Impact Factor: 4.35

LOGAN JOURNAL OF COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE, AND ROBOTICS.

11(3) 2024 LJCSAIR

ISSN: 3067-266X

COMPARATIVE ANALYSIS OF FAST FOURIER TRANSFORM ALGORITHMS FOR SIGNAL PROCESSING

Samuel Chukwudi Eze

Department of Computer Science, Ignatius Ajuru University of Education, Port Harcourt, Nigeria

Abstract: The Fast Fourier Transform (FFT) algorithm is based on efficiently computing the Discrete Fourier Transform (DFT) in different signal processing systems. It is an umbrella of different algorithms, with each having the capacity to perform relatively better than others depending on the scenario. This research was able to review and compare ten (10) FFT algorithms identifying strengths, weakness, and application using the stochastic method of comparison. The research summarized the findings in tabular form, thereby making it easy for researchers'' select which FFT algorithm is most appropriate.

Keywords: Fast Fourier Transform, Discrete Fourier Transform, Signal Processing, FFT Algorithms, Stochastic Comparison

Introduction

Fast Fourier Transform (FFT) algorithm is an efficient algorithm for calculating the Discrete Fourier Transform (DFT). The earlier methods of Discrete Fourier Transform (DFT) constituted a long computational process, and Fast Fourier Transform (FFT) can be used to reduce or shorten the length of computation. FFT then is simply a computation of discrete Fourier transform in an algorithm format where the computational part will be reduced. This is achieved by creating Finite Impulse Response (FIR) filters mathematically. These filters then serve as the basis or the parameters for processing signals from audio, communication, consumer electronics and other signal processing activities.

In other to create the FIR filters, Fourier series algorithm in its simpler form is represented mathematically.

Using the algorithm, the discrete composite of a signal can be grouped into;

i. Symmetric sequence of odd length ii. Symmetric sequence of even length iii. Anti-Symmetric sequence of odd length iv. Anti-Symmetric sequence of even length

The uniqueness of the different techniques or variants of FFT algorithms lies on how each algorithm performs its calculation to arrive at its grouping of the discrete signal sequences, and what decision can be deduce from it in respect to application of it.

There are two main families of FFT algorithms: the Cooley-Tukey algorithm and the Prime Factor algorithm. These differ in the way they map the full FFT into smaller sub-transforms. The Cooley-Tukey algorithms uses two types of common routine: mixedradix (general-*N*) algorithms and radix-2 (power of 2) algorithms. The power

indices can be raised to 4, 8 and more depending on level of computation required to give an accurate result. Each type of algorithm can be further classified by additional characteristics, such as whether it operates in-place or uses additional scratch space, whether its output is in a sorted or scrambled order, and whether it uses decimation in-time or-frequency iterations.

Mixed-radix algorithms work by factorizing the data vector into shorter lengths. These can then be transformed by small-*N* FFTs. Typical programs include FFTs for small prime factors, such as 2, 3, 5, . . . which are highly optimized. The small-*N* FFT modules act as building blocks and can be multiplied together to make longer transforms. By combining a reasonable set of modules, it is possible to compute FFTs of many different lengths. If the small-*N* modules are supplemented by an $O(N^2)$ general-*N* module then an FFT of any length can be computed, in principle. Of course, any lengths which contain large prime factors would perform only as $O(N^2)$.

Radix-2 algorithms, or "power of two" algorithms, are simplified versions of the mixedradix algorithm. They are restricted to lengths which are a power of two. The basic radix-2 FFT module only involves addition and subtraction, so the algorithms are very simple. Radix-2 algorithms have been the subject of much research into optimizing the FFT. Many of the most efficient radix-2 routines are based on the "split-radix" algorithm. This is actually a hybrid which combines the best parts of both radix-2 and radix-4 ("power of 4") algorithms (Sorenson, Heideman and Burrus, 1986)

(Pierre, 1986)

The prime factor algorithm (PFA) is an alternative form of general-*N* algorithm based on a different way of recombining small-*N* FFT modules (Clive, 1983) (Clive, 1985). It has a very simple indexing scheme which makes it attractive. However it only works in the case where all factors are mutually prime. This requirement makes it more suitable as a specialized algorithm for given length The various algorithms that exist within the FFT performs well for different applications. Identifying the one that best suit the scenario is challenging especially for programmers and developers that may not be conversant or knowledgeable enough to make a selection. This work attempt to offer research solution to the constraint by reviewing, comparing and stating applications supported by the ten different variants of the FFT algorithms. The aim of this study was to classify the variants of Fast Fourier Transform algorithms. The objectives of this research were to; identify the variants of the Fast Fourier Transform (FFT) algorithms; classify the variants of the FFT algorithms accordingly to computing areas; recommend the best fit FFT algorithm for computing problem domain. This work reviewed ten (10) techniques of FFT, which are as stated as follows; Bluestein's FFT Algorithm Prime Factor FFT Algorithm Cooley-Tukey FFT Algorithm iv. Radix-2 FFT Algorithm Mixed Radix FFT Algorithm Hexagonal FFT Algorithm Split Radix FFTA Winograd FFT Algorithm ix. Rader FFT Algorithm x. Bruun's FFT Algorithm

Related Literature

An effective algorithm for commuting DFT is FFT. For the best answers, various FFT algorithms and structures have been put forth. By utilizing symmetry behavior versus real valued signals, Aravind et al. (2017) proposed the design of a 16 point DIF FFT based pipelined architecture for memory efficient hardware to decrease processing time and power consumption. A modified canonic signed digit multiplier (CSDM) is used in place of complex multipliers in order to reduce power consumption.

