Computation of DFT using FFT algorithm

- FFT is a highly efficient procedure for computing the DFT of the finite series & requires less no. of computations than that of direct evaluation of DFT.

DFT Transform Pair

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

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

Twiddle factor → $W_N = e^{-j\frac{2\pi}{N}}$

Why FFT is needed?

DFT

- To evaluate N-point DFT
- It requires
- N(N+1) complex additions
- N^2 complex multiplications

FFT

- To evaluate N-point DFT
- It requires
- $N \log_2 N$ complex additions
- $\frac{N}{2} \log_2 N$ complex multiplications

What is radix-2 FFT?

- If the number of output points N can be expressed as a power of 2, i.e $N= 2^M$,M is integer.

- Then this algorithm is known as radix-2 FFT algorithm.

Advantages of FFT over DFT

- FFT is the algorithm used to compute DFT fast
- Computationally efficient than direct computation of DFT
- Exploit periodicity and symmetry properties of DFT
- It makes use of the periodicity and symmetry properties of twiddle factor W_N^k to reduce the DFT computation time.

Applications of FFT

- Linear filtering
- Correlation
- Spectrum Analysis

- Properties of Twiddle factor:

Symmetry Property

$$W_N^{k+N/2} = -W_N^k$$

Periodicity Property

$$W_N^{k+N} = W_N^k$$

Radix-2 DIT FFT algorithm

- The N-point DFT of a sequence x(n) converts the time domain N point sequence x(n) to a frequency domain N-point sequence X(k).
- In DIT the x(n) is decimated and smaller point DFTs are performed.
- The results of smaller point DFTs are combined to get the result of N-point DFT.
- Here, the N-point DFT can be realized from two no. of N/2 point DFTs, the N/2 point DFT can be realized from two no. of N/4 point DFTs and so on.

Radix-2 DIT FFT algorithm Cont..

- **Normal order**

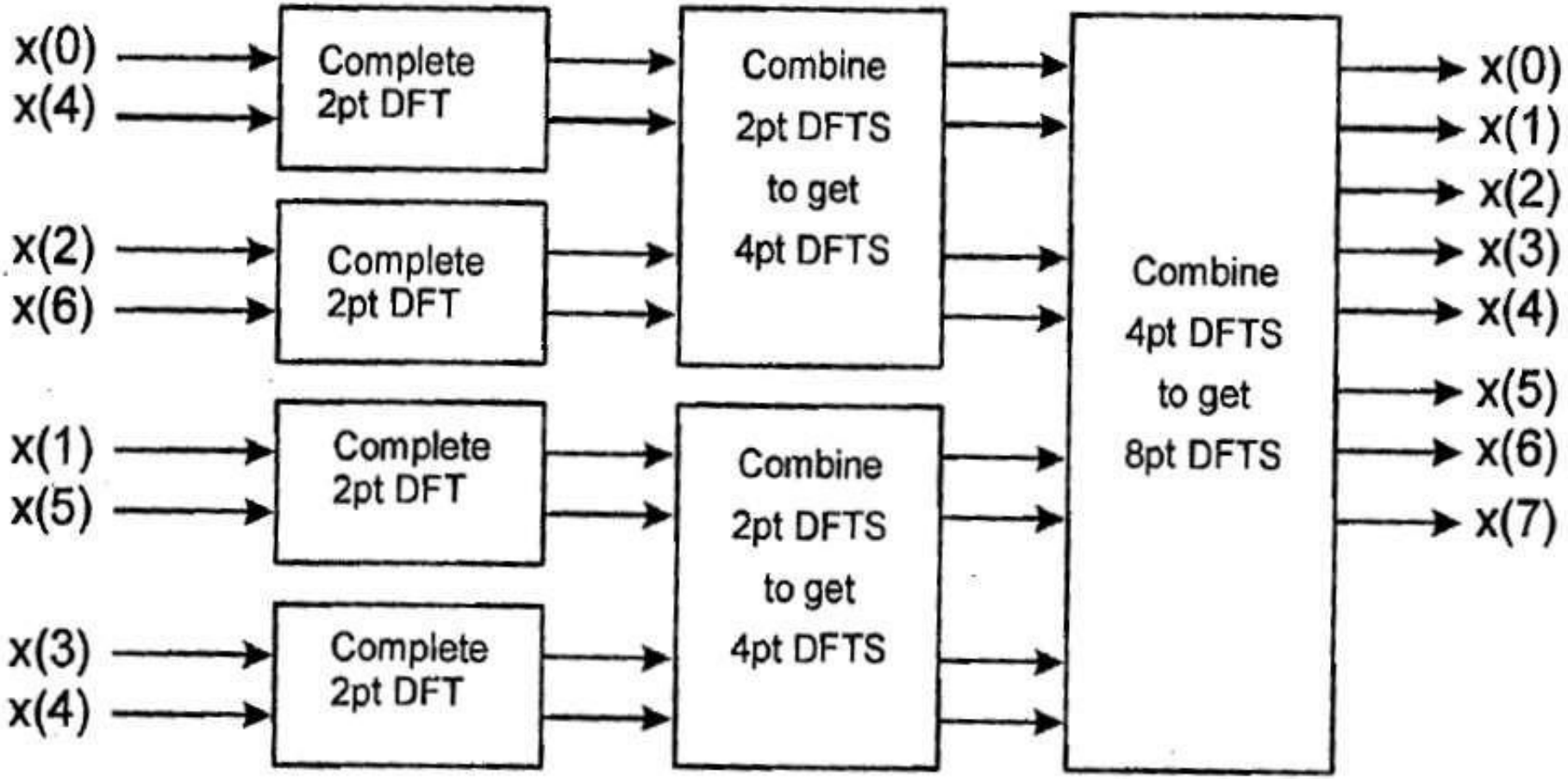
- x(0)-000
- x(1)-001
- x(2)-010
- x(3)-011
- x(4)-100
- x(5)-101
- x(6)-110
- x(7)-111

Output should be in normal order

- **Bit reverse orde**

- x(0)-000
- x(4)-100
- x(2)-010
- x(6)-110
- x(1)-001
- x(5)-101
- x(3)-011
- x(7)-111

Radix-2 DIT FFT algorithm Cont..



