ISSN No:-2456-2165

2D FFT without using 1D FFT A PREPRINT

Vamshi Kumar Kurva
Department of Mathematics
Indian Institute of Space Science and Technology Valiyamala, Kerala

Abstract:- FFT for multi-dimensional input is usually obtained by applying FFT on each dimension. FFT algorithm has an asymptotic complexity of $O(N \log N)$. 2D FFT is especially important in the areas of image processing. Here we propose a new technique which can directly be applied on 2D image without using 1D FFT on rows and columns. It extends the concept of FFT to two dimensions. This too has an asymptotic complexity of $O(N \log N)$.

Keywords:- *Fourier Transform, DFT, FFT, 2D-FFT.*

I. INTRODUCTION

Fourier transform is a way of decomposing a given signal into a combination of sine and cosine waves with multiple frequencies. When the signal is continuous, we get a spectrum of frequencies that are densely packed and hence we get a Fourier spectrum, which is continuous indicating the strength of each frequency component present in the signal. When the signal is discrete it becomes Discrete Fourier Transform(DFT) and the spectrum is also discrete. DFT transforms N samples of a discrete signal to the same number of discrete frequency samples. Computing DFT of a signal directly has a complexity of $O(N^2)$. Since the computation of DFT is costly, several methods have been proposed to reduce the complexity.

Fast Fourier Transform(FFT) is an algorithm to compute DFT efficiently. It has an asymptotic complexity of $O(N \log N)$ and arithmetic complexity(Total number of additions and multiplications) of $O(3N/2 \log N)$ [James W. Cooley, 1965]. It greatly reduces the number of operations for very large N. FFT for 2-dimensional signals can be obtained by applying FFT on rows and then columns or vice versa. Since then several variations of FFT have been proposed. All of them have the complexity of $O(N \log N)$. The lower bound on the complexity of FFT algorithms is still an unsolved problem in computer science [Ailon, 2013]. Our proposed method also has asymptotic complexity of $O(N \log N)$ and arithmetic complexity of $O(3N \log N)$.

II. FFT IN ONE DIMENSION

Let $f(n)$ be an input signal, $n = 0, 1, 2, \ldots N-1, N = 2^m$.
Let $F(k)$ be the frequency spectrum of $f(n)$. Then

$$F(k) = \sum_{n=0}^{N-1} f(n)\, e^{-2\pi i\, kn/N} \tag{1}$$

Let W_N be a complex number, $W_N = e^{2\pi i/N}$, then

$$F(k) = \sum_{n=0}^{N-1} f(n)\, W_N^{kn} \tag{2}$$

ISSN No:-2456-2165

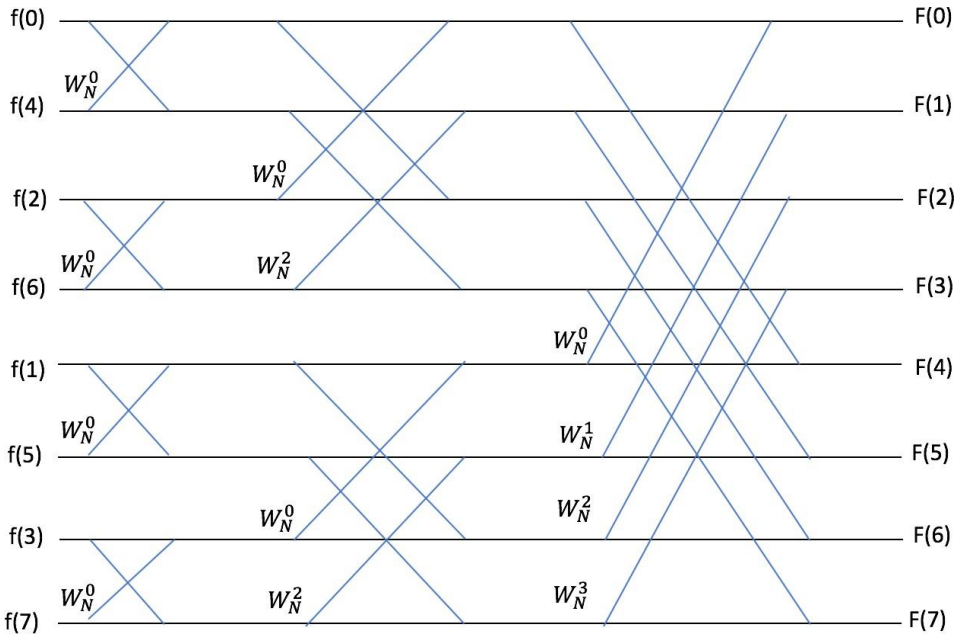

Fig 1:- 8-point DIT FFT algorithm

DFT requires N multiplications and $N-1$ additions for a single sample. So, the total number of operations required are $N(2N-1)$.

