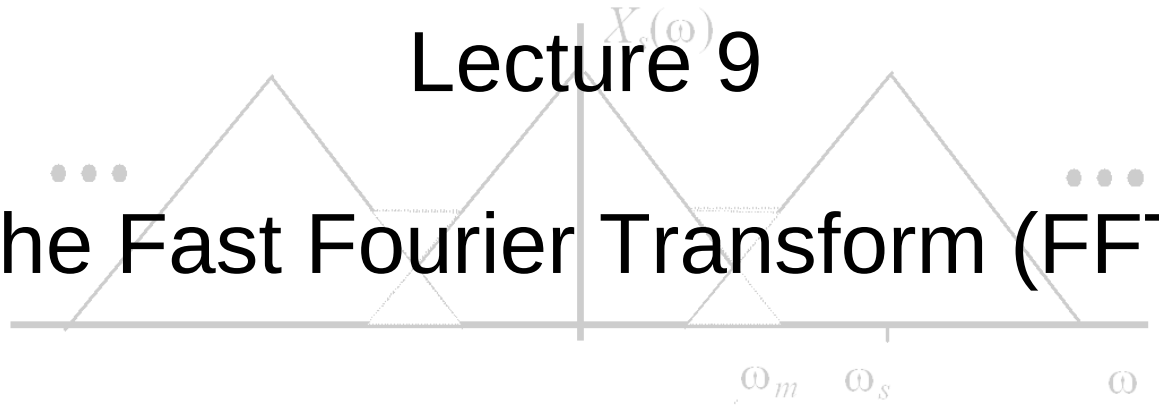
Lecture 9

The Fast Fourier Transform (FFT)



Reduced-complexity fast implementation of the DFT

Dr. Dimitris Agrafiotis, Dept. Of Electrical & Electronic Engineering

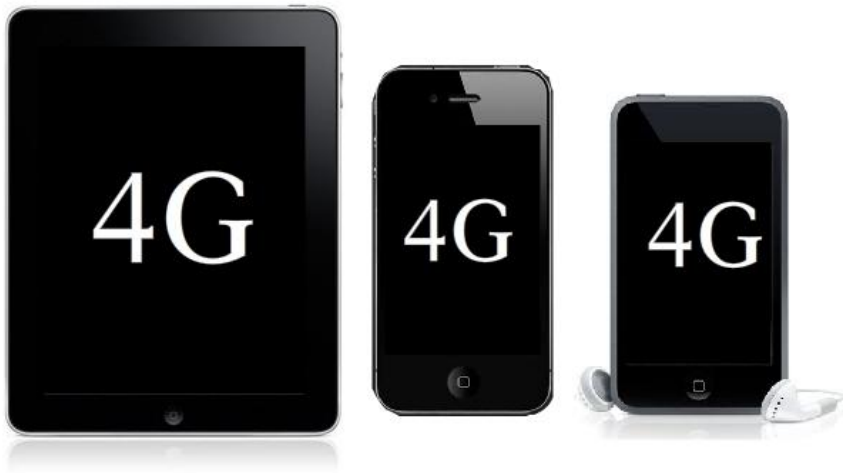
The Fast Fourier Transform

Discrete Fourier Transform - DFT

Why a fast Implementation ?

Wi-Fi

- Uses OFDM for transmitting /receiving data
- OFDM employs the DFT / IDFT
- A 64-DFT is computed once every 4 µsec
- A 64-DFT computed 250,000 times/second



- Next generation mobile phone standard
- Uses OFDM to support high data rates
- A 1024-DFT is computed once every ~67 µsec
- A 1024-DFT computed 15,000 times/second

The Fast Fourier Transform

Direct Implementation of the DFT

Matrix Representation of the DFT

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

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j 2\pi kn/N}$$
$$n = 0,1,\ldots, N-1$$

DFT : Transformation of N samples in time to N samples in frequency

$\Rightarrow$ DFT is a coordinate transformation in an N-dimensional space

$\Rightarrow$ DFT can be expressed in Matrix notation

$$\mathbf{X}_N = \mathbf{W}_N \mathbf{x}_N$$

$$\mathbf{x}_N = \mathbf{W}_N^{-1} \mathbf{X}_N$$

$\mathbf{X}$	$\mathbf{W}$	$\mathbf{x}$
(Nx1) element vector of frequency coefficients	is the NxN DFT matrix	(Nx1) element vector of time domain coefficients

The Fast Fourier Transform

Direct Implementation of the DFT

Matrix Representation of the DFT – The Twiddle Factor

DFT

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad k = 0,1,\ldots, N-1$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N} \qquad n = 0,1,\ldots, N-1$$

Matrix notation

known as the twiddle factor

$$w_N = e^{-j2\pi/N}$$

$$e^{-j2\pi kn/N} = e^{-j(2\pi/N)^{kn}}$$

$$w_N^{kn} = e^{-j2\pi kn/N}$$

$$\mathbf{X}_N = \mathbf{W}_N \mathbf{x}_N$$

$$\mathbf{x}_N = \mathbf{W}_N^{-1} \mathbf{X}_N$$

DFT

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

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

The Fast Fourier Transform

Direct Implementation of the DFT

Matrix Representation of the DFT – Forward Transform

$$X[k] \qquad W(k,n) = w_N^{kn} = e^{-j(2\pi/N)^{kn}} \qquad x[n]$$

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} w_N^{0x0} & w_N^{0x1} & w_N^{0x2} & \cdots & w_N^{0x(N-1)} \\ w_N^{1x0} & w_N^{1x1} & w_N^{1x2} & \cdots & w_N^{1x(N-1)} \\ w_N^{2x0} & w_N^{2x1} & w_N^{2x2} & \cdots & w^{2x(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_N^{(N-1)x0} & w_N^{(N-1)x1} & w_N^{(N-1)x2} & \cdots & w^{(N-1)x(N-1)} \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

$$\mathbf{X}_N = \mathbf{W}_N \mathbf{x}_N$$

DFT

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

$$k = 0,1,\ldots, N-1$$

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

$$n = 0,1,\ldots, N-1$$

The Fast Fourier Transform

Direct Implementation of the DFT

Matrix Representation of the DFT - Inverse Transform

$x[n]$ | $W^{-1}(k,n) = w_N^{kn\,*}$ | $X[k]$

$$\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix} = \boxed{\mathbf{W}^{-1} = \frac{1}{\mathbf{N}}\mathbf{W}^{*},\ w_N^{kn\,*} = e^{j2\pi kn/N}} \times \begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix}$$

$$w_N^{kn} \times w_N^{kn\,*} = e^{-j2\pi kn/N} \times e^{j2\pi kn/N} = 1$$

$$\mathbf{x}_N = \mathbf{W}_N^{-1}\mathbf{X}_N$$

DFT

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

$$k = 0,1,\ldots, N-1$$

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

$$n = 0,1,\ldots, N-1$$

The Fast Fourier Transform

Direct Implementation of the DFT

Some observations for faster calculation of the DFT

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \left(e^{-j\frac{2\pi}{N}}\right) & \left(e^{-j\frac{2\pi}{N}}\right)^2 & \cdots & \left(e^{-j\frac{2\pi}{N}}\right)^{N-1} \\ 1 & \left(e^{-j\frac{2\pi}{N}}\right)^2 & \left(e^{-j\frac{2\pi}{N}}\right)^4 & \cdots & \left(e^{-j\frac{2\pi}{N}}\right)^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \left(e^{-j\frac{2\pi}{N}}\right)^{N-1} & \left(e^{-j\frac{2\pi}{N}}\right)^{2(N-1)} & \cdots & \left(e^{-j\frac{2\pi}{N}}\right)^{(N-1)(N-1)} \end{bmatrix}$$

Complex Conjugate Symmetry

$$w_N^{k(N-n)} = e^{(-j\frac{2\pi}{N})k(N-n)} =$$

$$= e^{-j\frac{2\pi}{N}kN} e^{j\frac{2\pi}{N}kn} = w_N^{kn^*} \qquad (1)$$

$$\left.\begin{matrix} k(N-n) \\ k=2,n=1 \end{matrix}\right\} = 2(N-1) = \left\{\begin{matrix} kn \\ k=2,n=N-1 \end{matrix}\right.$$

$$\left.\begin{matrix} (N-k)n \\ k=1,n=2 \end{matrix}\right\} = (N-1)2 = \left\{\begin{matrix} kn \\ k=N-1,n=2 \end{matrix}\right.$$

$$= w_N^{2(N-1)} \overset{(1)}{=} w_N^{2^*}$$

Pre-calculate values for some matrix coefficients and reuse these values for other coefficients

The Fast Fourier Transform

Direct Implementation of the DFT

Some observations for faster calculation of the DFT

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \left(e^{-j\frac{2\pi}{8}}\right) & \left(e^{-j\frac{2\pi}{8}}\right)^2 & \cdots & \left(e^{-j\frac{2\pi}{8}}\right)^7 \\ 1 & \left(e^{-j\frac{2\pi}{8}}\right)^2 & \left(e^{-j\frac{2\pi}{8}}\right)^4 & \cdots & \left(e^{-j\frac{2\pi}{8}}\right)^{2\times 7} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \left(e^{-j\frac{2\pi}{8}}\right)^7 & \left(e^{-j\frac{2\pi}{8}}\right)^{2\times 7} & \cdots & \left(e^{-j\frac{2\pi}{8}}\right)^{7\times 7} \end{bmatrix}$$

Example for N=8

- Pre-calculate values for matrix coefficients
- Observe any relationships and exploit them
- Reuse these values for other coefficients
- $w_N^{k(N-n)} = w_N^{kn^*}$

$A^0 = w_N^0 = 1$

$A^1 = w_8^1 = e^{-j\frac{2\pi}{8}} = e^{-j\frac{\pi}{4}} = \frac{(1-j)}{\sqrt{2}}$

$A^2 = w_8^2 = e^{(-j\frac{2\pi}{8})^2} = e^{-j\frac{4\pi}{8}} = e^{-j\frac{\pi}{2}} = -j$

$A^3 = A^2 A = \frac{(1-j)}{\sqrt{2}} \times -j = \frac{(1-j)\times -j}{\sqrt{2}} = -A^*$

