	1,038	0,528

Tabelle 5: Kernabhängigkeit der allgemeinen Restfunktion (G) gegenüber zentrierter Korrektur (C). Die Intervalle sind bedingte hierarchische Resampling-Intervalle; siehe Methodik.

Kern	Modus	G/C	Bedingtes 95%-Intervall	C (ns/Sample)
P	FT8	2,60	[2,35; 2,71]	4,57
P	FT4	2,62	[2,32; 2,75]	4,50
P	FT2	2,60	[2,53; 2,97]	4,55
E	FT8	5,19	[5,05; 5,33]	5,96
E	FT4	4,94	[4,70; 5,08]	6,37
E	FT2	4,43	[4,27; 4,61]	7,04
OHNE BINDUNG	FT8	2,63	[2,62; 2,67]	4,16
OHNE BINDUNG	FT4	5,03	[2,41; 5,17]	6,46
OHNE BINDUNG	FT2	5,53	[1,98; 5,85]	5,62

Positive und zentrierte Zustände benötigen annähernd dieselbe Zeit. Der wesentliche Gewinn entsteht dadurch, dass die allgemeine Restbildung nicht nach jedem Schritt nötig ist. Maskierung im Quelltext verbessert das Ergebnis nicht durchgehend. Die Q48-Variante mit Umrechnung bei jedem Schritt und die beiden Blockvarianten sind langsamer als die skalare zentrierte Korrektur. Das gilt für die hier getesteten Implementierungen; schnellere Integer-DDS- oder ausdrücklich vektorisierte SIMD-Entwürfe sind damit nicht ausgeschlossen.

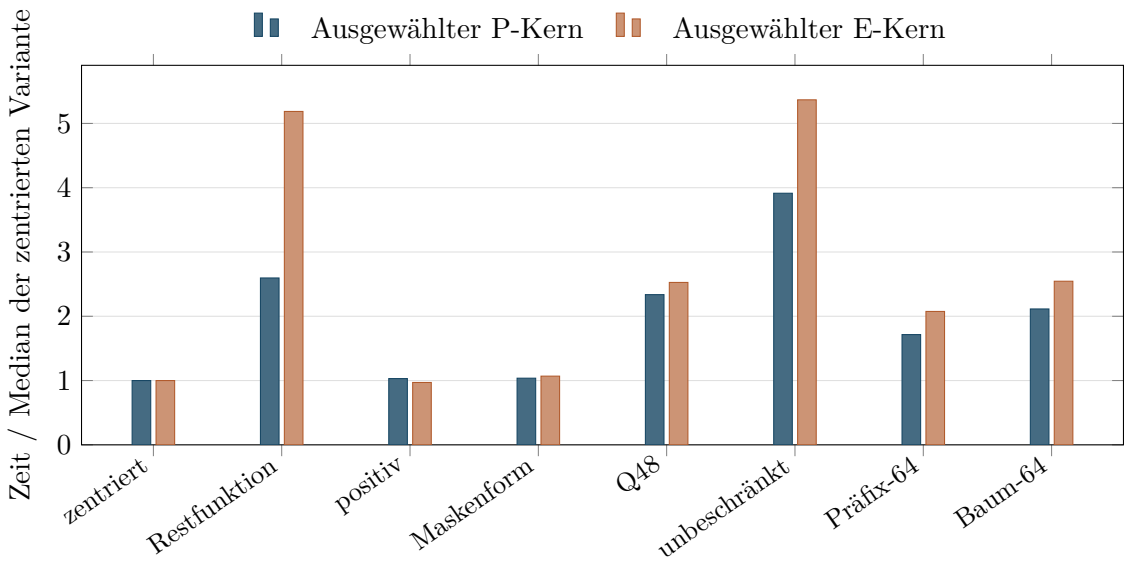

Abbildung 1: FT8-Architekturkosten, innerhalb jeder Kernklasse getrennt normiert. Die Abbildung zeigt Punktschätzer; Intervall- und Empfindlichkeitsdaten stehen in der maschinenlesbaren Zusammenfassung.

Tabelle 6: Getrennte Wiederholung des historischen faktoriellen Harness mit reeller Ausgabe. Diese Verhältnisse dürfen nicht mit dem neuen Vergleich komplexer Ausgabe gleichgesetzt werden.

