International Journal of Precious Engineering Research and Applications (IJPERA)
www.ijpera.com ISSN: 2456-2734,
Volume 3, Issue 1 (Jan- May) 2018), PP. 01-15

Development of Fast Performance Evaluation Algorithm to Optimize the Efficiency of the Cooley-Turkey Fast Fourier Transform Algorithm for Dip

Amannah and H.C. Inyiama
Constance Izuchukwu Department Of Computer Science University Of Nigeria, Nsukka,Enugu State.
Department Of Computer Science University Of Nigeria, Nsukka, Enugu State.
Corresponding Author: Amannah

ABSTRACT: This research was designed to develop a performance evaluation algorithm to optimize the Cooley-Tukey FFT algorithm necessary for the processing of digital signals. The fast numerical algorithm developed in this study is abbreviated with improved C-TNADSP. The methodology adopted in this work was iterative and incremental development design. The major technologies used in this work are the numerical algorithms and the c^{++} programming technologies and the wave concept technology. The c^{++} served as a signal processing language simulator (SPLS). In the Improved Cooley-Tuley FFT algorithm, the DSP input is encountered first. It is subjected to some numerical processing which included testing for efficiency on the C++ platform. This test platform provided the basis for comparison leading to the desired Fast Numerical Algorithms for Digital Signal Processing.The improved C-TNADSPwas well investigated and compared across warehouse inputs sufficient enough to generate the required output sequence. The computing speed of the default algorithm was tested on the C^{++}platform. The execution time of the Cooley-Tukey was 3.44 seconds. Similarly, the execution time of the resultant improved algorithm was 1.74 seconds. On comparing the speed of the default algorithm with that of the new algorithm, we observed that there is (1.74) seconds speed improvement on the Cooley-Tukey algorithm. In line with these outcomes, the new algorithm is referred to as improved C-TNADSP. This result therefore shows that a version of algorithm with computing speed that is faster than the 0(nlogn) computing speed of the fast Fourier algorithms exist.The algorithms were tested on input block of width 1000 units, and above, and can be implemented on input size of 100 000, and 1000 000 000 without the challenge of storage overflow. The input samples tested in this work was the discretized pulse wave form with undulating shape out of which the binary equivalents were extracted. Other forms of signals may also be tested in this fast algorithm provided they are interpreted in the digital wave type. In order to optimized the advantage of the developed algorithms, the frequency index, K should be as defined in this study, that is $K = \pi e^{\theta}$ *iff* $0 < \theta \leq 8$

Date of Submission: 18-04-2018 Date of acceptance: 21-05-2018

I. INTRODUCTION

Background to the Study

Signals play an important role in our daily life. A signal is the variable parameter that contains information and by which information is transmitted in an electronic system or circuit Dictionary of Science (2005). The majority of the signals found in science are analog in nature. In analog signals, both dependent variable and independent variables are continuous. Such signals may be processed directly by analog systems (i.e., analog filters) for the purpose of changing their characteristics or extracting some desired information. Digital filters are used for two general purposes: (1) separation of signals that have been combined, and (2) restoration of signals that have been distorted in some way.Analog filters can be used for these same tasks; however, digital filters can achieve far superior results.

The most popular digital filters are described and compared in this work. There are only two ways that are common for information to be represented in naturally occurring signals. We will call this information represented in the time domain, and information represented in the frequency domain. Analog signals can also be processed digitally using Digital Signal Processing techniques (DSPTs). To process analog signals digitally, an interface between the analog signal and a digital processor is required. This interface is known as analog-to-digital converter (ADC). The output of the analog-to-digital converter is a digital signal. This digital signal is appropriate for digital processor. The digital signal processor may be a large programmable digital computer or a small microprocessor. In electronics, computer science and mathematics, a digital filter is a system that performs numerical operations on a sampled, discrete-time signal to reduce or enhance certain aspects of that signal.

Digital Signal Processing (DSP) is an area of Science and Engineering which has developed very rapidly over the past few decades. As a matter of fact, the techniques and applications of Digital Signal Processing (DSP) are as old as Newton and Gauss and also as new as today's digital computers and Integrated Circuits (ICs). This rapid development of Digital Signal Processing (DSP) has been a result of the significant advances in digital computer technology and IC fabrication techniques. Signal processing is a method of extracting information from the signal which in turn, depends upon the type of signal and the nature of information it carries. Thus, signal processing is concerned with representing signals in mathematical terms and extracting the information by carrying out algorithmic operations on the signal. Mathematically, a signal can be represented in terms of basic functions in the domain of the original independent variable or it can be represented in terms of basic functions in a transformed domain. Similarly, the information contained in the signal can also be extracted either in the original domain or in the transformed domain.

Digital signal processing techniques originated in the seventeenth century when finite difference methods, numerical integration methods, and numerical interpolation methods were developed to solve physical problems involving continuous variables and functions. There has been a tremendous growth since then. Today, Digital Signal Processing (DSP) techniques are applied in almost every field. The main reasons for such wide applications are due to the numerous advantages in Digital Signal Processing (DSP) techniques. As a matter of fact, digital circuits do not depend upon precise values of the digital signals for their operation. Also, digital circuits are less sensitive to changes in component values. They are also less sensitive to the variations in temperature, ageing and other external parameters. In a digital processor, the signals and systems coefficients are represented as binary words. This enables us to choose any accuracy by increasing or decreasing the number of bits in the binary words. The storage of digital data is very easy. Signals can be stored on various storage media such as magnetic tapes, disks and optical disks without any loss. On the other hand, the stored analog signals deteriorate rapidly as time progresses and hence cannot be recovered in their original form. Also, for processing very low frequency signals like seismic signals, analog circuits require inductors and capacitors of a very large size whereas digital processing is more suited for such type of applications.

Parallel signal processing tasks Digital Signal Processors to overtake general purpose processors from 2 to 3 orders in speed. This is because of architectural differences. Typical DSP application fields are audio signal processing, video signal processing and telecommunications devices. Digital signal processing requires a large amount of real-time calculations. The most common operation in digital signal processing is the sum of products calculation. Among such operations are well known convolution and Discrete Fourier Transform. To increase the speed, digital signal processors usually have many specialized arithmetic units, which can operate simultaneously. A signal is a function of independent variables such as time, distance, position, temperature, and pressure. This research is therefore designed to provide a compatible mechanism necessary for achieving the transition to digital processing using Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP). The major technologies used in this work are the numerical algorithms and the c^{++} programming technologies and the wave concept technology. Numerical algorithms are used as filters to manipulate or process digital signals so that their operation times can be determined and compared accordingly. A collection of algorithms called fast Fourier transform algorithms was filtered with digital signal input data. Their operation time was known to be O(Nlog N) seconds. Our Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP) was designed by decomposing, re-indexing, and simplification of the Cooley-Tukeyfast Fourier transform algorithm. The Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP)has a diminution in time of operation.

The c^{++} technology is used to implement the proposed Faster Cooley-Tukey Numerical Algorithm for Digital Signal Processing (FC-TNADSP). The c^{++} here acted as a signal processing language simulator (SPLS). The SPLS simulated the designed Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP) using sampled input data available in the warehouse. The wave concept technology is used to represent discrete data samples expressed in binary format. The undulating shape of the wave indicates the binary values it contains. The lower bound of the wave represents zero while the upper bound represents one. When the binary values are collected together, they can be further converted into numerical or decimal values at which point, they can be used in testing the algorithms, the existing and the proposed. The expected output of the proposed algorithm is also represented in decimal and not in binary this time.