$A^4 = A^2 \times A^2 = -1$

$A^5 = A^{1\times(N-3)} = A^{3^*} = -A$

$A^6 = A^{1\times(N-2)} = A^{2^*} = j$

$A^7 = A^{1\times(N-1)} = A^{1^*} = A^*$

Complex multiplication: $(a+bj)(c+dj) = (ac-bd)+(bc+ad)j$

The Fast Fourier Transform

Direct Implementation of the DFT

Example for N=8

$$w = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & A & -j & -A^* & -1 & -A & j & A^* \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -A^* & j & A & -1 & A^* & -j & -A \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -A & -j & A^* & -1 & A & j & -A^* \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & A^* & j & -A & -1 & -A^* & -j & A \end{bmatrix}$$

$$W^{-1} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & A^* & j & -A & -1 & -A^* & -j & A \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & -A & -j & A^* & -1 & A & j & -A^* \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -A^* & j & A & -1 & A^* & -j & -A \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & A & -j & -A^* & -1 & -A & j & A^* \end{bmatrix}$$

$$\mathbf{X}_8 = \mathbf{W}_8 \mathbf{x}_8$$

$$\mathbf{x}_8 = \mathbf{W}_8^{-1} \mathbf{X}_8, \quad \mathbf{W}^{-1} = \frac{\mathbf{1}}{\mathbf{N}} \mathbf{W}^*$$

The Fast Fourier Transform

Direct Implementation of the DFT

Complexity

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & A & -j & -A^* & -1 & -A & j & A^* \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -A^* & j & A & -1 & A^* & -j & -A \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -A & -j & A^* & -1 & A & j & -A^* \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & A^* & j & -A & -1 & -A^* & -j & A \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

N=8, 64 multiplications & 56 additions

N	DFT Multiplications
8	64
64	4096
256	65536
1024	1048576
4096	16777216

$X_N = W_N \times x_n$

[NxN] x [Nx1] matrix multiplication

N^2 complex multiplications

N(N-1) complex additions

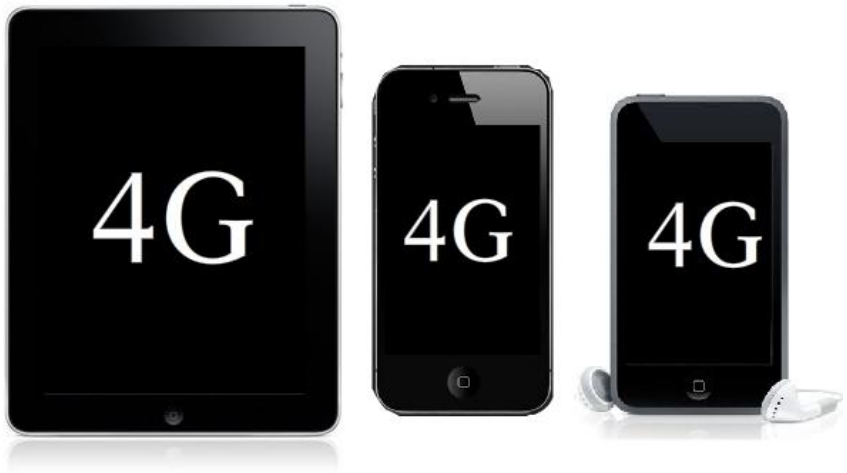
The computational complexity of a direct DFT implementation is of the order of N^2 : $O(N^2)$

Discrete Fourier Transform - DFT

Why a fast Implementation ?

Wi-Fi

- **A 64-DFT computed 250,000 times/second**
- 4096 x 250,000 multiplications / second
- 1,024,000,000 multiplications / second
- 1 billion multiplications per second !



- **A 1024-DFT computed 15,000 times/second**
- 1048576 x 15000 multiplications / second
- 15,728,640,000 multiplications / second
- 15 billion multiplications per second !

The Fast Fourier Transform - FFT

Algorithms for reduced-complexity fast implementation of the DFT

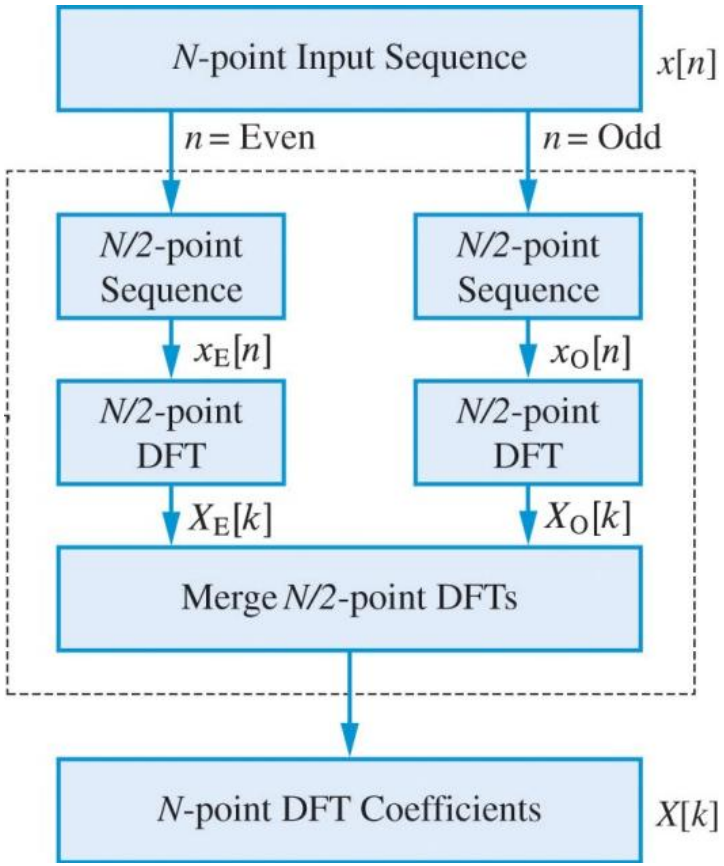
1. The Cooley & Tuckey algorithm
   - Decimation in Time (DIT) FFT
   - Implementation aspects
   - Decimation in Frequency (DIF) FFT

2. Other Fast Fourier Transform Algorithms
   - Radix 4
   - Split Radix
   - The Goertzel Algorithm

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Basic Principle: Decimate and Merge



N	DFT Multiplications
2	4
4	16
8	64
16	256
32	1024
64	4096
128	16384
256	65536

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Basic Principle: Decimate and Merge

1-point DFT : $X_1 = W_1 x_1$, $W_1 = [w_1^0] = 1$ $\Rightarrow X_1 = x_1$

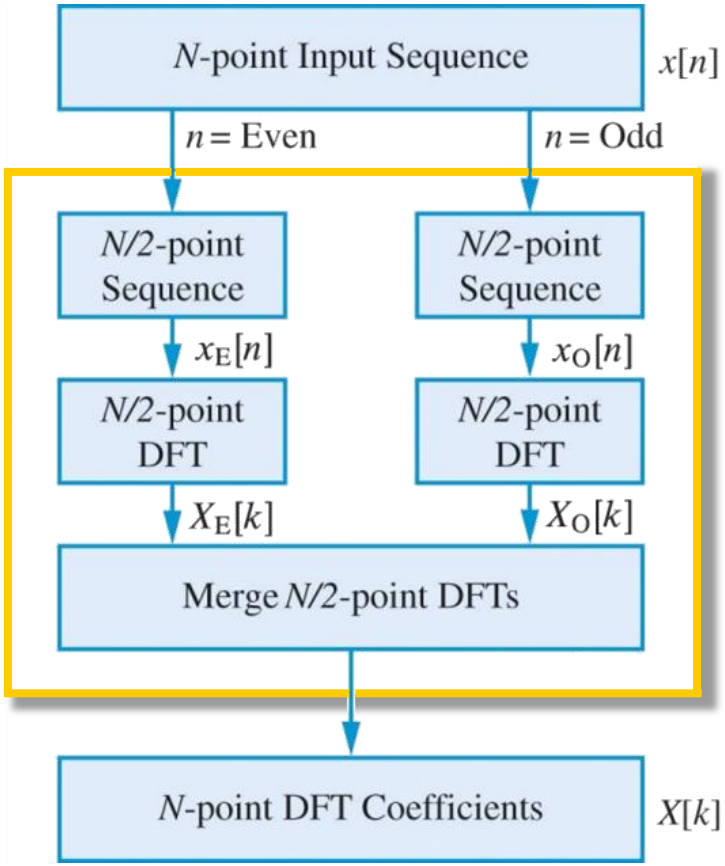
		N	DFT Multiplications
Break each into	2 1-point DFTs **Merge**	2	4
Break each into	2 2-point DFTs **Merge**	4	16
Break each into	2 4-point DFTs **Merge**	8	64
Break each into	2 8-point DFTs **Merge**	16	256
Break each into	2 16-point DFTs **Merge**	32	1024
Break each into	2 32-point DFTs **Merge**	64	4096
Break each into	2 64-point DFTs **Merge**	128	16384
Break this into	2 128-point DFTs **Merge**	256	65536

$\log_2(N)$ Merging stages

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Basic Principle: Decimate and Merge



N Point DIT FFT :

- Decimate sequence (odd and even subsequences)
- Calculate two N/2 point DFTs
- Merge the two N/2 point DFTS

1.Why is this equivalent with an N point DFT

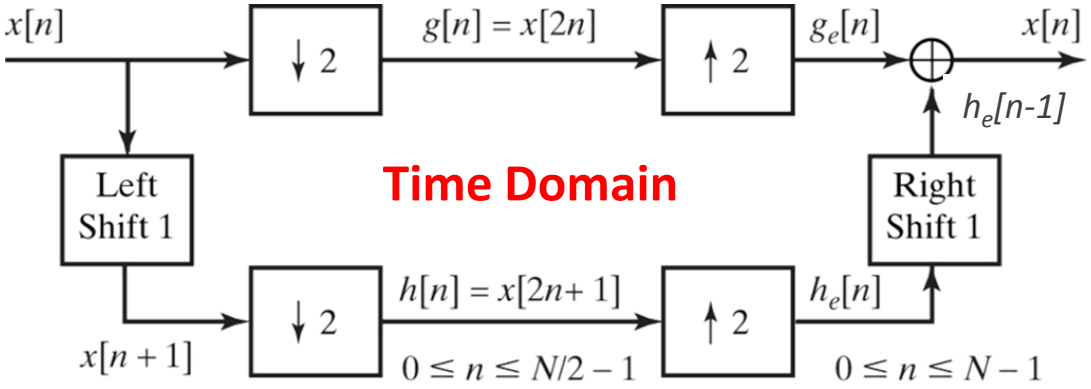
2.What is the complexity of the merging stage

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Frequency Domain Equivalents of Decimation and Merging

