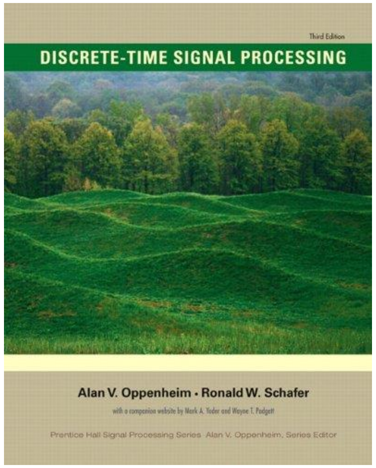


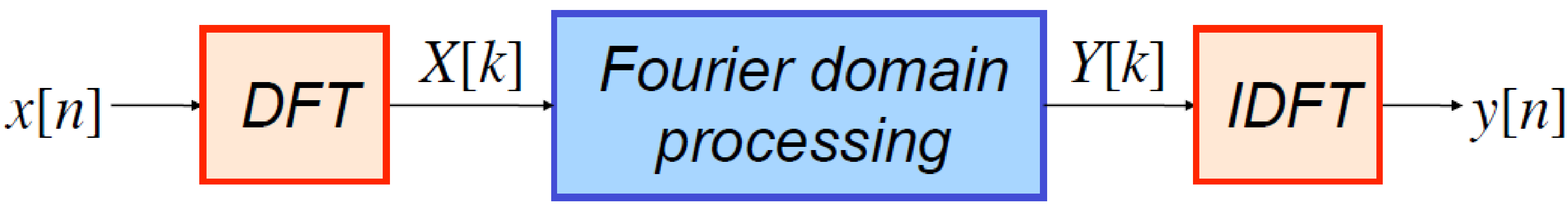
Lecture 15:
The Fast Fourier Transform

Outlines

- Calculation of the DFT
- The Fast Fourier Transform algorithm

1. Calculation of the DFT

- Filter design so far has been oriented to time-domain processing - cheaper!
- But: frequency-domain processing makes some problems very simple:

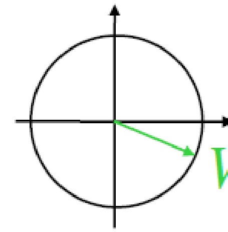


  - use all of $x[n]$, or use short-time windows
- Need an efficient way to calculate DFT

The DFT

- Recall the DFT:

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



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

W_N@$2\pi/N$

$\Rightarrow W_N^{\,r}$ *has only* *N distinct values*

  - discrete transform of discrete sequence

- Matrix form:

Structure ⇒ opportunities for efficiency

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

Computational Complexity

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

- N complex multiplies
  + N-1 complex adds per point (k)
  $\times$ N points ($k = 0..\ N$-1)
  - cpx mult: $(a+jb)(c+jd) = ac - bd + j(ad + bc)$
    = 4 real mults + 2 real adds
  - cpx add = 2 real adds
- N points: $4N^2$ real mults, $4N^2$-$2N$ real adds

Goertzel's Algorithm

- Now: $X[k] = \sum_{\ell=0}^{N-1} x[\ell] W_N^{k\ell}$

$$= W_N^{kN} \sum_{\ell} x[\ell] W_N^{-k(N-\ell)}$$

looks like a convolution

- i.e. $X[k] = y_k[N]$

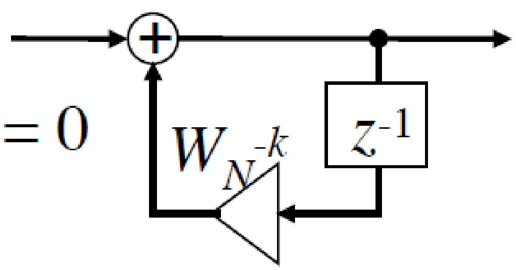
  where $y_k[n] = x_e[n] \circledast h_k[n]$