Statement of the Problem

Our contemporary global society is driving towards making all actions and activities digitalized. In Nigeria for instance the Communication and the Broadcasting Institution is determine to go all digitalized in the year 2015. The speed and scope of transmitting from analog to digital remain issues that need scientific resolution. In view of the foregoing and for effective transition from analog to digital transmission, an efficient computing algorithmic platform is a requirement. This research is therefore designed to develop an efficient numerical algorithm necessary for achieving the speed of processing digital signals in digital computers. The fast numerical algorithm to be developed will be called Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP)

- **Aim Objectives of the Study**

The aim of the study is a fast Cooley-Tukey numerical algorithm for digital signal processing. In order to attain this aim, the following objectives were considered;

a) To investigate the Cooley-Tukey fast numerical algorithm for Digital Signal Processing
b) To design a faster Cooley-Tukey numerical algorithm for Digital Signal Processing
c) To determine the computing speed improvement of the faster Cooley-Tukey numerical algorithm for Digital Signal processing
d) To apply warehouse input technology to test and compare the speed of the original Cooley-Tukeyalgorithm with the faster Cooley-Tukey numerical algorithm

- **Scope of the Study**

There are basically two types of digital signal processing (DSP) algorithms, namely Filtering Algorithms and Signal Analysis Algorithms. These algorithms can be implemented in any of the following forms; hardware, firmware, and software. In the hardware approach, the algorithm is implemented using digital circuitry, such as a shift register to provide the delaying operation; a special-purpose VLSI chip can be designed and fabricated to implement a specific filtering algorithm. In the firmware approach, the algorithm is implemented on read-only-memory (ROM) chip, additional control circuitry, and storage registers, are usually needed in the final firmware realization. These two approaches are customized and restricted to predefined hardware platforms that are predisposed to hardware incompatibility.

This study is restricted to the software approach of DSP-algorithm implementation on a minicomputer or a personal computer. A detailed discussion of hardware, firmware, and DSP chip implementation is beyond the scope of this study. The acronymic description of this research is Faster Cooley-Tukey Numerical Algorithm for Digital Processing (FC-TNADSP).

- **Significance of the Study**

When implemented, the algorithm designed in this work will be of significant assistance to the broadcasting industry in support of their bid to transit to an inclusive digital processing. Furthermore, the Digital Signal Processing algorithms of this research will provide computing framework for simulating signals associated with speech, image, communication, and so on.

II. RELATED LITERATURE

2.1The Application of Complex Digital Processing in Communications.

[1] Examined the application of complex digital processing in communications. They observed that the concept of DSP is coloured with the principles of mathematics. They opined that complex DSP has no obvious connection with our everyday experience. Their reason is occasioned by the fact that many DSP problems are explained mainly by means of real numbers mathematics. Their study agreed that some DSP problems are based on mathematics, such as Fast Fourier Transform (FFT), z-transform, representation of periodical signals and linear systems, and so on. Their research shows that the imaginary part of complex transformations is usually ignored or regarded as zero due to the inability to provide a readily comprehensible physical explanation. The result of their investigation shows that two DSPs exit namely complex DSP and real DSP. Their position explains that the principle of complex DSP is Complex math and that of real DSP is real math. Their approach is not universally applicable and can only be used with problems and applications which conform to the requirements of complex math techniques. Making a complex number entirely mathematically equivalent to a substantial physical problem is the real essence of complex DSP.

It is clear to see that the world has gone digital and almost being completely controlled by numbers. The concept of theoretical DSP which presupposes that DSP has no obvious connection with our everyday experience has become a mere tale. DSP can now be applied to virtually every aspect of our human endeavour. What were considered as complex DSPs occasioned by complex math can now be processed using simulateable numerical algorithmic processes such as the difference equation and the Fast Fourier Transform algorithms. In line with this the need of investigating the feasibility of fast numerical algorithms for DSP applications is both necessary and timely. This study is therefore needful and timely considering the fact that some countries are yet to integrate into the digital communication even as Nigeria is preparing to do so in the year to come.

2.2 The Interpolation in Digital Signal Processing and Numerical Analysis

[2] Investigated the Interpolation in Digital Signal Processing and Numerical Analysis. He reviewed some of the methods being used in interpolation such as Lagrange, Hermite, Shannon, and Piecewise polynomials interpolators, but there are lots of other methods that he did not mention. [2] Concluded that it is impossible to give a general comparison of them and say which one is the best. It depends on the case we are dealing with he added to choose a method he further added; "I don't think there will be any guarantee that the chosen method be the best possible one. Some people have compared some of these methods in their papers". For instance, [3] says "unlike small-kernel convolution methods, which have poor accuracy anywhere near the

Nyquist limit, the accuracy of the FFT method is maintained to high frequencies, very close to Nyquist". From [2], it is obvious that [2] did not effectively determine nor recommend an efficient algorithm for DSP and numerical analysis. Furthermore, the work did not also delve into detailed numerical principles compatible enough with DSP operations. [2] Dealt passively on FFT. [2] Did not also reconcile the place of DSP in relation to numerical algorithms. In view of the foregoing, we can therefore see the vacuum created by [2]; the non-determination of the place of efficient numerical algorithms for processing digital signals. Our study is poised to fill this gap. Hence our study is consequential, timely, and pivotal.

2.3 Efficient Digital Filters

In [4] investigation of Efficient Digital Filters, filters can be sharpened to meet the need of predefined DSP operations. His work shows that filters (FIR) can be sharpened either by reducing passband ripple or increasing stopband attenuation. His work proved that an existing system can still be used to perform the function of an improved new filter by simply sharpening the existing filter. This way, an old filter becomes new not by creation but by improvement. Simply stated, filter sharpening is a technique for creating a new filter from an old one. From this line knowledge it is obvious that DSP activities can be analyzed with filters and the filters can be sharpened in different ways.

[4]'s work centered on software filter. He did not however introduce math-based into his work. This cannot be exhaustively concluded considering the fact that DSP is mainly the analysis and manipulation of mathematical principles in relation to signals. In such case the math platform becomes the filter which could be interlink with software filters. To enhance the effectiveness of DSP operation numerical algorithms could best be considered especially if efficiency is the focus. This is an area that is compactable with contemporary communications systems. A research work addressing this gap is vital and timely. My research is aimed at investigating efficient numerical algorithms for convenient DSP operations. Its DSP filters will be math-based and could be sharpened to address the contemporary issues of DSP activity in signal driven society.

2.4Algorithms and Tools for Automatic Generation of DSP Hardware Structures

[5]'s research concentrated on hardware-based DSP. Its results are basically hardware structures. Some kinds of algorithms, namely the scheduling algorithms were employed but obviously not complex numerical algorithms. Furthermore, the basis for testing these algorithms is equally not specified in his work. And finally, the mode of analysis of the operations of DSP is not also delineated. In view of these articulated gaps in [5]'s work, an extended research work capable of addressing some of the concerns will be in order. This is why a work on efficient algorithms for DSP is required. My research is therefore aimed at addressing the need of efficient numerical algorithms for DSP.

2.5 SPL: A Language and Compiler for DSP Algorithms.

The formulas are represented in a language that the authors called SPL, an acronym for Signal Processing Language. The SPL work was conducted by [6]. This was about the design and implementation of a compiler that translates formulas representing signal processing transform into efficient C or FORTRAN programs. Their results showed that SPIRAL, which can be used to implement many classes of algorithms, produces program that perform as well as had systems like FFTW. SPL is a descendant of the TPL (Tensor Product Language), [7] that was developed for the automatic generation of FFT algorithms. [8, 9] differ in their findings. They hold that signal processing is not necessarily better suited for an approach like the one discussed above. Their approach is similar to that used by FFTW. The works of [10, 11] concentrated on Fast Fourier Transform (FFT) and FFTW. The SPL is a tool used to implement DSP in C or FORTRAN programs. However, the work stresses the method of translation and compilation of signals.