Decimation



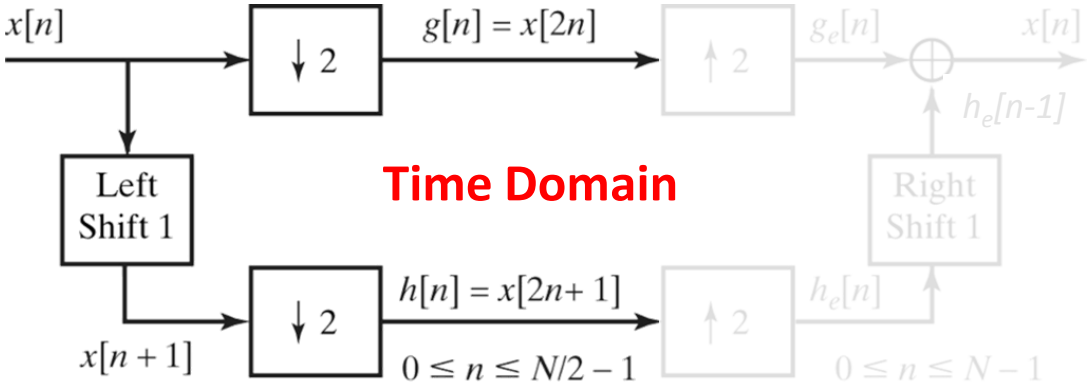
Merging

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Frequency Domain Equivalents of Decimation and Merging

Decimation



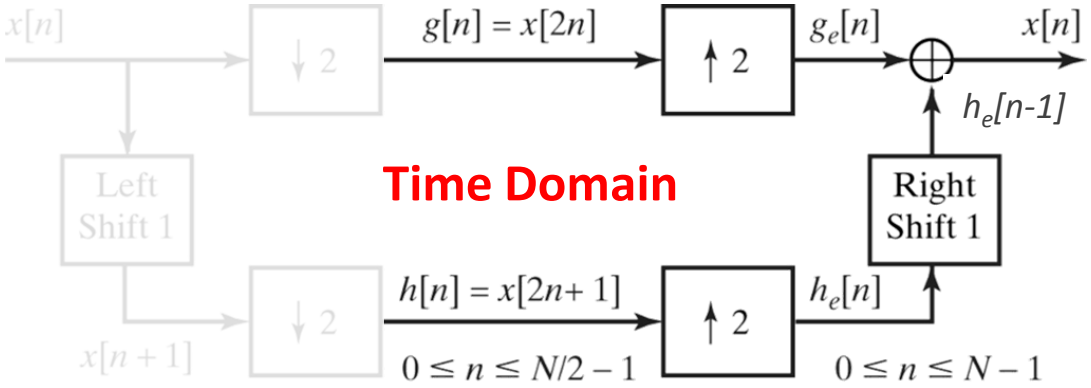
Frequency Domain Equivalent of Decimation (DTFT)

x[n]	$\Leftrightarrow X(\Omega)$
Left Shift 1 x[n+1]	$\Leftrightarrow e^{j\Omega}X(\Omega)$
↓ 2	$\Leftrightarrow \Omega = \Omega/2$
g[n]=x[2n] $0 \le n \le N/2-1$	$\Leftrightarrow G(\Omega) = \frac{1}{2}\left(X(\Omega/2) + X((\Omega-2\pi)/2)\right)$
h[n]=x[2n+1] $0 \le n \le N/2-1$	$\Leftrightarrow H(\Omega) = \frac{1}{2}\left(e^{j\Omega/2}X(\Omega/2) + e^{j(\Omega-2\pi)/2}X((\Omega-2\pi)/2)\right)$

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Frequency Domain Equivalents of Decimation and Merging



Frequency Domain Equivalent of Merging (DTFT)

g[n]=x[2n] $0 \le n \le N/2-1$	$\Leftrightarrow G(\Omega)=\frac{1}{2}\left(X(\Omega/2)+X((\Omega-2\pi)/2)\right)$
h[n]=x[2n+1] $0 \le n \le N/2-1$	$\Leftrightarrow H(\Omega)=\frac{1}{2}\left(e^{j\Omega/2}X(\Omega/2)+e^{j(\Omega-2\pi)/2}X((\Omega-2\pi)/2)\right)$
↑ 2 $G_e(\Omega)$ $H_e(\Omega)$	$\Leftrightarrow \Omega=2\Omega,\quad G_e(\Omega)=G(2\Omega)=(X(\Omega)+X(\Omega-\pi))/2$ $H_e(\Omega)=H(2\Omega)=(e^{j\Omega}X(\Omega)+e^{j(\Omega-\pi)}X(\Omega-\pi))/2$ (1)
Right Shift 1 $h_e[n-1]$	$\Leftrightarrow e^{-j\Omega}H_e(\Omega)=(X(\Omega)+e^{-j\pi}X(\Omega-\pi))/2=(X(\Omega)-X(\Omega-\pi))/2$ (2)
x[n]	$\Leftrightarrow G_e(\Omega)+e^{-j\Omega}H_e(\Omega)=_{(1,2)} X(\Omega)$

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Complexity of the merging stage (Derivation of FFT)

$$X(k)=\sum_{n=0}^{N-1}x[n]e^{-j2\pi kn/N}$$

$$k=0,1,\ldots,N-1$$

N-point DFT of N-point sequence

$$X[k]=\sum_{n=even}x[n]e^{-jnk\frac{2\pi}{N}}+\sum_{n=odd}x[n]e^{-jnk\frac{2\pi}{N}}=\sum_{r=0}^{N/2-1}x[2r]e^{-j2rk\frac{2\pi}{N}}+\sum_{r=0}^{N/2-1}x[2r+1]e^{-j(2r+1)k\frac{2\pi}{N}}$$

$$=\sum_{r=0}^{N/2-1}x[2r]e^{-jrk\frac{2\pi}{N/2}}+e^{-jk\frac{2\pi}{N}}\sum_{r=0}^{N/2-1}x[2r+1]e^{-jrk\frac{2\pi}{N/2}}$$

Two N/2 point DFTs of two N/2 point sequences

$$=G[k]+e^{-jk\frac{2\pi}{N}}H[k]$$

where $g[n]=x[2n]\overset{DFT}{\underset{N/2}{\leftrightarrow}}G[k], h[n]=x[2n+1]\overset{DFT}{\underset{N/2}{\leftrightarrow}}H[k]$

$$X[k]=G[k]+e^{-jk\frac{2\pi}{N}}H[k] \qquad k=0,1,\ldots,\frac{N}{2}-1$$

For k ≥ N/2 ?

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Complexity of the merging stage (Derivation of FFT)

$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,...., N/2-1$$

Two N/2 point DFTs of two N/2 point sequences

$$X[k+N/2] = \sum_{r=0}^{N/2-1} x[2r] e^{-jr(k+N/2)\frac{2\pi}{N/2}} + e^{-j(k+N/2)\frac{2\pi}{N}} \sum_{r=0}^{N/2-1} x[2r+1] e^{-jr(k+N/2)\frac{2\pi}{N/2}}$$

$$= \sum_{r=0}^{N/2-1} x[2r] e^{-jrk\frac{2\pi}{N/2}} \underbrace{e^{-jrN/2\frac{2\pi}{N/2}}}_{1} + e^{-jk\frac{2\pi}{N}} \underbrace{e^{-jN/2\frac{2\pi}{N}}}_{-1} \sum_{r=0}^{N/2-1} x[2r+1] e^{-jrk\frac{2\pi}{N/2}} \underbrace{e^{-jrN/2\frac{2\pi}{N/2}}}_{1}$$

$$= \sum_{r=0}^{N/2-1} x[2r] e^{-jrk\frac{2\pi}{N/2}} - e^{-jk\frac{2\pi}{N}} \sum_{r=0}^{N/2-1} x[2r+1] e^{-jrk\frac{2\pi}{N/2}} = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,...., \frac{N}{2}-1$$

From two N/2 point DFTs to an N point DFT

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Complexity of the merging stage