The parallel pipelined FFT architecture was introduced by Ayinala et al. in 2011. Real values and complex values are processed separately utilizing complex valued Fourier transform and real valued Fourier transform in the architecture, which is based on folding and resister reduction technique. Akkad, et al. (2018) presented the performance of real-time FFT algorithms in the High-Level Synthesis (HLS) environment, with the study's major

focus being on communication systems based on filter-based-multicarrier modulations (FBMC). This combination is evaluated using many combinations, including 4, 8, and 16-point radix-2.

The performance of a variable point FFT processor using a reconfigurable pipelined topology was presented by Bansal et al. in 2019. It was examined for sample points with variable N of 8, 16, 32, 64, 128, 256, and 512. The algorithm used in this work is the radix-2 common factor algorithm (CFA) with the Single Path Delay Commutator Feedback (SDF) architecture, which uses radix-2 butterfly structure to reduce the number of twiddle factors. The results are validated by modeling after the design and synthesis are completed using Xilinx ISE 14.1. The results of the simulation demonstrate that the device utilization employing is superior to the traditional FFTs. approach for producing circuit level equations that result in signal routing tables for radix-4 FFT was presented by Ganguly et al. in 2019. For simulation, BSIM4 with a 65 nm CMOS technology is employed with the SPICE modelling tool. A mismatch noise model is created to demonstrate the reduction in error at larger radix structures. For the investigation of nonideal effects caused by mismatch, Monte-Carlo simulation is used. A survey of three pipelined FFT designs, including radix-4 single-path delay commutator, radix-4 single path feedback (R4SDF), and radix-8 FFT with FPGA implementation, was provided by Hasan et al. (2018).

Comparing Radix-8 pipelined FFT to the other two Architectures, it has a higher power dissipation. Radix-4 single path feedback (R4SDF) outperformed the other two in terms of low area. The 32-point FFT processor was designed and built by Kumar et al. (2017) using VHDL in Xilinx ISE 14.2 and the outcomes were synthesized on Virtex -5 FPGA. The design used an 8-point FFT after a pipelined and parallel architecture. The 4G wireless communication system's OFDM technology can use the scalable FFT.

A pipelined radix-2 butterfly processor architecture that maximized both areas and frequency was presented by Kumar et al. (2015). This architecture consists of an FPGA-implemented 1024-point pipelined DIT processor with 64-bit fixed point and 32-bit real and imaginary components. This structure uses efficient addressing methods and is a dual RAM ping-pong burst I/O. On the Xilinx Virtex-6 xc6vlx550t-2ff1759, the estimated clock frequency is 385.804 MHz, and the estimated operating speed is 16.376ps.

A 256-point radix-4 algorithm-based FFT processor with good performance and minimal power consumption was presented by Mookherjee et al. in 2014. Fast switching is accomplished using multipath delay commutators with parallel processing. Throughput is increased eight times, resulting in 100% hardware usage. When compared to the standard radix-4 MDC structure, the structure uses 50% less power. The Xilinx FPGA Virtex 5 has been used to implement the design.

A distributed memory FFT architecture with excellent speed, a straightforward design, enhanced programmability, and fewer memory blocks was presented by Nash et al. (2018). The clock speed is accelerated by this arrangement. With the use of regulated word growth, the number of registers and LUTs is reduced. With an average increase of 94%, this decrease enables the circuit to accomplish its objectives, such as maximum throughput per logic cell, up to 181%. A radix-2 DIF-based FFT processor was presented by Nguyen et al. (2018). A parallel double-path delay Commutator (DDC) structure with continuous dual-input and dual output is utilized to boost throughput. By lowering the multipliers for the twiddle factor, the chip area can be decreased. It made use of both the newly-rolled CORDIC algorithm and canonical signed digit-based binary expression (CSDBE) for twiddle factor multiplication. This processor is implemented on Xilinx Virtex -7 FPGA. The results are shown for the improvement of speed, throughput and latency over the existing technology.

In terms of latency, the split-radix 4/8 FFT algorithm by Rauf et al. (2019) is equivalent to the radix 8 technique, however it performs better in terms of complexity. A Java automation tool is also used to build the VHDL description of an N-point split-radix 4/8 FFT processor, which can then be implemented on other platforms like

an FPGA. The design can be compared to fixed size designs in terms of resource utilization, such as speed and time, according to the results.

