Journal of Electronics, Electromedical Engineering, and Medical Informatics
Homepage: jeeemi.org; Vol. 8, No. 4, October 2026, pp: 1719-1734 e-ISSN: 2656-8632

RESEARCH ARTICLE OPEN ACCESS

TPE-FFT: A Two-Phase Execution Strategy for Low-Latency FFT-Based Sequential Genomic Signal Processing

Saravanakumar C, Usha Bhanu N, Subhashini N and Preethi Sambandam Raju

Department of Electronics and Communication Engineering, SRM Valliammai Engineering College, Kattankulathur, India

Corresponding author: Saravanakumar C. (e-mail: kanchi.saravana@gmail.com)

Author(s) Email: Usha Bhanu N (e-mail: ushabhanu123@gmail.com), Subhashini N (e-mail: subhashinink@gmail.com), Preethi Sambandam Raju (e-mail: srpreethi31090@gmail.com)

Abstract **The Fast Fourier Transform (FFT) is widely employed for frequency-domain analysis of genomic and biomedical signals; however, conventional block-oriented FFT processing generally requires the complete input sequence before the transform can be finalized. When genomic data are acquired sequentially, computations that depend on the final sample may remain unresolved until that sample arrives, resulting in additional post-arrival processing latency. This study presents a Two-Phase Execution Fast Fourier Transform (TPE-FFT), a dependency-aware execution strategy for radix-2 Decimation-in-Time (DIT) FFT processing. The proposed method exploits the linearity of the Discrete Fourier Transform (DFT) to separate the contribution of the final acquired sample from the contributions of the preceding samples. During Phase I, computations independent of the final sample are performed using a temporary zero representation, while the corresponding final-sample contribution is evaluated during Phase II and incorporated through an exact Fourier-domain correction. Variable-length genomic sequences are mapped to a common radix-2 transform length using zero-padding, and the proposed method is evaluated against a conventional radix-2 DIT-FFT implementation under the same computational conditions. A total of 1,075 genomic sequences are evaluated. TPE-FFT achieves positive measured execution-time reduction for 1,053 sequences (97.95%). The cumulative execution time decreases from 225.56469 s for the conventional implementation to 7.80065 s for TPE-FFT, corresponding to a 96.54% aggregate reduction and a 28.92× runtime ratio. For a representative 33,681-sample genomic sequence, execution time decreases from 15.15 s to 0.51 s, corresponding to a 96.63% reduction. The results indicate that dependency-aware scheduling can substantially reduce the computational workload remaining after final-sample arrival while retaining the underlying $O(N\log_2 N)$ FFT complexity. The proposed strategy therefore provides a computational basis for further investigation of low-latency FFT processing in sequential genomic signal-processing applications.**

Keywords **Two Phase Execution FFT; Low-latency computing; Genomic Signal Processing; Integrated Engineering Systems; Biomedical Signal Processing**.

I. Introduction

The Fast Fourier Transform (FFT) is a fundamental computational technique for evaluating the Discrete Fourier Transform (DFT) and is widely used in communications, biomedical signal processing, image processing, and embedded computing. Classical FFT formulations, including Cooley–Tukey, Split-Radix, Bruun, Rader, Prime-Factor, Hexagonal FFT, Bluestein, and Winograd algorithms, have primarily focused on reducing arithmetic complexity and improving computational efficiency [1]–[7]. These approaches have established the theoretical and architectural foundations for efficient frequency-domain processing across a wide range of applications.

In biomedical and genomic signal processing, the input data may be acquired sequentially rather than being available as a complete block before computation begins. Genomic sequences are also characterized by considerable variation in sequence length, which makes computational latency an important consideration in streaming and near-real-time processing environments. Conventional block-based FFT computation generally assumes that the complete input sequence is available before the

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

transform is finalized. Consequently, when the final sample x[N−1] arrives after the preceding samples, FFT operations that depend on it may remain unresolved until it arrives. This creates a post-arrival computational dependency that is distinct from the conventional objective of minimizing the total number of FFT operations. Several FFT architectures have addressed computational efficiency, data sequencing, throughput, and hardware utilization. Input-sequencing strategies have been investigated to reduce coefficient dependencies and improve the organization of butterfly operations [8]. Real-valued FFT architectures exploit conjugate symmetry to reduce arithmetic requirements, while single-processing-element, in-place, and pipelined implementations have been explored for biological signal processing [9]. Hardware-oriented radix-2 FFT implementations have also been optimized for high operating frequency using pipelined multiplier structures [10]. CORDIC-based architectures provide an alternative means of implementing complex rotations while reducing the requirement for stored rotation coefficients [11]. These studies demonstrate that existing FFT research has extensively addressed arithmetic efficiency, throughput, and hardware implementation; however, they do not specifically address the computational dependency created by the late arrival of the final sample in a sequentially acquired FFT block. Further developments have explored FFT computation from both algorithmic and architectural perspectives. Arithmetic restructuring has been used to reduce redundant operations within FFT butterfly structures [12], while optimized multiplier architectures have been investigated for reducing hardware and power requirements [13]. Alternative transform formulations, including FFT-based periodic interpolation and fractional Fourier transforms, have extended frequency-domain computation to specialized mathematical domains [14], [15]. Metaheuristic frequency-domain optimization has also been investigated [16], while processing-in-memory architectures and hardware–software co-design approaches have targeted high-throughput data-intensive applications [17], [18]. Broader developments in bioinformatics acceleration have incorporated increasingly parallel and emerging accelerator technologies [19], and frequency-domain processing has also been combined with secure computational frameworks for biological sequence analysis [20]. These developments demonstrate the continuing effort to improve computational efficiency, but their primary objectives differ from the late-sample dependency considered in this work.

FFT-based spectral processing has also been incorporated into biomedical and biological applications. Frequency-domain processing has been used as part of computational pipelines for retinal image analysis and disease detection [21], [22], while Fourier-domain techniques have been incorporated into biological monitoring and other signal-processing applications [23]. Other bioinformatics studies have combined numerical sequence representations [24]-[26], digital filtering, signal-processing architectures, and advanced Fourier-based techniques for genomic and biological data analysis [27]–[32]. These applications highlight the importance of efficient frequency-domain processing for biological data, but the computational latency associated specifically with sequential sample arrival remains insufficiently addressed. Recent research on FFT and related architectures has increasingly emphasized streaming, pipelining, and distributed execution [33]–[35]. Such architectures improve throughput by distributing FFT operations across processing stages or time intervals. However, in a conventional block-oriented streaming or pipelined FFT, completion of the complete transform remains dependent on the availability and propagation of the final input sample through the relevant butterfly stages. Thus, improving throughput does not necessarily eliminate the post-arrival dependency associated with the final sample. This distinction motivates an alternative execution strategy in which computations that are independent of the final sample are performed before that sample becomes available.

Despite extensive research on FFT arithmetic optimization, pipelining, streaming architectures, online transforms, and hardware acceleration, relatively limited attention has been given to the specific computational dependency created when the final sample of a sequentially acquired FFT block becomes available later than the preceding samples. Existing approaches primarily optimize total arithmetic complexity, throughput, hardware utilization, or repeated spectral updates, whereas the present study considers the temporal distribution of FFT computation with respect to the arrival of the final sample. The key novelty of TPE-FFT is therefore not a new DFT identity or a reduction in the asymptotic FFT complexity; instead, it is a dependency-aware execution schedule within a radix-2 DIT butterfly network. Computations independent of the final acquired sample are moved into a pre-arrival phase, while the remaining final-sample contribution is resolved after sample arrival through an exact Fourier-domain correction. This distinction enables the study to separately analyze total FFT complexity and the remaining computational workload after final-sample arrival.