First stage of computation

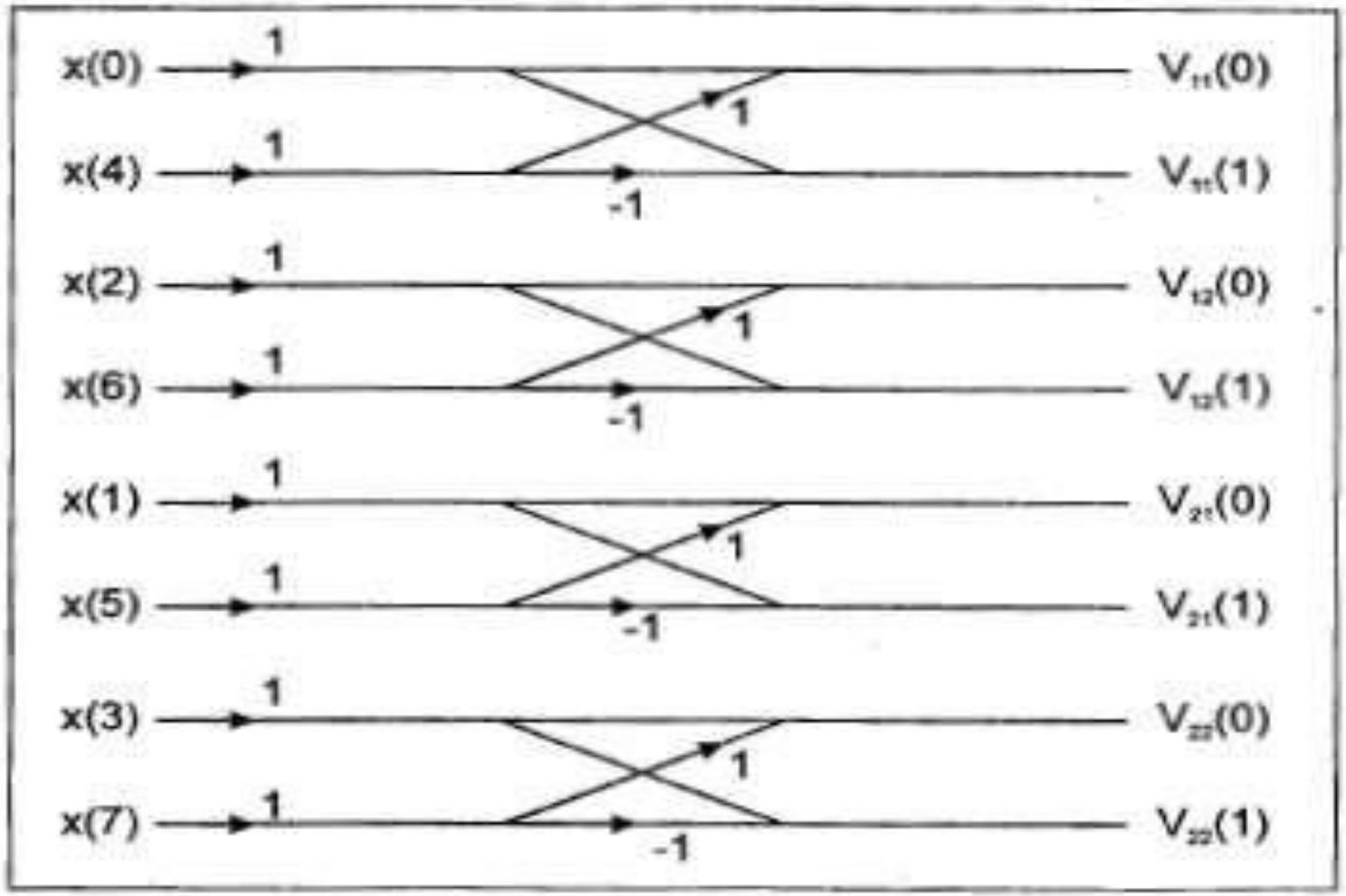
$$V_{11}(0) = x(0) + x(4) \qquad V_{11}(1) = x(0) - x(4)$$

$$V_{12}(0) = x(2) + x(6) \qquad V_{12}(1) = x(2) - x(6)$$

$$V_{21}(0) = x(1) + x(5) \qquad V_{21}(1) = x(1) - x(5)$$

$$V_{22}(0) = x(3) + x(7) \qquad V_{22}(1) = x(3) - x(7)$$

First stage of computation



Second stage of computation

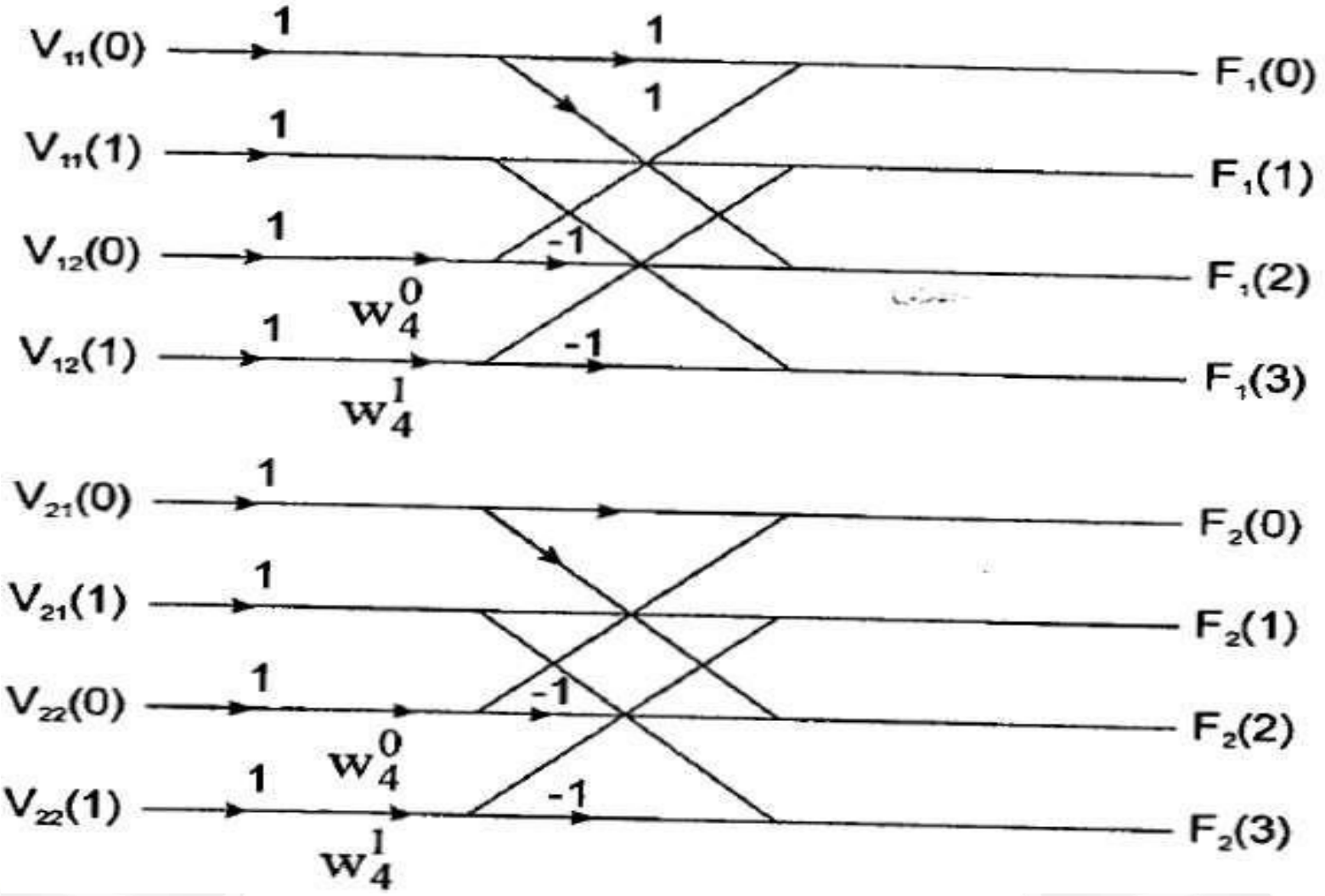
$$F_1(0) = V_{11}(0) + W_4^0 V_{12}(0) \qquad F_2(0) = V_{21}(0) + W_4^0 V_{22}(0)$$

