BINGHAMTON
UNIVERSITY
State University of New York

EEO 401
Digital Signal Processing
Prof. Mark Fowler

Note Set #26

- FFT Algorithm: Divide & Conquer Viewpoint
- Reading: Sect. 8.1.2 & 8.1.3 of Proakis & Manolakis

Divide & Conquer Approach

The previous note set's FFT development was somewhat ad hoc.

Here we develop a more formalized & generalized approach that can be used to develop other FFT approaches.

To illustrate the basic ideas consider the case of an N-pt DFT where N is not prime can be factored into two integer factors:

$$\boxed{N = LM}$$

Can always zero-pad out to appropriate number

We now can use either of two mappings from 2-D indices l,m to the actual time index n:

$$\boxed{n = Ml + m}$$

"Row-Wise Input Mapping"

$$\boxed{n = l + mL}$$

"Column-Wise Input Mapping"

We use one or the other of these mappings to convert the input to a matrix and then take DFTs along either the rows or columns.

We now can use either of two mappings from 2-D indices p,q to the actual DFT result index k:

$$\boxed{k = Mp + q}$$

"Row-Wise Output Mapping"

$$\boxed{k = p + qL}$$

"Column-Wise Output Mapping"

$$\boxed{n = Ml + m}$$

"Row-Wise Input Mapping"

$l \backslash m$	0	1	2		$M-1$
0	$x(0)$	$x(1)$	$x(2)$	$\cdots$	$x(M-1)$
1	$x(M)$	$x(M+1)$	$x(M+2)$	$\cdots$	$x(2M-1)$
2	$x(2M)$	$x(2M+1)$	$x(2M+2)$	$\cdots$	$x(3M-1)$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$L-1$	$x((L-1)M)$	$x((L-1)M+1)$	$x((L-1)M+2)$	$\cdots$	$x(LM-1)$

DFTs Down Columns

(a)

$$\boxed{n = l + mL}$$

"Column-Wise Input Mapping"

$l \backslash m$	0	1	2		$M-1$
0	$x(0)$	$x(L)$	$x(2L)$	$\cdots$	$x((M-1)L)$
1	$x(1)$	$x(L+1)$	$x(2L+1)$	$\cdots$	$x((M-1)L+1)$
2	$x(2)$	$x(L+2)$	$x(2L+2)$	$\cdots$	$x((M-1)L+2)$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$L-1$	$x(L-1)$	$x(2L-1)$	$x(3L-1)$	$\cdots$	$x(LM-1)$

DFTs Across Rows

Now to illustrate how to use this machinery… Use

- Column-wise input mapping $\quad n = l + mL$
- Row-wise output mapping $\quad k = Mp + q$

$$\left.\begin{aligned} n &= l + mL \\ k &= Mp + q \end{aligned}\right\} \quad X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

$$\Rightarrow \quad X[p,q] = \sum_{m=0}^{M} \sum_{l=0}^{L-1} x[l,m] \underbrace{W_N^{(Mp+q)(l+mL)}}_{= \underbrace{W_N^{MLmp}}_{=W_N^{Nmp}=1} \underbrace{W_N^{mLq}}_{W_{N/L}^{mq}} \underbrace{W_N^{Mpl}}_{=W_{N/M}^{pl}} W_N^{lq}}$$

$$\underbrace{X[p,q] = \sum_{l=0}^{L} \underbrace{\left\{ W_N^{lq} \underbrace{\left[\sum_{m=0}^{M-1} x[l,m] W_{N/L}^{mq} \right]}_{\triangleq F[l,q]} \right\}}_{\triangleq G[l,q]} W_{N/M}^{pl}}_{=X[p,q]}$$