The Chip for FFT processor for OFDM transceiver system was created by Sood et al. (2013). The transmitter and receiver components of the OFDM employ the IFFT and FFT, respectively. Utilizing VHDL, the chip design is compiled in Xilinx 14.2. Additionally, the design is produced on the Virtex-5 FPGA using an estimation.

A high resolution FFT and IFFT processor was presented by Teymourzadeh et al. in 2014. High power consumption, device complexity, and computational efficiency are only a few of the limitations that high resolution complexity is subject to because the CPU is also performing floating point computations. By adjusting the variable length and particular hardware architecture, the efficiency can be raised. The authors tackled the problems by modifying the statistical method of quantization merging with signal to quantization noise ratio for DSP applications and obtaining the best outcome in terms of hardware consumption and data processing capacity.

Memory-based FFT with pipelined architecture was introduced by Wey et al. in 2007. Regarding space usage, cost savings, and delay, the design produced the best outcomes. For an 8192-point FFT, the MBFFT core required a chip area of 2.04 mm and a frequency support of 198 MHz. A single channel pipelined variable-length FFT architecture was presented by Wang et al. in 2019. For usage in radar imaging DSP applications, a high performance, high throughput FFT processor is presented. xc6vlx760ff1760-1 is utilized to synthesis the design on the Xilinx FPGA, and Modelsim and MATLAB are used for functional simulation.

A radix-2 pipelined 64-point FFT structure for local area networks with 67 clock cycles was presented by Wang et al. in 2010. Contrary to standard ROM structure, twiddle factors can be accessible directly in this situation. Also suggested is a fresh approach to address mapping. This processor uses two pipelines, one for reading and storing data and the other for the execution of complex tasks.

An approach for the variable length radix-4 DIF-FFT was presented by Yang et al. (2015). A variable word length at 16, 64, 256, and 1024 point FFT is available from the CPU. For high throughput, a single path delay feedback pipelined topology is chosen. Analyzed is the signal to quantization noise ratio for various FFT lengths. The Xilinx Virtex6, xc6vcx240t, FPGA is used for the design and synthesis. We discover that the proposed design offers better SQNR and short latency.

A variable length mixed radix multipath delay feedback MDF-FFT processor was presented by Yang et al. in 2017. In order to reduce the number of multipliers, a flexible length-based FFT architecture is implemented using a 4 parallel, radix-6, and mixed- 2/3/4 structure. To reduce space and power usage, CSD multiplier is chosen for constant factor multiplication for radix-6 and radix-3 butterfly structures. The design creates twiddle factors in parallel by using CORDIC, several adders, and multipliers. In comparison to earlier work, the architecture performs better in terms of area, power consumption, and latency. FFT transforms a time-domain signal into a frequency-domain signal that enables straightforward computations based on the frequency band required for continuous data that may be accessible in a variety of bands.

FFT is a crucial method used in spectrum analyzers and orthogonal frequency division multiplexing (OFDM). At the physical layer, such as WiMAX, 3GPP, LTE, and high-speed

LAN protocols, OFDM is employed in 4G and 5G wireless technology. For the 4G cellular technology to offer multicast broadband applications and video processing, significant data rates like 100 Mbps are required. FFT offers efficient matrix-vector multiplication, filtering, and large-integer multiplication operations. Cost, power, and frequency restrictions are vulnerabilities in wireless system design. The FFT calculation processing and analysis enhances the performance of the electronics system by lowering testing, cost, power, and time complexity. FFT architectures are the subject of ongoing research to reduce hardware complexity with varying

length and data width on several platforms, including platforms such as CPUs, GPUs and FPGA. For the implementation of electronic system prototypes, the FPGA is frequently utilized. Manufacturing companies can release their products before the final ASIC since the FPGA is changeable. In terms of slices, memory, delay, and throughput, various FFT architectures offer varying hardware complexity. The study can be directed at determining the suitability of a particular FFT design to calculate the best possible resource consumption on FBGA. **Computing Problem Domains**

The FFTs are efficient algorithms for calculating Discrete Fourier Transform (DFT). The general principle or concept of FFT algorithms is the use of a divide and conquer strategy to factorise the matrix **W** into smaller sub-matrices, typically reducing the operation count to:
$O(N\sum fi)$ *if* N can be factorised into integers such as $f_1, f_2, f_3 \ldots f_n$

FFT series uses $f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$ as its computational framework. The differences lies within the extent to which the divide-and-conquer rule is applied to arrive at the required value that is acceptable for meaningful and accurate deduction.
To apply it in a simpler form, the algorithm representation;

$$\chi[\mathrm{K}] \equiv \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

is used in creating the finite impulse response and grouped into the previously identified groupings.
For instance, when a signal is identified in a system, based on the wave length and modulation, points (rise and fall of the signal on the oscillation or wavering bar) are identified and grouped into even or odd. Thus creating 2 mathematical equations.
Further reiterating the divide and conquer technique, the FFT representation is based on the observation that multiple operations can be share when calculating the output frequencies of the FFT. This is done by decomposing the equation of the DFT which can be represented in a flow graph, binary tree, matrix factorisation and a triangular matrix representation.