$$F_1(1) = V_{11}(1) + W_4^1 V_{12}(1) \qquad F_2(1) = V_{21}(1) + W_4^1 V_{22}(1)$$

$$F_1(2) = V_{11}(0) - W_4^0 V_{12}(0) \qquad F_2(2) = V_{21}(0) - W_4^0 V_{22}(0)$$

$$F_1(3) = V_{11}(1) - W_4^1 V_{12}(1) \qquad F_2(3) = V_{21}(1) - W_4^1 V_{22}(1)$$

Second stage of computation



Third stage of computation

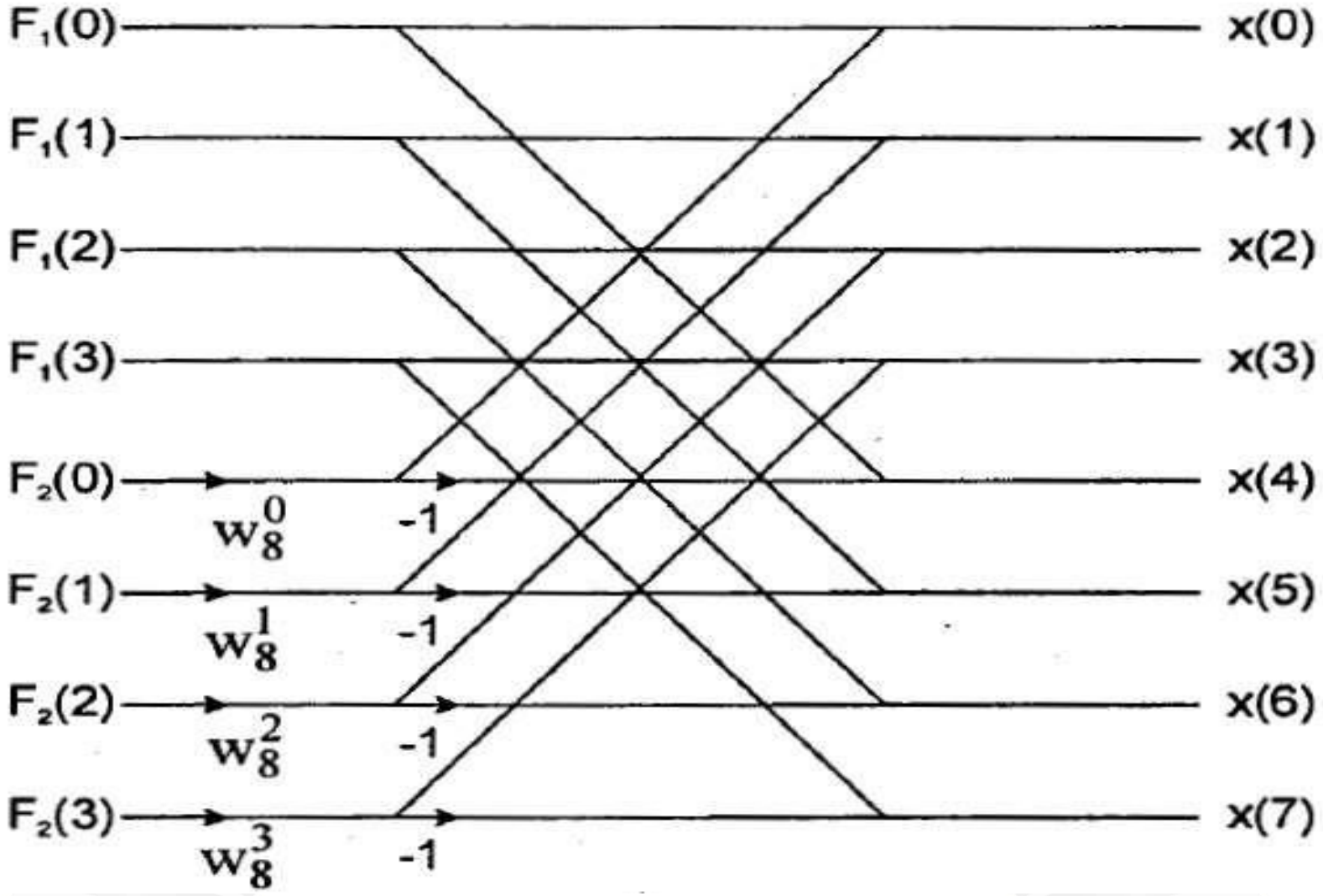
$$X(0) = F_1(0) + W_8^0 F_2(0) \qquad X(4) = F_1(0) - W_8^0 F_2(0)$$