Modus	Historisch G-L/C-L	Neue Reproduktion, reelle Ausgabe
FT8	5,55	5,56
FT4	5,24	5,21
FT2	4,99	5,19

Der historische Wiederholungslauf prüft, ob sich das frühere Verhalten unter den aktuellen Rechnerbedingungen wieder zeigt. Die Zeiten müssen dabei nicht exakt übereinstimmen. Die Unterschiede zwischen P-/E-Kernen und frei zugewiesenen Läufen erklären, warum die früheren Faktoren 5,55/5,24/4,99 keine rechnerunabhängigen Konstanten sind. Auch Ausgabetyp und Übersetzungskontext unterscheiden sich zwischen dem historischen und dem erweiterten Test.

Gesonderte Durchsatztests starten unabhängige Aufrufe der zentrierten Wellenformerzeugung mit 1, 2, 4 und 8 Arbeitsthreads. Jede Einstellung umfasst fünf Blöcke und acht Frames je Thread, einschließlich Thread-Erzeugung, Arbeitsspeicherbelegung und Cache-Sperren. Diese Tests erlauben freie CPU-Zuweisung; kerngebundene Läufe sind ausgeschlossen. Die vollständige Durchsatztabelle bleibt als Rohdatum erhalten. Sie misst gleichzeitig berechnete unabhängige Frames, keine Vektorisierung eines einzelnen Oszillators, und belegt nicht die Skalierung eines Decoders mit zwölf Arbeitsthreads.

7 Numerische Genauigkeit und Signalprüfung

Die herausgelöste zentrierte Variante stimmt in allen 60 Framekonfigurationen Byte für Byte mit der produktiven Ausgabe überein. Periodisch gleichwertige Alternativen müssen dagegen nicht dieselben Bytes liefern. Nach einer Million konstanter positiver Schritte erreichen zentrierte Korrektur und allgemeine Restbildung einen maximalen Kreisfehler von ungefähr $6{,}86 \cdot 10^{-11}$ rad gegenüber der Long-Double-Referenz mit derselben dargestellten Periode. Der positive Zustand

Tabelle 7: Schlechtestes Ergebnis über den 60-Frame-Korpus gegenüber der Long-Double-Referenz. IQ-Maximum und RMS verwenden den komplexen Betrag; Phasenwerte schließen Rampenbereiche mit Referenzamplitude von höchstens 0,1 aus.

Architektur	Max. IQ-Fehler	Max. RMS-Fehler	Max. Phasenfehler (rad)
Zentriert	8,43e-08	3,20e-08	1,30e-07
Restfunktion	8,43e-08	3,20e-08	1,30e-07
Positiv	8,43e-08	3,20e-08	1,30e-07
Maskenform	8,43e-08	3,20e-08	1,30e-07
Q48-DDS	8,43e-08	3,20e-08	1,30e-07
Unbeschränkt	2,22e-06	8,80e-07	2,27e-06
Serielles Präfix-64	8,43e-08	3,20e-08	1,30e-07
Baumpräfix-64	8,43e-08	3,20e-08	1,30e-07

erreicht $1{,}53 \cdot 10^{-10}$ rad, Q48 etwa $2{,}38 \cdot 10^{-9}$ rad und der unbegrenzte Akkumulator etwa $4{,}40 \cdot 10^{-6}$ rad.

Diese Werte gehören zu einer bestimmten Schrittfolge. Zufällige vorzeichenbehaftete Inkremente verhalten sich anders; die Zahlen sind daher keine allgemeinen Driftraten. Winzige positive Schritte können in Q48 vollständig verschwinden. Die Randfalltests bestätigen außerdem den Mittelpunktsfall der Restfunktion und den Endpunktfall des positiven Zustands aus dem mathematischen Abschnitt.

