Cooley & Tukey, 1965 → Fast Fourier Transform (FFT)

Divide & Combine Approach.

DFT As an Operator : A Matrix Representation →

$$X(K) = \sum_{n=0}^{N-1} x[n]\, e^{-\frac{j2\pi}{N} \cdot Kn} = \sum_{n=0}^{N-1} x[n] \left[e^{-\frac{j2\pi}{N}}\right]^{Kn}$$

$\left[e^{-\frac{j2\pi}{N}}\right] \rightarrow W_N \rightarrow$ Twiddle Factor or phase factor

$$\boxed{X(K) = \sum_{n=0}^{N-1} x[n]\, W_N^{Kn}} \quad \rightarrow (1) \quad ; \; K = 0, 1, 2, \ldots, N-1.$$

&

$$x[n] = \frac{1}{N} \sum_{K=0}^{N-1} X(K) \left[e^{\frac{j2\pi}{N}}\right]^{Kn} \quad ; \quad n = 0, 1, 2, 3, \ldots N-1$$

$$\boxed{x[n] = \frac{1}{N} \sum_{K=0}^{N-1} X(K)\, W_N^{-Kn}} \quad \rightarrow (2) \quad ;$$

Note eqn (1) & (2) both represents individually a set of N eqns. e.g. If we analyse eqn (1) for different values of K we get

Direct DFT computation Needs

- N^2: Complex Multiplications
- $N(N-1)$ Complex additions.
- $\approx N^2$: Complex additions

To store N^2 complex Coeff: W_N^{kn}

Computation complexity $\rightarrow O[N^2]$

Inacceptable in practice

$$X(0) = x[0] W_N^{(0)(0)} + x[1] W_N^{(0)(1)} + x[2] W_N^{(0)(2)} + \ldots + x[N-1] W_N^{(0)(N-1)}$$

$$X(1) = x[0] W_N^{(1)(0)} + x[1] W_N^{(1)(1)} + x[2] W_N^{(1)(2)} + \ldots + x(N-1) W_N^{(1)(N-1)}$$

$$X(2) = x[0] W_N^{(2)(0)} + x(1) W_N^{(2)(1)} + x[2] W_N^{(2)(2)} + \ldots + x(N-1) W_N^{(2)(N-1)}$$

$$\vdots$$

$$X(N-1) = x[0] W_N^{(N-1)(0)} + x(1) W_N^{(N-1)(1)} + x[2] W_N^{(N-1)2} + \ldots + x_{N-1} W_N^{(N-1)(N-1)}$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & & & & \\ 1 & & & & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix}$$

($K \downarrow$, $n \rightarrow$)

$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$ (periodicity property)

$W_N^{k+N/2} = -W_N^{kn}$ (symmetry property)

$N \times N$ DFT MATRIX.

Reduce Computational complexity to $O[N \log N]$ for radix 2 FFT using Periodicity & symmetry property of W_N^{kn} in K & n.

Properties of DFT Matrix :-

1. It is a square matrix and is symmetrical about principal diagonal.
2. The enteries in zeroth row & zeroth column is all 1's.
3. In general, the g entry at k^{th} row & n^{th} column $0 \le k, n \le N-1$ is given by $W_N^{k \cdot n}$.

Similarly

IDFT can also be represented in the form of a matrix

$$x(0) = \frac{1}{N}\left[X(0)W_N^{-(0)(0)} + X(1)W_N^{-(0)\cdot 1} + X(2)W_N^{-(0)\cdot 2} + \cdots + X(N-1)W_N^{-(0)(N-1)}\right]$$

$$x(1) = \frac{1}{N}\left[X(0)W_N^{-(1)\cdot(0)} + X(1)W_N^{-(1)(1)} + X(2)W_N^{-(1)(2)} + \cdots + X(N-1)W_N^{-(1)(N-1)}\right]$$

$$x(2) = \frac{1}{N}\left[X(0)W_N^{-(2)\cdot(0)} + X(1)W_N^{-(2)(1)} + X(2)W_N^{-(2)(2)} + \cdots + X(N-1)W_N^{-(2)(N-1)}\right]$$

$$\vdots$$

$$x[N-1] = \frac{1}{N}\left[X(0)W_N^{-(N-1)\cdot(0)} + X(1)W_N^{-(N-1)(1)} + X(2)W_N^{-(N-1)(2)} + \cdots + X(N-1)W_N^{-(N-1)^2}\right]$$

Hence

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \cdots & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & \cdots & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix}\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix}$$