III. MATERIAL AND METHOD

The Fast Fourier Transform (FFT) is surely the most widely used signal processing algorithm. It is the basic building block for a large percentage of algorithms in current usage. An FFT is a way to compute the same result more quickly. This work investigated the various species of FFT taking into cognizance their methodologies, results, speed, and storage of their output sequence. The essence is to develop a fast algorithm for computing digital signals.The methodology adopted in this work was iterative and incremental development design. This is also known as iterative and incremental build model for software development. The basic idea behind this method is to develop a system (algorithm) through repeated cycles (iterates) and in smaller portions at a time (incremental), allowing the software developer (researcher) to take advantage of what was learned during development of earlier parts or versions of the system. Incremental development means that different parts of a software project are continuously integrated into the whole, instead of a monolithic approach where all the different parts are assembled in one or a few milestones of the project. So as a first result, iterative development doesn't need to be incremental and vice versa, but these methods are good fit.

Considering the fact that the Fast Fourier Transform algorithm already exist, and also that our work is aimed at developing a faster one, we can see that our work has a relationship with available system. For this reason our methodology is step down to iterative reconstruction methodology. This method has four basic

components beginning with requirements and climaxing with implementation and test. Fig.1 below illustrates these components of reconstructive iterative methodology.

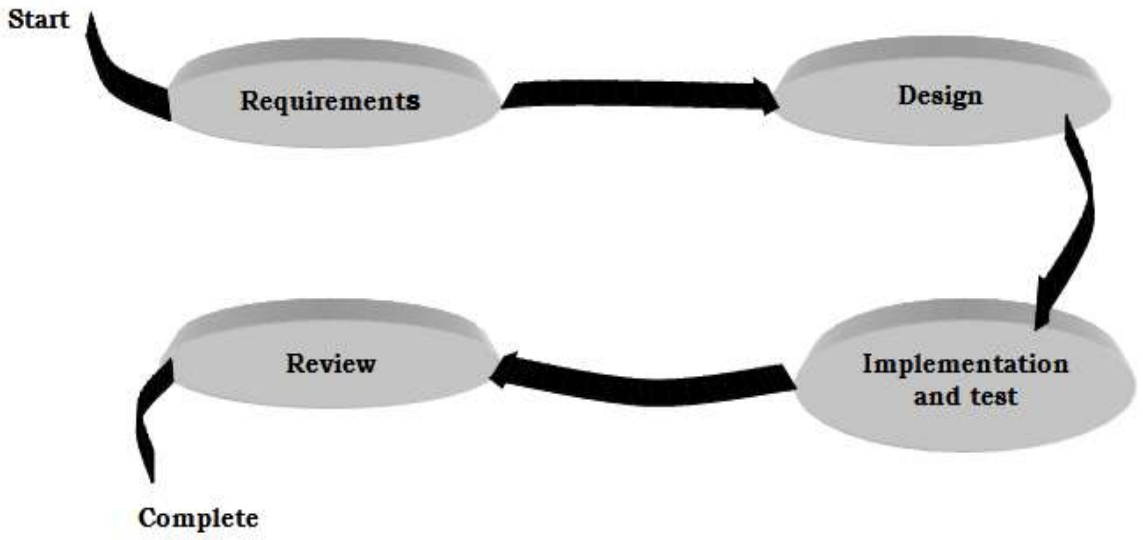


Fig.1: Reconstructive Iterative Development Model

3.1 An overview of the Existing System

The existing system is generally referred to as Fast Fourier Transforms (FFTs). An FFT is any method devised to compute the same results in
O(NlogN) operations. All know FFT algorithms require O(NLogN) operations. There are species of FFTs. In this work, we investigated three FFTs, namely;

- The prime-factor FFT algorithm by Good 9]
- the Bruun FFT algorithm by Bruun [10], and
- the Cooley-Tukey FFT algorithm by(14)

Each of the three FFTs investigated is traced and transformed from the Discrete Fourier Transform (DFT). The DFT is of two phases; the forward and inverse phases. Each of the four-FFTs is described below beginning with the DFT.

Discrete Fourier Transform (DFT)

DFT Formula

Let xo,. . ., X_{N-1} be complex numbers. The DFT is defined by the formula:

$$Xk = \sum_{n=0}^{N-1} xn \; e^{-i\,2\pi k \frac{n}{N}}, \qquad k= 0\ldots N-1 \qquad \ldots (1)$$

Evaluating this definition directly requires O(N^2) operations.

Inverse Discrete Fourier Transform (IDFT)

The inverse transform of DFT:

$$\text{DFT: F } F[n] = \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}nk}$$

The required inverse DFT becomes

$$F[k] = \frac{1}{N}\sum_{n=0}^{N-1} F[n] e^{+j\frac{2\pi}{N}nk} \qquad \ldots (2)$$

From the inverse transform formula, the contribution to $f[k]$ of $F[n]$ and $F[N-n]$ is

$$fn[k] = \frac{1}{N}\left[F[n] e^{j\frac{2\pi}{N}nk} + F[N-n\} e^{j\frac{2\pi}{N}(N-n)k} \right]$$

For all $f[k]$ real, $F[N-n] = \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}(N-n)k}$

But $e^{-j\frac{2\pi}{N}(N-n)k} = \underbrace{e^{-j}}_{1}\underbrace{2\pi k}_{for\ all\ K}\ e + j\frac{2\pi n}{N}k = \quad e + j\frac{2\pi}{N}nk$

i.e. $F[N-n] = F^{*}(n)$ (i.e. the complex conjugate)

Comparing equations (1) and (2) we infer that the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a $\frac{1}{N}$ factor. Any fast Fourier Transform (FFT) algorithm can easily be adapted from it.

3.2 Fast Fourier Transform (FFT) Algorithms

An FFT is a way to compute the same result more quickly. An FFT can compute the same DFT in only O(NlogN) operations. The difference in speed can be enormous especially for long data sets where N may be in the thousands or millions. In practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to N/log(N). This huge improvement made the calculation of DFT practical. FFTs are of great importance to a wide variety of applications from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

Species of FFTs

There are two main families of FFT algorithm; the Cooley-Tukey algorithm (C-TA), and the Prime Factor algorithm (PFA). Other sub-species of the FFT algorithms include Bruun's FFT algorithm and the Cooley-Tukey's FFT algorithm.

a. The Bruun's FFT Algorithm

Bruun's algorithm is a fast Fourier transform (FFT) algorithm based on an unusual recursive polynomial factorization approach, proposed for powers of two by [12] and generalized to arbitrary even composite sizes. Nevertheless, Bruun's algorithm illustrates algorithmic framework that can express both itself and the Cookey-Tukey algorithm, and thus provides an interesting perspective on FFTs that permits mixtures of two algorithms and other generalizations.

The Brunn's algorithm can be expressed from the DFT as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}^2 nk},\ K = 0,\ldots,\ N-1 \qquad \ldots (3)$$

For convenience, N roots of unity is denoted by

$$WN^{n}(n = 0, \ \ldots \ N-1)$$

$$W^{n}N = e^{-\frac{2\pi}{N}^2 n}$$

The polynomial x (z) with coefficients x_n is defined as

$$x(z) = \sum_{n=0}^{N-1} x_n z^n \qquad \ldots (4)$$

The DFT can then be understood as a reduction of this polynomial; that is X_k is given by

$$X_k = (w^k N) = x(z) \bmod (z - W^k N) \qquad \ldots (5)$$

Where **mod** denotes the polynomial remainder operation.