Compute *M*-pt DFTs of Rows

Apply Twiddle Factors

Compute *L*-pt DFTs of Colns

Illustration of this Approach

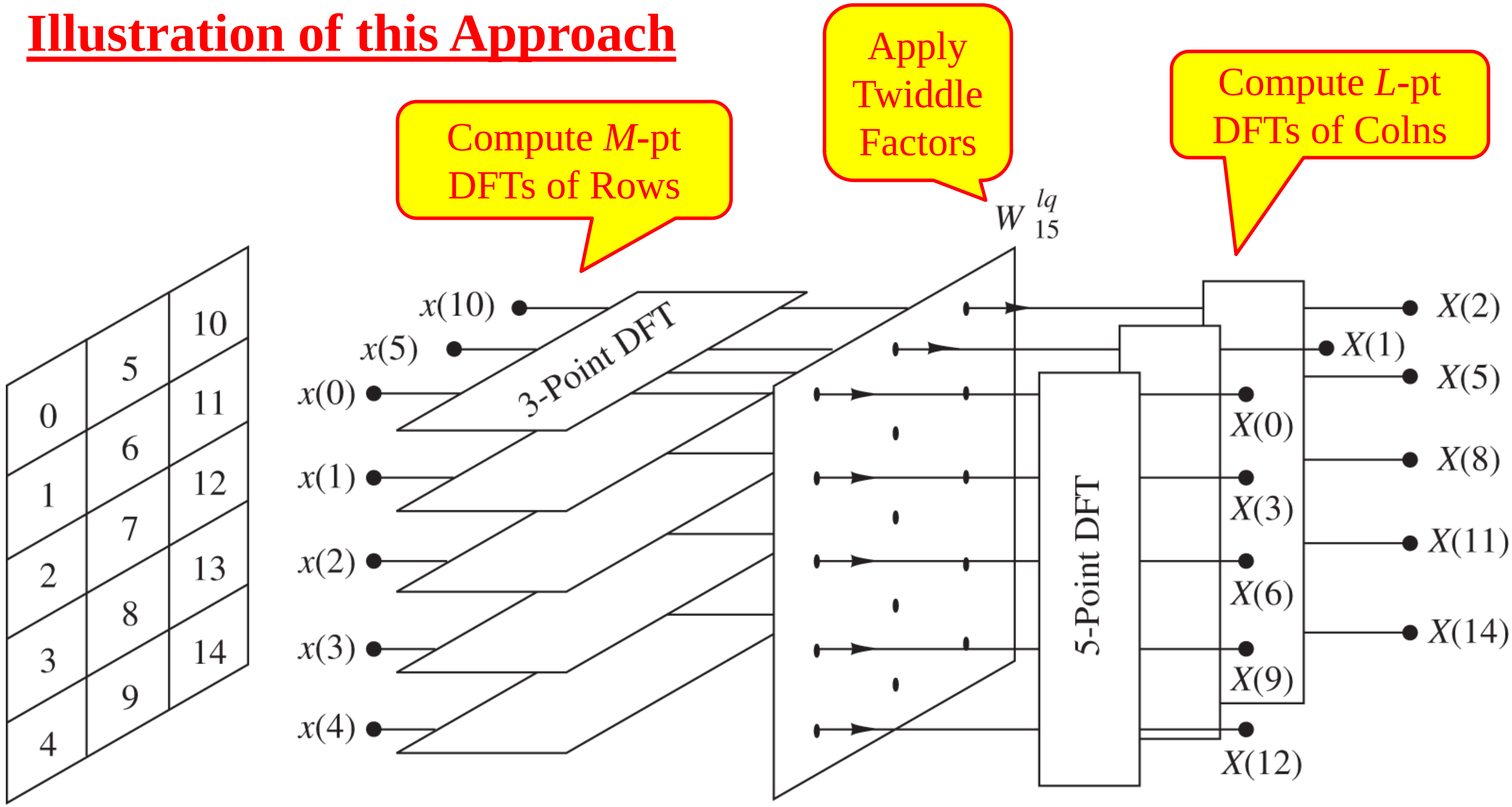


Figure 8.1.3 Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

Although this LOOKS more complicated… it is actually more efficient!

Application to Develop Dec-In-Time Radix-2 FFT

Let $N = 2^{v}$

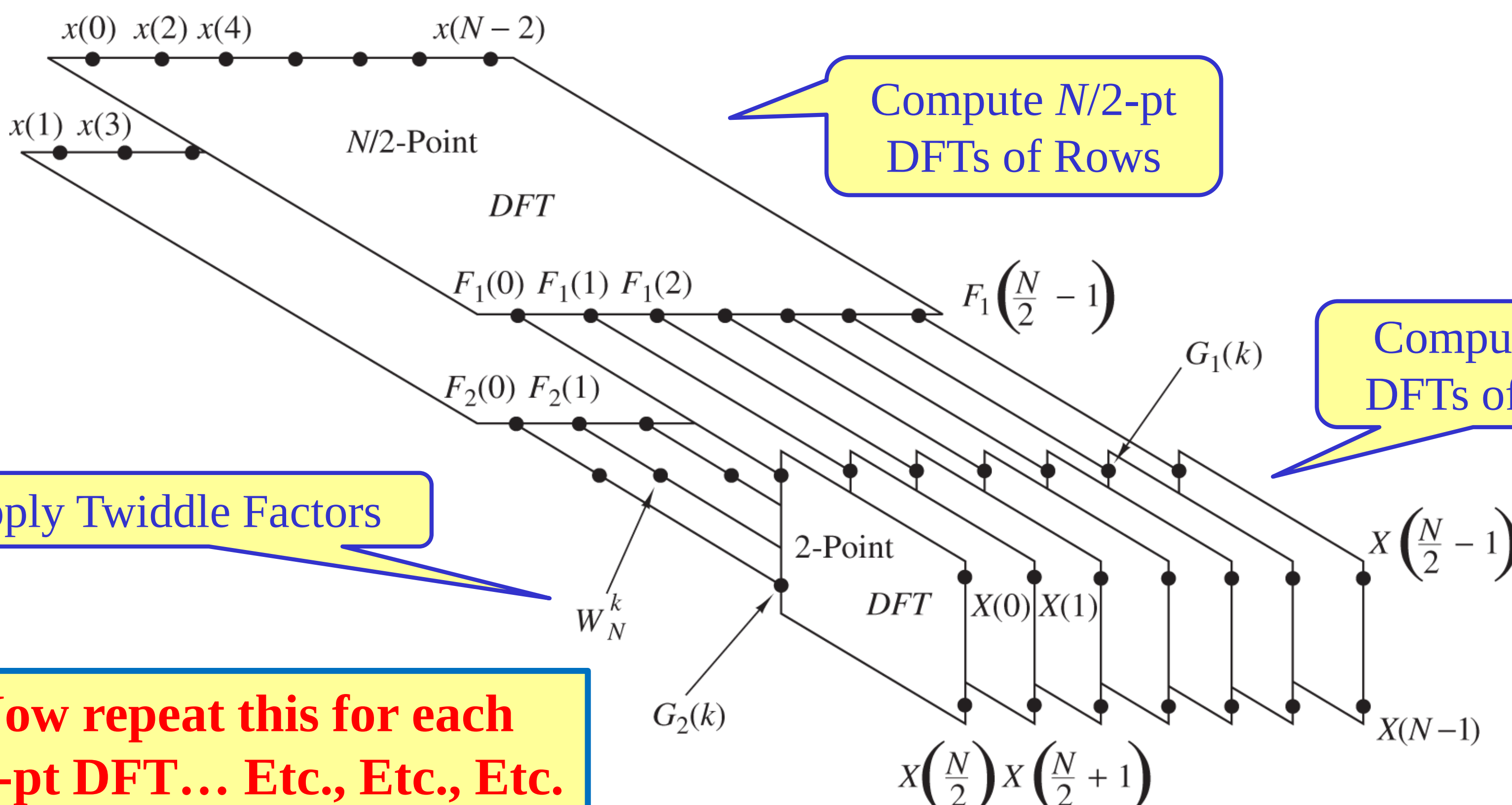
We apply the divide-and-conquer approach with $\boldsymbol{M = N/2}$ **&** $\boldsymbol{L = 2}$