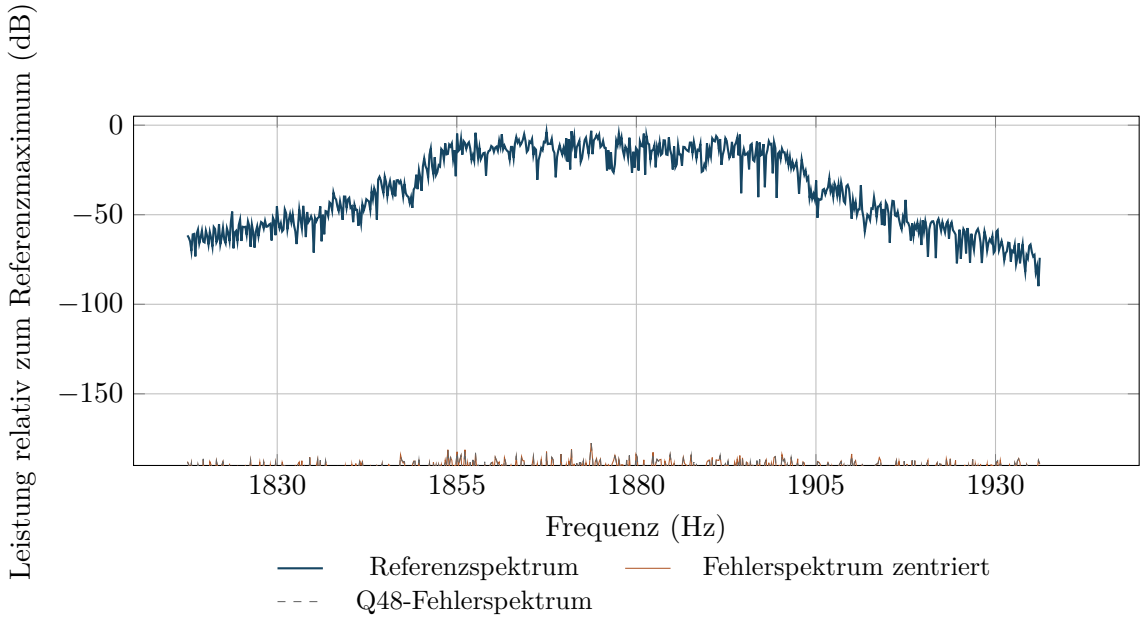

Abbildung 2: Ein erzeugter FT8-Telemetrieframe bei 1855 Hz und 48 ksample/s. Fehlerspektren beruhen auf IQ-Differenzen und nicht auf der Differenz logarithmischer Leistungskurven. Dies ist ein Softwarevergleich und keine Sendermessung.

Für die drei ausgewählten Frames beträgt die integrierte Energie der Differenz zwischen zentrierter Ausgabe und Referenz etwa −150 dBc. Q48 liefert auf diesem Niveau des Float-Ausgabefehlers ein ähnliches Ergebnis. Die relativen Parseval-Fehler liegen unter $7 \cdot 10^{-13}$. Die Referenzenergie außerhalb des gewählten Analysebands beträgt etwa −38,1, −48,3 und −50,9 dBc für FT8, FT4 und FT2. Diese Werte hängen vom Beobachtungsfenster, den Rampen und den Bandgrenzen ab. Sie bestätigen weder eine Emissionsmaske noch eine Verringerung von RF-Nebenkeulen.

Referenz und Varianten verwenden weiterhin dieselben Double-Pulsschritte, dieselbe abschließende Float-Umrechnung und dieselben Rampenkoeffizienten.

8 Praktische Bedeutung für den vollständigen Decoder

Tabelle 8: Unveränderte Wellenformkomponente im vollständigen Einthread-Aufnahmedecoder; Mediane aus drei Läufen nach dem Aufwärmen. Die ideale Grenze entfernt diese Komponente vollständig und hält alle übrige Arbeit konstant.

Modus	Gesamt (ms)	Welle (ms)	Aufrufe	Anteil (%)	Ideale Grenze
FT8	1574,7	33,47	254	2,13	1,022
FT4	177,7	6,63	18	3,73	1,039
FT2	144,6	3,46	5	2,39	1,024

Die Wellenformerzeugung beansprucht in diesen Läufen 2,13% der verstrichenen FT8-Decoderzeit, 3,73% bei FT4 und 2,39% bei FT2. Dieser geringe Anteil begrenzt den Nutzen weiterer Optimierung. Bei vollständigem Wegfall würde die Laufzeit um genau diese Anteile sinken. Die entsprechenden Geschwindigkeitsgewinne wären 2,17%, 3,87% und 2,45%. Zeitersparnis und Geschwindigkeitsgewinn haben unterschiedliche Bezugsgrößen.