Cooley-Tukey Fast Fourier Transform Algorithm

The Cooley-Tukey FFT algorithm uses the divide-and-conquer approach to auxiliary complex multiplications. It is suited for any composite length in a general form with $N=2^n$. It the base FFT algorithm in which every other FFT algorithm relies on. For instance, the coefficient of N raised to the power of 2, established the path for Radix-2 decimation in time (DIT) algorithm in which the input sequence is divided into decimated sequences having different phases within time. The divide and conquer multiplication is as follows in a simplified way;

$$N_2 \cdot N + N_2 \cdot N_1^2$$
$$= N_1 \cdot N_2(N_1 + N_2) < N_1^2 \cdot N_2^2$$
$$\text{if } N1, N2 > 2,$$

Showing clearly that the divide and conquer approach has reduced the number of multiplications needed to compute the DFT.
The mapping of the length of the transform is a composite of $N= N_1 * N_2$ ….
The natural decimation of the initial sequence will be;

$$I_{n_1} = \{n_2 N_1 + n_1\},$$
$$n_1 = 0, \ldots, N_1 - 1, \quad n_2 = 0, \ldots, N_2 - 1,$$

Which can also be given as;

$$X(z) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_2 N_1 + n_1} z^{-(n_2 N_1 + n_1)},$$

If the algorithm procedure is to be summarised in simple terms as the rearranging the calculations or exploiting the symmetries of the complex exponential weights into to obtain an efficiency in software implementation, and improving the computational efficiency when input series consist solely of real numbers.

The Cooley-Tukey FFT algorithm has been applied to vocoding, digital filtering, analysis of under-water sound recordings which is utilised by aquatic scientist when studying creatures under the sea, analysis of electroencephalographic data and other areas.

Bluestein's FFT Algorithm

Bluestein's FFTA algorithm can also be used in the computation of prime length DFTs in the $O(N \lg N)$ operations. It is similar to Rader's FFT, however, unlike Rader's FFT,

Bluestein's algorithm is not restricted to prime length and it can compute other kinds of transform

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{-kn}, \quad k = 0, 1, 2, \ldots, N-1,$$

where

$$W \triangleq \exp(j2\pi/N)$$

denotes a primitive N th root of unity, is multiplied and divided by $W^{(n^2+k^2)/2}$ to obtain

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W^{-kn} W^{\frac{1}{2}(n^2+k^2)} W^{-\frac{1}{2}(n^2+k^2)} \\ &= W^{-\frac{1}{2}k^2} \sum_{n=0}^{N-1} \left[x(n) W^{-\frac{1}{2}n^2} \right] W^{\frac{1}{2}(k-n)^2} \\ &= W^{-\frac{1}{2}k^2} (x_q * w_q)_k, \end{aligned}$$

$*$' Denotesconvolution (§), and the sequences x_q and w_q are defined by where `

$$\begin{aligned} x_q(n) &\triangleq x(n) W^{-\frac{1}{2}n^2}, \quad n = 0, 1, 2, \ldots, N-1 \\ w_q(n) &\triangleq W^{\frac{1}{2}n^2}, \quad n = -N+1, -N+2, \ldots, -1, 0, 1, 2, \ldots, N-1, \end{aligned}$$

n

where the ranges of given are those actually required by the convolution sum above.

Beyond the required minimum ranges for n1, the sequences may be extended by zeros, and as such, convolution may be implemented in cyclic for even N and negacyclic for odd N, using zero-padding

The larger cyclic convolution size $N_2 \geq 2N-1$ may be chosen a power of 2, which need not be larger than, $4N-3$

$N_2 - N + 1 : N_2 - 1$

Within this larger cyclic convolution, the negative –π indexes map to in the usual way.

In summarising the Bluestein's FFT algorithm, the basic feature is its complexity of $\boldsymbol{N\,lg\,N}$ for any positive integer DFT length N whatsoever even when N is prime.

It is also used to compute a contiguous subset of DFT frequency samples that is uniformly spaced set of samples along the unit cycle with $\boldsymbol{N\,lg\,N}$ complexity. It can similarly compute samples of the z transform along a sampled spiral of the form , where z is any complex number and k = k_0 + z^k----------k_0 + N-1 with a complexity of $\boldsymbol{N\,lg\,N}$

Although the Bluestein's FFT algorithm works well in computing complex samples, it is relatively slow compared to the Cooley and Tukey FFT algorithm.