$$X(1) = F_1(1) + W_8^1 F_2(1) \qquad X(5) = F_1(1) - W_8^1 F_2(1)$$

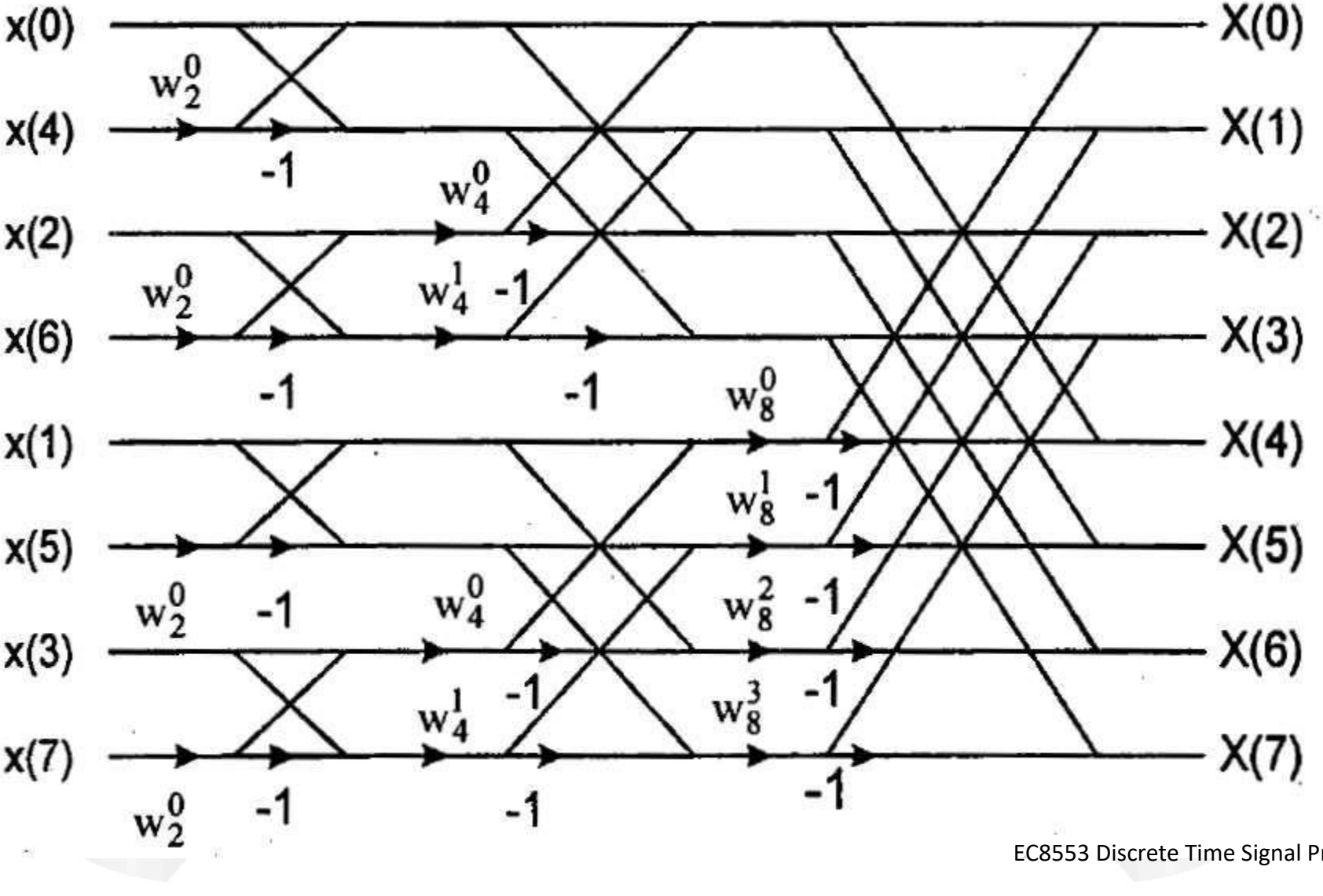
$$X(2) = F_1(2) + W_8^2 F_2(2) \qquad X(6) = F_1(2) - W_8^2 F_2(2)$$

