Efficiency Comparison of DFT/IDFT Algorithms for OFDM Implementation via MATLAB

Ahmed Ismail El Hoofy

*Abstract*

— Orthogonal Frequency Division Multiplexing (OFDM) is widely used in many digital communication systems due to its advantages such us high bit rate, strong immunity to multipath and high spectral efficiency. In other words, we could say that Multiple sub-channels (sub-carriers) carry samples sent at a lower rate. This is achieved by using Almost same bandwidth with wide-band channel. One of the most important parts of this procedure is Fast Fourier Transformation (FFT), which plays a significant role in making a change in the world of communication and we will discuss how this is approached later in the paper. In this paper we investigate various algorithms for performing Fast Fourier Transformation (FFT)/Inverse Fast Fourier Transformation (IFFT). Those algorithms are being used in the process of implementation the FFT/IFFT part in Orthogonal Frequency division multiplexing (OFDM)

***Keywords used: - —* OFDM system, DFT, IDFT, FFT, IFFT, DIT-FFT, DIF-FFT, Cooley-Tukey (C-T) algorithm****, Radix-2, TX, RX, Radix-2 DIT, Radix-2 DIF, Radix-4 DIF, PFA, QPSK, BPSK and BER.**

# **INTRODUCTION**

O

RTHOGONAL Frequency Division Multiplexing (OFDM) is a technique widely used in many digital communication systems such as Digital Television (DTV), Digital Audio Broadcasting (DAB), Terrestrial Digital Video Broadcasting (DVB-T), Digital Subscriber Line (DSL) broadband internet access, standards for Wireless Local Area Networks (WLANs), standards for Wireless Metropolitan Area Networks (WMANs), and 4G mobile communications. It has many advantages such us high bit rate, strong immunity to multipath and high spectral efficiency. However, one of the most serious components of OFDM is Inverse Fourier Transform (IFFT) and Fourier Transform (FFT) of the transmitted and received OFDM signal respectively, since this Fourier Transform is a fundamental role player in the performance of the OFDM.

The OFDM system design is subsequently divided into three parts which convey the three main apparatuses of a communication system: the transmitter, channel, and receiver. These three major components are divided more into smaller sub-components, which will be briefly discussed in this paper.

The aim of this paper is to provide various algorithms and comparison between them with respect to implementation of IFFT and FFT in OFDM and how each algorithm is different. We will bring out the advantage sand disadvantages of each algorithm, respectively. Those topics are considered important for any communication system engineer

.

Finally, the conclusion will discuss the observations of the algorithms and any important results regarding the implementation FFT/IFFT part for the project.

# **Overview**

* 1. ***System Block Diagram***

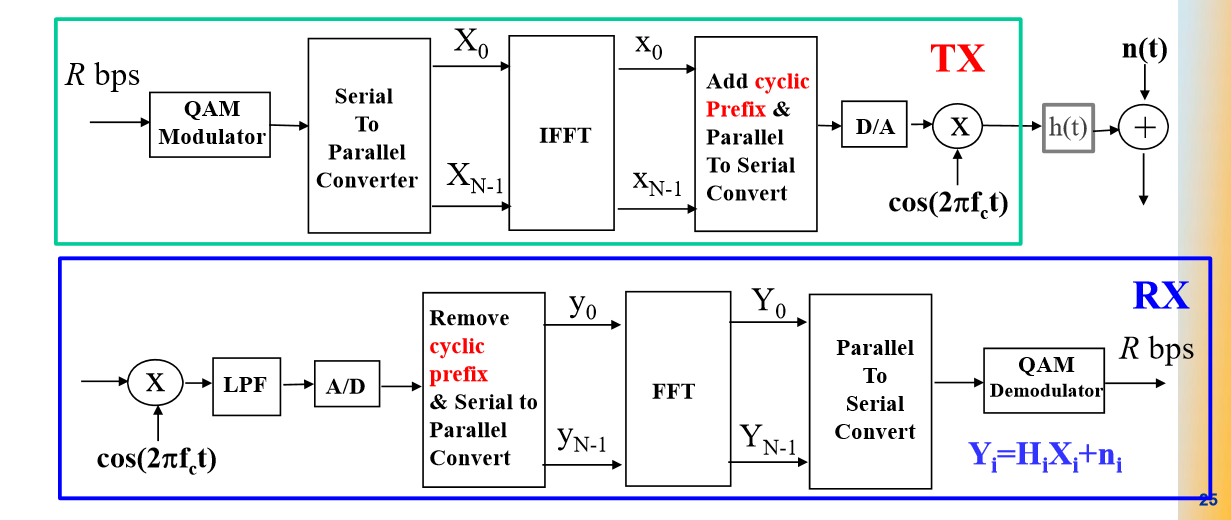
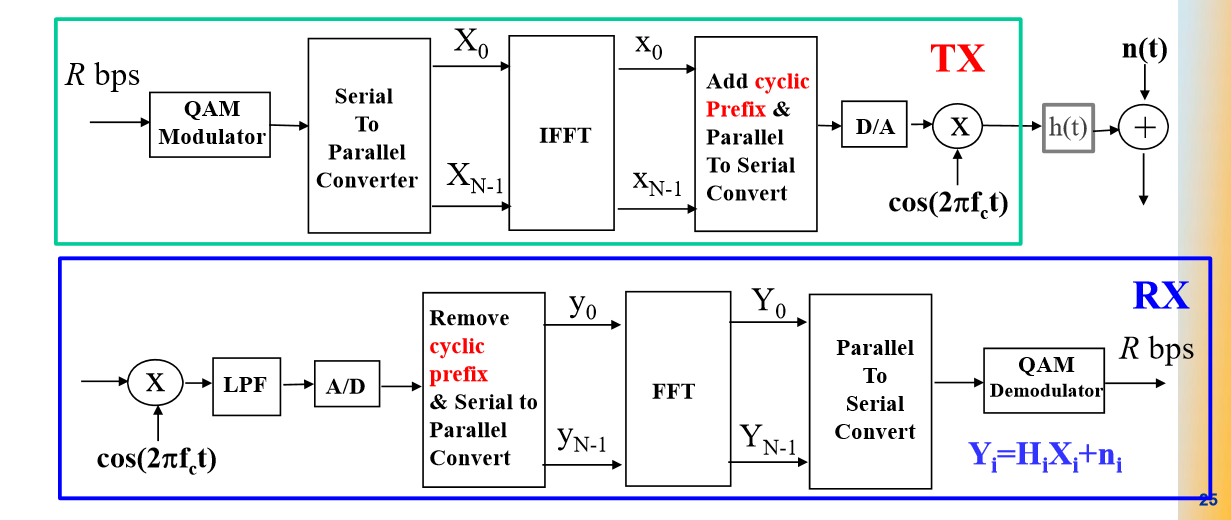


Figure 1. System block diagram

A general system diagram is shown in figure 1. OFDM is divided into three parts: - Transmitter, Channel and Receiver. While the complexity of the diagram may be deceptive, it in fact supplies a foundation in which to start the design of the OFDM system. The Transmitter (TX) is subdivided in 4 subdivisions and the Receiver is mostly the inverse of the Receiver, which make sense to retrieve the sent information as we already have from the beginning.

As shown in the diagram, the communication system uses IFFT in the transmitter (TX)side, while it introduces FFT at the receiver (RX) side. The difference between them will be discussed later.

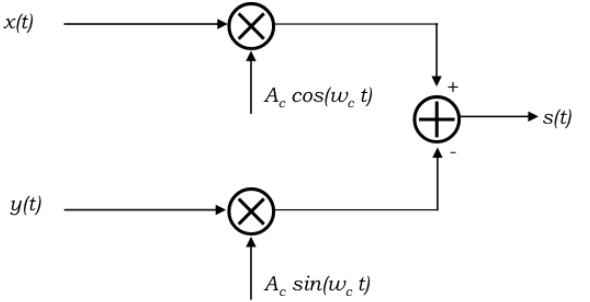
* 1. ***Transmitter***

*Figure 2. Zoomed-in Transmitter Block Diagram*

A close overview of the Transmitter part is shown in figure2. According to the shown diagram, the TX consists of (Quadrature-Amplitude Modulation (QAM)-modulator block, serial-to-parallel block, IFFT block, Cyclic Prefix block and the Digital to analog conversion part. Each sub-block will be briefly discussed in following sections.

**b.*i*  QAM-Modulator**

Quadrature-Amplitude Modulation (QAM) is a method of combining two amplitude-modulated (AM) signals into a single channel, thereby doubling the effective bandwidth. Additionally, QAM is used with pulse amplitude modulation (PAM) in digital systems, especially in wireless applications. there are two carriers, each having the same frequency but differing in phase by 90 degrees (one quarter of a cycle, from which the term quadrature arises). One signal is called the Imaginary part of the transmitted signal, and the other is called the real signal. Mathematically, one of the signals can be represented by a sine wave, and the other by a cosine wave, respectively.



*Figure 3, QAM modulator schematic Diagram.*

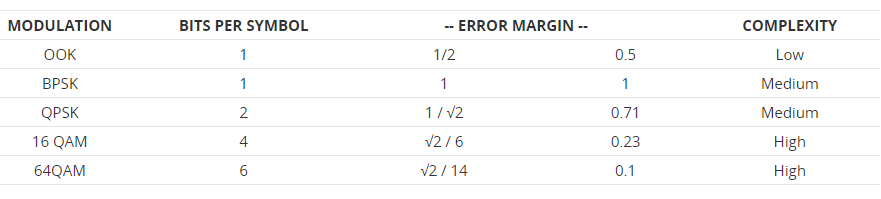
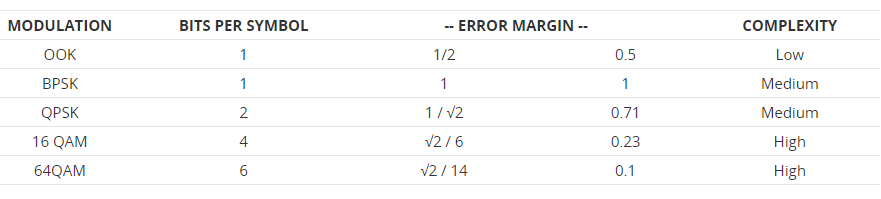
According to figure 3, the *x(t)* is the Real part of the signal while the *y(t)* is the imaginary part. S*(t) = x(t)Accos (wc (t)) + y(t) Acsin (wc(t)) 🡪 s(t) = x(t) + j y(t)* Therefore, the transmitted signal or bits are being sent to two different carriers as shown in figure 3. The main reason for that is because QAM utilizes both amplitude and phase components to supply a form of modulation that can supply elevated levels of spectrum usage efficiency.

Moreover, we could say that QAM is a form of modulation which is widely used for modulating data signals onto a carrier used for radio communications. Since both amplitude and phase variations are present it may also be considered as a mixture of amplitude and phase modulation. The major benefit of QAM modulation variants is efficient usage of bandwidth. This is since QAM represent a greater number of bits per carrier. For example, 16QAM maps 4 bits per carrier, 64QAM maps 6 bits per carrier, 256QAM maps 8 bits per carrier and so on.

Although QAM appears to increase the efficiency of transmission for radio communications systems by using both amplitude and phase variations, it has several drawbacks. The first is that it is more susceptible to noise because the states are closer together so that a lower level of noise is needed to move the signal to a different decision point. Receivers for use with phase or frequency modulation are both able to use limiting amplifiers that can remove any amplitude noise and thereby improve the noise reliance. This is not the case with QAM.

