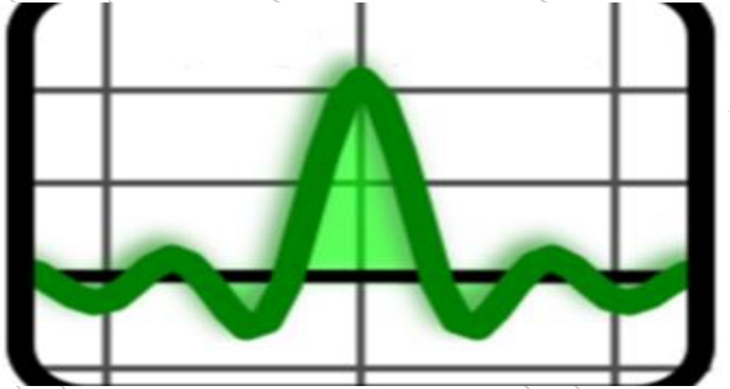


Signals & Systems For Computer Engineering

Prof.Dr. B.Berk ÜSTÜNDAĞ
Istanbul Technical University
Faculty of Computer Engineering and Informatics
bustundag@itu.edu.tr

BLG354E / CRN: 21162
13th Week Lecture

FAST FOURIER TRANSFORM (FFT) ALGORITHM

Let the sequences f[n] represent the even-numbered and g[n] represent the odd-numbered samples of x[n]

$$f[n] = x[2n]$$

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

Since N-point DFT of x[n] is, $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$ $\quad W_N = e^{-j(2\pi/N)}$ $\quad k = 0, 1, \ldots, N-1$

If x[n] is a sequence of <u>even</u> and finite length N where x[n]=0 n<0, n≥N then

$$f[n] = x[2n] = 0 \quad n < 0 \qquad f\left[\frac{N}{2}\right] = x[N] = 0 \quad \rightarrow \quad f[n] = 0 \quad n < 0, n \geq \frac{N}{2}$$

$$g[n] = x[2n+1] = 0, \quad n < 0 \qquad g\left[\frac{N}{2}\right] = x[N+1] = 0 \quad \rightarrow \quad g[n] = 0 \quad n < 0, n \geq \frac{N}{2}$$

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn} = \sum_{m=0}^{(N/2)-1} x[2m] W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x[2m+1] W_N^{(2m+1)k}$$

Since $W_N^2 = \left(e^{-j(2\pi/N)}\right)^2 = e^{-j(4\pi/N)} = e^{-j(2\pi/N/2)} = W_{N/2}$

$$\rightarrow \quad X[k] = \sum_{m=0}^{(N/2)-1} f[m]W_{N/2}^{mk} + W_N^k \sum_{m=0}^{(N/2)-1} g[m]W_{N/2}^{mk}$$

$\rightarrow$ X[k] $= F[k] + W_N^k G[k] \qquad k = 0, 1, \ldots, N-1$

$$\left\{ \begin{aligned} F[k] &= \sum_{n=0}^{(N/2)-1} f[n]W_{N/2}^{kn} \\ G[k] &= \sum_{n=0}^{(N/2)-1} g[n]W_{N/2}^{kn} \end{aligned} \right. \qquad k = 0, 1, \ldots, \frac{N}{2}-1$$

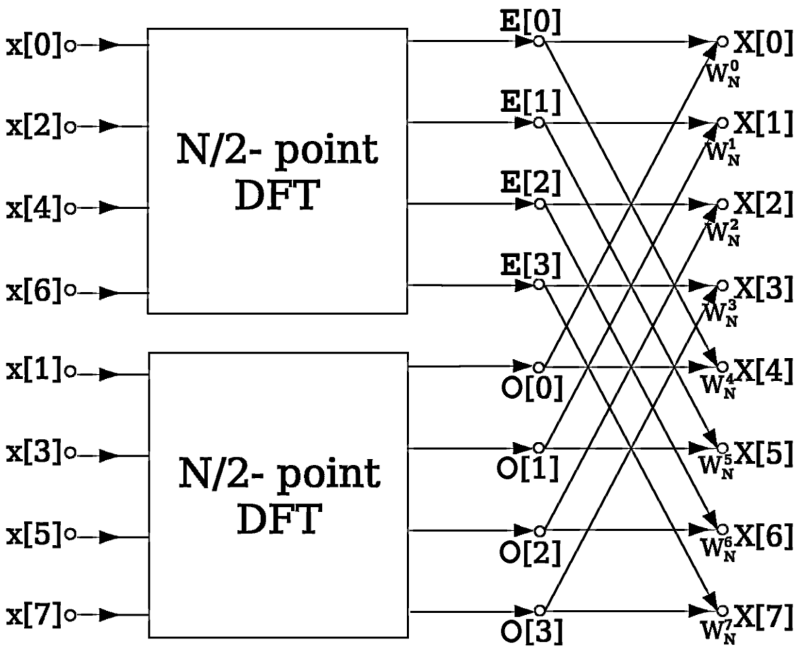
Where, F[k] is (N/2)-point DFTs of f[k]
G[k] is (N/2)-point DFTs of g[k]

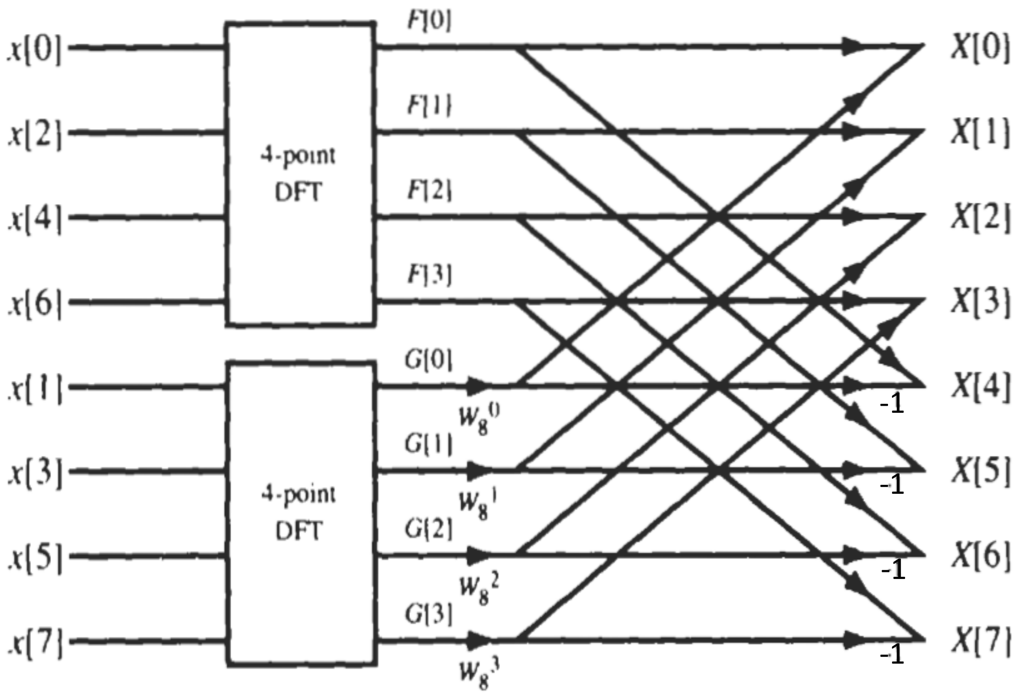
Since $W_N^{N/2} = (e^{-j(2\pi/N)})^{(N/2)} = e^{-j\pi} = -1 \rightarrow W_N^{k+N/2} = W_N^k W_N^{N/2} = -W_N^k$

$$X[k] = F[k] + W_N^k G[k] \qquad k = 0, 1, \ldots, \frac{N}{2}-1$$

$$X\left[k + \frac{N}{2}\right] = F[k] - W_N^k G[k] \qquad k = 0, 1, \ldots, \frac{N}{2}-1$$

Flow graph for an 8-point decimation-in-time FFT algorithm





Time Complexity Reduction:

Total number of complex multiplications based on $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$ is N^2