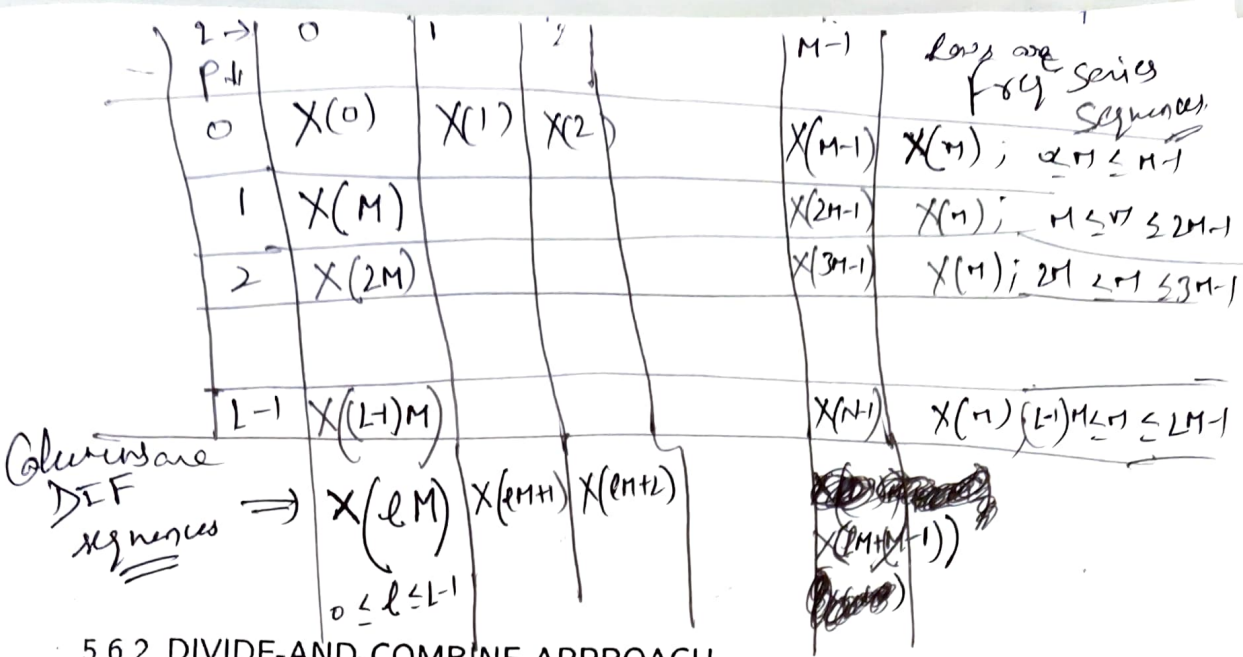
$N \times N$; IDFT Matrix = DFT Matrix

Imp:

Note: $W_N^N = W_N^{-N} = \left(e^{-j\frac{2\pi}{N}}\right)^{-N} = e^{j2\pi} = 1$

$W_N^{\pm N/2} = \left[e^{-j\frac{2\pi}{N}}\right]^{\pm\frac{N}{2}} = e^{\pm j\pi} = -1$