A. DIT-FFT

Decimation in Time FFT works by repeatedly decimating the signal into even and odd parts and then calculating the FFT for each sequence recursively.

$$
\begin{aligned}
F(k) &= \sum_{n=0}^{N-1} f(n)\, W_N^{kn} \\
&= \sum_{n=0}^{N/2-1} f(2n)\, W_N^{2kn} + \sum_{n=0}^{N/2-1} f(2n+1)\, W_N^{k(2n+1)} \\
&= \sum_{n=0}^{N/2-1} f(2n)\, W_{N/2}^{kn} + W_N^k \sum_{n=0}^{N/2-1} f(2n+1)\, W_{N/2}^{kn} \\
&= F_e(k) + W_N^k\, F_o(k)
\end{aligned}
$$

where $Fe(k)$ and $Fo(k)$ are DFTs of even and odd sequences of length $N/2$. Due to periodicity $Fe(k) = Fe(k + N/2)$, $Fo(k) = F0(k + N/2)$ for $k = 0, 1, 2..N/2\_1$. Decimate in time recursively until we are left with only sequences of length 1, for which the transform is same as the element.

Since $W_N^{k+N/2} = -W_N^k$, $F(k) = F_e(k) + W_N^k F_o(k)$ and $F(k+N/2) = F_e(k) - W^k{}_N F_o(k)$. i.e. in each stage it takes 1 multiplication and 2 additions to generate 2 outputs. Therefore the total number of operations required to calculate DFT are $3N/2 \log_2(N)$. Since we decimate the sequence recursively every time, the first stage of input is the bit reversal order and the output is in proper order. Figure 1 shows the 8-point DIT FFT algorithm and figure 2 showsthe basic computational diagram for DIT-FFT. Algorithm 1 shows how to generate the bit-reversal order for N-point sequence.

B. DIF-FFT

DIF-FFT breaks the sequence into first half and second half and finds the DFT recursively for each sub-sequence.

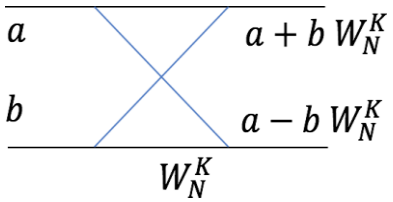

Fig 2:- Basic computational diagram for DIT FFT

Algorithm 1 1D bit reversal generator

```
1: procedure REVERSAL(N)
2:     Order = array[N]
3:     Order[0] ← 0
4:     Order[1] ← 1
5:     p ← 0
6:     i ← 1
7:     while i < log N do
8:         p ← p + 1
9:         l ← 2^p
10:        m ← 0
11:        while m < N do
12:            Order[m] ← 2□ Order[m]
13:            Order[m + l] ← Order[m] + 1
14:            m ← m + 1
15:        i ← i + 1
16:    return Order
```

$$F(k) = \sum_{n=0}^{N-1} f(n)\, W_N^{kn}$$

$$= \sum_{n=0}^{N/2-1} f(n)\, W_N^{kn} + \sum_{n=0}^{N/2-1} f\left(n+\frac{N}{2}\right) W_N^{k(n+\frac{N}{2})}$$

$$= \sum_{n=0}^{N/2-1} f(n)\, W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{N/2-1} f\left(n+\frac{N}{2}\right) W_N^{kn}$$

$$= \sum_{n=0}^{N/2-1} f(n)\, W_N^{kn} + e^{-i\pi k} \sum_{n=0}^{N/2-1} f\left(n+\frac{N}{2}\right) W_N^{kn}$$

When k is even $e^{-i\,\pi k} = 1$

ISSN No:-2456-2165

$$F(2k) = \sum_{n=0}^{N/2-1} f(n)\, W_N^{2kn} + \sum_{n=0}^{N/2-1} f(n+\frac{N}{2})\, W_N^{2kn}$$

$$= \sum_{n=0}^{N/2-1} (f(n)+f(n+\frac{N}{2}))\, W_{N/2}^{kn}$$

$$n=0$$

This is the $N/2$ point DFT of $N/2$ length sequence obtained by adding the first and last half of the input sequence. When k is odd $e^{-i\pi k} = -1$