$$\boxed{n = l + mL}$$

"Column-Wise Input Mapping"

$l \backslash m$	0	1	2		$M-1$
0	$x[0]$	$x[2]$	$x[4]$	$\cdots$	$x[N-2]$
1	$x[1]$	$x[3]$	$x[5]$	$\cdots$	$x[N-1]$

DFTs Across Rows



Compute $N/2$-pt DFTs of Rows

Compute 2-pt DFTs of Colns

Apply Twiddle Factors

Now repeat this for each $N/2$-pt DFT… Etc., Etc., Etc.

Application to Develop Dec-In-Frequency Radix-2 FFT $N = 2^{\nu}$

We apply the divide-and-conquer approach with **$M = 2$ & $L = N/2$**

$$\boxed{n = l + mL}$$

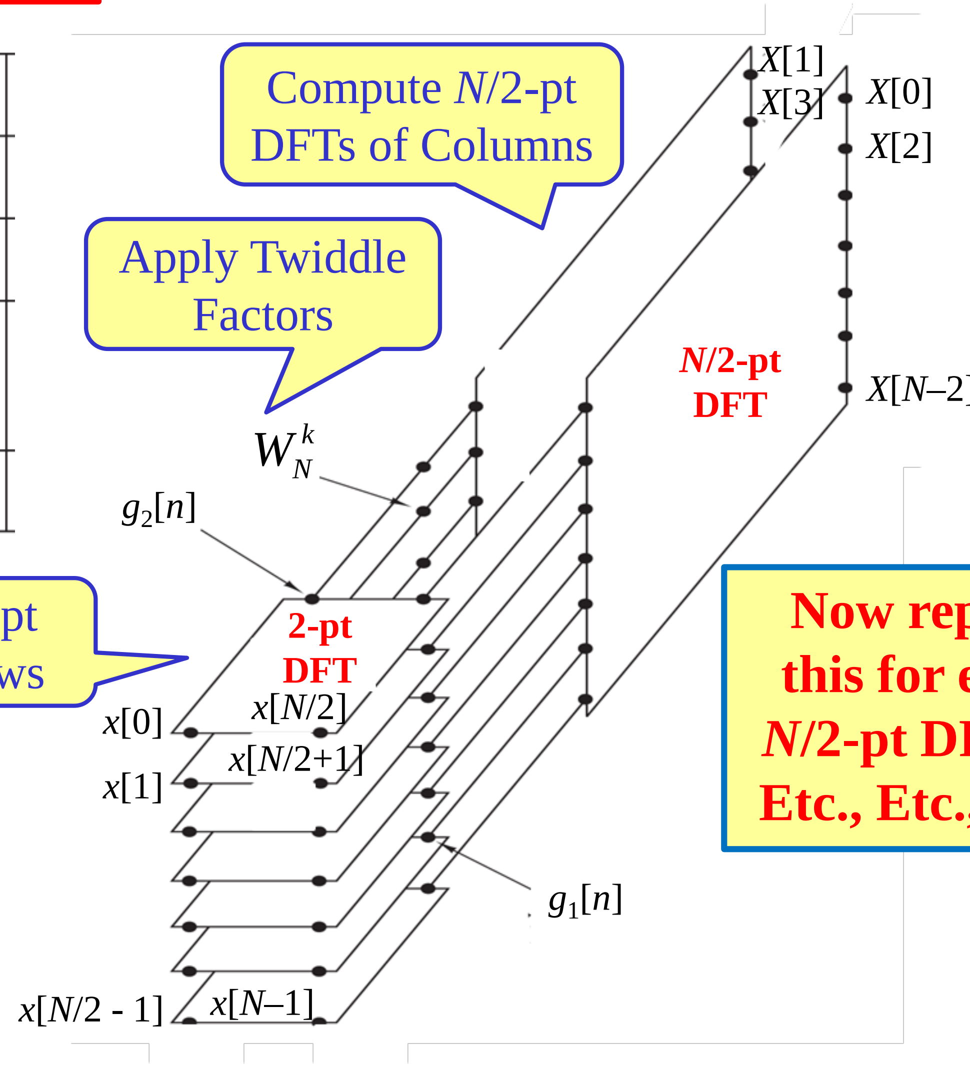
"Column-Wise Input Mapping"