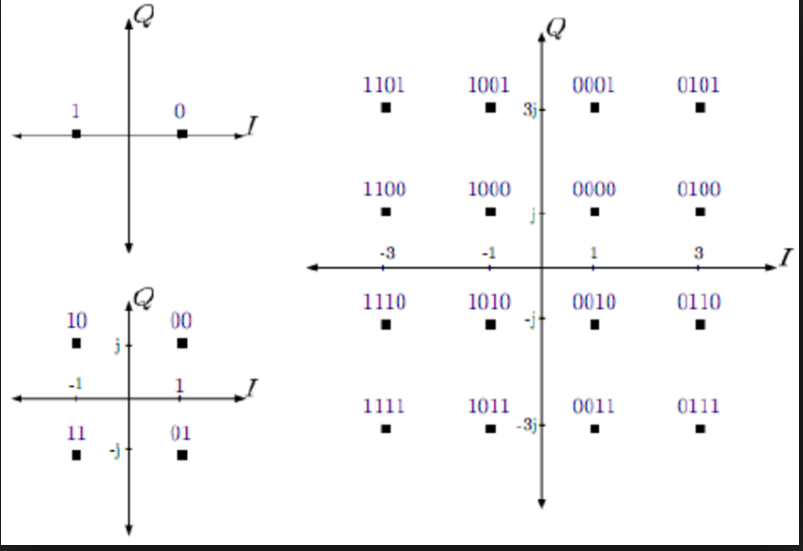
The second limitation is also associated with the amplitude part of the signal. When a phase or frequency modulated signal is amplified in a radio transmitter, there is no need to use linear amplifiers, while when using QAM that holds an amplitude part, linearity must be kept. Unfortunately, linear amplifiers are less efficient and consume more power, and this makes them less attractive for mobile applications.

*Table 1, summary of types of modulation with complexity.*

****

Typically, it is found that if data rates above those that can be achieved using 8-PSK are needed, it is more usual to use quadrature amplitude modulation. This is because it has a greater distance between adjacent points in the I - Q plane and this improves its noise immunity. As a result, it can achieve the same data rate at a lower signal level.

However, the points no longer the same amplitude. This means that the demodulator must detect both phase and amplitude. Also, the fact that the amplitude varies means that a linear amplifier must amplify the signal.

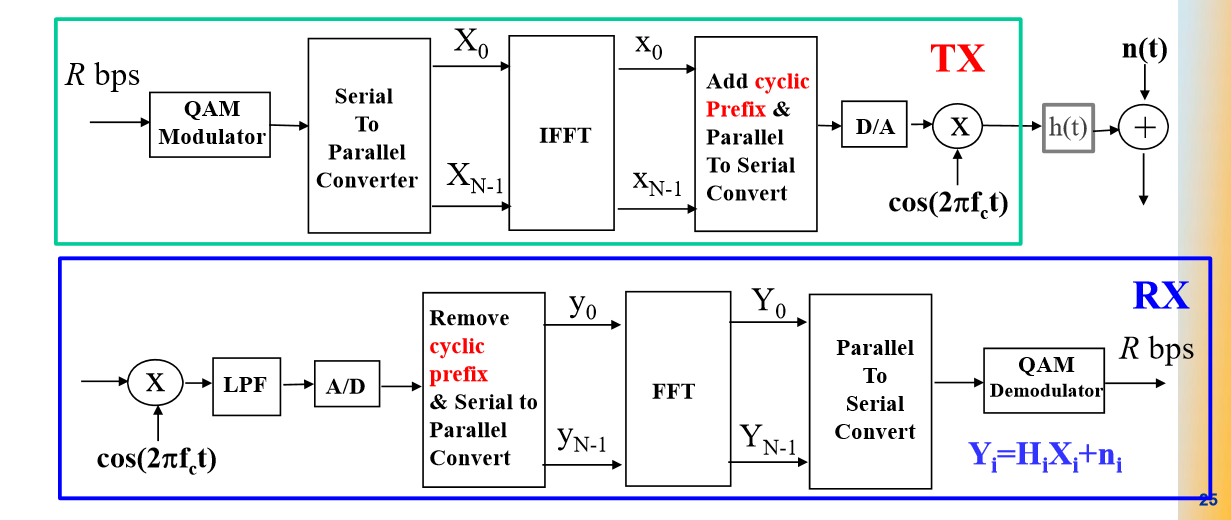


*Figure 4, BPSK, 4-QAM, and 16-QAM constellation Diagrams.*

Apparently, As the QAM order increases, so the distance between the different points on the constellation diagram decreases and there is a higher possibility of data errors being introduced. To use the high order QAM formats, the link must have an exceptionally good Eb/No otherwise data errors will be present. When the Eb/No deteriorates, then other the power level must be increased, or the QAM order reduced if the bit error rate is to be preserved.

Accordingly, there is a balance to be made between the data rate and QAM modulation order, power, and the acceptable bit error rate. Whilst further error correction can be introduced to mitigate any deterioration in link quality, this will also decrease the data throughput.

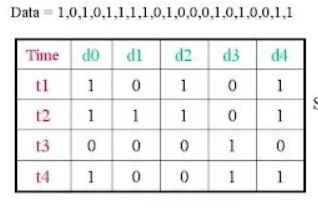
**b.*i*  Serial-to- parallel (S/P) converter**



*Figure 5, Serial-to- parallel (S/P) converter*

After the input bits are modulated by QAM modulator, they are converted into parallel bits. In other words, the input serial data stream is formatted into the word size needed for transmission. For example, if we have a serial input bit called Data = [1,0,1,0,1,1,1,1,0,1,0,0,0,1,0,1,0,0,1,1]. Those bits are converted as following: -

Table 2, an approach for serial-to-parallel conversion



The aim of the serial to parallel converter is to receive the data that is going to be transmitted. The serial to parallel converter receives the M serial bits to be transmitted, and those bits will be divided into N sub-blocks of mn bits each sub-block called **symbols**. The amount of bit of each channel can be different Those N sub-blocks will be mapped by the constellation modulator using Gray codification, this way an + jbn values are obtained in the constellation of the modulator.

The serial to parallel converter at the receiver has the function to receive the data that is going to be demodulated, with the same structure as it was at the transmitter.

To store the M bits a buffer that will contain all the input data into different memory positions is needed. To obtain the M data bits at the output we will need the buffer to stop reading data, another possibility is that the amount of data stored at the buffer is 2M, this way is not necessary to stop the reading, this way can read continuously.

**b.*ii*  Inverse Fast Fourier Transform (IFFT)**

The OFDM modulation can be obtained through an IDFT. Mathematicians figured out various efficient algorithms to reach a fast implementation of IDFT, they called it Fast Fourier Transform (FFT and IFFT). IFFT can be used for reducing the time of processing and the used hardware.

The demodulation, in the same way, can be made by DFT, or better, by FFT, that is its efficient implementation. FFT calculates DFT with a great reduction for operations, leaving several existent redundancies in the direct calculation of DFT. The **only limitation** of those algorithms is that is only valid for sequences of length 2N otherwise modified algorithms with less efficiency are needed.

In this Project, we implemented various FFT and IFFT algorithms as an approach for a comparison between these algorithms and how their computation time and bit error rate is different

The implemented algorithms are: -

1)- Cooley-Tukey (C-T) algorithm

2)- Prime Factor algorithm (PFA) or “Good-Thomas algorithm”

3)- Radix\_2 Decimation-in-time/Frequency (DIT/F)

4)-Radix\_4 Decimation in frequency (DIF)

**IFFT and FFT in OFDM: -**

* The **OFDM uses** the very efficient algorithm of the **FFT** to perform the QAM modulation (in the transmitter) and demodulation (in the receiver) of the channels.
* Unlike in most of the applications of **IFFT**, the main purpose of this operation in **OFDM** is NOT to convert a signal from frequency domain to time domain.
* The IFFT is performed on the samples only to ease single demodulation procedure for modulated signals of all orthogonal frequencies used in the system, i.e., if we do not use IFFT, we will have to use different demodulation technique for different carrier frequency.

**b.*iii*  Adding Cyclic Prefix & (P/S) Converter**

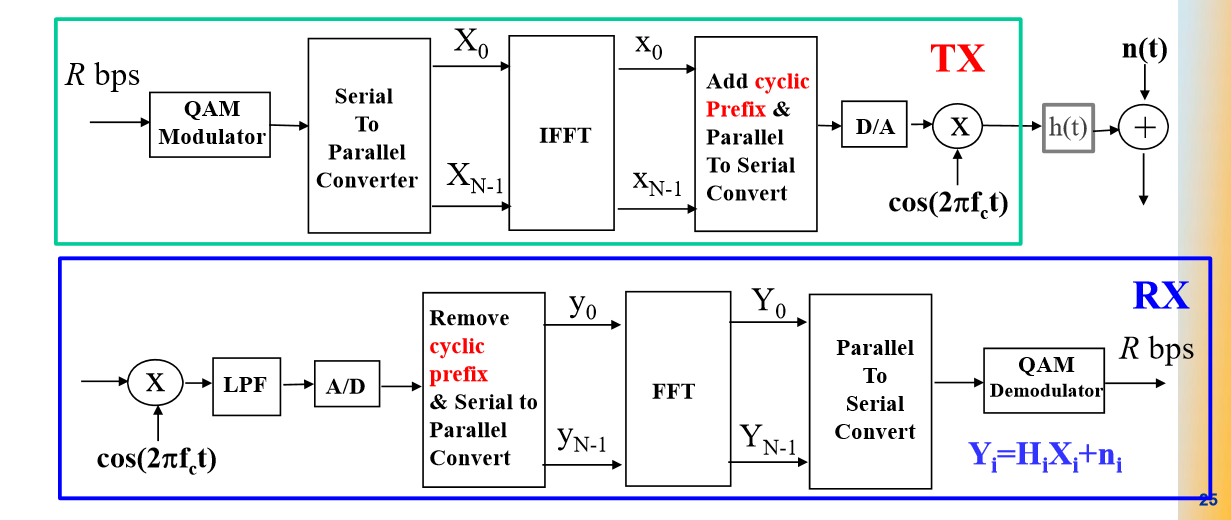


Figure 6, Adding Cyclic prefix & P/S converter Block

Use of **cyclic prefix (CP)** is a key element of enabling the OFDM signal to work reliably. The cyclic prefix acts as a buffer region or guard interval to protect the OFDM signals from **intersymbol** interference. Cyclic Prefix replace the conventional null guards of the OFDM symbol. This cyclic extension converts the linear convolutive channel to simulate a channel performing cyclic convolution, thus ensuring orthogonality over a time dispersive channel, and cutting ISI completely between subcarriers if the cyclic extension still is longer than the impulse response of the channel:

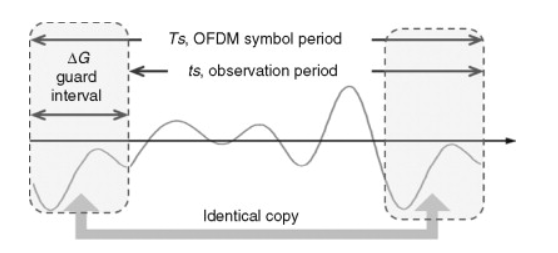


Figure 6, Graphical Representation of adding the CP to a signal

As shown in figure 6, To Add a CP part to your signal, you would have to copy the very last part of the desired signal or information bits and add it to the incredibly early beginning of the signal. Obviously, this will increase the time of your symbol Ts and decrease your Bs (Bandwidth). So, to achieve this You should increase OFDM symbol duration, i.e. increasing the number of subcarriers. Therefore, the main reason that we for using Cyclic Prefix is that it primarily acts as a guard band between successive symbols to overcome intersymbol interference (ISI).

While, the (P/S) Converter takes the extended symbols and converts them back from parallel bits in a serial extended bit to be ready for channel transmission. In other Words, we can say that the parallel to serial (P/S) converter is only the opposite function of the serial to parallel (S/P) converter, and it is placed just before sending the data through the channel by the digital to analog converter, at the transmitter.