The key to fast algorithms like the Bruun's comes from the fact that one can perform this set of N remainder operations in recursive stages. If each level of the factorization splits every polynomial into an O(1) (constant-bounded) number of smaller polynomials each with an O(1) number of non-coefficients, then themodulo operations for that level take O (N) time; since there will be a logarithmic number of levels, the overall complexity is O(N log N).

b. The Prime Factor Algorithm (PFA)

The prime-factor algorithm (PFA), also called the Good-Thomas algorithm is a Fast Fourier Transform (FFT) algorithm that re-expresses the discrete Fourier transform (DFT) of a size N = N1N2 as a two–dimensional N1 x N2, DFT, but only for the case where N1 and N2 are relatively prime. These smaller transforms of size N1 and N2 can then be evaluated by applying PFA recursively or by using some other FFT algorithm.

The PFA algorithm can be expressed from the DFT as:

$$Xk = \sum_{n=0}^{N-1} x_n e - \frac{2\pi^2}{N} nk, \qquad K = 0,\ldots N-1$$

The PFA involves a re-indexing of the input and output arrays, which when substituted into the DFT formula transforms it into two nested DFTs (a two-dimensional DFT). By the prime factorization theorem, every integer N can be factored into a product of prime numbers P_i raised to an integer power mi ≥ 1

$$N = \coprod_{i=1}^{np} pi^{mi} \qquad \ldots (6)$$

Eq(8) is called the prime factor algorithm (PFA)

$$K_1 N_2^{-1} N_2 + K_2 N_1^{-1} N_1 = \sum_{n1=0}^{N_1-1} \left(\sum_{n_{2=}0}^{N_2-1} x_{n_1} N_2 + n_2 N_1 e^{-\frac{2\pi i}{N_2}} n_2 K_2 \right) . e^{-\frac{2\pi i}{N_1}} n_1 k_1 \qquad \ldots (7)$$

The inner and outer sums are simply DFTs of size N_2 and N_1, respectively.

We have used the fact that $N_1^{-1} N_1$ is unity when evaluated modulo N_2 in the inner sum's exponent and vice-versa for the outer sum's exponent.

Sine Cooley-Tukey FFT algorithm is a combination of DFT of Even-indexed of X_n with DFT of odd-indexed part of X

c. Cooley-Tukey FFT algorithm

The Cooley-Tukey algorithm was named by [14].Sine Cooley-Tukey FFT algorithm is a combination of DFT of Even-indexed of X_n with DFT of odd-indexed part of X_m, we can expect a slightly different result.

$$X_K = \sum_{m=0}^{N/2-1} X_2 me - \frac{2\pi}{N}^{(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e \frac{-2\pi i}{N} (2m+1)k \qquad \ldots (8)$$

3.3 The Proposed System

The proposed system is abbreviated as FC-TNADSP, which means Fast Algorithm for Digital Signal process. The FC-TNADSP is a numerical-based algorithm for computing discrete sequence digital signals. The proposed system works with digital signal inputs. These inputs are collected form warehouse. The essence is to test the FC-TNADSP with a view to determining its computing speed in comparison with other FFT algorithms especially the FFTs investigated in this work.

The efficiency of the FFT algorithms is already determined. Evaluating the DFT's sums directly involves N^2 complex multiplications and N(N-1) complex additions of which O(N) operations can be saved by eliminating trivial operations such as multiplications by

Constraints of the Existing System

Analysis of the four FFTs showed that each of them resulted from the DFT. Computing the DFT directly from the definition is often too slow to be practical. The DFT is considered naïve because computing the DFT of N points takes $0(N^2)$ arithmetical operations while an FFT can compute the same DFT in only 0(NlogN) operations. The difference in speed can be enormous especially for long data sets where N may be in the thousands or millions. The difference shows that the FFTs are improvements upon the DFT. In practice the computation time can be reduced by several orders of magnitude in such case, and the improvement is roughly proportional to N/log (N). There are however defined constraints of the four FFT Algorithms. Table1 below illustrates some of the constraints of the FFT algorithms.

Table 1: Constraints of the FFT Algorithms-Adapted from [51]

FFT Algorithms	Cost	Complexity	Operating frequency	Operation
Bruun	Moderate	Less	Good	Computes DFT of real co-efficient
Prime factor	Low	Moderate	Better	Re-expresses the DFT but only for the case where N_1 and N_2 are relatively prime
Cooley-Tukey	Low	Less	Good	Computes N-PT DFT with $N=2^t$

Table 1 above shows the strength and weakness of each of the FFT algorithms. However, this work is aimed at developing a faster FFT algorithm with cost, complexity, and operating frequency advantage. The known speed so far for FFT is 0(NlogN). An FFT algorithm with less than O (NlogN) operation can therefore be considered faster. This is the purpose and intent of this work.

3.4 Architecture of the Proposed System

The architecture of the proposed system as shown in fig. 2 below describes all the steps and stages necessary for the development of the proposed fast numerical algorithm for digital signal processing. The

architecture clearly illustrates the process and procedures of the iterative development model adopted in this research. The architecture begins with the DSP data input (requirement step in the iterative model), followed by the determination of the present and proposedalgorithms (the design step of the iterative model), followed by the comparison of both algorithms (the implementation and test step of the iterative model) and climaxed with the output and decision state (the review step of the iterative model). The output and decision stage eventually leads to the desired and expected result of the research, which is the fast numerical algorithm for digital signal processing abbreviated FC-TNADSP.

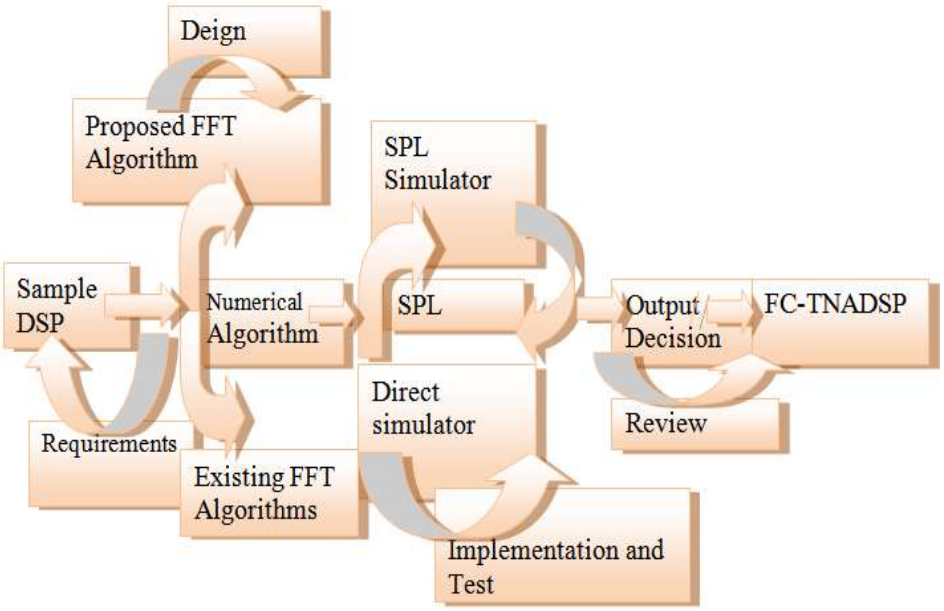


Fig.2: Architecture of the Proposed System

3.5 Design of the Proposed System

The proposed algorithm results from further decomposition, re-indexing and simplification of the Cooley-TukeyFast Fourier Transform algorithm. The procedure is outlined below:

The Cooley-Tukey FFT Algorithm

The Cooley-Tukey is the most common FFT algorithm. This is a divide and conquer algorithm that recursively breaks down a DFT of any composite size $N = N_1N_2$ into many smaller DFT_s of sizes N_1 and N_2, along with O(N) multiplications by complex roots of unity traditionally called twiddle factor.