$l \backslash m$	0	1
0	$x[0]$	$x[N/2]$
1	$x[1]$	$x[N/2+1]$
2	$x[2]$	$x[N/2+2]$
	$\vdots$	$\vdots$
$L-1$	$x[N/2-1]$	$x[N-1]$

Compute $N/2$-pt DFTs of Columns

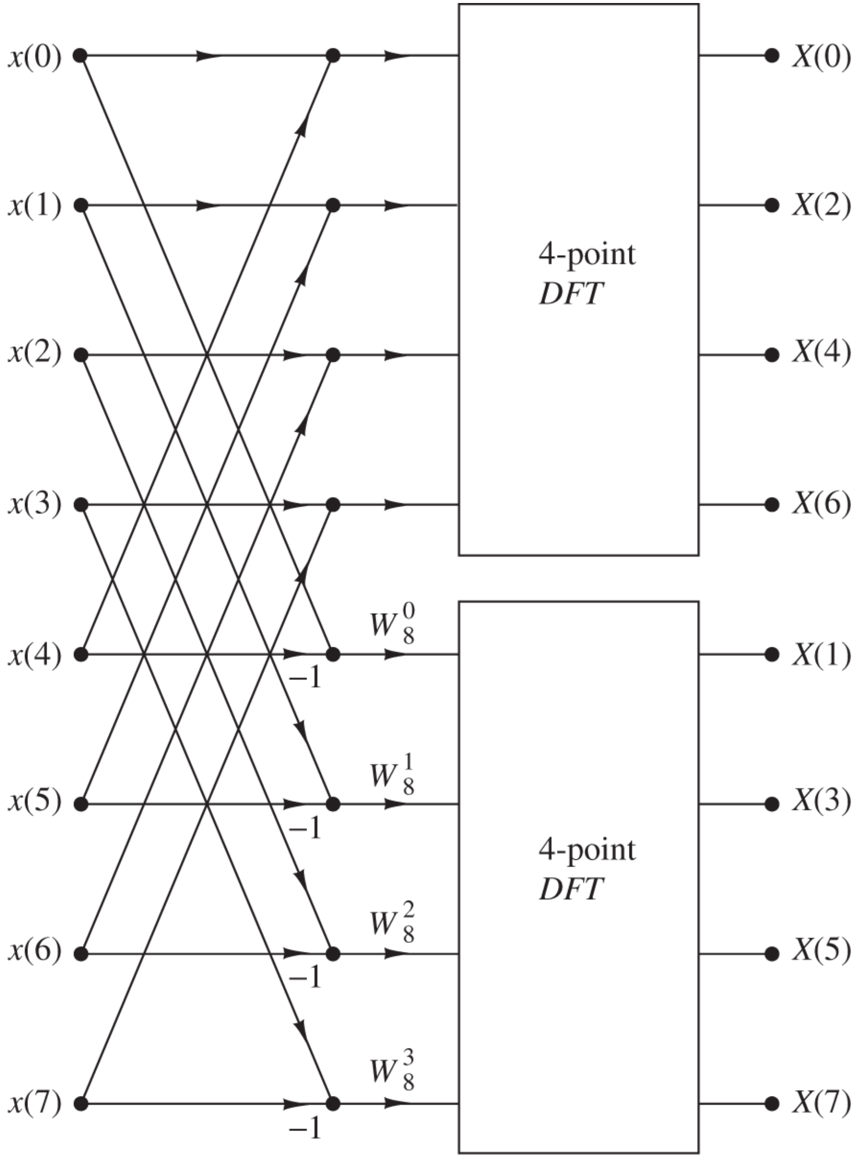
Apply Twiddle Factors

Compute 2-pt DFTs of Rows



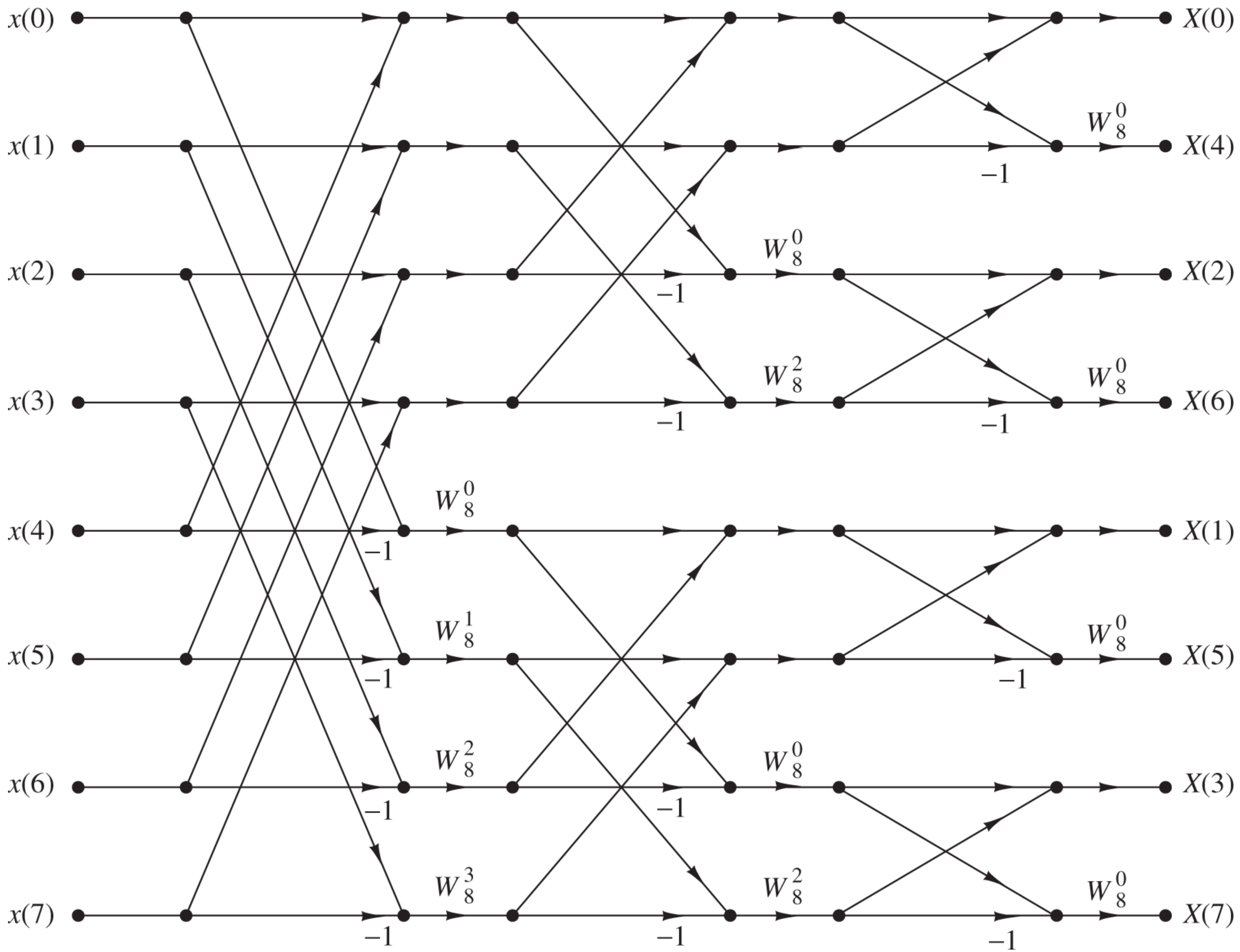
Now repeat this for each $N/2$-pt DFT… Etc., Etc., Etc.