**b.*iii*  Digital-to-Analog Converter (D/A) & Channel**

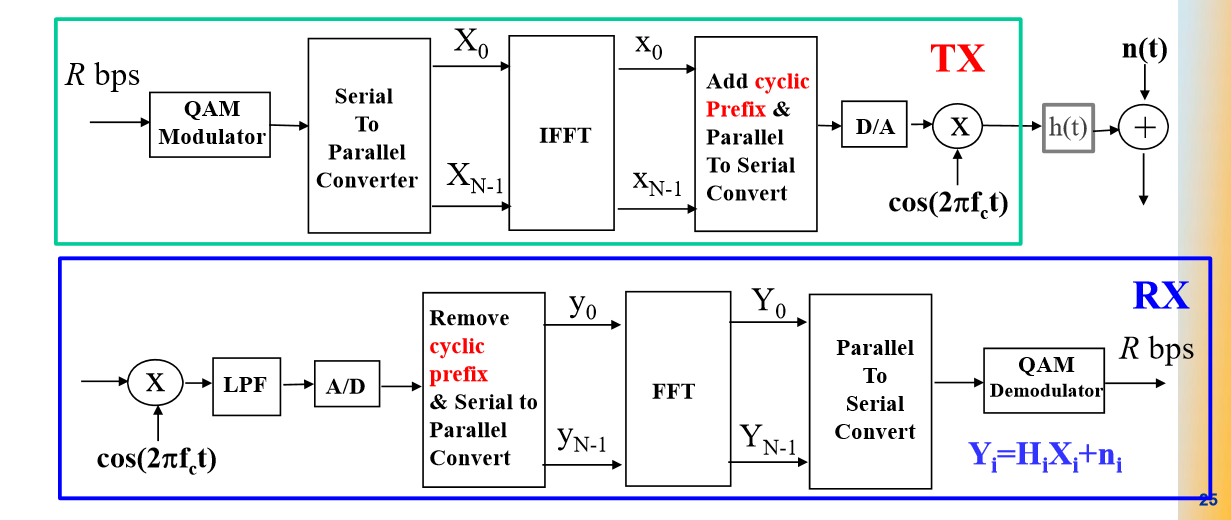


Figure 8. block Diagram of (D/A) converter, followed by passing the signal through the channel.

The main use of D/A converter is that it takes digital data and transforms it into an analog audio signal. Afterward, it sends that analog signal to an amplifier. For example, when you hear digital recordings, you are listening to an analog signal that was converted from digital by a D/A converter.

Communications Channels to the way this information flows within the organization and with other organizations. For the flow of information and for a manager to handle his employees, it is important for an effectual communication channel to be in place.

In **an analog channel** model, the transmitted message is modelled as an analog signal. The model can be a linear or non-linear, time-continuous, or time-discrete (sampled), memoryless or dynamic (resulting in burst errors), time-invariant or time-variant (also resulting in burst errors), baseband, passband (RF signal model), real-valued or complex-valued signal model. The model may reflect some impairments: - 1)- AWGN Noise. 2)- Signal Distortion. 3)- Phase shift or Amplitude Shift.

**c. Receiver**

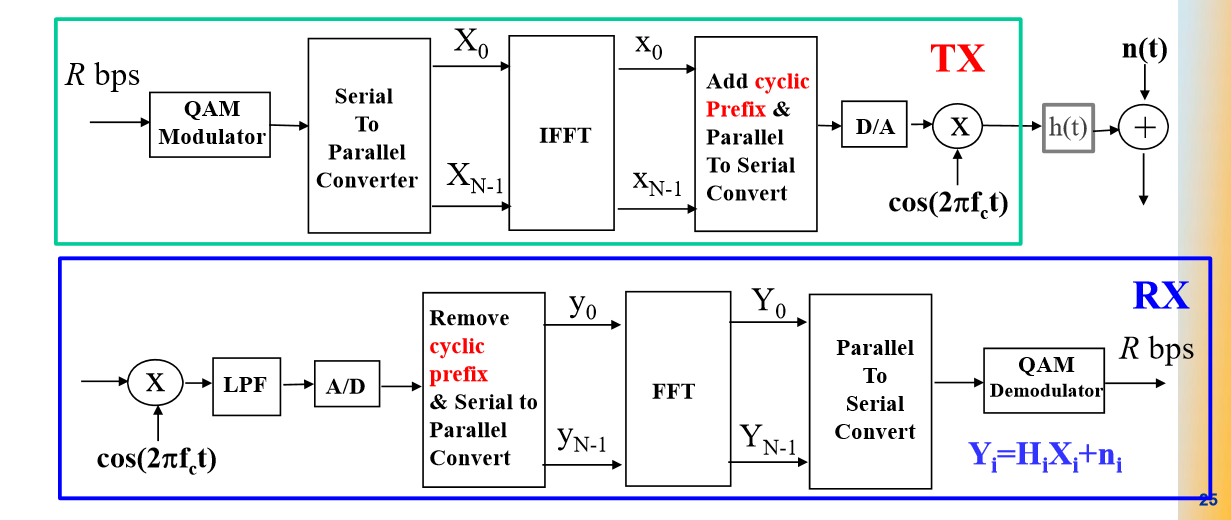


Figure 9. Receiver

A close overview of the Receiver part is shown in figure 7. According to the shown diagram, the RX consists of Low-Pass-Filter (LPF), Analog-to Digital Converter (A/D), Removing Cyclic prefix and (S/P) Converter block, FFT block and parallel-to-serial (P/S) converter. Each sub-block will be the opposite operation of the Transmitter sub-blocks. This Process is done in order to retrieve the sent information bits.

***c.i Demodulation, LPF and A/D blocks***

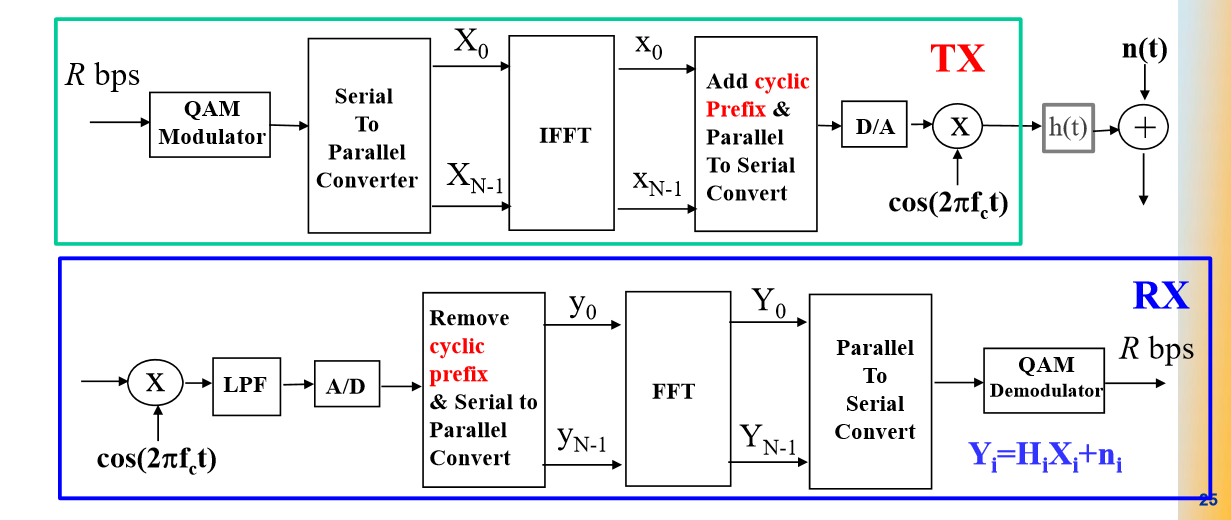


Figure 9. Zoomed in blocks Diagram of the received signal Demodulation and (D/A) Conversion

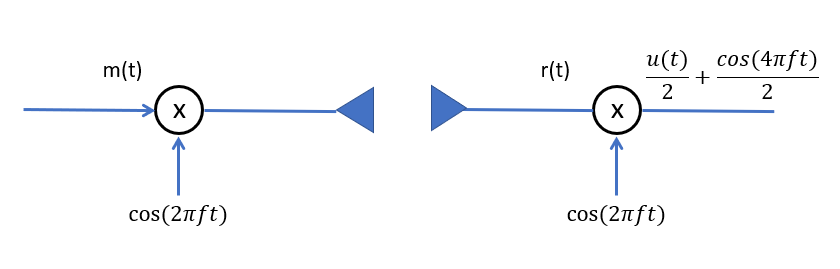
The use of cos(2πfct) could be explained by the following Diagram: - 

Figure 10. Up conversion and Down conversion Schematic Diagram

Firstly, as shown in figure 10, the modulated message has zero center frequency(baseband). it is also Impractical to have antennas at that frequencies. Additionally, it Causes interference if everyone wants to use baseband. Therefore, the receiver needs to perform an operation of down-conversion because the received signal is a high frequency signal in Radio Frequency (RF) and Processing the data at these frequencies needs high clock digital circuits, which is impractical. As a solution to this issue, we need to convert the data back to baseband and process the low frequency signals for decoding bits.

Secondly, the Low-pass Filter (LPF) is needed as well to get rid of aliasing effects, as these faults are impossible to correct for digitally. In a simpler explanation, we could say that Aliasing is considered a fundamental problem in the digitalization process whereby part of the information in the waveform is lost and can't be recreated. So, the high frequency of the spectrum (in this case high frequency is everything above half of the sample frequency) needs to adequately filter out before the AD conversion occurs.

Lastly, the A/D converter which is essential in changing the received filtered signal from analog in to digital. The main purpose of converting the signal from analog to digital is to manipulate the data using a microprocessor. Therefore, we need to convert the analog signals to the digital signals, so that the microprocessor will be able to read, understand and manipulate the data.

***c.ii Removing Cyclic Prefix and (S/P) converter***

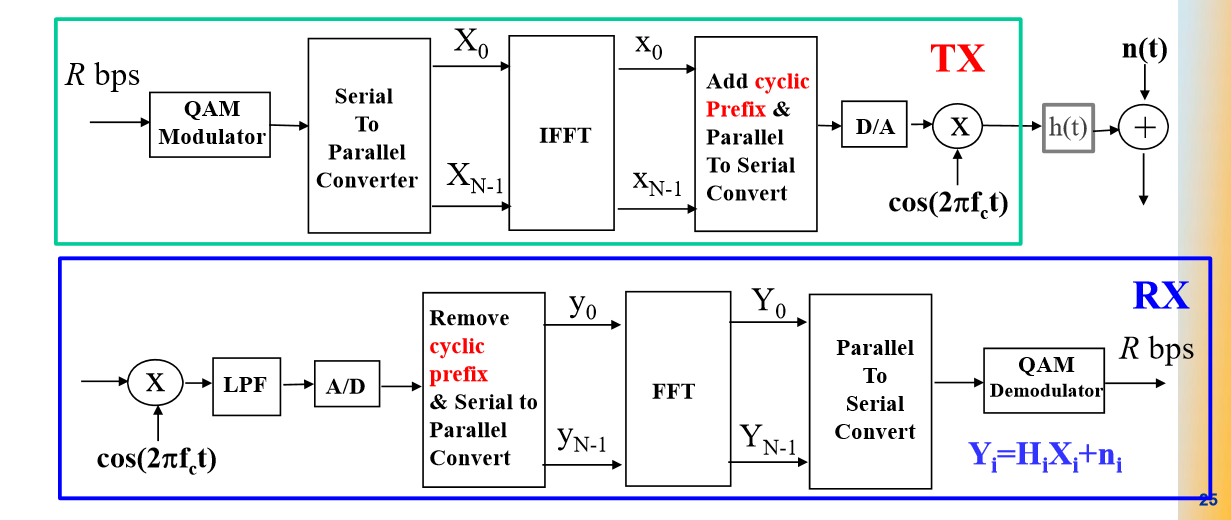


Figure 11. Removing Cyclic Prefix and (S/P) conversion Diagram

As mentioned before in the Transmitter section, the purpose of the cyclic prefix is to combat ISI and intra-symbol interference. The cyclic prefix is a guard time made up of a replica of the time-domain OFDM waveform. The basic premise is to replicate part of the back of the OFDM signal to the front to create the guard period. The duration of the cyclic prefix is chosen such that it is longer than the maximum delay spread which could be called τmax caused by the multipath channel. Additionally, the starting point for sampling of the OFDM symbol on the receiver side must be somewhere in the interval (τmax, Tcp) as shown in Figure12. This ensures that the received signal contains all the channel multipath effects.

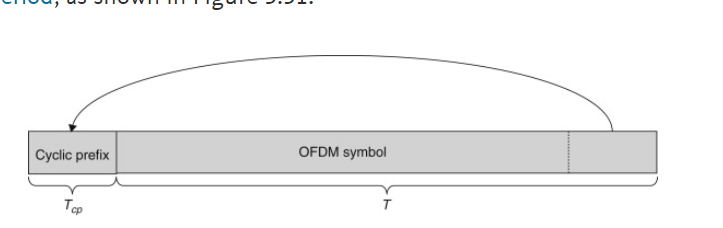


Figure 12. Cyclic Prefix Time duration diagram

Once the cyclic prefix of proper length has been removed, the received signal is decomposed into separate subcarriers using the DFT. Then, to equalize the gain of the desired signal, the subcarriers are multiplied with the inverse of the channel frequency response across each of the subcarriers.

