EE482/682: DSP APPLICATIONS

CH5: FREQUENCY ANALYSIS AND DFT

OUTLINE

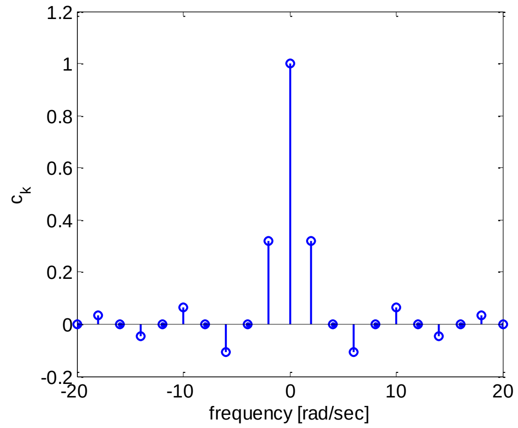
- Fourier Series
- Fourier Transform
- Discrete Time Fourier Transform
- Discrete Fourier Transform
- Fast Fourier Transform
- Butterfly Structure
- Implementation Issues

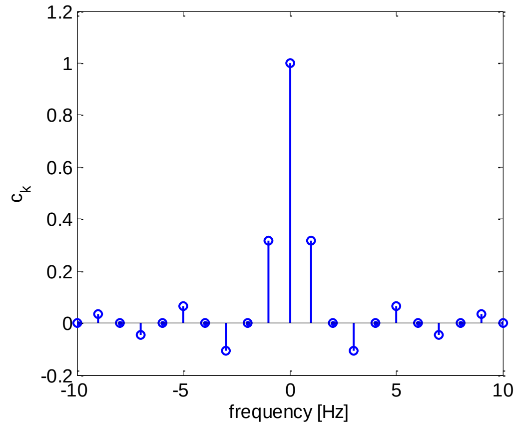
FOURIER SERIES

- Periodic signals
  - $x(t) = x(t + T_0)$
- Periodic signal can be represented as a sum of an infinite number of harmonically-related sinusoids
  - $x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t}$
  - c_k - Fourier series coefficients
    - Contribution of particular frequency sinusoid
  - $\Omega_0 = 2\pi/T_0$ - fundamental frequency
  - k – harmonic frequency index
- Coefficients can be obtained from signal
  - $c_k = \frac{1}{T_0}\int_0^{T_0} x(t) e^{-jk\Omega_0 t}$
  - Notice c_0 is the average over a period, the DC component

FOURIER SERIES EXAMPLE

- Example 5.1
- Rectangular pulse train
- $x(t) = \begin{cases} A & -\tau < t < \tau \\ 0 & else \end{cases}$
- $c_k = \dfrac{A\tau}{T_0}\dfrac{\sin(k\Omega_0\tau/2)}{k\Omega_0\tau/2}$
- $T = 1;$
- $\Omega_0 = 2\pi * \dfrac{1}{T} = 2\pi$
- Magnitude spectrum is known as a line spectrum
  - Only few specific frequencies represented





FOURIER TRANSFORM

- Generalization of Fourier series to handle non-periodic signals
- Let $T_0 \to \infty$
  - Spacing between lines in FS go to zero
    - $\Omega_0 = 2\pi/T_0$
- Results in a continuous frequency spectrum
  - Continuous function
- The number of FS coefficients to create "periodic" function goes to infinity

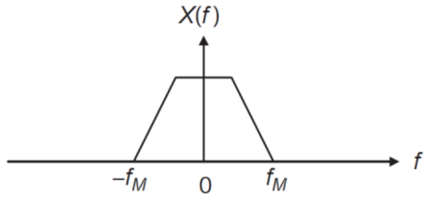
- Fourier representation of signal
  - $x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\Omega)e^{j\Omega t}d\Omega$
  - Inverse Fourier transform
- Fourier transform
  - $X(\Omega) = \int_{-\infty}^{\infty} x(\mathrm{t})e^{-j\Omega t}dt$
- Notice that a periodic function has both a FS and FT
  - $c_k = \frac{1}{T_0}X(k\Omega_0)$
  - Notice a normalization constant to account for the period

DISCRETE TIME FOURIER TRANSFORM

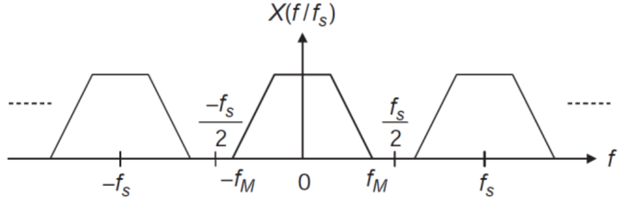
- Useful theoretical tool for discrete sequences/signals
- DTFT
  - $X(\omega) = \sum_{n=-\infty}^{\infty} x(nT)e^{-j\omega nT}$
  - Periodic function with period 2π
    - Only need to consider a 2π interval $[0,2\pi]$ or $[-\pi,\pi]$
- Inverse FT
  - $x(nT) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)e^{j\omega nT}\, d\omega$
  - Notice this is an integral relationship
    - $X(\omega)$ is a continuous function
    - Sequence $x(n)$ is infinite length

SAMPLING THEOREM

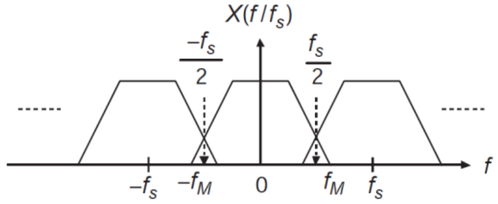
- Aliasing – signal distortion caused by sampling
  - Loss of distinction between different signal frequencies
- A bandlimited signal can be recovered from its samples when there is no aliasing
  - $f_s \geq 2f_m, \quad \Omega_s \geq 2\Omega_m$
    - f_s, Ω_s - signal bandwidth
- Copies of analog spectrum are copied at f_s intervals
  - Smaller sampling frequency compresses spectrum into overlap



(a) Spectrum of bandlimited analog signal.



(b) Spectrum of discrete-time signal when the sampling theorem $f_M \leq f_s/2$ is satisfied.



(c) Spectrum of discrete-time signal that shows aliasing when the sampling theorem is violated.

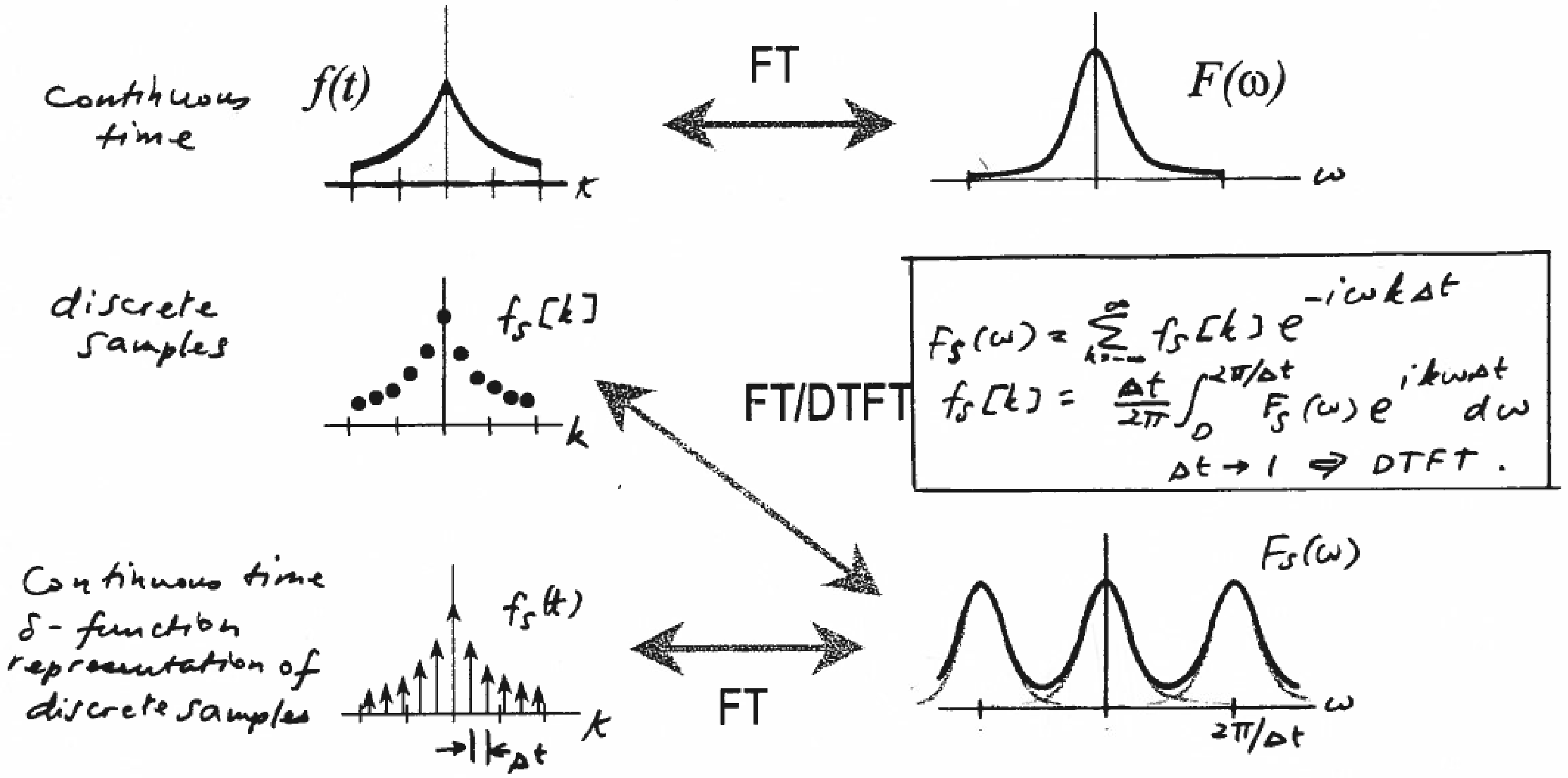
Figure 5.1 Spectrum replication of discrete-time signal caused by sampling

DISCRETE FOURIER TRANSFORM

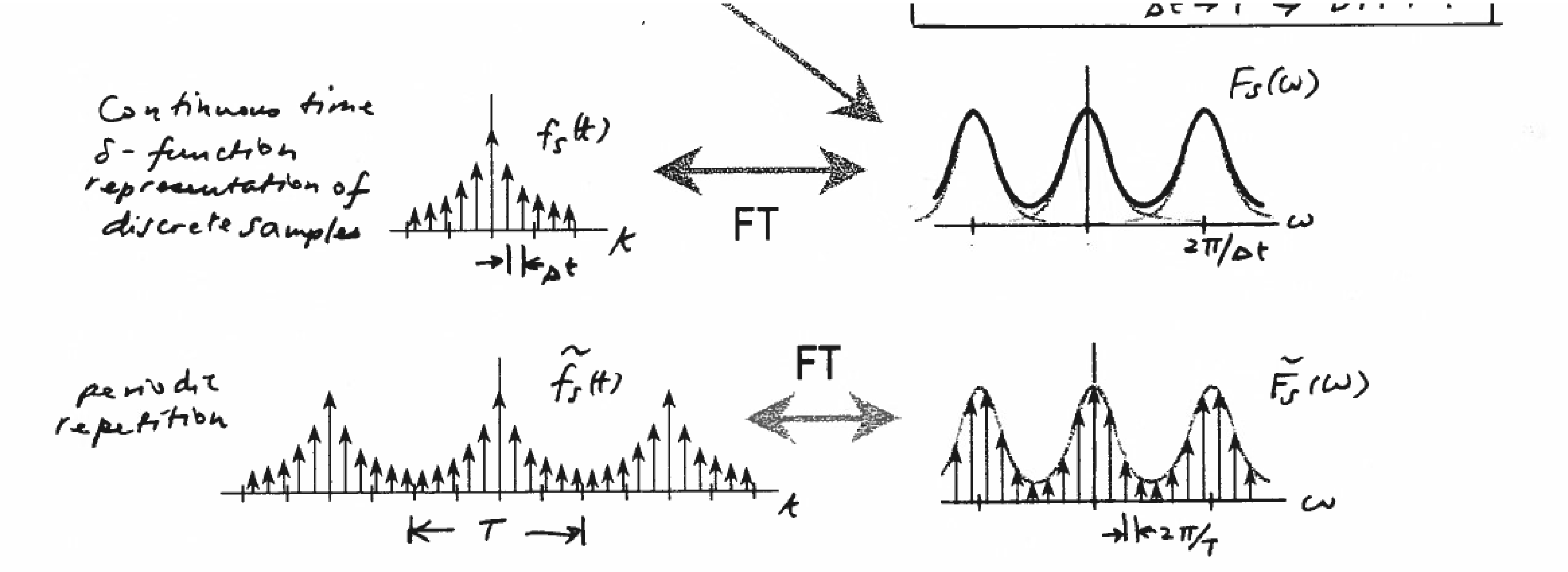
- Numerically computable transform used for practical applications
  - Sampled version of DTFT
- DFT definition
  - $X(k) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/\mathrm{N})kn}$
  - $k = 0, 1, \dots, N-1$: frequency index
  - Assumes $x(n) = 0$ outside bounds $[0, N-1]$
- Equivalent to taking N samples of DTFT $X(\omega)$ over the range $[0, 2\pi]$
  - N equally spaced samples at frequencies $\omega_k = 2\pi k/N$
    - Resolution of DFT is $2\pi/N$
- Inverse DFT
  - $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j(2\pi/\mathrm{N})kn}$

RELATIONSHIPS BETWEEN TRANSFORMS

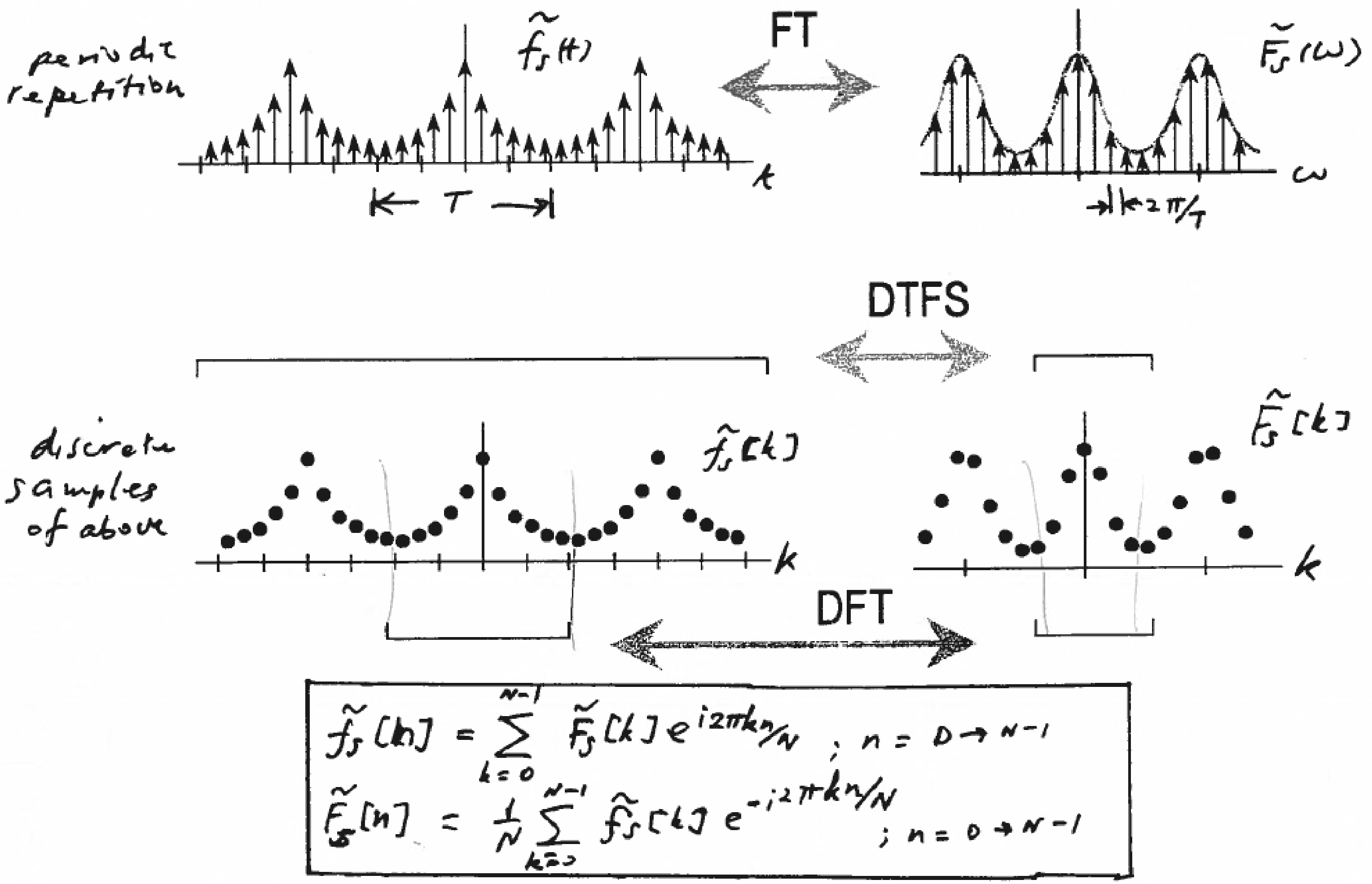
A bird's eye view of the relationship between FT, DTFT, DTFS and DFT



