Fast Fourier Transform (FFT)

FFT proposed by Cooley & Tukey in 1965. The FFT is a highly efficient procedure for computing the DFT of a finite series and requires less no. of computations than that of direct evaluation of DFT. It reduces the computation by taking advantage of the fact that the calculation of the coefficient of the DFT can be carried out iteratively. Due to this, FFT computation technique is used in digital spectral analysis, filter simulation, auto-correlation & pattern recognition.

The FFT is based on decomposition & breaking of the transform into smaller transforms & combining them to get the total transform. FFT reduces the computation time required to compute a discrete Fourier transform & improves the performance by a factor 100 or more over direct evaluation of DFT.

* Why FFT is needed?

The direct evaluation of DFT using the formula $X(k) = \sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi nk}{N}}$ required N^2 complex multiplications and $N(N-1)$ complex addition. Thus for reasonably large values of N (in order of 1000) direct evaluation of the DFT requires an inordinate amount of computation. By using FFT algorithms the no. of computation can be reduced. For ex. for a N pt DFT, the no. of complex multiplication required using FFT is $\frac{N}{2}\log_2 N$. If $N = 16$ the no. of complex multiplication required for direct evaluation of DFT is 256, whereas by using DFT only 32 multications are required.

For $N = 32$ no. of complex multiplications is equal to $\frac{32}{2}\log_2 32 = 16 \times 5 = 80$

* What is FFT

FFT is an algorithm used to compute the DFT. It makes use of symmetry & periodicity properties of twiddle factor W_N^k to effectively reduce the DFT computation time. It is based on the fundamental principle of decomposing the computation of DFT of a sequence of length N into successively smaller discrete Fourier transform. The FFT algorithm provides speed increase factor, when compared with direct computation of the DFT of approximately 64 & 205 for 256 pt & 1024 pt transform respectively.

* What is decimation in time algorithm (DIT)

DIT algorithm is used to calculate the DFT of a N pt sequence. The idea is to break the N pt sequence into two sequences, the DFTs of which can be combined to give the DFT of the original N pt sequence. Initially the N pt sequence is divided into two $\frac{N}{2}$ pt sequences $x_e(n)$ & $x_o(n)$ which have even & odd members of $x(n)$ respectively. The $\frac{N}{2}$ pt DFTs of these two sequences are evaluated and combined to give the N pt DFT. Similarly the $N/2$ pt DFTs can be expressed as a combination of $N/4$ pt DFTs. This process is continued until we are left with 2 pt. DFT. This algorithm is called decimation in Time because the sequence $x(n)$ is often split into smaller subsequences.

FAST FOURIER TRANSFORM

* Decimation in Time FFT Algorithm (DIT - FFT)

DFT of an N point sequence is given by

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

let $x(n) = x_e(n) + x_o(n)$

$x_e(n) = x(2n)$; $n = 0, 1, \ldots \frac{N}{2}-1$

$x_o(n) = x(2n+1)$; $n = 0, 1, \ldots \frac{N}{2}-1$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\, W_N^{(2n+1)k}$$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\, W_N^{2nk}\, W_N^{k}$$

Now $W_N = e^{-j\frac{2\pi}{N}}$

$W_N^{2nk} = e^{-j\frac{2\pi 2kn}{N}} = e^{-j\frac{2\pi nk}{N/2}} = W_{N/2}^{nk}$

Substituting

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, W_{N/2}^{nk} + W_N^{k} \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\, W_{N/2}^{nk}$$

let

$x(2n) = g(n)$

$x(2n+1) = h(n)$

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} g(n)\, W_{N/2}^{nk} + W_N^{k} \sum_{n=0}^{\frac{N}{2}-1} h(n)\, W_{N/2}^{nk}$$

$$\therefore \boxed{X(k) = G(k) + W_N^{k} H(k)}$$

$G(k) = DFT\{x(2n)\}$ $\quad H(k) = DFT\{x(2n+1)\}$

FFT N = 2

$$X(k) = \sum_{n=0}^{1} x(n)\, W_2^{nk}$$

$$X(0) = \sum_{n=0}^{1} x(n)\, W_2^{n\cdot 0} = x(0)\, W_2^{0} + x(1)\, W_2^{0}$$

$$X(1) = \sum_{n=0}^{1} x(n)\, W_2^{n\cdot 1} = x(0)\, W_2^{0} + x(1)\, W_2^{1}$$

Butterfly

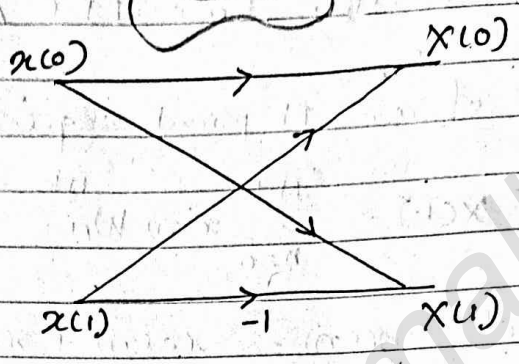
$W_2^l = e^{-j\frac{2\pi l}{2}} = e^{-j\pi l}$

$l=0 \quad W_2^0 = e^0 = 1$

$l=1 \quad W_2^1 = e^{-j\pi} = -1$

$\therefore X(0) = x(0) + x(1)$

$X(1) = x(0) - x(1)$



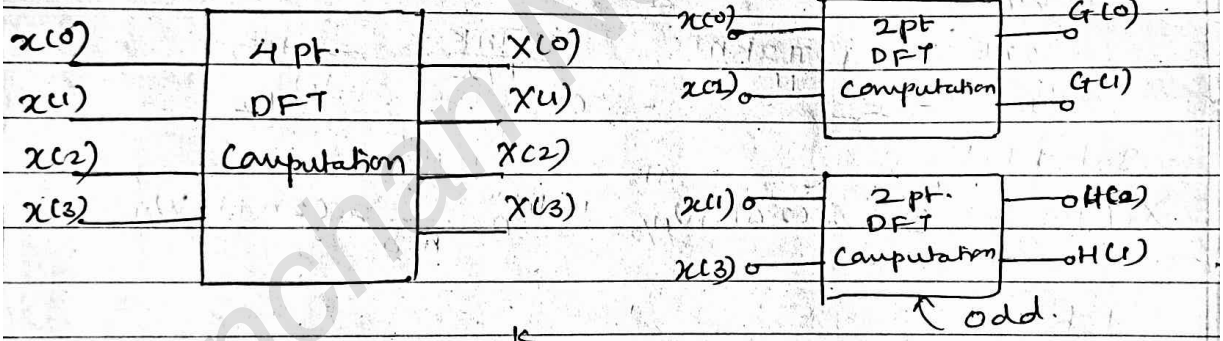
Butterfly Diagram.

N = 4

$X(k) = G(k) + W_4^k H(k)$

$G(k) = DFT\{x(2n)\} = DFT\{x(0), x(2)\}$

$H(k) = DFT\{x(2n+1)\} = DFT\{x(1), x(3)\}$



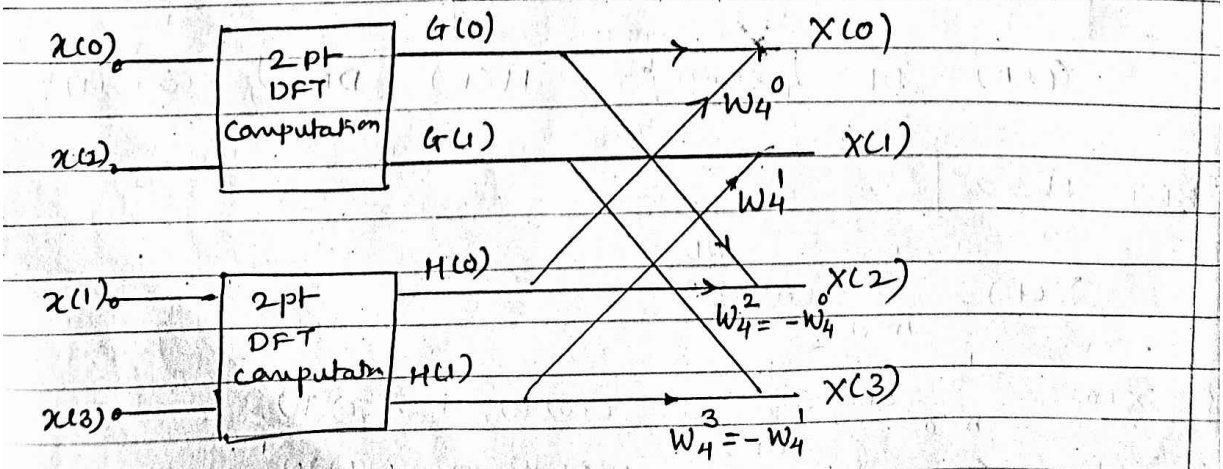
$X(k) = G(k) + W_4^k H(k)$

$X(0) = G(0) + W_4^0 H(0)$

$X(1) = G(1) + W_4^1 H(1)$

$X(2) = G(2) + W_4^2 H(2) = G(0) + W_4^2 H(0)$

$X(3) = G(3) + W_4^3 H(3) = G(1) + W_4^3 H(1)$



$$G(K) = \sum_{n=0}^{1} x(2n)\, W_2^{nK}$$

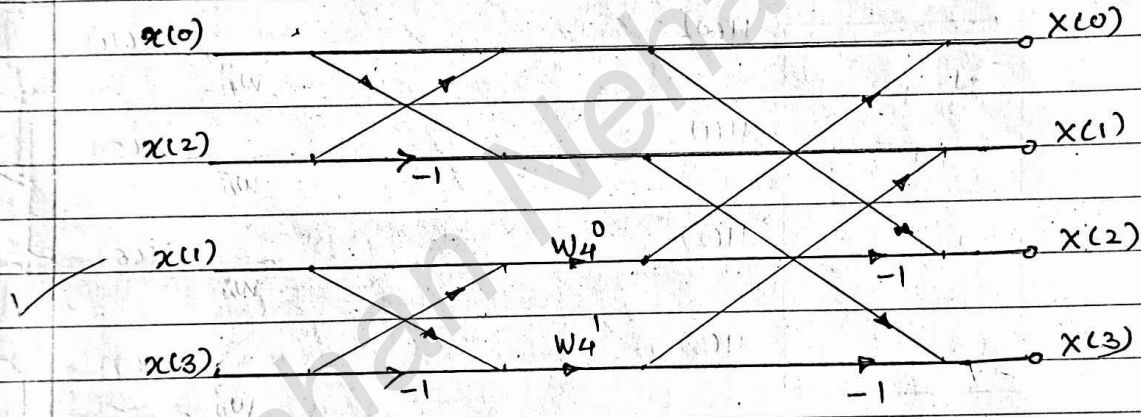
$$G(0) = \sum_{n=0}^{1} x(2n)\, W_2^{n.0} = x(0)\, W_2^{0} + x(2)\, W_2^{0}$$