The proposed Two-Phase Execution Fast Fourier Transform (TPE-FFT) separates the computation into two execution phases. In Phase I, the final sample is temporarily represented by zero, and all butterfly operations that are independent of that sample are executed as early as permitted by the dependency

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

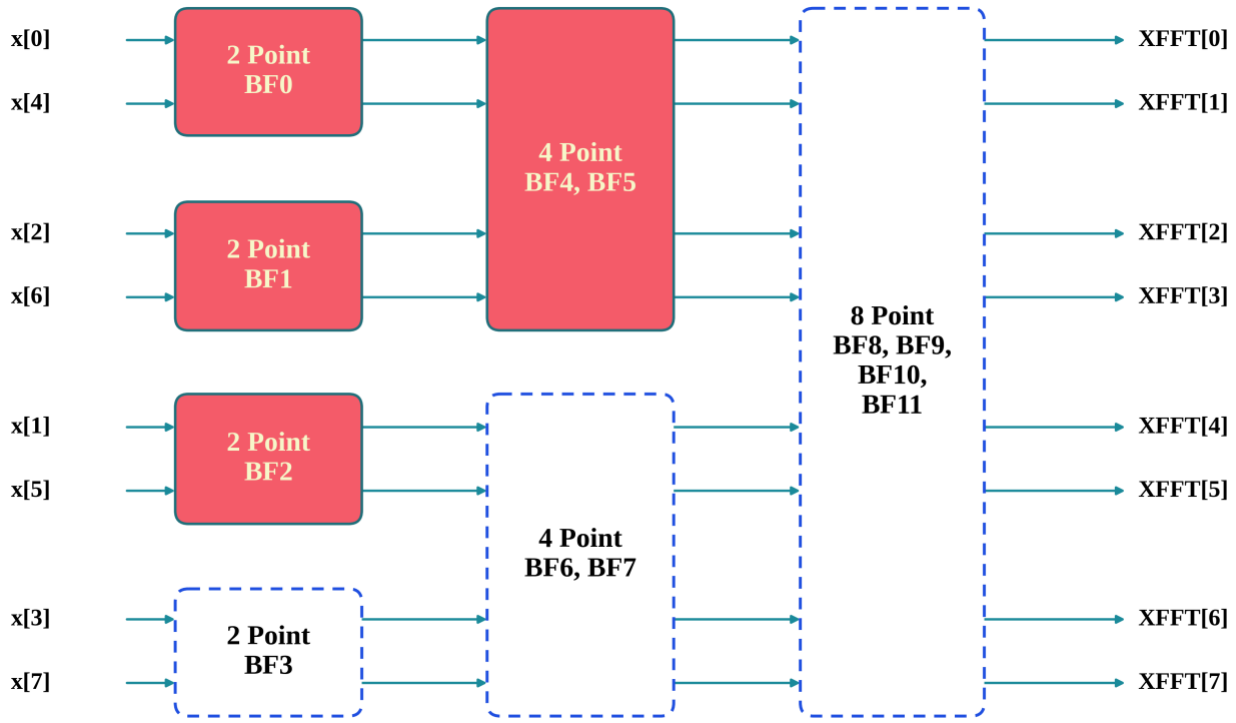


Fig. 1. Radix-2 decimation-in-time (DIT) butterfly dependency structure for an 8-point FFT, illustrating the computational stages affected by the arrival of the final input sample, x[7], under zero-based indexing.

graph. In Phase II, the actual final sample is incorporated through its exact Fourier contribution. This scheduling principle is motivated by DFT linearity. The principal contributions of this paper are therefore: (1) a dependency-aware two-phase execution schedule for a radix-2 DIT FFT; (2) an exact final-sample correction derived directly from the DFT definition; (3) an explicit separation of total FFT complexity from post-arrival workload; and (4) a sequence-level evaluation across 1,075 genomic inputs, including statistical analysis of positive and negative performance cases.

II. Materials and Methods

A. Genomic Dataset and Numerical Representation

The genome sequence data for this study were downloaded from the National Center for Biotechnology Information (NCBI) GenBank database [36]. It contains 1075 nucleotide sequences of human retinal genes, chosen for their relevance to biomedical signal analysis and their length variation. The sequence size varies from a few hundred bases to a few million bases for longer sequences, covering the entire genome, which is well suited for testing the performance of FFT algorithms with different loads. Genomic sequences are represented using nucleotide symbols A, C, G, and T, which must be converted into numerical form prior to spectral analysis. In this work, a weighted numerical mapping scheme was employed, where each nucleotide is assigned, a numerical value based on its occurrence characteristics, as described in prior studies [37]. This encoding enables the transformation of symbolic DNA sequences into discrete-time numerical signals suitable for FFT-based processing.

B. Conventional Radix-2 DIT FFT

The baseline method used for comparison is the conventional radix-2 Decimation-in-Time (DIT) FFT algorithm. In this approach, FFT computation is initiated only after the complete input sequence becomes available. The algorithm performs a sequence of butterfly operations across multiple stages, where a significant number of arithmetic operations remain dependent on the final input sample as shown in Fig. 1. The XFFT[0] can be calculated only upon receiving the x[7] sample.

$$XFFT[0] = [(x[0] + x[4]) + (x[2] + x[6])W_8^0] + [(x[1] + x[5]) + (x[3] + x[7])W_8^0]W_8^0 \quad (1)$$

$$XFFT[1] = [(x[0] - x[4]) + (x[2] - x[6])W_8^2] + [(x[1] - x[5]) + (x[3] - x[7])W_8^2]W_8^1 \quad (2)$$

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

$$XFFT[2] = [(x[0] + x[4]) - (x[2] + x[6])W_8^0] + [(x[1] + x[5]) - (x[3] + x[7])W_8^0]W_8^2 \quad (3)$$

$$XFFT[3] = [(x[0] - x[4]) - (x[2] - x[6])W_8^2] + [(x[1] - x[5]) - (x[3] - x[7])W_8^2]W_8^3 \quad (4)$$

$$XFFT[4] = [(x[0] + x[4]) + (x[2] + x[6])W_8^0] - [(x[1] + x[5]) + (x[3] + x[7])W_8^0]W_8^0 \quad (5)$$

$$XFFT[5] = [(x[0] - x[4]) + (x[2] - x[6])W_8^2] - [(x[1] - x[5]) + (x[3] - x[7])W_8^2]W_8^1 \quad (6)$$

$$XFFT[6] = [(x[0] + x[4]) - (x[2] + x[6])W_8^0] - [(x[1] + x[5]) - (x[3] + x[7])W_8^0]W_8^2 \quad (7)$$

$$XFFT[7] = [(x[0] - x[4]) - (x[2] - x[6])W_8^2] - [(x[1] - x[5]) - (x[3] - x[7])W_8^2]W_8^3 \quad (8)$$

This dependency results in increased execution latency, particularly for long and sequentially acquired input signals as depicted in Eq. (1-8) [19]. For instance, if the input sample is not gathered, the butterfly components BF3 and from BF6 to BF11 cannot be calculated until the data input x(7) is received. The last term in each equation requires the last sample x[7]. Eq. (9) [19] provides the number of incomplete butterfly components required to compute the N-point FFT with Q stages, once the last incoming data has been received.

$$Number\ of\ Butterfly = \sum_{q-1}^{Q} 2^{q-1} \quad (9)$$

In an 8-point FFT, x[7] denotes the eighth input sample under zero-based indexing. Before x[7] arrives, butterfly computations that are independent of x[7] can be evaluated using a zero-substituted partial input. The remaining influence of x[7] is subsequently incorporated through the exact correction term derived from the DFT definition. Moreover, since the number of stages is log2N, the number of pending butterfly units in terms of N is given by Eq. (10) [19].

$$Number\ of\ Pending\ Butterfly = \sum_{q-1}^{Q} 2^{q-1} = N - 1 \quad (10)$$

Since a single butterfly unit requires two complex additions and one complex multiplication, from Equation (10), 2(N-1) complex additions and (N-1) complex multiplications are yet to be computed on the arrival of the last sample. Table 1 presents the computations that are necessary in the event of an augmentation in the number of data points, denoted as N. This indicates 2that a significant amount of time is committed, which has an impact on the system's performance, particularly when the sample data is inherently longer.