Radix-2 decimation-in-time (DIT) FFT is the simplest and most common form of the cooley-Tukey algorithm. Radix-2 DIT divides a DFT of size into two interleaved DFTs (hence the name "radix-2") of size N/2 with each recursive state. The DFT has a forward and inverse form which is defined as:

$$Xx = \sum_{n=0}^{N-1} xn \,.e^{-i2\overline{\wedge}k\, n/N} \quad \text{(Forward DFT)} \qquad \ldots (9)$$

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k^{\,ei\,2\pi kn/N} \quad \text{(Inverse DFT)} \qquad \ldots (10)$$

Eq (10) can simply be re-expressed as:

$$X_k = \sum_{n=0}^{N-1} xne^{-\frac{2\pi}{N}nk} \qquad \ldots (11)$$

Where **K** is an integer ranging from0 to N-1 Radix –2 DIT first computes the DFTs of the even indexed inputs ($x_{2m} = x_0, x_2 \ldots x_{N-2}$) and of the odd-indexed inputs ($x_{2m+1} = x_1, x_3, \ldots x_{N-1}$) and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime to 0 (NLogN).

The above Radix–2 DIT algorithm shows that the DFT can be rearranged as a function of$x_n$with two parts; a sum over the even-numbered indices n = 2m and a sum over the odd-numbered indices n = 2m + 1.
The sum over the even-numbered indices combined with the sum over the odd-numbered indices yields:

$$Xk = \sum_{m=0}^{N/2-1} x_{2m}\, e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N2/-1} x_{2m+1}\, e^{-\frac{2\pi i}{N}(2m+1)k} \qquad \ldots (12)$$

A common multiplier in Eq (12) is $e - \frac{2\pi i}{N} k$

If the DFT of the Even-indexed inputs x_{2m} is denoted by E_k and the DFT of the odd–indexed inputs x_{2m+1} *is* denotedbyO_k, and $E_k + O_k$ results to:

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e - \frac{2\pi i}{N/2} mk}_{DFT\ of\ Even-indexed\ of\ xm} + e - \frac{2\pi i}{N} K \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e - \frac{2\pi i}{N/2} mk}_{DFT\ of\ Odd-indexed\ part\ of\ xm} \quad \ldots (13)$$

$$= E_k + e - \frac{2\pi i}{N} k \quad O_k$$

$$X_K = \sum_{m=0}^{N/2-1} X_{2me} - \frac{2\pi}{N}^{(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e \frac{-2\pi i}{N} (2m+1)k \quad \ldots (14)$$

Substituting the unity factor $\left(e^{2\overline{\wedge} i} = 1\right)$ in eqn (29) we have;

$$X_K = \sum_{m=0}^{N/2-1} X_{2m} \left(- \frac{1}{N} 2mk \right) + \sum_{m=0}^{N/2-1} x_{2m+1} \left(\frac{-1}{N} (2m+1) K \right) \quad \ldots (15)$$

On simplifying we have:

$$X_K = \sum_{m=0}^{N/2-1} - \frac{1}{N} K \quad \left[X_{2m} 2m + X_{2m+1} 2_{m+1} \right] \quad \ldots (16)$$

$$X_K = N^{-1} K \sum_{m=0}^{N/2-1} \left(X_{2m} \right) 2m + \left(X_{2m+1} \right) 2m + 1 \quad \ldots\ldots (17)$$

$$X_K = N^{-1} K \sum_{m=0}^{N/2-1} \left(X_{2m} \right)(2_m) + \left(X_{2m+1} \right)(2m+1)$$

Because of the periodicity of the DFT,

$$E_k + \frac{N}{2} = E_k \text{ and} \quad \ldots (18a)$$

$$O_k + \frac{N}{2} = O_k \quad \ldots (18b)$$

The outcomes of the DFT periodicity, Eq (17a) and (17b) when substituted into Eq(9) yields:

$$X_K = \begin{cases} E_K + e^{-} \frac{2\pi i}{N} k \quad O_K \quad for \quad 0 \leq k < N/2 \\ E_K - N/2 + e^{-} \frac{2\pi i}{N} k \quad Ok - N/2 \quad for \quad N/2 \leq k < N \end{cases} \quad \ldots (18c)$$

Since the twiddle factor $e - 2\pi ik / N$ obeys the relation:

$$e - 2\pi i(k + N/2) = e - \frac{2\pi i}{N} k - \pi i$$

$$= e^{-\pi i} e^{-} \frac{2\pi i}{N} k$$

$$= -e^{-} \frac{2\pi i}{N} k \quad \ldots (19)$$

This situation cuts the number of twiddle factor calculation to half, i.e. $= 0 \leq k < \frac{N}{2}$, this condition results to:

$$X_k = E_K + e^{-} \frac{2\pi i}{N} k \quad O_K \quad \ldots (19a)$$

$$X_k + \frac{N}{2} = E^K - e^{-} \frac{2\pi ik}{N} \quad O_K + e - \frac{2\pi i}{N} k \; O_k \quad \text{--- (20)}$$

Eqn (17) is a resultant equation from the forward and inverse DFTs. This equation has four products, one addition, and one exponentiation. Eqn (11) in its original source (eqn 8) had eleven products one addition

and two divisions. But upon re-indexing and subsequent simplifications, the parameter operators were reduced to four products, one addition, and one exponentiation. This drastic reduction in the number of operators accounts for the desired speed of the new algorithm.

Eqs (19) and (20), express the DFT of length N recursively in terms of two DFTs of size N/2, is the core of the radix-2 DIT Fast Fourier Transform. The algorithm gains its speed by re-using the results of intermediate computations to compute multiple DFT outputs. The final outputs are obtained by a+/- combination of E_k and $O_k Exp(-2\pi ik/N)$, which is simply a size -2 DFT (sometime called a butterfly).

3.6 Implementation Architecture

The various components of the software modules and sub-modules of the proposed system are clearly described in fig. 3 below.

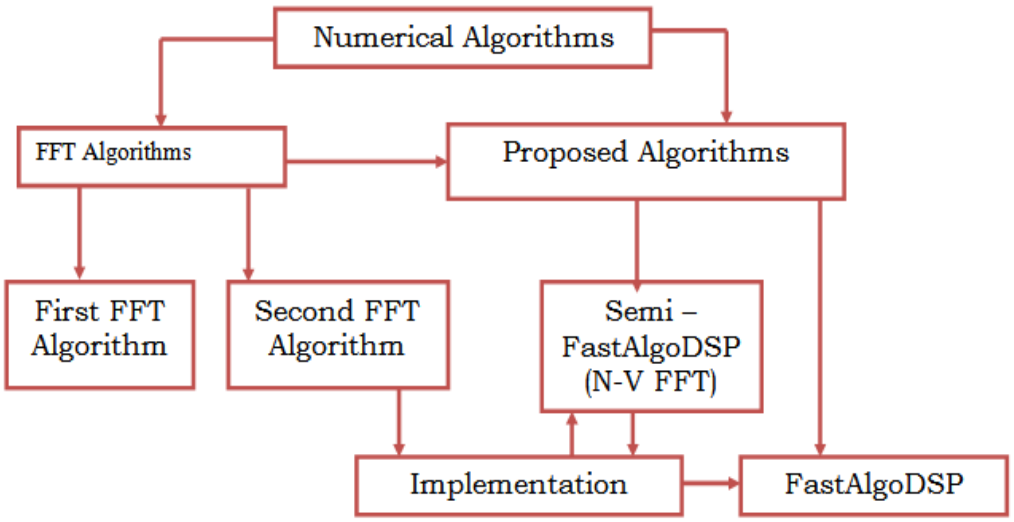


Fig. 3: Implementation Architecture of the Proposed System

In fig. 3 above, the main modules of the system include FFT algorithms and the proposed algorithm. Each of the main modules consists of sub-modules. The sub-modules of the FFT algorithms are first FFT algorithm and the second FFT algorithm. The sub-modules of the proposed algorithm are semi fast AlgoDSP and FC-TNADSP. The sub-modules are implemented (tested) leading to the determination of the semi-fastAlgo and the subsequent FC-TNADSP, the expected result of the system.

