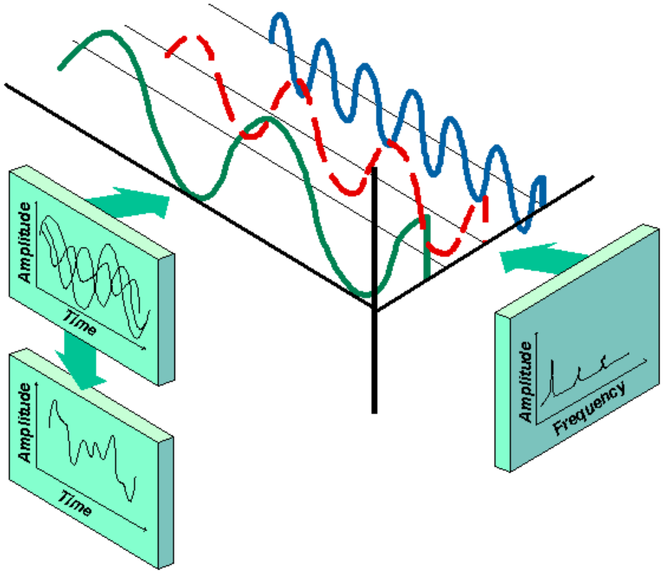
Ch3.3: Fast Fourier Transform

Information source and input transducer → Source Coding → Channel Coding → Modulator

Ch3.3: FFT



- **Questions to be answered:**
  - **Fast-Fourier Transform:** **How can we efficiently compute DFT.**

Channel

Information sink and output transducer ← Source Decoding ← Channel Decoding ← Demodulator (Matched Filter)

Reading: Mitra (Ch11.3.2, Decimation-in-time FFT algorithm)

FFT Applications

- OFDM
  - DAB
  - HDTV
  - Wireless LAN Networks
    - 1 HIPERLAN/2
    - 2 IEEE 802.11a
    - 3 IEEE 802.11g
  - IEEE 802.16 Broadband Wireless Access Systems

Computation Complexity of DFT

- Recall the definition of DFT,

n complex multiplication!

Synthesis equation	$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{kn} \quad for \quad 0\le k\le N-1$	DFT	where $W_N=e^{-j\frac{2\pi}{N}}$
Analysis equation	$x[n]=\frac{1}{N}\sum_{n=0}^{N-1}X[k]W_N^{-kn} \quad for \quad 0\le n\le N-1$	IDFT	

calculated before!

- Each $X[k]$ involves N and $(N-1)$ complex multiplications and complex additions respectively. Assume that the values of W_N^{kn} are pre-computed.
- 1 complex multiplication requires 4 real multiplications and 2 real additions.

$$(a+jb)(c+jd)=(ac-bd)+j(ad+bc)$$

- 1 complex addition requires 2 real additions.

$$(a+jb)+(c+jd)=(a+c)+j(b+d)$$

Computation Complexity of DFT

- As $X[k]$ must be computed for N different value of k, we require
  - N^2 complex multiplications and $N(N-1)$ complex additions.

  ➔ That is $4N^2$ real multiplications and $N(4N-2)$ real additions.

  ➔ Complexity is of order of N^2 , denotes as $O(N^2)$

$\Rightarrow O(N \log N)$

N	Complex Multiplication N^2	Complex Addition $N(N-1)$
2	4	2
8	64	56
32	1024	922
64	4096	4022
128	16384	16256
2^{10}	1048576	1047522
2^{20}	$\sim 10^{12}$	$\sim 10^{12}$

Efficient Computation of DFT

- There are several algorithms to reduce the computation complexity:
  - The Goertzel Algorithm
    - To avoid the computation or storage of all the coefficients W_N^{kn} in

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

    - The drawback of this algorithm is slightly less efficient than the direct method.
  - Decimation-in-time (DIT) Fast Fourier Transform (FFT) algorithm
    - Decomposing the time-domain sequence $x[n]$ of length N into successively small sub-sequences.
  - Decimation-in-frequency (DIF) Fast Fourier Transform (FFT) algorithm
    - Decomposing the frequency-domain sequence $X[k]$ of length N into successively small sub-sequences.

Efficient Computation of DFT

- Taking the advantages of the properties of $W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$
  - Complex conjugate symmetry:

$n \to N-n$

$$W_N^{k[N-n]} = e^{-j\frac{2\pi}{N}k(N-n)} = e^{-j2\pi k}e^{+j\frac{2\pi}{N}kn} = e^{+j\frac{2\pi}{N}kn} = W_N^{-kn} = \left(W_N^{kn}\right)^*$$

  - Periodicity in n and k:

$n \to n+N$

$k \to k+N$

$$W_N^{k(n+N)} = e^{-j\frac{2\pi}{N}k(n+N)} = e^{-j2\pi k}e^{-j\frac{2\pi}{N}kn} = e^{-j\frac{2\pi}{N}kn} = W_N^{kn}$$

$$W_N^{(k+N)n} = e^{-j\frac{2\pi}{N}(k+N)n} = e^{-j\frac{2\pi}{N}kn}e^{-j2\pi n} = e^{-j\frac{2\pi}{N}kn} = W_N^{kn}$$

- Assumptions:
  - $N = 2^v$ where v is a positive integer, although there exist FFT algorithms for other values of N.
  - $x[n]$ is complex sequence although it also works for real $x[n]$.

Decimation-in-Time FFT Algorithm

- Decomposing the time-domain sequence $x[n]$ of length N into successively small sub-sequences.