1.Split signal into even and odd samples

$$g[n] = x[2n], \quad h[n] = x[2n+1]$$

2.Perform two N/2 point DFTs

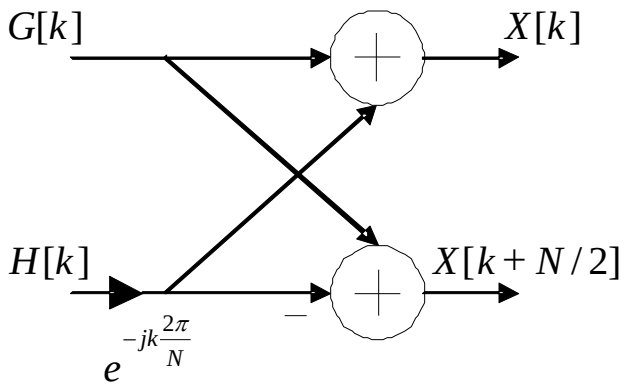
$$G[k], \quad H[k]$$

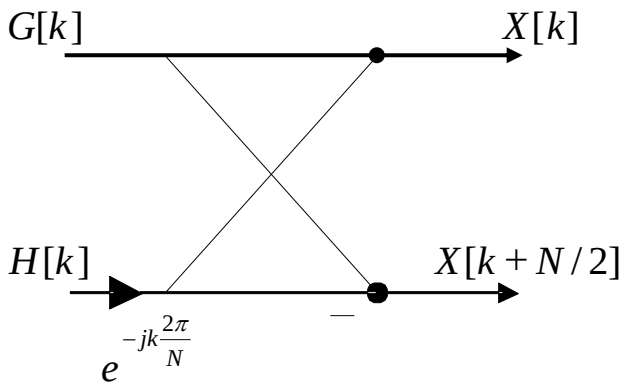
3.Combine into N point DFT

$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$0 \le k < N/2$$





Butterfly

1 multiplication

2 additions

N/2 butterflies at each stage

$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

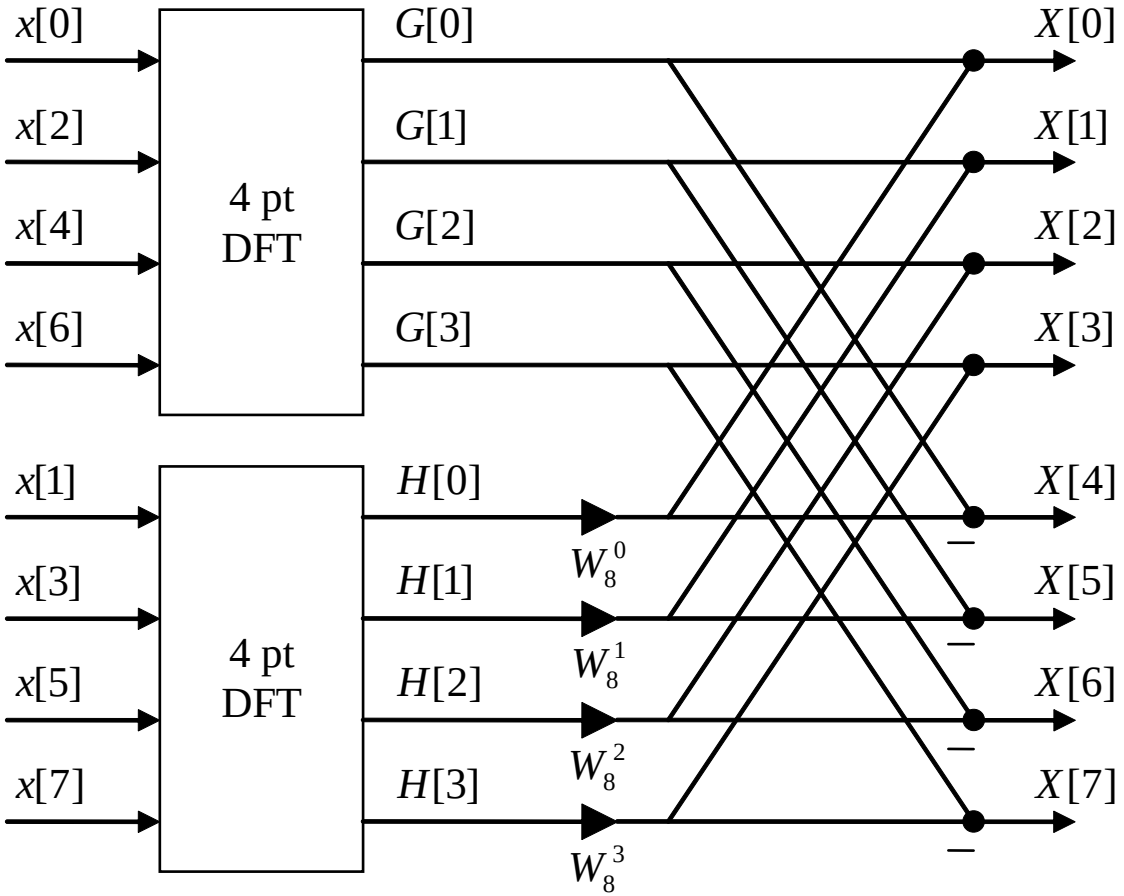
$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,\ldots,\frac{N}{2}-1$$

$$e^{-jk\frac{2\pi}{N}} = W_N^k$$

The Fast Fourier Transform

Example : DIT FFT for N=8



$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

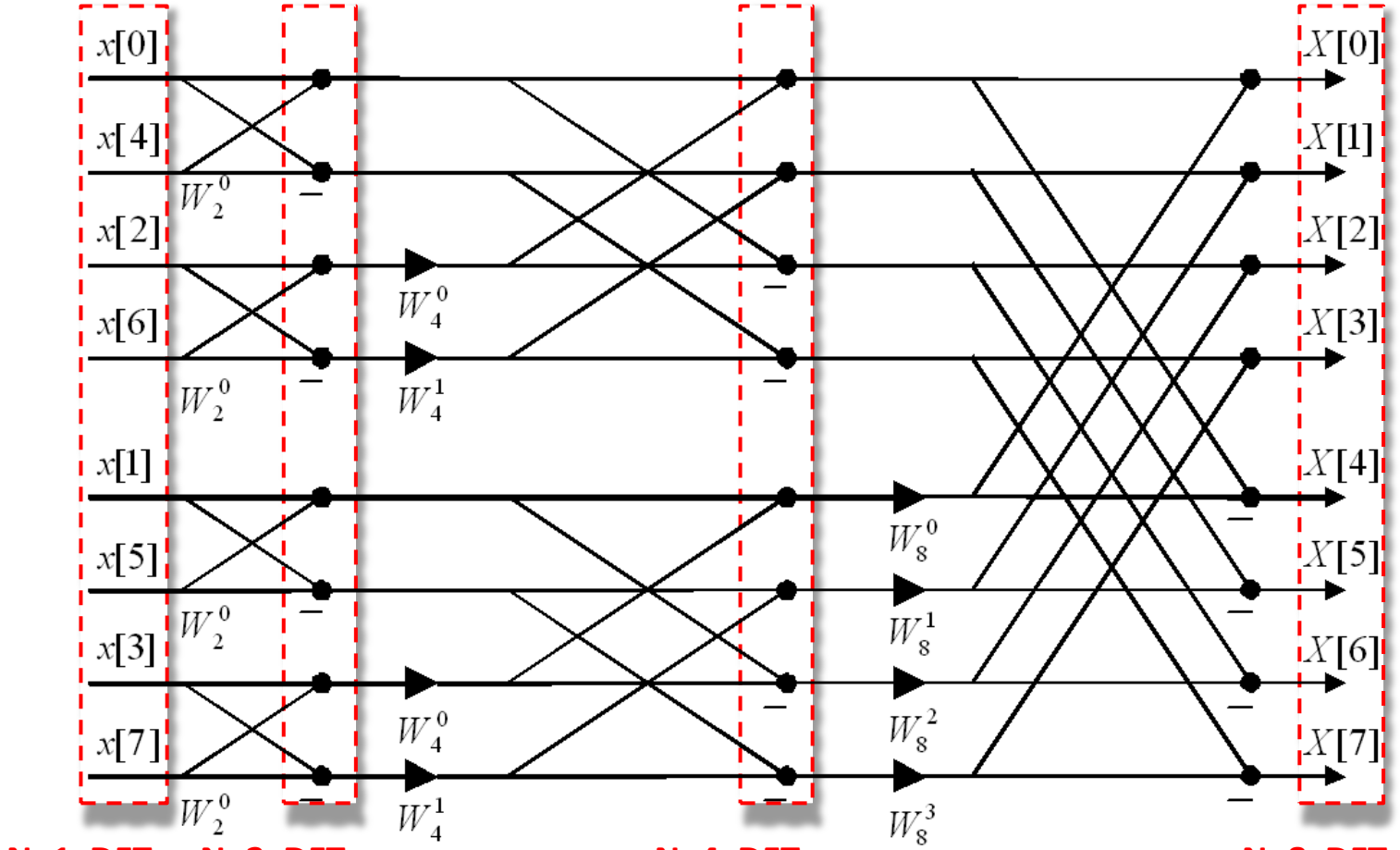
$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,....,\frac{N}{2} - 1$$