$W_4 = -j$; $W_2 = -1$; $W_8 = \frac{1}{\sqrt{2}} - \frac{j}{\sqrt{2}}$



5.6.2 DIVIDE-AND-COMBINE APPROACH

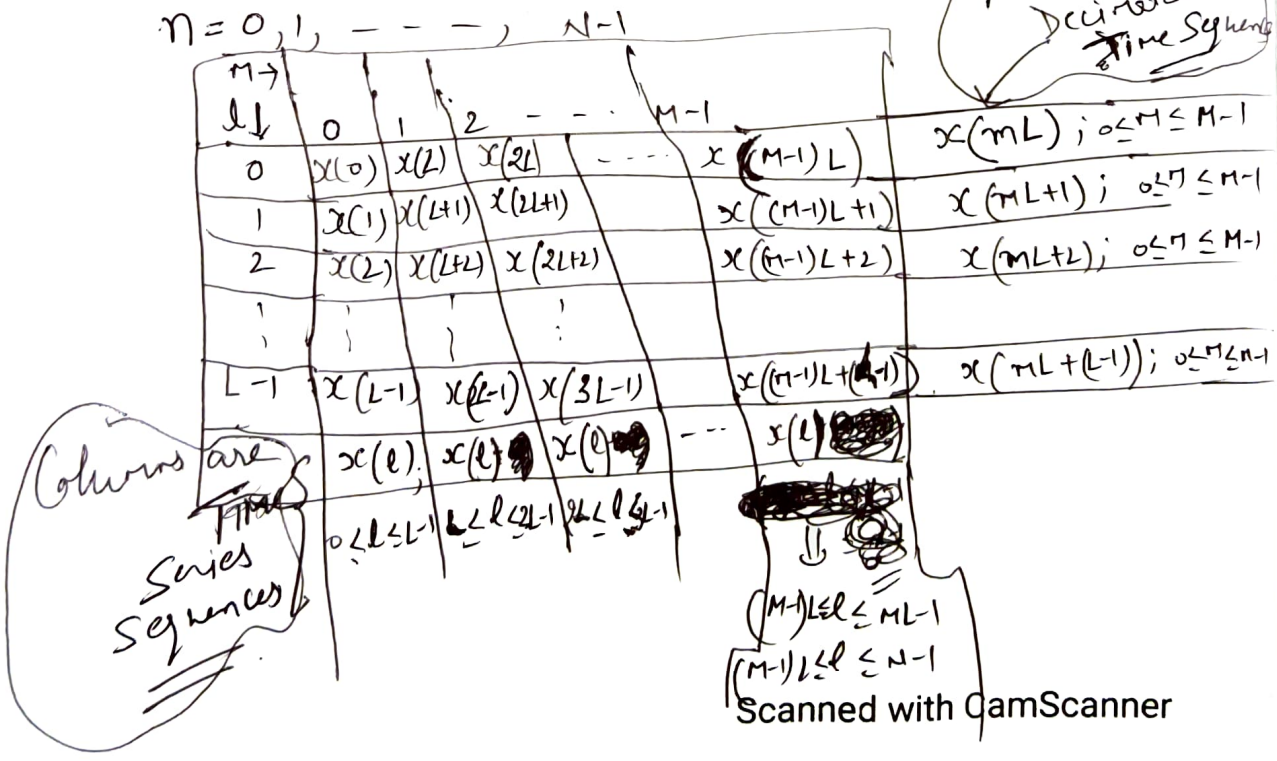
To reduce the DFT computation's quadratic dependence on N, one must choose a composite number $N = LM$ since

$$L^2 + M^2 \ll N^2 \quad \text{for large } N$$

Now divide the sequence into M smaller sequences of length L, compute M smaller L-point DFTs, and then combine these into a larger DFT using L smaller M-point DFTs. This is the essence of the divide-and-combine approach. Let $N = LM$; then the indices n and k in (5.46) can be written as

$$n = \ell + Lm, \; 0 \le \ell \le L-1, \; 0 \le m \le M-1$$

$$k = q + Mp, \; 0 \le p \le L-1, \; 0 \le q \le M-1 \qquad (5.48)$$



Finally, after "unwinding" this array in the row-wise fashion, we obtain the required 15-point DFT $X(k)$. The total number of complex operations required for this divide-and-combine approach is 135, whereas the direct approach for the 15-point DFT requires 225 complex operations. Thus the divide-and-combine approach is clearly efficient. □

The divide-and-combine procedure can be further repeated if M or L are composite numbers. Clearly, the most efficient algorithm is obtained when N is a highly composite number, that is, $N = R^{\nu}$. Such algorithms are called *radix-R* FFT algorithms. When $N = R_1^{\nu_1} R_2^{\nu_2} \ldots$, then such decompositions are called *mixed-radix* FFT algorithms. The one most popular and easily programmable algorithm is the radix-2 FFT algorithm.

5.6.3 RADIX-2 FFT ALGORITHM

Let $N = 2^{\nu}$; then we choose $L = 2$ and $M = N/2$ and divide $x(n)$ into two $N/2$-point sequences according to (5.48) as

$$\left.\begin{aligned} g_1(n) &= x(2n) \\ g_2(n) &= x(2n+1) \end{aligned}\right\}; \quad 0 \le n \le \frac{N}{2} - 1$$

The sequence $g_1(n)$ contains even-ordered samples of $x(n)$, while $g_2(n)$ contains odd-ordered samples of $x(n)$. Let $G_1(k)$ and $G_2(k)$ be $N/2$-point DFTs of $g_1(n)$ and $g_2(n)$, respectively. Then (5.49) reduces to

$$X(k) = G_1(k) + W_N^k G_2(k), \quad 0 \le k \le N-1 \tag{5.61}$$

This is called a *merging formula*, which combines two $N/2$-point DFTs into one N-point DFT. The total number of complex multiplications and additions reduces to

$$C_N = \frac{N^2}{2} + N = o\left(N^2/2\right)$$

This procedure can be repeated again and again. At each stage, the sequences are decimated and the smaller DFTs combined. This decimation ends after ν stages when we have N one-point sequences, which are also one-point DFTs. The resulting procedure is called the *decimation-in-time* FFT (DIT-FFT) algorithm, for which the total number of complex multiplications is

$$C_N = N\nu = N \log_2 N$$

Clearly, if N is large, then C_N is approximately linear in N, which was the goal of our efficient algorithm. Using additional symmetries, C_N can be reduced to $\frac{N}{2} \log_2 N$. The signal flowgraph for this algorithm is shown in Figure 5.20 for $N = 8$.