C. Proposed Fast Fourier Transform

The proposed Two-Phase Execution Fast Fourier Transform (TPE-FFT) reorganizes the execution schedule of a conventional radix-2 Decimation-in-Time (DIT) FFT without modifying the underlying Fourier-transform definition. The central principle is to separate the contribution of the final input sample from the contributions of the preceding samples. Before the final sample becomes available, a partial transform is computed by temporarily assigning zero to the final input position. Once the actual final sample arrives, its corresponding Fourier-domain contribution is evaluated and incorporated into the previously computed partial spectrum. The formulation can be derived directly from the N-point Discrete Fourier Transform (DFT), given by Eq. (11) [19]

$$\boldsymbol{X[k]} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} \boldsymbol{x[n]W_N^{nk}}, \quad \boldsymbol{k = 0, 1, \ldots, N-1} \quad (11)$$

where $W_N = e^{-j2\pi/N}$ denotes the N-point twiddle factor. Separating the contribution of the final input sample x[N−1] from the remaining samples yields as in Eq. (12) [19]

$$\mathrm{X[k]} = \sum_{\mathrm{n=0}}^{\mathrm{N-2}} \mathrm{x[n]W_N^{nk}} + \mathrm{x[N-1]W_N^{(N-1)k}} \quad (12)$$

Table 1. Conventional post-arrival arithmetic requirements under the counting convention used in this study

Number of data points N	Post-arrival complex multiplications	Post-arrival complex additions
8	7	14
16	15	30
64	63	126
128	127	254
256	255	510
512	511	1022

The first term represents the spectral contribution of all samples preceding the final sample and is defined as the partial spectrum in Eq. (13) [19],

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

$$X_p[k] = \sum_{n=0}^{N-2} x[n] W_N^{nk} \quad (13)$$

Consequently, the complete DFT can be reconstructed as

$$X[k] = X_p[k] + x[N-1] W_N^{(N-1)k}, \quad k = 0.1 \ldots\ldots, N-1 \quad (14)$$

This formulation demonstrates that the proposed two-phase representation is mathematically equivalent to the conventional DFT. The decomposition itself is a direct consequence of DFT linearity and is therefore not claimed as a new Fourier-transform identity. Instead, the contribution of TPE-FFT lies in exploiting this decomposition as a dependency-aware execution strategy within the radix-2 DIT butterfly network. Specifically, butterfly computations that do not depend on the final sample are scheduled during the pre-arrival phase, while the residual contribution associated with the final sample is resolved after its arrival.

Eq. (14) [19] can be expanded to Eq.(15 - 22) [19] to compute 8 point FFT in comparison to conventional computation of FFT.

$$X[0] = [(x[0] + x[4]) + (x[2] + x[6])W_8^0] + [(x[1] + x[5]) + (x[3])W_8^0] + x[7]W_8^0 \quad (15)$$

$$X[1] = [(x[0] - x[4]) + (x[2] - x[6])W_8^2] + [(x[1] - x[5]) + (x[3])W_8^2] - x[7]W_8^1 \quad (16)$$

$$X[2] = [(x[0] + x[4]) - (x[2] + x[6])W_8^0] + [(x[1] + x[5]) - (x[3])W_8^0] + x[7]W_8^2 \quad (17)$$

$$X[3] = [(x[0] - x[4]) - (x[2] - x[6])W_8^2] + [(x[1] - x[5]) - (x[3])W_8^2] - x[7]W_8^3 \quad (18)$$

$$X[4] = [(x[0] + x[4]) + (x[2] + x[6])W_8^0] - [(x[1] + x[5]) + (x[3])W_8^0] + x[7]W_8^0 \quad (19)$$

$$X[5] = [(x[0] - x[4]) + (x[2] - x[6])W_8^2] - [(x[1] - x[5]) + (x[3])W_8^2] - x[7]W_8^1 \quad (20)$$

$$X[6] = [(x[0] + x[4]) - (x[2] + x[6])W_8^0] - [(x[1] + x[5]) - (x[3])W_8^0] + x[7]W_8^2 \quad (21)$$

$$X[7] = [(x[0] - x[4]) - (x[2] - x[6])W_8^2] - [(x[1] - x[5]) - (x[3])W_8^2] - x[7]W_8^3 \quad (22)$$

D. Complexity and Latency Model

The proposed TPE-FFT does not alter the asymptotic arithmetic complexity of the radix-2 FFT. Both the conventional FFT and TPE-FFT retain an overall computational complexity of O(Nlog2N). The principal objective of TPE-FFT is instead to redistribute the computational workload over the data-arrival interval, thereby reducing the amount of computation that remains after the final input sample becomes available. Accordingly, total computational workload and post-arrival computational workload are considered separately in the analysis. Under the operation-counting convention adopted in this study, the conventional late-arrival computation requires approximately N−1 complex multiplications and 2(N−1) complex additions. In comparison, the TPE-FFT correction phase requires approximately N/2 complex multiplications and N complex additions. These expressions represent the operations associated with the post-arrival phase and should not be interpreted as a reduction in the total number of arithmetic operations required to evaluate the complete FFT. Table 2

Table 2. Comparison of post-arrival arithmetic requirements for conventional FFT and TPE-FFT Range of Improvement in Speed of Calculation.

Transform length N	Conventional multiplications	Conventional additions	TPE-FFT multiplications	TPE-FFT additions
8	7	14	4	8
16	15	30	8	16
64	63	126	32	64
128	127	254	64	128
256	255	510	128	256
512	511	1022	256	512

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

Algorithm 1. Two-Phase Execution FFT

Input: Sequential samples x[0],x[1],…,x[N−1]

(1) Determine the input sequence length L and the effective radix-2 transform length $M=2^{\lfloor \log_2 L \rfloor}$

(2) Initialize the input and working buffers of length M.

(3) Store the genomic samples x[0],x[1],…,x[L−1] in their arrival order and append zeros for the remaining M−L positions when L<M.

(4) Identify the last acquired sample x[L-1]. Treat the Zero -padding positions x[L], x[L+1]…x[M-1] as known values at t=0 and include them in the pre-arrival phase, x[L-1] is assigned a temporary value of zero. the effective transform, x[M−1]. During the pre-arrival phase, this sample is assigned a temporary value of zero.

(5) Execute all radix-2 DIT butterfly operations whose results are independent of the actual value of x[L−1].

(6) Maintain the required butterfly data paths, twiddle-factor indices, and intermediate values during the first execution phase.

(7) When the actual final sample x[L−1] becomes available, generate the correction vector

$$\mathrm{C}[k] = \mathrm{x}[\mathrm{L}-1]\mathrm{W}_{\mathrm{M}}^{(\mathrm{L}-1)\mathrm{k}}, \mathrm{k} = 0,1,\ldots,\mathrm{M}-1$$

(8) Add the correction vector to the partial spectrum according to

$$\mathrm{X}[\mathrm{k}] = \mathrm{X}_{\mathrm{P}}[\mathrm{k}] + \mathrm{C}[\mathrm{k}]$$

(9) Perform the required output-ordering or bit-reversal operation if it has not already been incorporated into the butterfly schedule.

(10) Return the complete M-point spectrum.

summarizes the theoretical post-arrival arithmetic requirements for representative transform lengths.

E. TPE-FFT Execution Procedure

The execution procedure of TPE-FFT is summarized in Algorithm 1. The algorithm receives the input samples sequentially and divides the computation into a pre-arrival phase and a final-sample correction phase. For an eight-point FFT, the final input sample is x[7]. During Phase I, x[7] is temporarily assigned a zero value, allowing all butterfly operations independent of x[7] to be completed. During Phase II, the actual value of x[7] is incorporated through the correction term $\boldsymbol{x[7]W_8^{7k}}$. The complete spectrum is consequently obtained as in Eq. (23) [19].

$$X[k] = X_P[k] + x[7]W_8^{7k}, k = 0,1,\ldots,7 \qquad (23)$$

This operation produces the same frequency-domain result as the conventional eight-point DFT while enabling a portion of the computation to be completed before the final sample becomes available. The procedure is depicted in Fig. 2.