Wir wollen diesen Zusammenhang nun allgemein beschreiben. Dazu verwenden wir das Amdahl-Modell. Sei p der Zeitanteil der Komponente im Ausgangszustand und S ihr Beschleunigungsfaktor. Wir setzen die gesamte ursprüngliche Laufzeit gleich eins. Die unveränderte Arbeit benötigt dann $1 - p$, die beschleunigte Komponente p/S. Der gesamte Beschleunigungsfaktor ist der Kehrwert ihrer Summe:

$$S_{\text{total}} = \frac{1}{1 - p + p/S}, \qquad S_{\text{total}} \leq \frac{1}{1 - p}. \tag{5}$$

Für die obere Grenze nehmen wir an, dass die Komponente überhaupt keine Zeit mehr benötigt. In diesem Fall bleibt nur die übrige Arbeit. Sie gilt für den bereits optimierten Generator, diese Aufnahmen und die gemessene Konfiguration mit einem Thread. Sie begrenzt weder den kumulierten Gewinn früherer Änderungen noch sagt sie andere Arbeitslasten voraus.

Mit demselben Modell lässt sich eine hypothetische Rücknahme der Optimierung beschreiben. Benötigt die allgemeine Restbildung um den Faktor G mehr Zeit als die derzeitige zentrierte Komponente, steigt die Gesamtlaufzeit bei sonst gleicher Arbeit um den Faktor $1 + (G - 1)p$. Dafür wird kein exakter Empfangspfadwert behauptet: Abtastrate, Cache-Zustand und Eingaben können G verändern. Insbesondere darf das Verhältnis der Sendewellenformen bei 48 ksample/s nicht einfach für ein gemessenes Verhältnis im Empfangspfad eingesetzt werden.

9 Experimentelle Erweiterung: der CQ-/QSO-Modus 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.

REIST16 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.

9.1 Beobachteter Yaesu-/Pluto-Dialog

Der Benutzer stellte den Original-Screenshot `docs/images/REIST16_QSO.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.

9.2 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 Kernelpuffer; 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, Falschdekodierrate oder Langzeitzuverlässigkeit. Eine unabhängige Wiederholung oder ein Vergleich unter gleichen Bedingungen mit FT8 wird nicht behauptet. Der Modus bleibt experimentell.

10 Diskussion

Der Vergleich spricht für begrenzte Zustandsupdates in dieser GFSK-Implementierung. Er zeigt zugleich, dass ein positives Intervall praktisch dieselbe Geschwindigkeit erreichen kann wie ein zentriertes. Die Zentrierung erleichtert die Darstellung vorzeichenbehafteter Phasenfehler. Unter den genannten binary64-Voraussetzungen ist zudem die korrigierende Subtraktion exakt. Wie groß der Zeitvorteil gegenüber der Bibliotheks-Restfunktion ausfällt, hängt stark vom Kerntyp und Übersetzungskontext ab.

Die langsameren Alternativen zeigen offene Aufgaben. Q48 ersetzt die Zustandsrundung durch Schrittquantisierung und Umrechnungsaufwand. Eine effizientere Umrechnung als der Aufruf von `llround` bei jedem Schritt könnte das Ergebnis verändern. Ebenso entscheidet die für den Standardbefehlssatz übersetzte Baumvariante nicht über den Nutzen von SIMD. Ausdrückliche AVX2-Scans, gemeinsame Puls- und Phasendarstellungen, vektorisierte Umrechnung und die gebündelte Berechnung unabhängiger Oszillatoren sind mögliche weitere Tests. Jede Änderung der Additionsreihenfolge braucht dabei eine neue Fehleranalyse.

Für den vermessenen vollständigen Decoder setzt das Zeitprofil weiteren Gewinnen allein durch die Wellenformerzeugung jedoch eine enge Grenze. Forschung mit dem Ziel, mehr Nachrichten zu decodieren, sollte auch Synchronisation und Fehlerkorrektur untersuchen. Das getrennte Decoder-Dossier behandelt diese Fragen anhand eigener Maße für Erfolg, Aufwand und Falschannahmen.

11 Grenzen der Aussagekraft

Die Laufzeitdaten stammen von einem einzigen physischen Rechner und einer Compilerversion. Es gibt keine AMD- oder ARM64-Messungen, Hardware-Zähler- oder Energiedaten und keine RF-Messungen. Die Kerntypen wurden geprüft, Energieverwaltung und Hintergrundlast aber nicht kontrolliert. Drei Prozessläufe erlauben nur begrenzte Aussagen über Temperatur- und Zuweisungseffekte. Die durch erneutes Ziehen der Daten bestimmten Intervalle können systematische Unsicherheit daher unterschätzen. Auch ist das Verhältnis zusammengefasster Mediane nicht dasselbe wie der Median gepaarter Verhältnisse.