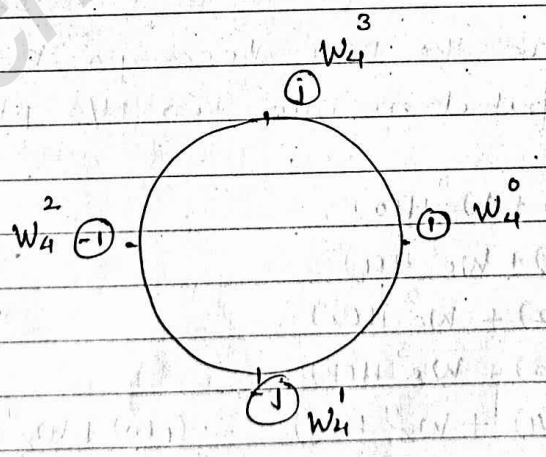
$$G(1) = \sum_{n=0}^{1} x(2n)\, W_2^{n.1} = x(0)\, W_2^{0} + x(2)\, W_2^{1}$$

$$H(K) = \sum_{n=0}^{1} x(2n+1)\, W_2^{nK}$$

$$H(0) = \sum_{n=0}^{1} x(2n+1)\, W_2^{0} = x(1)\, W_2^{0} + x(3)\, W_2^{0}$$

$$H(1) = \sum_{n=0}^{1} x(2n+1)\, W_2^{n.1} = x(1)\, W_2^{0} + x(3)\, W_2^{1}$$





N = 8

$$X(k) = G(k) + W_8^k H(k)$$
$$G(k) = L(k) + W_8^{2k} M(k)$$
$$H(k) = E(k) + W_8^{2k} F(k)$$

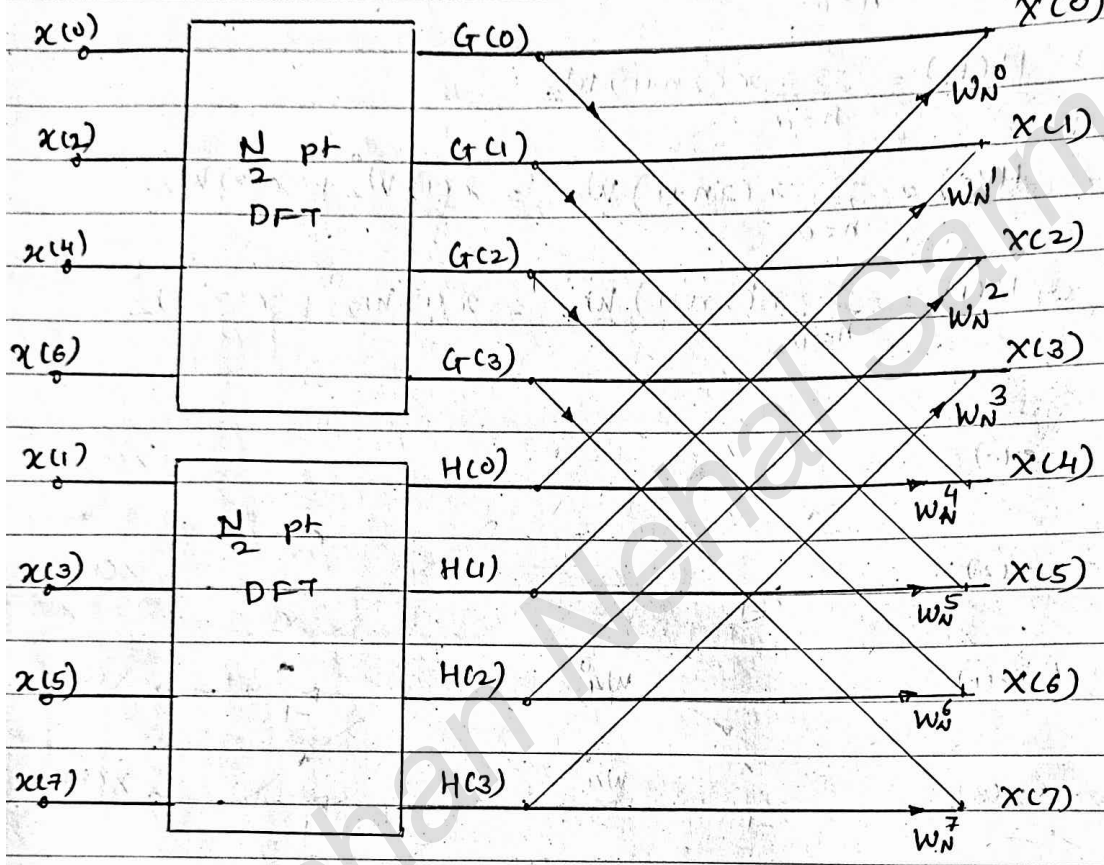


Fig. Flow graph of the DIT decomposition of an N-pt DFT computation into two N/2 pt DFT computations $^{N=8}$

$$X(0) = G(0) + W_8^0 H(0)$$
$$X(1) = G(1) + W_8^1 H(1)$$
$$X(2) = G(2) + W_8^2 H(2)$$
$$X(3) = G(3) + W_8^3 H(3)$$
$$X(4) = G(4) + W_8^4 H(4) = G(0) + W_8^4 H(0)$$
$$X(5) = G(5) + W_8^5 H(5) = G(1) + W_8^5 H(1)$$
$$X(6) = G(6) + W_8^6 H(6) = G(2) + W_8^6 H(2)$$
$$X(7) = G(7) + W_8^7 H(7) = G(3) + W_8^7 H(3)$$

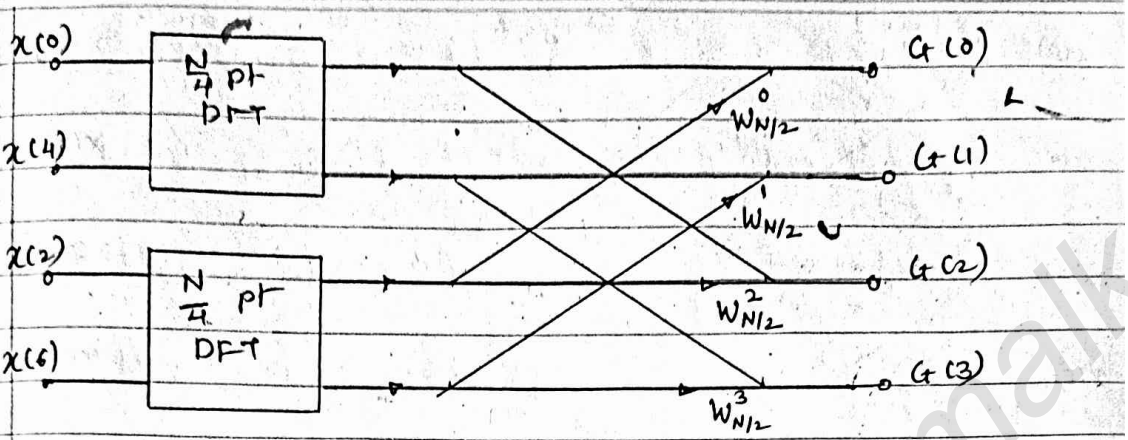


Fig ⑥ Flow graph of the DIT decomposition of an N/2 pt DFT computations into two N/4 pt DFT computation