Number of complex multiplications in evaluating F[k] or G[k] is $(N/2)^2$

Total number of complex multiplications of N-point decimation-in-time algorithm is $2(N/2)^2 + N = N^2/2 + N$

Example:

x[n]={1, 1, -1, -1, -1, 1, 1, -1}   find DFT of x[n] by using decimation-in-time FFT algorithm

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}$$

$$W_N = e^{-j(2\pi/N)}$$

$$\mathbf{W}_N^T = \mathbf{W}_N$$

W_4^k and W_8^k are

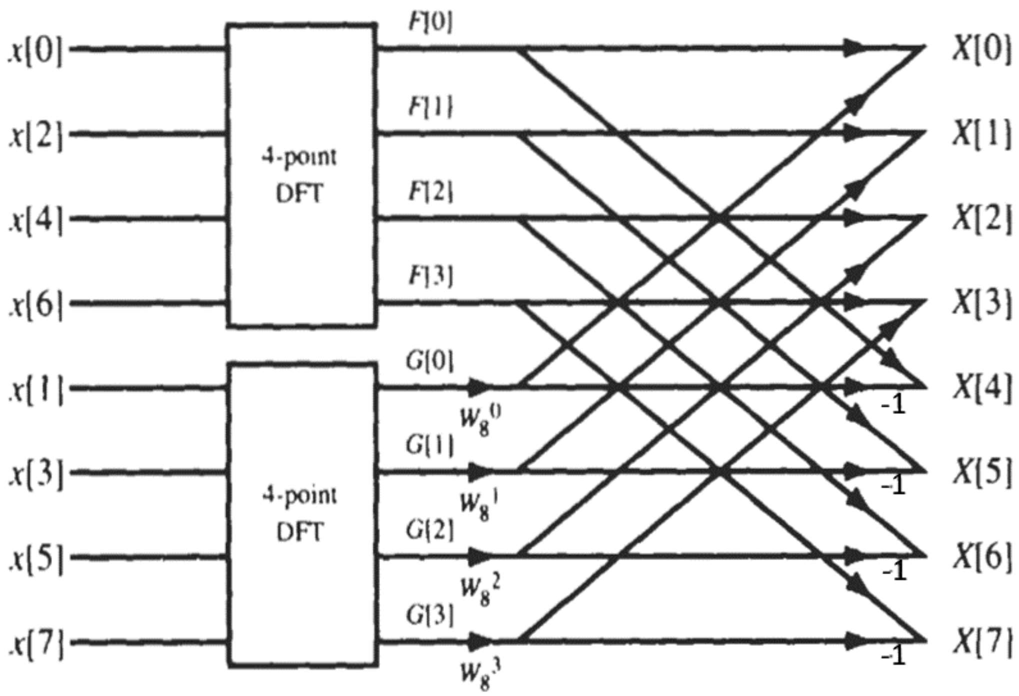
$$\begin{cases} W_4^0 = 1 \quad W_4^1 = -j \quad W_4^2 = -1 \quad W_4^3 = j \\ W_8^0 = 1 \quad W_8^1 = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \quad W_8^2 = -j \quad W_8^3 = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \\ W_8^4 = -1 \quad W_8^5 = -\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}} \quad W_8^6 = j \quad W_8^7 = \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}} \end{cases}$$

$$f[n] = x[2n] = \{x[0], x[2], x[4], x[6]\} = \{1, -1, -1, 1\}$$

$$g[n] = x[2n+1] = \{x[1], x[3], x[5], x[7] = \{1, -1, 1, -1\}$$

$$\begin{bmatrix} F[0] \\ F[1] \\ F[2] \\ F[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2+j2 \\ 0 \\ 2-j2 \end{bmatrix} \qquad \begin{bmatrix} G[0] \\ G[1] \\ G[2] \\ G[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} F[0] \\ F[1] \\ F[2] \\ F[3] \end{bmatrix} = \begin{bmatrix} 0 \\ 2+j2 \\ 0 \\ 2-j2 \end{bmatrix} \qquad \begin{bmatrix} G[0] \\ G[1] \\ G[2] \\ G[3] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \\ 0 \end{bmatrix}$$



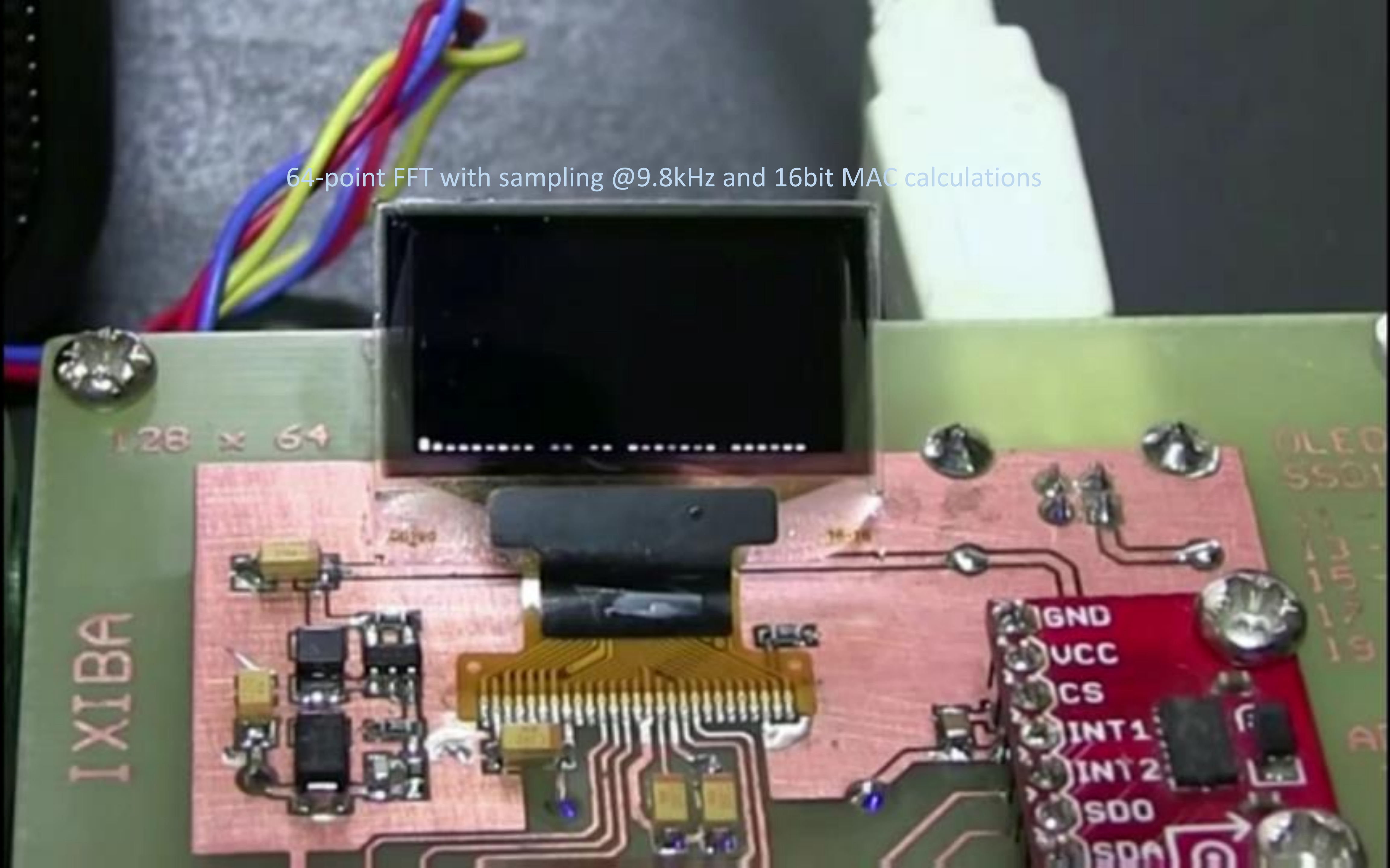
$$X[0] = F[0] + W_8^0 G[0] = 0$$
$$X[1] = F[1] + W_8^1 G[1] = 2 + j2$$
$$X[2] = F[2] + W_8^2 G[2] = -j4$$
$$X[3] = F[3] + W_8^3 G[3] = 2 - j2$$
$$X[4] = F[0] - W_8^0 G[0] = 0$$
$$X[5] = F[1] - W_8^1 G[1] = 2 + j2$$
$$X[6] = F[2] - W_8^2 G[2] = j4$$
$$X[7] = F[3] - W_8^3 G[3] = 2 - j2$$