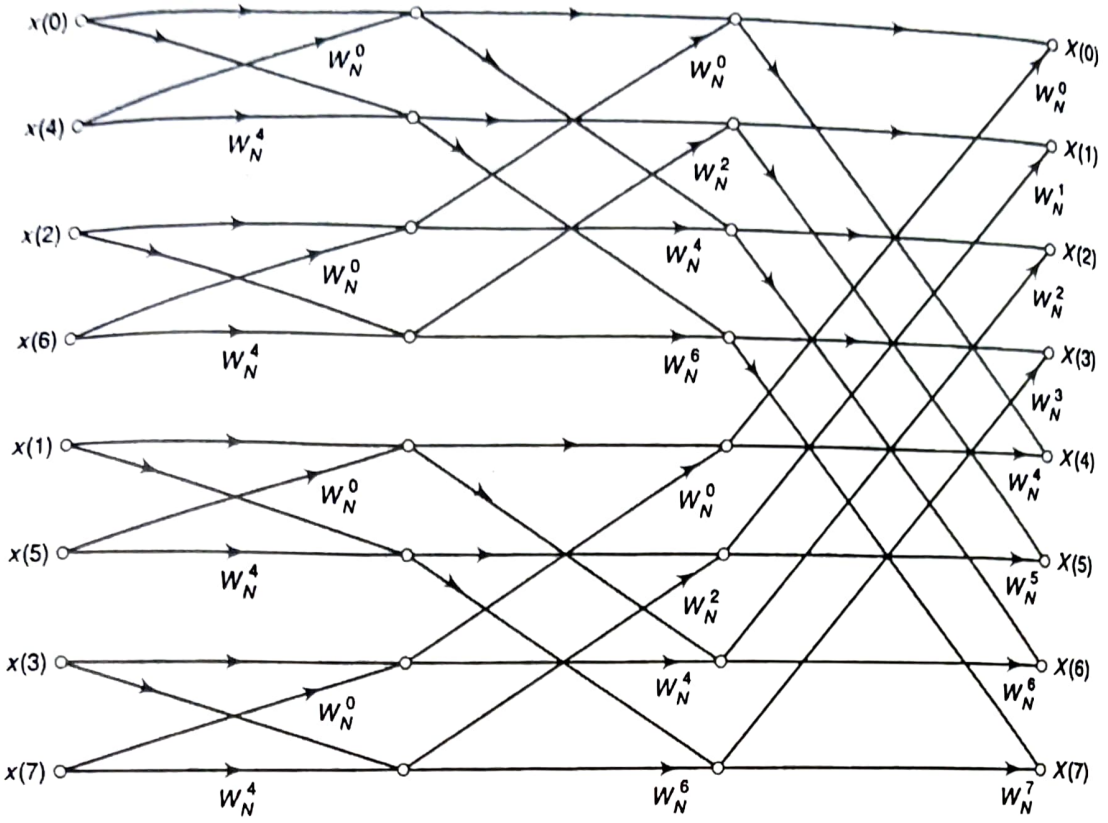


FIGURE 5.20 *Decimation-in-time FFT structure for* $N = 8$

In an alternate approach, we choose $M = 2$, $L = N/2$ and follow the steps in (5.49). Note that the initial DFTs are two-point DFTs, which contain no complex multiplications. From (5.50),

$$\begin{aligned} F(0,m) &= x(0,m) + x(1,m)W_2^0 \\ &= x(n) + x(n+N/2), \quad 0 \le n \le N/2 \\ F(1,m) &= x(0,m) + x(1,m)W_2^1 \\ &= x(n) - x(n+N/2), \quad 0 \le n \le N/2 \end{aligned}$$

and from (5.51),

$$\begin{aligned} G(0,m) &= F(0,m)W_N^0 \\ &= x(n) + x(n+N/2), \quad 0 \le n \le N/2 \\ G(1,m) &= F(1,m)W_N^m \\ &= [x(n) - x(n+N/2)]\,W_N^n, \quad 0 \le n \le N/2 \end{aligned} \tag{5.62}$$

Let $G(0,m) = d_1(n)$ and $G(1,m) = d_2(n)$ for $0 \leq n \leq N/2 - 1$ (since they can be considered as time-domain sequences); then from (5.52) we have

$$\begin{aligned} X(0,q) = X(2q) &= D_1(q) \\ X(1,q) = X(2q+1) &= D_2(q) \end{aligned} \tag{5.63}$$

This implies that the DFT values $X(k)$ are computed in a decimated fashion. Therefore, this approach is called a *decimation-in-frequency* FFT (DIF-FFT) algorithm. Its signal flowgraph is a transposed structure of the DIT-FFT structure, and its computational complexity is also equal to $\frac{N}{2}\log_2 N$.