Das historische und das erweiterte Experiment unterscheiden sich in Ausgabetyp, Eingaben und Übersetzungskontext. Beide schließen Pulserzeugung und Rampen ein. Ihre Zeiten beschreiben daher einen vollständigen Wellenformkern, keine isolierte Phasenaddition. Die numerische Referenz verwendet dieselben Double-Schritte und dieselbe dargestellte Periode. Lokale decimal-100-Prüfungen validieren nicht den gesamten Gaußpuls auf 100 Stellen. Zwei Telemetrienachrichten je Einstellung und ein endlicher Testkorpus beweisen keine Bytegleichheit für alle Eingaben. Die Spektren betreffen ausgewählte erzeugte Frames; Audioschnittstelle, USB-Pfad, Verstärker, Antenne und FT-991A wurden nicht vermessen.

Für die Replikation auf einem echten AMD- oder ARM64-Rechner liegt ein Messskript bei. Es verwendet neue Ausgabeverzeichnisse und zeichnet Quellhashes, Werkzeugkette und Ausführungsbedingungen für dieselbe Wellenformaufgabe auf. Diese Replikation steht noch aus. Eine x86-VM oder ein emulierter ARM-Prozess auf diesem i9 wäre keine zweite physische Plattform. Auch die Long-Double-Genauigkeit muss für jedes ABI erfasst werden; der hier verwendete 64-Bit-Signifikand ist keine allgemeine Eigenschaft.

Eine externe Replikation, unabhängige mathematische Prüfung und Journalbegutachtung stehen noch aus.

12 Schlussfolgerungen

Sind Phase und Schritte begrenzt, hält eine einzige Korrektur den Oszillatorzustand innerhalb einer Periode. Die Begründung über reellen Zahlen gilt auch für das beschriebene binary64-Update, weil die korrigierende Subtraktion exakt ist. Rundung bei der vorhergehenden Addition und bei der Amplitudenberechnung muss weiterhin berücksichtigt werden.

Auf dem vermessenen i9-14900K ist dieses Update deutlich schneller als die allgemeine Restbildung. Ein positiver Zustand ist etwa gleich schnell wie ein zentrierter; die gemessenen Verhältnisse unterscheiden sich zwischen Performance- und Efficiency-Kernen. REIST liefert den Ausgangspunkt für die zentrierte Zustandsformulierung. Der empirische Beitrag liegt im kontrollierten Vergleich und dessen numerischer Prüfung. Das Anwendungsprofil bestimmt die praktische Größenordnung: Weitere Optimierung allein der Wellenformerzeugung kann die Verarbeitung dieser Dateidecoder-Aufgaben höchstens um etwa 2,2–3,9% beschleunigen. Größere Empfangsverbesserungen erfordern eigene Nachweise im übrigen Empfänger.

13 Verfügbarkeit von Daten und Code

Der Anwendungscode ist unter `https://github.com/rudolfstepan/wsjtx` verfügbar. Die vermessene produktive Basis ist Commit `46ebbd78c663bf07eed054570df4888a6268e156`; die Forschungsquellen und Version 0.2 dieser Studie wurden mit Commit

`3f1a297ed05e4351553ee50c33934c1cd3d0675b` veröffentlicht. Das beigefügte Paket identifiziert die aktuellen Dokumente und die tatsächlich vermessenen Quellversionen durch Hashes. Version 0.4 ergänzt die gesondert ausgewiesene REIST16-Implementierung und den Hardwareversuch des Benutzers. Die historischen Phasenmessungen bleiben unverändert. Forschungsmanuskripte, Rohdaten und PDFs werden im privaten Repository des Autors geführt; das öffentliche Anwendungsrepository enthält Protokoll, Bedienungsdokumentation und Original-QSO-Screenshot.

Historische Rohdaten bleiben unter `docs/papers/reist-ftx/data`; die erweiterte Studie verwendet `docs/papers/reist-ftx/journal/results`. Versionierte Python-Skripte erzeugen Tabellen und Diagramme, ohne die beiden Datensätze zusammenzulegen. Die bereitgestellte ursprüngliche REIST-PDF und optionale FT2-Aufnahmen Dritter werden nicht weitergegeben.