Prime Factor FFT Algorithm

The Prime Factor FFT Algorithm works with mutual prime numbers basing its operation on the factorisation theorem that every integer **can** be uniquely factored into product of prime numbers P_i raised to an integer power $\boldsymbol{N}$ $m_i \geq 1$: (Clive, 1983)

$$N = \prod_{i-1}^{n_p} P_i^{m_i}$$

Although the mixed-radix Cooley-Tukey FFT can be used to implement a length of DFT $\boldsymbol{N}$ using DFTs of length P_i , for factors of $\boldsymbol{N}$ that are mutually prime such as and for $P_i^{m_i}$ $P_j^{m_j}$ $i \neq j$. The Prime Factor Algorithm (PFA) is also known as Good Thomas is more efficient in comparison to Cooley-Tukey based on Clive (1985) work. However, it is only applicable to mutually prime factors of $\boldsymbol{N}$, consequently, it is ideal to hybrid or combine it with other algorithms that works with any integer factor. PFA and Winograd are closely related with PFA seemingly faster, though it is arguable or contestable.

Radix-2 FFT Algorithm

In radix-2, when a power of 2, say is an integer, then the decimation in time decomposition can be performed, $N = 2^k \text{ where } K > 1$ utile ac DFT is length 2. A length 2 DFT requires no multiplies. The overall result is called a radix 2 FFT, and when the decimation is performed using frequency (Decimation in Frequency), a different radix 2 FFT is derived. $K = 1 \text{ times}$ (Duan, et al, 2011)

Based on Mookherjee, DeBrunner, and DeBrunner (2014) Radix-2 is a complex algorithm when put together $\boldsymbol{N}$ DFT from $N/2$ length-2 DFTs in a radix-2 FFT, the multiples required or needed are those used to combine two small DFTs to make a $\boldsymbol{N}$ DFT twice as long. Since there are approximately complex, multiplies needed for each stage of $\boldsymbol{N}$ the DIT decomposition, and only lg stages of DIT where lg denotes the

x $\mathcal{O}(N^2)$

N $\mathcal{O}(N \lg N)$

$\mathcal{C}(N)$ $\mathcal{C}$ m

$\mathcal{C}(N) \leq cN \lg N \text{ for all } N > m$

$N \log N \text{ or } \mathcal{O}(N \lg N)$

log-base-2 of, the total number of multiplies for length DIT is reduced from to $\mathcal{O}(N \lg N)$ $where$ $\mathcal{O}(\mathcal{X})$ means on order of, The precise complexity of means that given any implementation of a length- radix-2 FFT, there exist a constant and the such that the computational complexity satisfies

Complexity of the radix-2 FFT can be said to be (Teymourzadeh, Abigo, and Hoong. 2014)

Radix-4 FFT algorithm is a variant of radix-2. That means, it is same principle but the exponential is to the power of 4 (N^4). Meaning the input is 4, and the output is 4. Due to its high exponential power, it is said to be more efficient than the radix-2.

However, it is established that a combination of any of its variants (radix-2 or raidx-4) with other known FFT algorithms such as the prime factor FFT algorithm will yield better results.

Ganguly, Chakraborty, and Banorjee (2019) researched on improved variant of radix-4 is the VLSI design based on radix-4 decimation in frequency that is geared towards a reduction in the effect due to the mismatch of transistors in wireless communication such as OFDM that uses receivers and transmitters.

Mixed Radix FFT Algorithm

The Mixed Radix FFT algorithm is a combination of the indices of the existing radix. Mixed radix is generally used when the number of points of the FFT, N is not a power of the radix. For instance, N=128 is not a power of 4 and therefore, it cannot be implemented using radix4. However, it can be implemented with mixed radix-2 and 4.

Split Radix

The term Split Radix refers to a decomposition to time called Decimation in Time (DIT) that combines portions of one radix 2 and two radix 4 FFTs. The work of Sorenson, Heideman, and Burrus (1996) states that the basic and simple idea, is to use different radix for the even part of the transform (radix-2) and for the odd part radix-4. This is supported by the work of Pierre (1986). Theoretically, it is more efficient than a pure radix 2 algorithm because it minimises real arithmetic operations. It is adaptable for real and symmetric data while achieving the minimum known number of operations for FFT prime power of 2 length. On modern general-purpose processes however, computation time is often not minimised by minimising the arithmetic operation count.

Hexagonal FFT algorithm

Hexagonal FFT algorithm is a sampling technique and is most efficient sampling pattern for a 2-dimentional circularly band limited function. The process involves sampling lattice. It is also used in solving the discrete Poisson equation a rectangle using a regular hexagonal grid and results obtained for a model dirichlet problem. For a given grid size of usual 5-point, the hexagonal is more accurate and relatively fast but less efficient.

Winograd FFT Algorithm

Winograd FFT algorithm is designed to improve the multiplicative computation of FFT algorithms with highly structured data flow, dependent on the transform sizes such as N, Np^2, p^k, and odd prime. It is believed or accepted that algorithms have operational counts close to its implementation, and as such, the efficiency of an algorithm can be determined not only by its simplicity of implementation (application), but also its computational count in relation to suitability for parallel operation or computing (Tolimieri, Lu, and Johnson. 1990)

The algorithm starts with the multiplicative ring-structure of the indexing set using the prime factor algorithm and compute the results by factorisation and combining it with Rader. This process significantly reduces the number of multiplication operations and does not increase the number of additional operations. It has properties such multiplication reduction and at same time uses the properties of the FFT.