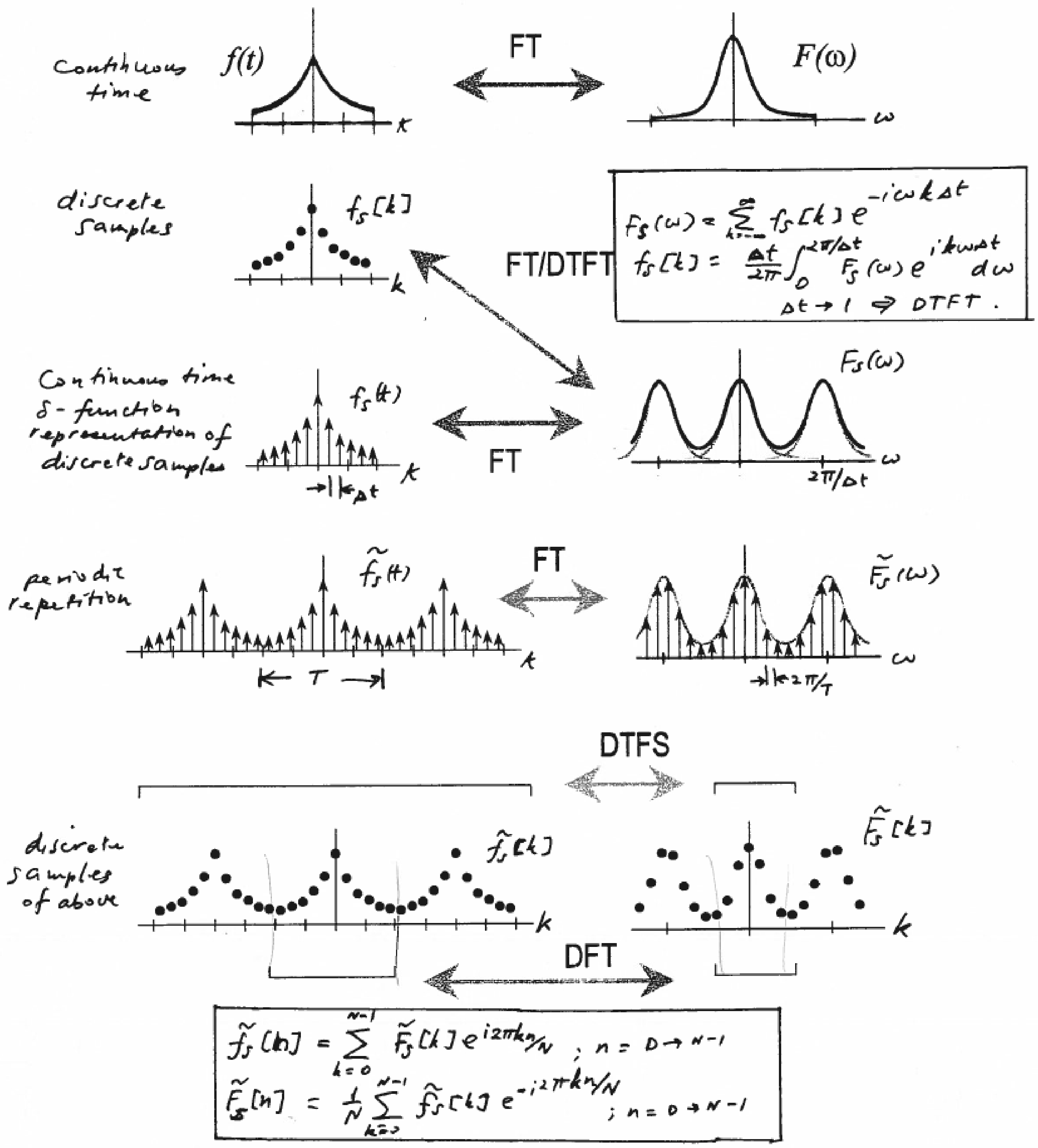
RELATIONSHIPS BETWEEN TRANSFORMS



RELATIONSHIPS BETWEEN TRANSFORMS



A bird's eye view of the relationship between FT, DTFT, DTFS and DFT



DFT TWIDDLE FACTORS

- Rewrite DFT equation using Euler's
- $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/\mathrm{N})kn}$
- $X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}$
  - $k = 0,1,\dots,N-1$
  - $W_N^{kn} = e^{-j(2\pi/N)kn} = \cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2\pi kn}{N}\right)$
- IDFT
- $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j(2\pi/\mathrm{N})kn}$
- $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, W_N^{-kn},$
  - $k = 0,1,\dots,N-1$

- Properties of twiddle factors
  - W_N^k - N roots of unity in clockwise direction on unit circle
  - Symmetry
    - $W_N^{k+N/2} = -W_N^k,\ 0 \le k \le \frac{N}{2} - 1$
  - Periodicity
    - $W_N^{k+N} = W_N^k$
- Frequency resolution
  - Coefficients equally spaced on unit circle
  - $\Delta = f_s/N$

DFT PROPERTIES

- Linearity
  - $DFT[ax(n) + by(n)] = aX(k) + bY(k)$
- Complex conjugate
  - $X(-k) = X^*(k)$
    - $1 \leq k \leq N-1$
    - For $x(n)$ real valued

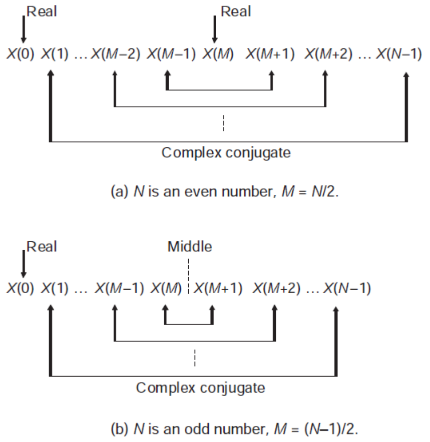


Figure 5.2 Complex-conjugate property for N is (a) an even number and (b) an odd number

  - Only first $M+1$ coefficients are unique
  - Notice the magnitude spectrum is even and phase spectrum is odd

- Z-transform connection
  - $X(k) = X(z)|_{z=e^{j(2\pi/N)k}}$
  - Obtain DFT coefficients by evaluating z-transform on the unit circle at N equally spaced frequencies $\omega_k = 2\pi k/N$
- Circular convolution
  - $Y(k) = H(k)X(k)$
  - $y(n) = h(n) \otimes x(n)$
  - $y(n) = \sum_{m=0}^{N-1} h(m)x((n-m)_{mod\ N})$
    - Note: both sequences must be padded to same length

FAST FOURIER TRANSFORM

- DFT is computationally expensive
  - Requires many complex multiplications and additions
  - Complexity $\sim 4N^2$
- Can reduce this time considerably by using the twidle factors
  - Complex periodicity limits the number of distinct values
  - Some factors have no real or no imaginary parts
- FFT algorithms operate in $N \log_2 N$ time
  - Utilize radix-2 algorithm so $N = 2^m$ is a power of 2

FFT DECIMATION IN TIME

- Compute smaller DFTs on subsequences of $x(n)$
- $X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}$
- $X(k) = \sum_{m=0}^{N/2-1} x_1(m)\, W_N^{k2m} + \sum_{m=0}^{N/2-1} x_2(m)\, W_N^{k(2m+1)}$
  - $x_1(m) = g(n) = x(2m)$ - even samples
  - $x_2(m) = h(n) = x(2m+1)$ – odd samples
- Since $W_N^{2mk} = W_{N/2}^{mk}$
  - $X(k) = \sum_{m=0}^{N/2-1} x_1(m)\, W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} x_2(m)\, W_{N/2}^{km}$
    - $N/2$-point DFT of even and odd parts of $x(n)$
  - $X(k) = G(k) + W_N^k H(k)$
    - Full N sequence is obtained by periodicity of each $N/2$ DFT

FFT BUTTERFLY STRUCTURE

- Full butterfly (8-point)
- Simplified structure

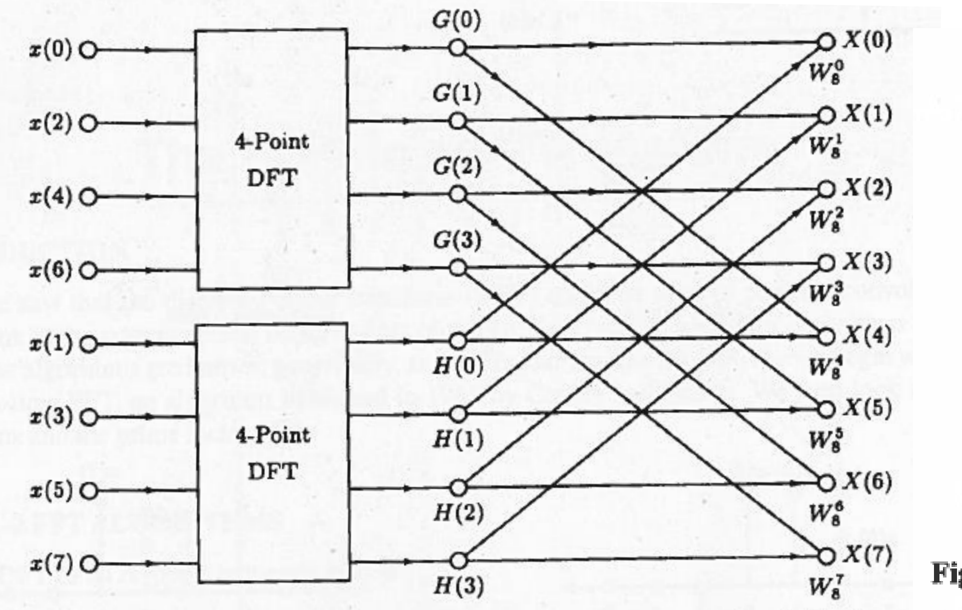


Fig. 7-2. An eight-point decimation-in-time FFT algorithm after the first decimation.

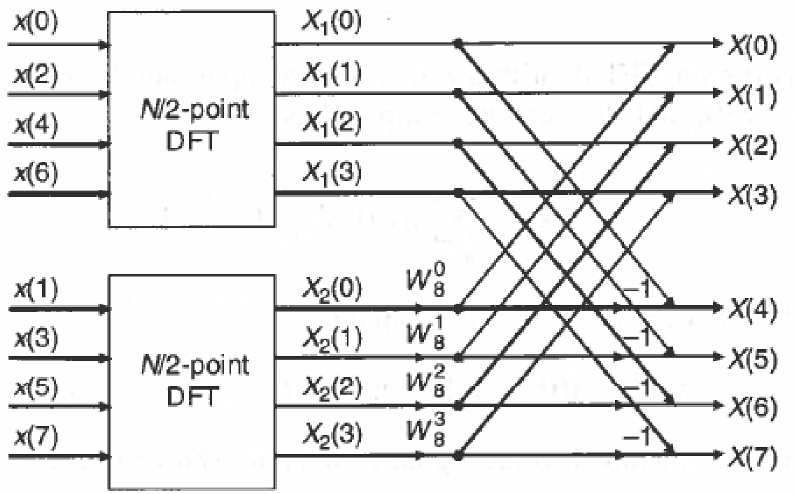


Figure 5.4 Decomposition of N-point DFT into two $N/2$-point DFTs, $N = 8$

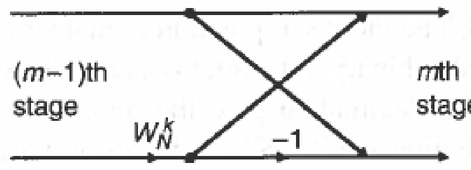


Figure 5.5 Flow graph for butterfly computation

FFT DECIMATION

- Repeated application of even/odd signal split
  - Stop at simple 2-point DFT

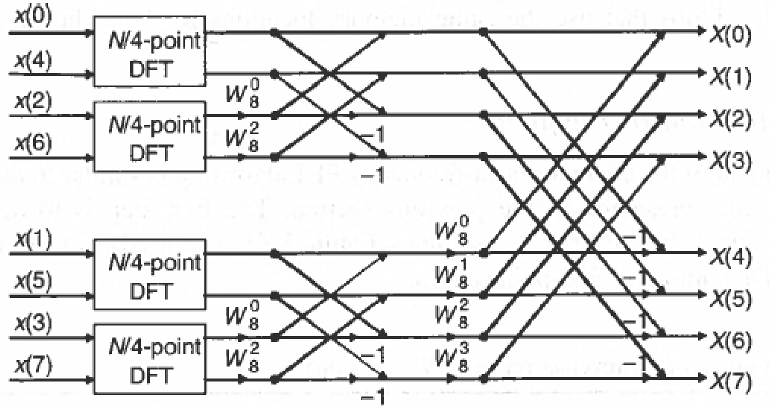


Figure 5.6 Flow graph illustrating second step of N-point DFT, $N=8$

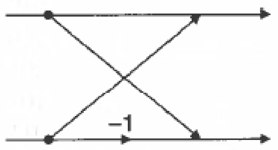


Figure 5.7 Flow graph of two-point DFT

- Complete 8-point DFT structure

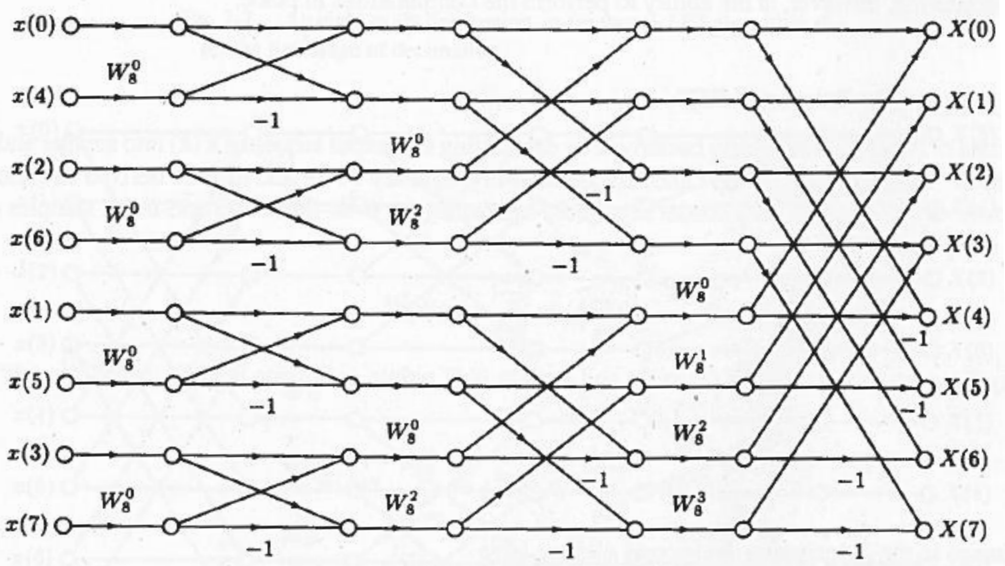


Fig. 7-6. A complete eight-point radix-2 decimation-in-time FFT.