N = 8

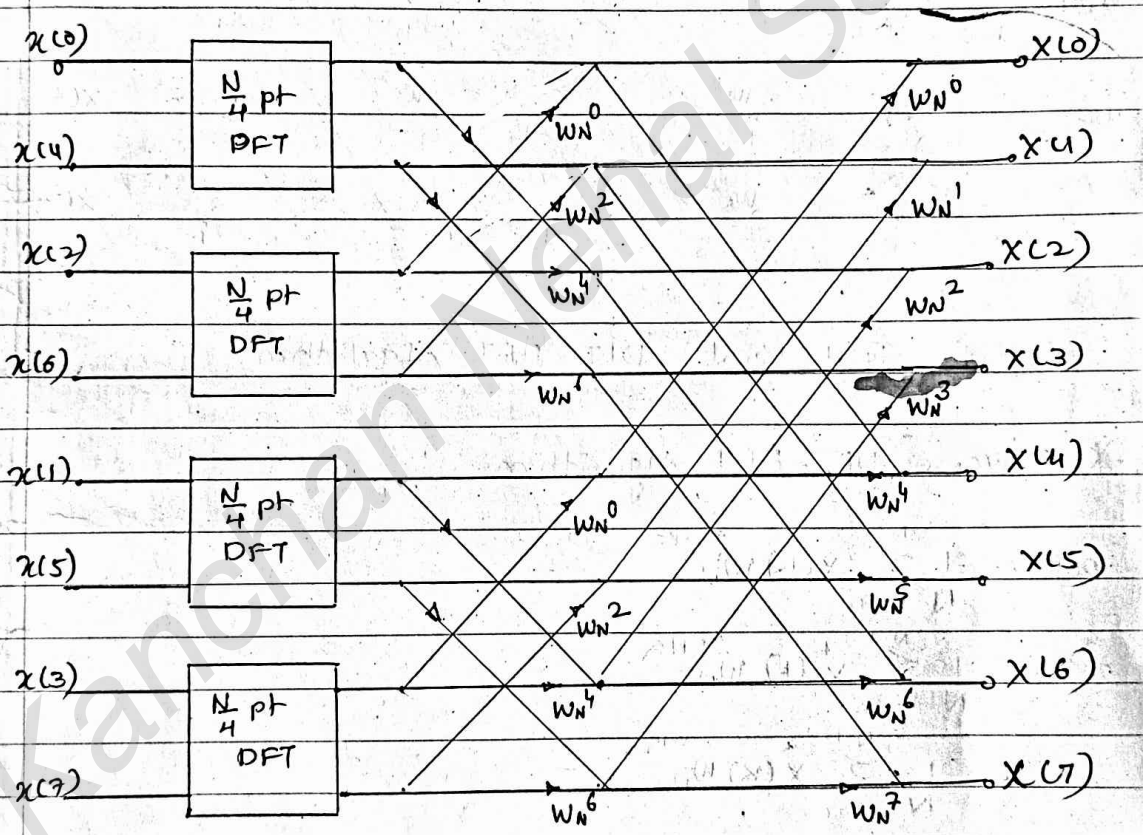


Fig. ⑦ Result of substituting Fig ⑥ into fig ⑤

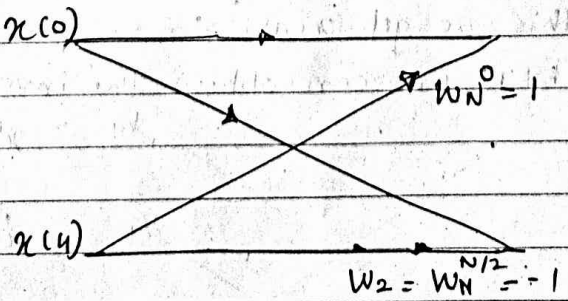


Fig ⑧ Flow graph of a two pt DFT

Substituting in Fig ⑦ we get Fig ⑨

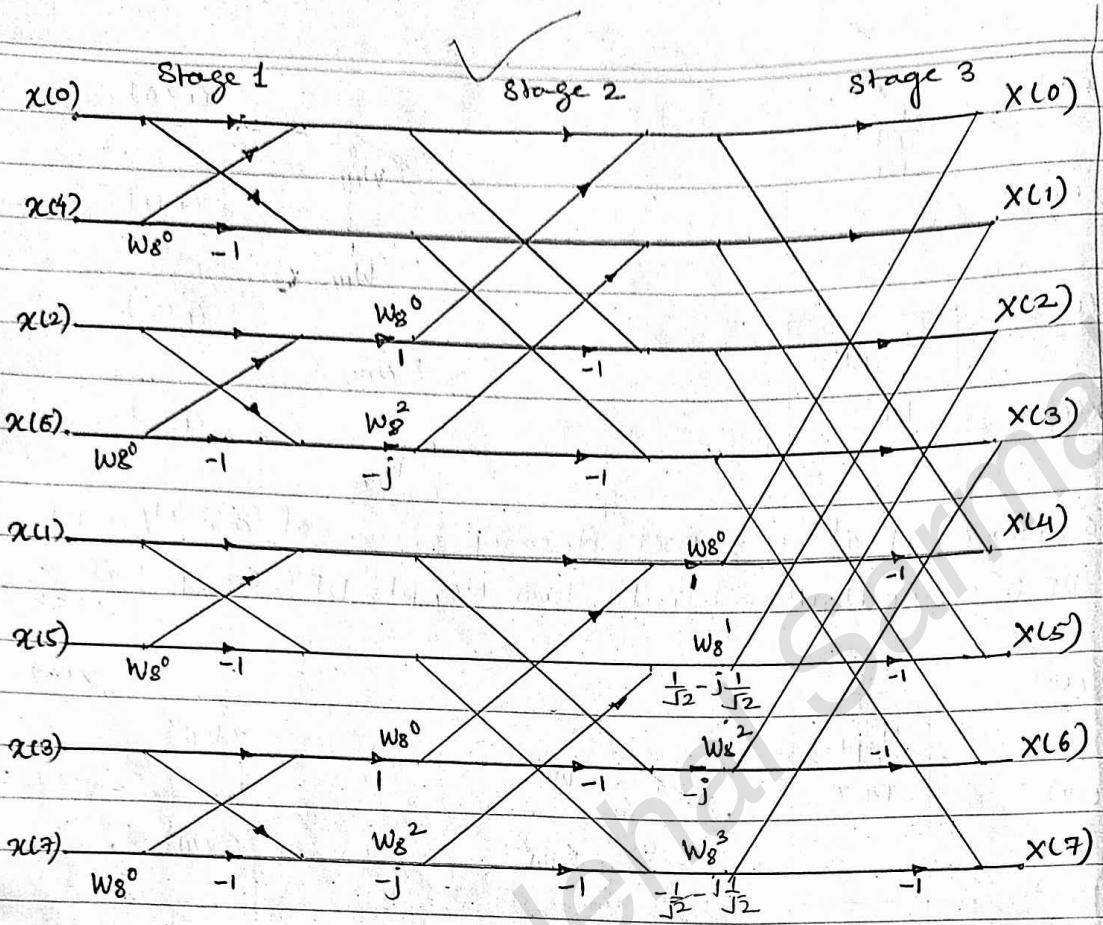


Fig ⑨ Eight Point DIT-FFT Algorithm.

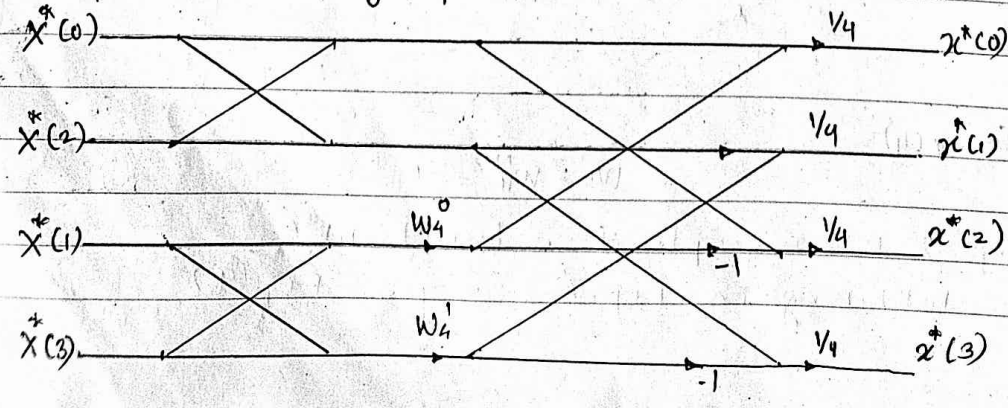
✱ Inverse DIT-FFT Algorithm

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

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

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

∴ In DIT-FFT algorithm, if we replace i/p & make it $X^*(k)$ and divided by N, we will get $x^*(n)$. If we take conjugate of this we get $x(n)$. So simply replacing x by X^* & dividing by N we can obtain the inverse DFT



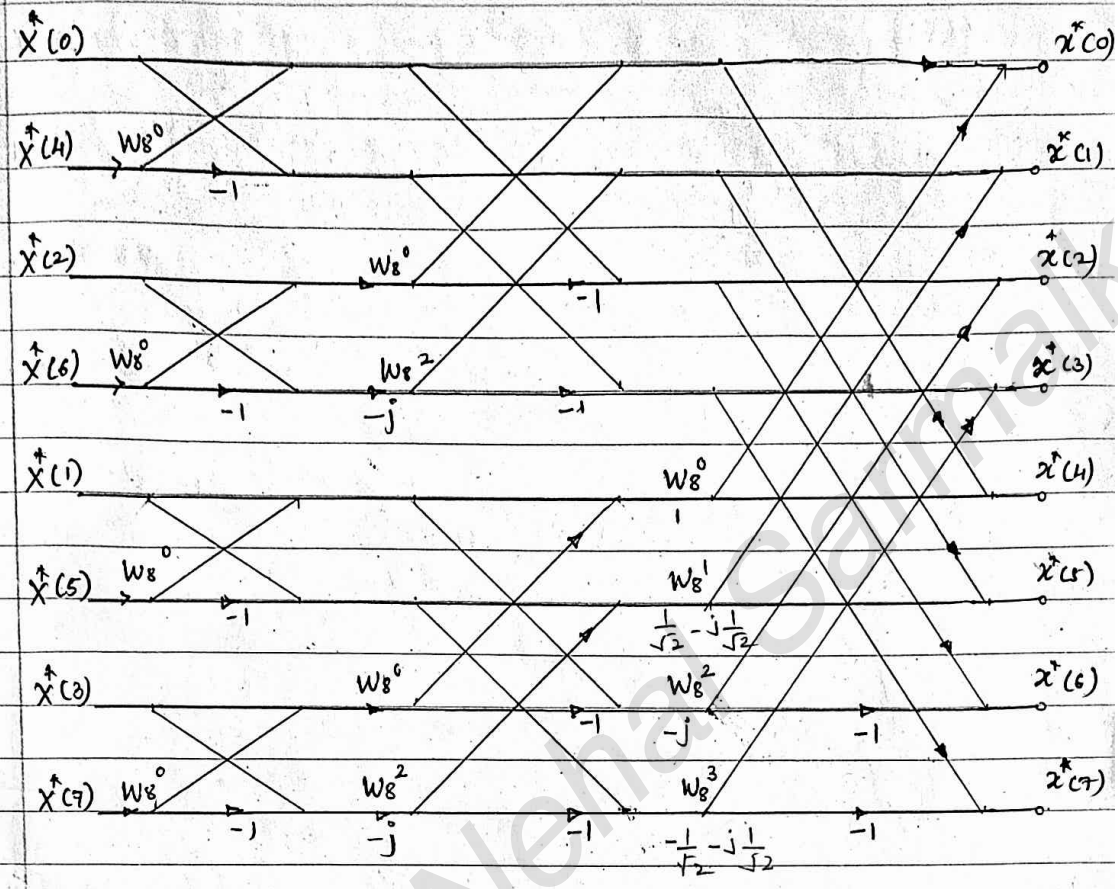
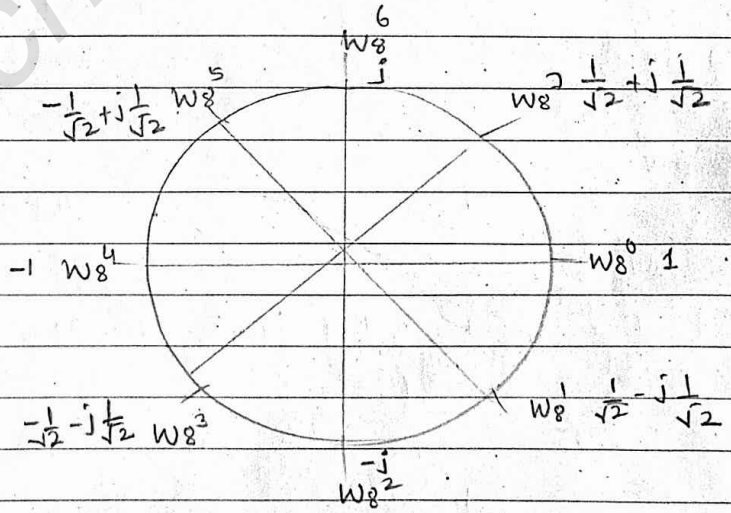
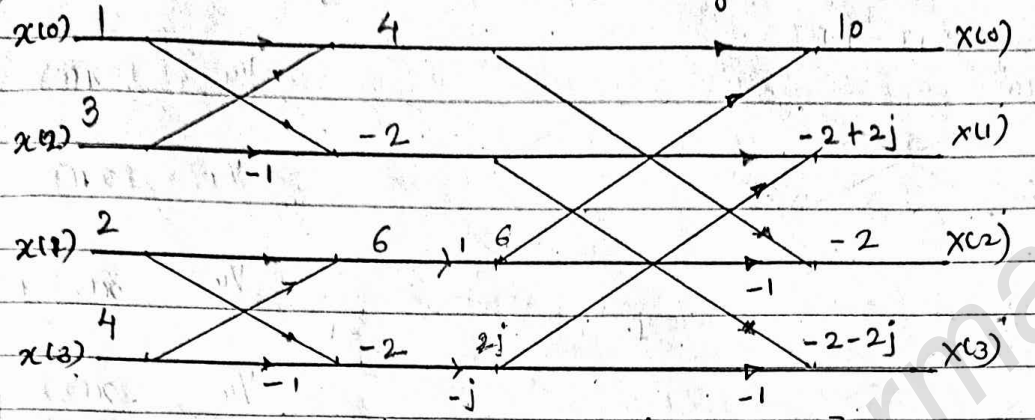


Fig 11. 8 point inverse DIT-FFT flow graph.