X[5], X[6], and X[7] are conjugates of X[3], X[2], and X[1]

Comparison of computational complexity for the direct computation of DFT versus the FFT algorithm

No of Points	Direct Computation		DIT-FFT algorithm		Speed improvement for multiplications $\frac{N^2}{N/2 log_2 N}$
	Complex multiplications N^2	Complex additions $N^2 - N$	Complex multiplications $N/2 log_2 N$	Complex additions $N log_2 N$	
4	16	12	4	8	4 times
8	64	56	12	24	5.3 times
16	256	240	32	64	8 times
32	1024	992	80	160	12.8 times
64	4096	4032	192	384	21.3 times
256	65536	65280	1024	2048	64.0 times
1024	1048576	1047552	5120	10240	204.8 times

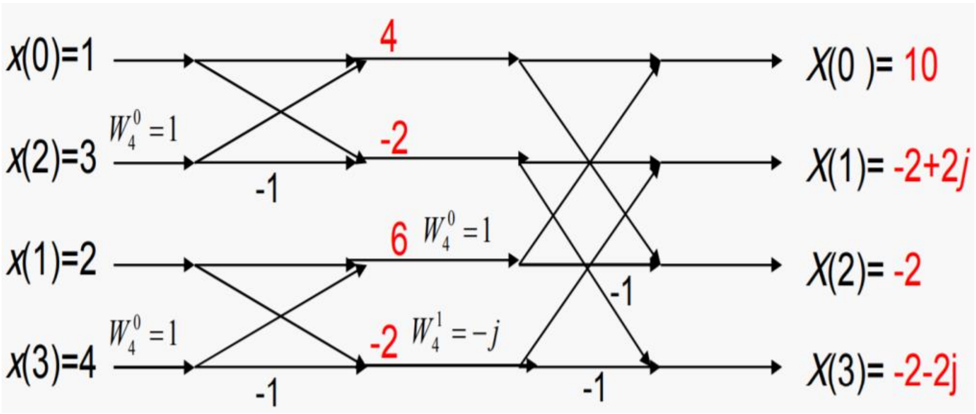
64-point FFT with sampling @9.8kHz and 16bit MAC calculations

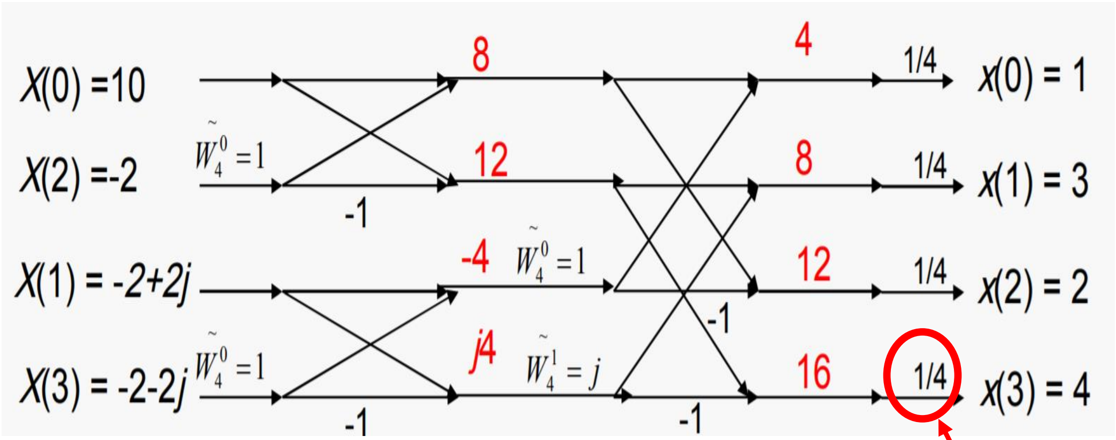


Example:

Inverse DIT-FFT

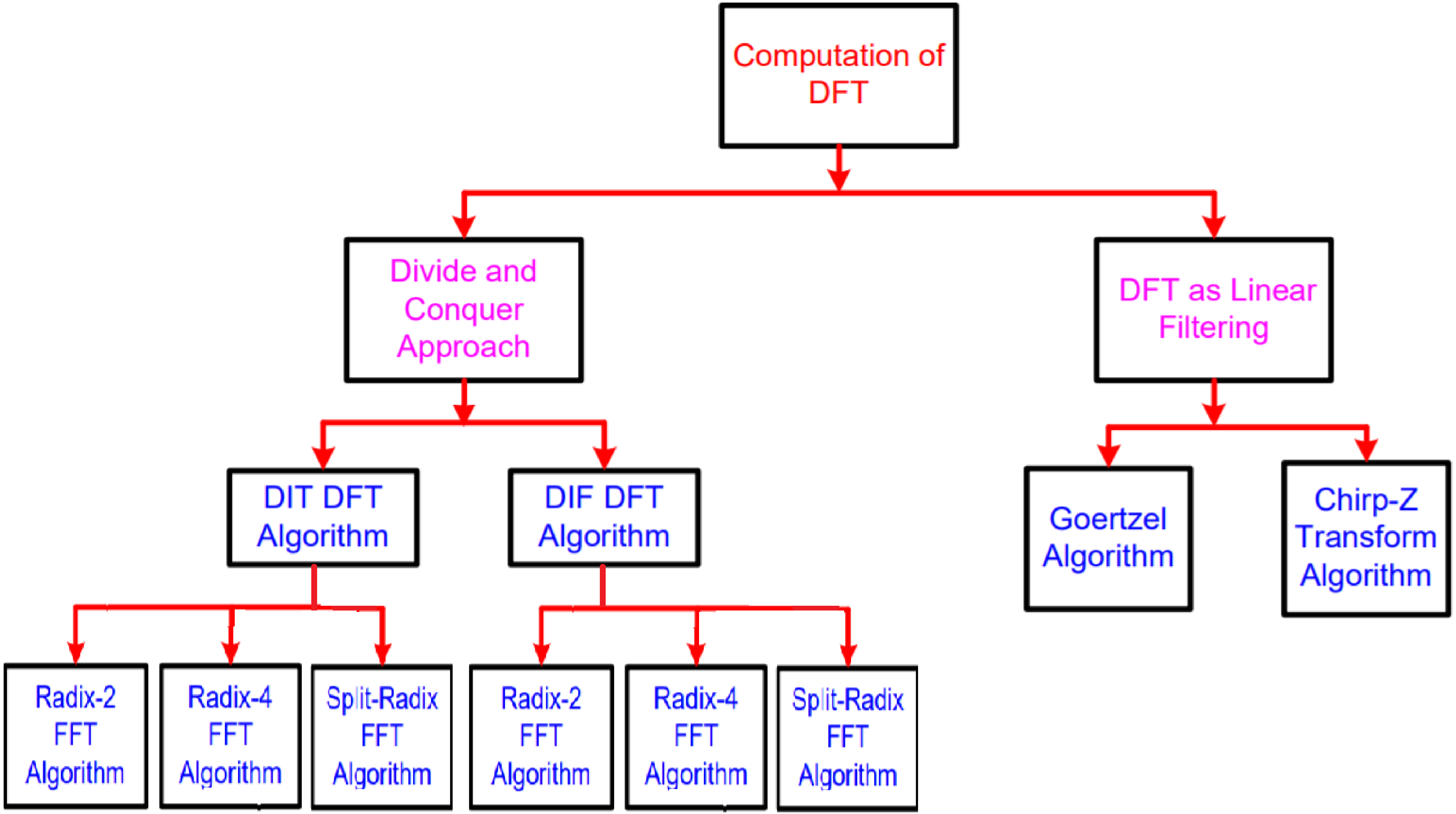
A signal sequence x[n] is given as x[n]={1, 2, 3, 4} and x[n] = 0 elsewhere. DFT for the first four points and show that IFFT recovers the signal from the spectral representation.