5.6.4 MATLAB IMPLEMENTATION

MATLAB provides a function called `fft` to compute the DFT of a vector `x`. It is invoked by `X = fft(x,N)`, which computes the N-point DFT. If the length of `x` is less than `N`, then `x` is padded with zeros. If the argument `N` is omitted, then the length of the DFT is the length of `x`. If `x` is a matrix, then `fft(x,N)` computes the N-point DFT of each column of `x`.

This `fft` function is written in machine language and not using MATLAB commands (i.e., it is not available as a `.m` file). Therefore, it executes very fast. It is written as a mixed-radix algorithm. If N is a power of two, then a high-speed radix-2 FFT algorithm is employed. If N is not a power of two, then N is decomposed into prime factors and a slower mixed-radix FFT algorithm is used. Finally, if N is a prime number, then the `fft` function is reduced to the raw DFT algorithm.

The inverse DFT is computed using the `ifft` function, which has the same characteristics as `fft`.

$N = 2^v$ ← Radix-2 FFT Algo^n → $N = R^v$

Radix-2 DIT FFT Algorithm

Approach:

i) Divide & Combine Recursively using $L=2$, $M=\frac{N}{2}$;

ii) Process $x(l,m)$ row-wise; by using DIT sequences, $x(2n)$ & $x(2n+1)$; $0 \le n \le \frac{N}{2}-1$

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

$$= \sum_{n=\{2m;\ 0\le m\le \frac{N}{2}-1\}} x[n]\, W_N^{kn} + \sum_{n=\{2m+1;\ 0\le m\le \frac{N}{2}-1\}} x[n]\, W_N^{kn}$$

$n = 2m$

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

$x[n] = x[2n] + x[2n+1]$

N-DFT of $x[n]$ = $\frac{N}{2}$-DFT of $x[2n]$ + $\frac{N}{2}$-DFT of $x[2n+1]$

$X(k) = X_1(k) + W_N^k X_2(k)$

N-DFT; $\frac{N}{2}$-DFT; $\frac{N}{2}$-DFT

Radix-R DIT FFT

$x[Rn]$, $x[Rn+1]$, ..., $x[Rn+(R-1)]$ → Radix-R DIT FFT → $X(k)$, ..., $X(k+\frac{N}{R}(R-1))$

$0 \le n, k \le \frac{N}{R}-1$

1. Perfectly fits if $N = R$.
2. For $N > R$; Nesting division needs to be considered.

Core of Radix-2 DIT formulation

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x(2m)\, W_{\frac{N}{2}}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} x(2m+1)\, W_{\frac{N}{2}}^{km} \qquad 0 \le k \le N-1 \quad (C1)$$

$X(k) = X_1(k) + W_N^k X_2(k)$

first $\frac{N}{2}$-DFT; $X_1(k)$ — Second $\frac{N}{2}$-DFT, $X_2(k)$

N-DFT $[x[n]] = \frac{N}{2}$-DFT $[x(2n)]$ + $W_N^k \cdot \frac{N}{2}$-DFT $[x(2n+1)]$

Merger/ Combination formula to get N-DFT using two $\frac{N}{2}$-DFTs of $x(2n)$ & $x(2n+1)$.

Require ⇒ $2\left(\frac{N}{2}\right)^2 + N$ Complex multiplications.

$$X_1(k+\tfrac{N}{2}) = X_1(k)$$
$$X_2(k+\tfrac{N}{2}) = X_2(k)$$
$$W_N^{k+\frac{N}{2}} = -W_N^k$$
$$0 \le k \le \tfrac{N}{2}-1$$

Faster Radix-2 DIT formulation needs $\frac{N}{2}$ lesser complex multipliers.

$$X(k) = X_1(k) + W_N^k X_2(k) \quad (C11)$$

$$X\left(k+\tfrac{N}{2}\right) = X_1\left(k+\tfrac{N}{2}\right) + W_N^{k+\frac{N}{2}} X_2\left(k+\tfrac{N}{2}\right)$$

$$X\left(k+\tfrac{N}{2}\right) = X_1(k) - W_N^k X_2(k). \quad (C12)$$

$0 \le k \le \frac{N}{2}-1$

Requires: $2\left(\frac{N}{2}\right)^2 + \frac{N}{2}$ number of complex multiplications