- Separate $x[n]$ of length N into two $(N/2)$-point sequences

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{odd} x[n] W_N^{kn} + \sum_{even} x[n] W_N^{kn} \qquad for \quad 0 \le k \le N-1$$

- Substitute $n = 2r$ for n even and $n = 2r + 1$ for n odd,

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{(2r+1)k} \qquad for \quad 0 \le k \le N-1$$

$$= \sum_{r=0}^{(N/2)-1} x[2r] W_N^{2rk} + W_N^{k} \sum_{r=0}^{(N/2)-1} x[2r+1] W_N^{2rk}$$

Decimation-in-Time FFT Algorithm

- With the property $\boxed{W_N^{2rk} = e^{-j(2\pi/N)2rk} = e^{-j[2\pi/(N/2)]rk} = W_{N/2}^{rk}}$

$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_N^{2rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_N^{2rk} \quad for \quad 0 \le k \le N-1$$

$$= \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk}$$

DFT of even

DFT of odd number

$$= G[k] + W_N^k \boxed{H[k]} \quad for \quad 0 \le k \le N-1$$

(periodic
check of N)

where $G[k]$ is the $(N/2)$-point DFT of the even-numbered points of $x[n]$ and $H[k]$ is the $(N/2)$-point DFT of the odd-numbered points of $x[n]$.

Decimation-in-Time FFT Algorithm

- The $N/2$-point DFTs of the even- and odd-numbered sequences, $G[k]$ and $H[k]$, are calculated for the range $k = 0\ldots(N/2\text{-}1)$
- However, the values of the $N/2$-point DFT vary periodically with period $N/2$
  - $G[k] = G[k+N/2]$ ← Period and $H[k] = H[k+N/2]$
- Therefore, we can compute all points of the N-point DFT using two $N/2$-point DFTs as follows

$$X[k] = G[k] + W_N^k H[k] \qquad 0 \le k \le \frac{N}{2} - 1$$

$$= G[k+\tfrac{N}{2}] + W_N^{k+N/2} H[k+\tfrac{N}{2}]$$

一樣

$$X[k+N/2] = G[k] + W_N^{k+N/2} H[k]$$

$$= G[k] - W_N^k H[k] \qquad 0 \le k \le \frac{N}{2} - 1$$

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

$$= e^{-j\pi} = \text{-}1$$

Decimation-in-Time FFT Algorithm

- Therefore, we can compute the *N*-point of DFT by

$$\boxed{\begin{aligned} X[k] &= G[k] + W_N^k H[k] \quad \mathit{for} \quad 0 \le k \le (N/2)-1 \\ X[k+N/2] &= G[k] - W_N^k H[k] \quad \mathit{for} \quad 0 \le k \le (N/2)-1 \end{aligned}}$$

- Each pair of $X[k]$ and $X[k + N/2]$ is known as butterfly computation.