FFT DECIMATION IN TIME IMPLEMENTATION

- Notice arrangement of samples is not in sequence – requires shuffling
  - Use bit reversal to figure out pairing of samples in 2-bit DFT

Table 5.1 Example of bit-reversal process, $N = 8$ (3-bit)

Input sample index		Bit-reversed sample index	
Decimal	*Binary*	*Binary*	*Decimal*
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

- Input values to DFT block are not needed after calculation
  - Enables in-place operation
    - Save FFT output in same register as input
  - Reduce memory requirements

DFT Algorithm

- The Fourier transform of an analogue signal x(t) is given by:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt$$

- The Discrete Fourier Transform (DFT) of a discrete-time signal x(nT) is given by:

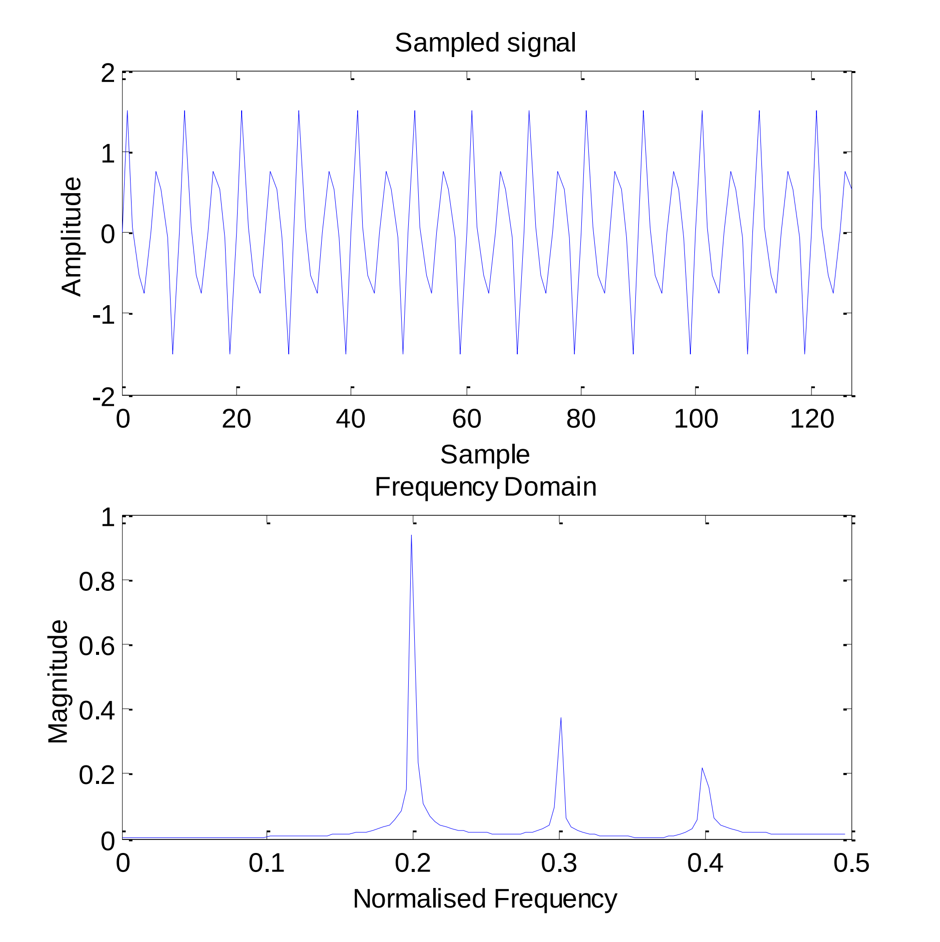
$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}$$

  - Where:

$$k = 0, 1, \ldots N-1$$
$$x(nT) = x[n]$$

DFT Algorithm

- If we let: $e^{-j\frac{2\pi}{N}} = W_N$ then: $X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}$



DFT Algorithm

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

x[n] = input

X[k] = frequency bins

W = twiddle factors

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

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

:

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

:

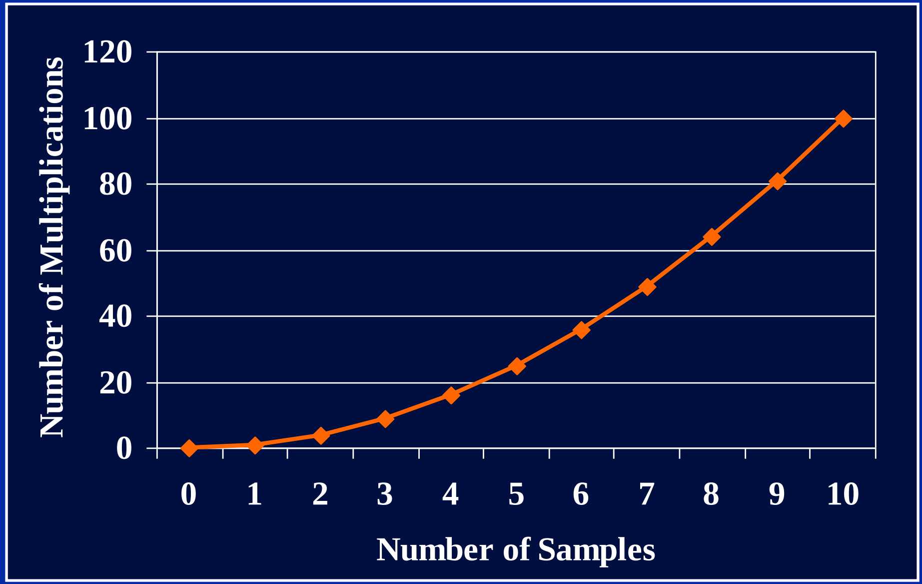
$$X(N-1) = x[0]W_N^{0} + x[1]W_N^{(N-1)*1} + \ldots + x[N-1]W_N^{(N-1)(N-1)}$$

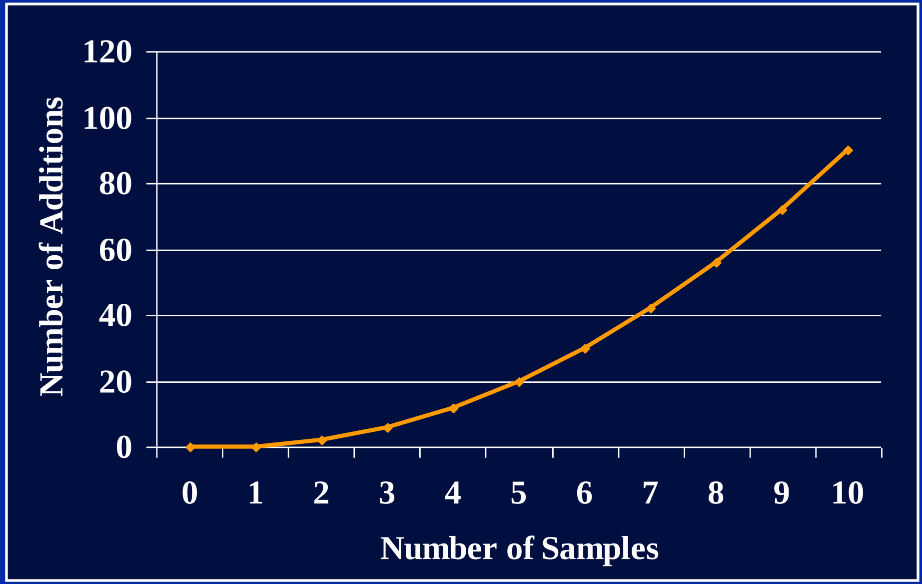
Note: For N samples of x we have N frequencies representing the signal.

Performance of the DFT Algorithm

- The DFT requires N^2 (NxN) complex multiplications:
  - Each X(k) requires N complex multiplications.
  - Therefore to evaluate all the values of the DFT (X(0) to X(N-1)) N^2 multiplications are required.
- The DFT also requires (N-1)*N complex additions:
  - Each X(k) requires N-1 additions.
  - Therefore to evaluate all the values of the DFT (N-1)*N additions are required.

Performance of the DFT Algorithm





- **Can the number of computations required be reduced?**

DFT $\rightarrow$ FFT

- A large amount of work has been devoted to reducing the computation time of a DFT.
- This has led to efficient algorithms which are known as the Fast Fourier Transform (FFT) algorithms.

DFT $\rightarrow$ FFT

$$X(k)=\sum_{n=0}^{N-1} x[n] W_N^{nk}; \quad 0 \le k \le N-1 \qquad [1]$$

x[n] = x[0], x[1], …, x[N-1]

- Lets divide the sequence x[n] into even and odd sequences:
  - x[2n] = x[0], x[2], …, x[N-2]
  - x[2n+1] = x[1], x[3], …, x[N-1]

DFT → FFT

- **Equation 1 can be rewritten as:**

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

- **Since:**

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

$$W_N^{(2n+1)k} = W_N^k \cdot W_{\frac{N}{2}}^{nk}$$

- **Then:**

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

$$= Y(k) + W_N^k Z(k)$$

DFT → FFT

- The result is that an N-point DFT can be divided into two N/2 point DFT's:

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

N-point DFT

- Where Y(k) and Z(k) are the two N/2 point DFTs operating on even and odd samples respectively:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{nk}$$
$$= Y(k) + W_N^k Z(k)$$

Two N/2-point DFTs

DFT → FFT

- **Periodicity** and **symmetry** of W can be exploited to simplify the DFT further:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{nk} \qquad [3]$$

$$\vdots$$

$$X\left(k+\frac{N}{2}\right) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)} + W_N^{k+\frac{N}{2}} \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)}$$

Or:

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

: Symmetry

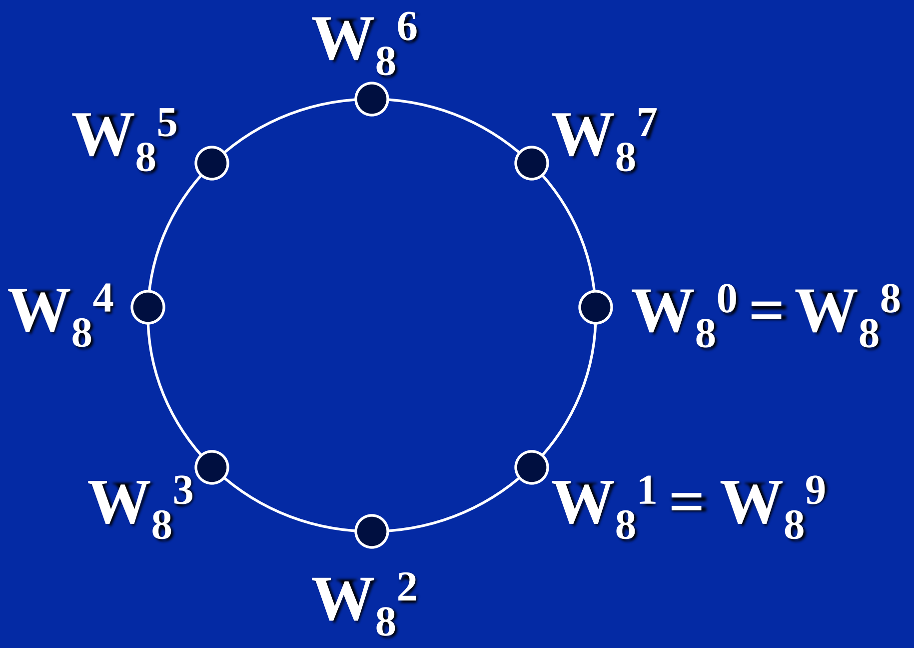
And:

$$W_{\frac{N}{2}}^{k+\frac{N}{2}} = e^{-j\frac{2\pi}{N/2}k} e^{-j\frac{2\pi}{N/2}\frac{N}{2}} = e^{-j\frac{2\pi}{N/2}k} = W_{\frac{N}{2}}^k$$

: Periodicity

DFT → FFT

- **Symmetry and periodicity:**



$$W_N^{k+N/2} = -W_N^k$$
$$W_{N/2}^{k+N/2} = W_{N/2}^k$$
$$W_8^{k+4} = -W_8^k$$
$$W_8^{k+8} = W_8^k$$

DFT → FFT

- **Finally by exploiting the symmetry and periodicity, Equation 3 can be written as:**

$$X\left(k+\frac{N}{2}\right) = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] W_{\frac{N}{2}}^{nk} - W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] W_{\frac{N}{2}}^{nk}$$
$$= Y(k) - W_N^k Z(k) \qquad [4]$$

DFT → FFT

$$X(k) = Y(k) + W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

$$X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

- **$Y(k)$ and $W_N^k Z(k)$ only need to be calculated once and used for both equations.**
- **Note: the calculation is reduced from 0 to N-1 to 0 to (N/2 - 1).**

DFT → FFT

$$X(k) = Y(k) + W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

$$X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)$$

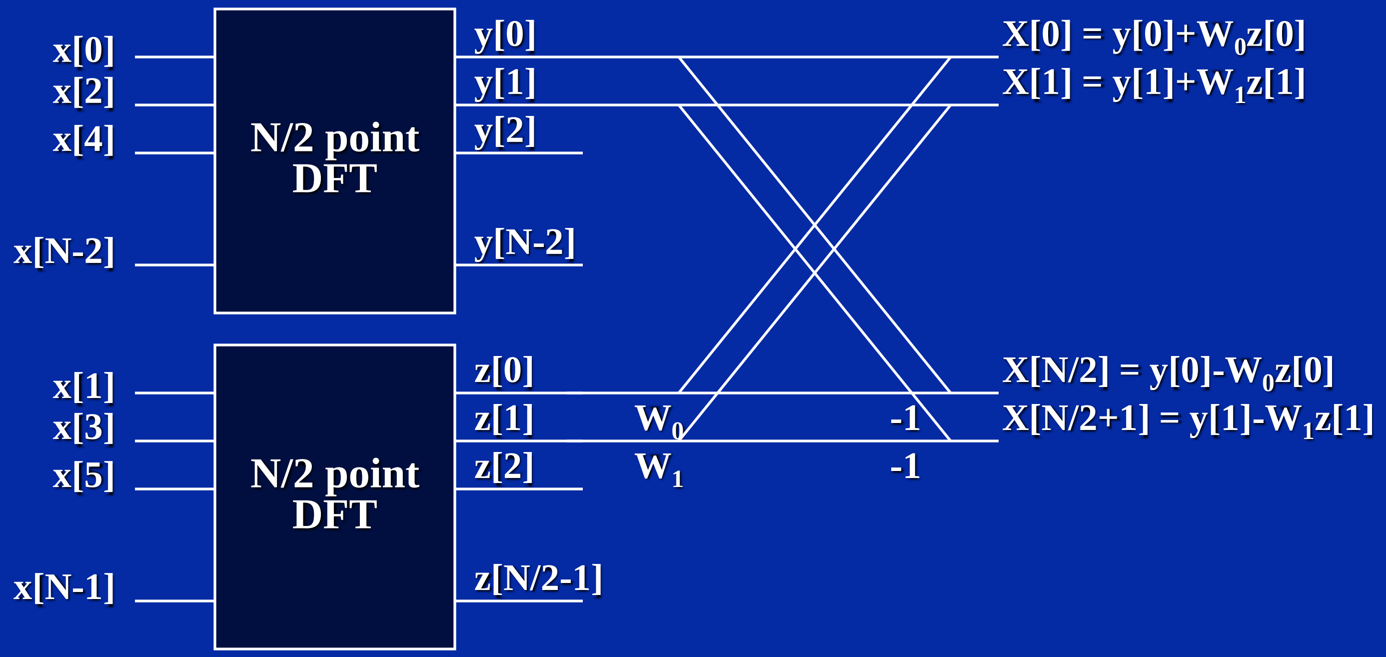
- **Y(k) and Z(k) can also be divided into N/4 point DFTs using the same process shown above:**

$$Y(k) = U(k) + W_{\frac{N}{2}}^k V(k) \qquad Z(k) = P(k) + W_{\frac{N}{2}}^k Q(k)$$

$$Y\left(k + \frac{N}{4}\right) = U(k) - W_{\frac{N}{2}}^k V(k) \qquad Z\left(k + \frac{N}{4}\right) = P(k) - W_{\frac{N}{2}}^k Q(k)$$