Next, the (S/P) conversion was explained before. the reason it is used after removing the cyclic prefix is that it is needed in order to send every bit by itself into the FFT block Diagram to be processed. This will decrease the Computation time and the uses of Hardware.

***c.ii Fast Fourier Transform (FFT) block Diagram***

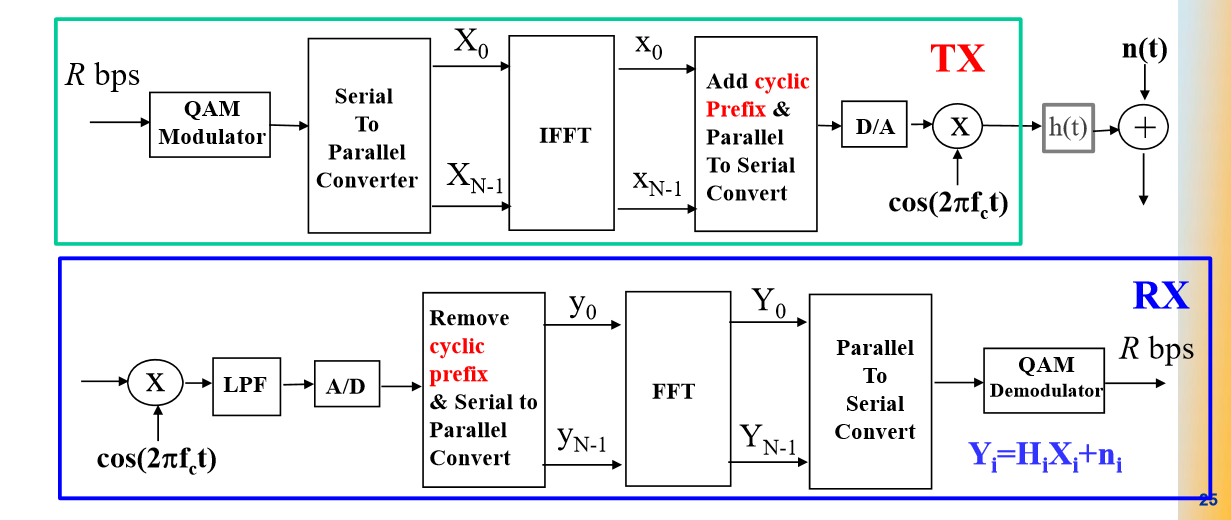


Figure 13. FFT block Diagram

The IFFT is constructed in the Transmitter path where the OFDM, while the FFT created in the receiver side where symbols are converted back to the frequency domain by FFT. To reduce the mathematical operations used in the calculation of DFT and IDFT one uses the fast Fourier transform algorithm FFT and IFFT which corresponds to DFT and IDFT, respectively. So, in summary the signal is easier synthesized in discrete frequency domain in the transmitter and to transmit it must be converted to discrete time domain by IFFT.

***c.iii Parallel-to-series (P/S) converter***

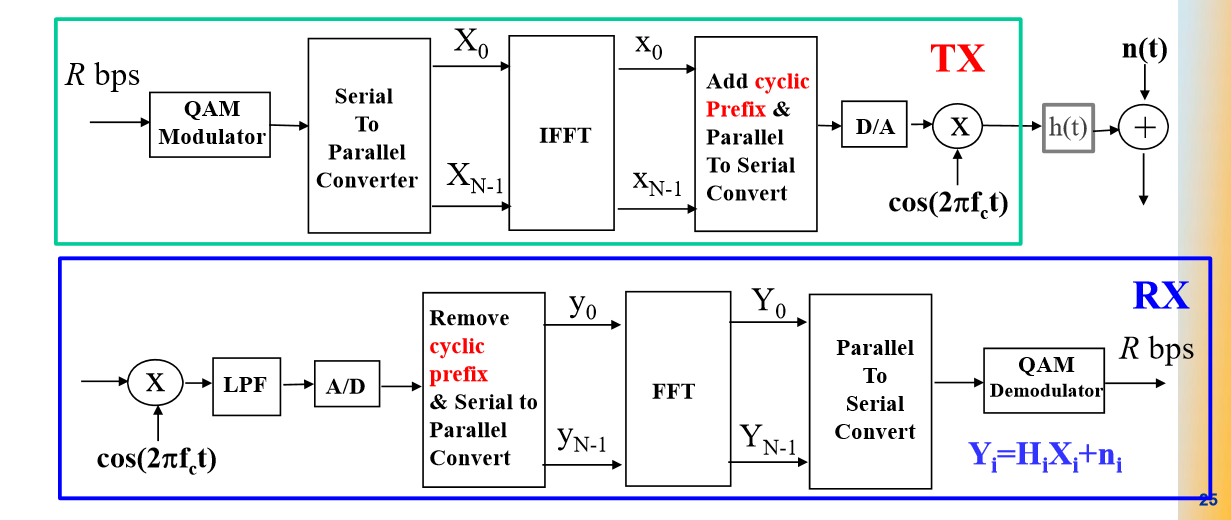


Figure 14. (P/S) block Diagram

As discussed before in the Transmitter section, the (P/S) Converter takes the extended symbols and converts them back from parallel bits in a serial extended bit to be ready for channel transmission. While, in the Receiver, the extracted bits from the FFT outputs are in parallel to each other. In order to retrieve our serial information bits that was originally sent entered through the transmitter block diagram, we would have to convert the output of the FFT process from parallel in to serial bits. In other Words, we can say that the parallel to serial (P/S) converter is only the opposite function of the serial to parallel (S/P) converter, and it is placed also just Right after receiving the output of the FF, at the Receiver.

***c.iii QAM -Demodulator***

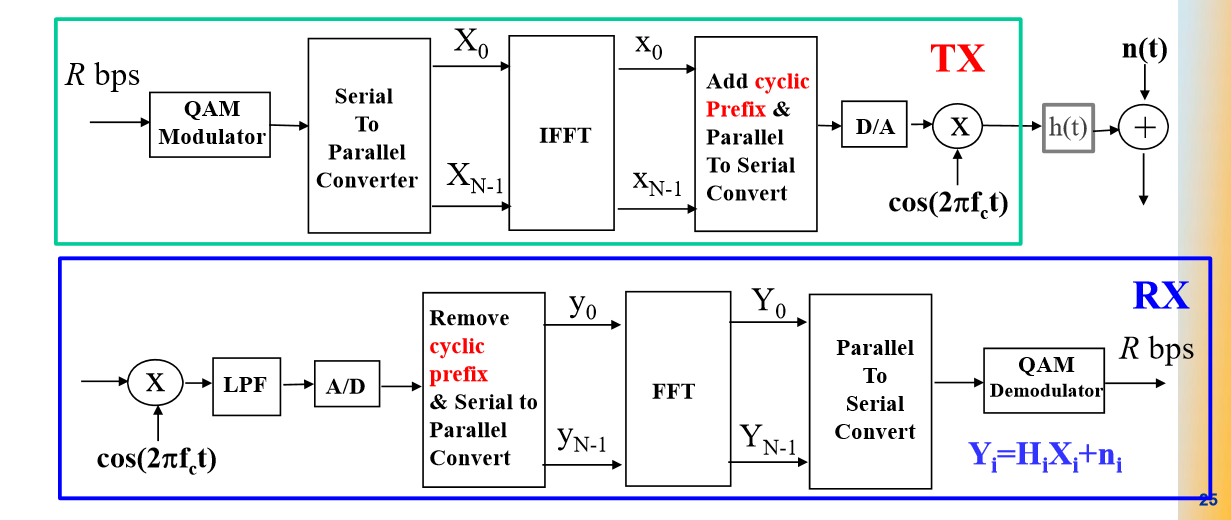


Figure 15. QAM-Demodulator block Diagram

The demodulation section is by far the most complicated part of the QAM model. The demodulator must detect the phase and amplitude of the signal, decode the symbol based on the phase and amplitude and then finally convert the data back to a serial stream. To complete the symbol demodulation, recovery of both the carrier and the symbol clock is required. With both a sub-carrier and cthe time duration recovered, any symbol can now be determined from the incoming signal. The heart of the demodulator is a time limited integration. First the signals are split and multiplied by sin (2πfct) and cos (2πfc t). The clock is then used to reset an integrator that acts on each of these input signals. The two analog signals must then be sampled and held just before the end of each period.

Finally, the output of the QAM-demodulator is the desired R bits that were already sent from the very early beginning in the transmitter side of the system. Obviously, there will be some lost and distorted bits, and this is due to various reasons, some of those reasons are the channel noise and the signal to noise ratio (SNR).

Now, we will focus on the FFT and IFFT algorithms that were used in the project to achieve a good comparison between the FFT/IFFT implementations.

**III. FFT Implementations**

1. ***Radix-2***

To understand the basics of an FFT, it is often useful to look to a special flow diagram.

Figure 16 shows a diagram for an 8-point radix-2 DIT-FFT (decimation in time-FFT). There are several ways to calculate a radix-2 FFT because the derivation from the DFT can be performed differently. Finally, we end up with the distinction of decimation in time and decimation in frequency, depending on how the twiddle factors are arranged in the butterfly. In addition, we can have bit-reversed inputs or outputs. The scrambling caused by the bit-reversal can be corrected in the first or the last stage of the FFT.

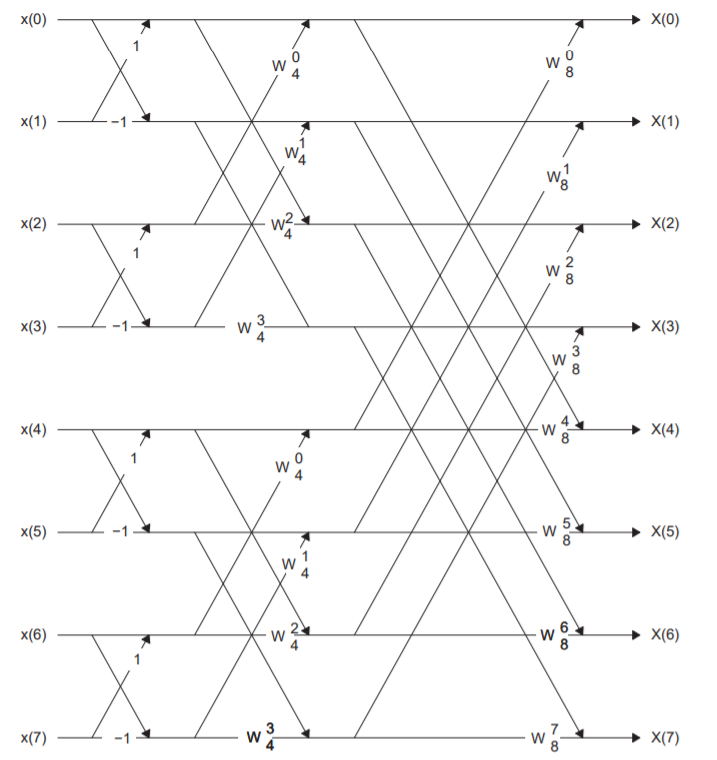


Figure 16. 8-Point DIT-Radix2-FFT

Noticeably, in stage 1, no multiplication is needed, since the twiddle factor: -

**WN0**

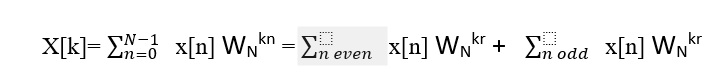
The Radix-2 algorithm can be summarized into 3 main steps: -