This algorithm is arithmetic efficient, programming simplicity, good structuring, parallelisation, and linear coverage as main advantages. (Silverman, 1977) The concept of Winograd FFT algorithm is defined as follows;

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk}$$

As the N-point DFT, where

$$k \in [0, N-1] \quad \text{and} \quad W_N \equiv e^{\frac{-j2\pi}{N}}$$

Rader FFT Algorithm

The Rader FFT algorithm computes the DFT of prime sizes by expressing the DFTas a cyclic convolution. The algorithm depends on the periodicity of the DFT kernel or base, and it directly transform prime order with a similar property. Rader algorithm proposes a replacement of all complex multiplications by either real or imaginary data. Thus, substantially reducing the number of multiplications required by the algorithm. It can be modified to gain a factor of two savings for the case of DFTs of real data by using a modified re-indexing permutation to obtain two half size cyclic convolutions of real data. (Engelberg, 2017)

The Rader FFT algorithm showed that by reorganising computations in a different way, you could also make the calculation of the DFT more efficient when the number of elements is prime.

Considering the DFT description in which $a^l = a^l \bmod p + n_l p$, we find that

$$\begin{aligned}
Y_m &= \sum_{k=0}^{p-1} \exp(-2\pi jkm/p) y_k \\
&= y_0 + \sum_{l=0}^{p-2} \exp(-2\pi j(a^l \bmod p)m/p) y_{a^l \bmod p} \\
&\overset{\exp(-2\pi j n_l pm/p)=1}{=} y_0 + \sum_{l=0}^{p-2} \exp(-2\pi j(a^l \bmod p + n_l p)m/p) y_{a^l \bmod p} \\
&= y_0 + \sum_{l=0}^{p-2} \exp(-2\pi j a^l m/N) y_{a^l \bmod p}.
\end{aligned}$$

it is noticeable that the change here is to the order in which the elements are summed. The following logic may apply if we consider the value of $y_{a^m \bmod p_i}$

$$Y_{a^m \bmod p} = y_0 + \sum_{l=0}^{p-2} \exp(-2\pi j a^l a^m / p) y_{a^l \bmod p}$$
$$= y_0 + \sum_{l=0}^{p-2} \exp(-2\pi j a^{l+m} / p) y_{a^{-(-l)} \bmod p}.$$

The Rader FFT algorithm can be used when working with a relatively simple set of algorithms to implement FFTs with the zero-padding option because simple implementation of the FFT might only implement the algorithm for powers of two. The computation cost implication is an increase in the number of additions and a greater sensitivity to round off noise

Bruun's FFT Algorithm

Bruun's FFT algorithm is based on an unusual recursive polynomial factorisation approach for powers of two and generalised to arbitrary even composite sizes. Its operations involve only real coefficients until the last computation stage. It also proposed an original scheme particularly suited for real data. This algorithm has not seen widespread usage due to the rate of efficiency achieved or attained by the Cooly-Tukey FFT algorithm.

Mittal, Khan, and Seinivas (2017) Bruun's FFT can be considered to be an ideal algorithm for software defined radio (SDR) by creating a single block of the algorithm that can be configured to compute FFT, DCT and DST due to its modularity nature and low computational complexity.

The issue with Bruun's FFT is that it has an inherent high noise ratio. Despite its seemingly setback, it is said to be a promising method of computation of different discrete transforms especially FFT. It utilizes fact that two can also be factored using the following equation

$$1+aZ^{2q}+Z^{4q} = \left(1+\sqrt{2-a}Z^{q}+Z^{2q}\right)\left(1-\sqrt{2-a}Z^{q}+Z^{2q}\right)$$

Bruun's algorithm just like split-radix algorithm, not only reduces the number of computations but also maintains the regularity of a structure of SDR. It allows computation of cos/sin transform of a input signal with small changes

Methodology

In other to effectively compare the algorithms, the stochastic method of ordinal and optimal techniques was used. The ordinal technique was used to identify the unique and similar features of the algorithms considering each within the ambience of the FFT. The review considered the computation structure and the representational pattern of the various algorithms. The optimal technique was used to review efficiency in relation to computational difficulties such as cost and time as it relates to the hardware architecture.

Discussion

In reviewing the FFT suites, several variants were identified of which ten were selected using random selection method, and five classification criteria was identified. These are;

i. Computation rate
   a. time taking to complete computation
   b. speed – that is how fast or quickly the computation is done or completed

ii. resource requirement – this involves processing power and other computer resources that will be dedicated or reserved for the algorithm while it is been computed.

iii. Bit rate of the algorithm – deals with the composition of the algorithm which forms the mathematical framework or model. Which also shows the complexity of the FFT variant. This usually directly or indirectly influence the computation rate and the resources required

iv. Architectural design – variants that specifies the system firmware in which it can be executed successfully or yield maximum throughput.