- **The process continues until we reach 2 point DFTs.**

DFT → FFT

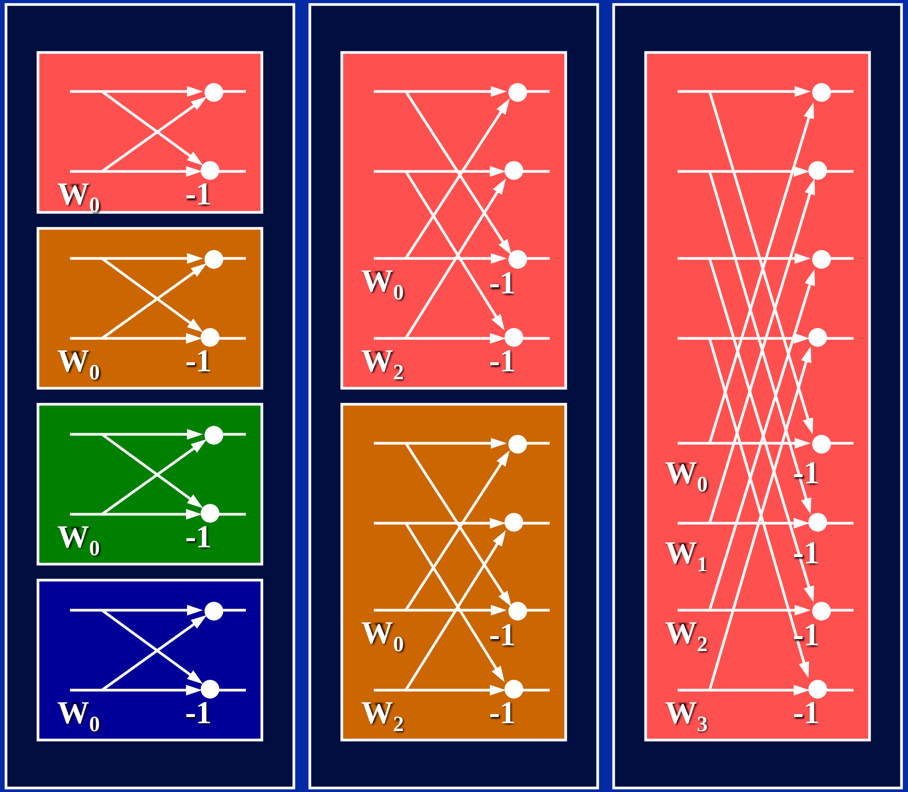


- **Illustration of the first decimation in time FFT.**

FFT Implementation

- To efficiently implement the FFT algorithm a few observations are made:
  - Each stage has the same number of butterflies (number of butterflies = N/2, N is number of points).
  - The number of DFT groups per stage is equal to ($N/2^{stage}$).
  - The difference between the upper and lower leg is equal to $2^{stage-1}$.
  - The number of butterflies in the group is equal to $2^{stage-1}$.

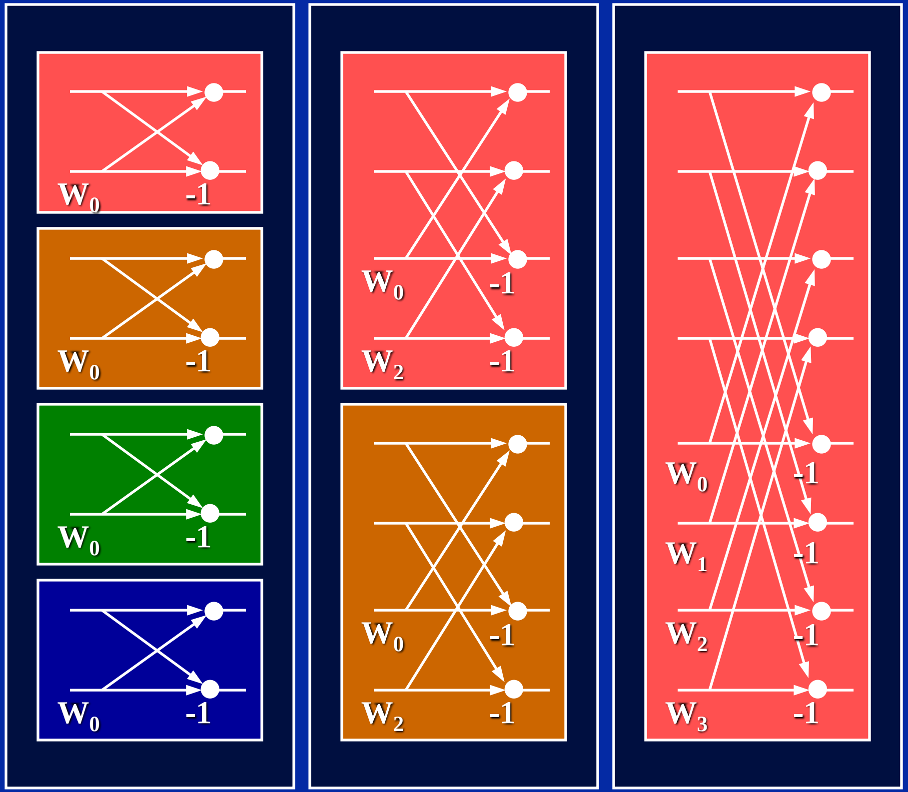
FFT Implementation



Example: 8 point FFT

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation

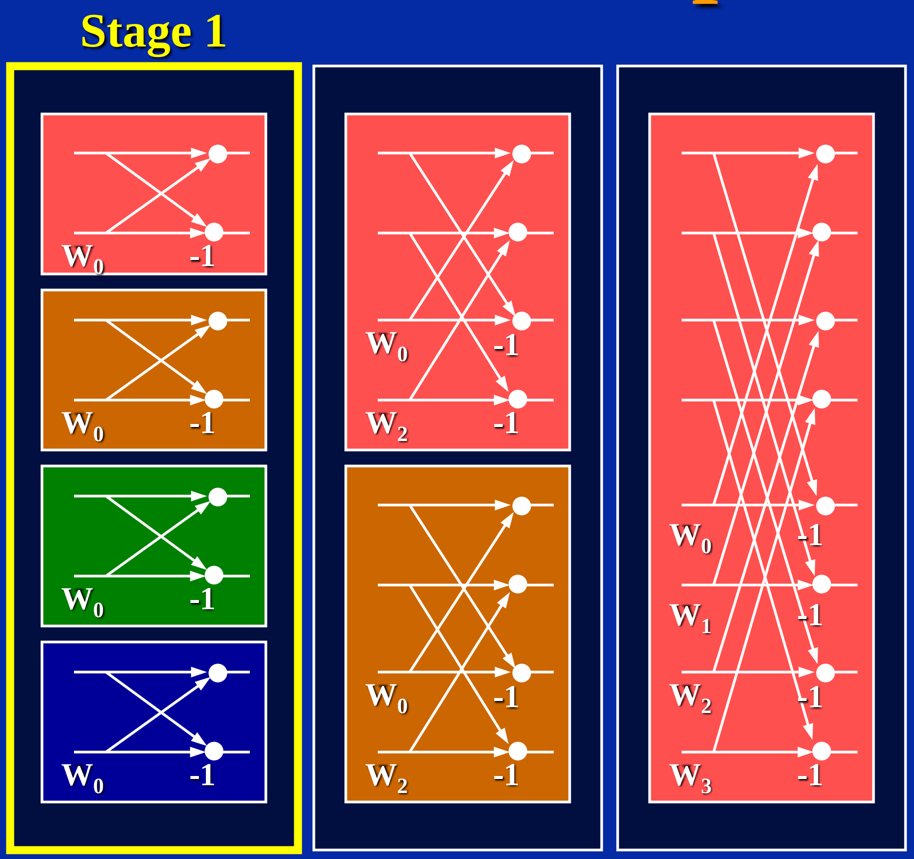


Example: 8 point FFT

(1) Number of stages:

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



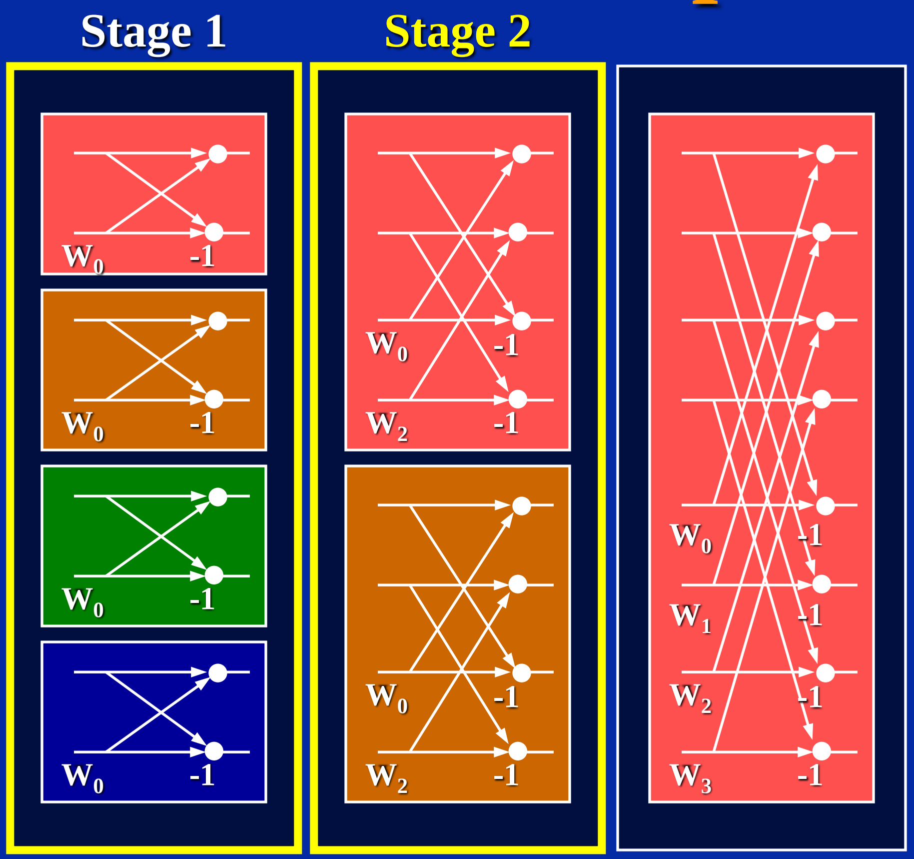
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



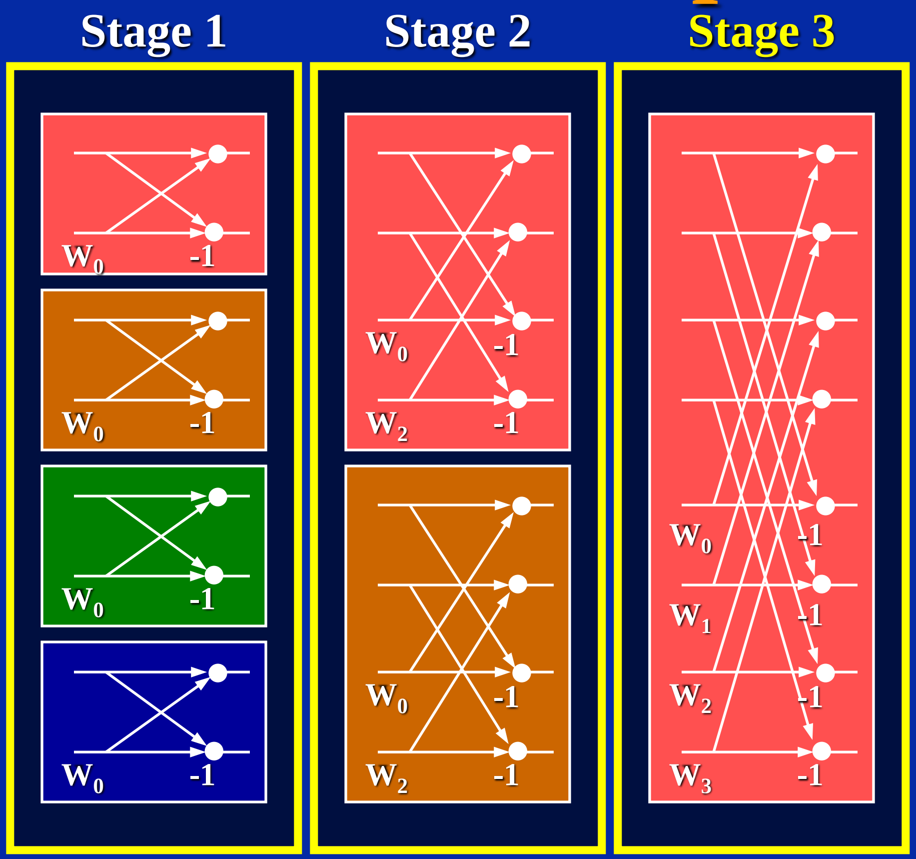
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



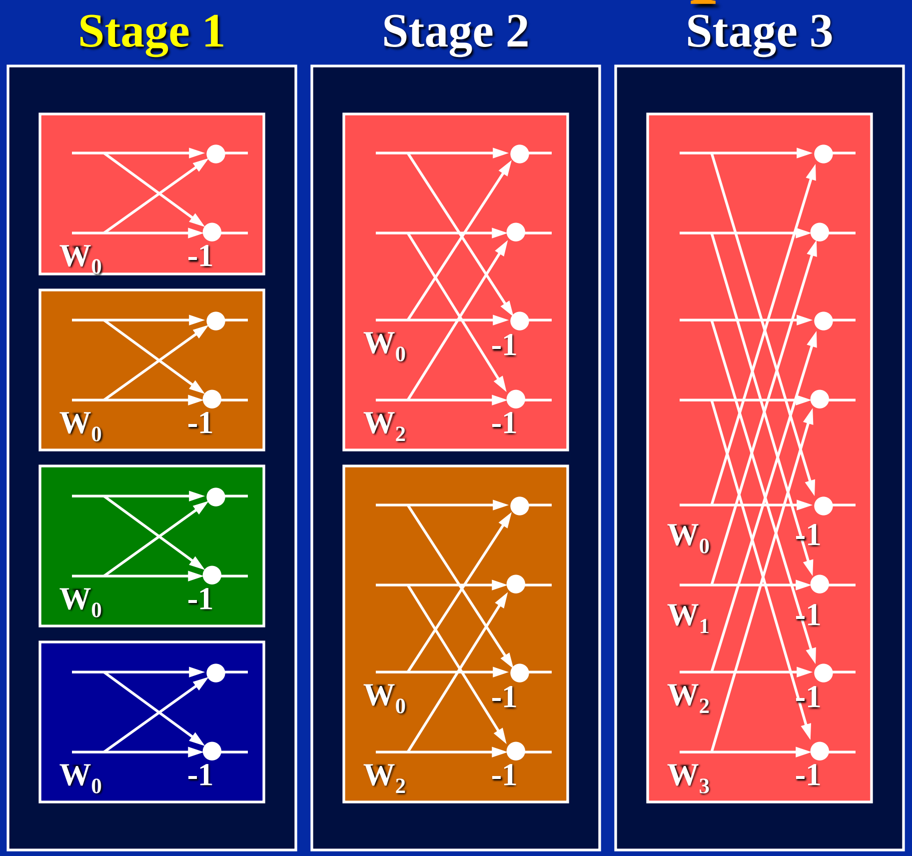
Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