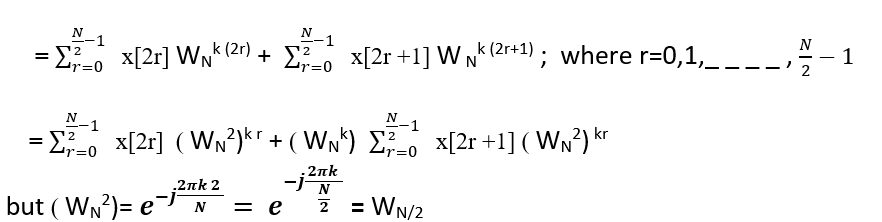
• Build a big FFT from smaller ones

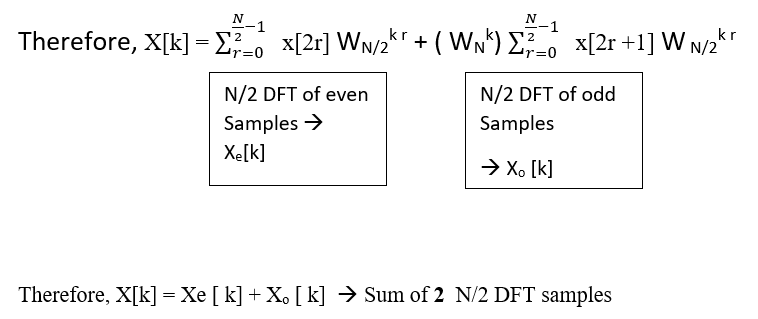
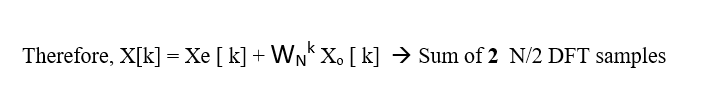
• Assume N=2m (Input array length is powers of 2)

• Separate *x(n)* into even and odds indexed sequences

**X[K] is the Fourier Transform of *x(n)***





According to the last, Mathematical Formulas, the Radix-2 FFT algorithm is a doable method in reaching a satisfactory resultant for DFT. The Fast Fourier Transform (FFT) converts a time-based signal into its corresponding frequency-based signal by manipulating the sum of orthogonal components. The time-based signal spectrum is indicated by the phase and amplitude of those components. Inverse Fast Fourier Transform (IFFT) does the reverse process, thus converting the spectrum back to time signal. Every point of data present in the spectrum is called a bin.

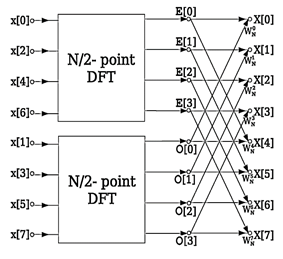


Figure 17, DIT-FFT block Diagram

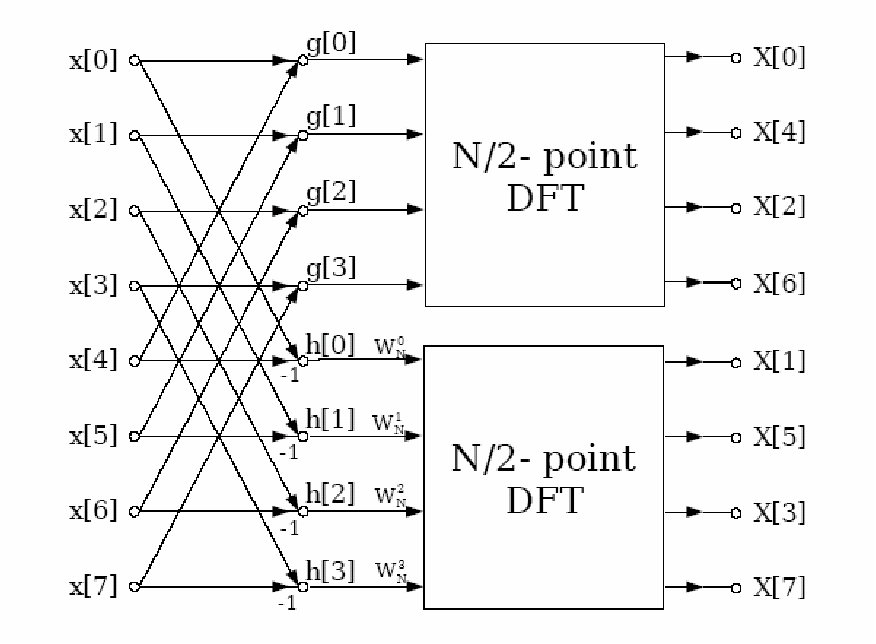
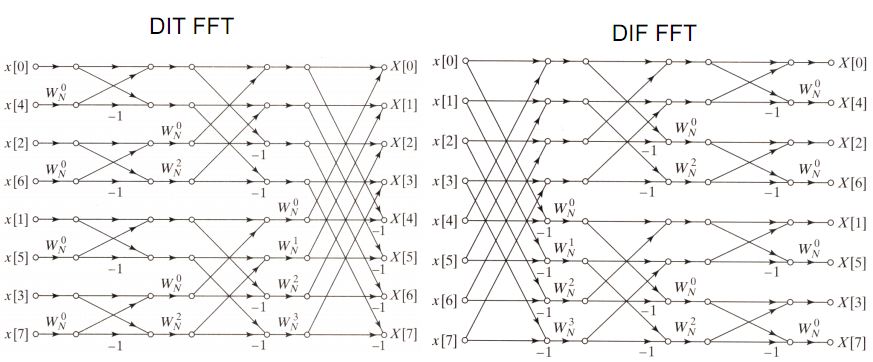
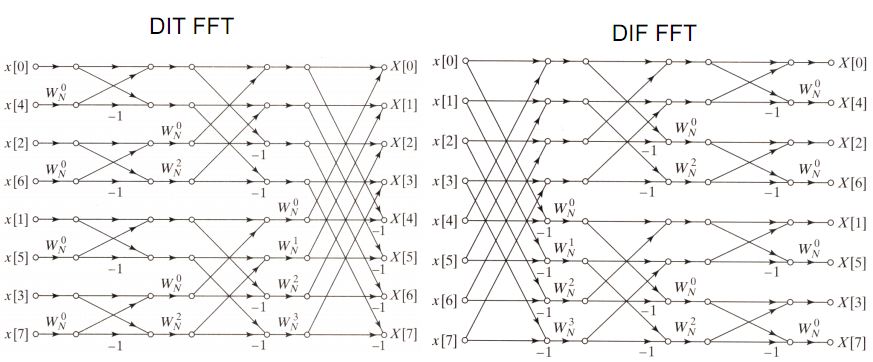


Figure 18, DIF-FFT block Diagram



*Figure 19, Radix-2 Decimation in Time (DIT) Domain FFT Algorithm.*



*Figure 20, Radix-2 Decimation in Time (DIF) Domain FFT Algorithm.*

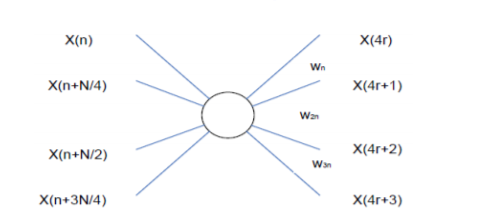
1. ***Radix-4***

Radix-4 is another FFT algorithm which was surveyed to improve the speed of functioning by reducing the computation; this can be obtained by change the base to 4. For a same number if base increases the power/index will decreases. For radix-4 the number of stages is reduced to 50% since N=43 (N=4M) i.e. only 3 stages. Radix-4 is having four inputs and four outputs and it follows in-place algorithm. The radix-4 DIT-FFT recursively partitions a DFT into four quarter-length DFTs of groups of every fourth time sample. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix-4 FFTs require only 75% as many complex multiplications as the radix-2 FFTs. The radix-4 decimation-in-time and decimation-in-frequency Fast Fourier transforms (FFTs) gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs.

In Radix-4 FFT Algorithms N points input signal are derived like the following equations: -

1. *WN0 \* WN/4kn*
2. *WNn \* WN/4kn*
3. *WN2n \* WN/4kn*
4. *WN3n \* WN/4kn*

Where k = 0, 1 ----N/4 – 1 and used the property WN4kn = *WN/4kn*, X (4r), X (4r+1), X (4r+2), and X (4r+3) are N/4- point DFTs. Every N/4 output is a sum of four input samples all multiplied by -1, j, or +1 –j. **Twiddle Factors** are multiplied by above sum. Every N/4-sample DFTs is divided into four N/16- sample DFTs. Every N/16 DFT is distributed further in four N/64-Pt.and so on.



*Figure 21, Radix 4 FFT algorithm*

A basic radix-4 Processing element (Butterfly) is represented in Figure 3. These are the expression of Radix 4 FFT algorithms. The radix 4 Butterfly holds 3 complex multiplications and 12 complex additions N/4 butterfly involves in each stage and number of stages is *log(4N)* for N-point sequence. Therefore, the number of complex multiplications is *3N / 4log(4N*) and number of complex additions is *12N / log(4N).*

**In comparison of radix 2 FFT**, number of complex multiplications are reduced by 25% but number of complex additions are increased by 50%.

1. ***Cooley-Tukey Algorithms***

Cooley-Tukey is an algorithm that efficiently computes the DFT and reduces the complexity of the DFT. It was introduced by Gauss, but it was not recognized at that time. In 1965 Cooley and Tukey published a paper regarding this algorithm and explained how to perform it on the computer. At that time because digital computers were growing and there was a need to compute the DFT fast, this algorithm got recognized. It uses butterfly method to compute the FFT. The Cooley–Tukey (C-T) set of algorithms comprise the most common fast Fourier transform (FFT) algorithms. They re-express the discrete Fourier transform (DFT) of an arbitrary composite size N = N1N2 in terms of smaller DFTs of sizes N1 and N2, recursively, to reduce the computation time to O(Nlog2(N)) for highly-composite N (smooth numbers). Usually, either N1 or N2 is a small factor (not necessarily prime), called the radix (which can differ between stages of the recursion). If N1 is the radix, the Cooley–Tukey algorithm is called decimation in time (DIT), whereas if N2 is the radix, it is decimation in frequency (DIF). A radix-2 DIT FFT is the simplest and most generic form of the Cooley–Tukey algorithm, although highly optimized Cooley–Tukey implementations generally use some other forms of the algorithm. Radix-2 DIT decomposes a DFT of size N into two interleaved DFTs (hence the name "radix-2") of size N/2 with each recursive stage, eventually resulting in a combining stage containing only size-2 DFTs called "butterfly", operations (so-called because of the shape of the data-flow diagrams, as shown in figure 3.

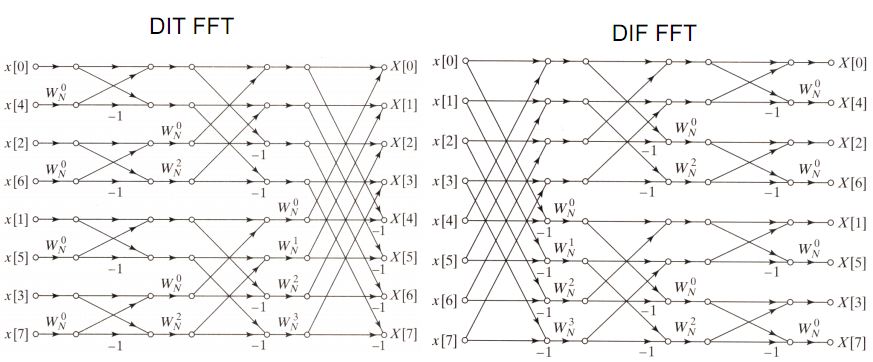


Figure 22. Flow graph of 8-point DIT FFT radix-2 Cooley–Tukey algorithm.

Besides the radix -2 Cooley- Tukey algorithm, other implementations with radixes of 4 and 8 are also used. Matter of fact, the value of the radix (2, 4, 8) shows that the total number of points used for the transformation can be expressed as 2x, 4x or 8x, accordingly. Therefore, the C-T algorithm can execute parallel and independent butterfly operations with 2, 4 or 8 input/output values, in each of the algorithm phases (for radix-(x), the number of phases is logx (N)).

Mixed-radix (also called split-radix) algorithms work by factorizing the data vector into smaller lengths. These can then be transformed by FFTs with small number of points, noted as small-N FFT [24]. Typical programs include FFTs for small prime factors, such as 2, 3 or 5, which are highly optimized. Actually, the idea of this algorithm is to use many multiplied small-N FFT modules and combine them in order to make longer transforms. If the small -N modules are supplemented by an *O(N2)* general-N module then an FFT of any length can be computed. Of course, any lengths which contain large prime factors would perform only as *O(N2)*.