$$x_e[n] = \begin{cases} x[n] & 0 \le n < N \\ 0 & n = N \end{cases}$$

$$h_k[n] = \begin{cases} W_N^{-kn} & n \ge 0 \\ 0 & n < 0 \end{cases}$$

$x_e[n]$

$x_e[N] = 0$



$y_k[n]$

$y_k[-1] = 0$

$y_k[N] = X[k]$

Goertzel's Algorithm

- Separate 'filters' for each $X[k]$
  - can calculate for just a few values of k
- No large buffer, no coefficient table
- Same complexity for full $X[k]$ ($4N^2$ mults, $4N^2 - 2N$ adds)
  - but: can halve multiplies by making the denominator real:

$$H(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^{k} z^{-1}}{1 - 2\cos\frac{2\pi k}{N} z^{-1} + z^{-2}}$$

evaluate only for last step

2 real mults per step

2. Fast Fourier Transform FFT

- Reduce complexity of DFT from $O(N^2)$ to $O(N \cdot \log N)$
  - grows more slowly with larger N
- Works by decomposing large DFT into several stages of smaller DFTs
- Often provided as a highly optimized library

Decimation in Time (DIT) FFT

- Can rearrange DFT formula in 2 halves:

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

$k = 0..\ N\text{-}1$

Arrange terms in pairs...

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

Group terms from each pair

$$= \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m] \cdot W_{\frac{N}{2}}^{mk}}_{X_0[<k>_{N/2}]} + W_N^k \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m+1] \cdot W_{\frac{N}{2}}^{mk}}_{X_1[<k>_{N/2}]}$$

*N/2 pt DFT of x for **even** n* *N/2 pt DFT of x for **odd** n*

Decimation in Time (DIT) FFT

$x[n]$ **for even** n $\qquad$ $x[n]$ **for odd** n

$$\mathrm{DFT}_N\{x[n]\} = \mathrm{DFT}_{\frac{N}{2}}\{x_0[n]\} + W_N^k\,\mathrm{DFT}_{\frac{N}{2}}\{x_1[n]\}$$

- We can evaluate an N-pt DFT as two $N/2$-pt DFTs (plus a few mults/adds)
- But if $\mathrm{DFT}_N\{\bullet\} \sim O(N^2)$