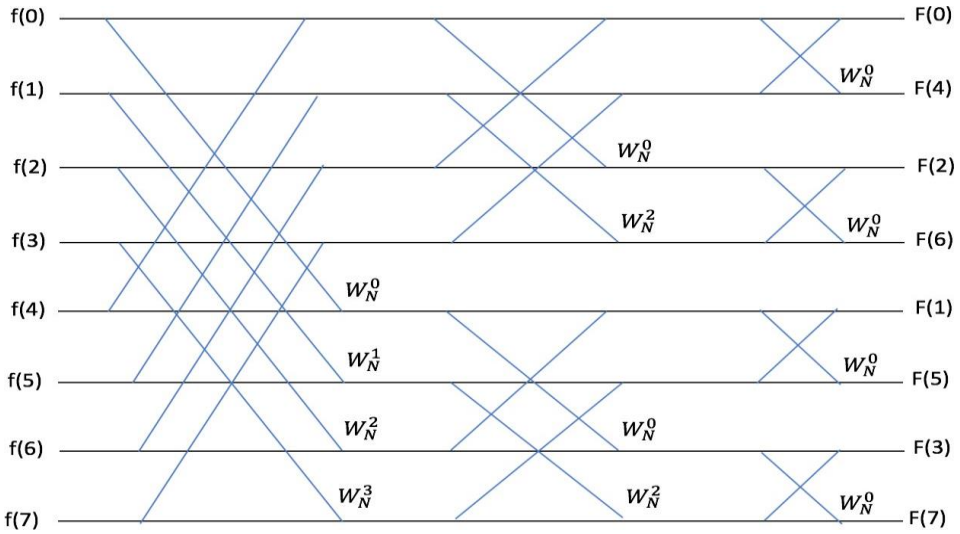

Fig 3:- 8-point DIF FFT algorithm

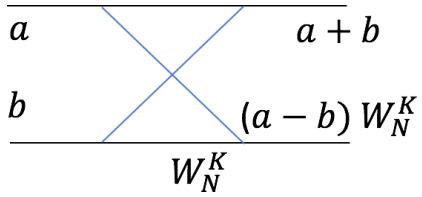

Fig 4:- Basic computational diagram for DIF FFT

$$F(2k+1) = \sum_{n=0}^{N/2-1} f(n)\, W_N^{n(2k+1)} - \sum_{n=0}^{N/2-1} f(n+\frac{N}{2})\, W_N^{2kn}$$

$$= \sum_{n=0}^{N/2-1} (f(n)-f(n+\frac{N}{2}))\, W_N^{2kn}\, W_N^n$$

$$= \sum_{n=0}^{N/2-1} (f(n)-f(n+\frac{N}{2}))\, W_N^n W_{N/2}^{kn}$$

$$n=0$$

This is the $N/2$ point DFT of the sequence obtained by subtracting the second half of the input sequence from the first half, and multiplying the resulting sequence by W_N^n. Here too in each stage it needs two additions and one multiplication for generating two outputs. So total number of operations required are $3N \log_2(N)$. Figure 3 shows the 8-point DIF FFT algorithm and figure 4 shows the basic computational diagram for DIF-FFT. The colour scheme is used here only to indicate the set of values being considered for processing at one time with same twiddle factor values.

III. DFT IN TWO DIMENSIONS

Let $f(m, n)$, $m = 0, 1, 2..M-1$, $n = 0, 1, 2..N-1$ be a two dimensional signal, then DFT is given by

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, e^{-2\pi i \left(\frac{mu}{M} + \frac{nv}{N}\right)} \qquad (3)$$

Let $W_M = e^{\frac{-2\pi}{M} i}$ and $W_N = e^{\frac{-2\pi}{N} i}$, then

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, W_M^{mu}\, W_N^{nv} \qquad (4)$$

A. 2D DIT-FFT

Let's ignore the constant factor of $1/MN$ while deriving the FFT for two dimensions.