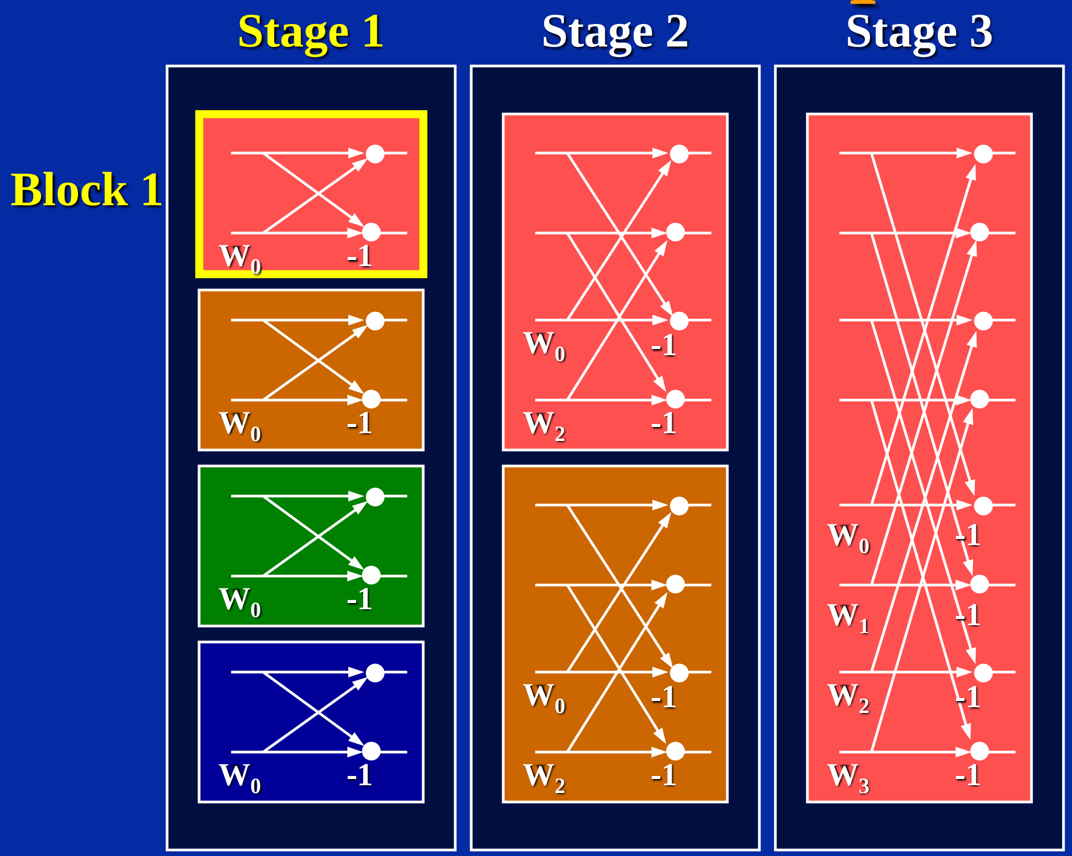
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1:**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

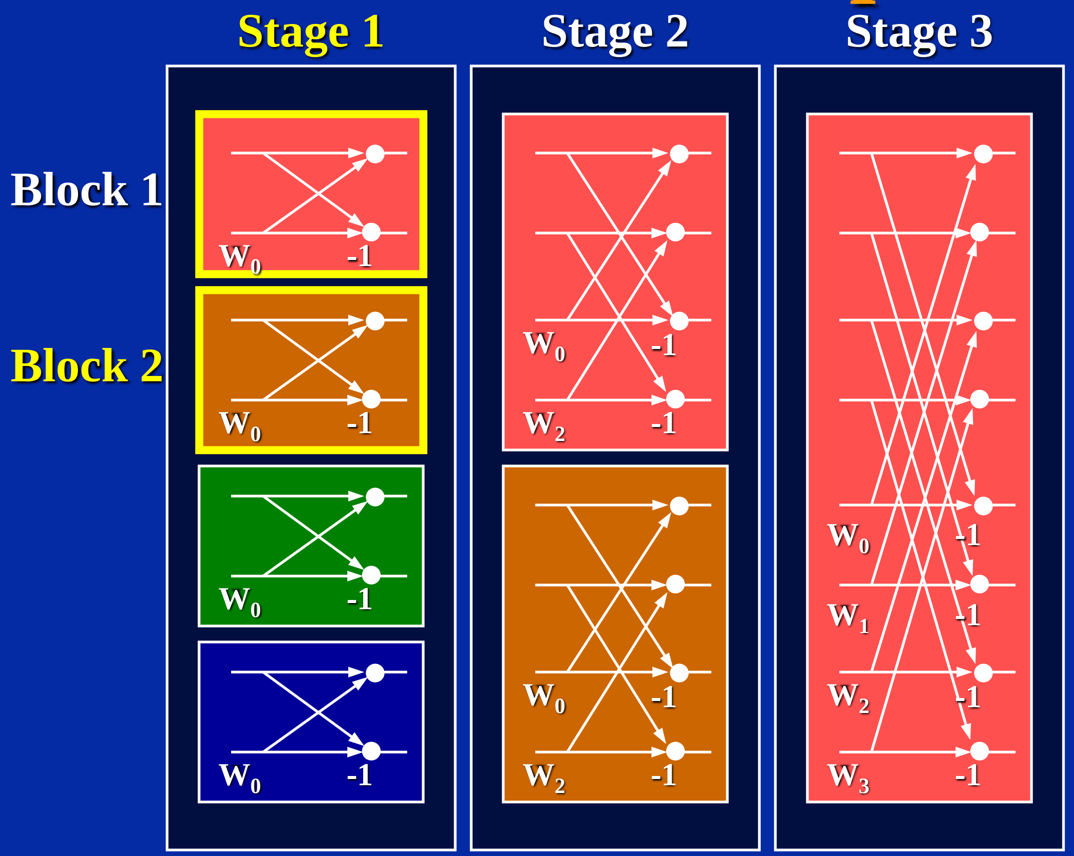
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1: $N_{blocks} = 1$**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

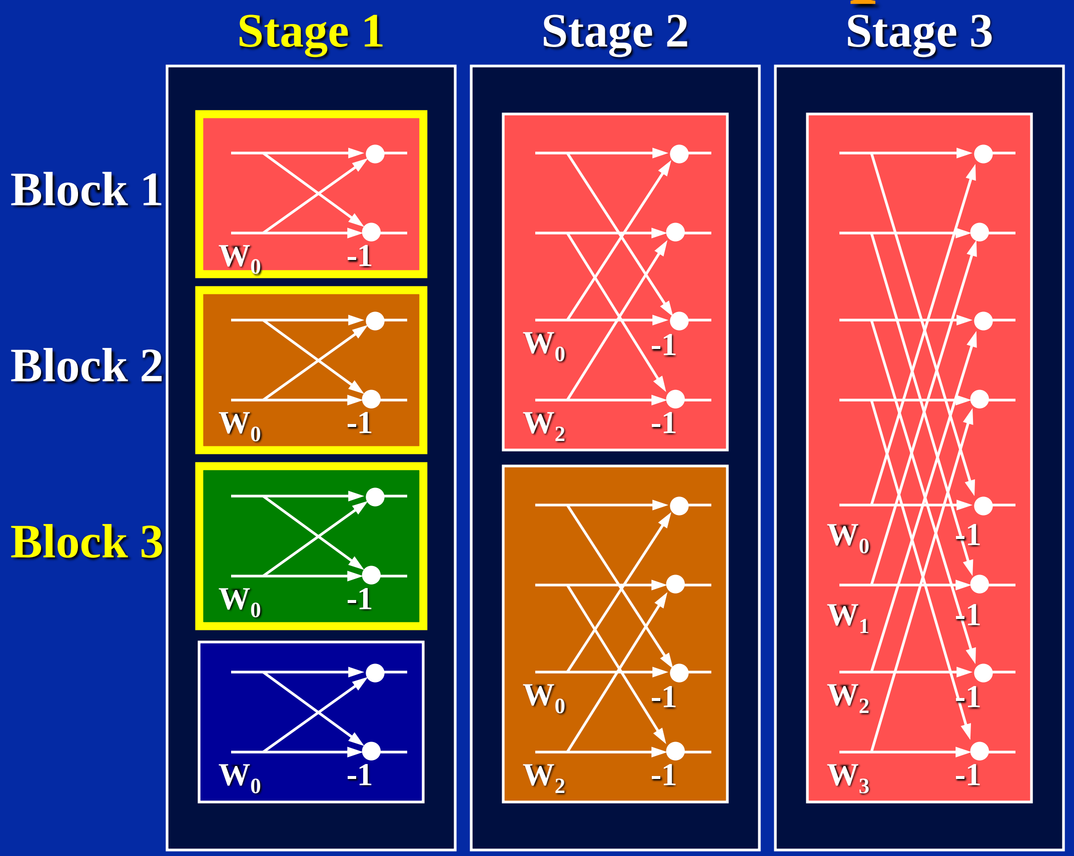
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1: $N_{blocks} = 2$**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

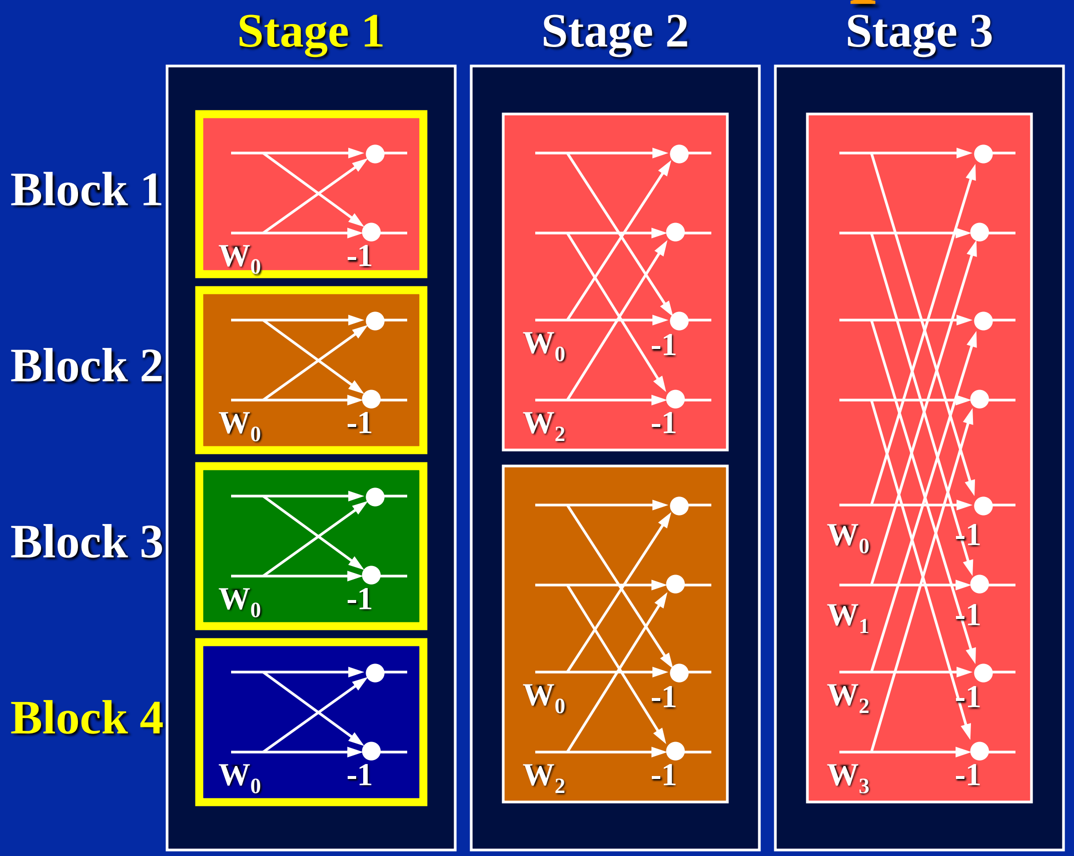
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1: $N_{blocks} = 3$**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

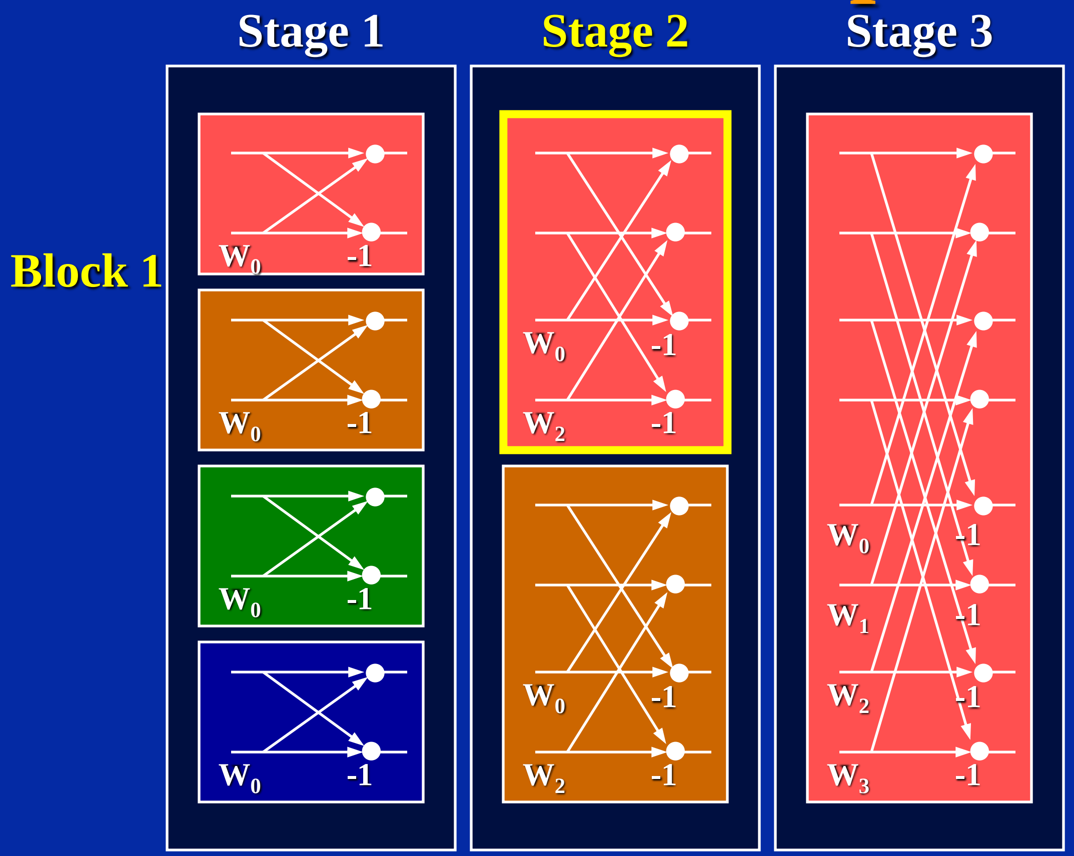
- $N_{stages} = 3$

(2) Blocks/stage:

- **Stage 1: $N_{blocks} = 4$**

- **Decimation in time FFT:**
  - **Number of stages = $\log_2 N$**
  - **Number of blocks/stage = $N/2^{stage}$**
  - **Number of butterflies/block = $2^{stage-1}$**