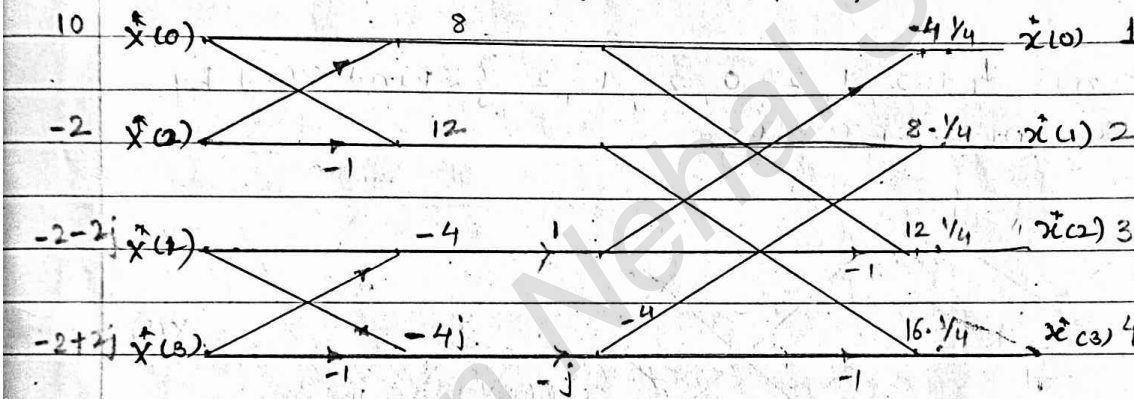


Ex.1 Find DFT of $x(n) = \{1, 2, 3, 4\}$ using DIT - FFT



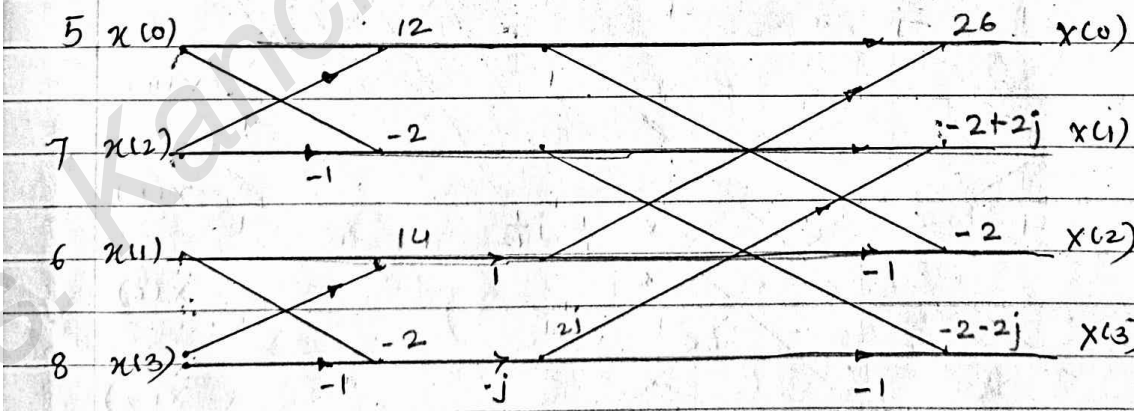
$X(K) = \{10, -2+2j, -2, -2-2j\}$

Ex. Find IDFT of $X(K) = \{10, -2+2j, -2, -2-2j\}$ using DIT-FFT



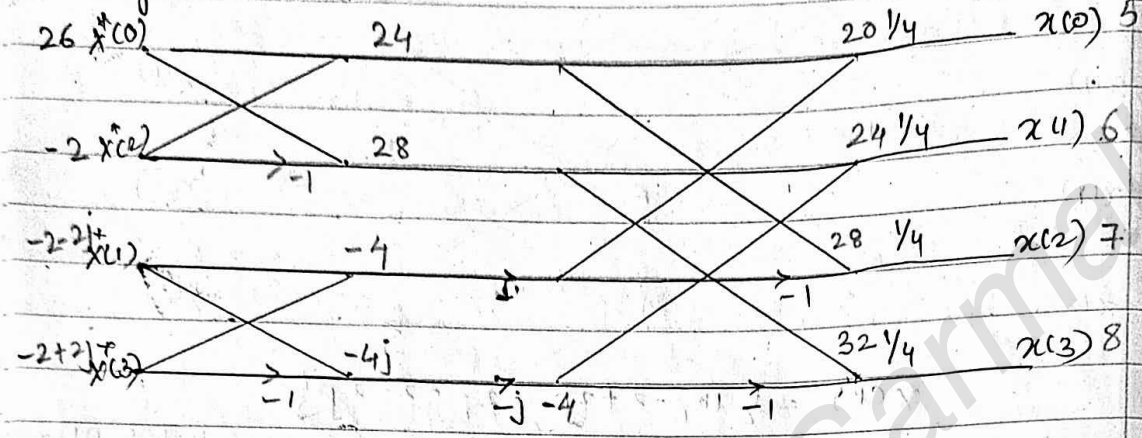
$x(n) = \{1, 2, 3, 4\}$

Ex. Find DFT of $x(n) = \{5, 6, 7, 8\}$ using DIT-FFT



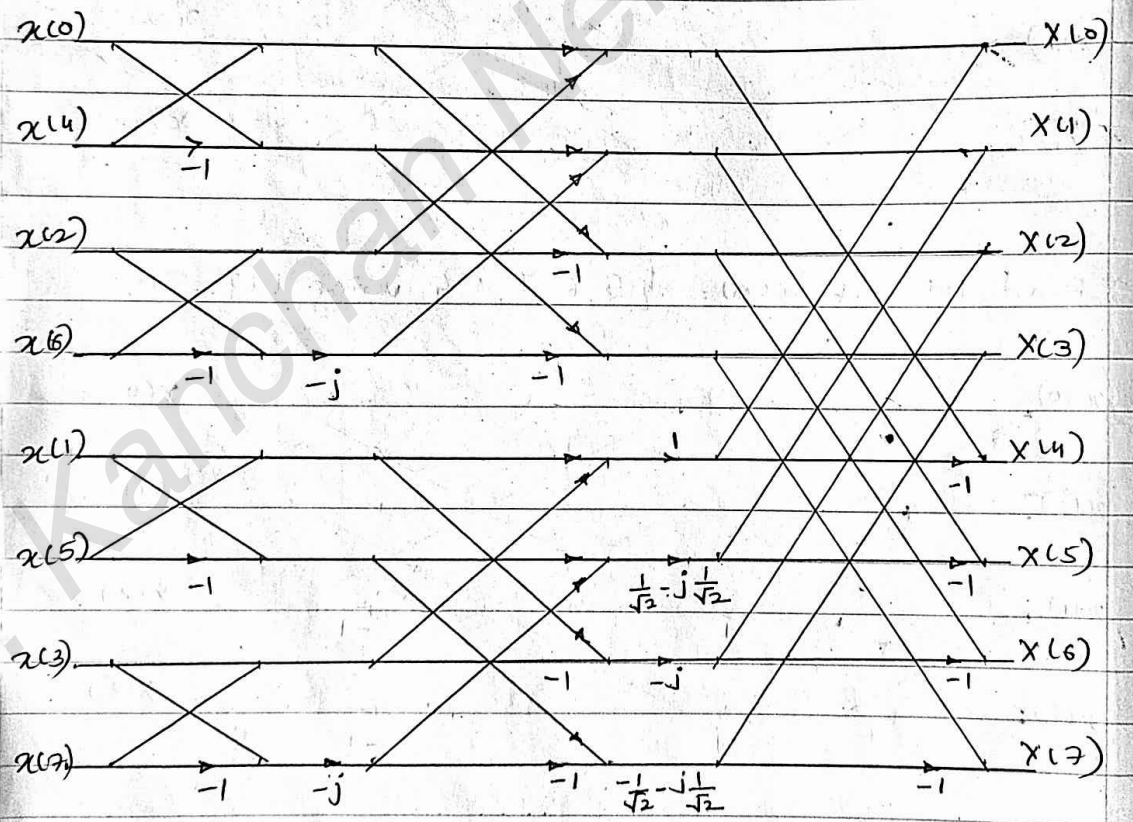
$X(K) = \{26, -2+2j, -2, -2-2j\}$

$X(k) = \{26, -2+2j, -2, -2-2j\}$ Find $x(n)$ using DIT-FFT.



$x(n) = \{5, 6, 7, 8\}$

Ex. $x(n) = \{1, 2, 1, 2, 0, 2, 1, 2\}$ Find $X(k)$ by using DIT-FFT algo.



$X(k) = \{11, 1, -1, 1, -5, 1, -1, 1\}$

Ex. Find $x(n)$ using DIT-IFFT

* Convolution ex on last page.

* DECIMATION IN FREQUENCY ALGORITHM

The DFT of a N pt. sequence is given by

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

Step I : To obtain X(k) for even values of k

Replace k by 2k

$$X(2k) = \sum_{n=0}^{N-1} x(n) W_N^{2nk} \cdot W_N^{n \cdot 2k}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{2nk} W_N^{n \cdot 2k} + \sum_{n=N/2}^{N-1} x(n) W_N^{n \cdot 2k}$$

Substituting $n = n + \frac{N}{2}$ in second summation

$$n = \frac{N}{2} \Longrightarrow n = 0$$

$$n = N-1 \Longrightarrow n = \frac{N}{2} - 1$$

$$\therefore X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{n \cdot 2k} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{\left(n+\frac{N}{2}\right)k \cdot 2}$$

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{n \cdot 2k} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{n \cdot 2k} \cdot W_N^{N \cdot k}$$

Now $W_N^{n2k} = e^{\frac{-j2\pi n \cdot 2k}{N}} = e^{\frac{-j2\pi nk}{N/2}} = W_{N/2}^{nk}$

& $W_N^{Nk} = e^{\frac{-j2\pi Nk}{N}} = e^{-j2\pi k} = 1$

Substituting $X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_{N/2}^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_{N/2}^{k}$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n+\frac{N}{2}\right) \right] W_{N/2}^{nk}$$

Let $x(n) + x\left(n+\frac{N}{2}\right) = g(n)$

$$\therefore X(2k) = \sum_{n=0}^{\frac{N}{2}-1} g(n) W_{N/2}^{nk}$$

$X(2k) = DFT\{g(n)\}$

$X(2k) = G(k)$

Where $g(n) = x(n) + x(n+\frac{N}{2})$

Step II : To obtain $X(k)$ for odd values of k

Replace k by $2k+1$

$$X(2k+1) = \sum_{n=0}^{N-1} x(n) W_N^{n(2k+1)}$$

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

Substituting $n = n + \frac{N}{2}$ in second summation

$$n = \frac{N}{2} \Longrightarrow n = 0$$

& $n = N-1 \Longrightarrow n = \frac{N}{2} - 1$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{n(2k+1)} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\tfrac{N}{2}) W_N^{(n+\frac{N}{2})(2k+1)}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{n2k} W_N^{n} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\tfrac{N}{2}) W_N^{n(2k+1)} W_N^{\frac{N}{2}(2k+1)}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{n2k} W_N^{n} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\tfrac{N}{2}) W_N^{n2k} W_N^{n} W_N^{Nk} W_N^{N/2}$$

Now $W_N^{n.2k} = e^{\frac{-j2\pi n.2k}{N}} = e^{\frac{-j2\pi nk}{N/2}} = W_{N/2}^{nk}$