First step of decomposition

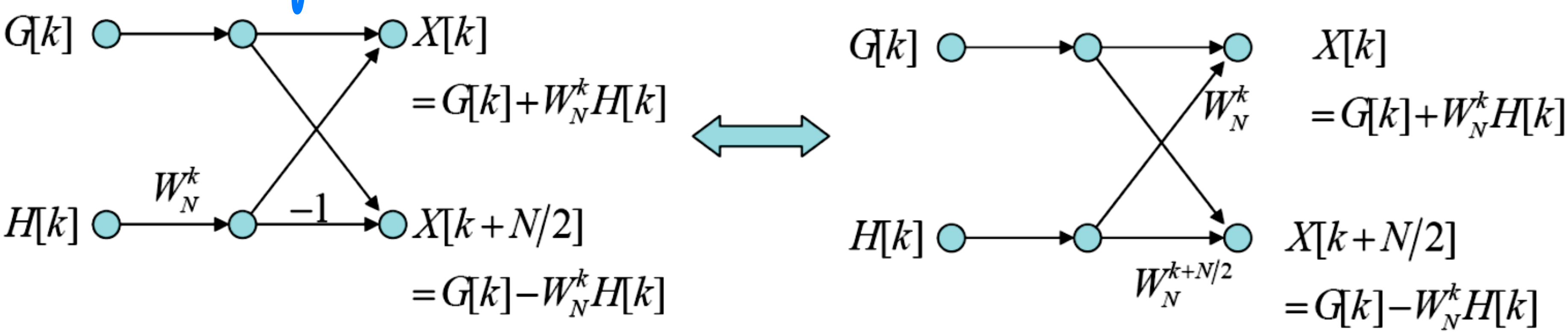
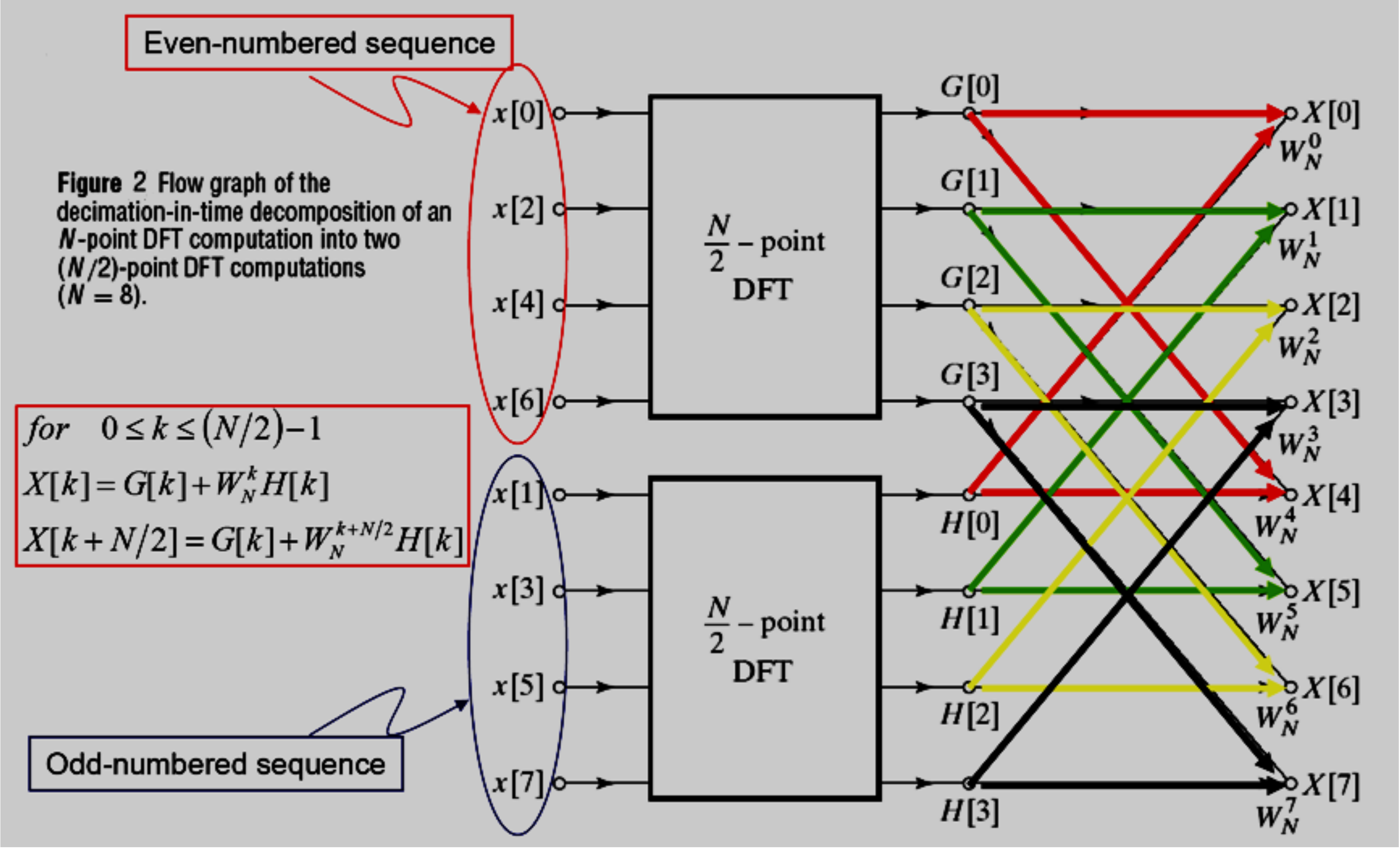


Fig. 1 Flow Graph Representation of N-Point DFT Computation Using Two N/2-Point DFTs

Decimation-in-Time FFT Algorithm - Example



Decimation-in-Time FFT Algorithm

- Compare the computation load of DIT-FFT with (N/2)-point DFT and N-point DFT

	Complex Multiplications	Complex Additions
N-point DFT	N^2	$N(N-1)$
DIT-FFT with N/2-point DFT		
Step 1: N/2-point DFT $G[k]$	$(N/2)^2$	$(N/2)((N/2)-1)$
Strep 2: N/2-point DFT $H[k]$	$(N/2)^2$	$(N/2)((N/2)-1)$
Step 3:Combine $G[k]$ and $H[k]$ using butterfly computation	$N/2$	$2(N/2)$
Sub-total	$N^2/2+N/2$	$N^2/2$

For large N, the computational load of the DIT-FFT is approximately halved !

$$\begin{aligned}
&\mathit{for}\quad 0 \le k \le (N/2)-1 \\
&X[k] = G[k] + W_N^k H[k] \\
&X[k+N/2] = G[k] - W_N^k H[k]
\end{aligned}$$

Roughly halved!