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

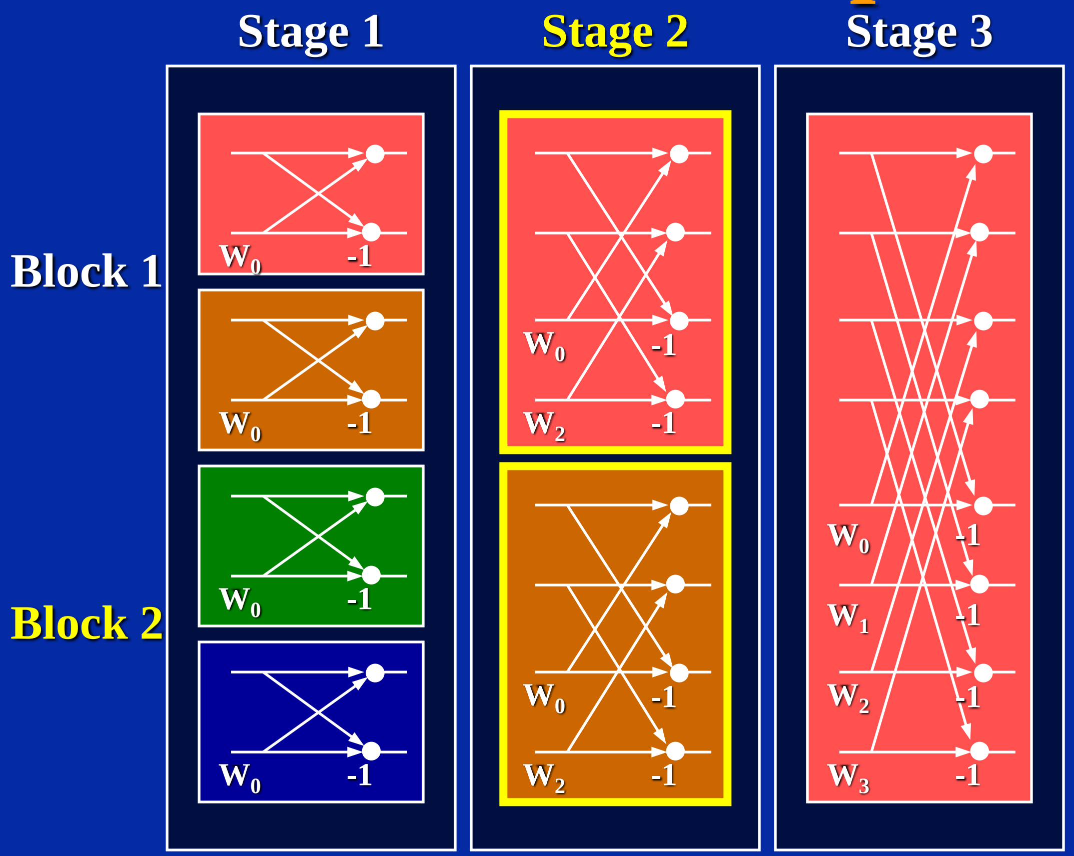
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

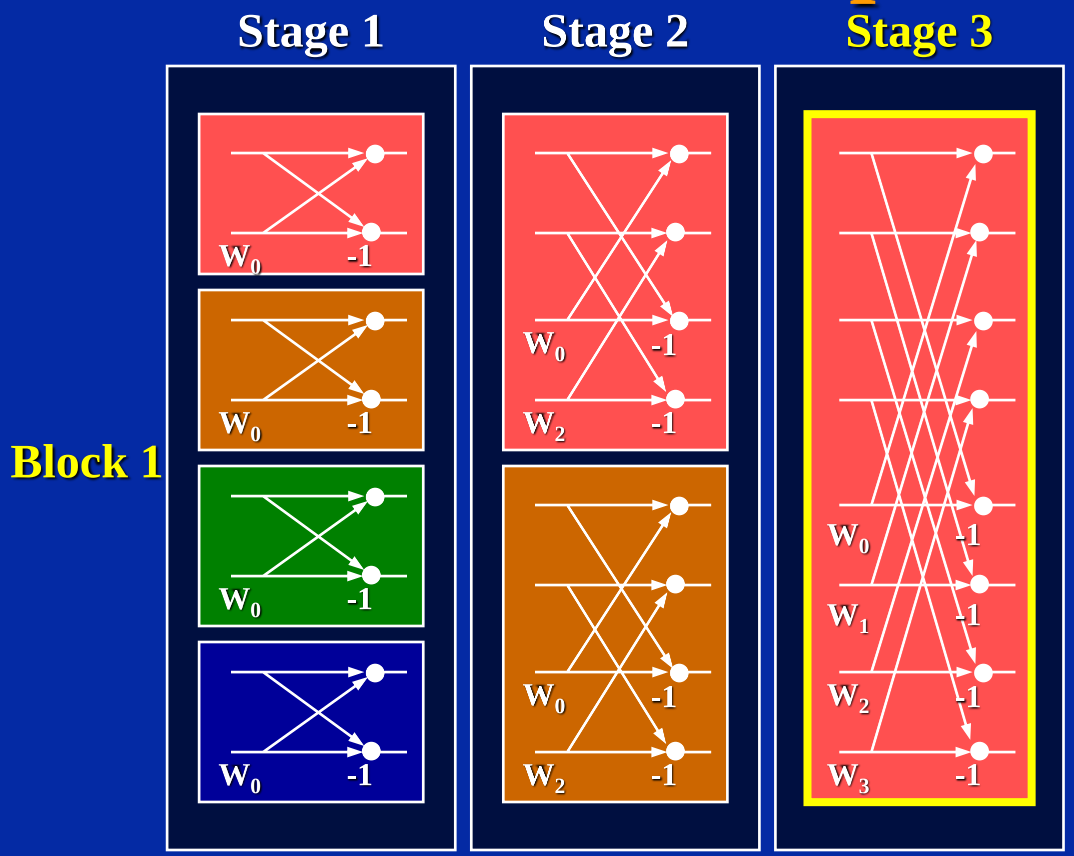
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

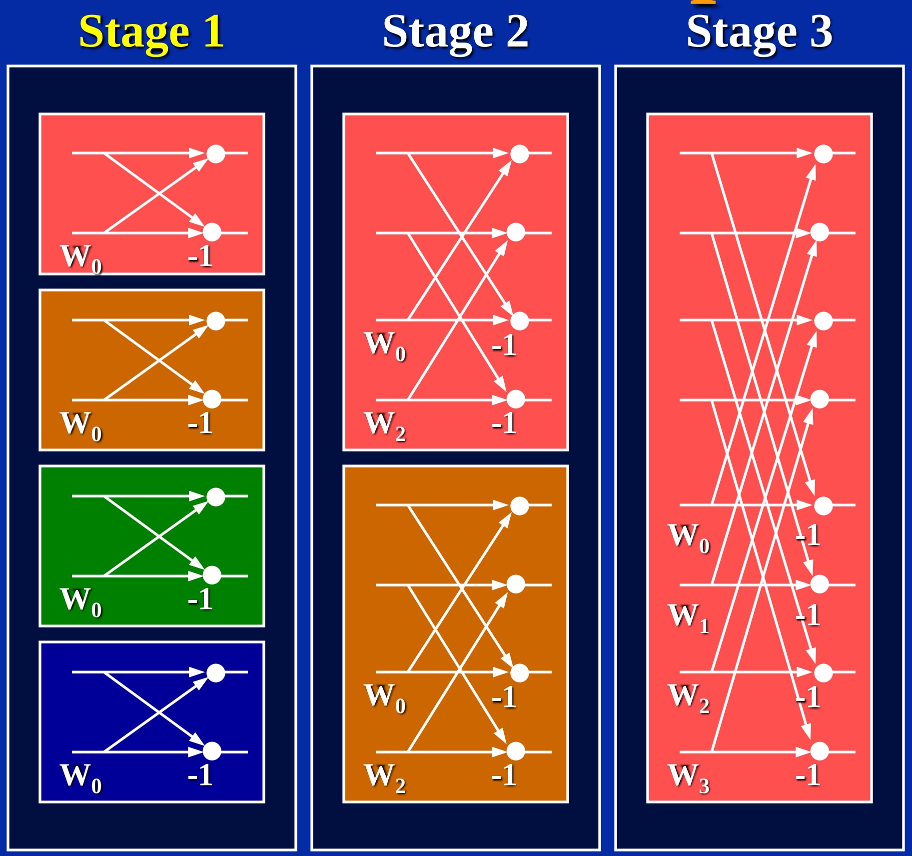
- $N_{stages} = 3$

(2) Blocks/stage:

- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

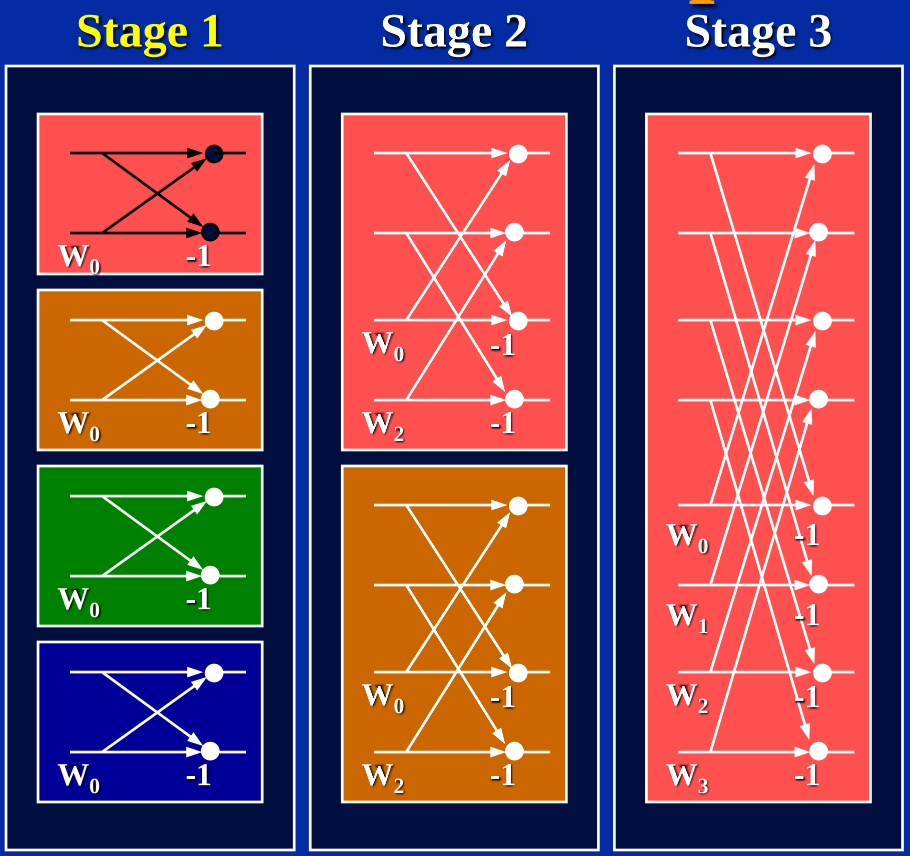
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1:

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

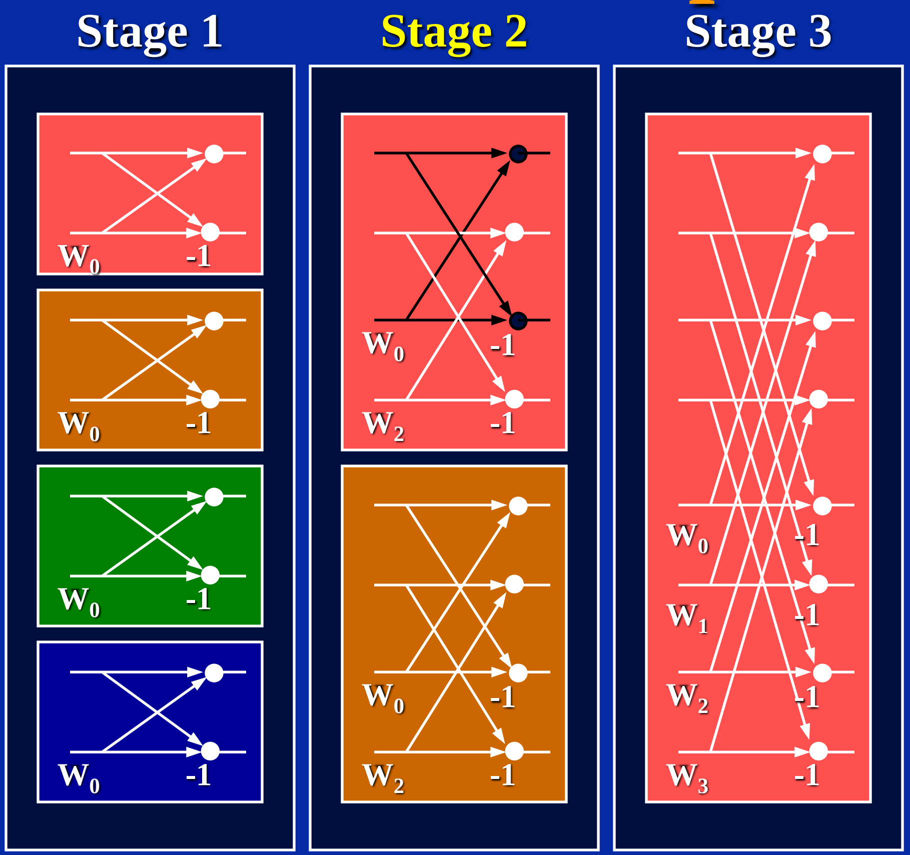
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

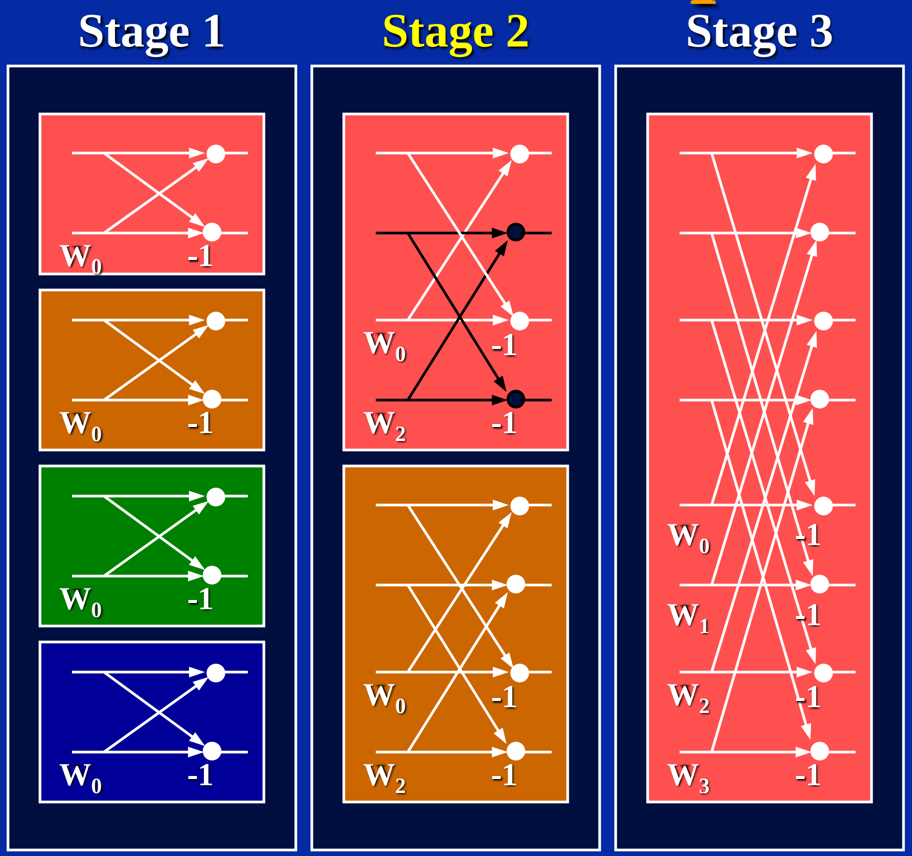
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