$$e^{-jk\frac{2\pi}{N}} = W_N^k$$

The Fast Fourier Transform

Example : DIT FFT for N=8



$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

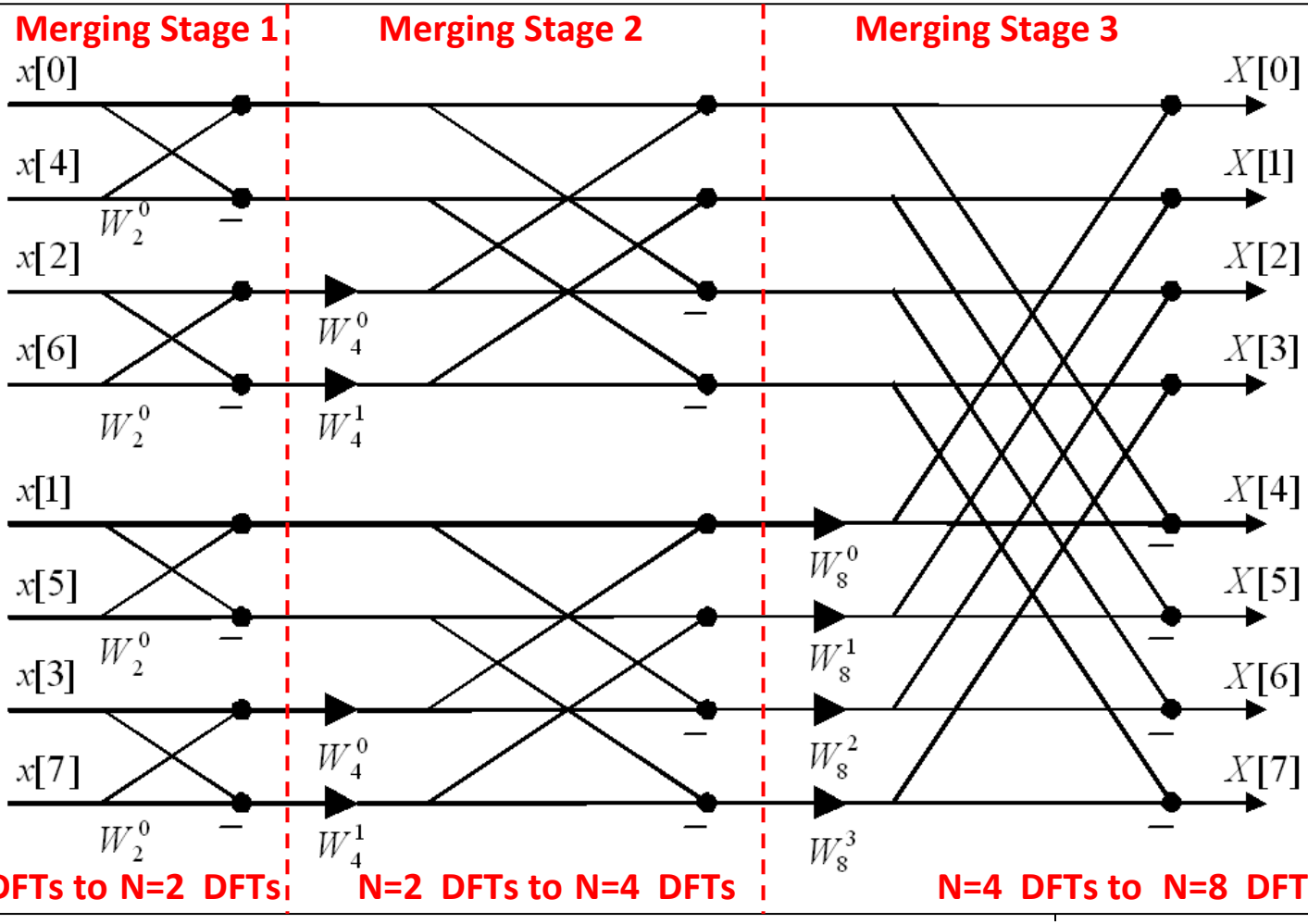
$$k = 0,1,...., \frac{N}{2} - 1$$

$$e^{-jk\frac{2\pi}{N}} = W_N^k$$

The Fast Fourier Transform

Example : DIT FFT for N=8

$\log_2(N)$ Stages



$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,\ldots,\frac{N}{2}-1$$

$$e^{-jk\frac{2\pi}{N}} = W_N^k$$

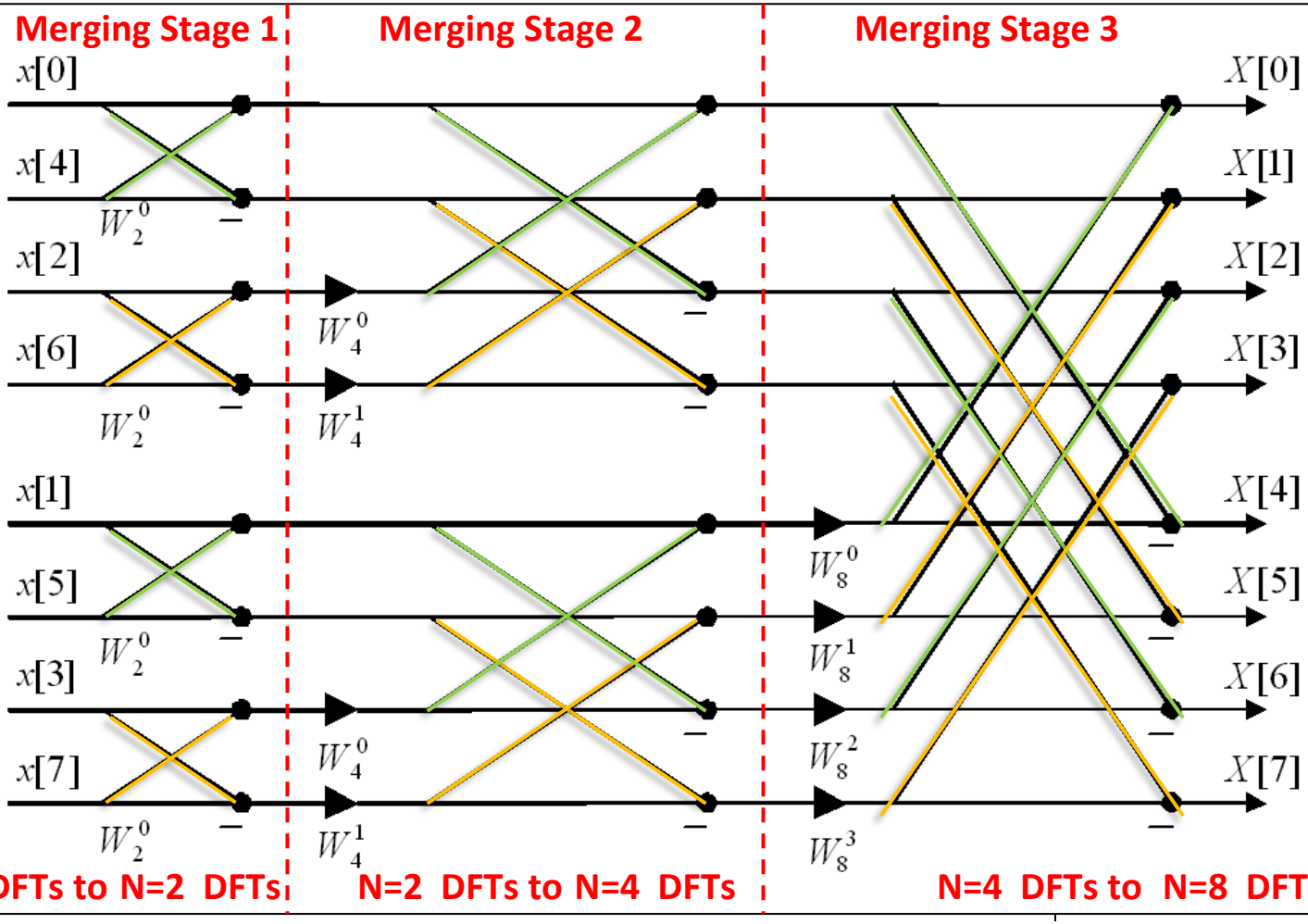
The Fast Fourier Transform

Example : DIT FFT for N=8

$\log_2(N)$ Stages

N/2 Butterflies per stage

1 multiply
2 additions
per Butterfly



$$X[k] = G[k] + e^{-jk\frac{2\pi}{N}} H[k]$$

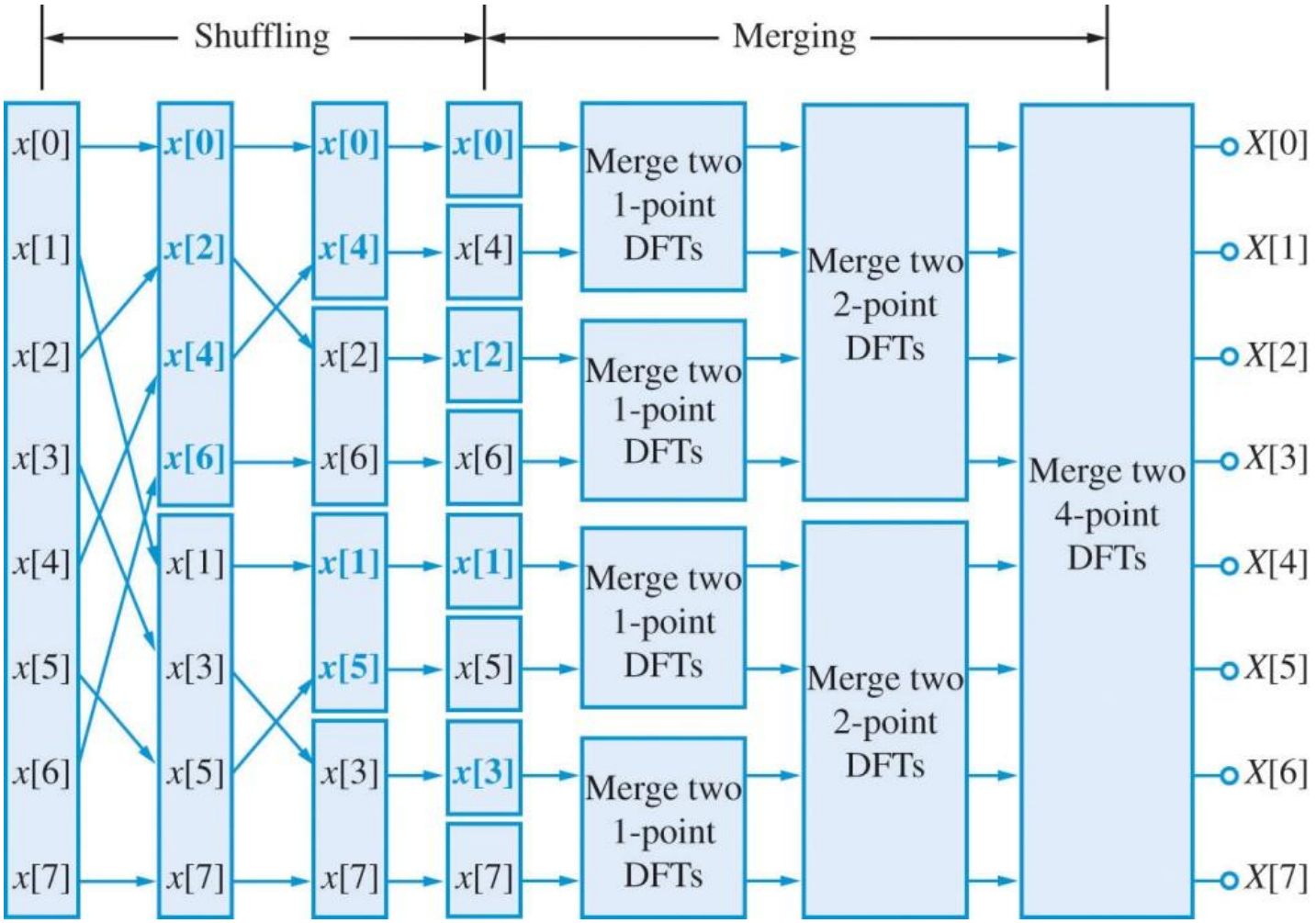
$$X[k+N/2] = G[k] - e^{-jk\frac{2\pi}{N}} H[k]$$

$$k = 0,1,\ldots,\frac{N}{2}-1$$

$$e^{-jk\frac{2\pi}{N}} = W_N^k$$

The Fast Fourier Transform

Implementation Aspects of the DIT FFT

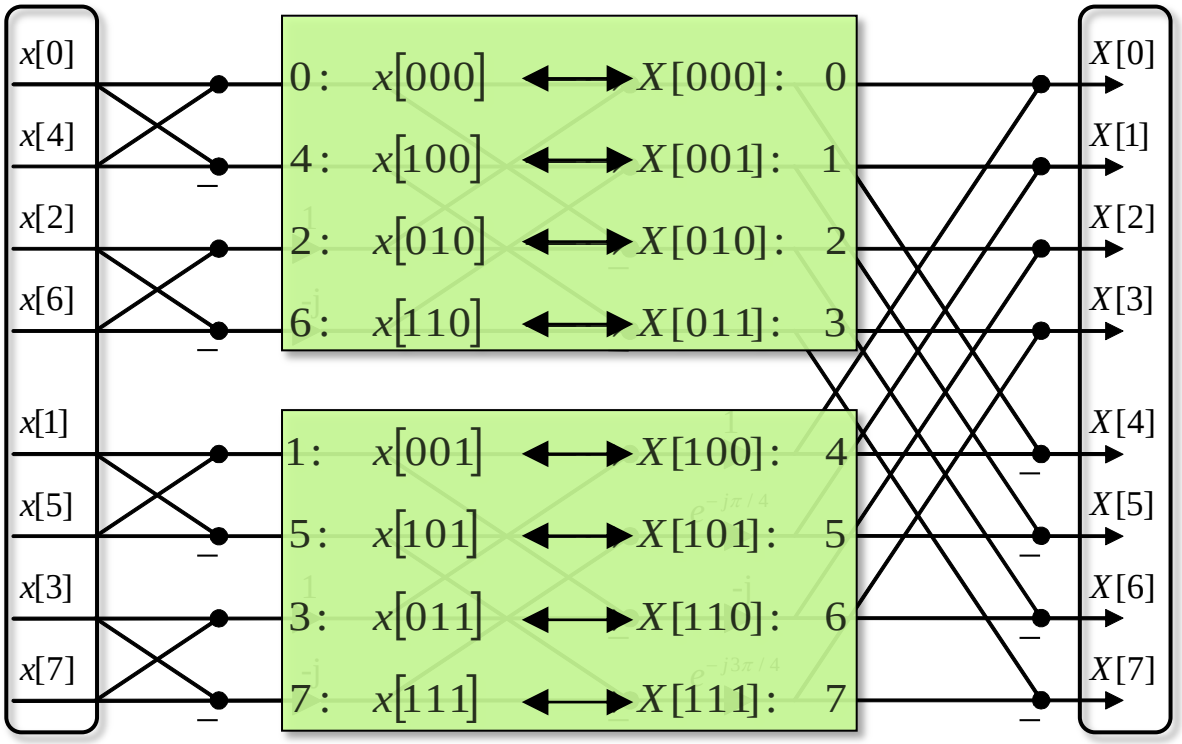


The Fast Fourier Transform

Implementation Aspects of the DIT FFT

Shuffling: Bit reversed addressing

Reordering is achieved by reading the binary representation of index n in reverse order