  then $\mathrm{DFT}_{N/2}\{\bullet\} \sim O((N/2)^2) = 1/4\ O(N^2)$

$\Rightarrow$ Total computation $\sim 2 \cdot 1/4\ O(N^2)$

$=$ 1/2 the computation $(+\varepsilon)$ of direct DFT

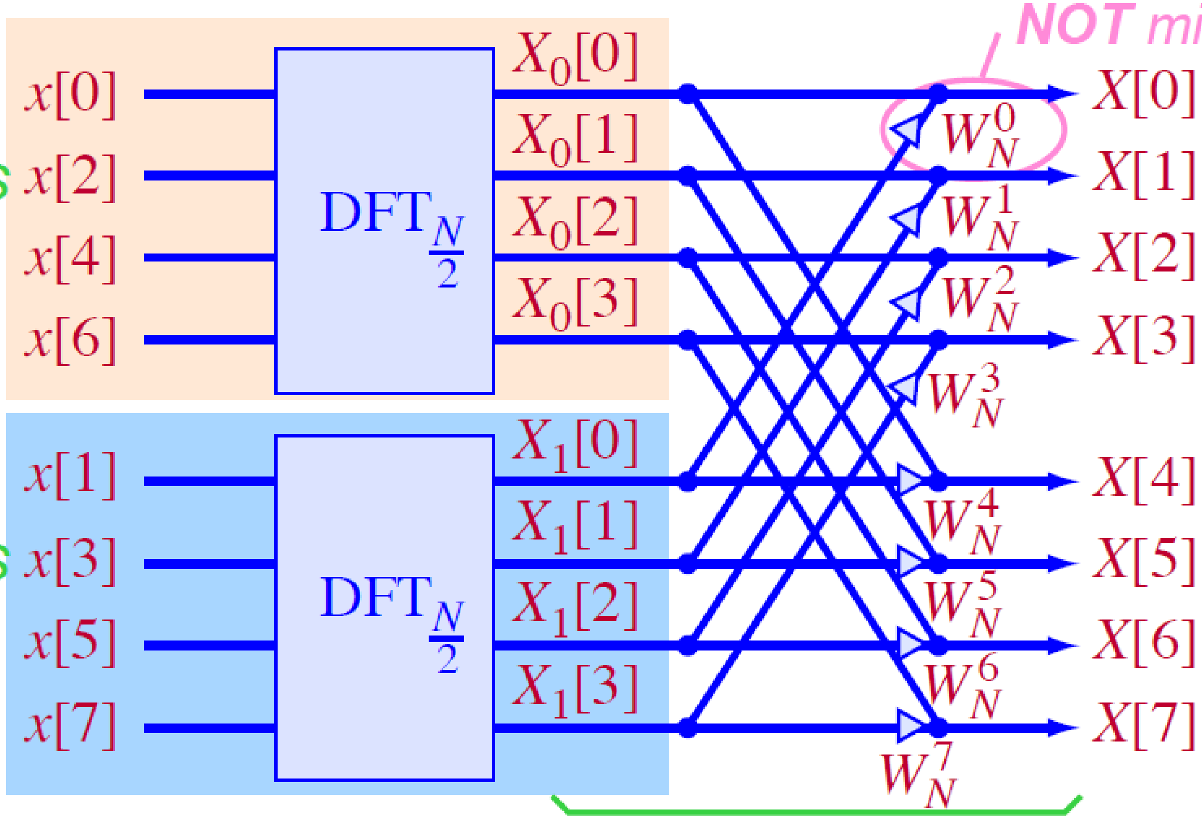
One-Stage DIT Flowgraph

$$X[k] = X_0\left[\langle k\rangle_{\frac{N}{2}}\right] + W_N^k X_1\left[\langle k\rangle_{\frac{N}{2}}\right]$$