Cooley-Tukey variant uses the coefficient in computing and support decimation in time. It is able to subdivide the mathematical model into subsection for easy calculation and brings the results of each subsection together to form a wholistic answer. It makes computation easy by deciphering the complexity. However, it has high-rate computation rate but controlled resource usage.

Bluestein is high-rate complexity variant. However, it is versatile as it can be used for prime length but not restricted to prime length only. The complexity of this variant is represented by convolution and sequences. Due to the high complexity, it is relatively slow, yet the complexity produces a verifiable accurate result, and it is appropriate for heavy computation scenario with high power resources.

The prime factor variant is characterised by low complexity by implementing the N integer and works efficiently with prime numbers. Due to its low complexity, it can easily be use along with other FFT variants, and it is also fast.

The radix series variants are closely related and work same way. The main difference amongst them is the ratio indices which defines the N power. The higher the N power indices, the higher the complexity ratio. Also, the mixed radix computation rate largely depends on the second variant with which the radix is been combined. If it is combined with Bluestein for instance, there is the assumption that it will also take in the benefits and setbacks of it.

Hexagonal variant of FFT is used for pattern sampling. Suitable for data signal analysis. Complexity is low, while Winograd is arithmetically efficient with a linear structure. Both has low complexity

Rader variant though said to the simple in application, the mathematically model used in representing it is complex. Furthermore, it accepts zero padding. Which means that the number of zero it is padded with will determine the computation ratio.

Bruun variant is based on recursive polynomial approach and as such it has high complexity and relatively slow in comparison to the Radix variant series. It also has high noise ratio. It is versatile.

5. Summary of Reviewed FFT Algorithm

Table 1 summarizes the identified variants and their main characteristics, highlighting the mode of computation, operation and application to specific areas.

Table 1

Classification of the FFT Algorithms and Application

FFT Algorithm	Operation and Computation	Application
Cooley & Tukey	Uses the divide and conquer technique. It is the bases for other FFT algorithms. It rearranges the calculations	Used for digital filtering

Bluestein	Used in computation of prime length but not restricted to prime length only. It can compute other kinds of transform. Relatively slow compared to Cooley-Tukey	Wireless signals and underwater signals.
Prime Factor	Works with mutual prime numbers. It is based on factorisation.	Engine or mechanical signal detection
Radix-2	Based calculation to the power of 2, 4, 8 etc as the case maybe. The efficieny of it is depended on the power raised to (coefficient). It can use DIT or DIF	Numbering system and pattern recognition.
Mixed Radix	Combination of indices for greater efficiency. For instance combining radix-2 and radix-4	Used when a hybrid or combination of any of the radix need to be used to improve efficiency.
Split Radix	Combines portions of radix-2 and radix-4. Alternative is to combine any of the radix FFT algorithm, depending on intended results.	Used in applications that requires high efficiency with minimal error rate such a medical signalling.
Hexagonal	Based on 2 dimensional circularly sampling and lattice grid size	Representation of harmonic detection
Winograd	Designed to improve multiplication computation. It reduces the number of multiplications intended for computing DFT	Speech and acoustics
Rader	Periodicity, reorganisation of computation in different way	Elementary number theory
Bruun	Recursive polynomial and factorisation	Software defined radio (SDR)

6. Conclusion

There is an abundant literature that highlights the capabilities of the FFT as a signal processing. However, not much literature is found in comparing the variants based on classification. Each identified variant has its strength of application and computation. Cooley and Tukey, the most common of the variant due to its divide and conquer technology is said to be simple and effective for digital filtering. Bluestein is mostly used for wireless signals and other high-level computation, but its computing power is high. The Radix variant series serve different purposes and can be applied in mechanical signal detection, numbering system, hybrid, numbering theory, medical signaling etc.

The existence of several variants offers programmers, developers, and analyst the opportunity to make a choice and select the most appropriate depending on the expected outcome, complexity of computation or field of application.

References

Akkad, G., Mansour, A., Hassan, B., Le Roy, F., & Najem, M. (2018, November). FFT radix-2 and radix-4 FPGA acceleration techniques using HLS and HDL for digital communication. In 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET) (pp. 1–5).

Aravind Kumar, M., & Manjunatha Chari, K. (2017). An efficient pipelined architecture for real-valued fast Fourier transform. International Journal of Electronics, 104(4), 692–708.

Ayinala, M., Brown, M., & Parhi, K. K. (2011). Pipelined parallel FFT architectures via folding transformation. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 20(6), 1068–1081.

Ganguly, A., Chakraborty, A., & Banerjee, A. (2019). A novel VLSI design of radix-4 DFT in current mode. International Journal of Electronics, 106(12), 1845–1863.