N/2-point DFT

$N = R^v$ | $N = 2^v = 8 = 2^3$

Radix-2 DIT FFT

Divide & Combine Recursively

using $L = 2$; $M = \frac{N}{2}$ ⇒ $x[2n]$ & $x[2n+1]$

$x[n]$

$n = 2n$ | $n = 2n+1$ | Combine

First Level Division

$x[2n]$ | $x[2n+1]$

2nd Level Division

$x[4n]$ | $x[4n+2]$ | $x[4n+1]$ | $x[4n+3]$

3rd Level Division

$x[8n]$ | $x[8n+4]$ | $x[8n+2]$ | $x[8n+6]$ | $x[8n+1]$ | $x[8n+5]$ | $x[8n+3]$ | $x[8n+7]$

Sufficient if $N = 2$

Initial Sequencing :→

$x[n] = \begin{bmatrix} x(8n) \\ x(8n+4) \\ x(8n+2) \\ x(8n+6) \\ x(8n+1) \\ x(8n+5) \\ x(8n+3) \\ x(8n+7) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix}$

for $N = 8$

just put $n = 0$

→ Divide until each sequence has only 1 sample

→ No. of Levels $= v = \log_R N = \log_2 8 = 3$.

Radix-2 DIT FFT ; $N \in \{1, 2, 4, 8, \ldots\}$

$N = 1$

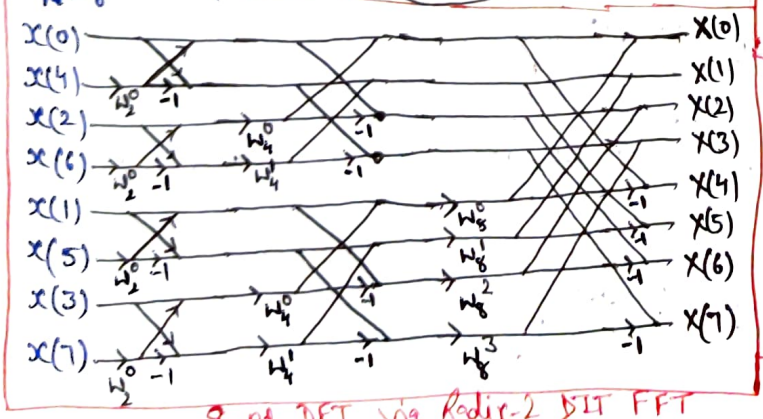
$x[0]$ ⟷ 1-pt. DFT ⟷ $X(0) = x(0)$

$n = 0$

$N = 2$

$x(0)$, $x(1)$ → 2-point DFT → $X(0) = x(0) + x(1)$, $X(1) = x(0) - x(1)$

$N = 8$

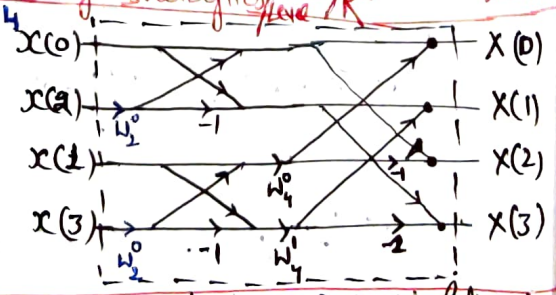


8-pt. DFT via Radix-2 DIT FFT

Note :- 1. Butterfly structure using Radix-2 DIT FFT

2. No. of butterflies/stage $= N/R$.

$N = 4$



4-point DFT via Radix-2 DIT FFT.

No. of butterflies at each level $= \frac{N}{R}$

Radix-2 DIF FFT

Approach → Divide & Combine recursively using $L = \frac{N}{2}$; $M = 2$ and using → process $x(p,m)$ column-wise Time-Series sequences; i.e. $x(l)$ & $x(l+N/2)$; $0 \le l \le \frac{N}{2}-1$

$\hookrightarrow N = R^{\nu} \Big|_{R=2}$

$$\begin{bmatrix} x[n] \\ x[n+\frac{N}{R}] \\ \vdots \\ x[n+\frac{N}{R}(R-1)] \end{bmatrix} \xrightarrow[\text{FFT}]{\text{Radix-R DIF}} \begin{bmatrix} X(Rk) \\ X(Rk+1) \\ \vdots \\ X(Rk+(R-1)) \end{bmatrix}$$

$$0 \le n, k \le \frac{N}{R} - 1$$

For $R=2$ →

$$\begin{matrix} x[n] \\ x[n+\frac{N}{2}] \end{matrix} \xrightarrow[\text{FFT}]{\text{Radix-2 DIF}} \begin{matrix} X(2k) \\ X(2k+1) \end{matrix}$$

⇓

Perfectly satisfied/fits for $N=2$

1) For $N \in \{4, 6, 8, \ldots\}$ o/p sequences will be in Nested (even-odd) even form & Nested (even-odd) odd form.