The well-known radix-2 Cooley–Tukey algorithm is a simplified version of the mixed-radix algorithm, realized using FFT modules whose lengths are only power of two. Radix-2 algorithms have been the subject of much research into perfecting the FFT. Many of the most efficient radix-2 routines are based on the “split-radix” algorithm. This is a hybrid which combines the best parts of both radix-2 ("power of 2") and radix-4 ("power of 4") algorithms, for computing distinctive partitions of the Fourier’s transformation.

The idea is coming from splitting an N point DFT to two N/2-point DFT. One is performed over odd samples and the other one over even sample. Expressions below show how splitting the DFT to two N/2-point DFTs will reduce the complexity of the computation.

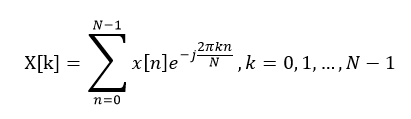


Figure 23. DFT Formula

As shown in figure 4., the mathematical representation of Discrete-Fourier Transform (DFT) goes from n=0 until n=N-1 points of summation which are the main part, and it has sub-component of multiplications that starts from K=0 until K=N-1.

This is the traditional DFT formula and we need N complex multipliers, and N-1 complex adds to compute that for each k, so for all N samples it will be O (N2) complexity. But with FFT we will get to the O (N log2N) complexity which is a significant difference in big N values. To start we define WN as and split out N points to two N/2 points and calculate the DFT on each one. So, we will have: -

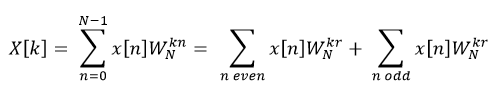


Figure 24. DFT Formula divided into even and odd components’ indices

Then we replace even ones with n = 2r and odd ones with n = 2r + 1, r = 0, 1, …, N/2 -1 and we will have: -

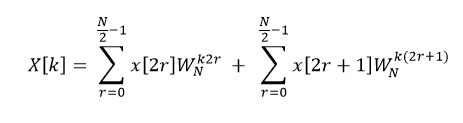


Figure 25. even and odd DFT Formula with n=2r

We will factor out the terms of **W** that does not depends on the **r**: -

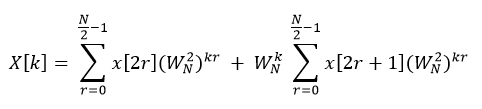


Figure 26. Factorized even and odd DFT Formula with n=2r

Based on the W features, we know that (WN)2 = = = WN/2, by using that we will have: -

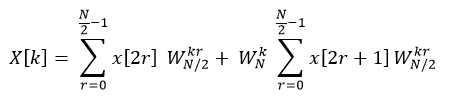


Figure 27. even and odd DFT Formula with WN/2

So, X[k] = Xe [K] + WNK Xo [K] and the complexity is going to be O (N2 /2 + N). Figure 8 shows how to split the N samples to two groups and then combine them back to generate the whole N point DFT: -

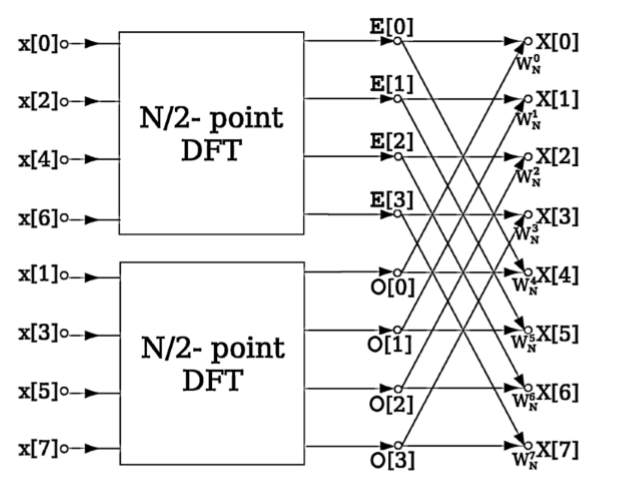


Figure 28. Splitting N point DFT to two N/2-point DFTs

To get the most efficient method we will split the sample points till we get to the 2-point FFT. To get there we need to split it log2N times. Figure 9 shows the Cooley Tukey splitting algorithm for an 8-point input. The order of the samples at input after the splitting is based on bit reverse order. It means when we stand for the index value of the sample input in binary and then reverse the bits, you will find the location of that sample. Like 4 is 0100 and when we reverse it, we will get 0010 which will be 2, or 6 (0010) will be 0100.

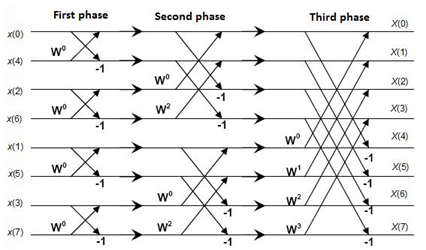
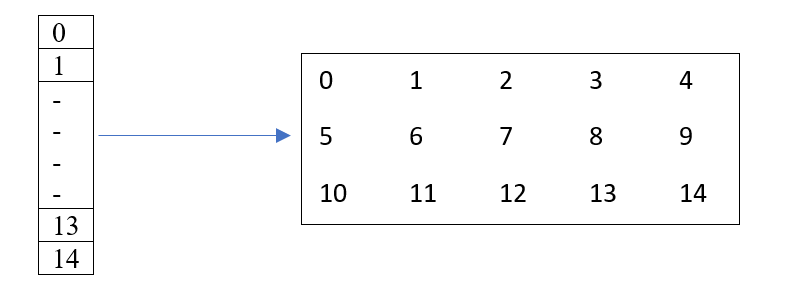


Figure 29. Cooley Tukey splitting for 8-point DFT

For more illustration, In Cooley-Tukey: -

If N=15 🡪 N=n1 \* n2; let n1 = 5, n2 = 3; where N=N1\*N2 and N1<=N2. The following steps are continued: -

**1)- Reshape the 1D vector into 2D Matrix**

****

2)- **2)- 5 DFTs of length 3 (**Column wise)



3)- **Multiply the 3x5 Matrix by an 3x5 kernel Matrix (Twiddle Factor!!)**

**\* WNbj**

4)**- 3 DFTs of length 5 (**Row wise)



5)- **Finally, reshape the o/p array from a 2D Matrix 🡪 1D o/p vector**

.

**Iv. The Prime Factor Algorithm (PFA) “Good-Thomas FFT algorithm”**

The prime-factor algorithm (PFA), also called the Good–Thomas algorithm (1958/1963), is a fast Fourier transform (FFT) algorithm that re-expresses the discrete Fourier transform (DFT) of a size N = N1\*N2 as a two-dimensional N1×N2 DFT, but only for the case where N1 and N2 are relatively prime. Furthermore, 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 = N1\*N2 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. A thorough derivation of the family of prime factor FFT algorithms has been given by Burrus

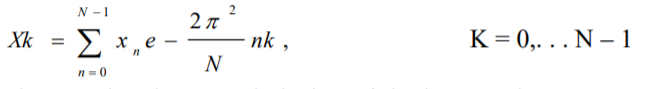
Briefly, the DFT of length N is defined by: -



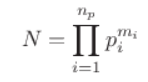
where *x(n)* and *z(k)* are complex, and we use the notation



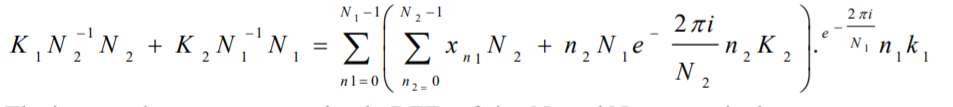
Therefore: -



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 Pi raised to an integer power mi > 1: -



The inner and outer sums are simply DFTs of size N2 and N1, respectively. We have used the fact that 1 1 N 1 N − is unity when evaluated modulo N2 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 Xn with DFT of odd-indexed part of X



To illustrate the derivation of the PFA with an example. Suppose that *N= N1\* N2*, where *N1* and *N2* are mutually prime. we can find integers *p, q, r, s (0 < p < N, 0 < q < N2, 0 < r < N2, 0 < s < N)* such that: -



We would have to use this **"Chinese Remainder Theorem" (CRT)** to define a mapping between the integers ***n, k*** *(0 ≤ n ≤ N- 1, 0≤ k ≤ N- 1)* and the corresponding integer pairs (*ni, n2*) and (*kl, k2*) where *0 ≤ n 1≤ N1, 0≤ n2<N2 , 0≤ k < N1, 0≤ k2 <N2*



where p, q are defined in equation –

The inverse map is also defined as: -



According to Previous equations, we could obtain: -



**In contrast of the Cooley-Tukey (C-T) algorithm, The Prime-Factor algorithm fixed the Twiddle Factor issue.**

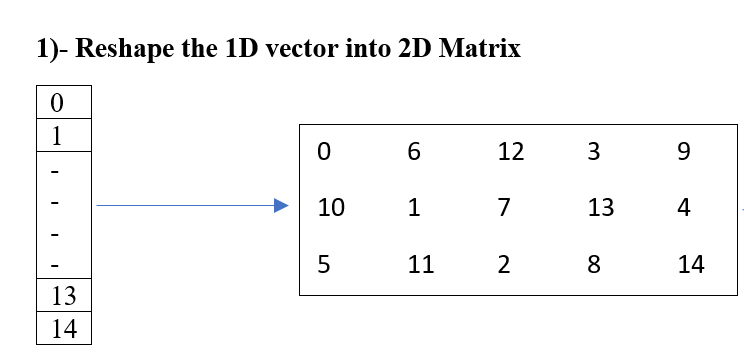
For simpler illustration: -

* If N=15 🡪 N=n1 \* n2; let n1 = 3, n2 = 5;
* It addresses the rule which says: - N=n1 \* n2; where n1, n2 must be Prime
* Relatively Prime: - N= (mo \* n1 + no \* n2) mod N;

*n= (i \* N2 \* n2 + j \* N1 \* n1) mod N*

*n = (10i -9j) mod 15 🡪 n = (10i +6j) mod 15*

**1)- Reshape the 1D vector into 2D Matrix**



**2)- 5 DFTs of length 3 (**Column wise)



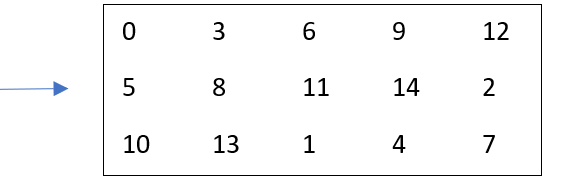
**3)- 3 DFTs of length 5 (**Row wise)