Decimation-in-Time FFT Algorithm - Example

Construct each of the 4-point DFTs using a combination of 2-point DFTs

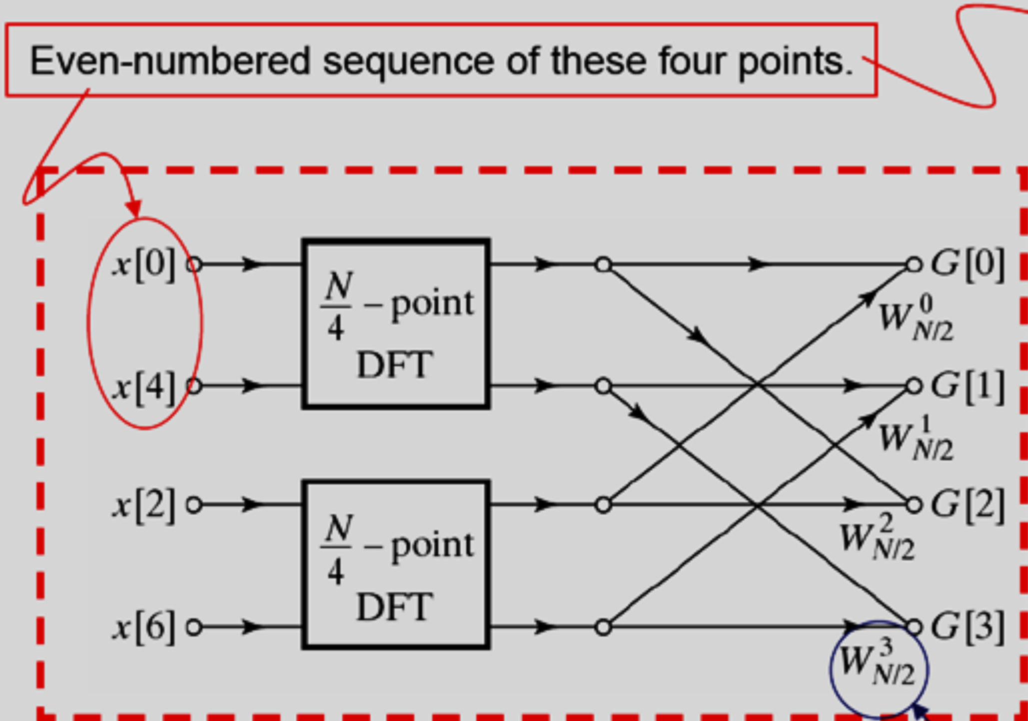


Figure 4 Flow graph of the decimation-in-time decomposition of an $(N/2)$-point DFT computation into two $(N/4)$-point DFT computations $(N = 8)$.

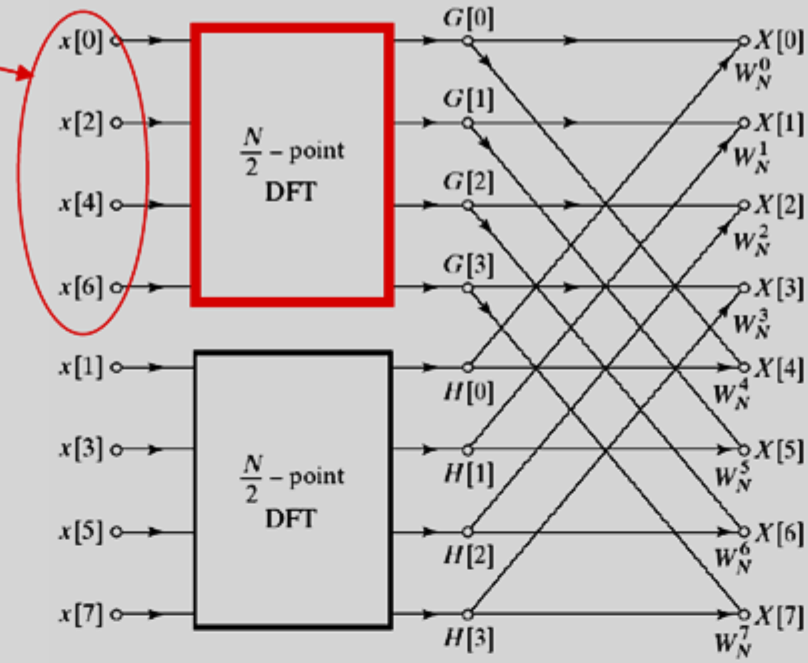


Figure 3 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $(N/2)$-point DFT computations $(N = 8)$.