DFT

$$X(k) \triangleq \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{kn} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{kn}$$

$(n \Rightarrow m)$; $n \Rightarrow m + \frac{N}{2}$; $0 \le m \le \frac{N}{2} - 1$

$$= \sum_{m=0}^{\frac{N}{2}-1} x[m] W_N^{km} + \sum_{m=0}^{\frac{N}{2}-1} x[m+\tfrac{N}{2}] W_N^{k(m+\frac{N}{2})}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} \left(x[m] + W_N^{\frac{N}{2}} x[m+\tfrac{N}{2}] \right) W_N^{km}$$

Radix-2 DIF FFT core equation

$$\boxed{X(k) = \sum_{m=0}^{\frac{N}{2}-1} \left(x(m) + (-1)^k x(m+\tfrac{N}{2}) \right) W_N^{km}} \quad 0 \le k \le N-1 \rightarrow (C2)$$

$k \in$ Even → $X(2k_0)$; $k \in$ odd → $X(2k_0+1)$

$0 \le k_0 \le \frac{N}{2} - 1$

$$X(2k_0) = \sum_{m=0}^{\frac{N}{2}-1} \left(x(m) + x(m+\tfrac{N}{2}) \right) W_N^{2k_0 m} \qquad 0 \le k_0 \le \frac{N}{2}-1$$

$$X(2k_0) = \sum_{m=0}^{\frac{N}{2}-1} \underbrace{\left(x(m) + x(m+\tfrac{N}{2}) \right)}_{= x_1[n]} W_{\frac{N}{2}}^{k_0 m} \rightarrow (C21)$$

$\frac{N}{2}$-Point DFT of $x_1[n]$

Similarly ⇓ $k = 2k_0 + 1$ in

$$X(2k_0+1) = \sum_{m=0}^{\frac{N}{2}-1} \underbrace{\left\{ \left(x(m) - x(m+\tfrac{N}{2}) \right) W_N^{m} \right\}}_{x_2[n]} W_{\frac{N}{2}}^{k_0 m} \rightarrow (C22)$$

N/2 - Point DFT of $x_2[n]$.