$$X(3) = F_1(3) + W_8^3 F_2(3) \qquad X(7) = F_1(3) - W_8^3 F_2(3)$$

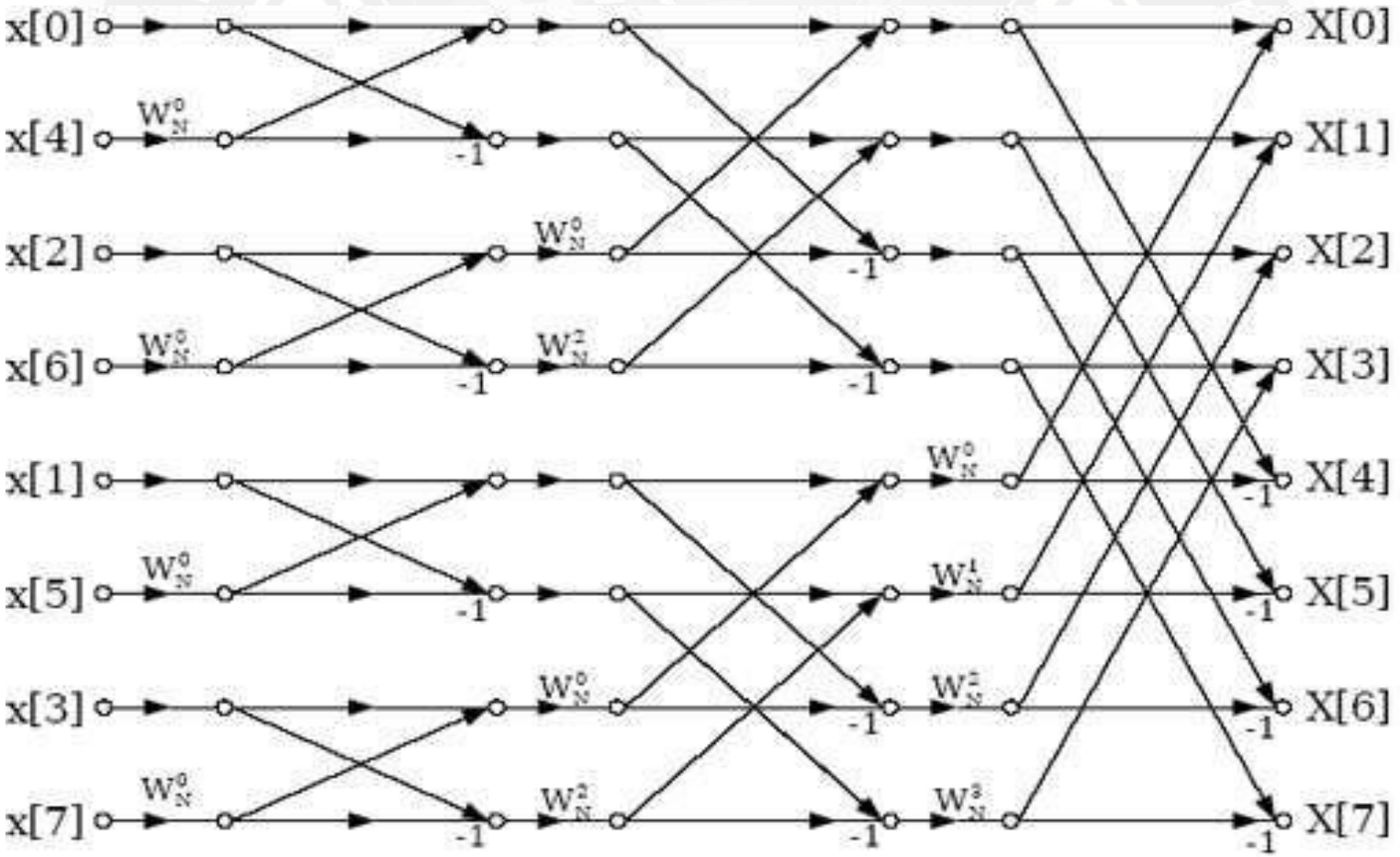
Third stage of computation



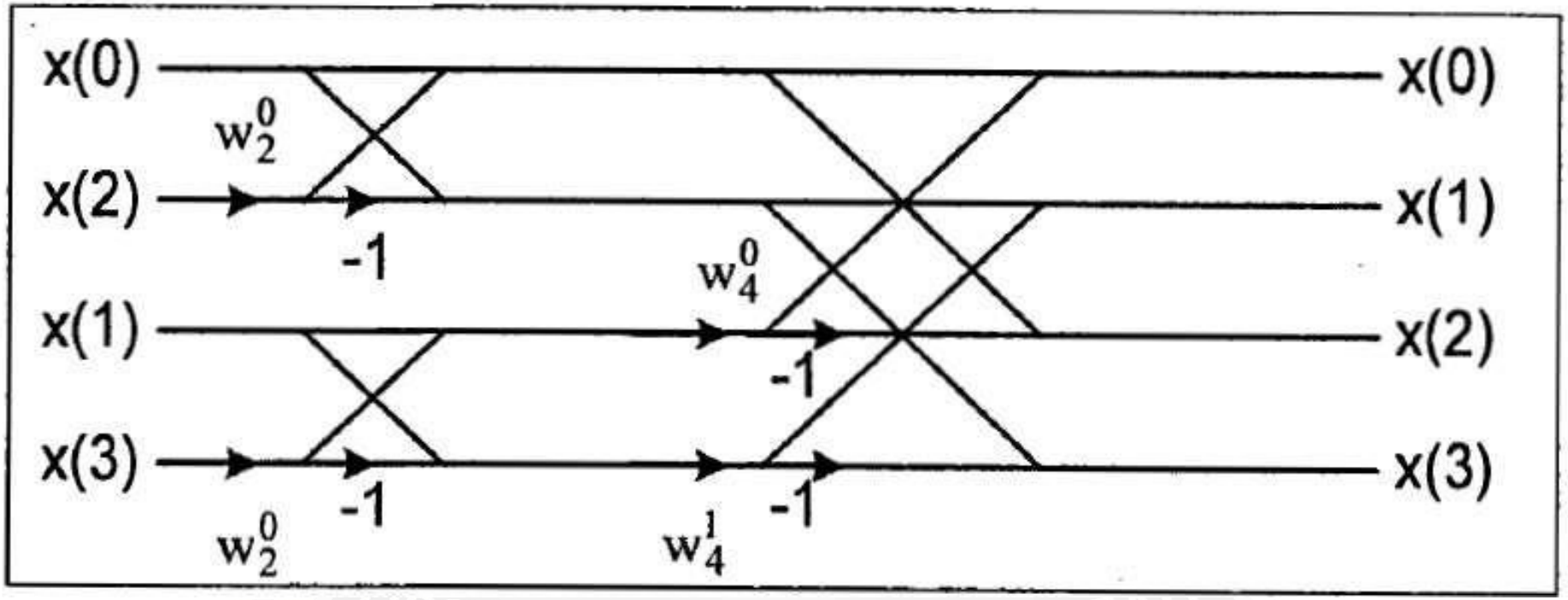
Butterfly diagram for 8-point radix-2 DIT FFT algorithm



Butterfly diagram for 8-point radix-2 DIT FFT algorithm



Butterfly diagram for 4-point radix-2 DIT FFT algorithm



Calculation of Twiddle factor values

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

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

$$W_2^0 = W_4^0 = W_8^0 = e^{\frac{-j2\pi(0)}{N}} = e^0 = 1$$

$$W_4^1 = e^{\frac{-j2\pi(1)}{4}} = e^{\frac{j\pi}{2}} = \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = -j$$

$$W_8^2 = e^{\frac{-j2\pi(2)}{8}} = e^{\frac{j\pi}{2}} = \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = -j$$

Calculation of Twiddle factor values

$$W_8^1 = e^{\frac{-j2\pi(1)}{8}} = e^{-\frac{j\pi}{4}} = \cos\frac{\pi}{4} - j\sin\frac{\pi}{4} = 0.707 - j0.707$$

$$W_8^3 = e^{\frac{-j2\pi(3)}{8}} = e^{\frac{-j3\pi}{4}} = \cos\frac{3\pi}{4} - j\sin\frac{3\pi}{4} = -0.707 - j0.707$$

Radix-2 DIF FFT algorithm

- In DIF algorithm, the frequency domain sequence X(k) is decimated.
- In this algorithm, N-point time domain sequence is converted to two number of N/2 point sequence.
- Then each N/2 point sequence is converted into two number of N/4 point sequence.
- This process is continued until N/2 number of 2 point sequences is obtained

Radix-2 DIF FFT algorithm Cont..