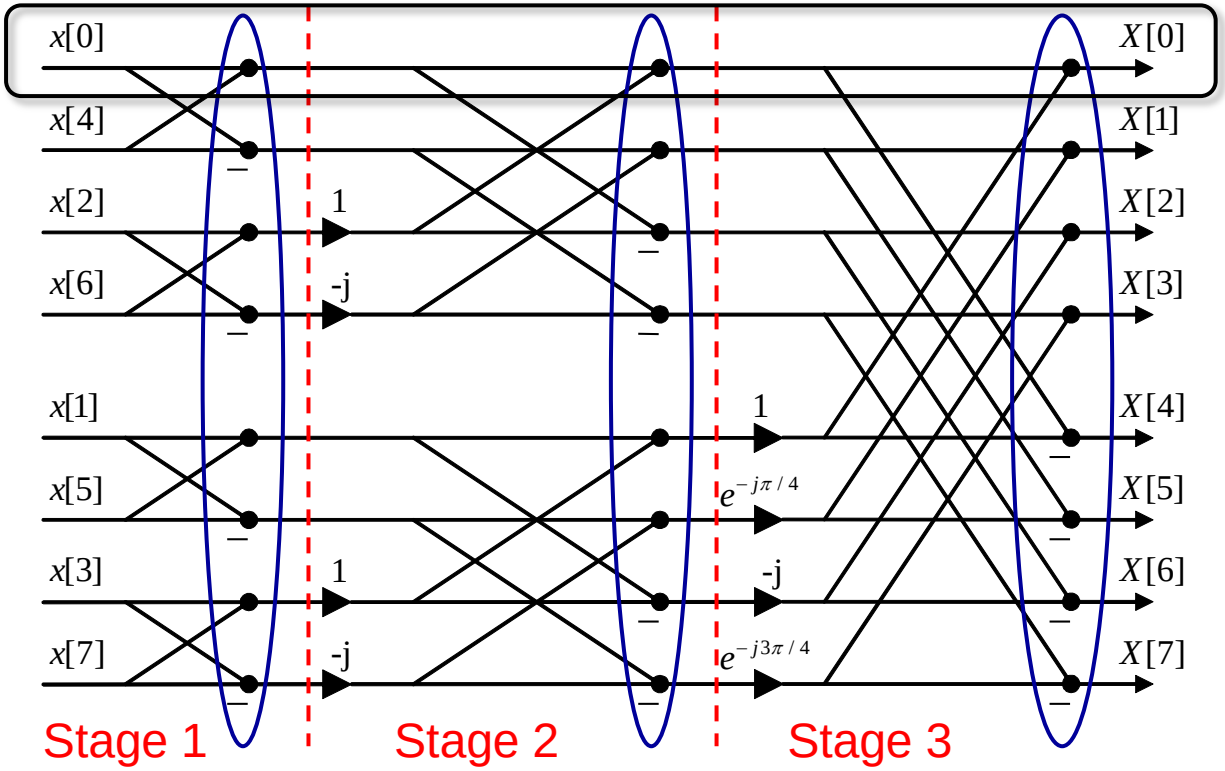
Outputs X[k], k=0,1,2,3,4,5,6,7 are stored in locations used by inputs x[n], n=0,4,2,6,1,5,3,7
Writing these indices in binary we find that they can be obtained by reversing them.

The Fast Fourier Transform

Implementation Aspects of the DIT FFT

In place computation

Node values (intermediate results) on the same horizontal path in the FFT flow-graph are only needed for computations in the following stage.



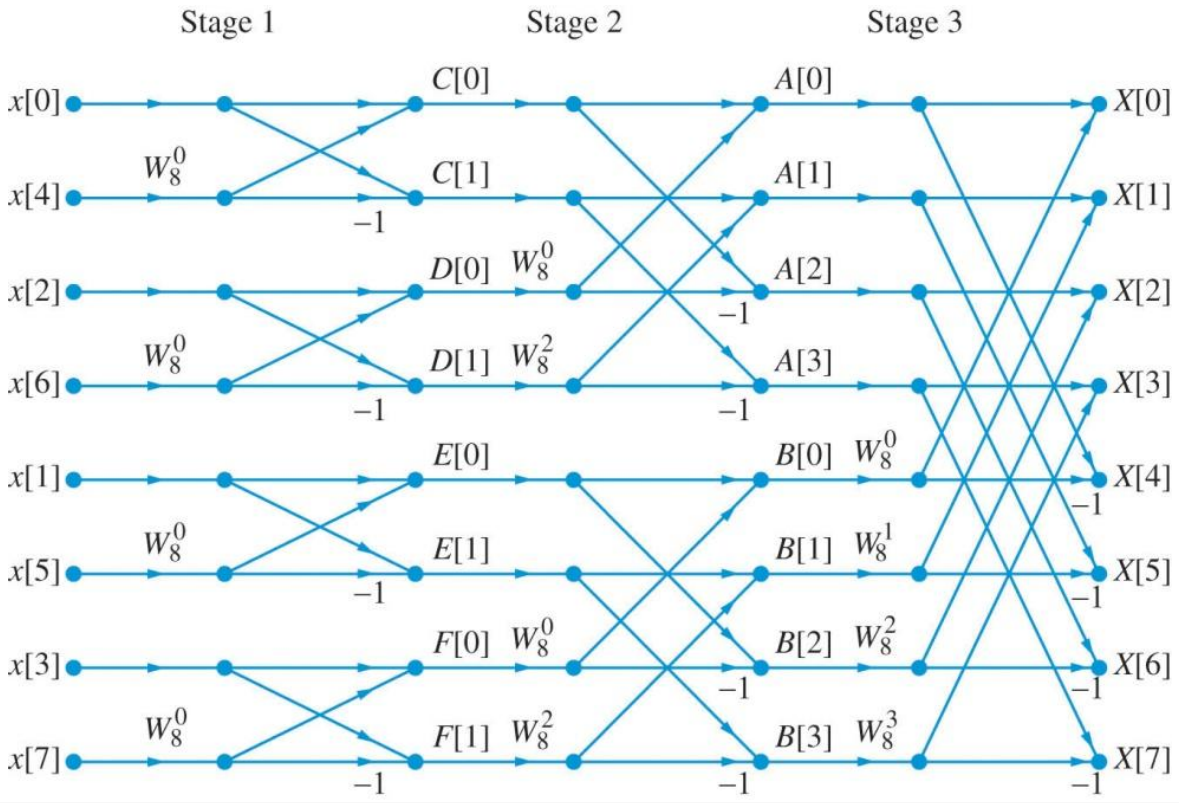
The 2 output values from each butterfly operation can be stored in the same memory location as the 2 input values (overwriting) => Fixed amount of storage: 2N Storage Registers

The Fast Fourier Transform

Implementation Aspects of the DIT FFT

Twiddle factors : Look-up tables

N/2 Twiddle factor values (those of stage $\log_2 N$) stored in look-up table



$$W_N^k = e^{-jk\frac{2\pi}{N}} \qquad W_{N/2}^k = e^{-jk\frac{2\pi}{N/2}} = e^{-j2k\frac{2\pi}{N}} = W_N^{2k}$$

The Fast Fourier Transform

Decimation in Time (DIT) FFT

Complexity (Radix 2 FFT with N a power of 2)

- $\log_2(N)$ merging stages
- N/2 butterflies per stage
- 1 multiplication and 2 additions per butterfly

DFT : Complexity $O(N^2)$	**FFT : Complexity $O(N\log_2 N)$**
$C_{m(DFT)} = N^2$ Multiplications	$C_{m(FFT)} = \frac{N}{2}\log_2(N)$ Multiplications
$C_{A(DFT)} = N(N-1)$ Additions	$C_{A(FFT)} = N\log_2(N)$ Additions

N	DFT multiplies	FFT multiplies	Ratio
64	4096	192	21
256	65536	1024	64
1024	1048576	5120	205
4096	16777216	24596	683

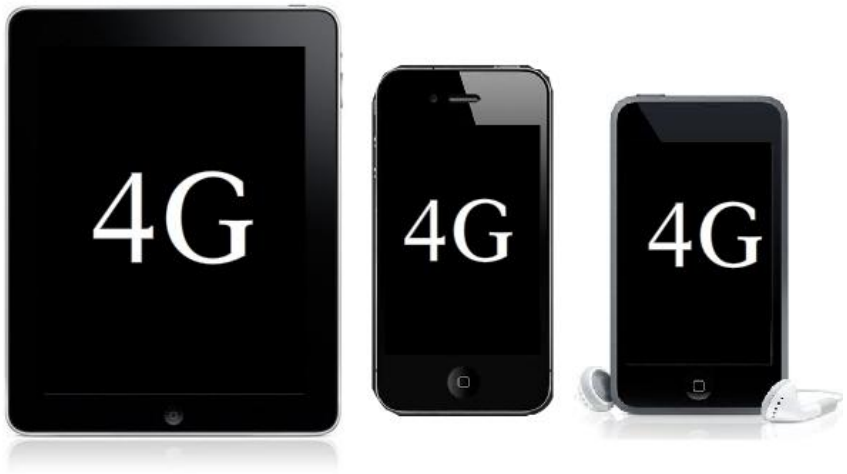
The Fast Fourier Transform

Discrete Fourier Transform - DFT

Why a fast Implementation ?

Wi-Fi

- A 64-DFT every 4 µsec (250,000 times/sec)
- **Direct Implementation of DFT:**
  - 4096 x 250,000 multiplications / second
  - 1,024,000,000 multiplications / second
- **FFT:**
  - 192 x 250,000 multiplications / second
  - 48,000,000 multiplications / second



- A 1024-DFT every 67 µsec (15,000 times/sec)
- **Direct Implementation of DFT:**
  - 1048576 x 15000 multiplications / second
  - 15,728,640,000 multiplications / second
- **FFT:**
  - 5120 x 15,000 multiplications / second
  - 76,800,000 multiplications / second

The Fast Fourier Transform

Discrete Fourier Transform Calculation

Fast Implementations

FFT Chip Comparisons

This table lists key features of commercial and academic FFT processors. The only requirement is that they are able to compute a 1024-point complex transform.