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, W_M^{mu}\, W_N^{nv}$$

$$= \sum_{m=0}^{M/2-1} \sum_{n=0}^{N/2-1} f(2m,2n)\, W_M^{2mu}\, W_N^{2nv} + \sum_{n=0}^{N/2-1} f(2m,2n+1)\, W_M^{2mu}\, W_N^{v(2n+1)}$$

$$+ \sum_{m=0}^{M/2-1} \sum_{n=0}^{N/2-1} f(2m+1,2n)\, W_M^{u(2m+1)}\, W_N^{2nv} + \sum_{n=0}^{N/2-1} f(2m+1,2n+1)\, W_{M\,u(2m+1)}\, W_{N\,v(2n+1)} \qquad (5)$$

$$= \sum_{m=0}^{M/2-1} \sum_{n=0}^{N/2-1} f(2m,2n)\, W_{M/2}^{mu}\, W_{N/2}^{nv} + W_N^v f(2m,2n+1)\, W_{M/2}^{mu}\, W_{N/2}^{nv}$$

$$+ \sum_{m=0}^{M/2-1} \sum_{n=0}^{N/2-1} W_M^u f(2m+1,2n)\, W_{M/2}^{mu}\, W_{N/2}^{nv} + W_M^u W_N^v f(2m+1,2n+1)\, W_{M/2}^{mu}\, W_{N/2}^{nv}$$

Since $W_M^{u+M/2} = -W_M^u$ and $W_N^{v+N/2} = -W_N^v$

$$F(u,v) = \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(2m,2n) + W_N f(2m,2n+1) + W_M f(2m+1,2n) + W_M^u W_N^v f(2m+1,2n+1) \; W_{M/2}^{mu} W_{N/2}^{nv} \quad (6)$$

$$F(u,v+\frac{N}{2}) = \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(2m,2n) - W_N f(2m,2n+1) + W_M f(2m+1,2n) - W_M^u W_N^v f(2m+1,2n+1) \; W_{M/2}^{mu} W_{N/2}^{nv} \quad (7)$$

$$F(u+\frac{M}{2},v) = \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(2m,2n) + W_N f(2m,2n+1) - W_M f(2m+1,2n) - W_M^u W_N^v f(2m+1,2n+1) \; W_{M/2}^{mu} W_{N/2}^{nv} \quad (8)$$

$$F(u+\frac{M}{2},v+\frac{N}{2}) = \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(2m,2n) - W_N f(2m,2n+1) - W_M f(2m+1,2n) - W_M^u W_N^v f(2m+1,2n+1) \; W_{M/2}^{mu} W_{N/2}^{nv} \quad (9)$$

Since each basic computation element needs four elements, this method applies only when $M = N$. It takes 4 multiplications and 8 additions to output 4 values in each stage. So, total number of operations required are $3N\log_2(N)$. Since the input in decimated in time recursively, the actual input to the FFT will be the bit reversed order. Algorithm 2 shows how to generate bit reversal order in 2 dimensions. Figure 5 shows the 2 dimensional DIF FFT algorithm for a 4 4 signal and figure 6 shows the basic computational diagram for DIT-FFT in two dimensions. The colour scheme is used here only to indicate the set of values being considered for processing at one time with same twiddle factor values and the blue lines indicate the one-to-one correspondence of the values in each stage.