- **Normal order**

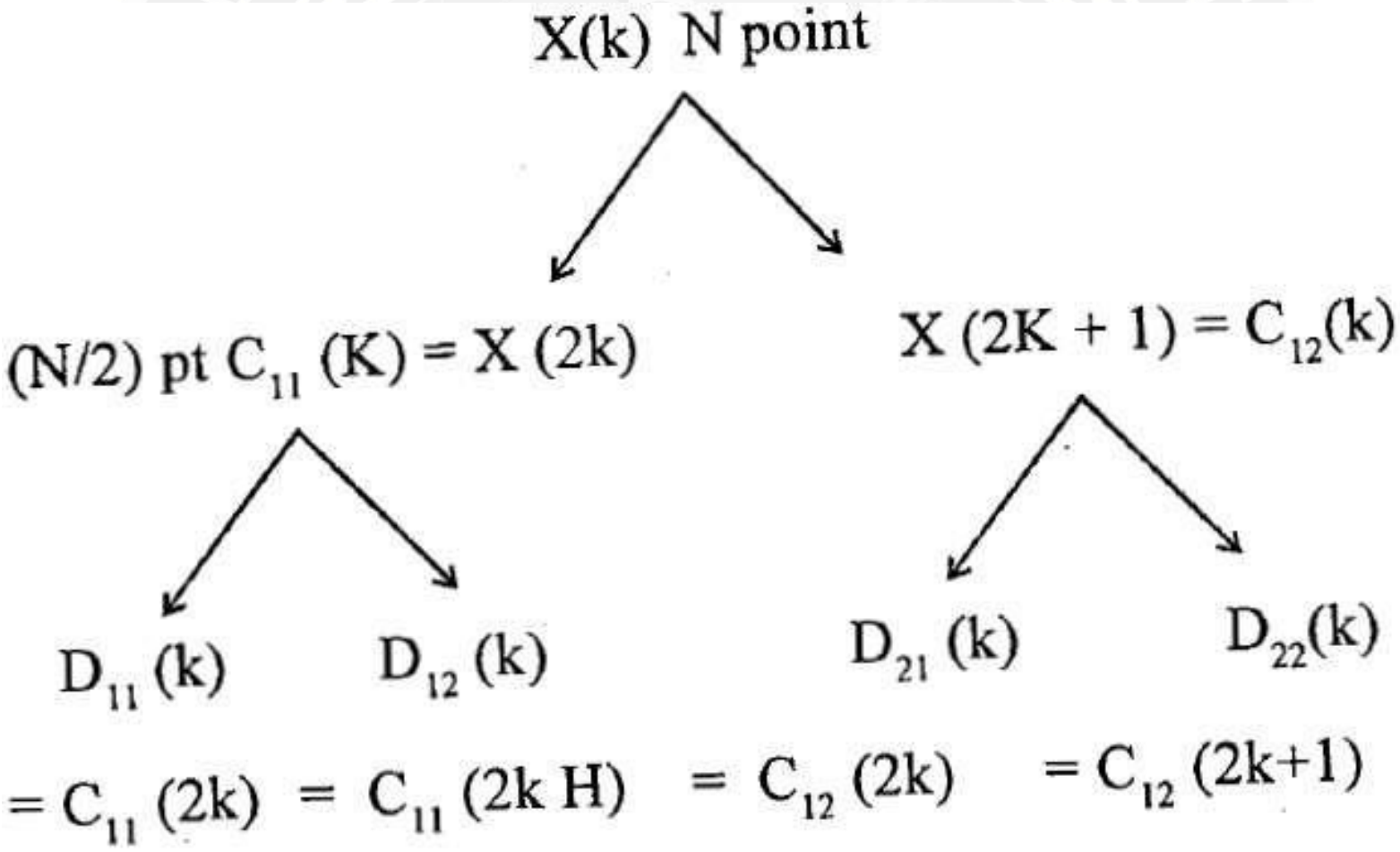
- x(0)-000
- x(1)-001
- x(2)-010
- x(3)-011
- x(4)-100
- x(5)-101
- x(6)-110
- x(7)-111