$W_N^{Nk} = e^{\frac{-j2\pi Nk}{N}} = e^{-j2\pi k} = 1$

$W_N^{N/2} = e^{\frac{-j2\pi N/2}{N}} = e^{-j\pi} = -1$

Substituting

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_{N/2}^{nk} W_N^{n} + \sum_{h=0}^{\frac{N}{2}-1} x(n+\tfrac{N}{2}) W_{N/2}^{nk} W_N^{n} (1)(-1)$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_{N/2}^{nk} W_N^{n} - \sum_{n=0}^{\frac{N}{2}-1} x(n+\tfrac{N}{2}) W_{N/2}^{nk} W_N^{n}$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x(n+\tfrac{N}{2})\right] W_N^{n} W_{N/2}^{nk}$$

let $\left[x(n) - x(n+\frac{N}{2})\right] W_N^{n} = h(n) W_N^{n}$

$$\therefore X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} h(n) W_{N/2}^{nk}$$

$$X(2k+1) = \text{DFT}\left\{h(n) W_N^{n}\right\}$$

$$X(2k+1) = H(k)$$

where $h(n) = \left[x(n) - x\left(n+\frac{N}{2}\right)\right]$

N = 4

Step I $X(2k) = DFT\{g(n)\}$

where $g(n) = x(n) + x\left(n+\frac{N}{2}\right)$

$= x(n) + x(n+2)$ $\because N=4$

$n=0$; $g(0) = x(0) + x(2)$

$n=1$; $g(1) = x(1) + x(3)$

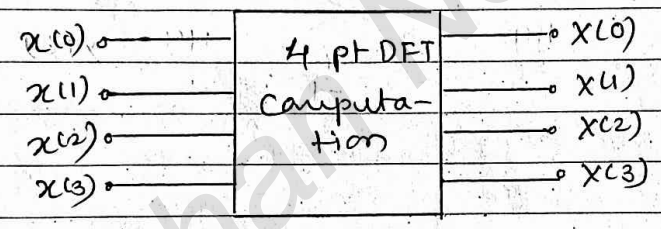
Step II $X(2k+1) = DFT\{h(n) W_N^n\}$

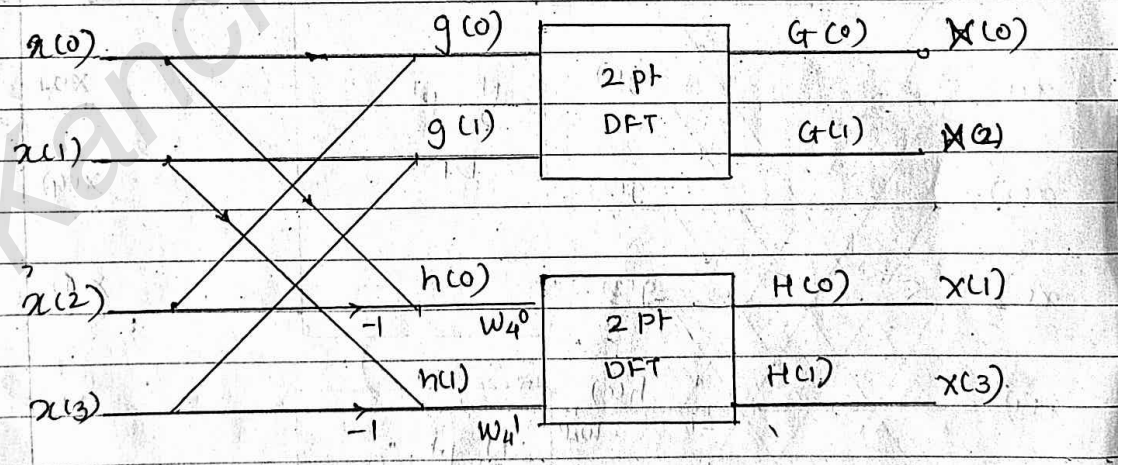
where $h(n) = x(n) - x\left(n+\frac{N}{2}\right)$

$= x(n) - x(n+2)$ $\because N=4$

$n=0$; $h(0) = x(0) - x(2)$

$n=1$; $h(1) = x(1) - x(3)$

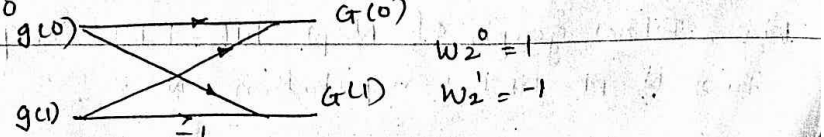




$$G(k) = \sum_{n=0}^{1} g(n) W_2^{nk}$$

$$G(0) = \sum_{n=0}^{1} g(n) \cdot W_2^{n\cdot 0} = g(0) W_2^0 + g(1) W_2^0$$

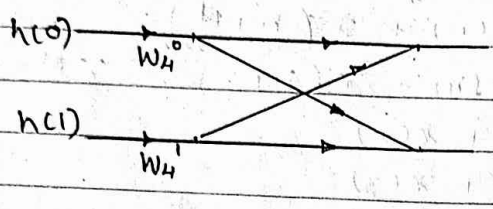
$$G(1) = \sum_{n=0}^{1} g(n) \cdot W_2^{n\cdot 1} = g(0) W_2^0 + g(1) W_2^1$$

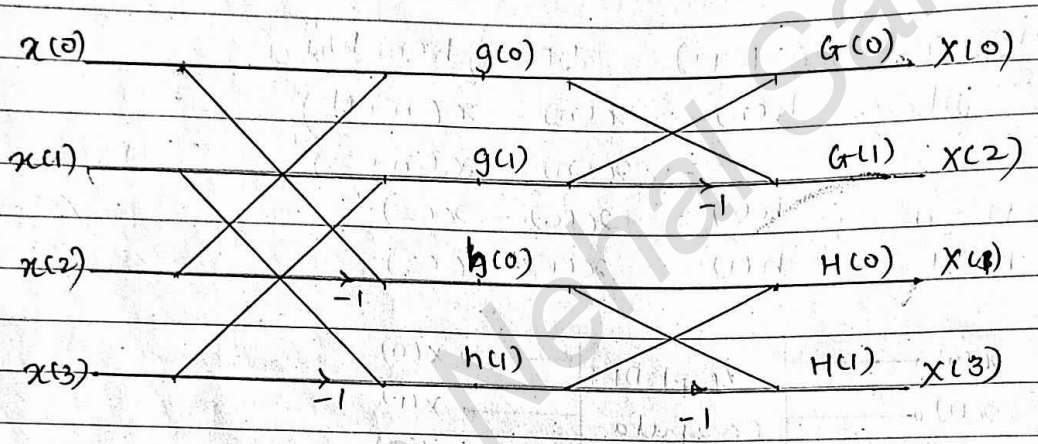


Similarly $H(k) = \sum_{n=0}^{N} h(n) W_4^{nk} W_2^{nk}$

$$H(0) = \sum_{n=0}^{1} h(n) W_4^{n \cdot 0} \cdot W_2^{n \cdot 0} = h(0) W_4^0 W_2^0 + h(1) W_4^1 \cdot W_2^0$$

$$H(1) = \sum_{n=0}^{1} h(n) W_4^{n} W_2^{n \cdot 1} = h(0) W_4^0 W_2^0 + h(1) W_4^1 W_2^1$$