F. Timing Protocol and Performance Metrics

The execution-time evaluation compares the conventional DIT-FFT and the proposed TPE-FFT under the same computational environment. The elapsed execution time of each implementation is measured using tic and toc functions (MATLAB 2025b on an Intel Core 7 Processor with 8GB RAM under Windows 10 OS). The conventional implementation executes the DIT-FFT computation, whereas the proposed implementation performs the two-phase execution and final-sample correction. The timing framework is designed to provide a consistent function-level comparison between the two approaches. The execution-time measurement therefore represents the computational time associated with the respective FFT implementations under identical experimental conditions. The percentage reduction in execution time is calculated using Eq. (24) [19]

$$R(\%) = \frac{T_{conv} - T_{TPE}}{T_{conv}} X\ 100 \qquad (24)$$

where Tconv represents the execution time of the conventional DIT-FFT and TTPE represents the execution time of TPE-FFT. A positive value of R indicates that the proposed method requires less execution time than the conventional implementation. Throughout this paper, the term execution-time reduction is used instead of speedup percentage to avoid ambiguity between percentage reduction in execution time and multiplicative acceleration. The MATLAB fft function is used as a reference transform within the signal-processing workflow for frequency-domain analysis, while the comparative execution-time evaluation focuses on the conventional DIT-FFT and TPE-FFT implementations.

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

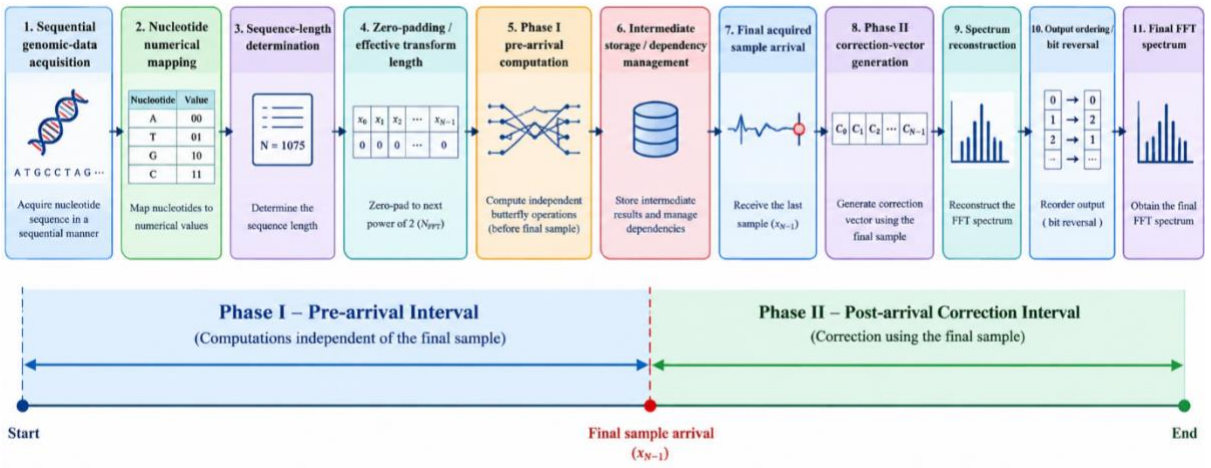


Fig. 2. System-level workflow of the proposed Two-Phase Execution Fast Fourier Transform (TPE-FFT), illustrating genomic sequence acquisition, nucleotide numerical mapping, zero-padding, pre-arrival FFT computation (Phase I), final-sample correction (Phase II), and spectral reconstruction.

G. Numerical Equivalence and Statistical Analysis

The mathematical equivalence of TPE-FFT to the conventional DFT follows directly from the formulation presented in Section II-C. Since the complete spectrum is obtained by adding the Fourier-domain contribution of the final input sample to the partial spectrum, the proposed formulation reconstructs the same DFT as the conventional computation. The performance evaluation considers 1,075 genomic sequences, with the conventional FFT and TPE-FFT execution times obtained for the same sequence in each paired observation. The sequence is therefore treated as the experimental unit for statistical comparison. Across the complete dataset, the mean execution-time reduction is 86.46%, with a median reduction of 94.15%. The standard deviation is 23.09 percentage points, while the interquartile range (IQR) is 8.32 percentage points. A paired-samples t-test was performed to determine whether the difference between the conventional and proposed execution times was statistically significant. The analysis produced t(1074)=8.31,p=2.79×10−16. A non-parametric Wilcoxon signed-rank test was additionally performed, yielding p=8.12×10−177. The 95% confidence interval for the mean paired execution-time difference was 0.15476–0.25038 s. The combination of descriptive and inferential statistics provides an overall assessment of the execution-time behavior of TPE-FFT across the evaluated genomic sequence dataset. The -level distribution is further examined in the Results section to identify both substantial performance sequence improvements and cases in which the proposed execution strategy provides limited or negative improvement.

III. Results

A. Sequence-Level Execution Time Reduction

The experimental evaluation comprises 1,075 genomic sequence observations. The conventional DIT-FFT implementation has a cumulative execution time of 225.56469 s, whereas TPE-FFT has 7.80065 s. Based on these measurements, TPE-FFT achieves an aggregate execution-time reduction of 96.54%, corresponding to a 28.92× lower runtime than the conventional implementation. Sequence-level execution-time comparison between the conventional DIT- FFT and TPE – FFT implementations for 1075 genomes are illustrated in Fig. 3. At the individual sequence level, 1,053 of the 1,075 sequences (97.95%) exhibit a positive execution-time reduction with TPE-FFT. The remaining 22 sequences (2.05%) exhibit negative reductions, indicating that the proposed

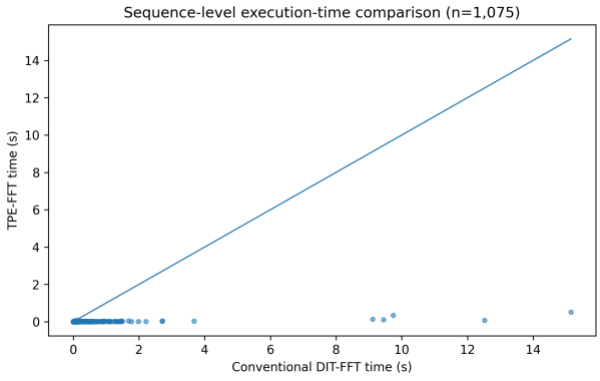


Fig. 3. Scatter plot comparing the measured execution times of the conventional radix-2 DIT-FFT and proposed TPE-FFT implementations across 1,075 genomic sequences. The diagonal reference line indicates equal execution times for both methods.

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

Table 3. Distribution of measured execution-time reduction across the 1,075 sequences.

Improvement range	Count	Percentage	Mean improvement	Median improvement	IQR
>80%	917	85.30%	93.61%	94.89%	4.98%
60–80%	75	6.98%	71.94%	72.96%	6.96%
40–60%	42	3.91%	51.60%	50.98%	5.06%
20–40%	8	0.74%	35.90%	37.57%	4.10%
0–20%	11	1.02%	7.15%	5.65%	5.16%
<0%	22	2.05%	−37.44%	−37.10%	24.55%

approach does not provide a uniform benefit for every input. These cases are analyzed separately in Section III-B. As shown in Table 3, the performance distribution is strongly concentrated toward higher execution-time reductions. In particular, 917 sequences (85.30%) achieve more than 80% reduction, while an additional 75 sequences (6.98%) achieve reductions between 60% and 80%. Thus, 92.28% of the evaluated sequences demonstrate an execution-time reduction exceeding 60%. The relatively small number of negative-performance cases, as in Fig. 4, emphasizes that the effectiveness of TPE-FFT depends on the relationship between the computational workload and the overhead associated with the two-phase execution strategy.

B. Negative Performance Cases and Sequence Length Dependence

Although TPE-FFT substantially reduces execution time for most evaluated sequences, 22 sequences (2.05%) show negative performance, as shown in Fig. 3. These observations indicate that the proposed execution schedule is not universally advantageous under all input conditions. In such cases, the overhead associated with two-phase scheduling, buffer operations, correction-vector generation, and function-level processing can become comparable to or greater than the benefit obtained by executing part of the FFT before the final sample arrives. This behavior is consistent with the underlying computational formulation. TPE-FFT does not eliminate the overall O(Nlog2N) complexity of the FFT; rather, it redistributes a portion of the computation from the post-arrival interval to the pre-arrival interval. Consequently, the practical benefit depends on the amount of computation that can be completed before the final sample becomes available and on the implementation overhead associated with the two-phase schedule. The arithmetic complexity of the underlying radix-2 FFT is therefore O(Nlog2N) and should not be characterized as exponential. The correction component grows approximately linearly with the transform length, whereas the benefit of early execution depends on the butterfly-dependency structure and the relative timing of sample acquisition and computation.