Input should be in normal order

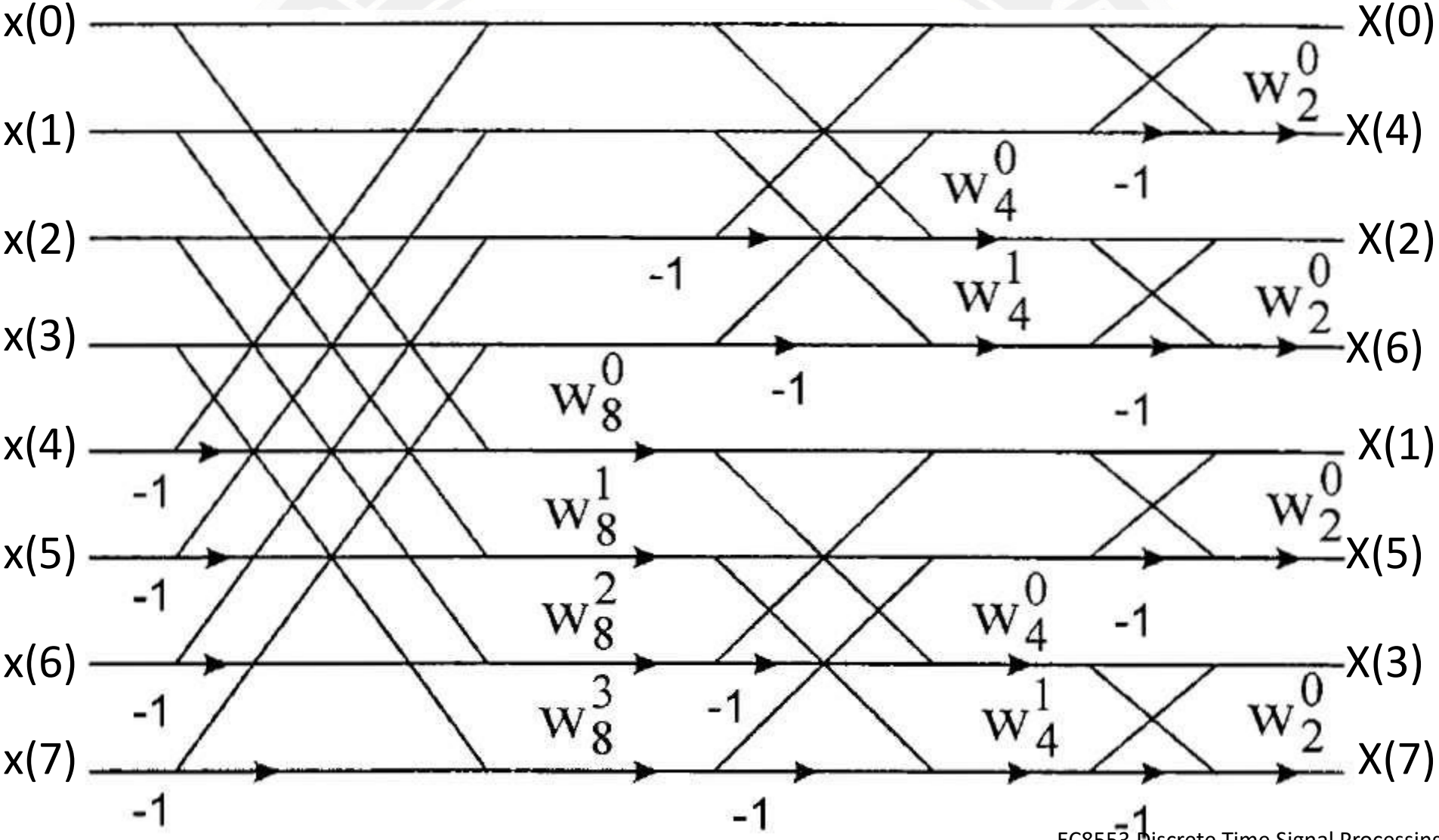
- **Bit reverse orde**

- x(0)-000
- x(4)-100
- x(2)-010
- x(6)-110
- x(1)-001
- x(5)-101
- x(3)-011
- x(7)-111