N = 8

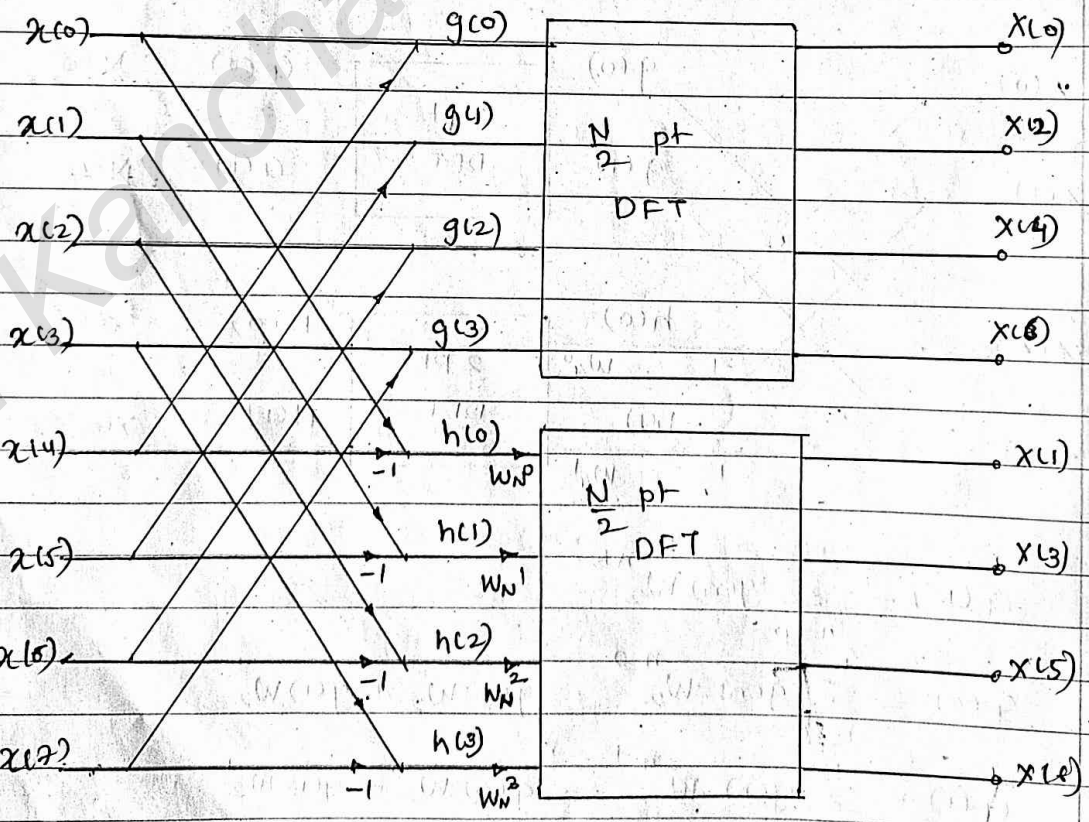
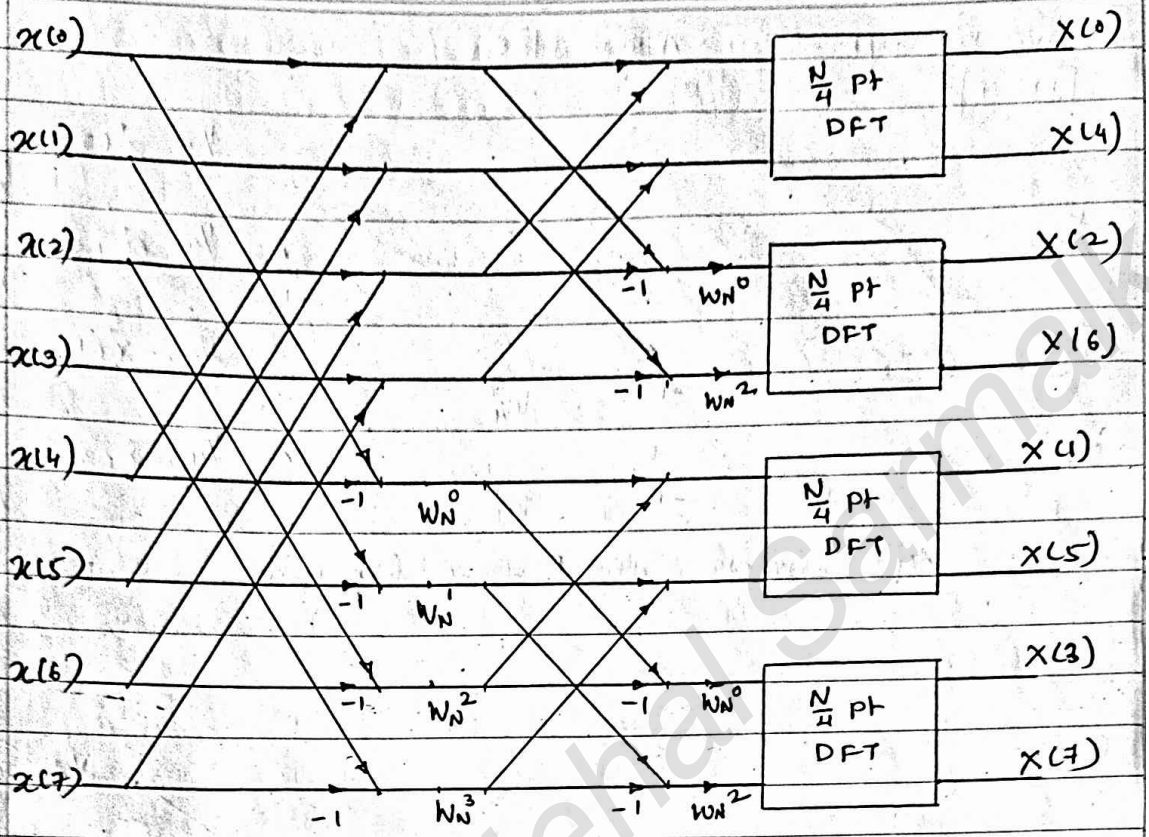
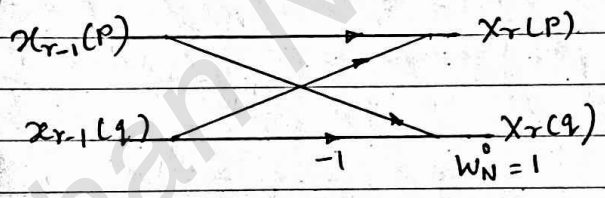
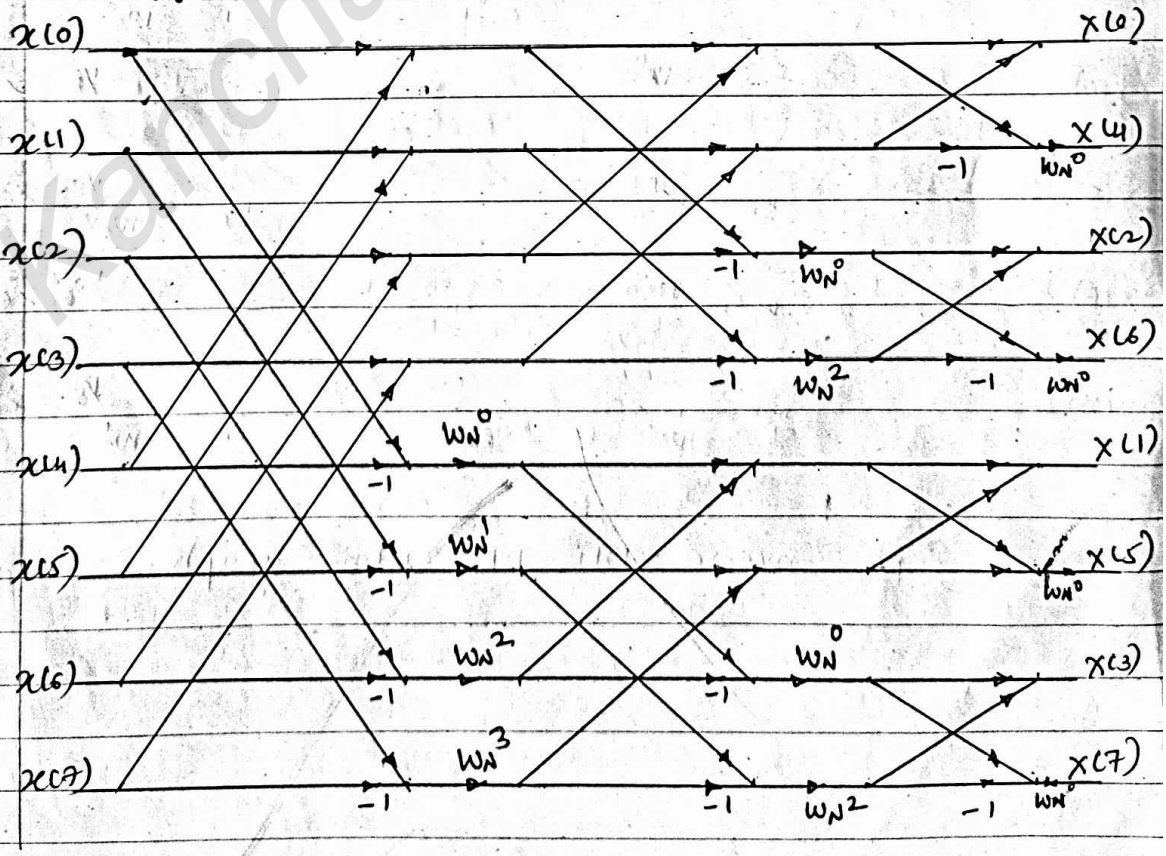


Fig. Flow graph of N pt DIF-FFT computation into two $\frac{N}{2}$ pt DFT computation N = 8

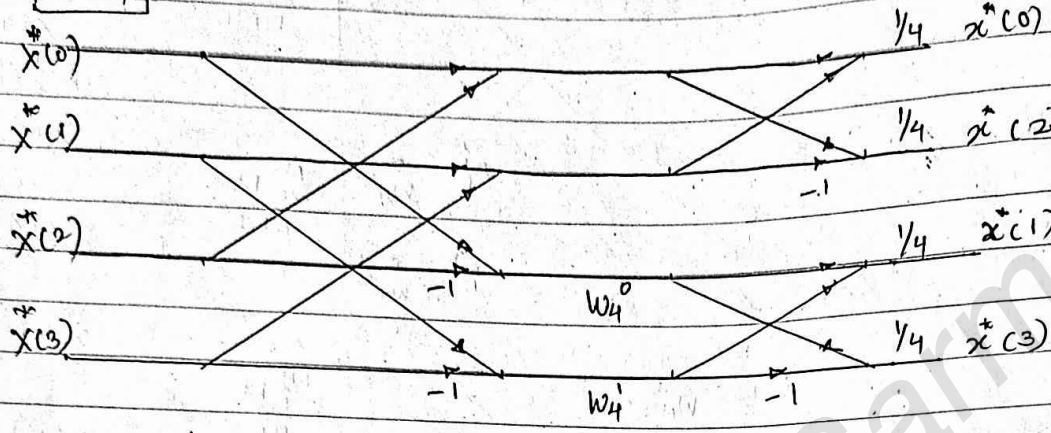






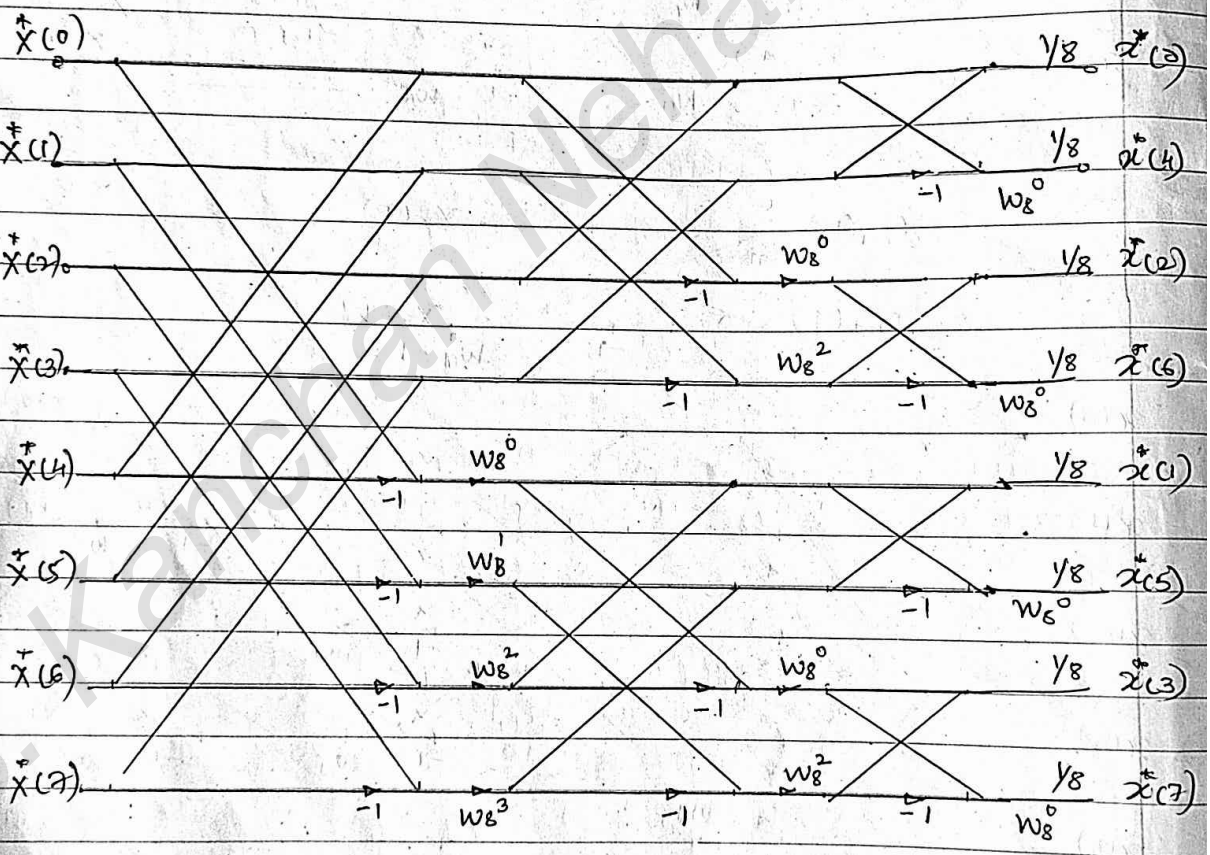
Inverse DIF-FFT Algorithm.