$$W^3_{N/2} = e^{-j\frac{2\pi}{N/2}3} = e^{-j\frac{2\pi}{N}6} = W^6_N$$

Decimation-in-Time FFT Algorithm - Example

Construct each of the 4-point DFTs using a combination of 2-point DFTs

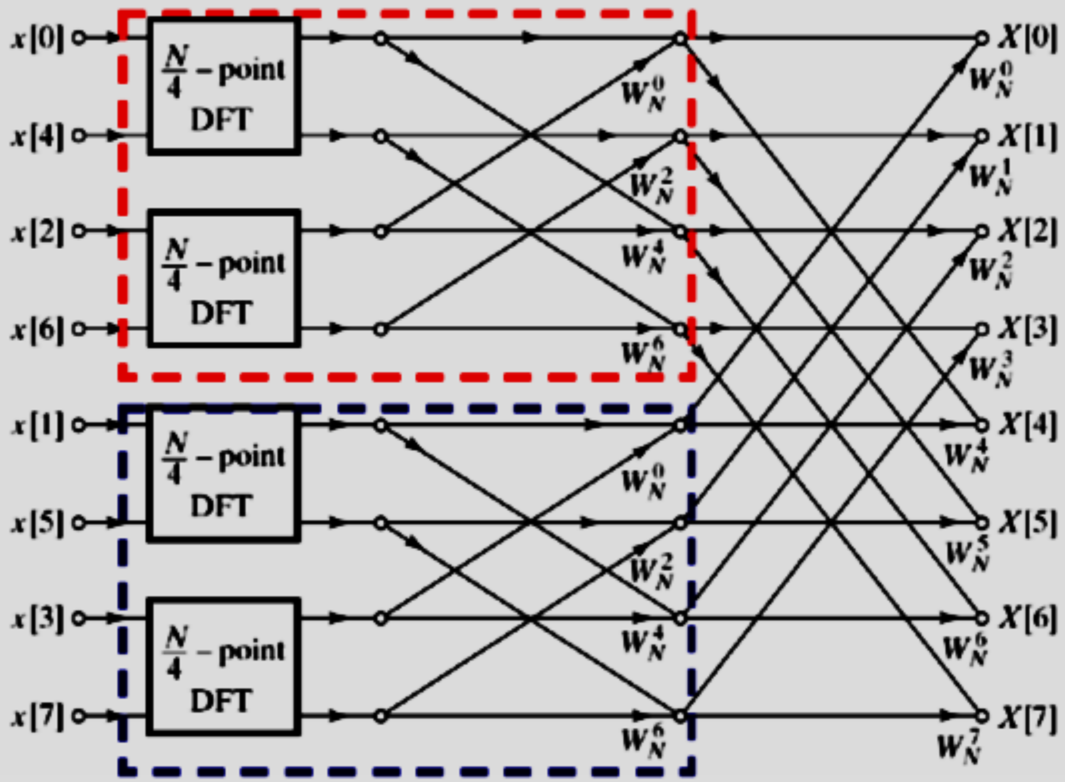


Figure 5 Result of substituting the structure of Figure 4 into Figure 3.

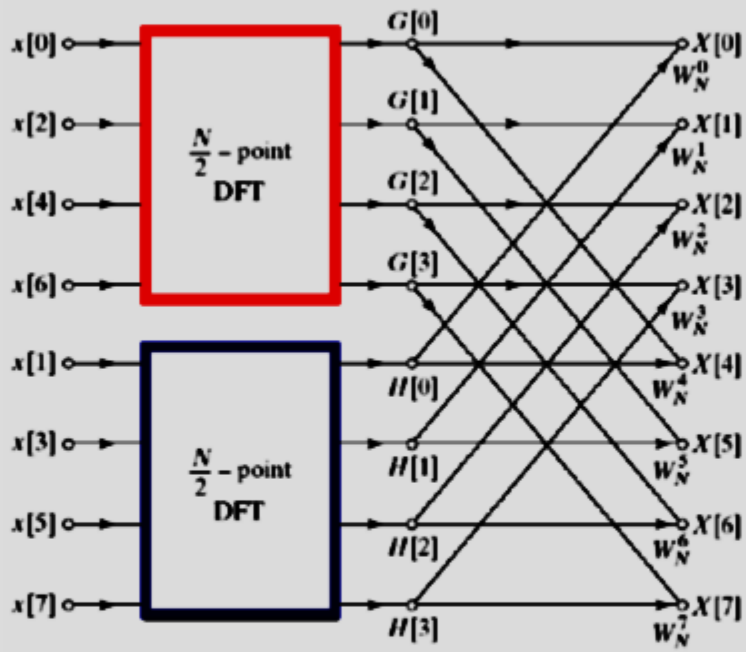
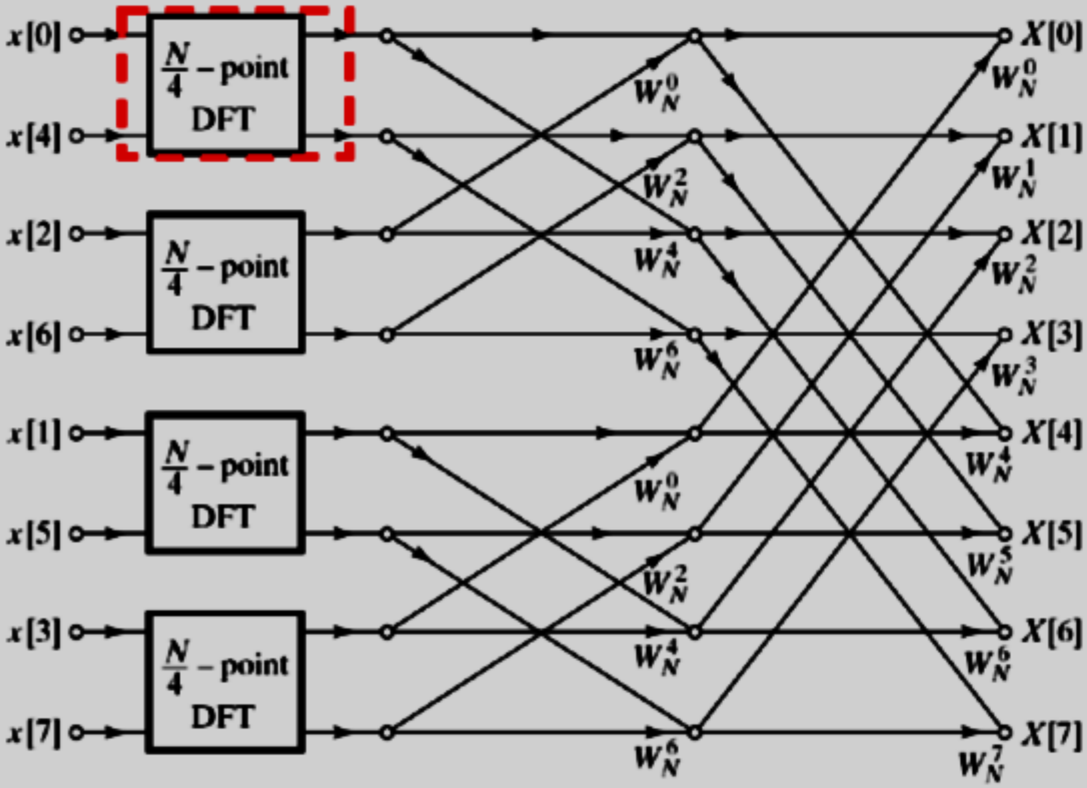


Figure 3 Flow graph of the decimation-in-time decomposition of an *N*-point DFT computation into two (*N*/2)-point DFT computations ($N = 8$).