$L = \frac{N}{2}$, $M = 2 \Rightarrow X(2k)$ & $X(2k+1)$

$X(k)$

$2k$ $2k+1$

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

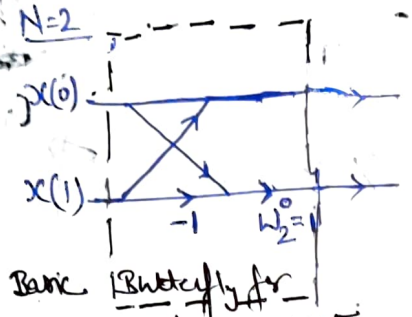
$X(4k)$ $X(4k+2)$ $X(4k+1)$ $X(4k+3)$

$X(8k)$ $X(8k+4)$ $X(8k+2)$ $X(8k+6)$ $X(8k+1)$ $X(8k+5)$ $X(8k+3)$ $X(8k+7)$

N=1

$x(0)$ — 1-pt DFT — $X(0)$

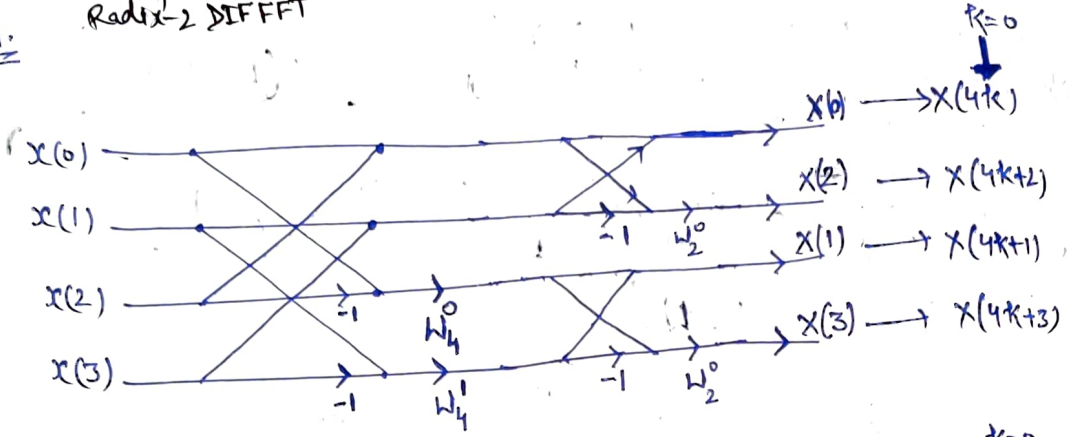
N=2



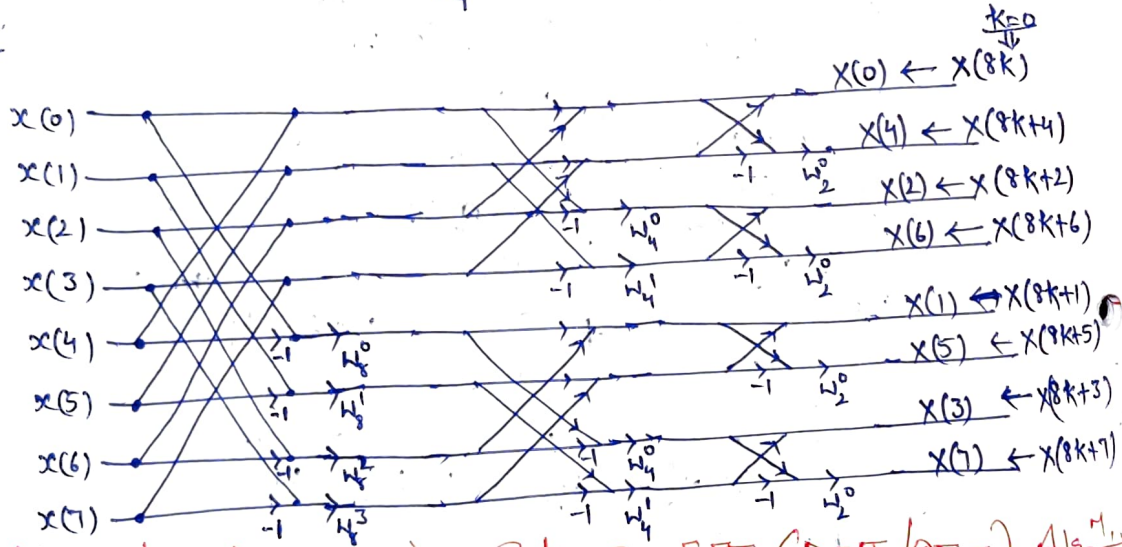
Basic Butterfly for Radix-2 DIFFFT

$0 \le k \le \frac{N}{2} - 1$

N=4



N=8



Computation of IDFT using Radix-R FFT (DIT / DIF) Algorithm

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}$$

Steps to Evaluate IDFT

Step-1: $Y(k) = \frac{X(N-k)}{N}$

Step-2: $k \to n \Rightarrow Y[n]$

Step-3: $Y[n] \xleftrightarrow{N\text{-pt. DFT}} x[k]$

using DIT/DIF (Radix-R) $\xrightarrow{k \to n}$ $x[n] : IDFT[X(k)]$.

Duality Property of DFT

$x[n] \xleftrightarrow{N\text{-DFT}} X(k)$

$X(n) \xleftrightarrow{N\text{-DFT}} N\,x[\langle -k \rangle_N]$

$$\frac{X(N-n)}{N} = \frac{X[\langle -n \rangle_N]}{N} \xleftrightarrow{N\text{-DFT}} x(k)$$

$k \to n$

$x[n] = IDFT[X$

Radix-r $\Rightarrow$ FFT

Computational Complexity

For radix-r system; i.e. $N = r^v$; $v = \log_r N$

- N → Total no. of samples or prints
- r → samples per butterfly
- v → Total No. of stages involved.

e.g. $N = 8 = 2^3 \Rightarrow v = 3$; $r = 2$; $\frac{N}{r} = 4$ butterflies per stage

(No. of samples per butterfly; No. of stages)

&

$\frac{N}{r}$ → No. of butterfly per stage

No. of Complex Multiplication per butterfly $= (r-1)$ [e.g. $= 1 \mid r=2$]

No. of complex addition per butterfly $= r(r-1)$ [e.g. $= 2 \mid r=2$]

GATE 2010

Hence

Total No. of Complex Multiplications or Addition using FFT = No. of Mult./add. per butterfly × No. of butterfly per stage × total stages

Total No. of Complex Multiplications $= \frac{(r-1)}{r} \times \frac{N}{r} \times \log_r N$ $\Rightarrow \frac{r-1}{r} N \log_r N$

Total No. of " Additions $= r(r-1)\frac{N}{r}\log_r N \Rightarrow (r-1) \cdot N \cdot \log_r N$