The observed negative-performance cases also indicate that a single universal crossover point should not be assumed without additional controlled

Table 4. Post-arrival arithmetic workload under the stated complex-operation counting convention

Length N	Conventional post-arrival multiplications	Conventional post-arrival additions	TPE-FFT correction multiplications	TPE-FFT correction additions
8	7	14	4	8
16	15	30	8	16
64	63	126	32	64
128	127	254	64	128
256	255	510	128	256
512	511	1022	256	512

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

experiments. In a practical implementation, an adaptive policy could be considered in which TPE-FFT is activated when the expected benefit from pre-arrival execution exceeds the associated scheduling and correction overhead.

Table 5. Statistical summary of execution-time measurements for the 1,075 genomic sequences.

Metric	Result
Number of sequences	1,075
Positive improvement	1,053 (97.95%)
Negative improvement	22 (2.05%)
Mean improvement	86.46%
Median improvement	94.15%
Standard deviation	23.09 percentage points
IQR	8.32 percentage points
Total conventional DIT-FFT time	225.56469 s
Total TPE-FFT time	7.80065 s
Aggregate execution-time reduction	96.54%
Aggregate runtime ratio	28.92×
Paired t-test	t(1074)=8.31, $p=2.79\times10^{-16}$
Wilcoxon signed-rank test	$p=8.12\times10^{-177}$
95% CI of mean paired-time difference	0.15476–0.25038 s

C. Post-Arrival Computational Workload

The principal computational advantage of TPE-FFT is reducing the arithmetic workload remaining after the final input sample becomes available. Under the operation-counting convention adopted in this study, the conventional late-arrival computation requires approximately N−1 complex multiplications and 2(N−1) complex additions. In comparison, the TPE-FFT correction phase requires approximately N/2 complex multiplications and N complex additions. It should be noted that the "post-arrival" refers specifically to operations that cannot be completed until the final acquired sample is available, rather than the entire FFT computation. Table 4 demonstrates that the proposed execution schedule reduces the arithmetic workload remaining in the final-sample processing phase. For example, for N=512, the conventional schedule requires 511 complex multiplications and 1,022 complex additions in the considered post-arrival stage, whereas TPE-FFT requires 256 complex multiplications and 512 complex additions for the correction phase. These operation counts describe the post-arrival computational workload and should not be interpreted as equivalent

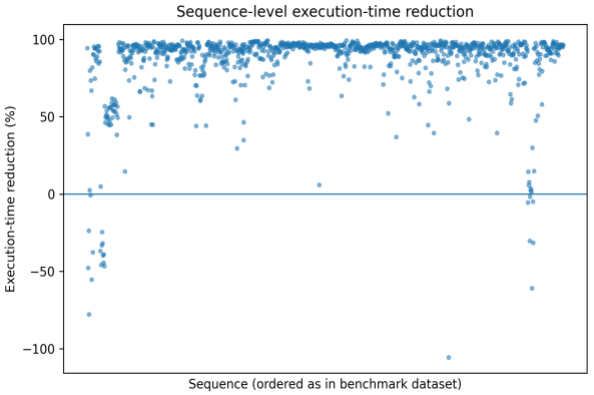


Fig. 4. Distribution of sequence-level execution-time reduction achieved by TPE-FFT relative to conventional DIT-FFT across 1,075 genomic sequences. Positive values indicate improved execution time, while negative values indicate performance degradation.

reductions in the total arithmetic complexity of the complete FFT. Similarly, they do not constitute direct measurements of FPGA/ASIC area, power, or energy consumption. Such hardware-level conclusions require synthesis, implementation, timing analysis, and power evaluation on a defined target platform.

D. Representative 33861 Long Sample Sequence

A representative long genomic sequence, NM_013231.6, containing 33,681 samples, was used to demonstrate the performance of TPE-FFT for a relatively large input sequence. The measured execution time decreases from 15.15 s for the conventional implementation to 0.51 s for TPE-FFT. The corresponding execution-time reduction is calculated as Eq. (25) [37]

$$R = \frac{15.15-0.51}{15.15} X\ 100 = 96.63\% \qquad (25)$$

Thus, the TPE-FFT implementation exhibits a 96.63% reduction in measured execution time, corresponding to a runtime ratio of approximately 29.71×. The proposed TPE-FFT is formulated for radix-2 processing; therefore, variable-length genomic sequences are mapped to an effective power-of-two transform length through zero-padding. For an input length L, the effective transform length is $M = 2^{\lceil log_2 L \rceil}$. Accordingly, the representative sequence NM_013231.6 containing 33,681 samples is zero-

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

padded to M=65,536 samples. The same effective transform length and zero-padded input are used for both the conventional DIT-FFT and TPE-FFT implementations. The reported execution-time comparison therefore represents a comparison at the same effective FFT length. The zero-padding does not introduce additional genomic information; it provides the power-of-two input required by the radix-2 implementation and results in a finer frequency-bin spacing.

E. Statistical Summary

The statistical characteristics of the 1,075 sequence-level measurements are summarized in Table 5. The results in Table 5 demonstrate a statistically significant difference between the paired execution-time measurements. The mean reduction of 86.46% is accompanied by a higher median reduction of 94.15%, indicating that the distribution is influenced by a relatively small number of lower-performing observations. The interquartile range of 8.32 percentage points indicates that the central 50% of observations are relatively concentrated around the higher-performance region. The statistical results should be interpreted together with the sequence-level distribution shown in Table 3 and Fig. 3. In particular, the presence of 22 negative-performance observations demonstrates that the proposed approach should not be regarded as uniformly faster for every input configuration.

whereas sliding DFT techniques maintain the transform of a moving data window, as shown in Table 6.

Sparse-update and sparse FFT approaches exploit sparsity in the input, update pattern, or frequency-domain representation. Time-distributed FFT architectures distribute FFT operations across multiple temporal processing intervals. TPE-FFT instead focuses specifically on identifying computations within the radix-2 DIT butterfly structure that can be executed before the final sample arrives. Execution times reported in previous studies are not treated as direct numerical baselines unless the corresponding algorithms are implemented and evaluated under identical hardware, software, transform length, numerical precision, input conditions, and timing boundaries. Accordingly, the related approaches are discussed here primarily from a methodological perspective, rather than through direct cross-study execution-time comparisons.

IV. Discussion

The experimental results demonstrate that the proposed TPE-FFT execution strategy can substantially reduce the measured execution time of the evaluated custom FFT implementation for the majority of genomic sequences. The most important interpretation of this result is that TPE-FFT provides a redistribution of computational workload, rather than an

Table 6. Methodological distinction between TPE-FFT and related Fourier-transform approaches.