N = 8 First Stage of Dec-in-Freq FFT

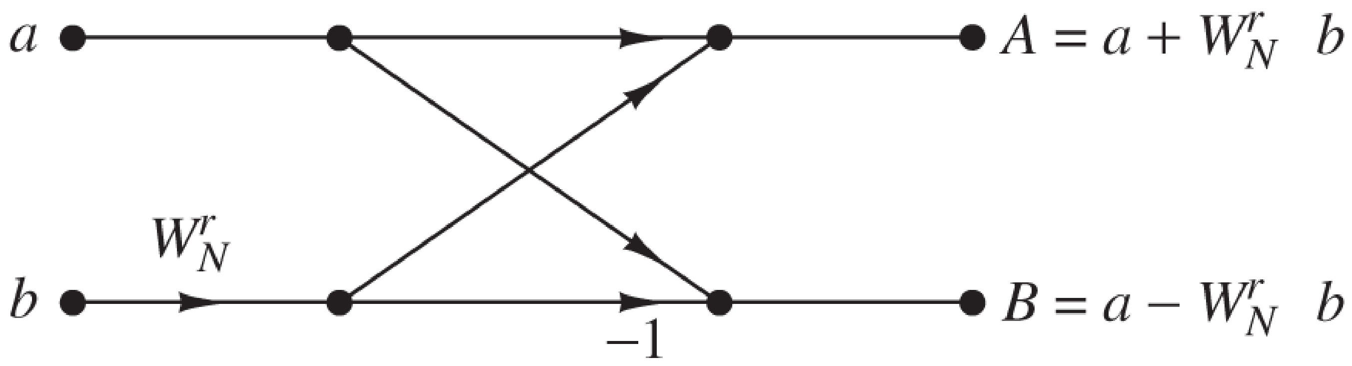


Complete Dec-in-Freq FFT for N = 8

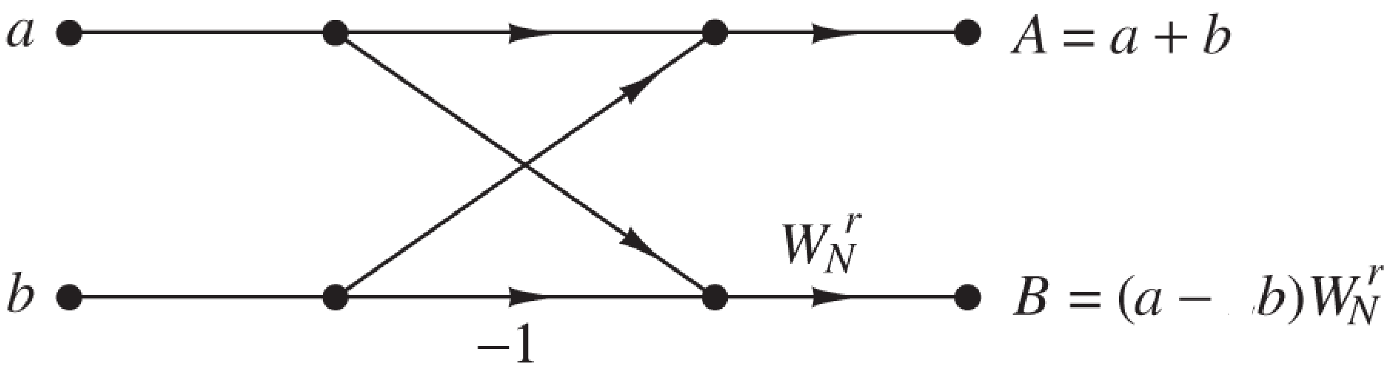
DFT Values are in Bit Reversed Order!



Butterfly Structure: DiT vs DiF

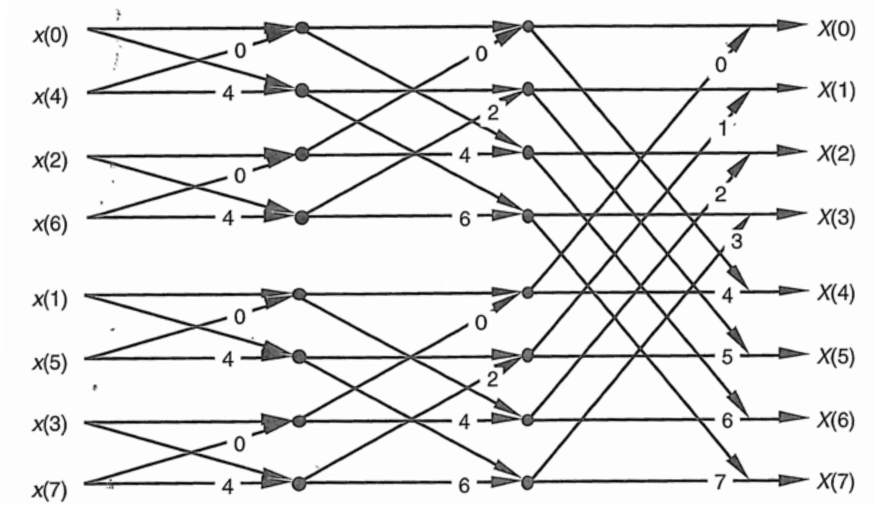