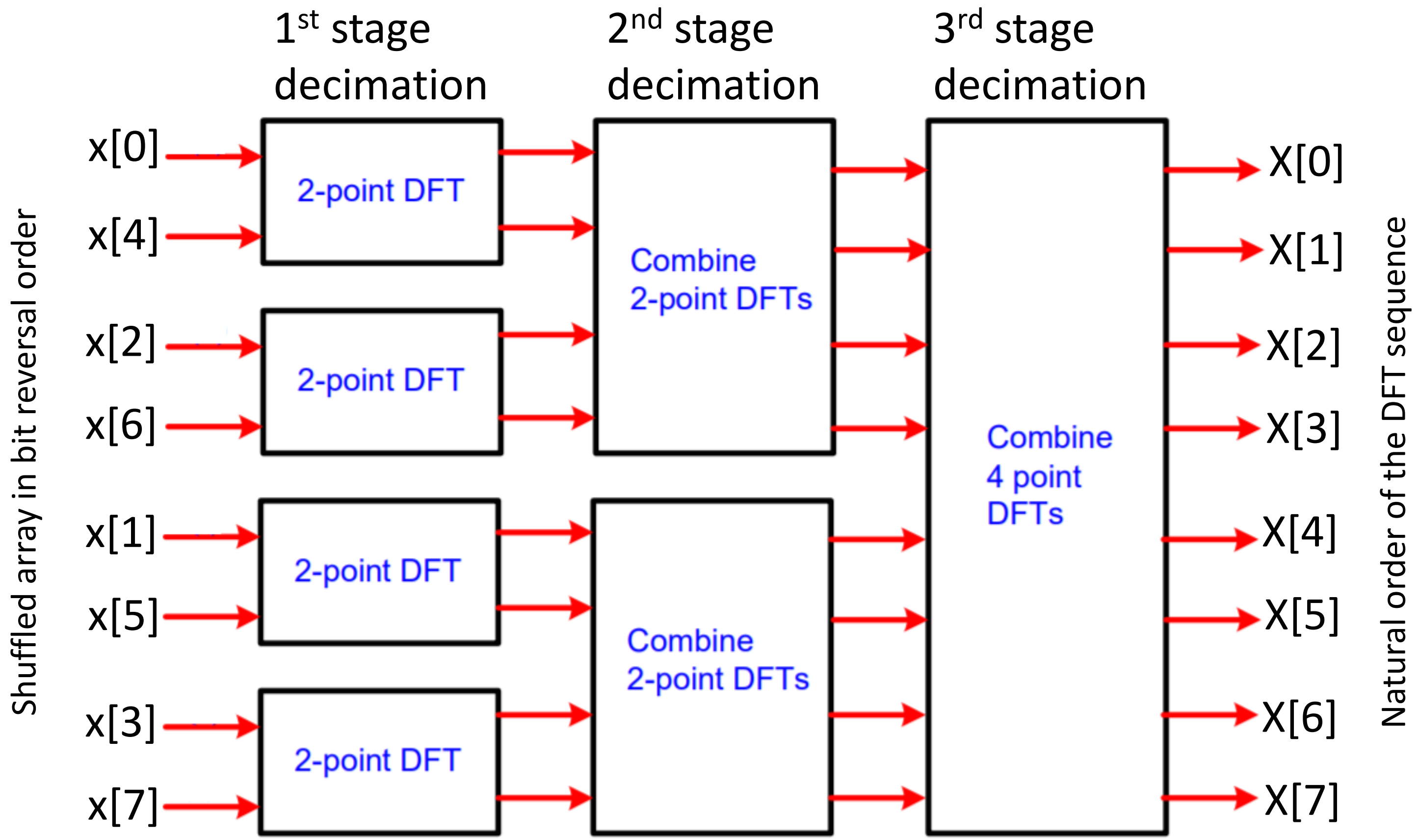
Inverse DFT can be calculated using the same method by changing the variable W_N and multiplying the result by 1/N

FFT Algorithms



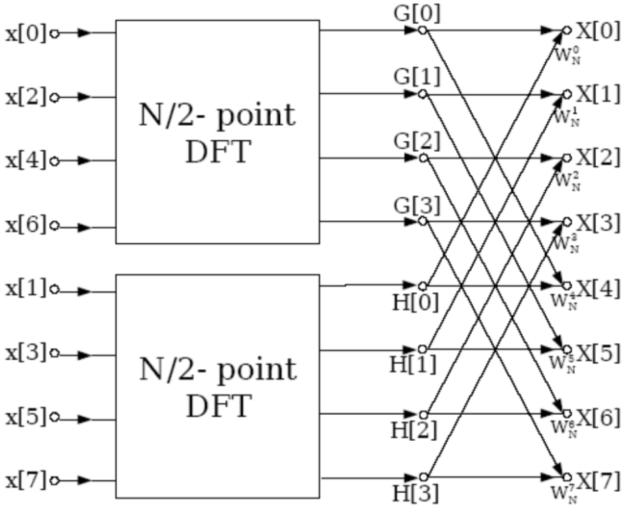
Radix-2 Algorithm

- Radix-2 is the most widely used FFT algorithm
- The sequence x[n] of length N is factored in such a way that $N = r_1 r_2 r_3 \ldots r_v$
- $r_1 = r_2 = r_3 = \ldots r_v = r$ so that $N = r^v$, where r is called the radix of the FFT algorithm.
- r = 2 is called radix-2 algorithm

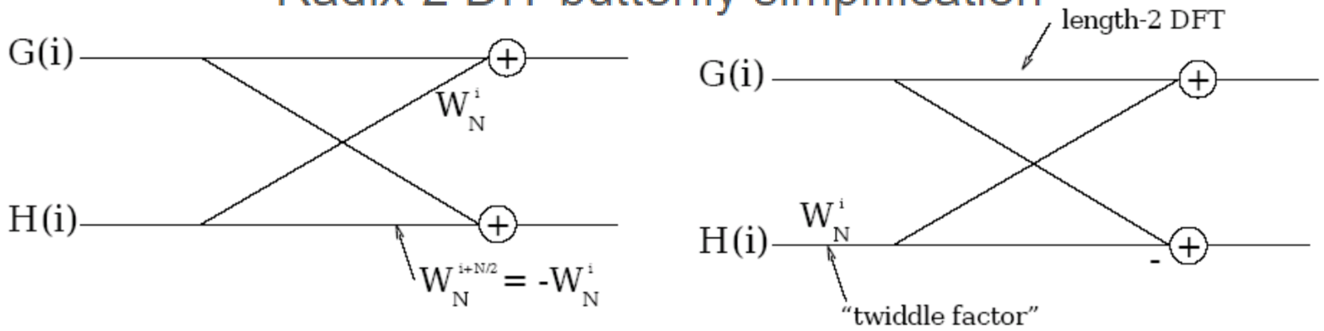


Radix-2 decimation-in-time FFT

Decimation in time of a length-N DFT into two length-$\frac{N}{2}$ DFTs followed by a combining stage