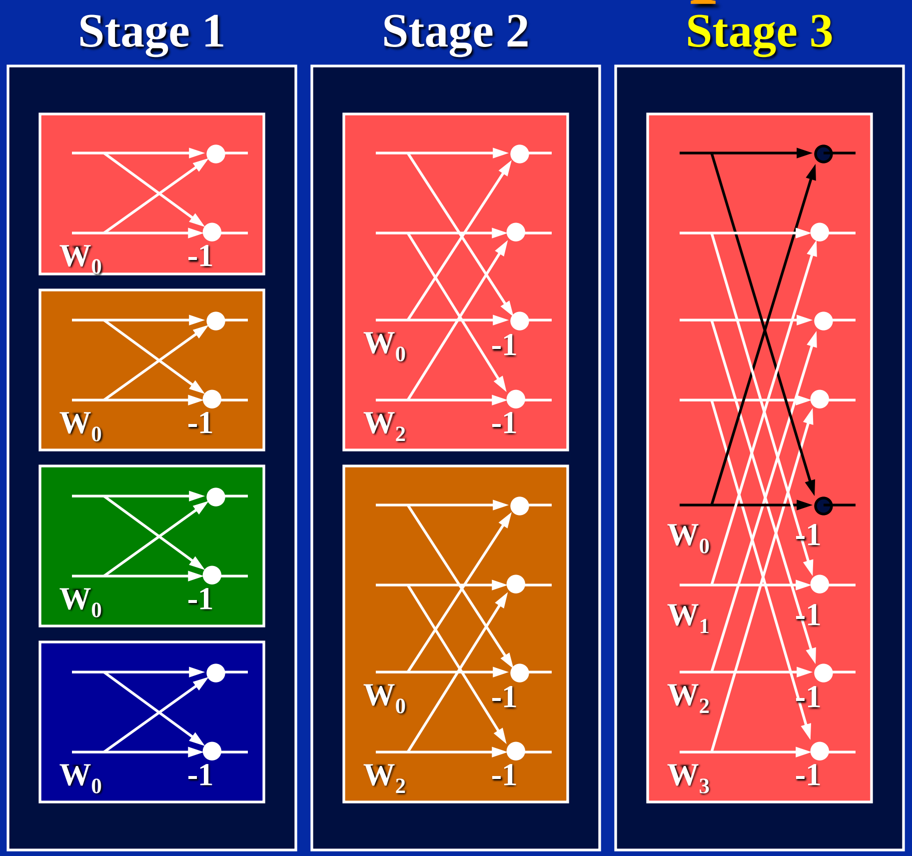
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

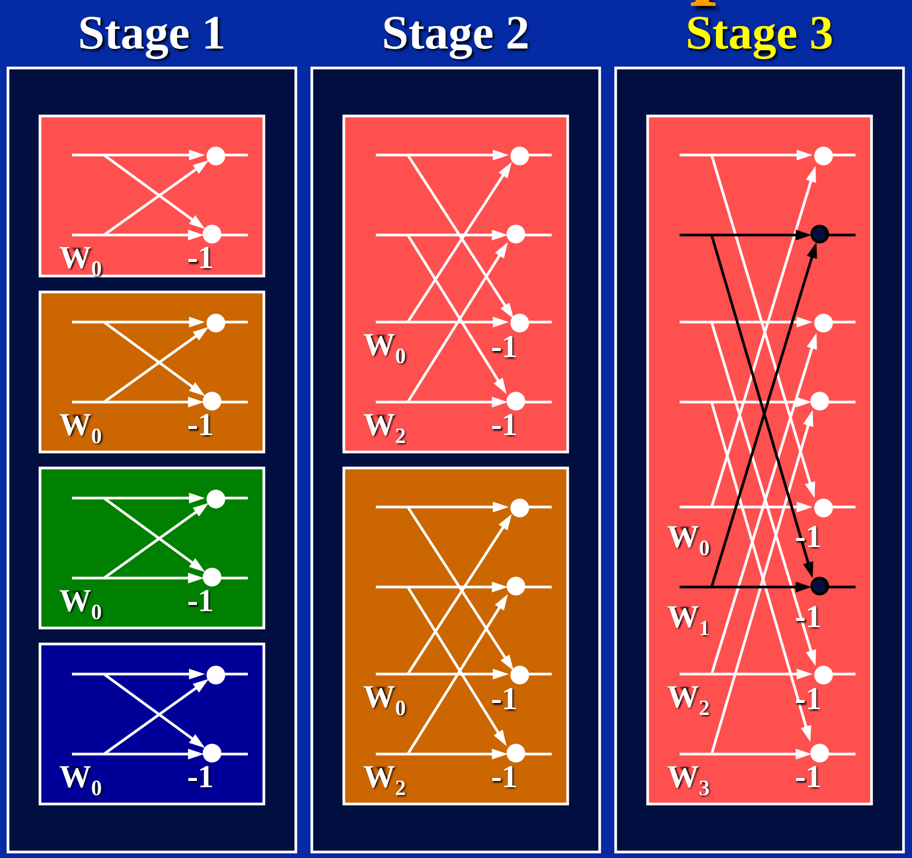
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 1$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

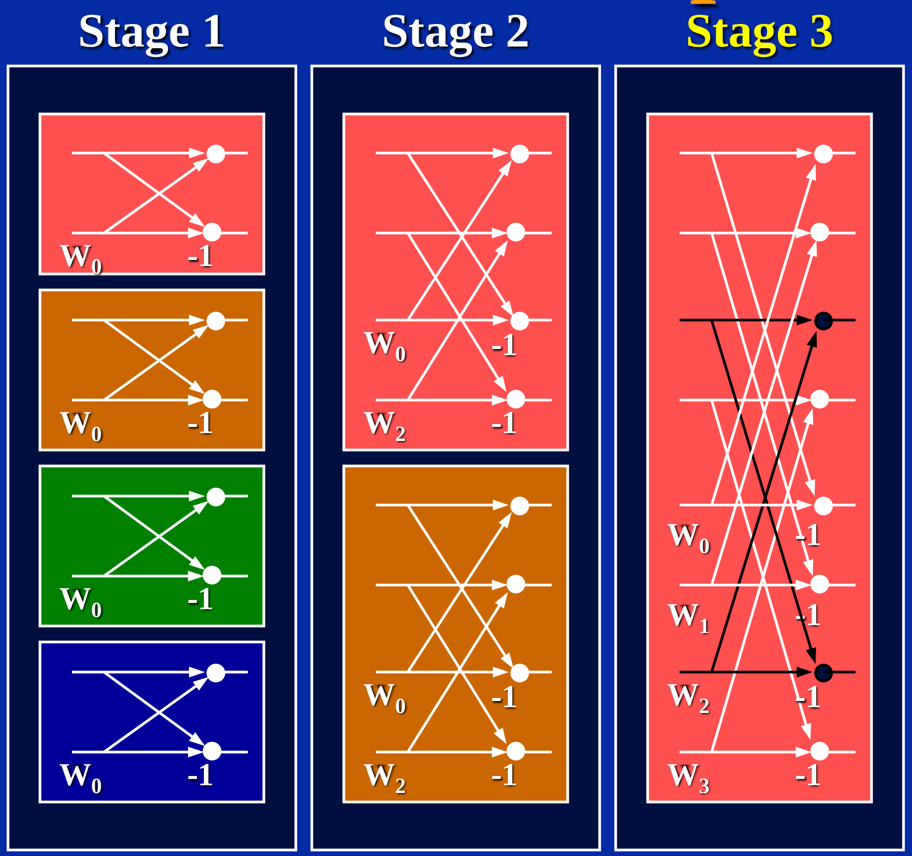
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 2$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

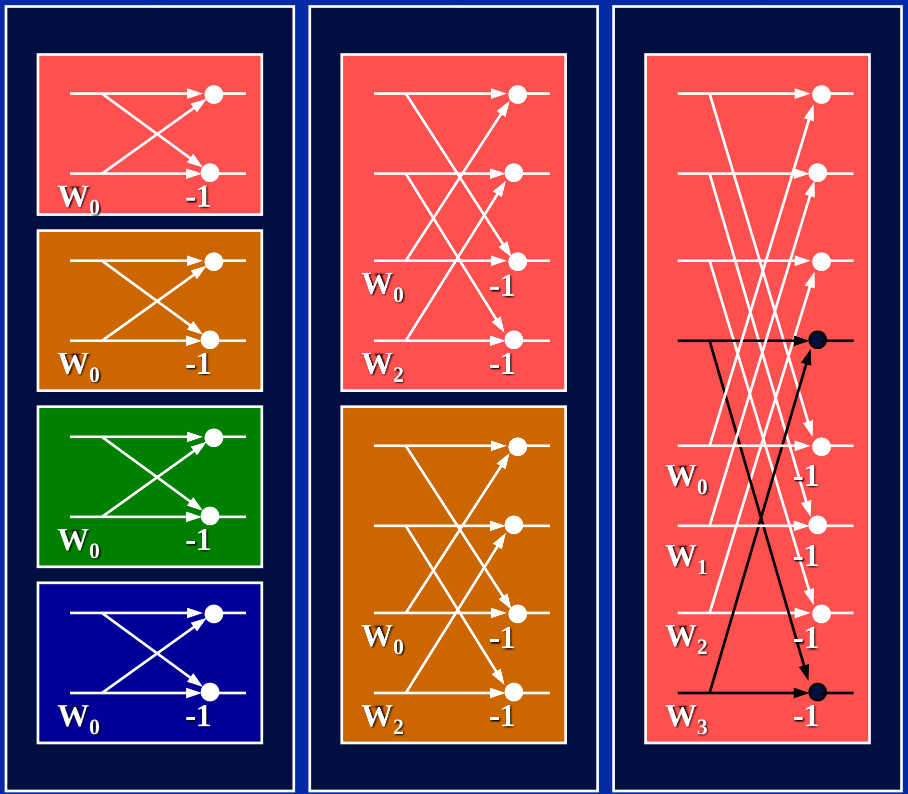
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 3$

- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



Example: 8 point FFT

(1) Number of stages:

- $N_{stages} = 3$

(2) Blocks/stage:

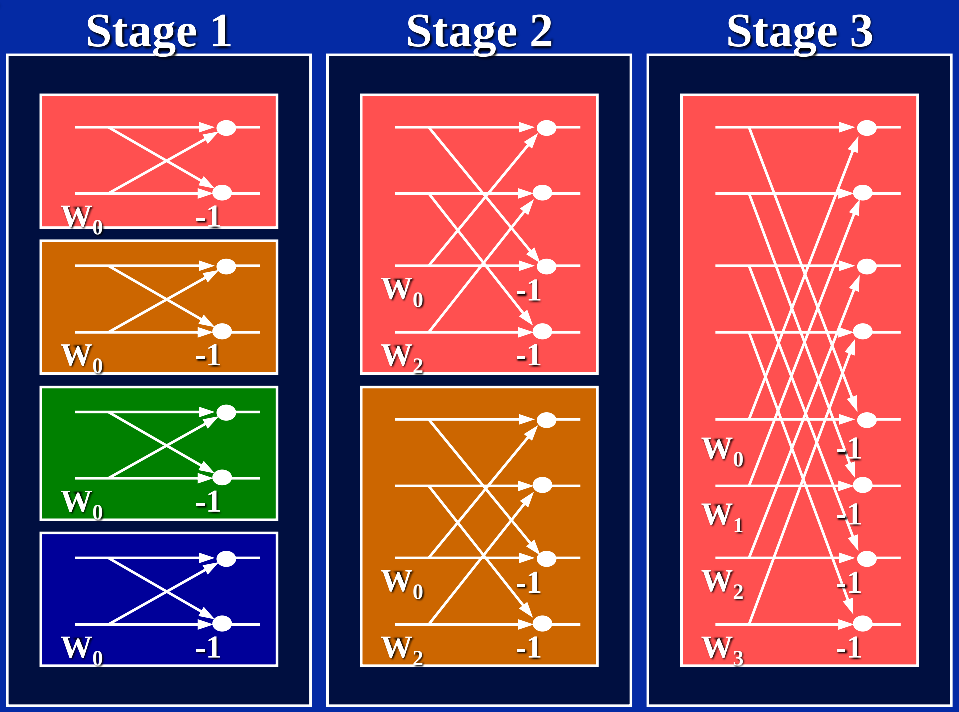
- Stage 1: $N_{blocks} = 4$
- Stage 2: $N_{blocks} = 2$
- Stage 3: $N_{blocks} = 1$

(3) B'flies/block:

- Stage 1: $N_{btf} = 1$
- Stage 2: $N_{btf} = 2$
- Stage 3: $N_{btf} = 4$

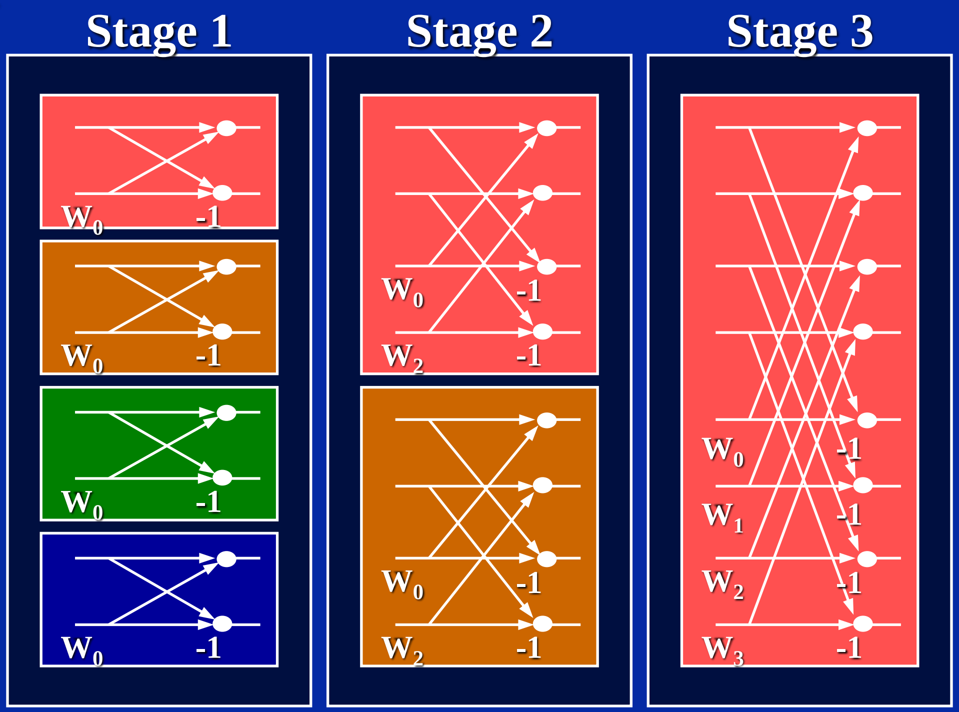
- **Decimation in time FFT:**
  - Number of stages = $\log_2 N$
  - Number of blocks/stage = $N/2^{stage}$
  - Number of butterflies/block = $2^{stage-1}$

FFT Implementation



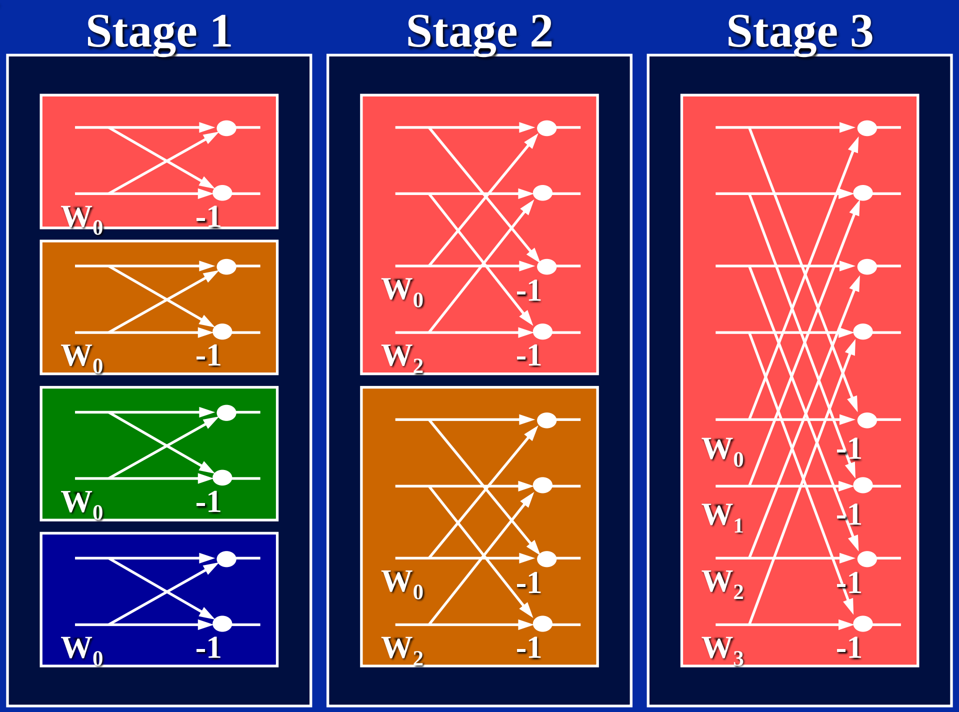
	Stage 1	Stage 2	Stage 3
Start Index	**0**	**0**	**0**
Input Index	**1**	**2**	**4**
Twiddle Factor Index	**N/2 = 4**		