Butterfly Structure: Dec-in-Time



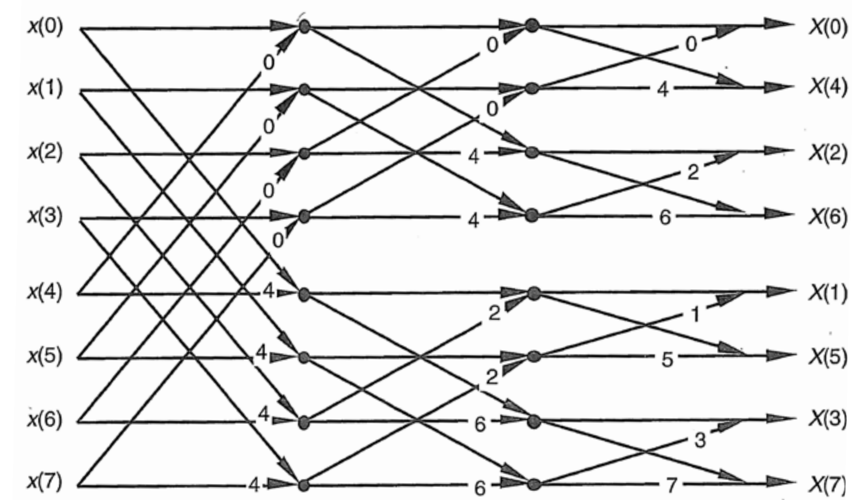
Butterfly Structure: Dec-in-Freq

3 Different Configurations of D-in-Time FFT



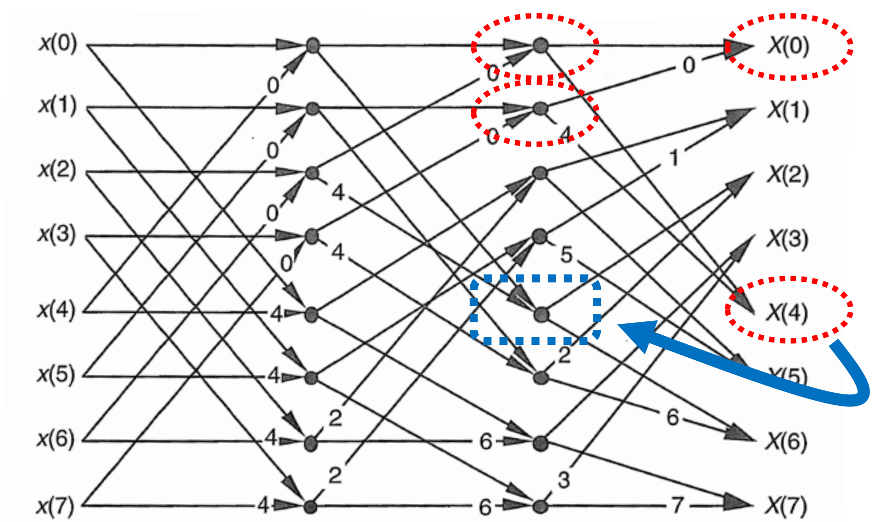
***N*=8 D-in-<u>Time</u> FFT w/ BR <u>Inputs</u>**