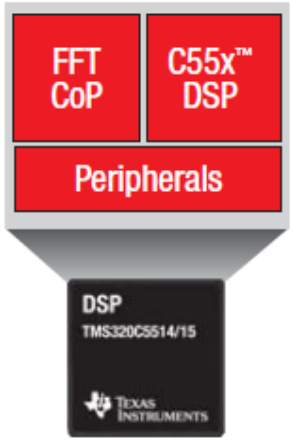
Processor	Year	CMOS Tech (µm)	Datapath Width (bits)	Dataword Format (fixed pt., block float, float pt.)	Supply Voltage (V)	Execution Time (µsec /1024-pt xform)	Power (mW)	Clock (MHz)	Number of Chips M=DataMem C=CoeffMem	Area, 0.5µ eff (mm^2)	I/O pads (pins)	Energy Efficiency (FFTs per Energy) *See below*	Processor
DASP/PAC Honeywell [1]	1988	1.2µm	16	block float	-	102µsec	2000 + 2000 + 5*250	-	1+1+4M+C	-	269/180/-	1.7	DASP/PAC Honeywell
PDSP16510A Zarlink (Plessey,Mitel)	1989?	1.4µm	16	block float	5.0	98µsec	3000	40	1	22	84	3.6	PDSP16510A Plessey
PDSP16515A Zarlink (Plessey,Mitel)	-	-	18	block float	5.0	87µsec	-	45	1	-	84	-	PDSP16515A Plessey
L64280 LSI [5]	1990	1.5µm	20	float	5.0	26µsec	20,000	40	10+10	233	-	2.9	L64280 LSI
Dassault Electronique [6]	1990	1.0µm	12	block float	5.0	6.4µsec	24,000?	40	6	255	299	3.4	Dassault Electronique
Y. Zhu, Univ. of Calgary [7]	1993	1.2µm	16	block float	5.0	155µsec	-	33	1+1+2M+C	15+	132	-	Y. Zhu Univ. of Calgary
TM-66 Texas Mem Sys	-	0.8µm	32	float	5.0	65µsec	7000+	50	1+1+M+	-	299/?	<3.4	TM-66 Texas Mem Sys
BDSP9124/9320 Butterfly DSP	-	0.8µm	24	block float	-	54µsec	-	60	1+1+2M+C+	-	262/68	-	BDSP9124/9320 Butterfly DSP
Cobra, Colorado State [10]	1994	0.75µm	23	-	5.0	9.5µsec	7700	40	16+	1104+	391	<12.4	Cobra Colorado State

See Blackboard for full list (list not up to date)

The Fast Fourier Transform

Discrete Fourier Transform Calculation

Fast Implementations



Technical details

TMS320C5514 DSP:

- Highly-integrated peripherals reduce system cost and enable more user-friendly portable features:
  - Three on-chip LDOs
  - High-speed USB 2.0
  - I^2S
  - UART
  - SPI
  - MMC/SD
  - GPIOs
- Up to 256 KB of on-chip memory saves both power and system cost by reducing the need for external memory

TMS320C5515 DSP:

- C5515 DSP builds on the C5514 DSP with an additional 64 KB on-chip memory (320 KB total)
- Up to 1024-point programmable FFT hardware accelerator
- Integrated LCD display controller and 10-bit, 4-channel SAR ADC – reduce system cost and enable more user-interactive portable features
- Scalable and pin-to-pin compatible with the C5514 DSP allowing for the ability to design an entire product portfolio using the same hardware and software platform

Applications

- Portable audio recording
- Wireless microphone
- Noise cancellation headphones
- Medical monitoring
- Biometrics
- Smart sensors

The Fast Fourier Transform

Discrete Fourier Transform Calculation-Further Efficiencies

Efficient computation of the DFT of 2 real-valued sequences

The FFT algorithm is designed to perform complex multiplications and additions. We can exploit this for the computation of the DFT of two real-valued sequences

Calculate the DFT of two real-valued sequences $x_1[n]$ and $x_2[n]$ of length-N

1. Define a complex-valued sequence x[n] as:

$$x[n] = x_1[n] + jx_2[n] \quad \Rightarrow \quad x_1[n] = \frac{x[n] + x^*[n]}{2}, \quad x_2[n] = \frac{x[n] - x^*[n]}{2j}$$

---

2. Linearity of DFT means that: $X_1[k] = \frac{DFT\{x[n]\} + DFT\{x^*[n]\}}{2}, \quad X_2[k] = \frac{DFT\{x[n]\} - DFT\{x^*[n]\}}{2j}$

---

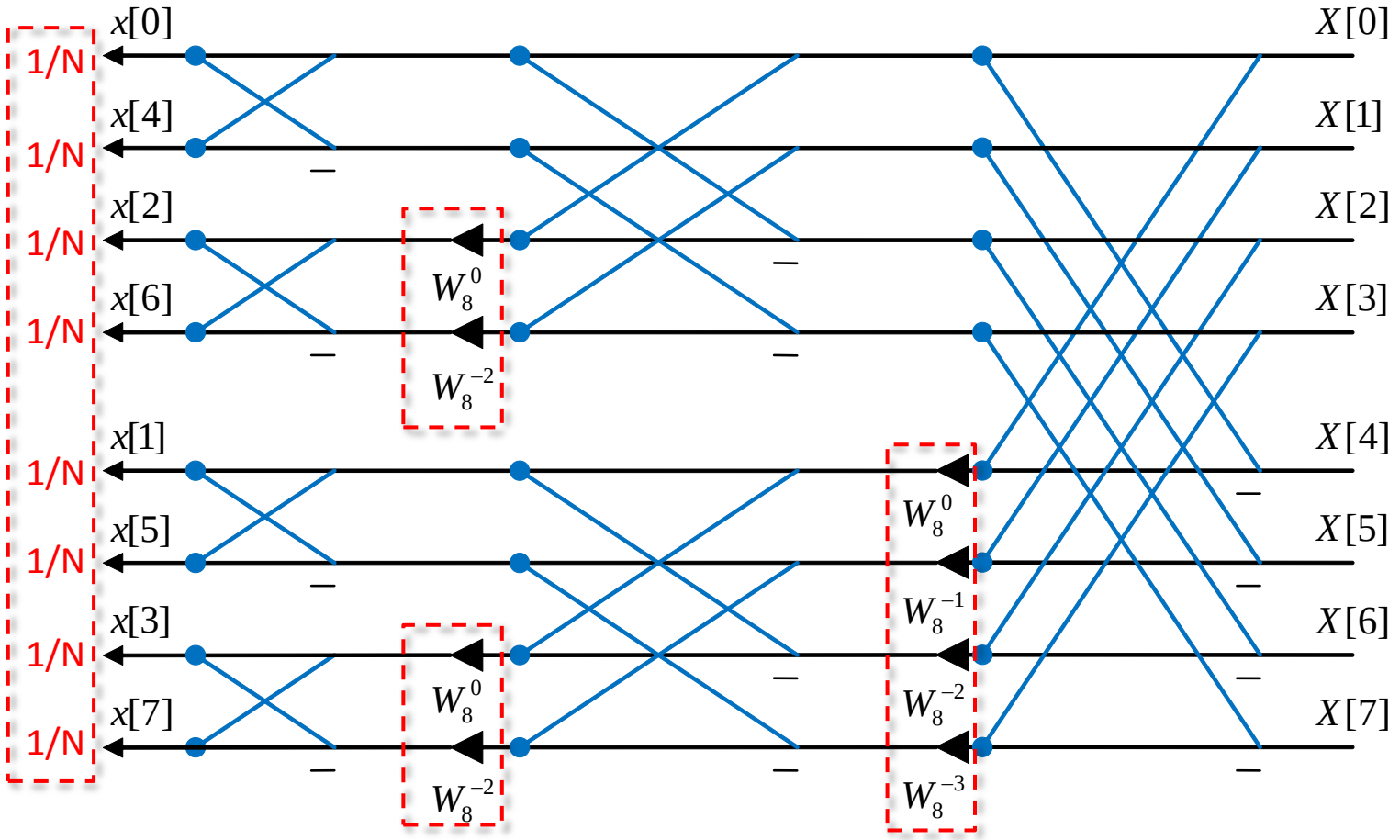
3. DFT Property $DFT\{x^*[n]\} = X^*[N-k]$ where $x[n] \overset{DFT}{\leftrightarrow} X[k]$ means that:

$$X_1[k] = \frac{X[k] + X^*[N-k]}{2}, \quad X_2[k] = \frac{X[k] - X^*[N-k]}{2j}$$

Two DFTs ($X_1[k]$ and $X_2[k]$) for the price of one ($X[k]$) plus some additional computation

The Fast Fourier Transform

Decimation in Time (DIT) Inverse FFT (IFFT)

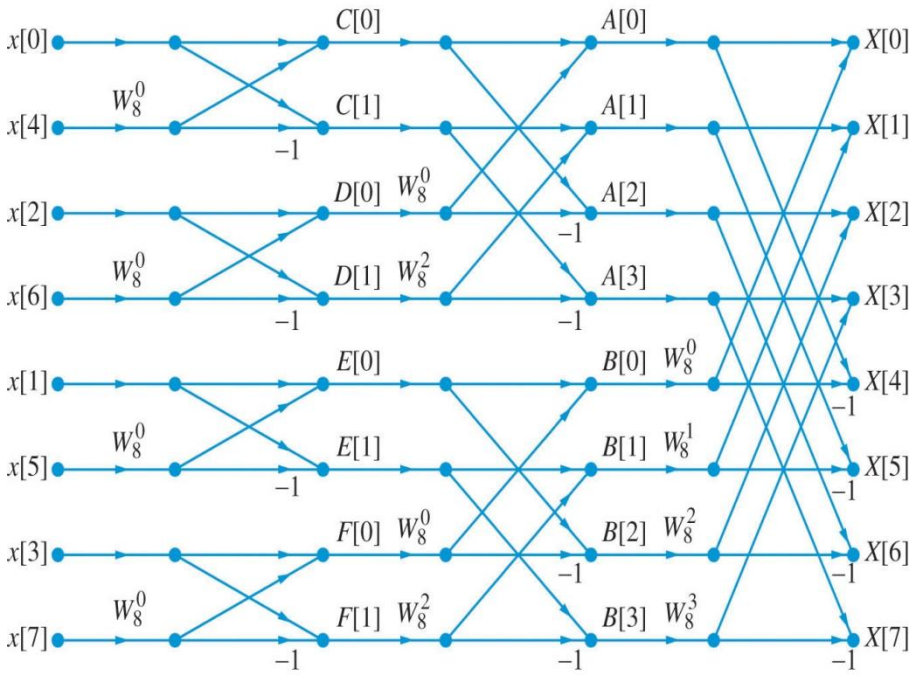


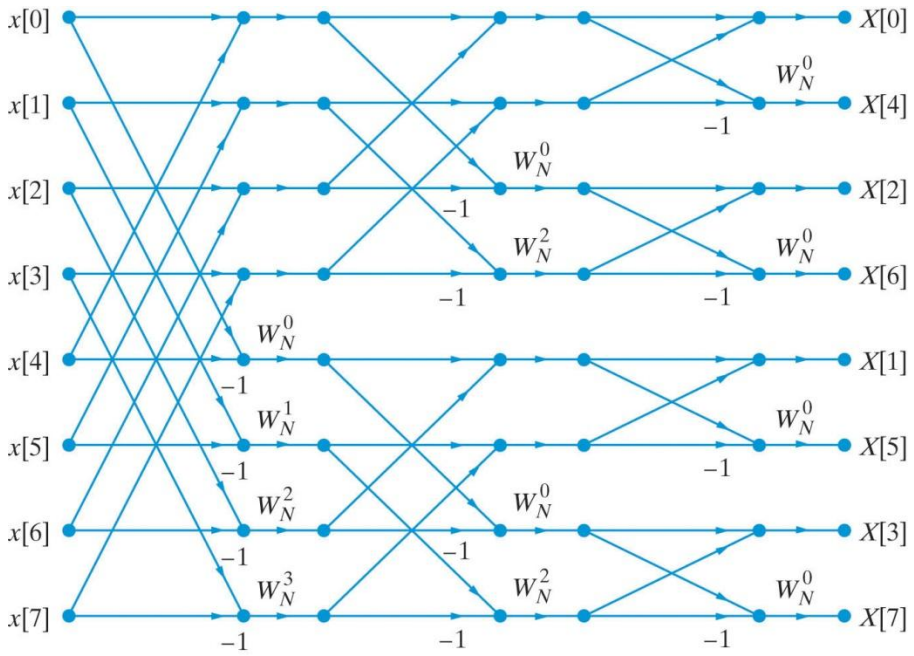
DFT	IDFT
$X(k)=\sum_{n=0}^{N-1}x[n]w_N^{kn}$ $k=0,1,\ldots,N-1$	$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X(k)w_N^{-kn}$ $n=0,1,\ldots,N-1$