FFT Implementation



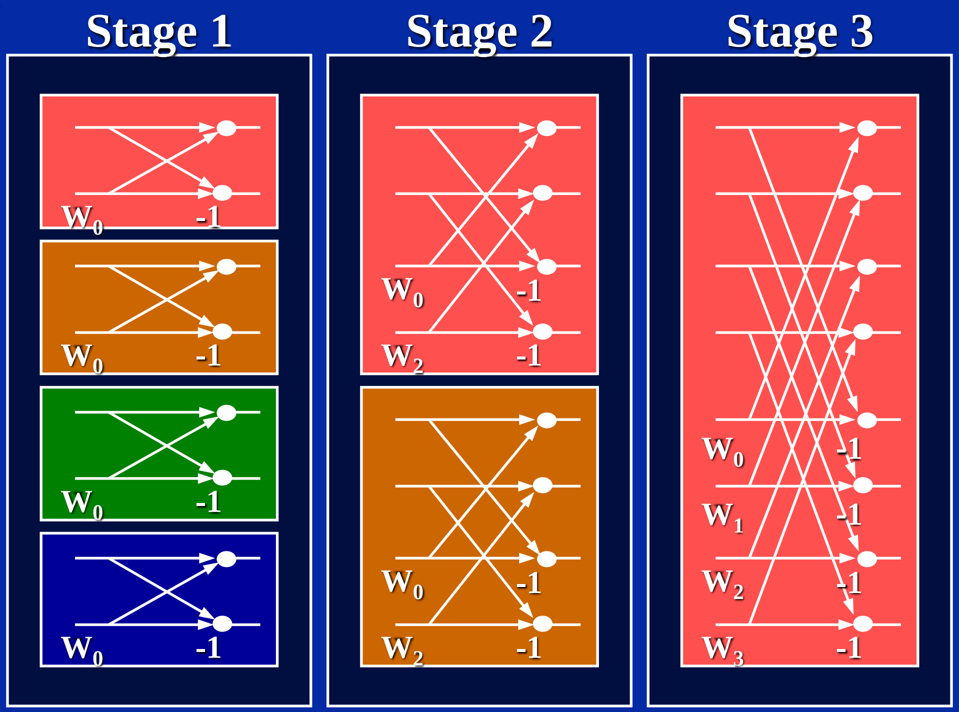
	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4	4/2 = 2	

FFT Implementation



	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4	4/2 = 2	2/2 = 1

FFT Implementation



	Stage 1	Stage 2	Stage 3
Start Index	0	0	0
Input Index	1	2	4
Twiddle Factor Index	N/2 = 4	4/2 = 2	2/2 = 1
Indicies Used	W_0	W_0, W_2	W_0, W_1, W_2, W_3

FFT DECIMATION IN FREQUENCY

- Similar divide and conquer strategy
  - Decimate in frequency domain
- $X(2k) = \sum_{n=0}^{N-1} x(n) W_N^{2nk}$
- $X(2k) = \sum_{n=0}^{N/2-1} x(n) W_{N/2}^{nk} + \sum_{n=N/2}^{N-1} x(n) W_{N/2}^{nk}$
  - Divide into first half and second half of sequence
- $X(2k) = \sum_{n=0}^{N/2-1} x(n) W_{N/2}^{nk} + \sum_{n=0}^{N/2-1} x\left(n+\frac{N}{2}\right) W_{N/2}^{\left(n+\frac{N}{2}\right)k}$
- Simplifying with twiddle properties
  - $X(2k) = \sum_{n=0}^{N/2-1} \left[x(n) + x\left(n+\frac{N}{2}\right)\right] W_{N/2}^{nk}$
  - $X(2k+1) = \sum_{n=0}^{N/2-1} W_N^{n} \left[x(n) - x\left(n+\frac{N}{2}\right)\right] W_{N/2}^{nk}$

FFT DECIMATION IN FREQUENCY STRUCTURE

- Stage structure

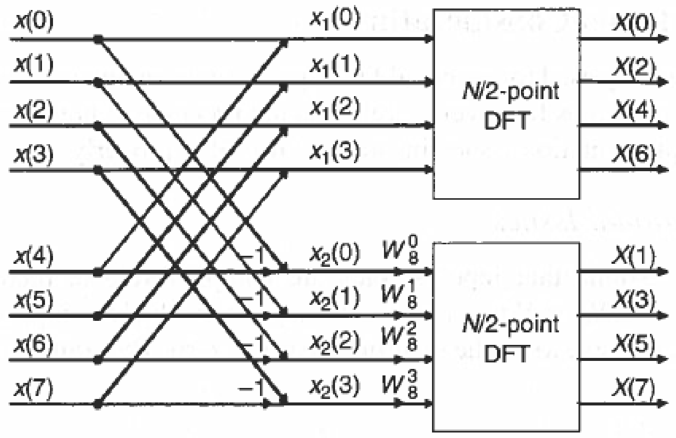


Figure 5.8 Decomposition of an N-point DFT into two N/2-point DFTs

- Full structure

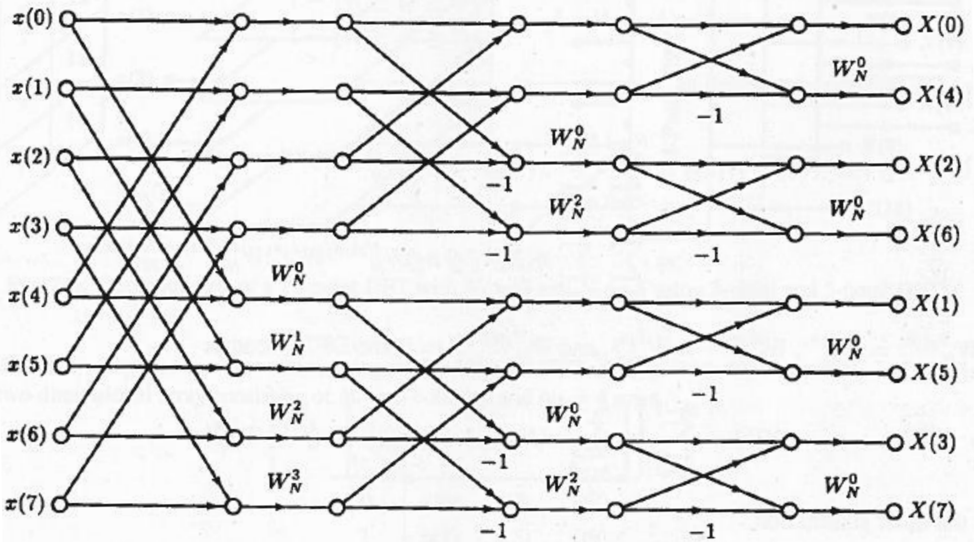


Fig. 7-8. Eight-point radix-2 decimation-in-frequency FFT.

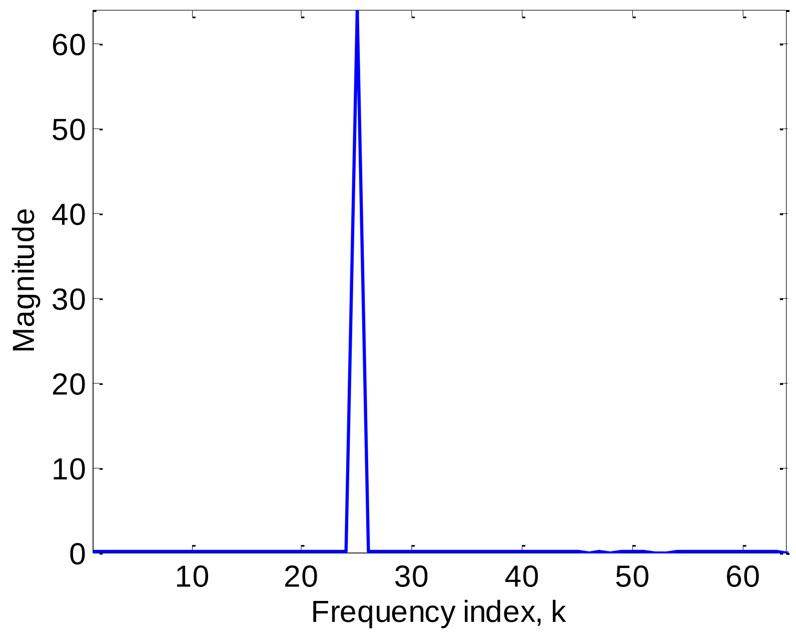
- Bit reversal happens at output instead of input

INVERSE FFT

- $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, W_N^{-kn}$
- Notice this is the DFT with a scale factor and change in twiddle sign
- Can compute using the FFT with minor modifications
  - $x^*(n) = \frac{1}{N}\sum_{k=0}^{N-1} X^*(k)\, W_N^{kn}$
    - Conjugate coefficients, compute FFT with scale factor, conjugate result
    - For real signals, no final conjugate needed
  - Can complex conjugate twiddle factors and use in butterfly structure

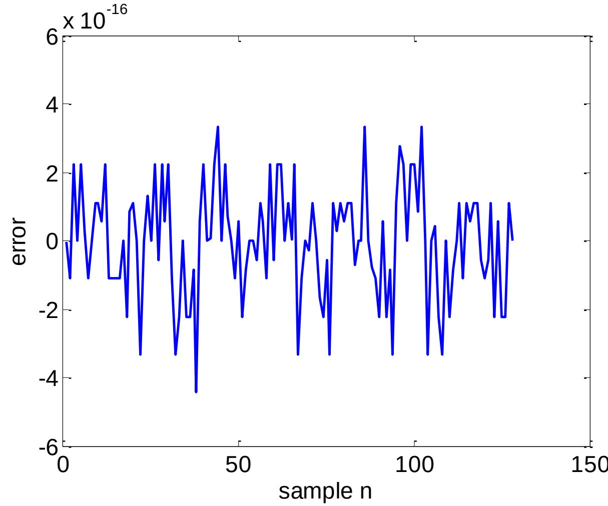
FFT EXAMPLE

- Example 5.10
- Sine wave with $f = 50$ Hz
  - $x(n) = \sin\left(\frac{2\pi f n}{f_s}\right)$
    - $n = 0,1,\ldots,127$
    - $f_s = 256$ Hz



- Frequency resolution of DFT?
  - $\Delta = f_s/N = \frac{256}{128} = 2$ Hz
- Location of peak
  - $50 = k\Delta \rightarrow k = \frac{50}{2} = 25$

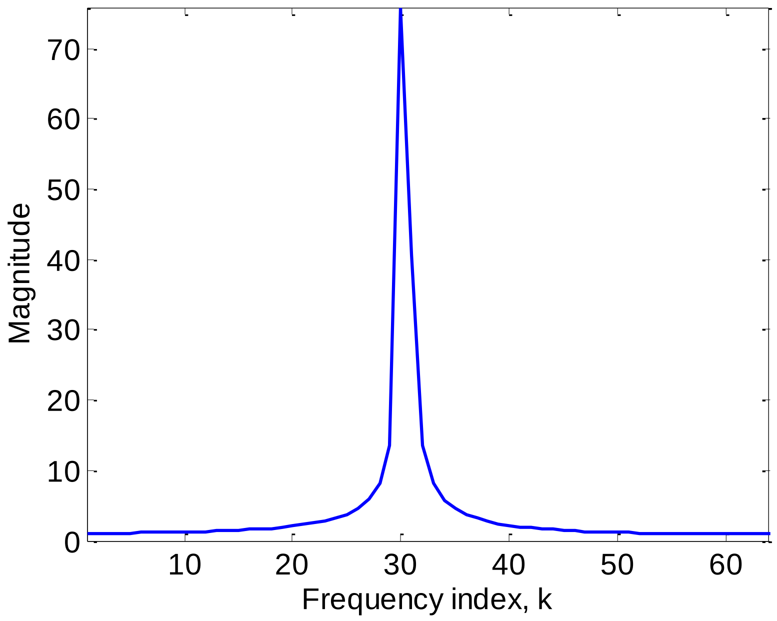
Reconstruction error



SPECTRAL LEAKAGE AND RESOLUTION

- Notice that a DFT is like windowing a signal to finite length
  - Longer window lengths (more samples) the closer DFT $X(k)$ approximates DTFT $X(\omega)$
- Convolution relationship
  - $x_N(n) = w(n)x(n)$
  - $X_N(k) = W(k) * X(k)$
- Corruption of spectrum due to window properties (mainlobe/sidelobe)
  - Sidelobes result in spurious peaks in computed spectrum known as spectral leakage
    - Obviously, want to use smoother windows to minimize these effects
  - Spectral smearing is the loss in sharpness due to convolution which depends on mainlobe width

- Example 5.15
  - Two close sinusoids smeared together

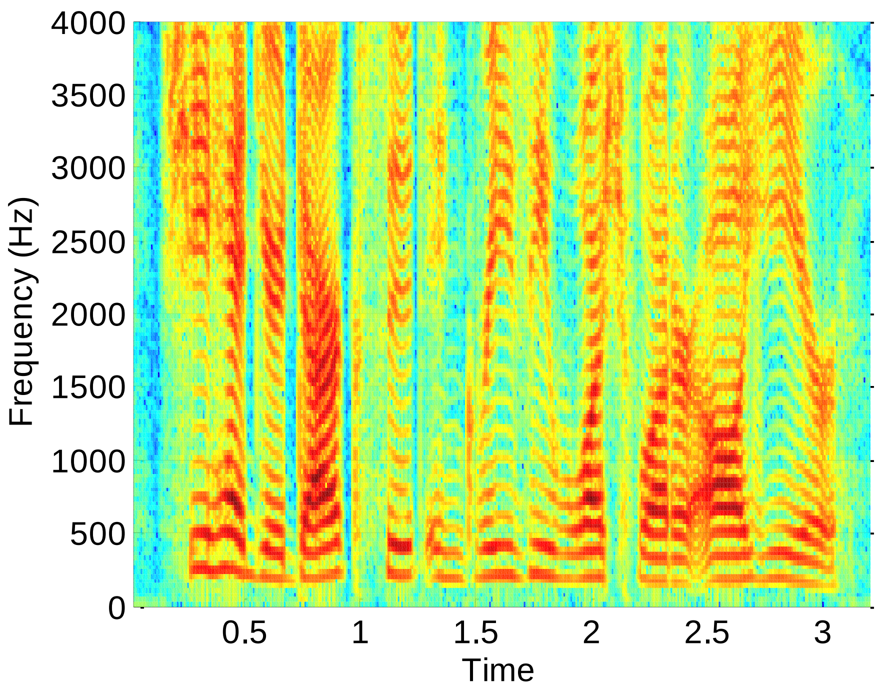


- To avoid smearing:
  - Frequency separation should be greater than freq resolution
  - $N > \frac{2\pi}{\Delta\omega}$, $N > f_s/\Delta f$

POWER SPECTRAL DENSITY

- Parseval's theorem
- $E = \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2$
  - $|X(k)|^2$ - power spectrum or periodogram
- Power spectral density (PSD, or power density spectrum or power spectrum) is used to measure average power over frequencies
- Computed for time-varying signal by using a sliding window technique
  - Short-time Fourier transform
  - Grab N samples and compute FFT
    - Must have overlap and use windows

- Spectrogram
  - Each short FFT is arranged as a column in a matrix to give the time-varying properties of the signal
  - Viewed as an image



"She had your dark suit in greasy wash water all year"

FAST FFT CONVOLUTION

- Linear convolution is multiplication in frequency domain
  - Must take FFT of signal and filter, multiply, and iFFT
  - Operations in frequency domain can be much faster for large filters
  - Requires zero-padding because of circular convolution
- Typically, will do block processing
  - Segment a signal and process each segment individually before recombining