N=4



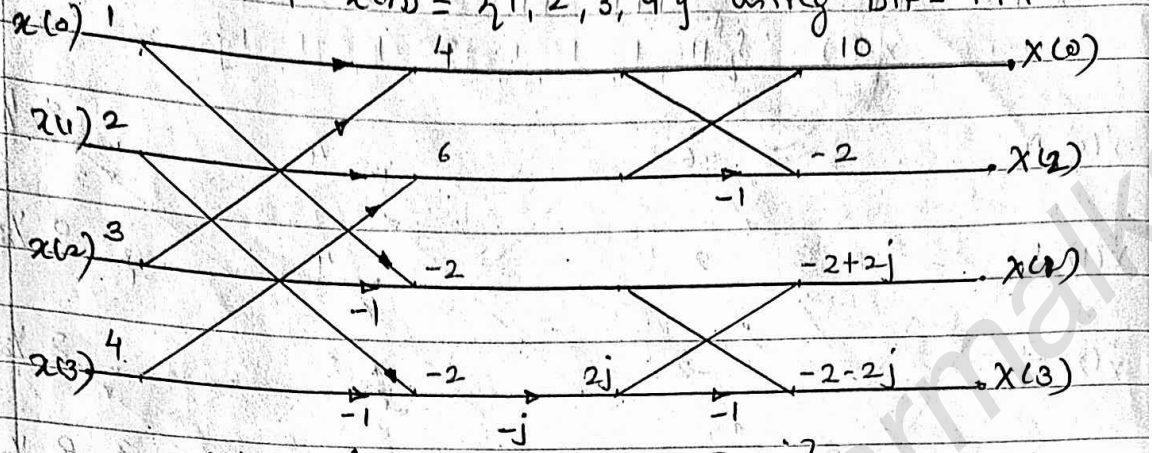
4pt. Inverse DIF-FFT Flowgraph.

N=8



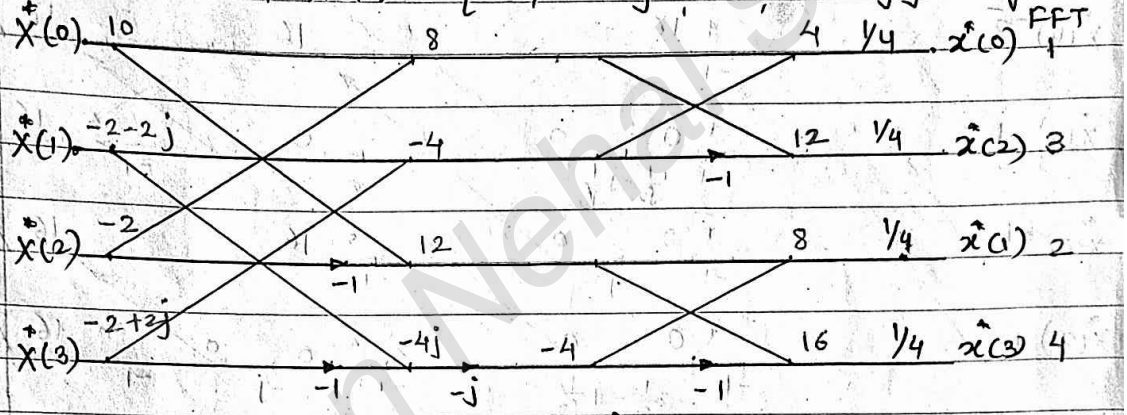
8 pt. Inverse DIF-FFT Flow graph.

Ex. Find DFT of $x(n) = \{1, 2, 3, 4\}$ using DIF- FFT



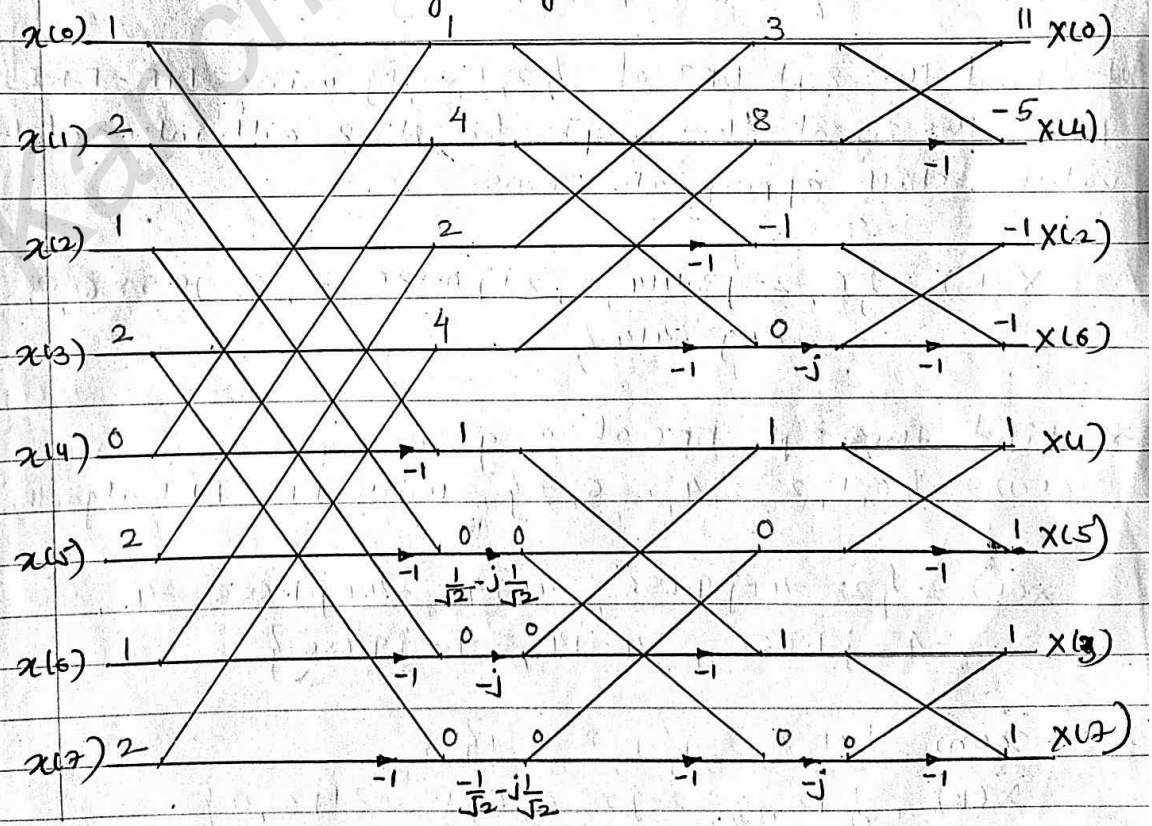
$X(k) = \{10, -2+2j, -2, -2-2j\}$

Ex. Find IDFT of $X(k) = \{10, -2+2j, -2, -2-2j\}$ using IDIF FFT



$x(n) = \{1, 2, 3, 4\}$

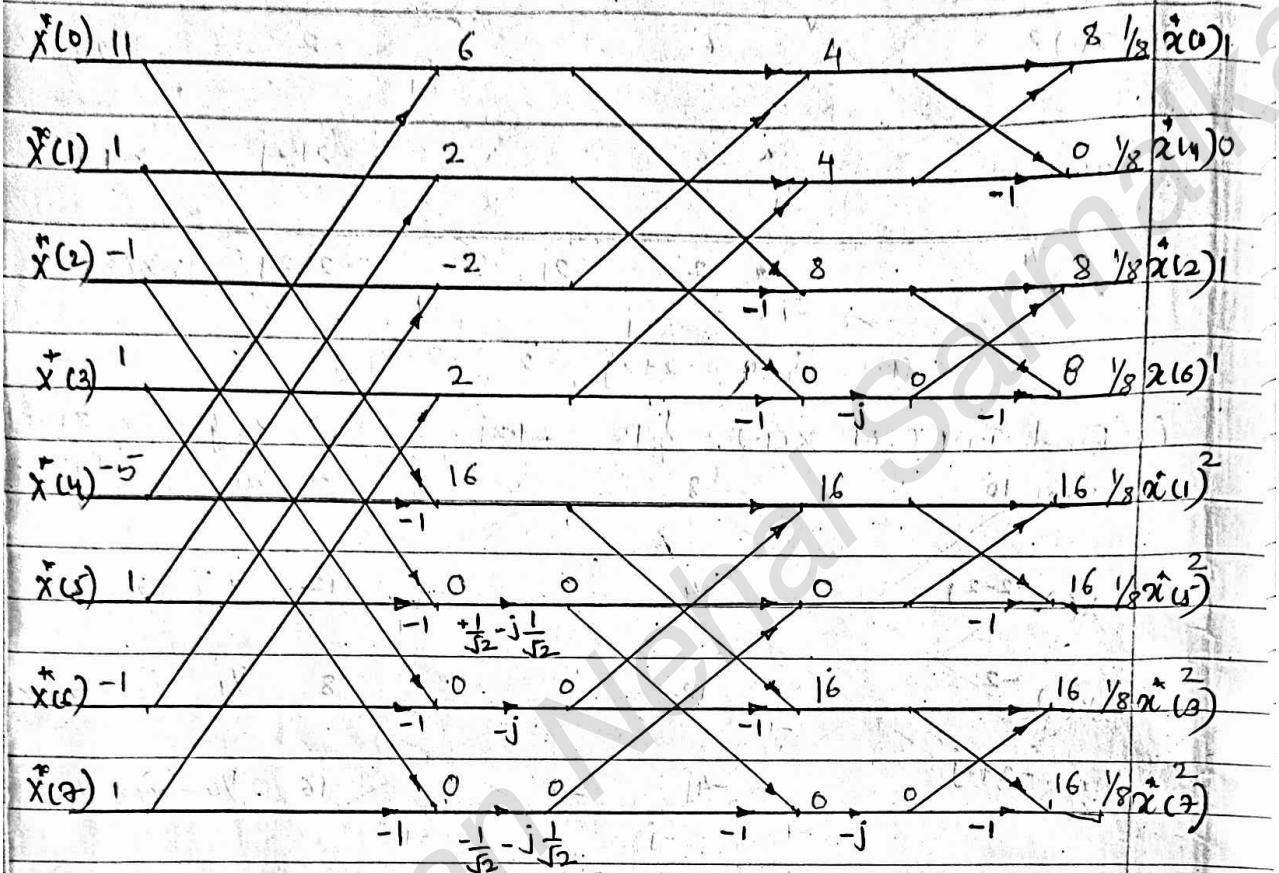
Ex. Find DFT of following using DIF-FFT $x(n) = \{1, 2, 1, 2, 0, 2, 1, 2\}$



$X(k) = \{11, 1, -1, 1, -5, 1, -1, 1\}$

Ex. Find IDFT using IDIF-FFT

$$X(k) = \{11, 1, -1, 1, -5, 1, -1, 1\}$$



$$x(n) = \{1, 2, 1, 2, 0, 2, 1, 2\}$$

* Find the 8 pt DFT of $\{2, 1, 2, 1\}$ using DIF-FFT Draw the signal flow graph for $N = 8$ with intermediate values, stuff appropriate zeros.

$$X(k) = \{6, 2-j3.414, 0, 2+j0.586, 2, 2-j0.586, 0, 2+j3.414\}$$

* Find the 8 pt DFT of a given sequence.
$x(n) = \{0, 1, 2, 3, 4, 5, 6, 7\}$ using DIF FFT algorithm