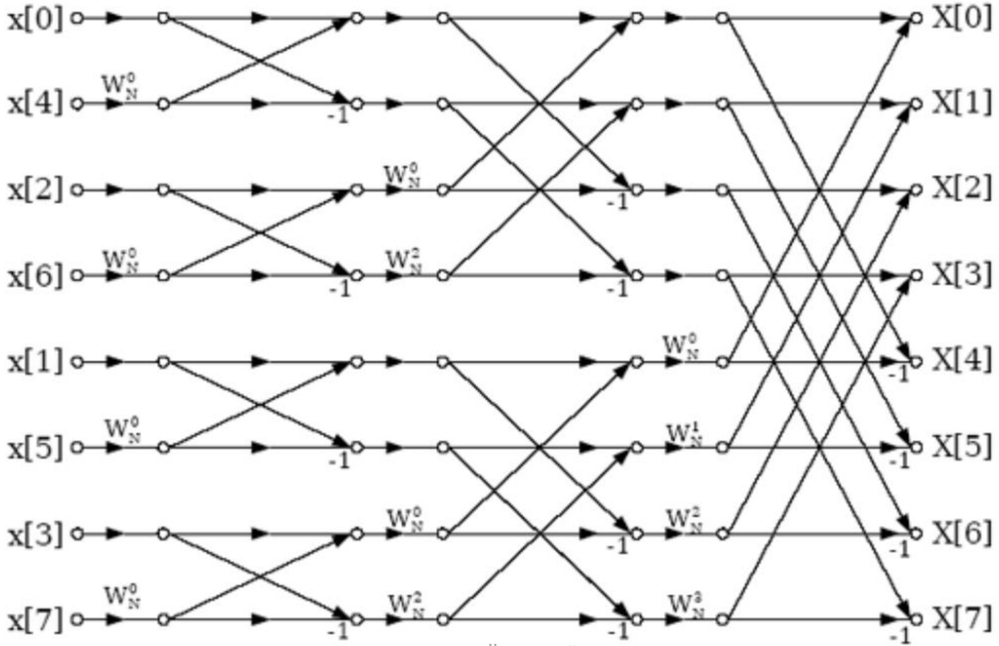
Radix-2 DIT butterfly simplification

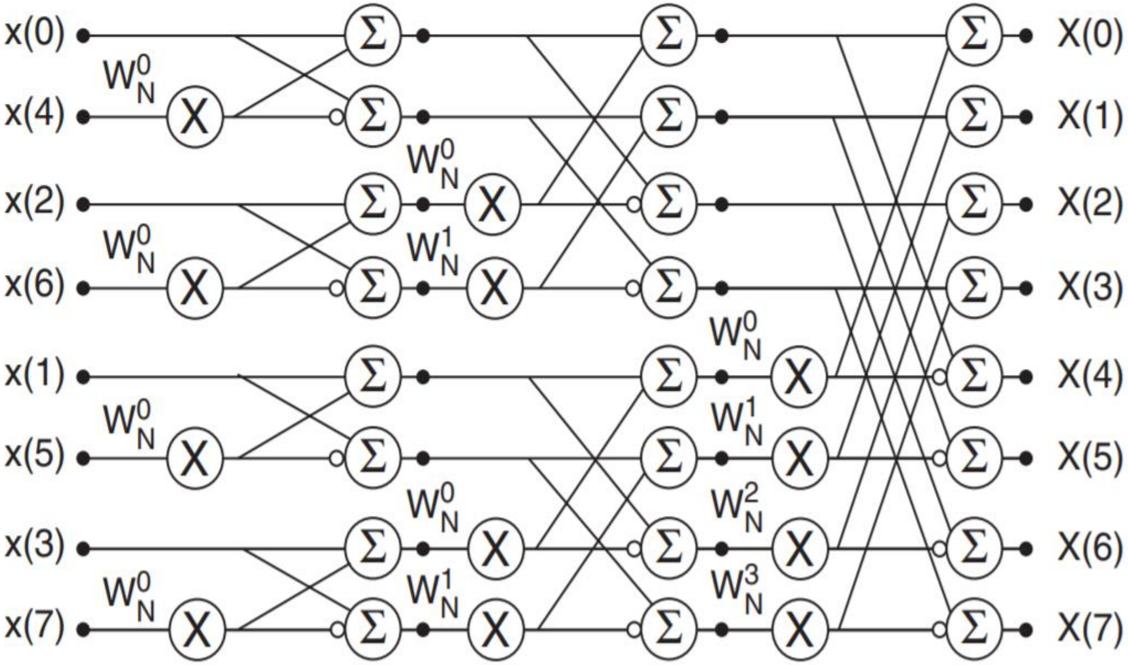


Computational cost of radix-2 DIT FFT

- $\frac{N}{2}\log_2 N$ complex multiplies
- $N \log_2 N$ complex adds

Radix-2 Decimation-in-Time FFT algorithm for a length-8 signal





Example: Find the DFT of the sequence x[n]={1,1,1,1,0,0,0,0} by using 8 point radix-2 DIT-FFT algorithm

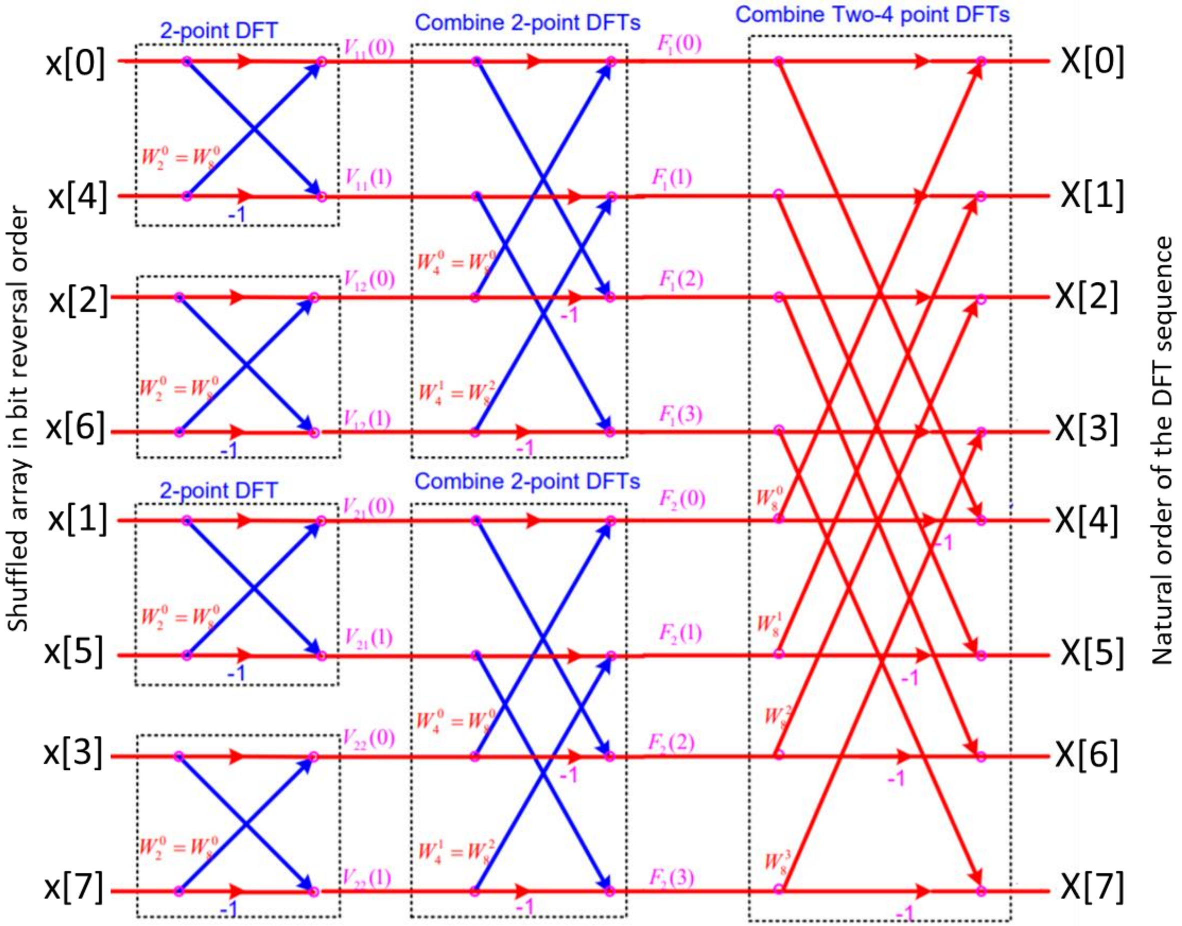
$$
\begin{aligned}
V_{11}(0) &= x(0) + W_8^0 x4 = 1 + 1(0) = 1 \\
V_{11}(1) &= x(0) - W_8^0 x4 = 1 - 1(0) = 1 \\
V_{12}(0) &= x(2) + W_8^0 x6 = 1 + 1(0) = 1 \\
V_{12}(1) &= x(2) - W_8^0 x6 = 1 - 1(0) = 1 \\
V_{21}(0) &= x(1) + W_8^0 x5 = 1 + 1(0) = 1 \\
V_{21}(1) &= x(1) - W_8^0 x5 = 1 - 1(0) = 1 \\
V_{22}(0) &= x(3) + W_8^0 x7 = 1 + 1(0) = 1 \\
V_{22}(1) &= x(1) - W_8^0 x5 = 1 - 1(0) = 1
\end{aligned}
$$