The Fast Fourier Transform

Decimation in Frequency (DIF) FFT

Recursive decomposition of the DFT coefficients





Decimation in Time FFT

Reordering (bit reversal) takes place before the merging part (applied to input sequence)

Decimation in Frequency FFT

Reordering (bit reversal) takes place after the merging part (applied to output coefficients)

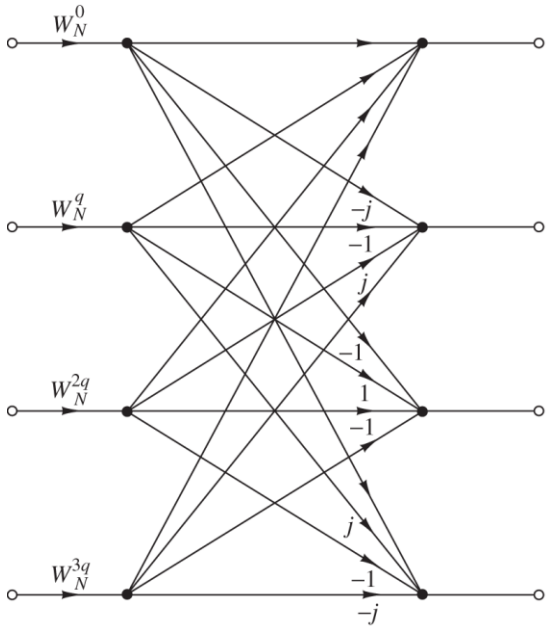
The Fast Fourier Transform

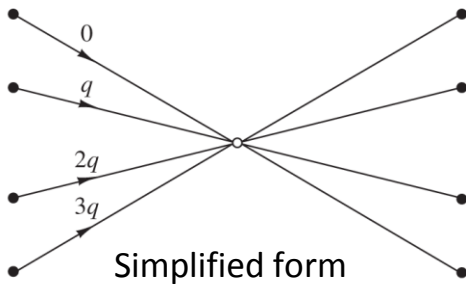
Other FFT Algorithms

Radix 4 FFT

- Previous algorithms were Radix-2. Same approach can be applied when N is power of R>2 leading to higher radix algorithms
- Radix-4 FFT: applicable when $N=4^v=2^{2v}$ (e.g. N=16 but not N=32)
- Basic idea: Decimate input by 4 recursively; compute 4-point DFTs through merging
- 25% less multiplications relative to Radix-2 FFT

Basic Butterfly Operation for Radix 4-FFT





Simplified form

q=0,1,2,3

Radix 4 Complexity

- v stages (2v for Radix2)
- N/4 butterflies per stage
- 3 multiplications per butterfly
- **Radix4 complex multiplications**

$$v \times \frac{N}{4} \times 3 = 3\left(\frac{N}{8} log_2 N\right)$$

- **Radix2 complex multiplications**

$$2v \times \frac{N}{2} \times 1 = 4\left(\frac{N}{8} log_2 N\right)$$

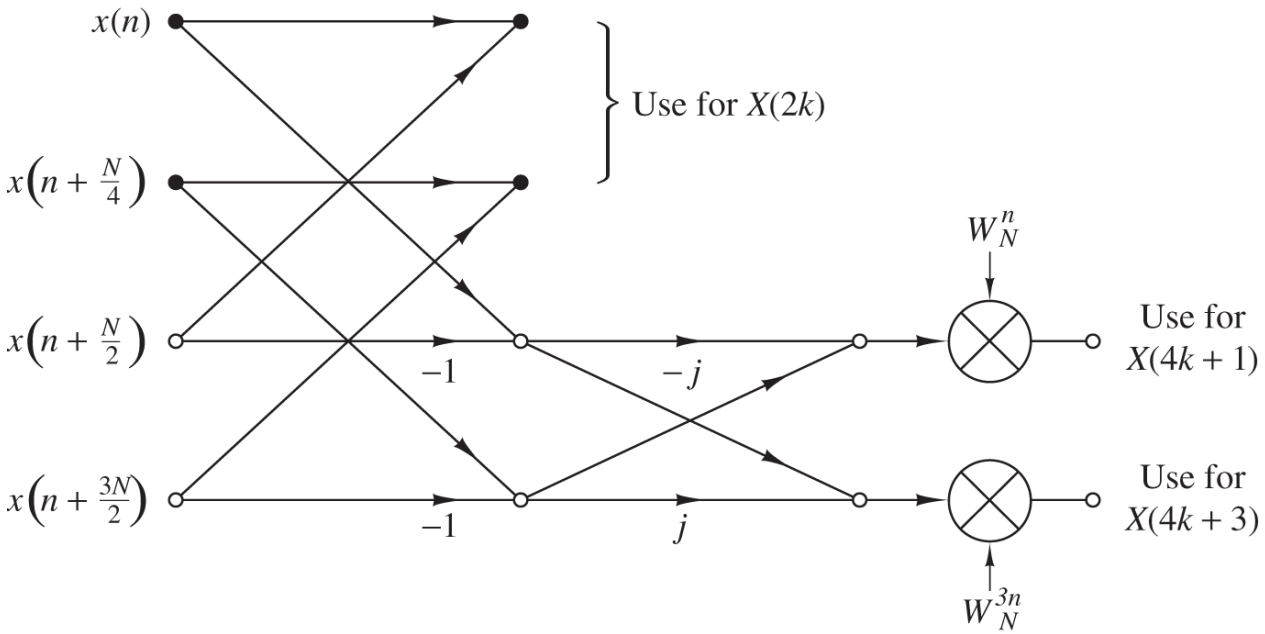
The Fast Fourier Transform

Other FFT Algorithms

Split Radix FFT

- Basic idea : In the DIF Radix-2 FFT even-numbered points of the DFT can be computed independently of the odd-numbered points - use different methods for independent parts
- Split-Radix FFT uses both Radix-2 (even points) and Radix-4 (odd points) DIF. It recursively expresses a DFT of length N in terms of one N/2 points DFT and two N/4 points DFTs.
- Further (small) reductions in number of computations achieved

Basic Butterfly Operation for SRFFT (notice irregular structure)



The Fast Fourier Transform

Other FFT Algorithms

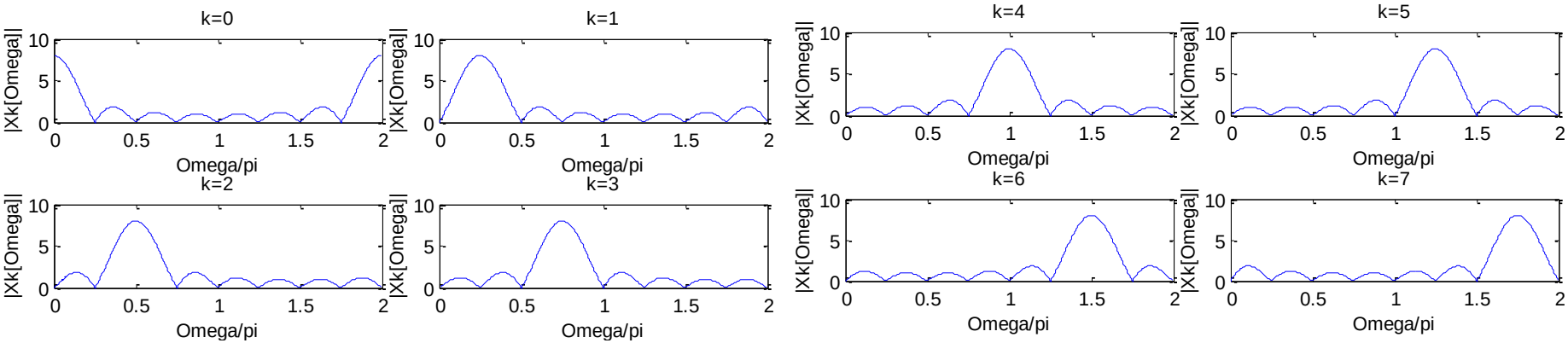
The Goertzel Algorithm

Some times not all points of the DFT are required – formulate DFT as a filter bank operation comprising of frequency selective resonators centred on frequencies k2π/N and calculate points needed.

$$\left.\begin{aligned} X[k] &= \sum_{r=0}^{N-1} x[r] W_N^{kr} \\ W_N^{-kN} &= e^{j(\frac{2\pi}{N})Nk} = e^{j2\pi k} = 1 \end{aligned}\right\} X[k] = W_N^{-kN} \sum_{r=0}^{N-1} x[r] W_N^{kr} \Leftrightarrow X[k] = \sum_{r=0}^{N-1} x[r] W_N^{-k(N-r)} \left\} X[k] = y_k[n]\right|_{n=N}$$

$$y_k[n] = \sum_{r=-\infty}^{\infty} x[r] W_N^{-k(n-r)} u[n-r]$$

Discrete convolution of the finite duration sequence x[n] with the sequence $W_N^{-kn}u[n]$ where X[k] is the value of the output when n=N



Choosing the fastest FFT algorithm

FFTW : Fastest Fourier Transform in the West

- FFT efficiency will vary with different machine architectures and different compilers
- FFTW : free software library that uses a planner to adapt its generalised Cooley-Tukey FFT algorithms to a given hardware platform thereby maximising efficiency
- Two stages : stage 1 planner; stage 2 : plan (program) execution.
- Planner searches over all possible factorizations of length N (decomposition into smaller transforms) and determines the one with the best performance for the target architecture
- The FFTW library is used in the implementation of FFT in Matlab
- http://www.fftw.org/