*"**twiddle factors**": always apply to odd-terms output **NOT** mirror-image*

Even points from $x[n]$

Odd points from $x[n]$



Same as $X[0..3]$ except for factors on $X_1[\bullet]$ terms

Classic FFT structure

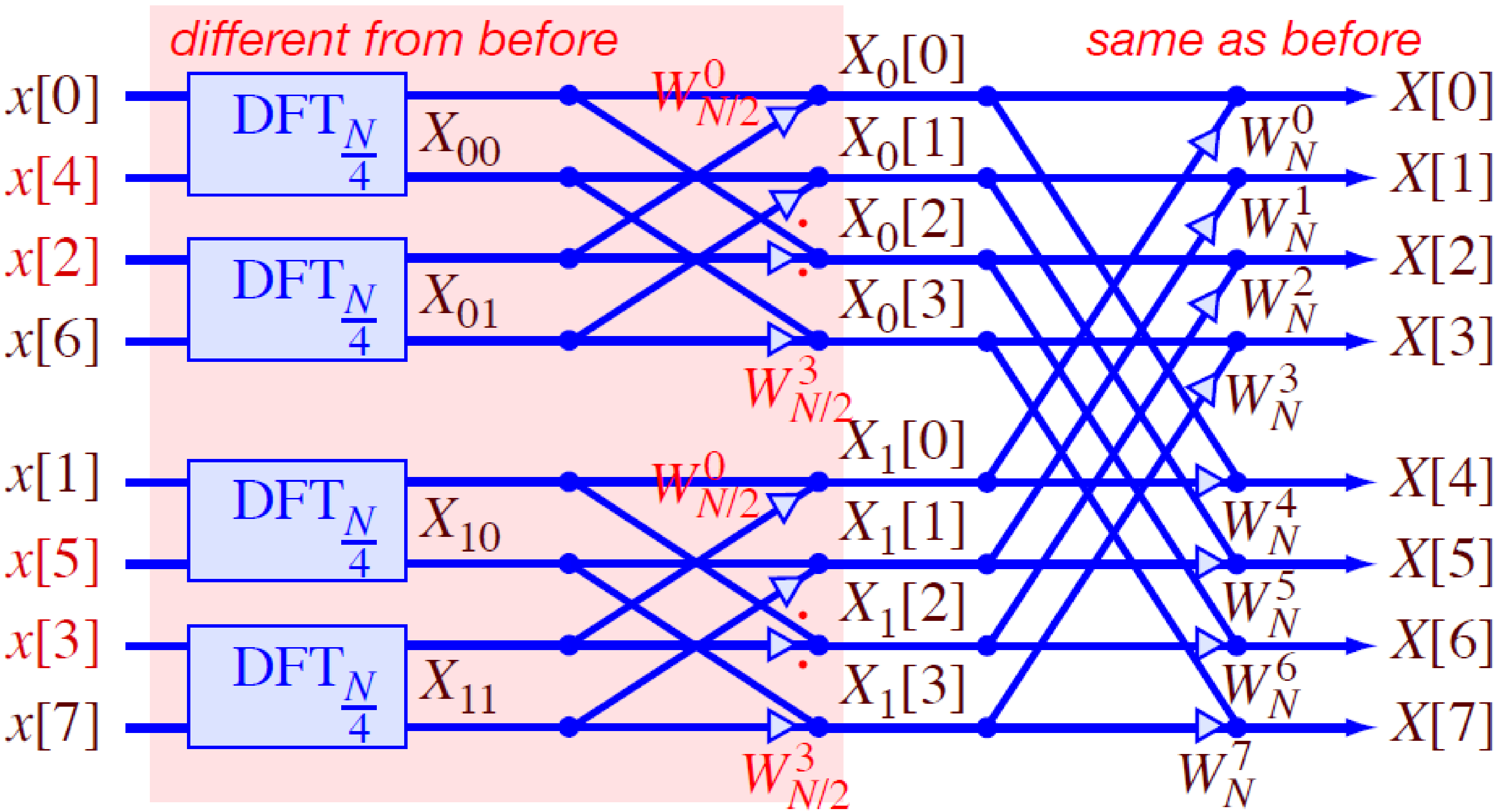
Multiple DIT Stages

- If decomposing one DFT_N into two smaller $\mathrm{DFT}_{N/2}$'s speeds things up ... Why not further divide into $\mathrm{DFT}_{N/4}$'s ?
- i.e. $X[k] = X_0\left[\langle k\rangle_{\frac{N}{2}}\right] + W_N^k X_1\left[\langle k\rangle_{\frac{N}{2}}\right]$
  $0 \le k < N$
- make: $X_0[k] = X_{00}\left[\langle k\rangle_{\frac{N}{4}}\right] + W_{\frac{N}{2}}^k X_{01}\left[\langle k\rangle_{\frac{N}{4}}\right]$
  $0 \le k < N/2$

  *N/4-pt DFT of **even** points in **even** subset of $x[n]$*

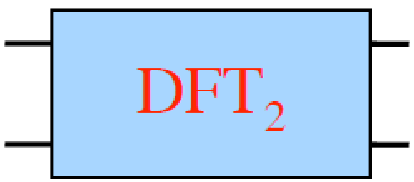
  *N/4-pt DFT of **odd** points from **even** subset*
- Similarly, $X_1[k] = X_{10}\left[\langle k\rangle_{\frac{N}{4}}\right] + W_{\frac{N}{2}}^k X_{11}\left[\langle k\rangle_{\frac{N}{4}}\right]$

Two-Stage DIT Flowgraph



Multi-Stage DIT FFT

- Can keep doing this until we get down to 2-pt DFTs:

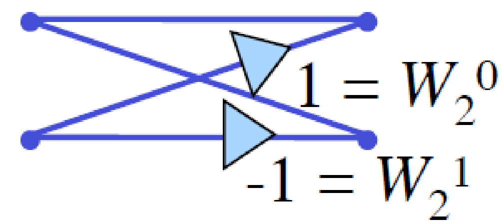


$X[0] = x[0] + x[1]$

$X[1] = x[0] - x[1]$

$\equiv$

"butterfly" element



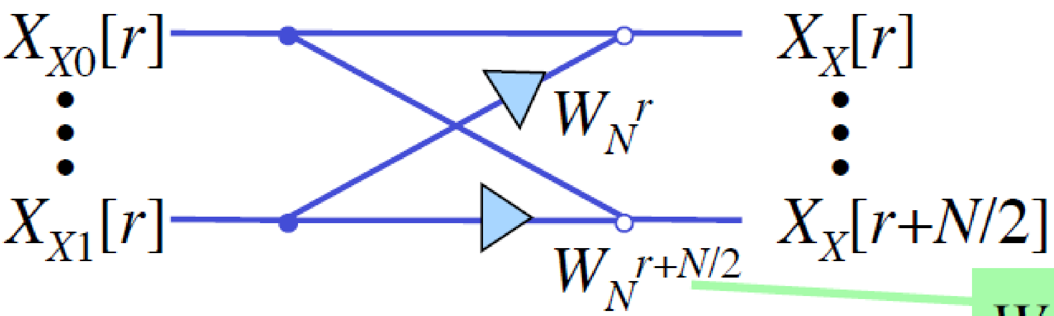
→ $N = 2^M$-pt DFT reduces to M stages of twiddle factors & summation ($O(N^2)$ part vanishes)