Decimation-in-Time FFT Algorithm - Example

Construct each of the 2-point DFTs using an elementary butterfly

Figure 5 Result of substituting the structure of Figure 4 into Figure 3.



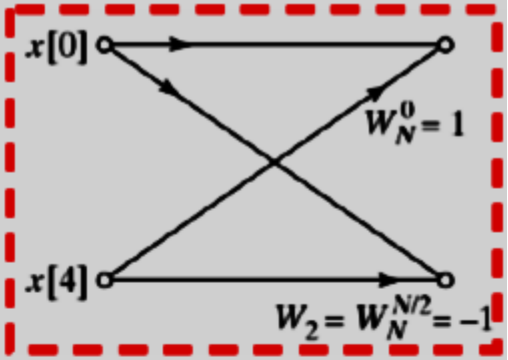


Figure 6 Flow graph of a 2-point DFT.

Decimation-in-Time FFT Algorithm - Example

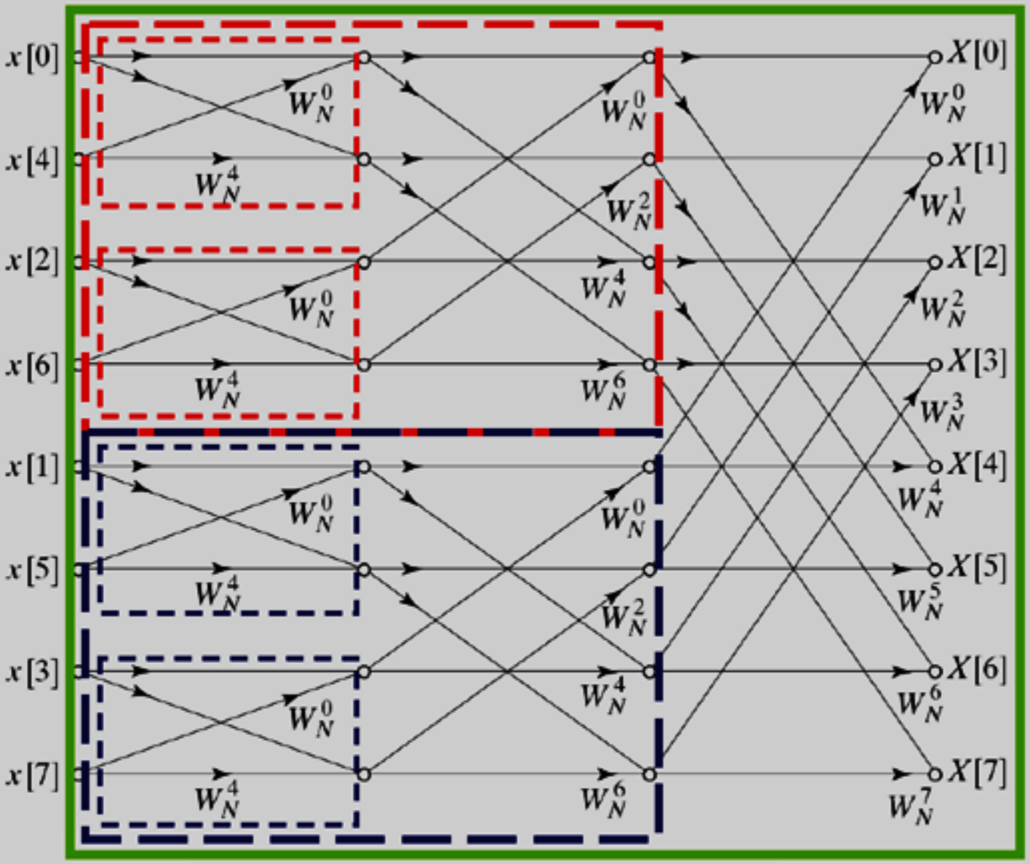


Figure 7 Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation.