`Build-Study.ps1` führt den dokumentierten Ablauf für Build, Auswertung und LaTeX aus und kann neue Messverzeichnisse anlegen. Die eigenständigen Wellenformexperimente benötigen GNU C++, Boost-Header, CMake, Ninja und Python; die historische Reproduktion zusätzlich Fortran. Vollständige Decoder- und Profiltests brauchen den gesamten Fork mit seinen üblichen Abhängigkeiten. Der Quellcode behält GPLv3 und die Urheberhinweise der Ausgangssoftware bei. Eine Publikationslizenz für den Entwurf wurde noch nicht gewählt; eine Journaleinreichung oder neue DOI-Registrierung ist nicht erfolgt.

Autorenbeiträge und KI-Unterstützung

Rudolf Stepan stellte die REIST-Formulierung, den bestehenden Quellcode-Fork, die Forschungsziele und das Originalmanuskript bereit. OpenAI Codex unterstützte Quelltextprüfung, experimentelle Implementierung, lokale Ausführung, Auswertung, Abbildungen und wesentliche Teile des Textentwurfs, einschließlich dieser Überarbeitung sowie der REIST16-Implementierung und Befundauswertung. Wissenschaftliche Prüfung und Verantwortung für eine Einreichung liegen beim Autor; journalspezifische KI-Richtlinien sind gesondert zu prüfen. Die ursprünglichen Entwickler der WSJT-X-Protokolle und Software sowie die dokumentierten FT2-Beitragenden behalten ihre Urheber- und Lizenzanerkennung.

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; öffentliche Metadaten nicht unabhängig abgerufen. SHA256 der Original-PDF im Audit.

[2] R. Stepan, *REIST Division als Zustandsprinzip in den GFSK Rechenkernen von FT8 FT4 und FT2*, technischer Bericht v1.0, 15. Oktober 2026, `docs/papers/reist-ftx/paper.tex`, mit Commit 46ebbd78c archiviert.

[3] Analog Devices, *MT-085: Fundamentals of Direct Digital Synthesis (DDS)*, technisches Tutorial. `https://www.analog.com/media/en/training-seminars/tutorials/mt-085.pdf`.

[4] D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, Bd.23, Nr.1, S.5–48, 1991. Maßgeblicher Nachdruck: `https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html`.

[5] G. E. Blelloch, *Prefix Sums and Their Applications*, Carnegie Mellon University, CMU-CS-90-190, November 1990. `https://www.cs.cmu.edu/afs/cs.cmu.edu/project/scandal/public/papers/CMU-CS-90-190.html`.

[6] 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`.

[7] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual*, CPUID-Blatt 1AH, Aufzählung hybrider Kerntypen. `https://cdrdv2-public.intel.com/868137/325462-089-sdm-vol-1-2abcd-3abcd-4.pdf`.

[8] Microsoft, *QueryThreadCycleTime function*, Win32-API-Dokumentation. `https://learn.microsoft.com/en-us/windows/win32/api/realtimeapiset/nf-realtimeapiset-querythreadcycletime`.

[9] L. Cordesses, "Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)," *IEEE Signal Processing Magazine*, Bd.21, Nr.4, S.50–54, 2004. DOI: 10.1109/MSP.2004.1311140.

[10] J. Vankka, *Direct Digital Synthesizers: Theory, Design and Applications*, Dissertation, Helsinki University of Technology, November 2000, Kapitel 5–6 und 9. `https://aaltodoc.aalto.fi/server/api/core/bitstreams/82529cd1-8a15-4773-bbc3-18a27ce66376/content`.

[11] J. Vankka, "Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis," *1996 IEEE Frequency Control Symposium*, S.942–950, 1996. `https://research.aalto.fi/en/publications/methods-of-mapping-from-phase-to-sine-amplitude-in-direct-digital/`.

[12] J. Vankka, "Spur Reduction Techniques in Sine Output Direct Digital Synthesis," *1996 IEEE Frequency Control Symposium*, S.951–959, 1996. `https://research.aalto.fi/en/publications/spur-reduction-techniques-in-sine-output-direct-digital-synthesis/`.

[13] K. I. Palomäki und J. Nurmi, “Taylor Series Interpolation-Based Direct Digital Frequency Synthesizer with High Memory Compression Ratio,” *Sensors*, Bd.25, Nr.8, Artikel 2403, 2025. DOI: 10.3390/s25082403.

[14] M. Vicanek, *A New Recursive Quadrature Oscillator*, vom Autor veröffentlichte technische Notiz, 21. Oktober 2015. `https://vicanek.de/articles/QuadOsc.pdf`.