<u>Can</u> be done "in-place"



***N*=8 D-in-<u>Time</u> FFT w/ BR <u>Outputs</u>**

<u>Can</u> be done "in-place"



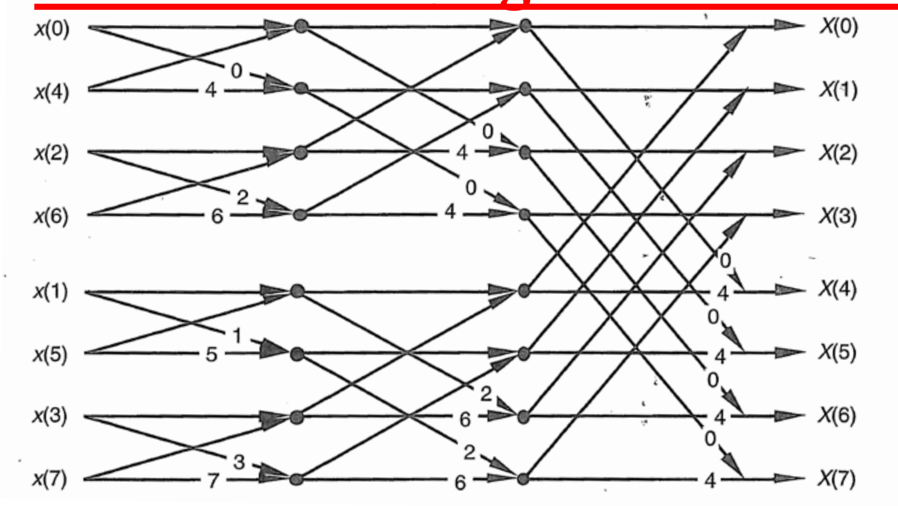
Diagrams from R. Lyon, *Understanding Digital Signal Processing*, 3rd Ed., Prentice-Hall, 2011

Writes over Data Needed Later!!

***N*=8 D-in-<u>Time</u> FFT w/ both sides "Normal Order"**

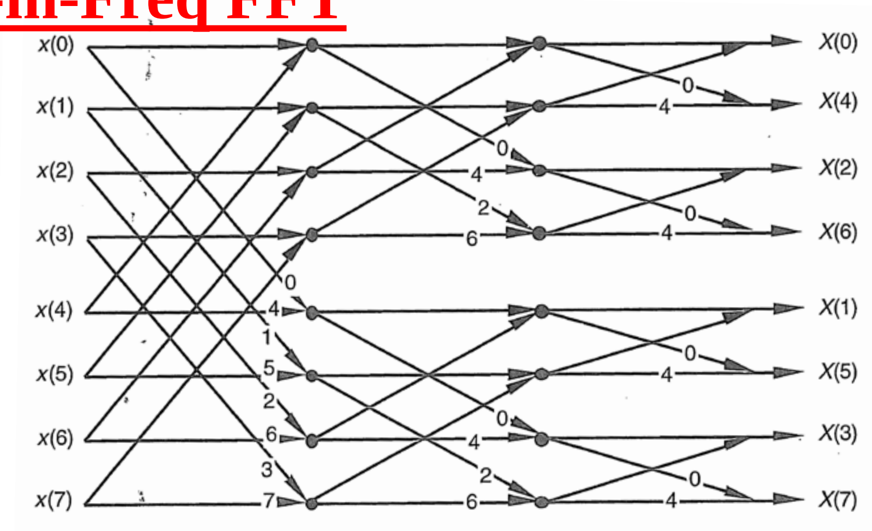
Can <u>NOT</u> be done "in-place"