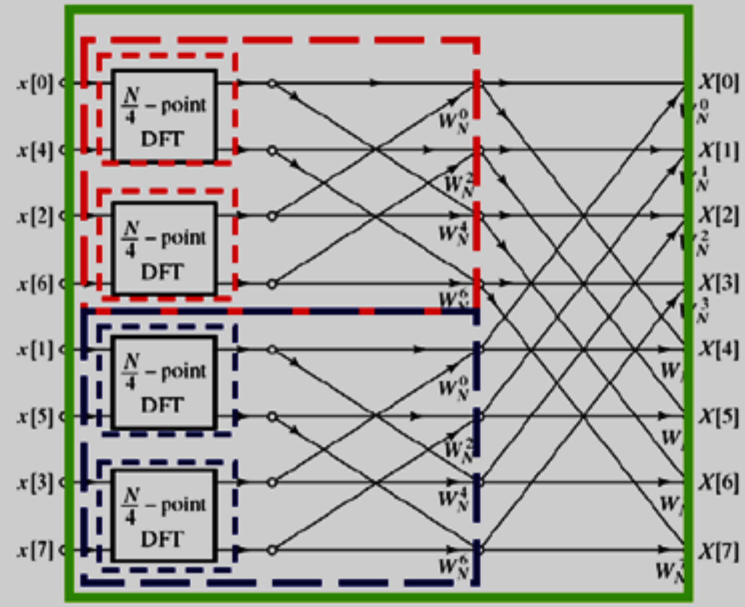
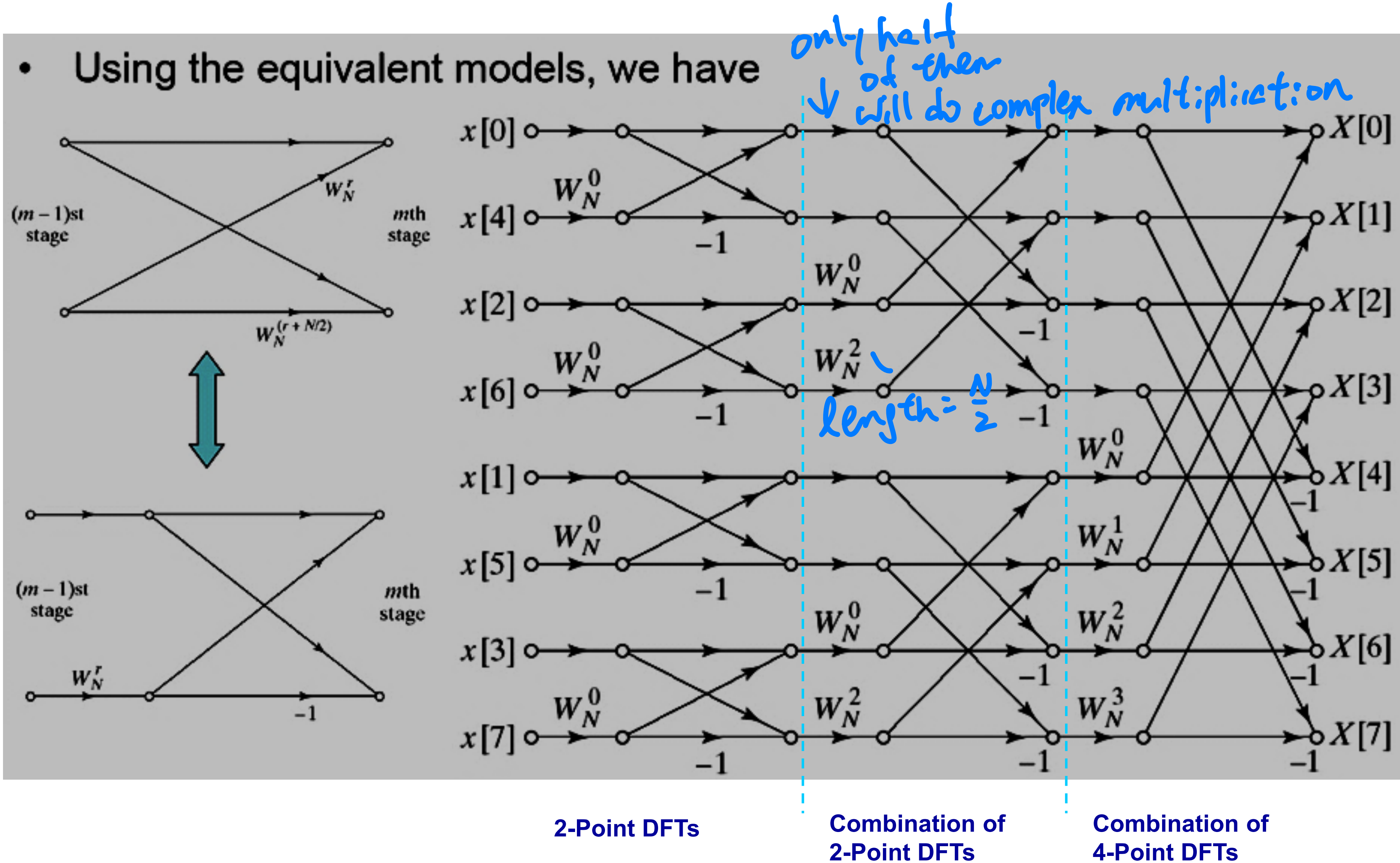


Figure 5 Result of substituting the structure of Figure 4 into Figure 3.

???