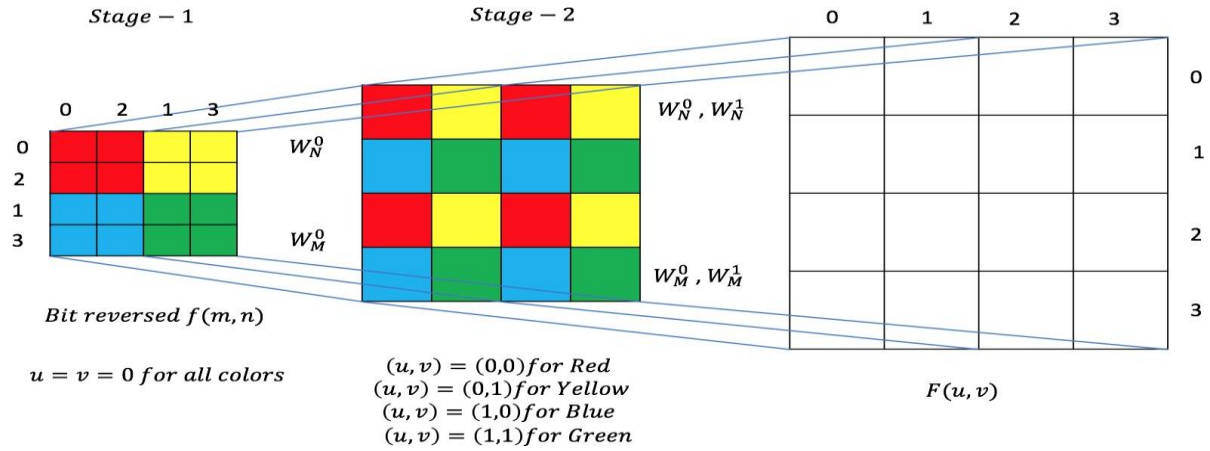

Fig 5:- DIT FFT algorithm for 4x4 signal

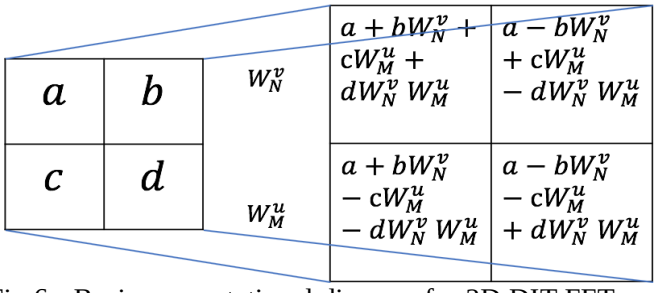

Fig 6:- Basic computational diagram for 2D DIT FFT

Algorithm 2 2D bit reversal generator

```
1: procedure 2D-REVERSAL(N )
2:     Order = array[N][N]
3:     Rever ← REVERSAL(N )                    d 1D bit reversal order
4:     i ← 0
5:     j ← 0
6:     while i < N do
7:         while j < N do
8:             Order [i][j] ← (Rever [i], Rever [j])
9:             j ← j +1
10:        i ← i +1
11:    return Order
```

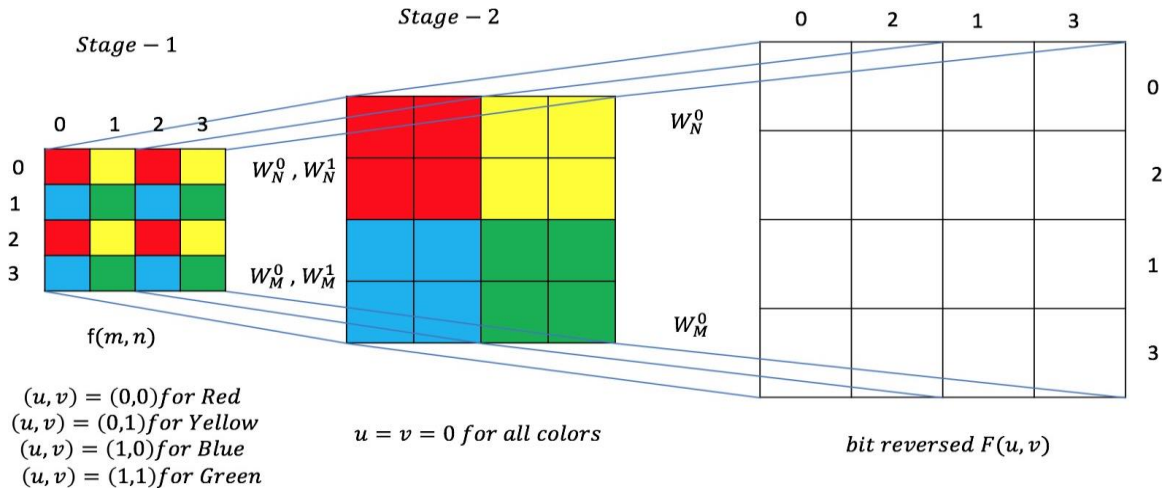

Fig 7:- DIF FFT algorithm for 4x4 signal

B. 2D DIF-FFT

$$F(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\, W_M^{mu}\, W_N^{nv}$$

$$= \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)\, W_M^{mu}\, W_N^{nv} + \sum_{n=N/2}^{N-1} f(m,n)\, W_M^{mu}\, W_N^{vn}$$

$$+ \sum_{m=M/2}^{M-1}\sum_{n=0}^{N/2-1} f(m,n)\, W_M^{um}\, W_N^{nv} + \sum_{m=M/2}^{N-1}\sum_{n=N/2} f(m,n)\, W_{um}\, W_{vn}$$

$$= \sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)\, W_M^{mu}\, W_N^{nv} + f\left(m, n+\frac{N}{2}\right) W_M^{mu}\, W_N^{v(n+\frac{N}{2})}$$