Table2: Performance Evaluation Metric for FASTAIGODSP

S/N	Performance metrics	Normal range	Abnormal range	Value of developed software	Rating of FC-TNADSP
1.	Fault Density (Fd)	0.0001 – 1.49	1.5 – 5.9	0.0019980	*HP
2.	Defect density (DD)	0.01 – 0.49	0.5 – 3.9	0.02	*HP
3.	Modular test coverage	40 -100	10 – 39%	48	*HP
4.	Requirement traceability (TM)	41 – 100	10 – 40	90%	*HP
5.	Software maturity index (SMI)	0.5 – 1.0	0.1 – 3.9	0.5263	*HP
6.	Cyclomtic complexity (C)	0.1 – 4.49	4.5 - ∞	4	*HP

Adapted from [12]*HP High Performance

To verify the correctness of the FC-TNADSP implementation, rigorous guarantees can be **obtained in less than (1.7sec) time by a** simple procedure checking the linearity, impulse response, and time-shift properties of the transform on random inputs [13].Table1 above indicates that the software developed in this study has high metric measurements. This shows that the FC-TNADSP software is reliable.

IV. DISCUSSION OF RESULTS

The result of this study shows that we can have faster numerical algorithms other than the FFT algorithms for the processing of digital signals. The faster algorithms resulted from the re-indexing and modification of the Cooley-Tukey FFT algorithm. The authors by this result therefore succeeded in developing an algorithm that is faster than the FFT algorithms. The speed advantage of the developed algorithms is significant enough that there is a necessary shift of efficiency rate from O (nlogn) to a speed that is faster by 1.70 seconds. In favour of this study, the faster algorithmis called FC-TNADSP.FC-TNADSP is faster than the FFT algorithms by 1.74 seconds.

5.1 Comparison of the Existing System with the Proposed System

The proposed system when compared with the existing system is of more efficiency than the existing system. Table 2 below illustrates the comparison more succinctly.

Table 3: Output Comparison of the FFT Algorithms with the FC-TNADSP

	NUMERICAL ALGORITHMS			**Validation of the FastAlgoDSP**
	FFT		**FC-TNADSP**	
Number of sampled DSP inputs (N)	**ALGORITHMS**	**Time (Sec)**	**Time (Sec)**	**Improvement upon FFT Algorithm (in Sec)**	**% Improvement**
N=1000	**Cooley-Tukey**	**3.44**	**1.70**	**1.74**	**50.58%**
N=1000 000	**Cooley-Tukey**	**3.44**	**1.70**	**1.74**	**50.58%**

5.2 Discussion of Results

The study investigated the Cooley-Tukeyalgorithmic model. The Bruun'sand Prime factor algorithms evaluated from the early analysis phase as they did not provide clear basis for further processing leading to the expected result. They however lacked the necessary parameters for efficient signal processing.The Cooley-Tukeyalgorithm is referred to in this study as default algorithm. The default algorithm was subjected to numerical decomposition, re-indexing and simplification in search of a faster algorithm for processing digital signals. The process yielded a faster algorithm.

The computing speed of the default algorithm was tested on the C^{++}platform. The execution time of the Cooley-Tukey was 3.44 seconds. Similarly, the execution time of the resultant improved algorithm was 1.74 seconds.Table.2 above shows that 1.74 second is new against the 3.44 seconds of the Cooley-Tukey. On comparing the speed of the default algorithmwith that of the new algorithm, we observed that there is (1.74) seconds speed improvement on the Cooley-Tukey algorithmas shown in table 2 above. In line with these outcomes, the new algorithm is referred to as FC-TNADSP-2. This result therefore shows that a version of algorithm with computing speed that is faster than the 0(nlogn) computing speed of the fast Fourier algorithms exist.

V. SUMMARY AND CONCLUSION

6.1 Summary

This study was aimed at designing afaster Cooley-Tukey numerical algorithm for digital signal processing. This aim was actually achieved. The study had as it basis the discrete Fourier Transform (DFT). This Algorithm was noted for a computing speed of $0(N^2)$. This algorithm is often too slow to be practical. This obvious drawback provided a basis for investigating other more practical algorithms namely the fast Fourier transform algorithms (FFT). The FFT algorithms exist in different spectrum, but only three were investigated; the Bruun's FFT, the PFA FFT, and the Cooley-Tukey FFT. The first two algorithms did not go beyond the surface analysis and as such limited our investigation to the Cooley-Tukeyalgorithm in the FFT spectrum above.

The Cooley-Tukey FFT algorithms was noted to be of O(nlogn) computing speed, a speed considered to be the fastest so far. Curious that improvement could be made out of the existing fast algorithms we carried out more in-depth and critical investigation into theCooley-TukeyFFT algorithm. Re-indexing, decomposition and subsequent simplification yielded a new design of the FFT algorithm in our investigation. The redesigned Cooley-TukeyFFT algorithm had less number of arithmetic operators which in effect was an indication of increase in computing speed. To justify this observation, we had to test the redesigned FFT algorithm on the C^{++}platform. The output of the test confirmed the improvementof the redesigned Cooley-Tukey FFT algorithm over the default Cooley-TukeyFFT algorithm. The redesigned Cooley-TukeyFFT algorithm eventually became the FC-TNADSP. It is called FastCooley-Tukeynumerical algorithm for Digital Signal Processing (FC-TNADSP). It is indeed faster than the defaultCooley-Tukeyalgorithm. FC-TNADSP has some 1.74 seconds computing speed improvement over the default FFT algorithms. It is therefore practical to conclude that this study truly achieved its aim with a tremendous positive impact on the existing algorithmic system.

6.2 Conclusion

In-depth analysis that proceeded the development of this study revealed that the DSP algorithms are predicated on the discrete Fourier transform (DFT) of $O(N^2)$ computing speed. The DFT gave rise to the fast Fourier transform (FFT) algorithms of O(nlogn) computing speed. There are spectrums of FFT algorithms. This study exhaustively investigated three; the prime factor FFT algorithm, the Bruun's FFT algorithm, and the Cooley-Tukey FFT algorithm. More critical evaluation redefined the focus of the study thereby narrowing the number of default FFT algorithms that were tested to the Cooley-TukeyFFT algorithm.The Cooley-Tukey FFT algorithm became the basis for comparing the new algorithm of this study.