Hassan, S. L. M., Sulaiman, N., & Halim, I. S. A. (2018). Low power pipelined FFT processor architecture on FPGA. In 2018 9th IEEE Control and System Graduate Research Colloquium (ICSGRC) (pp. 31–34).

Kumar, A., Kumar, A., Devrari, A., & Singh, S. (2017). Design and FPGA implementation of 32-point FFT processor. In Proceeding of International Conference on Intelligent Communication, Control and Devices, Advances in Intelligent Systems and Computing (pp. 285–292).

Mookherjee, S., DeBrunner, L., & DeBrunner, V. (2014, November). A high throughput and low power radix-4 FFT architecture. In 2014 48th Asilomar Conference on Signals, Systems and Computers (pp. 1266–1270).

Nash, J. G. (2018). Distributed-memory-based FFT architecture and FPGA implementations. Electronics, 7(7), 116.

Nguyen, N. H., Khan, S. A., Kim, C. H., & Kim, J. M. (2018). A high performance, resource-efficient, reconfigurable parallel-pipelined FFT processor for FPGA platforms. Microprocessors and Microsystems, 60, 96–106.

Rauf, A., Pasha, M. A., & Masud, S. (2019). Towards design and automation of a scalable split-radix FFT processor for high throughput applications. Microprocessors and Microsystems, 65, 148–157.

Sood, S., Singh, A., & Kumar, A. (2013). VHDL design of OFDM transceiver chip using variable FFT. Journal of Selected Areas in Microelectronics (JSAM), Singaporean Journal of Scientific Research (SJSR), 5, 47–58.

Mittal, S., Khan, M., & Srinivas, S. (n.d.). On the suitability of Bruun's FFT algorithm for software defined radio.

Engelberg, S. (2017). Elementary number theory and Rader's FFT. Society for Industrial & Applied Mathematics, 59(1), 671–678.

Garrido, M. (2017). A new representation of FFT algorithms using triangular matrices. IEEE Transactions on Circuit & System.

Duan, B., Wang, W., Li, X., Zhang, C., Zhang, P., & Sun, N. (2011). Floating-point mixed radix FFT core generation for FPGA and comparison with GPU and CPU. IEEE.

Ganguly, A., Chakraborty, A., & Banerjee, A. (2019). A novel VLSI design of radix-4 DFT in current mode. International Journal of Electronics.

Tolimieri, R., Lu, C., & Johnson, R. (1990). Modified Winograd FFT algorithm and its variants for transform size N = pk and their implementation. Advances in Applied Mathematics, 10, 228–251.

Silverman, H. (1977). An introduction to programming the Winograd Fourier transform algorithm (WFTA). IEEE Transactions on Acoustics Speech and Signal Processing, 25(2), 152–165.

Duhamel, P., & Vetterli, M. (1990). Fast Fourier transform: A tutorial review & a state of the art. Signal Processing, 19, 259–299.

Su, T., Yang, M., Jin, T., & Flesch, R. (2018). Power harmonic and interharmonic detection method in renewable power based on Nuttall double-window all-phase FFT algorithm. IET Journals.

Sorenson, H. V., Heideman, M. T., & Burrus, C. S. (1986). On computing the split-radix FFT. IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-34(1), 152–156.

Pierre, D. (1986). Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data. IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-34(2), 285–295.

Clive, T. (1983). A note on prime factor FFT algorithms. Journal of Computational Physics, 58, 198–204.

Clive, T. (1985). Implementation of a self-sorting in-place prime factor FFT algorithm. Journal of Computational Physics, 58, 283–299.

Teymourzadeh, R., Abigo, M. J., & Hoong, M. V. (2014). Static quantized radix-2 fast Fourier transform (FFT)/inverse FFT processor for constraints analysis. International Journal of Electronics, 101(2), 231–240.

Wey, C. L., Lin, S. Y., & Tang, W. C. (2007, May). Efficient memory-based FFT processors for OFDM applications. In 2007 IEEE International Conference on Electro/Information Technology (pp. 345–350). IEEE.

Wang, J., Xie, Y., & Yang, C. (2019). Single channel pipelined variable-length FFT processor design. The Journal of Engineering, 2019(21), 7709–7712.

Wang, B., Zhang, Q., Ao, T., & Huang, M. (2010, January). Design of pipelined FFT processor based on FPGA. In 2010 Second International Conference on Computer Modeling and Simulation (Vol. 4, pp. 432–435). IEEE.

Wang, S. S., & Li, C. S. (2008). An area-efficient design of variable-length fast Fourier transform processor. Journal of Signal Processing Systems, 51(3), 245–256.

Yang, C., Xie, Y. Z., Chen, L., Chen, H., & Deng, Y. (2015). Design of a configurable fixed-point FFT processor (pp. 1–4).

Yang, C., Wei, C., Xie, Y., Chen, H., & Ma, C. (2017). Area-efficient mixed-radix variable-length FFT processor. IEICE Electronics Express, 14(10), 20170232–20170232.