Approach	Primary objective	Relation to final-sample dependency	Direct benchmark in this study
Incremental / online DFT [38]	Update spectral information as samples arrive	Different update formulation	No
Sliding DFT [39]	Maintain the transform of a moving window	Different window model	No
Sparse-update / sparse FFT [40]	Exploit sparsity in updates or spectrum	Not specifically focused on final-sample dependency	No
Time-distributed FFT [41]	Distribute FFT operations across time	Closest related scheduling concept	No
TPE-FFT(Proposed)	Move final-sample-independent work before arrival	Explicitly targeted	Yes; custom DIT-FFT baseline

F. Comparison with related FFT Approaches

The proposed TPE-FFT addresses a specific execution-scheduling problem associated with the delayed availability of the final input sample. This objective differs from several established approaches for reducing FFT computation time or supporting continuously arriving data. Incremental and online DFT methods update spectral information as new samples become available, asymptotic reduction in the complexity of the Fourier transform. The underlying FFT computation remains O(Nlog2N). The principal difference is the temporal location at which the computation is performed. Conventional processing leaves a portion of the butterfly computation unresolved until the final input sample becomes available, whereas TPE-FFT executes final-sample-independent operations in

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

advance and subsequently performs a correction operation after the final sample arrives. This distinction is important when interpreting the reported 96.54% aggregate execution-time reduction. The measured reduction reflects the execution behavior of the evaluated implementations under the defined timing boundary. It should not automatically be interpreted as an equivalent reduction in total arithmetic workload, energy consumption, FPGA resource utilization, or end-to-end acquisition-to-output latency. The sequence-level results nevertheless demonstrate a strong performance trend. More than 97% of the evaluated sequences exhibit positive execution-time reduction, and 85.30% achieve reductions greater than 80%. The high median improvement of 94.15% further indicates that substantial reductions occur across the central portion of the dataset. At the same time, the 22 negative-performance cases demonstrate that the benefit depends on input characteristics and implementation overhead. The sequence length of these 22 sequences varies from 8 to 24 nucleotides. The negative cases can be attributed conceptually to the relative contribution of fixed processing overheads and the amount of computation that can be advanced before final-sample arrival. For smaller computational workloads, operations such as buffer management, correction-vector generation, function invocation, and data movement can become significant relative to the useful computation shifted to the pre-arrival phase. Consequently, TPE-FFT is better interpreted as a dependency-aware scheduling strategy rather than a universally faster replacement for conventional FFT processing.

The representative 33,681-sample sequence further illustrates the potential of the approach for long genomic inputs. The measured execution-time reduction of 96.63% indicates a substantial difference between the two evaluated implementations. However, because the input length is not a power of two, the effective transform length and non-power-of-two processing strategy should be explicitly reported when presenting this result. Such information is necessary to ensure that execution-time comparisons are interpreted consistently. The comparison with related FFT approaches also highlights the specific scope of the proposed method. Incremental or online DFT methods update spectral coefficients as new samples arrive, whereas sliding DFT methods maintain the transform of a moving data window. Sparse FFT approaches reduce computation by exploiting sparsity in the input, update pattern, or frequency-domain representation. Time-distributed FFT approaches distribute FFT operations over successive temporal intervals, thereby reducing the instantaneous processing requirement. Hardware-oriented FFT architectures additionally target throughput, area, power, or operating frequency through pipelining, parallelism, approximate arithmetic, or specialized multipliers. In contrast, TPE-FFT specifically identifies the portion of a radix-2 DIT butterfly dependency graph associated with the final acquired sample and schedules independent operations before that sample becomes available. Thus, the primary metric addressed by TPE-FFT is the computational workload remaining after final-sample arrival rather than total arithmetic complexity or steady-state throughput. From an engineering perspective, TPE-FFT is most relevant to sequential signal-processing systems in which data acquisition and computation can proceed concurrently. In such systems, conventional block-based FFT processing may leave a substantial portion of the transform unresolved until the final sample arrives. TPE-FFT moves final-sample-independent operations into the preceding acquisition interval and reserves a smaller correction workload for the final-sample event. This principle may be useful in streaming genomic signal-processing pipelines, sequential biological signal analysis, and other data-acquisition systems in which rapid post-arrival processing is desirable. The genomic sequences provide a useful variable-length computational workload for evaluating the proposed scheduling strategy. However, the present results primarily establish computational performance, rather than biological or clinical effectiveness. The study therefore does not claim direct improvement in retinal disease diagnosis, pathogen tracking, point-of-care systems, or clinical decision-making.

The following limitations should be considered when interpreting the present results. First, the experimental evaluation was performed using MATLAB implementations; therefore, the measured execution times characterize the evaluated software implementations rather than a dedicated hardware architecture. Second, FPGA or ASIC synthesis and implementation were not performed, and consequently hardware resource utilization, operating frequency, power consumption, and energy efficiency have not been experimentally established. Third, the measured execution time includes implementation-dependent overhead associated with the two-phase schedule, buffer handling, correction-vector generation, and function-level processing. This overhead can become significant for relatively small computational workloads and is reflected in the 22 sequences that exhibited

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

negative execution-time reduction. Fourth, the present evaluation primarily measures computational execution time and does not constitute a complete acquisition-to-output latency measurement for a physical streaming genomic-data acquisition system. Finally, although the mathematical formulation establishes exact DFT reconstruction, systematic numerical-equivalence testing across different floating-point and fixed-point precision formats remains an important extension. These limitations motivate future FPGA/ASIC implementation, repeated timing evaluation, end-to-end streaming experiments, precision analysis, and adaptive selection of TPE-FFT according to workload size.

V. Conclusion

This study presented TPE-FFT, a dependency-aware two-phase execution strategy for reducing the computational workload remaining after final-sample arrival in radix-2 FFT processing of sequential genomic data. By exploiting DFT linearity, TPE-FFT performs final-sample-independent computations before the arrival of the final sample and subsequently incorporates its contribution through an exact correction operation. Evaluation using 1,075 genomic sequences showed positive measured execution-time reduction for 1,053 sequences (97.95%). The cumulative execution time decreased from 225.56469 s for the conventional DIT-FFT implementation to 7.80065 s for TPE-FFT, corresponding to a 96.54% aggregate reduction and a 28.92× runtime ratio. For the representative 33,681-sample sequence, the measured execution time decreased from 15.15 s to 0.51 s, corresponding to a 96.63% reduction. Importantly, TPE-FFT does not reduce the asymptotic computational complexity of the FFT; both approaches remain $O(N\log_2 N)$. The principal benefit arises from redistributing computation from the post-arrival interval to the period preceding final-sample arrival. Future work will focus on end-to-end streaming latency, numerical-equivalence analysis under different precision formats, adaptive activation of TPE-FFT, and FPGA/ASIC implementation for hardware-level evaluation.

Acknowledgment

The authors acknowledge the National Center for Biotechnology Information (NCBI) for providing public access to the GenBank genomic sequence database used in this research.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

The genomic sequence data used in this study were obtained from the publicly accessible NCBI GenBank database. The authors have provided the following repository link:
https://github.com/csk-hub/AFFT

Code Availability

The Code is available at https://github.com/csk-hub/AFFT with release Initial Manuscript Release – TPE-FFT v1.0.0 under the MIT open License.

Author Contribution

C. Saravanakumar contributed to conceptualization, methodology, formal analysis, software implementation, and validation. N. Usha Bhanu contributed to conceptualization, data curation, investigation, supervision, and original draft preparation. N. Subhashini contributed to formal analysis and visualization. Preethi Sambandam Raju contributed to manuscript review and editing. All authors read and approved the final version of the manuscript.

Ethical Approval

Ethical approval was not required because the study used publicly available genomic sequence data and did not involve direct human participation, human experimentation, or animal experiments.

Informed Consent to Participate

Not applicable. The study involved secondary computational analysis of publicly available genomic sequence data and did not involve recruitment or direct participation of human subjects.

Consent for Publication

Not applicable. The manuscript does not report identifiable personal information, participant photographs, or individual clinical records.

Competing Interests

The authors declare that they have no competing financial or nonfinancial interests that could have influenced the research, analysis, or presentation of the findings.

Declaration of Generative AI and AI-Assisted Technologies

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

The authors declare that no generative artificial intelligence (AI) or AI-assisted technologies were used in the preparation of this manuscript.

Research Originality and Submission Declaration

The authors confirm that this manuscript represents original research, has not been published previously, and is not currently under consideration for publication elsewhere. All authors have reviewed and approved the submitted version and agree to its submission to the Journal of Electronics, Electromedical Engineering, and Medical Informatics.

Reproducibility Statement

This study's MATLAB code is on GitHub (https://github.com/csk-hub/AFFT), using 1,075 NCBI GenBank sequences processed on Windows 10.

References

[1] P. Duhamel and H. Hollmann, "Split radix FFT algorithm," Electron. Lett., vol. 20, no. 1, pp. 14–16, Jan. 1984, doi: 10.1049/el:19840011.

[2] G. Bruun, "z-transform DFT filters and FFTs," IEEE Trans. Acoust., Speech, Signal Process., vol. 26, no. 1, pp. 56–63, Feb. 1978, doi: 10.1109/TASSP.1978.1163096.