In search of a faster algorithm, the established default algorithms were subjected to re-indexing, decomposition, and simplification. The re-indexing stage is to define and substitute each parameter variable in the original FFT algorithm. This effort redefined the apparent structure of FFT algorithm. The decomposition process explored the impact of the twiddle factor of unity ($i2\pi / N = 1$). Applying this factor into the re-indexed algorithm, resized the algorithm downward filtering out unnecessary variable parameters. This

downsized algorithm provided the direction of the expected new faster algorithm with the elimination of non-unique arithmetic operators. The presence of the non-unique operators contributed to the constraints of the default FFT algorithm. Their absence or elimination contributed to the efficiency of the new algorithm. The simplification effort adjusted the number of multiplication, exponentiation, addition, and subtraction operations and operators. At this stage, the algorithm became narrower, simpler, and of course faster also.

The three processes described above yielded f aster Cooley-Tukey algorithm that when tested on the C^{++} platform, proved faster than the defaultCooley-Tukey FFT algorithm. The speed advantage of the new algorithm was as high 1.74 seconds. The index frequency, K is the speed factor in this study. The K factor determines the speed of the algorithm while the signal index, n determines the quantum of the signal. The frequency index is therefore defined as: $K = \pi / e^{\theta}$ *iff* $0 \le \theta \le 8$ else $K = \pi\theta$ for $0 \le 8 \le \theta$; θ =x= dimension of array of input data. The developed fast algorithmof this study is therefore known as FC-TNADSP.

6.3 Recommendations

The result of this study is certainly going to enhance the computing speed of digital signals. The developed algorithms are basically recommended for the processing of digital signals and not analog signals. However, it can also be used for analog signals only when they are converted to digital signals. The algorithms were tested on input block width of 1000, and above, and can be implemented on input size of 100 000, and 1000 000 000 without the challenge of storage overflow. In order to optimized the advantage of the developed algorithms, the frequency index, K should be as defined in this study, that is $K = \pi e^{\theta}$ *iff* $0 < \theta \le 8$

REFERENCES

[1]. Vladimir, P. andZlatka, N., Georgi, I., Miglen, O., (2011). Complex Digital Signal Processing in Telecommunications: Applications of Digital Signal Processing, Dr. Christian Cuadrado-Laborde (Ed.), 307-406.
[2]. Saeed, B. (2003). Interpolation in Digital Signal Processing and Numerical Analysis.New York: Springer-Verlag.
[3]. Fraser, D. (1989). Interpolation by the FFT Revisited *An Experimental Investigation*, IEEE Transactions on Acoustics, Speech, and Signal Processing, (37)5, pp. 665-675.
[4]. Matthew, P.D. (2000). Efficient Digital Filters, IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-
[5]. Pedro, F. Z. T. (2009). Algorithms and tools for automatic generations of DSP hardware structures, Edition, Boston: McGraw Hill.pp.18-23, 317-320.
[6]. Jianxin, X., Johnson, J.mRebert, Padua, D. (2001). SPL: A Language and Compiler for DSP algorithms.
[7]. Auslander, L., Johnson, J., Johnson, R.W (1996). Automatic Implementation of FFT Algorithms, Technical Report DU – MCS – 96 – 01, Department of Mathematicss and Computer Sciences, Drexel University, Philadelphia, PA, Presented at the DARPA ACMP PI Meeting.
[8]. Frigo, M. and Johnson, S.G. (1998). FFTW: An Adaptive Software architecture for the FFT. In ICASSP, Vol.3, pp 1381 – 1384
[9]. Frigo, M. (1999). A fast Fourier Transform Complier. In PLDI.
[10]. Blesser, B. &Kates, J.M. (1978). Digital processing in audio signals. In A.V. Oppenheim, editor. Applications of Digital Signal Processing, Prentice Hall, Englewood Cliffs NJ.
[11]. Cooley James W., and John W. Tukey (1965). An algorithm for the machine calculation of complex fourier series. Mathematics computer. Vol. 19, 1965

APPENDIX A: SOURCE CODE

DEFAULT EQUATION_2: COOLEY-TUKEY FAST FOURIER TRANSFORM ALGORITHM

```
// equation3.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include<iostream>
#include<string>
using namespace std;
int main(){
doublek,w,xk,n,m,f,xq,wq;
cout<<"enter the number of input size n"<<endl;
cin>>n;
//cout<<"enter the even valued input m"<<endl;
cin>>m;
w=exp((2*3.142)/n);
xq=(pow(w,-0.5))*(pow(n,2));
wq=(pow(w,0.5))*(pow(n,2));
k=.01219;//index of frequency; -N+1, -N+2, ... -1 0, 1, ..., N-1
//m=n=even-valued input; 0, 1, 2, ..., N-1
//m+1= odd-valued input
```

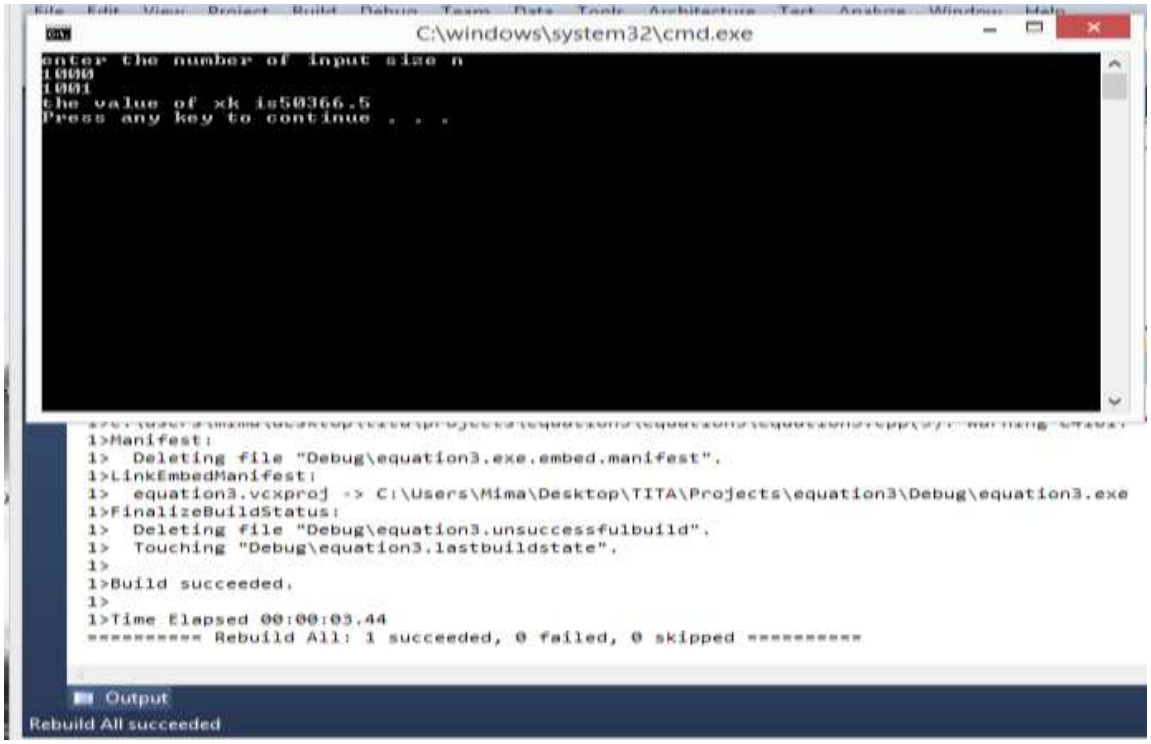
```
//N/2-1=upper bound
//xk=signal output
//n=index of signal x(n)
//xn=array lenth of input=maximum length of n
//f=2*m;
//w=exp((2*3.1416)/n);
n=m;
xk=(2*m)*exp(-2*3.142/n*2*m*k)+(2*(m+1))*exp(-2*3.142/n)*(2*(m+1))*k;//((-
//1.5*f*w*f*k)+(1.5*(f+1)+(w*f+1)*k));//-1.5=((n/2)-1:n=5,m=0)
cout<<"the value of xk is"<<xk<<endl;return 0; }
```