$$+\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m+\frac{M}{2},n)\,W_M^{u(m+\frac{M}{2})}\,W_N^{nv}+f(m+\frac{M}{2},n+\frac{N}{2})\,W_{M_{u(m+\frac{M}{2})}}^{2}\,W_{N_{v(n+\frac{N}{2})}}^{2}$$

$$=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)+f(m,n+\frac{N}{2})\,W_N^{N/2}+f(m+\frac{M}{2},n)\,W_M^{M/2}+f(m+\frac{M}{2},n+\frac{N}{2})\,W_N^{N/2}\,W_M^{M/2}{}^{um}\,{}^{vn}$$

$$=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)+f(m,n+\frac{N}{2})\,e^{-i\pi v}+f(m+\frac{M}{2},n)\,e^{-i\pi u}+f(m+\frac{M}{2},n+\frac{N}{2})\,e^{-i\pi(u+v)}\,W_M^{um}\,W_N^{vn} \quad (10)$$

when u is odd, $e^{-i\,\pi u}=-1$ and when u is even $e^{-i\,\pi u}=1$

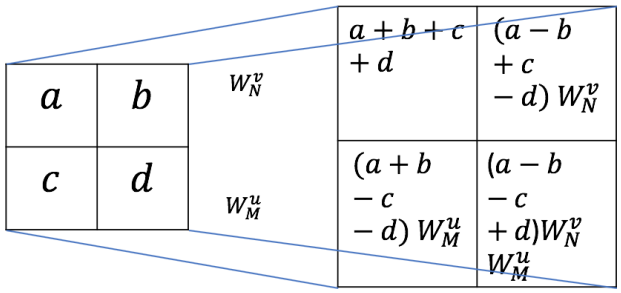

Fig 8:- Basic computational diagram for 2D DIF FFT

$$F(2u,2v)=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)+f(m,n+\frac{N}{2})+f(m+\frac{M}{2},n)+f(m+\frac{M}{2},n+\frac{N}{2})\,W_{M/2}^{mu}\,W_{N/2}^{nv} \quad (11)$$

$$F(2u,2v+1)=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)-f(m,n+\frac{N}{2})+f(m+\frac{M}{2},n)-f(m+\frac{M}{2},n+\frac{N}{2})\,W_N^{n}\,W_{M/2}^{mu}\,W_{N/2}^{nv} \quad (12)$$

$$F(2u+1,2v)=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)+f(m,n+\frac{N}{2})-f(m+\frac{M}{2},n)-f(m+\frac{M}{2},n+\frac{N}{2})\,W_M^{m}\,W_{M/2}^{mu}\,W_{N/2}^{nv} \quad (13)$$

$$F(2u+1,2v+1)=\sum_{m=0}^{M/2-1}\sum_{n=0}^{N/2-1} f(m,n)-f(m,n+\frac{N}{2})-f(m+\frac{M}{2},n)+f(m+\frac{M}{2},n+\frac{N}{2})\,W_M^{m}\,W_N^{n}\,W_{M/2}^{mu}\,W_{N/2}^{nv} \quad (14)$$

In this case the input sequence is in proper order and the output will be bit-reversed. This also involves 4 multiplications and 8 additions to get 4 values in each stage, hence the total number of required operations are $3N\log_2(N)$. Figure 7 shows the DIF FFT algorithm for a 4 4 signal, and figure 8 shows the basic computational diagram for DIF-FFT in two dimensions.

ISSN No:-2456-2165

IV. CONCLUSION AND FUTURE WORKS

This work is done to extend the concept of FFT to two-dimensions to see if we can gain any computational advantage. To our knowledge this is the first paper to do this analysis. It's concluded that it offers the same computational complexity as that of applying 1D-FFT on rows and columns. Although the effects of caching, speed and other implementation effects are not verified, the code is implemented in c++ and is available on *https://github.com/vamshikumarkurva/2-Dimensional-FFT* .

ACKNOWLEDGEMENTS

Thanks to Sai Subrahmanyam Gorthi, Litu Rout for valuable comments and feedback, and Nir Ailon for endorsing me on arXiv.

REFERENCES

[1]. Nir Ailon. A lower bound for fourier transform computation in a linear model over 2x2 unitary gates using matrix entropy. *arXiv:1305.4745v1*, 2013.
[2]. John W. Tukey James W. Cooley. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation, JSTOR*, 1965.