$$\begin{aligned} F_1(0) &= V_{11}(0) + W_8^0 V_{12}(0) \\ &= 1 + 1(1) = 2 \\ F_1(1) &= V_{11}(1) + W_8^0 V_{12}(1) \\ &= 1 + (-j)1 = 1 - j \end{aligned}$$

$$\begin{aligned} F_1(2) &= V_{11}(0) - W_8^0 V_{12}(0) \\ &= 1 - (-j)(1) = 1 - j \\ F_1(3) &= V_{11}(1) - W_8^0 V_{12}(1) \\ &= 1 - (-j)1.414 = 1 + j \end{aligned}$$

$$\begin{aligned} F_2(0) &= V_{21}(0) + W_8^0 V_{22}(0) \\ &= 1 + 1(1) = 2 \\ F_2(1) &= V_{21}(1) + W_8^0 V_{22}(1) \\ &= 1 + (-j)1 = 1 - j \end{aligned}$$

$$\begin{aligned} F_2(2) &= V_{21}(0) - W_8^0 V_{22}(0) \\ &= 1 - 1(1) = 0 \\ F_2(3) &= V_{21}(1) - W_8^0 V_{22}(1) \\ &= 1 - (-j)1 = 1 + j \end{aligned}$$



$$\begin{aligned} X(0) &= F_1(0) + W_8^0 F_2(0) \\ &= 2 + 1(2) = 4 \\ X(1) &= F_1(1) + W_8^1 F_2(1) \\ &= (1 - j1) + (0.707 - j0.7071)(1 - j) = \quad 1 - j2.414 \\ X(2) &= F_1(2) + W_8^2 F_2(2) \\ &= 0 + (-j)(0) = 0 \\ X(3) &= F_1(3) + W_8^3 F_2(3) \\ &= (1 + j) + (-0.7071 - j0.7071)(1 + j) = \quad 1 - j0.414 \\ X(4) &= F_1(0) - W_8^0 F_2(0) \\ &= 2 - 1(2) = 0 \\ X(5) &= F_1(1) - W_8^1 F_2(1) \\ &= (1 - j1) - (0.707 - j0.7071)(1 - j) = \quad 1 - j0.414 \\ X(6) &= F_1(2) - W_8^2 F_2(2) \\ &= 0 - (-j)(0) = 0 \\ X(7) &= F_1(3) - W_8^3 F_2(3) \\ &= (1 + j) - (-0.7071 - j0.7071)(1 + j) \end{aligned}$$

Decimation in Frequency FFT Algorithm

N-point DFT of x[n] is, $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$ $\qquad W_N = e^{-j(2\pi/N)}$ $\qquad k = 0, 1, \ldots, N-1$

If x[n] is a sequence of <u>even</u> and finite length N where x[n]=0 n<0, n≥N then

$$p[n] = x[n] + x\left[n + \frac{N}{2}\right] \qquad 0 \le n < \frac{N}{2}$$

$$q[n] = \left(x[n] - x\left[n + \frac{N}{2}\right]\right) W_N^n \qquad 0 \le n < \frac{N}{2}$$

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

variable change
$n = m + N/2$

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

$$W_N^{N/2} = (e^{-j(2\pi/N)})^{(N/2)} = e^{-j\pi} = -1 \rightarrow W_N^{(N/2)k} = (-1)^k \rightarrow X[k] = \sum_{n=0}^{(N/2)-1} \left\{ x[n] + (-1)^k x\left[n + \frac{N}{2}\right] \right\} W_N^{kn}$$

N/2 point DFT of p[n]

If k is even then by setting k=2r → $X[2r] = \sum_{m=0}^{(N/2)-1} p[n] W_N^{2rn} = \sum_{n=0}^{(N/2)-1} p[n] W_{N/2}^{rn} \qquad r = 0, 1, \ldots, \frac{N}{2} - 1$

N/2 point DFT of q[n]

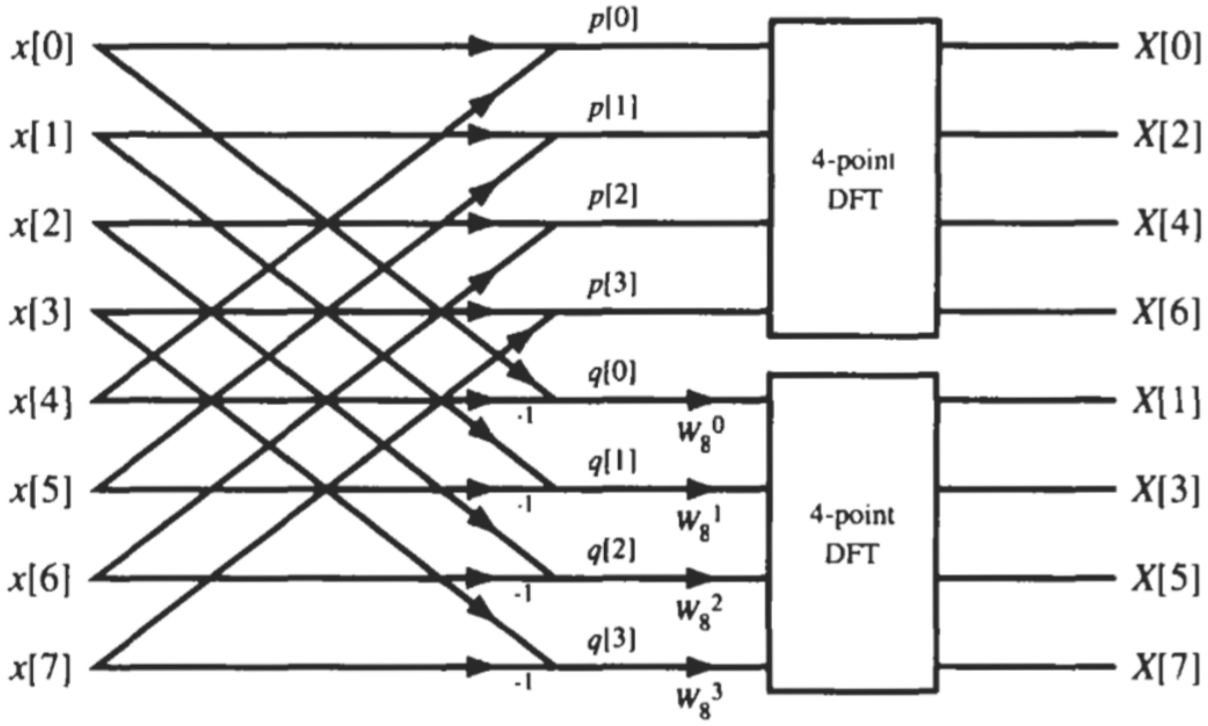
If k is odd then by setting k=2r+1 → $X[2r+1] = \sum_{m=0}^{(N/2)-1} q[n] W_N^{2rn} = \sum_{n=0}^{(N/2)-1} q[n] W_{N/2}^{rn} \qquad r = 0, 1, \ldots, \frac{N}{2} - 1$

If r is replaced by k then:

$X[2k] = P[k] \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$ where $P[k] = \sum_{n=0}^{(N/2)-1} p[n] W_{N/2}^{kn} \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$

$X[2k+1] = Q[k] \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$ where $Q[k] = \sum_{n=0}^{(N/2)-1} q[n] W_{N/2}^{kn} \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$

DIF-FFT Graph:



Comparison of DIT-FFT and DIF-FFT:

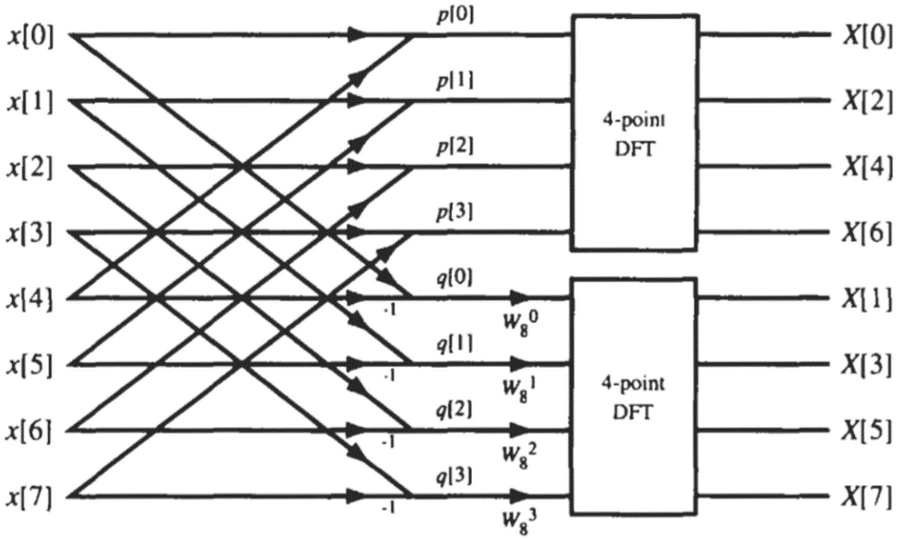
a) DIT-FFT algorithm reduces the number of complex multiplications required from N^2 to $N \cdot \log_2(N)$, whereas DIF-FFT algorithm reduces the number of complex multiplications from N^2 to $(N/2) \cdot \log_2(N)$
b) Input is bit reversed in DIT-FFT while the output is in natural order, whereas in DIF-FFT, input is in natural order while the output is in bit reversal order.
c) DIT-FFT refers to reducing samples in time domain, whereas DIF-FFT refers to reducing samples in frequency domain.
d) DIT-FFT splits the two DFTs into even and odd indexed input samples, whereas DIF-FFT splits the two DFTs into first half and last half of the input samples.
e) In DIT-FFT, butterflies are defined on the last pass of FFT, whereas in DIF-FFT, they are defined on the first pass of FFT.

Example:

$x[n]=\{1, 1, -1, -1, -1, 1, 1, -1\}$ find DFT of x[n] by using decimation-in-frequency (DIF) FFT algorithm

W_4^k and W_8^k are

$$\begin{cases} W_4^0 = 1 \quad W_4^1 = -j \quad W_4^2 = -1 \quad W_4^3 = j \\ W_8^0 = 1 \quad W_8^1 = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \quad W_8^2 = -j \quad W_8^3 = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \\ W_8^4 = -1 \quad W_8^5 = -\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}} \quad W_8^6 = j \quad W_8^7 = \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}} \end{cases}$$



$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}$$

$$W_N = e^{-j(2\pi/N)}$$

$$\mathbf{W}_N^T = \mathbf{W}_N$$

$$p[n] = x[n] + x\left[n + \frac{N}{2}\right] = \{(1-1), (1+1), (-1+1), (-1-1)\} = \{0, 2, 0, 2\}$$

$$q[n] = \left(x[n] - x\left[n + \frac{N}{2}\right]\right) W_8^n = \{(1+1)W_8^0, (1-1)W_8^1, (-1-1)w_8^2, (-1+1)W_8^3\} = \{2, 0, j2, 0\}$$

$$\begin{bmatrix} P[0] \\ P[1] \\ P[2] \\ P[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ -j4 \\ 0 \\ j4 \end{bmatrix}$$

$$\begin{bmatrix} Q[0] \\ Q[1] \\ Q[2] \\ Q[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ j2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2+j2 \\ 2-j2 \\ 2+j2 \\ 2-j2 \end{bmatrix}$$

$$\begin{aligned} X[0] &= P[0] = 0 \\ X[1] &= Q[0] = 2 + j2 \\ X[2] &= P[1] = -j4 \\ X[3] &= Q[1] = 2 - j2 \\ X[4] &= P[2] = 0 \\ X[5] &= Q[2] = 2 + j2 \\ X[6] &= P[3] = j4 \\ X[7] &= Q[3] = 2 - j2 \end{aligned}$$

Example:

x[n]={1, -1, -1, -1, 1, 1, 1, -1} find DFT of x[n] by using Radix-2 decimation-in-frequency FFT algorithm

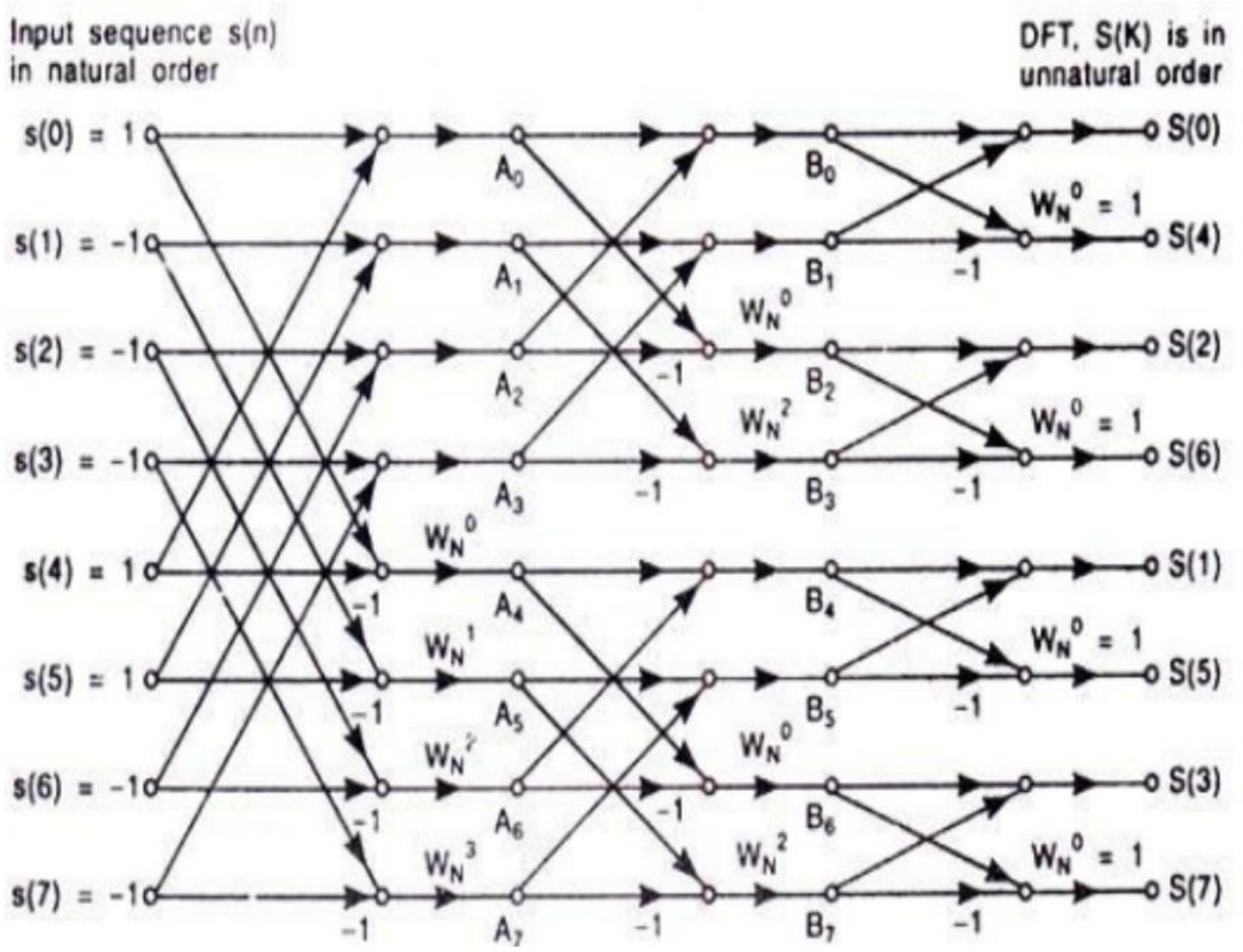
W_N = Phase roration factor $e^{-j2\pi/N}$