FASTNUMALGO_2: RE-INDEXEDCOOLEY-TUKEY FAST FOURIER TRANSFORM ALGORITHM

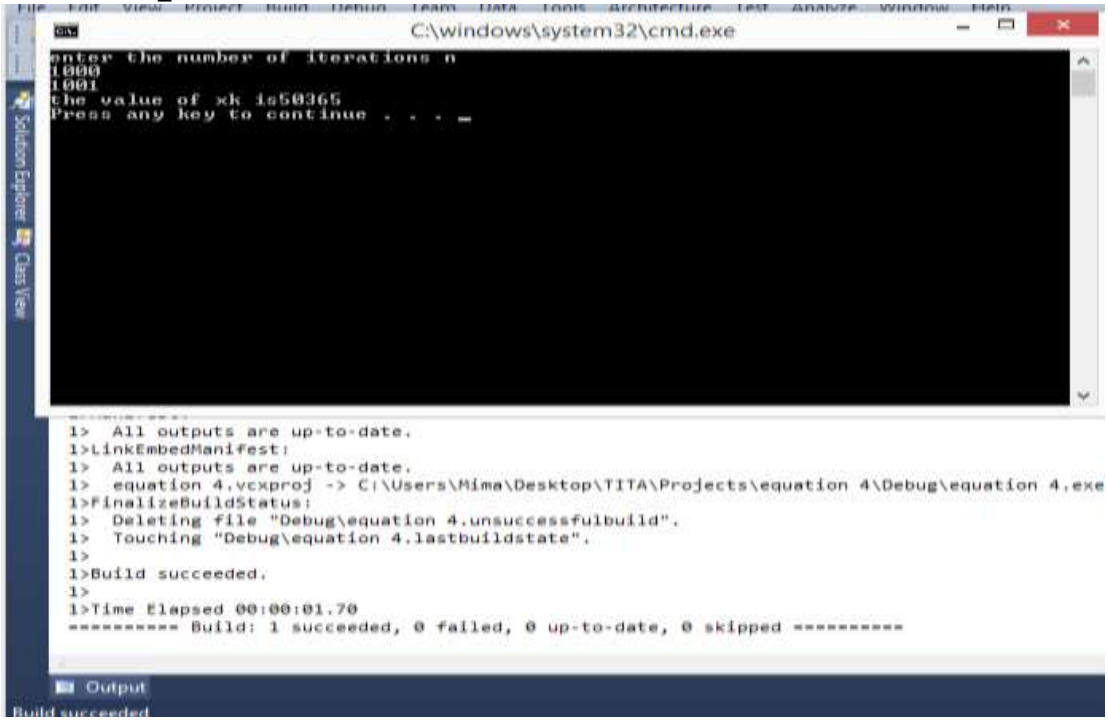
```
// equation 4.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include<iostream>
#include<string>
using namespace std;
int main(){
doublew,k,wq,xk,a,b,n,m,xq;
cout<<"enter the number of iterations n"<<endl;
cin>>n;
//cout<<"enter the even number, m"<<endl;
cin>>m;
w=exp((2*3.142)/n);
xq=(pow(w,-0.5))*(pow(n,2));
wq=(pow(w,0.5))*(pow(n,2));
k=6.2862;//index of frequency; -N+1, -N+2, ... -1 0, 1, ..., N-1
//m=n=even-valued input; 0, 1, 2, ..., N-1
//m+1= odd-valued input
//N/2-1=upper bound
//xk=signal output
//n=index of signal x(n)
//xn=array lenth of input=maximum length of n
//          x=1
n=m;
a=(4*m*m);
b=(2*m+1)*(2*m+1);
xk=(pow(n,-1))*k*(a+b);//(-1.5)*a*b;//-1.5=(n/2-1:n=5,n=0)
cout<<"the value of xk is"<<xk<<endl;
return 0;}
```

APPENDIX B: SAMPLE OUTPUTS

Default Equation_2:Cooley-Tukey Fast Fourier Transform Algorithm

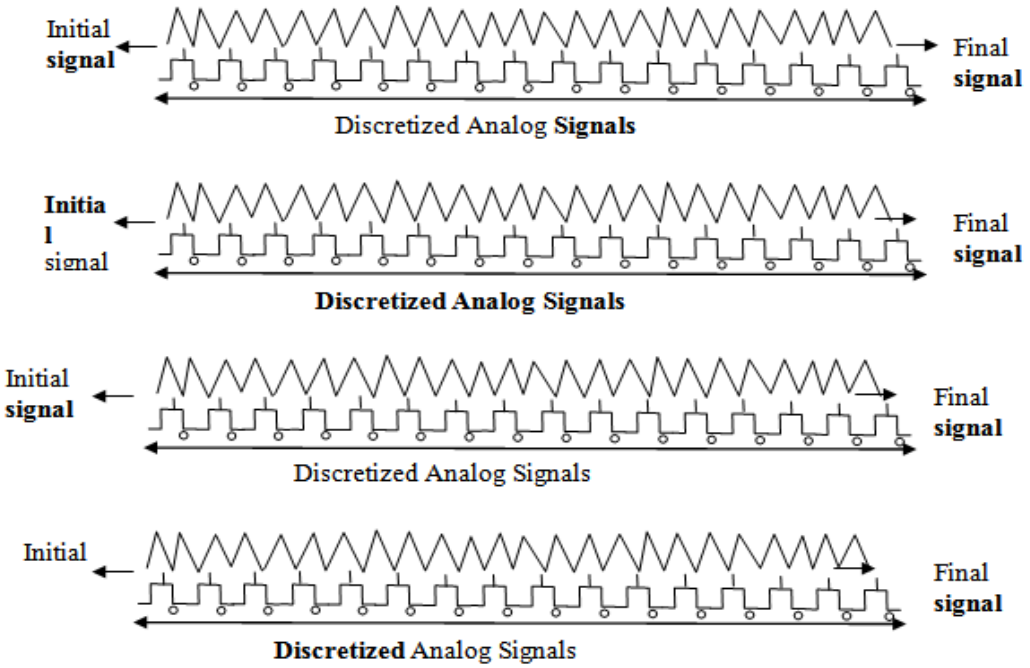


FASTNUMALGO_2: RE-INDEXEDCOOLEY-TUKEY FAST FOURIER TRANSFORM ALGORITHM



APPENDIX C: SAMPLED WAREHOUSE DIGITAL SIGNALS (ADAPTED FROM [20])

APPENDIX C: SAMPLED WAREHOUSE DIGITAL SIGNALS (ADAPTED FROM [20])



Translating Digital signals to numerical values

000 => 0 unit signals
1010 =>10 unit signals
110010 =>50 unit signals
1100100 => 100 unit signals
1111101000 =>1000 unit signals
1110001000110000 => 125 000 unit signals
1110001000110000 => 250 000 unit signals
1110001000110000 => 500 000 unit signals
1110001000110000000 = 1000 000 unit signals

Index (k)	Binary	Bit-reversed binary	Bit-reversed index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Amannah. " Development of Fast Performance Evaluation Algorithm to Optimize the Efficiency of the Cooley-Turkey Fast Fourier Transform Algorithm for Dip" International Journal of Precious Engineering Research and Applications (IJPERA), vol. 03, no. 01, 2018, pp. 01–15.