[3] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," Proc. IEEE, vol. 56, no. 6, pp. 1107–1108, Jun. 1968, doi: 10.1109/PROC.1968.6458.

[4] L. H. Thomas, "Using a computer to solve problems in physics," in Applications of Digital Computers, 1963, pp. 44–45.

[5] J. B. Birdsong and N. I. Rummelt, "The hexagonal fast Fourier transform," in Proc. 2016 IEEE Int. Conf. Image Process. (ICIP), 2016, pp. 1809–1812, doi: 10.1109/ICIP.2016.7532670.

[6] L. I. Bluestein, "A linear filtering approach to the computation of the discrete Fourier transform," IEEE Trans. Audio Electroacoust., vol. 18, no. 4, pp. 451–455, Dec. 1970, doi: 10.1109/TAU.1970.1162133.

[7] S. Winograd, "On computing the discrete Fourier transform," Math. Comput., vol. 32, no. 141, pp. 175–199, 1978, doi: 10.1090/S0025-5718-1978-0456761-0.

[8] A. Becoulet and A. Verguet, "A depth-first iterative algorithm for the conjugate pair fast Fourier transform," IEEE Trans. Signal Process., vol. 69, pp. 1537–1547, 2021, doi: 10.1109/TSP.2021.3060279.

[9] S. Sanjeet, B. D. Sahoo, and K. K. Parhi, "Comparison of real-valued FFT architectures for low-throughput applications using FPGA," in Proc. IEEE 64th Int. Midwest Symp. Circuits Syst. (MWSCAS), 2021, pp. 112–115, doi: 10.1109/MWSCAS47672.2021.9531878.

[10] D. Krishnegowda and P. Nagaraju, "Design and synthesis of a 256-point radix-2 DIT FFT core with DesignWare library using fixed-point number representation," in Proc. IEEE Delhi Sect. Conf. (DELCON), 2022, doi: 10.1109/DELCON54057.2022.9753181.

[11] Y. Zhao, H. Lv, J. Li, and L. Zhu, "High performance and resource-efficient FFT processor based on CORDIC algorithm," EURASIP J. Adv. Signal Process., vol. 2022, no. 1, Art. no. 55, 2022, doi: 10.1186/s13634-022-00855-6.

[12] C. Saravanakumar and N. Usha Bhanu, "A mutated addition-subtraction unit to reduce the complexity of FFT," Appl. Nanosci., vol. 12, pp. 1–11, Jan. 2022, doi: 10.1007/s13204-021-02278-5.

[13] D. K. J. Rajanediran, C. G. Babu, K. Priyadharsini, and M. Ramkumar, "Hybrid radix-16 Booth encoding and rounding-based approximate Karatsuba multiplier for fast Fourier transform computation in biomedical signal processing application," Integration, the VLSI J., vol. 98, Art. no. 102215, Sep. 2024, doi: 10.1016/j.vlsi.2024.102215.

[14] F. Ai and V. Lomakin, "Fast Fourier Transform periodic interpolation method for superposition sums in a periodic unit cell," Comput. Phys. Commun., vol. 304, Art. no. 109291, Nov. 2024, doi: 10.1016/j.cpc.2024.109291.

[15] G. Huang and F. Zhang, "The fast computation of multi-angle discrete fractional Fourier transform," Signal Process., vol. 218, Art. no. 109365, May 2024, doi: 10.1016/ j.sigpro. 2024. 109365.

[16] M. R. Saad et al., "Fourier transform optimizer: A novel physics-inspired metaheuristic algorithm for optimization problems," Knowl.-Based Syst., vol. 340, Art. no. 115651, May 2026, doi: 10.1016/j.knosys.2026.115651.

[17] O. Leitersdorf, Y. Boneh, G. Gazit, R. Ronen, and S. Kvatinsky, "FourierPIM: High-throughput in-memory Fast Fourier Transform and polynomial multiplication," Memories - Mater., Devices, Circuits Syst., vol. 4, Art. no. 100034, 2023, doi: 10.1016/j.memori.2023.100034.

[18] T. Olas, R. Wyrzykowski, M. Olas, M. Palkowski, and M. Gruzewski, "Towards the efficient use of RISC-V architecture in scientific computation," J. Comput. Sci., vol. 96, Art. no. 102853, 2026, doi: 10.1016/j.jocs.2026.102853.

[19] B. Schmidt and A. Hildebrandt, "From GPUs to

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

AI and quantum: Three waves of acceleration in bioinformatics," Drug Discovery Today, vol. 29, no. 6, Art. no. 103986, Jun. 2024, doi: 10.1016/j.drudis.2024.103986.

[20] X. Liu, Z. Tan, P. Tang, P. Lu, and X. Lu, "Research on a novel gene sequence prediction and homomorphic encryption method based on Mamba-VMD," Comput. Biol. Chem., vol. 120, Art. no. 108780, 2026, doi: 10.1016/j.cbac.2026.108780.

[21] M. C. S. Tang, S. S. Teoh, H. Ibrahim, and Z. Embong, "Neovascularization detection and localization in fundus images using deep learning," Sensors, vol. 21, no. 16, Art. no. 5327, 2021, doi: 10.3390/s21165327.

[22] B. Goutam, M. F. Hashmi, Z. W. Geem, and N. D. Bokde, "A comprehensive review of deep learning strategies in retinal disease diagnosis using fundus images," IEEE Access, vol. 10, pp. 57796–57823, 2022, doi: 10.1109/ACCESS.2022.3178372.

[23] R. R. Mantovani, G. L. Menezes, and J. R. R. Dórea, "Predicting respiration rate in unrestrained dairy cows using image analysis and fast Fourier transform," JDS Commun., vol. 5, pp. 310–316, 2024, doi: 10.3168/jdsc.2023-0442.

[24] S. Benarjee and N. K. Vaegae, "Hadamard code based numerical mapping approach for the identification of protein coding regions in genomic sequences using digital filters," Comput. Biol. Med., vol. 195, Art. no. 110565, 2025, doi: 10.1016/j.compbiomed.2025.110565.

[25] N. K. Vaegae, "Walsh code based numerical mapping method for the identification of protein coding regions in eukaryotes," Biomed. Signal Process. Control, vol. 58, Art. no. 101859, Apr. 2020.

[26] M. Raman Kumar and V. Naveen Kumar, "A numerical representation method for a DNA sequence using gray code method," in Soft Computing for Problem Solving: SocProS 2018, vol. 2, 2019, pp. 645–654.

[27] N. A. Kumar et al., "Verification of periodicity of exons in DNA using digital signal processor (TMS320C6748)," Procedia Comput. Sci., vol. 259, pp. 1612–1618, 2025, doi: 10.1016/j.procs.2025.04.116.

[28] E. Ndupuechi and C. Beecks, "Fast Fourier transform gated activation function (FFTGate)," Inf. Fusion, vol. 129, Art. no. 103987, May 2026, doi: 10.1016/j.inffus.2025.103987.

[29] M. Henry, "An ultra-precise fast Fourier transform—Part 2," Measurement, vol. 270, Art. no. 120779, Apr. 2026, doi: 10.1016/j.measurement.2026.120779.

[30] M. Garrido, "A survey on pipelined FFT hardware architectures," J. Signal Process. Syst., vol. 94, pp. 1345–1364, 2022, doi: 10.1007/s11265-021-01655-1.

[31] S.-W. Chiu and K. K. Parhi, "Efficient decomposition of multistage composite length FFT for complex and real signals," IEEE Trans. Signal Process., vol. 74, pp. 892–903, 2026, doi: 10.1109/TSP.2026.3664785.

[32] R. Li, C. Liu, Y. Liang, J. Fu, Q. Zhang, and J. Xuan, "A fast algorithm for the symmetric discrete Fourier transform," in Proc. 2025 Int. Conf. Equipment Intell. Operation Maintenance (ICEIOM), Urumqi, China, 2025, pp. 401–406, doi: 10.1109/ICEIOM65271.2025.11239746.