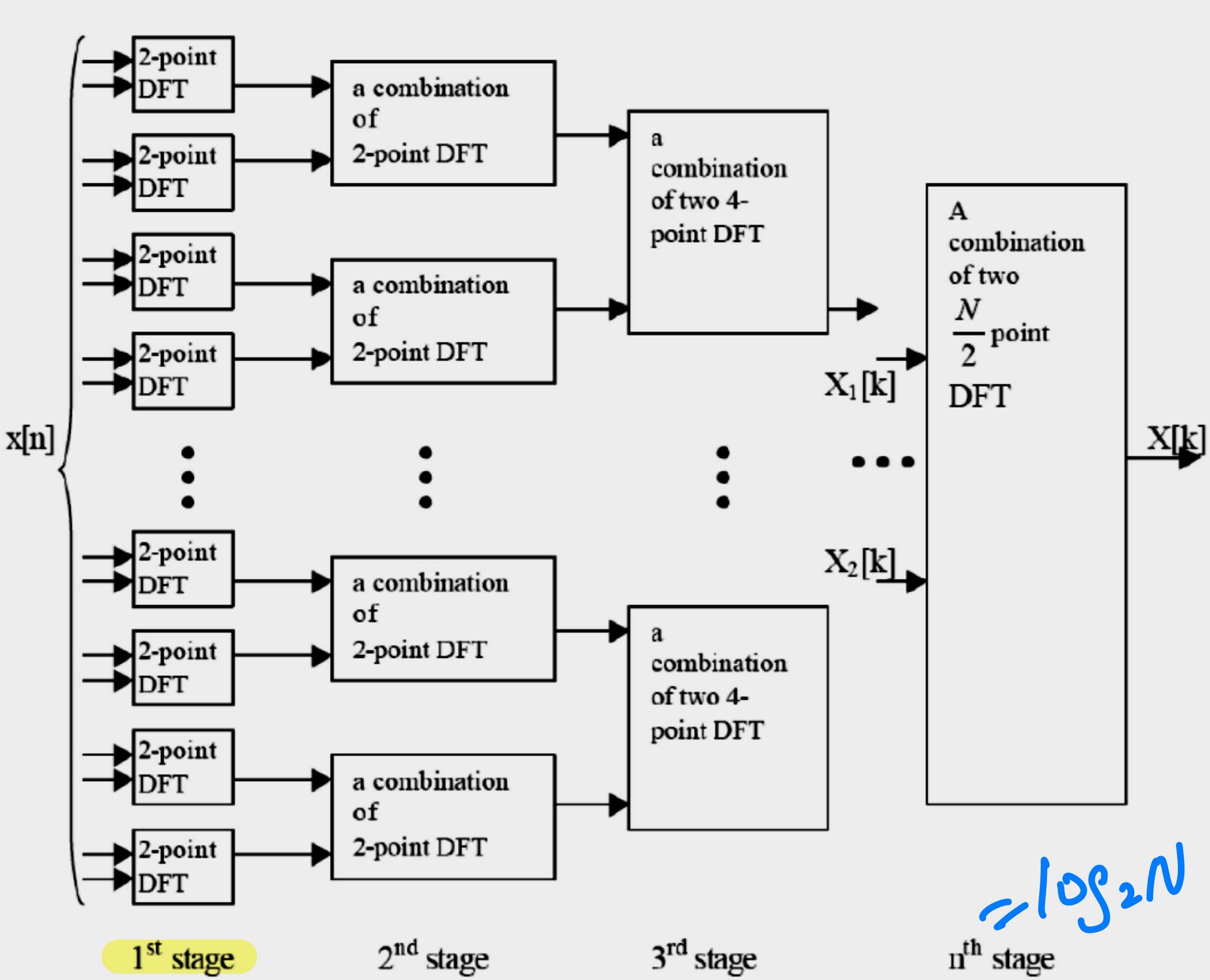
Decimation-in-Time FFT Algorithm - Example

- Using the equivalent models, we have



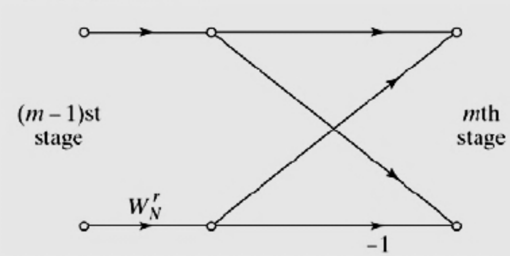
Decimation-in-Time FFT Algorithm

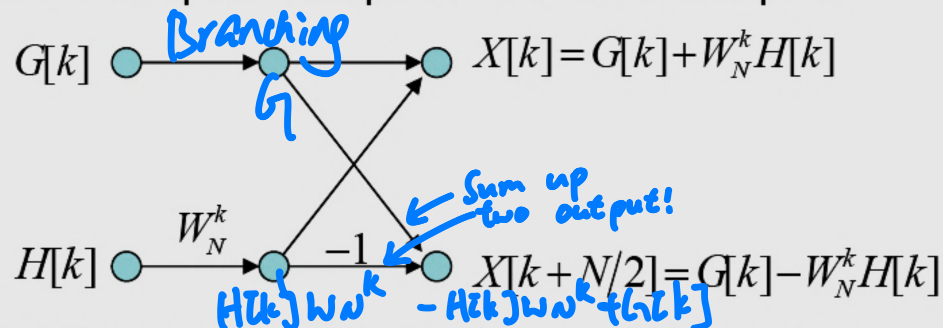
- In general, an N-point FFT (a radix-2 FFT and $N = 2^n$) is evaluated by



Decimation-in-Time FFT Algorithm

- Comparison the computation load of DIT-FFT and *N*-point DFT:
  - In the previous diagram, each stage requires (*N* / 2) butterfly computations.
  - Each butterfly computation requires 1 complex multiplications and 2 complex additions.





  - Therefore, each stage requires (*N* / 2) complex multiplications and *N* complex additions.
  - The computation load of total *n* stages needs

$$\boxed{\begin{aligned} &\text{Complex multiplications}: \quad \frac{N}{2}n = \frac{N}{2}\log_2 N \\ &\text{Complex additions}: \quad Nn = N\log_2 N \end{aligned}}$$

$= O(N \log N)$

Decimation-in-Time FFT Algorithm

	Direct computation load of DFT		FFT	
N	Complex Multiplication N^2	Complex Addition $N(N-1)$	Complex Multiplication $(N/2)\log_2 N$	Complex Addition $N\log_2 N$
2	4	2	1	2
8	64	56	12	24
32	1024	922	80	160
64	4096	4022	192	384
128	16384	16256	448	896
2^{10}	1048576	1047522	5120	10240
2^{20}	$\sim 10^{12}$	$\sim 10^{12}$	$\sim 10^{7}$	$\sim 2 \times 10^{7}$

FFT can give orders of magnitude complexity savings compared with direct evaluation of DFT !!