**4) -o/p 2D matrix resultant from these equations: -**

*K= (i \* K1 \* n2 + j \* k2 \* n1) mod N*

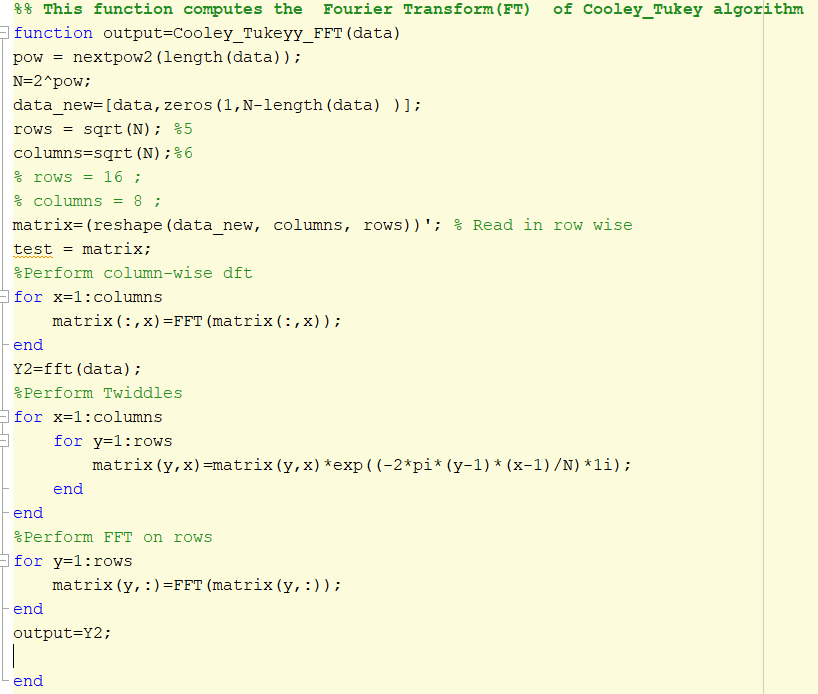
*K = (I \* 5k1 + j \* 3 k2) mod 15*

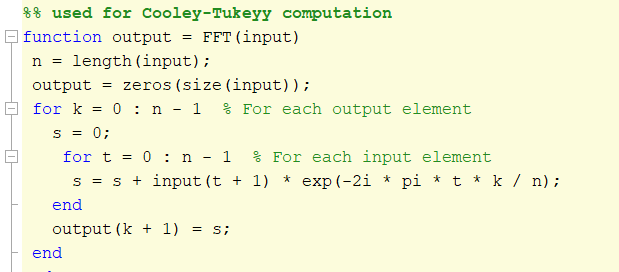


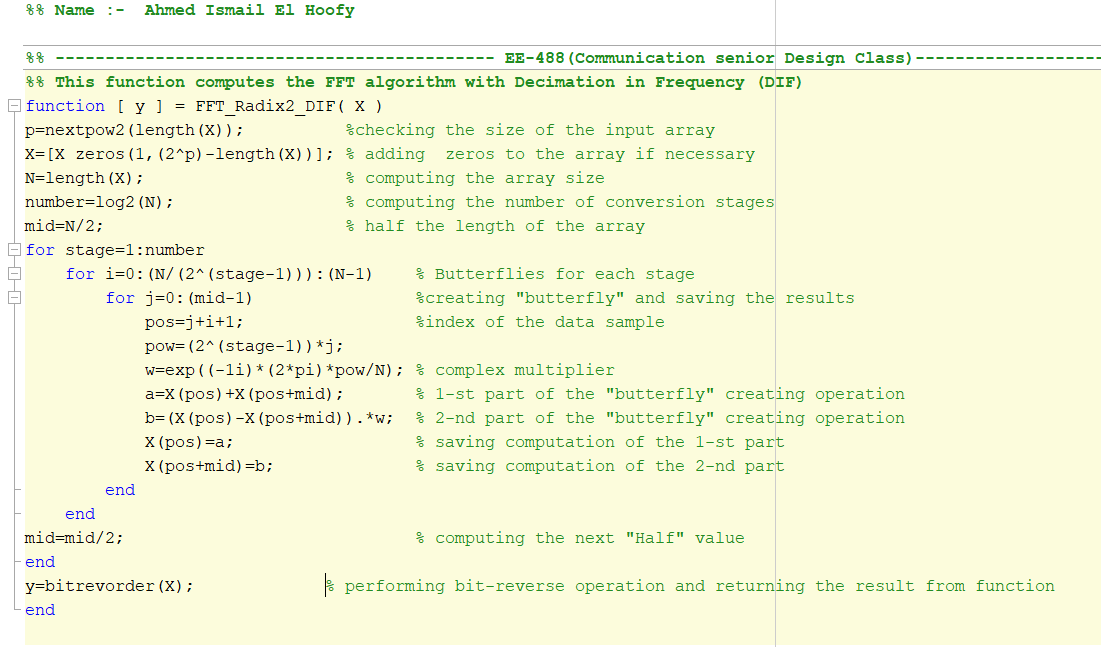
**5)- Finally, reshape the o/p array from a 2D Matrix 🡪 1D o/p vector**

Noticeably, we got rid of the Twiddle factor **WNbj** and this have saved so much computation time.

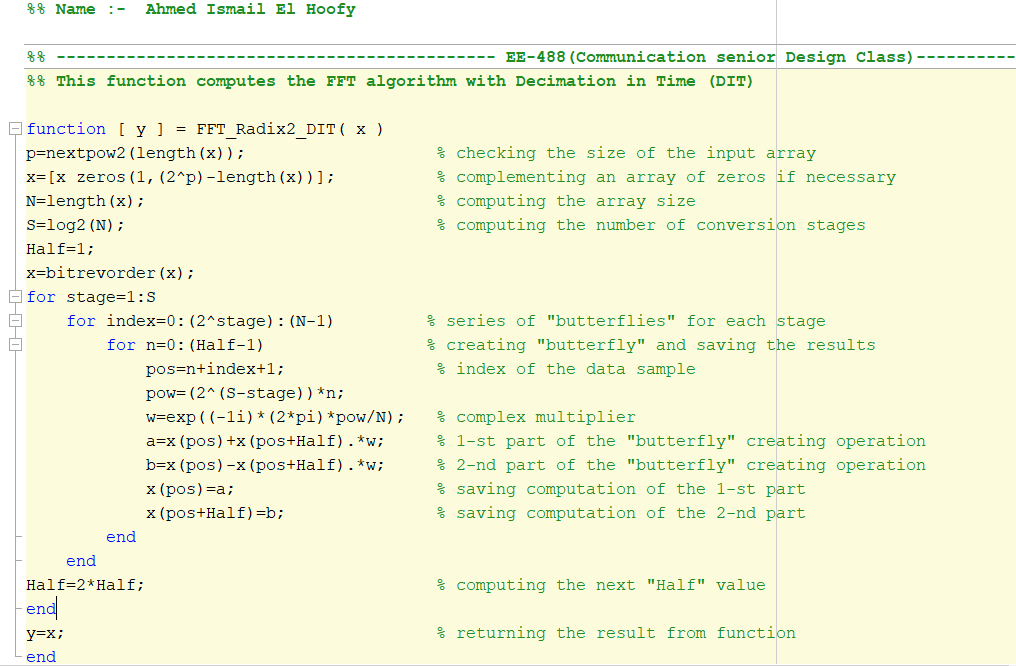
***IV. MATLAB Implementations***



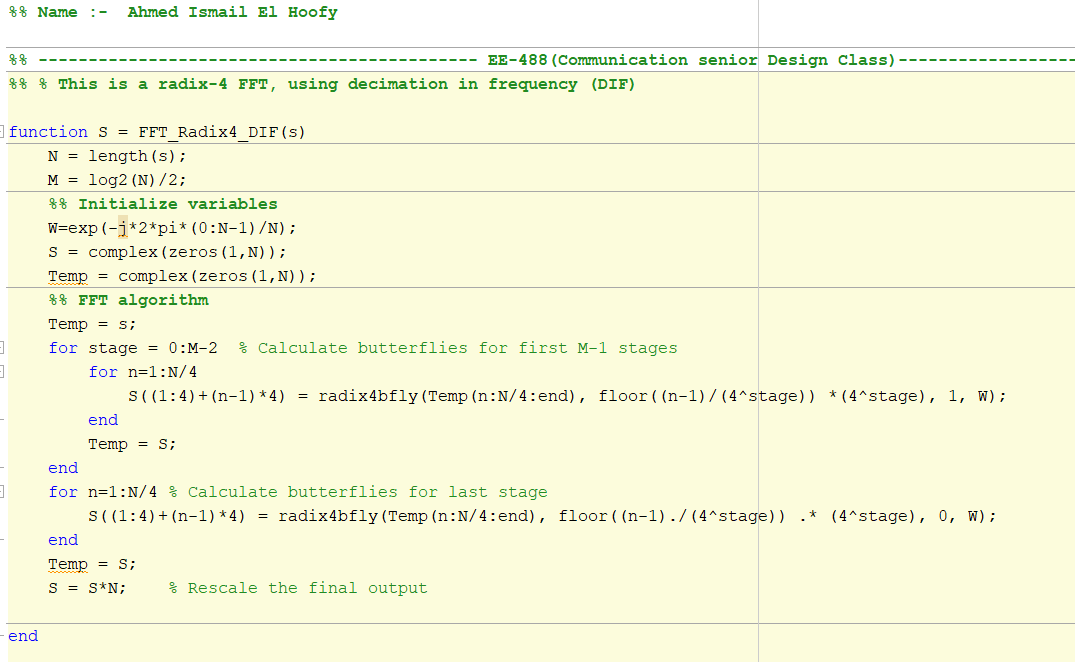
Fig*ure 30. MATLAB code for Cooley-Tukey algorithm*

*Figure 31. MATLAB code for a called function inside Cooley-Tukey algorithm code.*

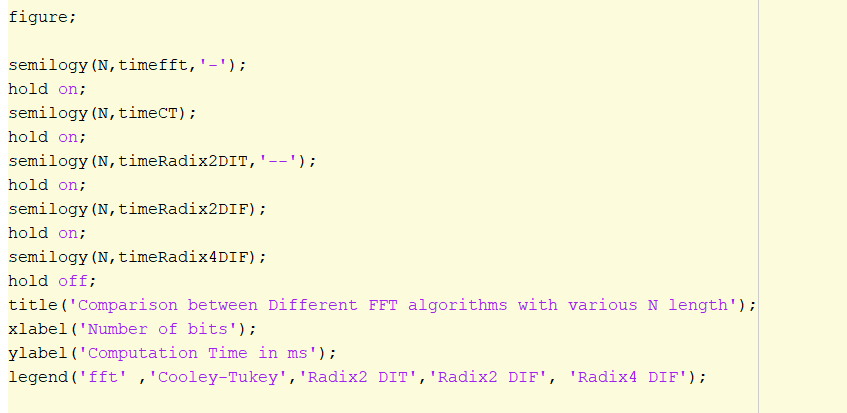
*Figure 32. MATLAB code for Radix2-DIF algorithm*