[33] Y. Zhou, W. Cao, L. Liu, S. Agaian, and C. L. P. Chen, "Fast Fourier transform using matrix decomposition," Inf. Sci., vol. 291, pp. 172–183, 2015, doi: 10.1016/j.ins.2014.08.022.

[34] S.-C. Pei and K.-W. Chang, "Fast sparse DFT computation for arbitrary length by circular convolution," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2025, pp. 1–5, doi: 10.1109/ICASSP49660.2025.10890724.

[35] W. Shi, J. Wu, Y. Yuan, Z. Mo, C. Wu, and J. He, "An area–energy-efficient 64–2048 point FFT with approximate plane-fitting complex multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2025, doi: 10.1109/TVLSI.2025.3550470.

[36] National Center for Biotechnology Information (NCBI), "GenBank," [Online]. Available: https://www.ncbi.nlm.nih.gov. Accessed: Apr. 25, 2026.

[37] C. Saravanakumar and N. Usha Bhanu, "A constructive approach to numerical mapping scheme of nucleotides for preprocessing in machine learning," in Proc. 2022 12th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence), Noida, India, 2022, pp. 482–486, doi: 10.1109/Confluence52989.2022.9734186.

[38] B. Liu, B. Yan, and Q. Zou, "Optimal time-distributed fast Fourier transform: Application to online iterative learning control—Experimental high-speed nanopositioning example," Mechatronics, vol. 41, pp. 114–124, 2017, doi: 10.1016/j.mechatronics.2016.11.007.

[39] I. Domuta and T. P. Palade, "Sliding DFT and zero padding," in Proc. 2019 42nd Int. Conf. Telecommunications Signal Process. (TSP), Budapest, Hungary, 2019, pp. 154–157, doi: 10.1109/TSP.2019.8769076.

[40] T. Zhang et al., "FLASH: An efficient hardware accelerator leveraging approximate and sparse

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

FFT for homomorphic encryption," in Proc. 2025 Design, Automation Test Europe Conf. (DATE), Lyon, France, 2025, pp. 1–7, doi: 10.23919/DATE64628.2025.10992915.

[41] M. Bharathi, K. Mohanarangam, and Y. J. M. Shirur, "Efficient signal processing: Harnessing distributed arithmetic for high-speed FFT operations," in Proc. 2025 Int. Conf. Intell. Innov. Technol. Comput., Electr. Electron. (IITCEE), Bangalore, India, 2025, pp. 1–6, doi: 10.1109/IITCEE64140.2025.10915512.

Author Biography

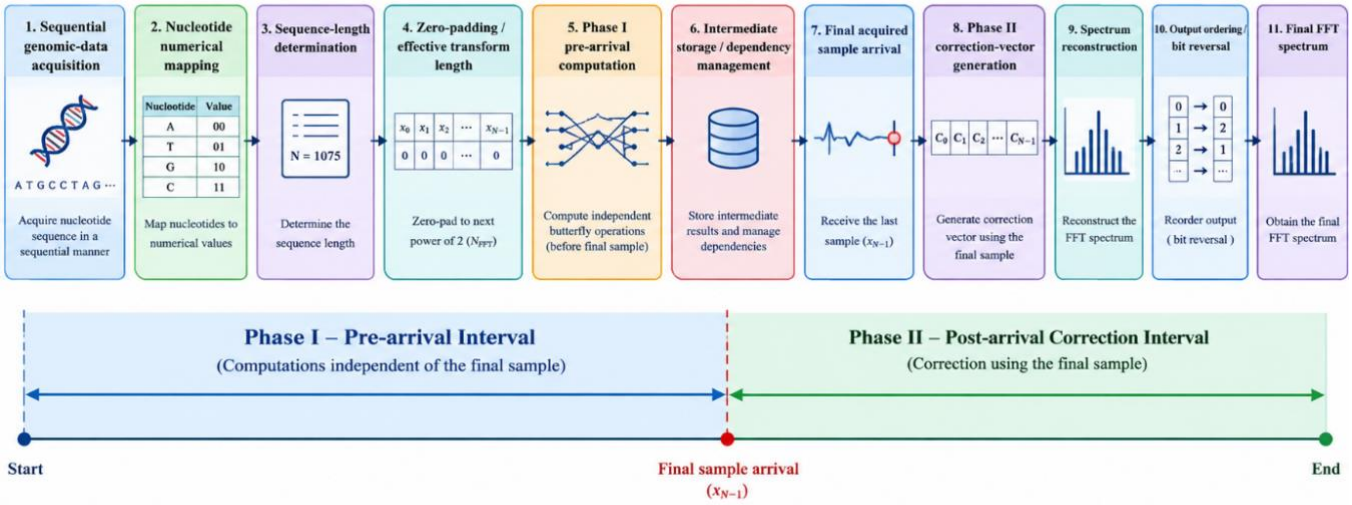
Saravanakumar C is an Associate Professor in the Department of Electronics and Communication Engineering at SRM Valliammai Engineering College. He holds a Ph.D. from Anna University, a post-graduation degree from Thanthai Periyar Government Institute of Technology, and an undergraduate degree from Pallavan Engineering College. With over 20 years of academic and research experience, his expertise spans digital signal processing, FPGA implementations, machine learning, and bioinformatics. An accomplished researcher, he has authored 30 Scopus-indexed and 13 Web of Science-indexed documents, alongside guiding a TNSCST-funded project. He holds senior professional memberships across several prestigious bodies, serving as Fellow of the Institution of Electronics and Telecommunication Engineers, Fellow of the Institution of Engineers, Member of the Institute of Electrical and Electronics Engineers, and Life Member of the Indian Society for Technical Education.

Usha Bhanu N is a Professor in the Department of Electronics and Communication Engineering at SRM Valliammai Engineering College, Kattankulathur, India. Born in India, she obtained her doctoral degree from Anna University, Chennai, where she is now a recognized research supervisor. Under her guidance, two candidates have successfully completed their Ph.D., and three others are currently pursuing their doctoral studies. Her primary research expertise includes VLSI design, digital signal processing, and image processing. An accomplished academician, she has managed a TNSCST-funded project and organized specialized AICTE-funded STTP and ISTE refresher programs. She has published 15 Scopus-indexed and 4 Web of Science-indexed documents, and holds life memberships in the Institution of Electronics and Telecommunication Engineers, the Indian Society for Technical Education, and the Computer Society of India.

Subhashini N is an Associate Professor in the Department of Electronics and Communication Engineering at SRM Valliammai Engineering College. She holds a Ph.D. specializing in wireless sensor networks and possesses approximately 25 years of extensive teaching experience. A prolific researcher, her publications encompass articles across 16 international journals, multiple book chapters, and a dedicated textbook on microwave engineering. Her innovative work is further reflected in six patents, including one successfully granted. Demonstrating strong capabilities in research acquisition and project management, she has successfully executed a DST-SERB project secured with 23 Lakhs in funding, alongside a SEED project funded at 3 Lakhs. Complementing her academic and research achievements, she maintains active professional engagements through her memberships in the Institution of Electronics and Telecommunication Engineers, the Indian Society for Technical Education, and the Computer Society of India.

Preethi Sambandam Raju is an Associate Professor in the Department of Electronics and Communication Engineering at SRM Valliammai Engineering College, Kattankulathur, Tamil Nadu. She graduated with her bachelor's degree from the Anand Institute of Higher Technology in 2012, earned her master's degree in Applied Electronics from St. Joseph's College of Engineering in 2014, and completed her Ph.D. from Anna University in 2021. With over 12 years of academic experience, her research interests encompass the Internet of Things, image and video processing, wireless sensor networks, machine learning, and artificial intelligence. Dr. Raju has published approximately 18 papers in reputed national and international journals and conferences. A recipient of the Young Researcher, Women's Achiever, and Best Emerging Faculty awards, she holds life memberships in the Institution of Electronics and Telecommunication Engineers, the Indian Society for Technical Education, and the Computer Society of India.

Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).



Manuscript received 16 July 2026; Revised 14 September 2026; Accepted 20 September 2026; Available online 30 October 2026
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v8i4.1933
Copyright © 2026 by the authors. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).