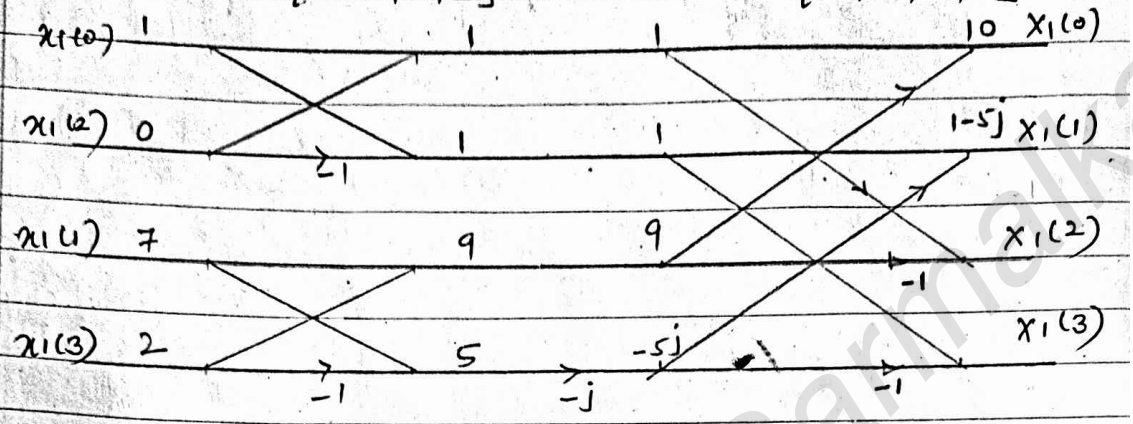
$$X(k) = \{28, -4+j9.656, -4+j4, -4+j1.656, -4, -4-j1.656, -4-j4, -4-j9.656\}$$

* $x(n) = \{1, 2, 2, 1, 1, 2, 2, 1\}$

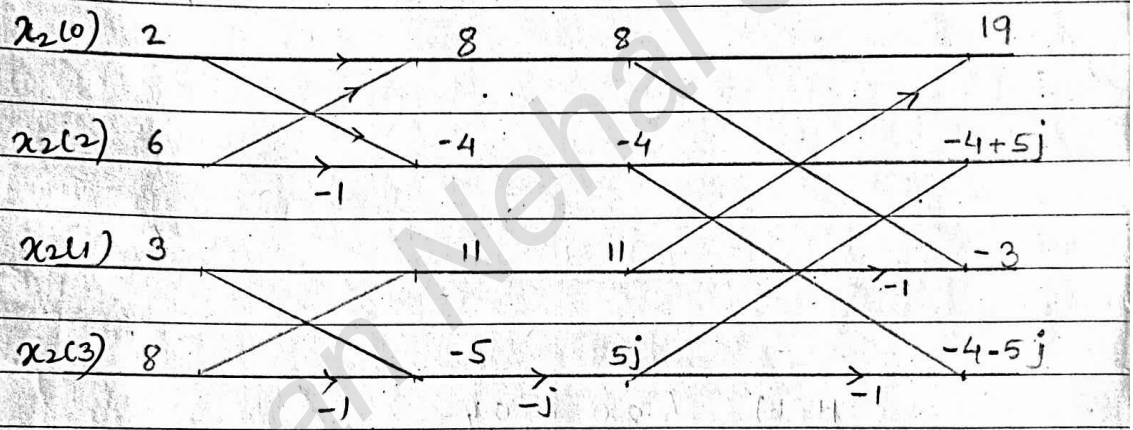
$$X(k) = \{12, 0, 2-j2, 0, 0, 0, -2+j2, 0\}$$

* By means of FFT-IFFT technique compute the circular convolution of the sequence $x_1(n) = \{1, 2, 3, 4\}$
$x_2(n) = \{5, 6, 7, 8\}$ Ans- $\{66, 68, 66, 60\}$

Ex. Perform circular convolution using DIT-FFT only.
$x_1(n) = \{1, 7, 0, 2\}$ $x_2(n) = \{2, 3, 6, 8\}$



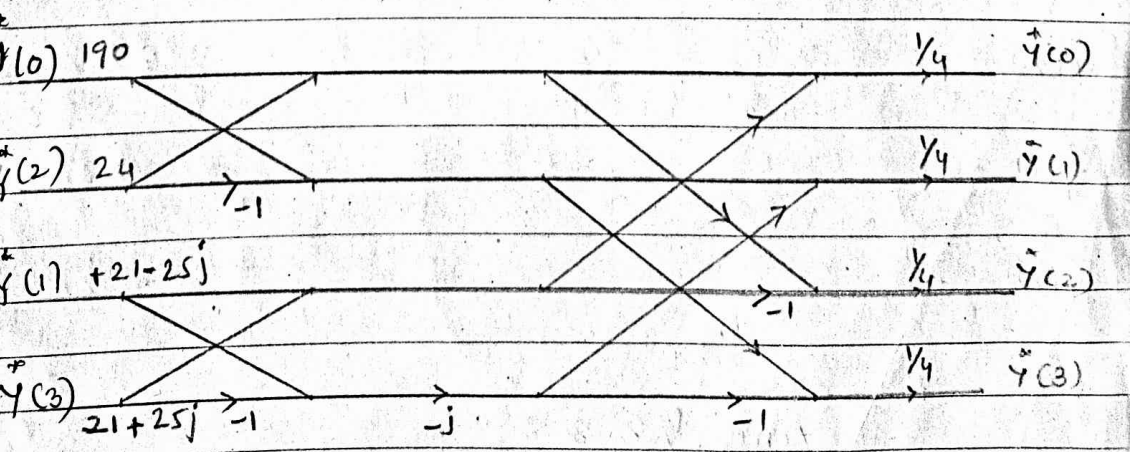
$X_1(k) = \{10, 1-5j, -8, 1+5j\}$



$X_2(k) = \{19, -4+5j, -3, -4-5j\}$

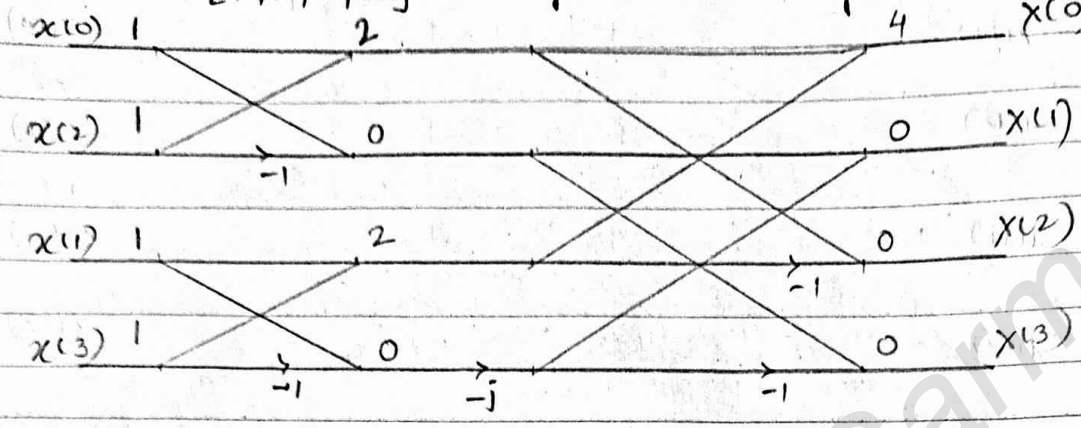
$Y(k) = X_1(k) \cdot X_2(k)$:

$$= \begin{bmatrix} 10 \\ 1-5j \\ -8 \\ 1+5j \end{bmatrix} \begin{bmatrix} 19 \\ -4+5j \\ -3 \\ -4-5j \end{bmatrix} = \begin{bmatrix} 190 \\ +21+25j \\ 24 \\ 21-25j \end{bmatrix} \begin{matrix} \\ -4+5j+20j+25 \\ \\ -4-5j-20j+25 \end{matrix}$$

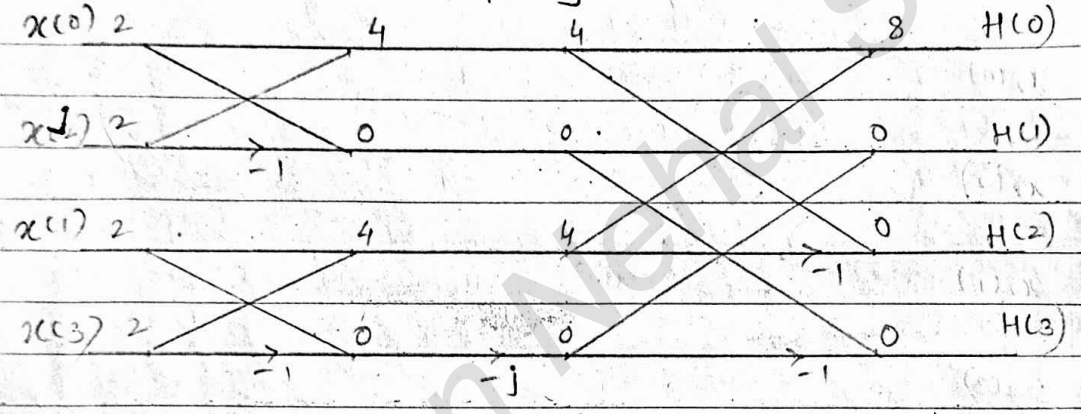


$\{64, 29, 43, 54\}$

Ex. Find circular convolution of $x(n) = \{1,1,1,1\}$ & $h(n) = \{2,2,2,2\}$ using DIT-FFT algo.



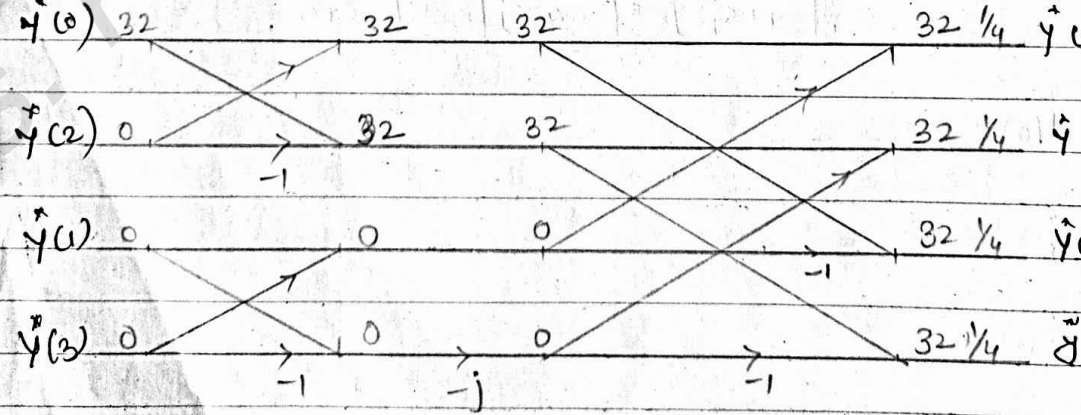
$X(k) = \{4,0,0,0\}$



$H(k) = \{8,0,0,0\}$

$Y(k) = X(k) * H(k)$

$$= \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 8 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 32 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



$y(n) = \{8,8,8,8\}$