$W_8^0 = e^{-j(2\pi/8)0} = e^0 = 1$

$W_8^1 = e^{-j(2\pi/8)1} = e^{-j\pi/4} = \frac{1-j}{\sqrt{2}}$

$W_8^2 = e^{-j(2\pi/8)2} = e^{-j\pi/2} = -j$

$w_8^3 = e^{-j(2\pi/8)3} = e^{-j3\pi/4} = \frac{-(1+j)}{\sqrt{2}}$

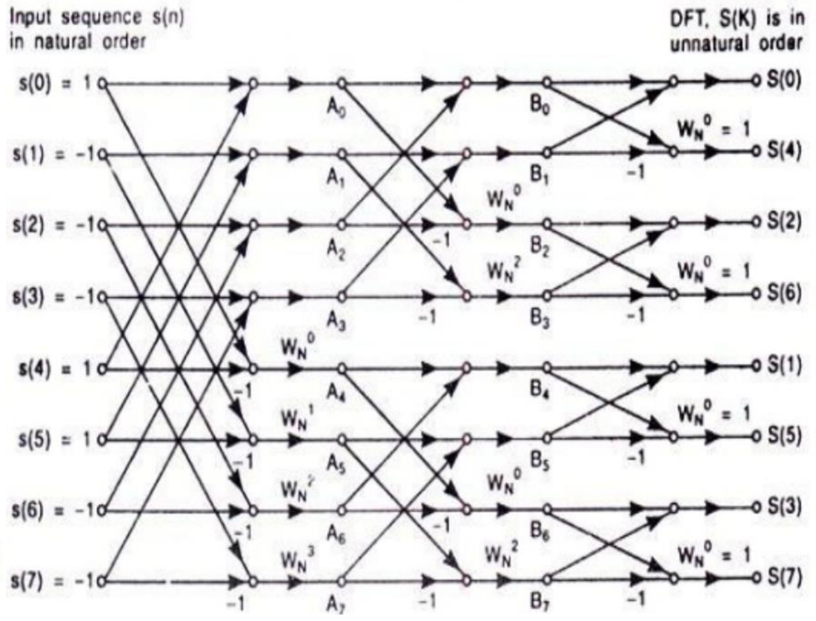


Stage1:

$$A_0 = s(0) + s(4) = 1 + 1 = 2$$
$$A_1 = s(1) \quad + s(5) = -1 + 1 = 0$$
$$A_2 = s(2) + s(6) = -1 + 1 = 0$$
$$A_3 = s(3) + s(7) = -1 - 1 = -2$$

$$A_4 = [s(0)+(-1)\ s(4)]\ W_8^{\,0} \ =0$$
$$A_5 = [s(1) + (-1)\ s(5)]W_8^{\,1} \ =-\sqrt{2}(1-j)$$
$$A_6 = [s(2) + (-1)\ s(6)]W_8^{\,2} \ =2j$$

A7=0



Stage3:

$$S(0) = B_0 + B_1 = 2 + (-2) = 0$$

$$S(4) = B_0 + (-1)\ B_1 = 2 + (-1)\ (-2) = 4$$

$$S(2) = B_2 + B_3 = 2 + (-2j) = 2-2j$$
$$S(6) = B_2 + (-1)B_3 = 2 + (-1)\ (-2j) = 2 + 2j$$

$$S(1) = B_4 + B_5 = 2j + [-\sqrt{2}(1-j)] = 2j - \sqrt{2} + \sqrt{2j} = \sqrt{2}+(2+\sqrt{2})j$$

$$S(5) = B_4 + (-1)B_5 = 2j + (-1)[-\sqrt{2}(1-j)] = 2j + \sqrt{2} - \sqrt{2j} \ = \sqrt{2} + (2-\sqrt{2})j$$

$$S(3) = B_6 + B_7 = -2j + \sqrt{2}(l+j) = -2j + \sqrt{2} + \sqrt{2j} = \sqrt{2} + (-2+\sqrt{2})\ j$$

$$S(7) = B_6 + (-1)B_7 = -2j + (-1)\sqrt{2}(1+j) = -2j - \sqrt{2} - \sqrt{2}j$$

Stage2:

$$B_0 = A_0 + A_2 = 2 + 0 = 2$$
$$B_1 = A_1 + A_3 = 0 + (-2) = -2$$
$$B_2 = [A_0 + (-1)A_2]W_8^0 = [2-0] \times 1 = 2$$
$$B_3 = [A_1 + (-1)A_3]\ W_8^2 = [0 + (-1)\ (-2)] \times (-j) = -2j$$
$$B_4 = A_4 + A_6 = 0 + 2j = 2j$$
$$B_5 = A_5 + A_7 = [-\sqrt{2}(l-j)] + 0 = -\sqrt{2}(l-j)$$
$$B_6 = [A_4 + (-1)\ A_6] = [0 + (-1)\ 2j] \times 1 = -2j$$
$$B_7 = [A_5 + (-1)A_7]W_8^2 = [-\sqrt{2}(1-j) + (-1) \times 0] \times (-j) = \sqrt{2}(1+j)$$

S(k)={S(0), S(1), S(2), S(3), S(4), S(5), S(6), S(7)}

S(k)={0, √2+(2+ √2)j, 2-2j, √2+(-2+ √2)j, 4, √2+(2- √2)j, 2+2j, - √2-(2+ √2)j}

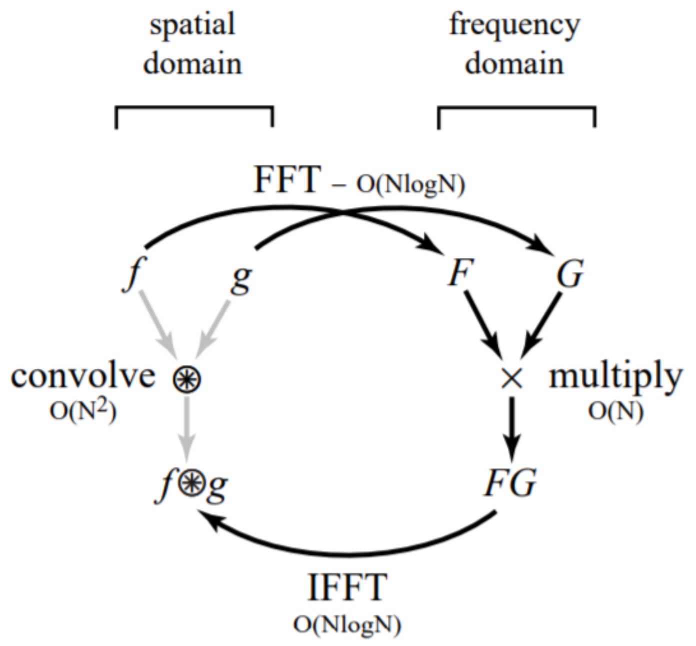
Time Complexity Advantage of DFT Convolution

Definition of convolution simplifies when working with finite sequences. If we assume that f and g have the same length N and they are periodic (f and g "wrap around") then we get circular convolution:

$$h[x] = \sum_{t=0}^{N-1} f[t]g[x - t \bmod N] \qquad \text{For x= 0 to N-1}$$

Fourier transform of the convolution of two signals is the product of their Fourier transforms: f∗g ↔ FG.
DFT of the circular convolution of two signals is the product of their DFT'

Convolution computing by its mathematical definition with a straightforward algorithm is expensive.
It requires N^2 multiplies and adds (MACs).



◊ If we use the FFT algorithm, then the two DFT's and one inverse DFT have a total cost of 6N log N real multiplies

◊ Multiplication of transforms in the frequency domain has a negligible cost of 4N multiplies.

◊ Fourier convolution has time complexity advantage for large N. <u>FFT improves this advantage rate</u>.

◊ Circular convolution algorithm can be modified to do standard "linear" convolution by padding the sequences with zeros.