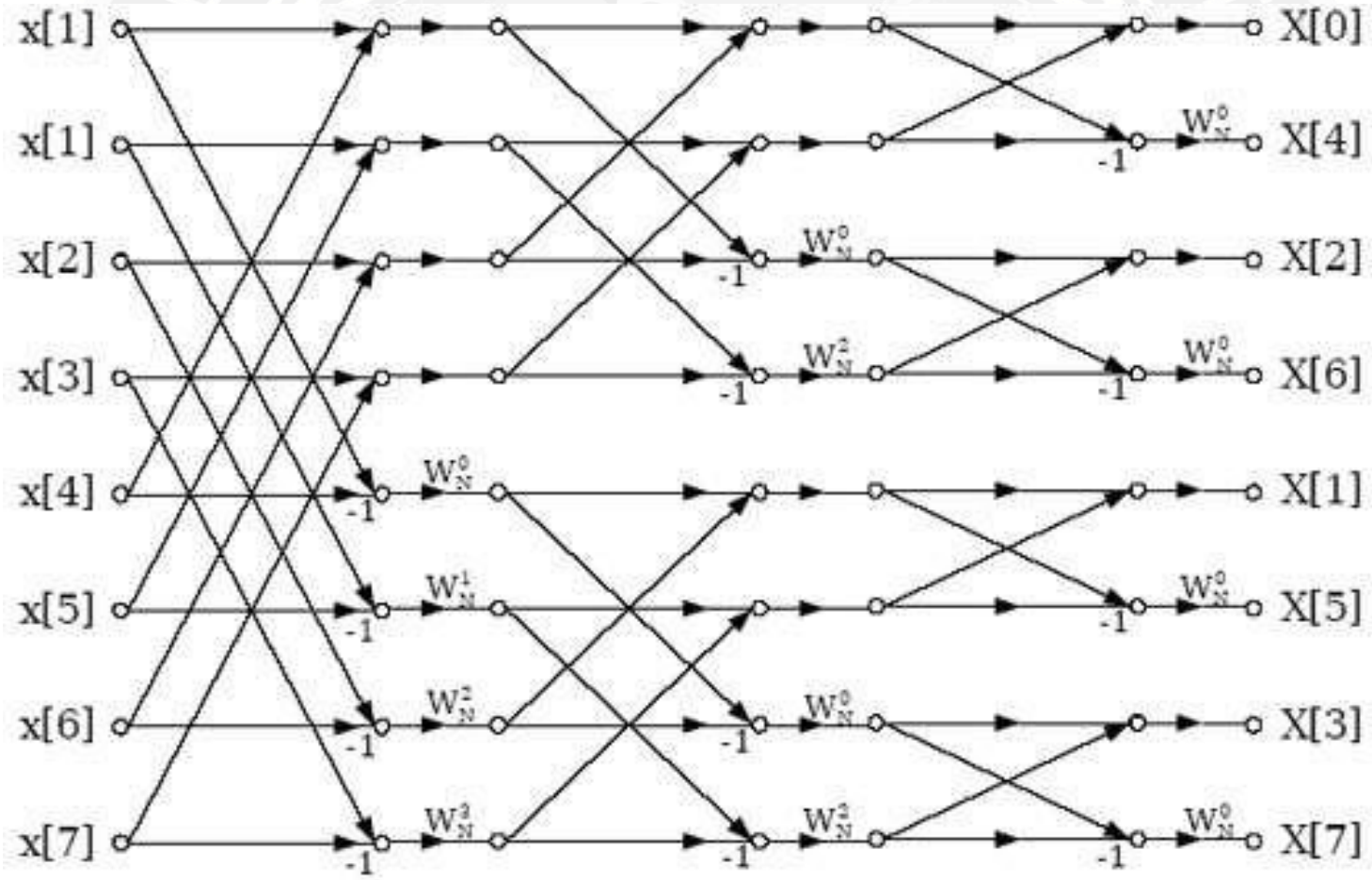
Radix-2 DIF FFT algorithm



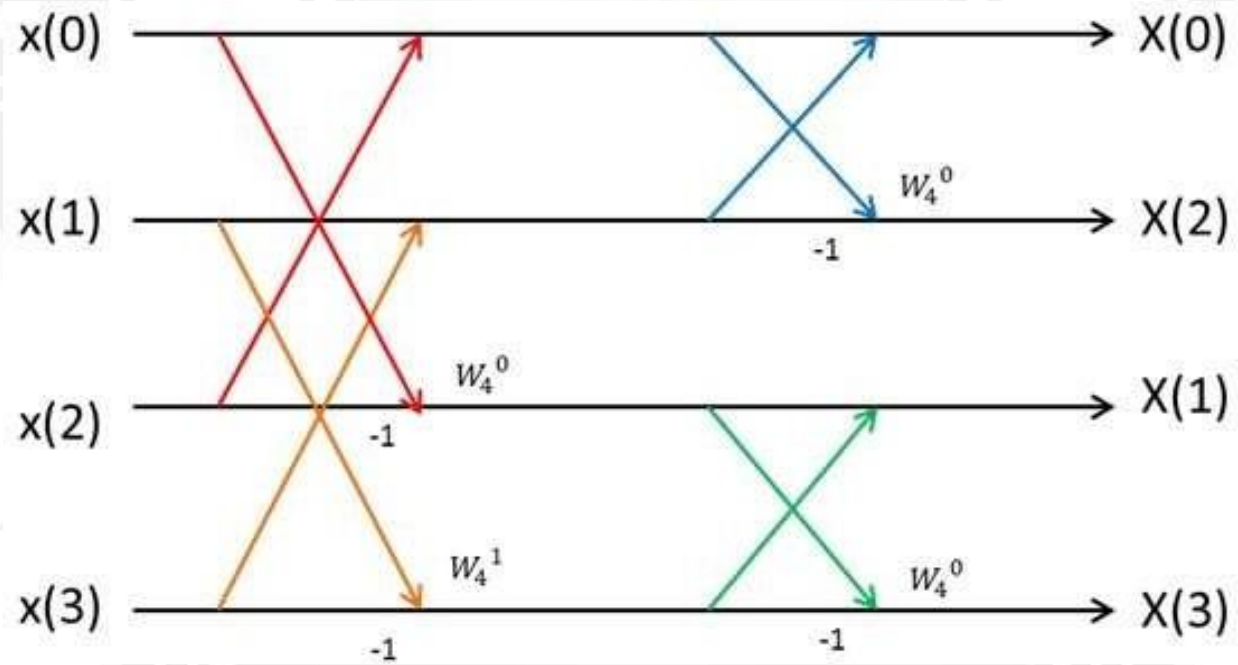
Butterfly diagram for 8-point radix-2 DIF FFT algorithm



Radix-2 DIF FFT algorithm



Butterfly diagram for 4-point radix-2 DIF FFT algorithm



Differences between DIT & DIF FFT algorithm

DIT FFT

- Time domain sequence is decimated
- Input should be in bit reversed order & output will be in normal order
- Complex multiplication takes place before the add-subtract operation

DIF FFT

- Frequency domain sequence is decimated
- Input should be in normal order & output will be in bit reversed order
- Complex multiplication takes place after the add-subtract operation

Similarities in DIT & DIF algorithm

- For both algorithms the value of N should be such that N= 2^M & there will be M stages of butterfly computation with N/2 butterfly per stage
- Both algorithms involve same number of operations. The total no. of complex additions are $N \log_2 N$ & the total no. of complex multiplications are $\frac{N}{2} \log_2 N$
- Both algorithm require bit reversal at some place during computation

Relationship between exponential forms and twiddle factors (W) for Periodicity = N

Sr. No.	Exponential form	Symbolic form
01	$e^{-j2\pi n/N} = e^{-j2\pi(n+N)/N}$	$W_{N_n} = W_{N_{n+N}}$
02	$e^{-j2\pi(n+N/2)/N} = -e^{-j2\pi n/N}$	$W_{N_{n+N/2}} = -W_{N_n}$
03	$e^{-j2\pi k} = e^{-j2\pi Nk/N} = 1$	$W_{N_{N+K}} = 1$
04	$e^{-j2(2\pi/N)} = e^{-j2\pi/(N/2)}$	$W_{N_2} = W_{N/2}$

Matrix Relations

- The DFT samples defined by

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

can be expressed in NxN matrix as

$$[\mathrm{X(k)}] = \left[\sum_{\mathrm{n}=0}^{N-1} W_N^{nk}\right][\mathrm{x(n)}]$$

where

$$\mathbf{X} = [X[0] \quad X[1] \quad \cdots\cdots \quad X[N-1]]^T$$

$$\mathbf{x} = [x[0] \quad x[1] \quad \cdots\cdots \quad x[N-1]]^T$$

Matrix Relations

$\sum_{k=0}^{N-1} W_N^{nk}$ can be expanded as NXN **DFT matrix**

$$\sum_{k=0}^{N-1} W_N^{nk} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix}$$

Matrix Relations

- Likewise, the IDFT is

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

can be expressed in NxN matrix form as

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

Matrix Relations

$\sum_{k=0}^{N-1} W_N^{-nk}$ can also be expanded as NXN **DFT matrix**

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

Observe:

$$\sum_{k=0}^{N-1} W_N^{-nk} = \begin{bmatrix} 1 \end{bmatrix} = \left[1 \Big/ N \sum_{n=0}^{N-1} W_N^{nk} \right]^*$$

The inversion can be had by Hermitian conjugating j by –j and dividing by N.

Zero-Padding

- Consider an L-point input sequence x(n) and a P-point impulse response h(n)
- The linear convolution of these two sequence y(n) has finite duration with length *(L+P-1)*
- For the circular convolution and linear convolution to be identical, the circular convolution must have a length of at least *(L+P-1)* points.

Zero-Padding

- The circular convolution can be achieved by multiplying the DFTs of x(n) and h(n).
- Since the length of the linear convolution is *(L+P-1)* points, the DFTs that we compute must also be of at least that length, i.e., both x(n) and h(n) must augmented with sequence values of zero.
- The process is called *Zero-Padding*

Zero padding

- While finding N-point DFT of x(n), if the length of x(n) is M then, we should add N-M number of zeros in x(n).
- This is called zero padding
- Uses:
  - Frequency spectrum is good
  - DFT is used in linear filtering because of zero padding