*Figure 33. MATLAB code for Radix4-DIT algorithm*

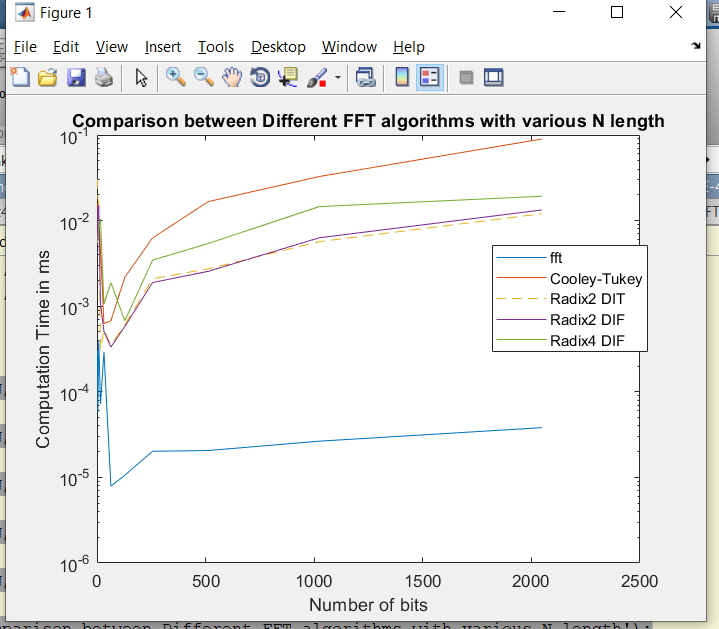


*Figure 34. MATLAB code for Radix4-DIF algorithm*

**

*Figure 35. MATLAB code for Plotting the computational time of each algorithm.*

# **Results**

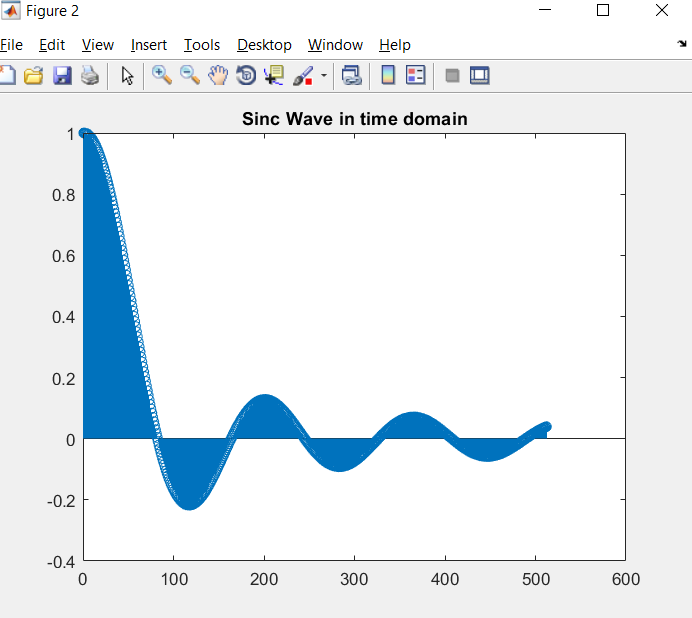


*Figure 35. MATLAB stem plot algorithms’ computation time*

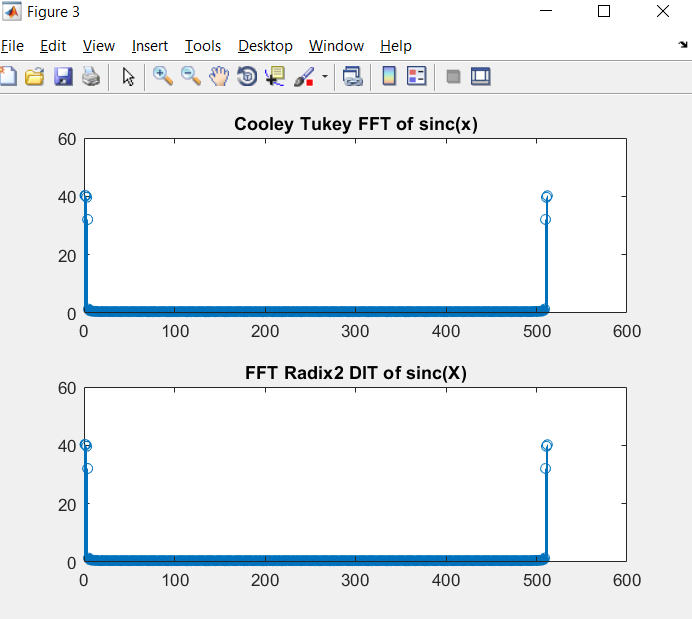
*Table.2 comparison between implemented algorithms according to their computational time*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | fft | C-T | Radix2 DIT | Radix2 DIF | Radix4 DIF |
| 2 | 0.0003244 | 0.0217646 | 0.0085196 | 0.0464582 | 0.0085216 |
| 4 | 6.31e-05 | 0.0111653 | 0.008993 | 0.0170315 | 0.0037754 |
| 16 | 0.000222 | 0.005911 | 0.0080122 | 0.0087032 | 0.004397 |
| 32 | 5.4e-05 | 0.003321 | 0.0009161 | 0.0002579 | 0.0093259 |
| 64 | 0.0002646 | 0.0005927 | 0.0005816 | 0.0005085 | 0.0010352 |
| 128 | 8.80e-06 | 0.000544 | 0.0003008 | 0.000303 | 0.0018238 |
| 256 | 9.6e-06 | 0.0019674 | 0.0005489 | 0.0005873 | 0.0006568 |
| 512 | 1.31e-05 | 0.0037609 | 0.0011209 | 0.0012531 | 0.0017302 |
| 1024 | 1.72e-05 | 0.0107849 | 0.0024177 | 0.0025 | 0.0033848 |
| 2048 | 2.4e-05 | 0.0324369 | 0.0056631 | 0.0053531 | 0.0091038 |

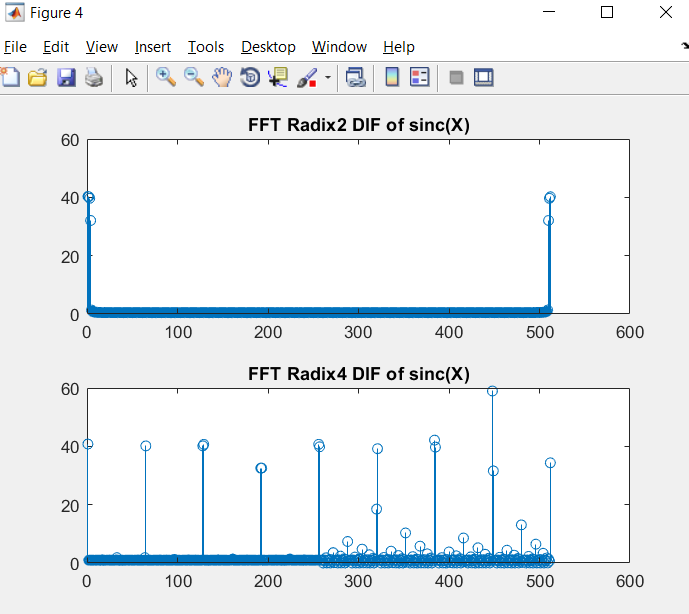
As shown in Figure 35 and Table 2, the Cooley-Tukey code takes higher time to compute a specific number of bits through Fourier Transform process. Additionally, this is compared to the other implemented algorithms, which shows that the best used algorithm is fft which is the built-in malt lab code and it is followed by Radix\_4 DIF which saves 25% of the time that is wasted in Radix\_2.



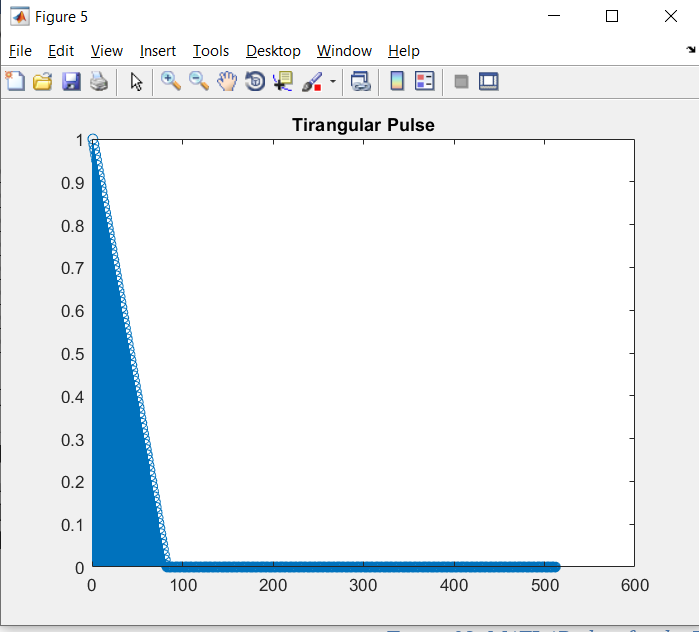
*Figure36. MATLAB plot of a sinc function in time domain with N=2048*



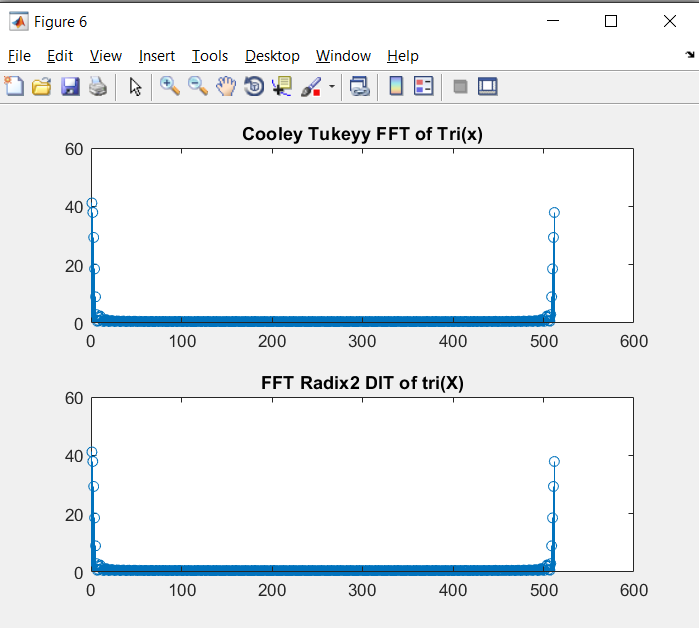
*Figure 37. MATLAB plots for the FT of the sinc function with Cooley Tukey and Radix 2 DIT algorithms, respectively.*



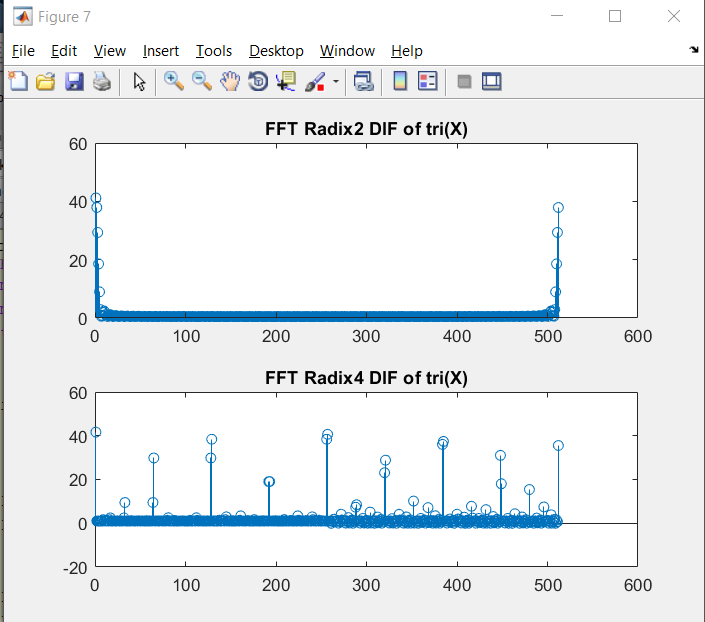
*Figure 38. MATLAB plots for the FT of the sinc function with Radix2 DIF and Radix4 DIF algorithms, respectively.*



*Figure39. MATLAB plot of a Triangular pulse function in time domain with N=2048.*

**

*Figure 40. MATLAB plots for the FT of the Triangular pulse function with Cooley Tukey and Radix 2 DIT algorithms, respectively.*

**

*Figure 40. MATLAB plots for the FT of the Triangular pulse function with Radix2 DIF and Radix4 DIF algorithms, respectively.*

Noticeably, the resultant graphs from either the FT of sinc function or the triangular pulse function, show a great response of each algorithm and how is that each algorithm is different from the other.

# **Conclusion**

# The never-ending aspiration for more efficient calculation of DFT, motivated by its truly widespread expansion in applications, could not neglect this opportunity. Although there is number of researches made in this area, as well as the number of diverse tracks and ideas within it, is certainly surprising. All the work done in this area proves that by making the right combination of FFT algorithm, planning and platform, desired performance results can be achieved. In general, all the efficient FFT implementations needs specific algorithms that is tailored to support the computations involved in the FFT algorithm used. In this paper we consider that most FFT algorithms include data dependent operations that limit the execution speed of the algorithm.

# Therefore, we suggest that the execution of the basic DFT/IDFT computations can supply execution speed-up, even though the DFT/IDFT involves more computations than many FFT algorithms. Considering that the basic DFT/IDFT computation is represented as summation of products, we propose improvements of the multiply and add operations over complex numbers. In our approach the multiplications involve only one add operation, since the operands are given in polar coordinate system. The additional cost that should be paid for this is the conversion to rectangular form, which involves two multiplications and calculation of sin and cos functions, using look-up tables. Furthermore, even though many researches dedicated their work on improving DFT calculating performance, we noticed a gap in examining pure DFT prospects for optimization and parallelization, where we recognize an enormous potential. For future continuing this work we plan to implement the proposed design expecting meaningful results

# **References**

[1] – F. Buchali, R. Dischler, A. Klekamp, M. Bernhard, and D. Efinger, “Realisation of a real-time 12.1 Gb/s optical OFDM transmitter and its application in a 109 Gb/s transmission system with coherent reception”, 35th European Conference on Optical Communication, Germany, 2009.

[2] – R. Schmogrow and M. Winter, et al, “101.5 Gbit/s real-time OFDM transmitter with 16QAM modulated subcarriers”, OSA/OFC/NFOEC 2011, vol. A247, pp. 529-551, April 1955.

[3] – C. Lin, Y. Yu, and L. Van, “A Low-power 64-point FFT/IFFT design for IEEE 802.11a WLAN application”, IEEE International Symposium on Circuits and Systems, Greece, 2006.

[4]- R. E. Blahut, “Fast Algorithms for Signal Processing,” United Kingdom: Cambridge University Press, 2010.