→ real mults $< M \cdot 4N$, real adds $< 2 \cdot M \cdot 2N$

→ complexity $\sim O(N \cdot M) = O(N \cdot \log_2 N)$

FFT Implementation Details

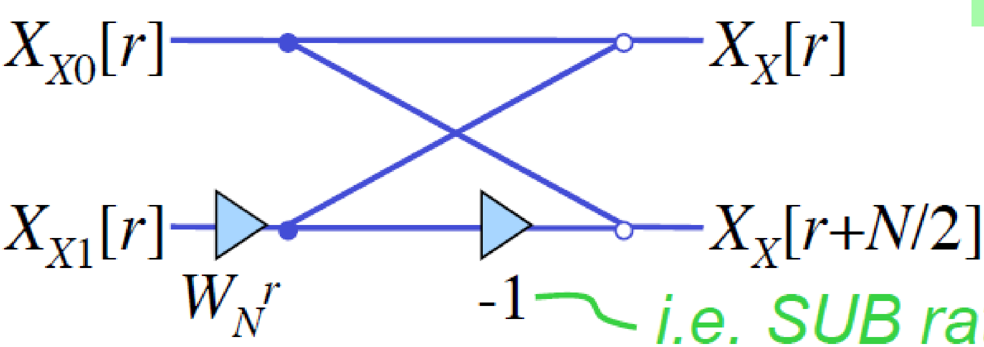
- Basic butterfly (at any stage):



2 cpx mults

$$W_N^{r+\frac{N}{2}} = e^{-j\frac{2\pi\left(r+\frac{N}{2}\right)}{N}}$$
$$= e^{-j\frac{2\pi r}{N}} \cdot e^{-j\frac{2\pi N/2}{N}}$$
$$= -W_N^r$$

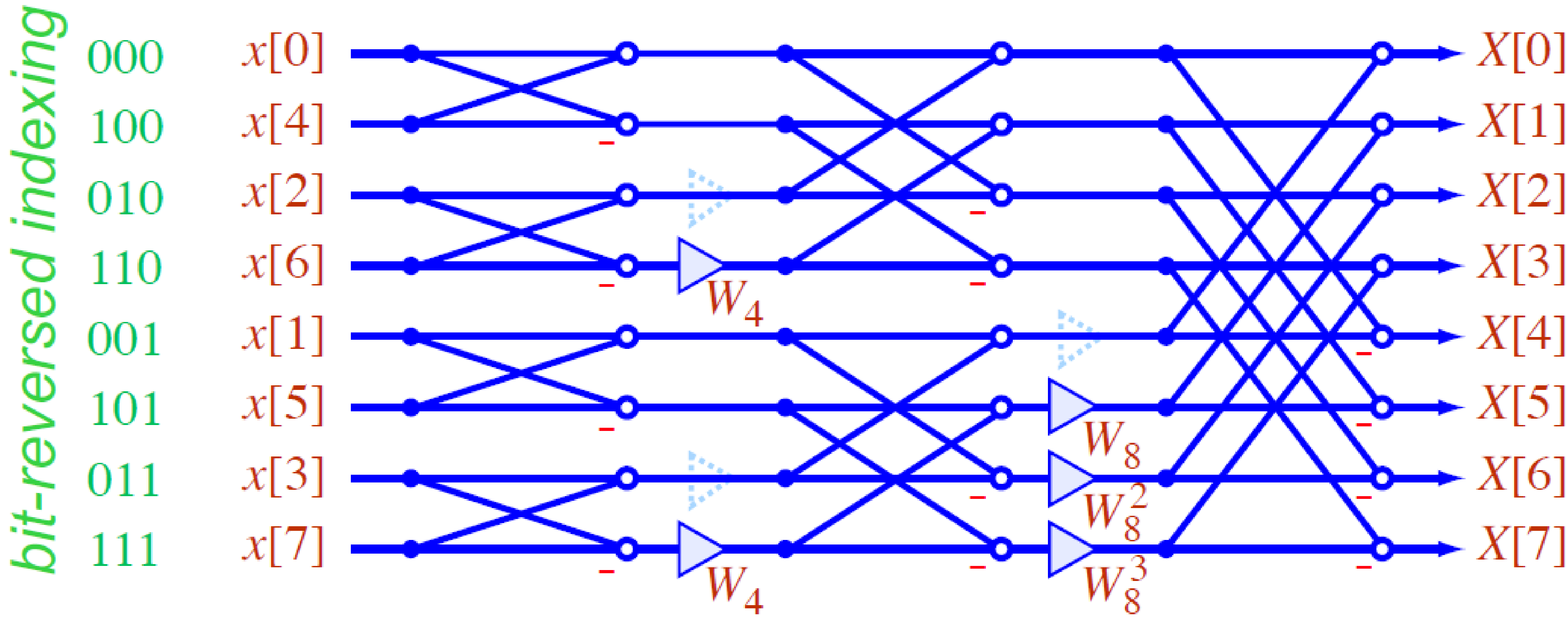
- Can simplify:



just one cpx mult!

i.e. SUB rather than ADD

8-pt DIT FFT Flowgraph



- -1's absorbed into summation nodes
- W_N^0 disappears
- 'in-place' algorithm: sequential stages

FFT for Other Values of N

- Having $N = 2^M$ meant we could divide each stage into 2 halves = "radix-2 FFT"
- Same approach works for:
  - $N = 3^M$ radix-3
  - $N = 4^M$ radix-4 - more optimized radix-2
  - etc...
- Composite $N = a \cdot b \cdot c \cdot d \rightarrow$ mixed radix (different N/r point FFTs at each stage)
  - .. or just zero-pad to make $N = 2^M$

Inverse FFT

only differences from forward DFT

- Recall IDFT: $$x[n] = \boxed{\frac{1}{N}} \sum_{k=0}^{N-1} X[k] W_N^{\boxed{-}nk}$$

- Thus:

***Forward** DFT of* $x'[n] = X^*[k]\big|_{k=n}$

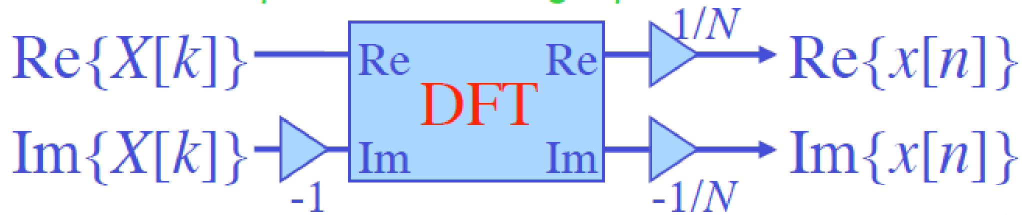
i.e. time sequence made from spectrum

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

- Hence, use FFT to calculate IFFT:

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

pure real flowgraph



DFT of Real Sequences

- If $x[n]$ is pure-real, DFT wastes mult's
- Real $x[n] \rightarrow$ Conj. symm. $X[k] = X^*[-k]$
- Given two real sequences, $x[n]$ and $w[n]$ call $y[n] = j \cdot w[n]$, $v[n] = x[n] + y[n]$
- N-pt DFT $V[k] = X[k] + Y[k]$
  <u>but</u>: $V[k]+V^*[-k] = X[k]+\underbrace{X^*[-k]}_{X[k]}+Y[k]+\underbrace{Y^*[-k]}_{-Y[k]}$

$\Rightarrow X[k]={}^1/_2(V[k]+V^*[-k])$, $W[k]={}^{-j}/_2(V[k]-V^*[-k])$

- i.e. compute DFTs of **two** N-pt real sequences with a single N-pt DFT