<u>3 Different Configurations of D-in-Freq FFT</u>



***N*=8 D-in-<u>Freq</u> FFT w/ BR <u>Inputs</u>**

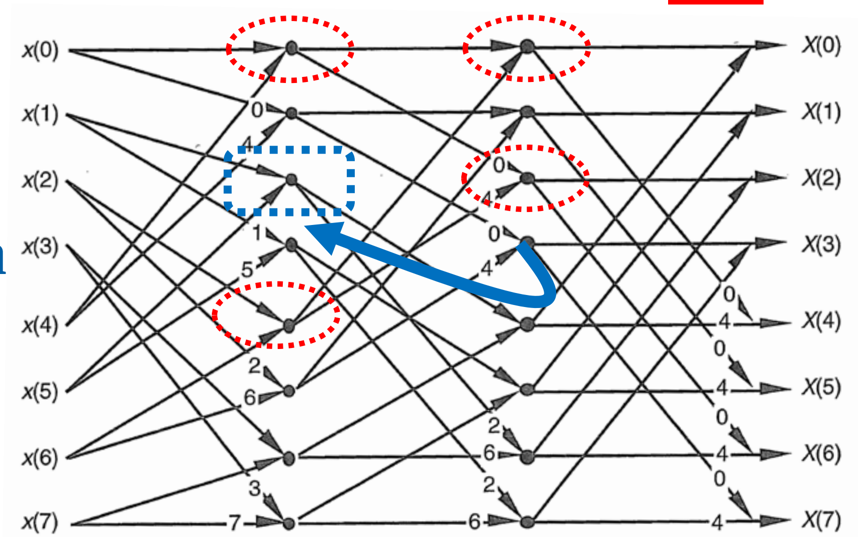
<u>Can</u> be done "in-place"



***N*=8 D-in-<u>Freq</u> FFT w/ BR <u>Outputs</u>**

<u>Can</u> be done "in-place"

Writes over Data Needed Later!!



Diagrams from R. Lyon, *Understanding Digital Signal Processing*, 3rd Ed., Prentice-Hall, 2011

***N*=8 D-in-<u>Freq</u> FFT w/ both sides "Normal Order"**

Can <u>NOT</u> be done "in-place"

Implementation Issues

- We've looked at two radix-two methods.
  - Other radices: 4 & 8
  - split radix (2 and 4)

TABLE 8.2 Number of Nontrivial Real Multiplications and Additions to Compute an N-point Complex DFT

	Real Multiplications				Real Additions			
N	Radix 2	Radix 4	Radix 8	Split Radix	Radix 2	Radix 4	Radix 8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1,032	976	972	964
128	712			516	2,504			2,308
256	1,800	1,392		1,284	5,896	5,488		5,380
512	4,360		3,204	3,076	13,566		12,420	12,292
1,024	10,248	7,856		7,172	30,728	28,336		27,652

Source: Extracted from Duhamel (1986).

- In-Place computation requires only 2N memory locations
  - But complicates the indexing & control operations
  - Doubling the memory to 4N locations can be advantageous
    - Reduces complexity of indexing & control
    - Allows natural ordering for both input and output
- In general, many factors come into play when determining best method
  - Parallelism, HW vs SW, fixed-point vs floating-point, etc.
- Also… no need to develop distinct IFFT algorithm

$$x[n]=\frac{1}{N}\sum_{n=0}^{N-1}X[k]e^{j2\pi kn/N}=\frac{1}{N}\left[\sum_{n=0}^{N-1}X^*[k]e^{-j2\pi kn/N}\right]^* \Rightarrow \boxed{IDFT\{X[k]\}=\frac{1}{N}conj\{DFT\{X^*[k]\}\}}$$

Two Tricks for Real-Valued Signals

1. Efficient DFT of two Real-Valued Signals

Let $x_1[n]$ and $x_2[n]$ be real-valued signals, each length N

Form: $x[n] \triangleq x_1[n] + jx_2[n]$

Then we have $$x_1[n] = \frac{x[n] + x^*[n]}{2} \quad \& \quad x_2[n] = \frac{x[n] - x^*[n]}{2j}$$

Thus

$$X_1[k] = \frac{DFT\left\{x[n]\right\} + DFT\left\{x^*[n]\right\}}{2} \quad \& \quad X_2[k] = \frac{DFT\left\{x[n]\right\} - DFT\left\{x^*[n]\right\}}{2j}$$

But… DFT{ $x^*[n]$} = $X^*[N - k]$

$$\boxed{X_1[k] = \frac{1}{2}\left[X[k] + X^*[N-k]\right] \quad \& \quad X_2[k] = \frac{1}{2j}\left[X[k] - X^*[N-k]\right]}$$

2. Efficient DFT of 2*N*-pt Real-Valued Signal

Let $g[n]$ be a real-valued signal of length $2N$

Then define: $x_1[n] \triangleq g[2n]$ & $x_2[n] \triangleq g[2n+1]$

And: $x[n] = x_1[n] + \mathrm{j}\, x_2[n]$

From Trick #1 we have

$$X_1[k] = \frac{1}{2}\left[X[k] + X^*[N-k] \right] \qquad \& \qquad X_2[k] = \frac{1}{2j}\left[X[k] - X^*[N-k] \right]$$

But using ideas from Dec-in-Time FFT we know

$$\begin{aligned} G[k] &= \sum_{n=0}^{N-1} g[2n] W_{2N}^{2nk} + \sum_{n=0}^{N-1} g[2n+1] W_{2N}^{(2n+1)k} \\ &= \sum_{n=0}^{N-1} x_1[n] W_{2N}^{2nk} + \sum_{n=0}^{N-1} x_2[n] W_{2N}^{(2n+1)k} \end{aligned}$$

So then we get

$$G[k] = X_1[k] + W_{2N}^{k} X_2[k], \quad k = 0,1,\ldots,N-1$$

$$G[k+N] = X_1[k] - W_{2N}^{k} X_2[k], \quad k = 0,1,\ldots,N-1$$

So computing one *N*-pt DFT of *x*[*n*] gets us the 2*N*-pt DFT of *g*[*n*]!