Discrete Fourier Transform

- Discrete Fourier transform (DFT) pairs

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

N complex multiplications
N-1 complex additions

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

$$\text{where } W_N^{-kn}=e^{-j\frac{2\pi}{N}kn}$$

- DFT/IDFT can be implemented by using the same hardware
- It requires N^2 complex multiplications and N(N-1) complex additions

Decimation in Time

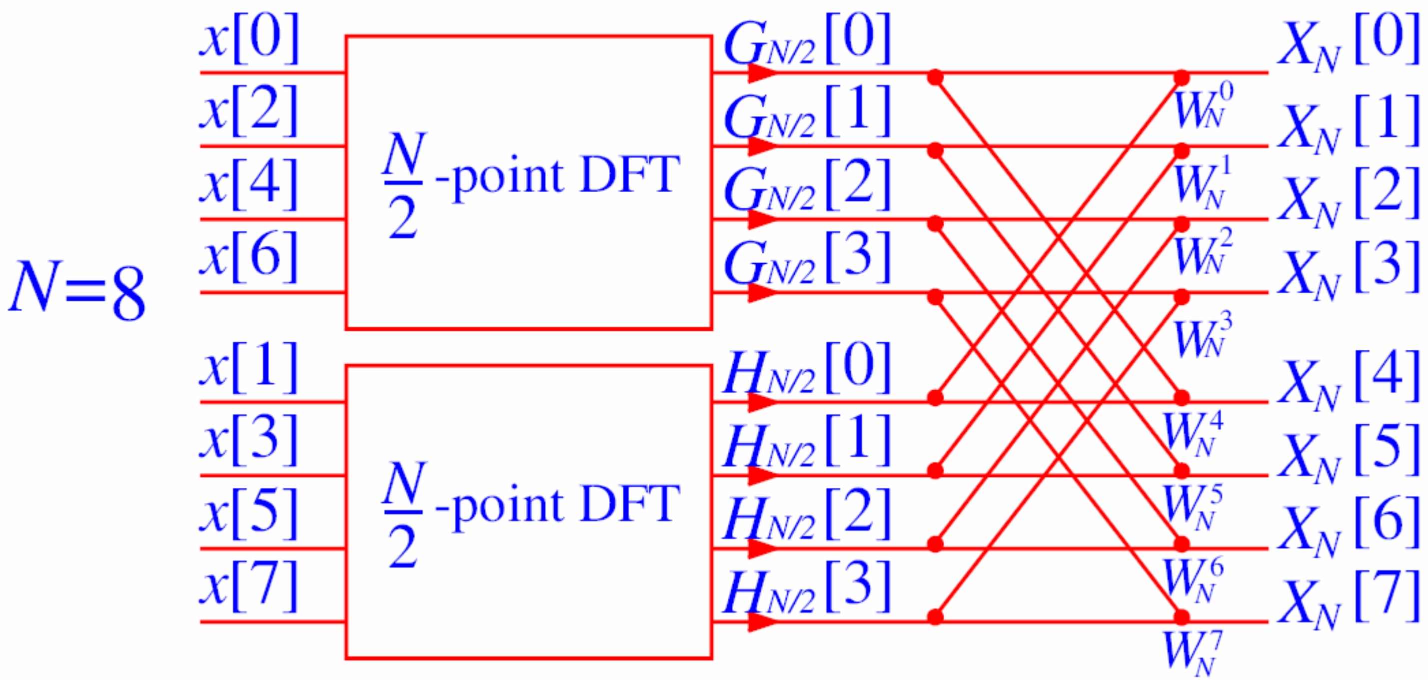
$$
\begin{aligned}
X_N[k] &= \sum_{n=0}^{N-1} x[n]\, W_N^{kn} \qquad \longleftarrow \quad N\text{-point DFT} \\
&= \sum_{\boxed{n \text{ even}}} x[n]\, W_N^{kn} + \sum_{\boxed{n \text{ odd}}} x[n]\, W_N^{kn} \\
&= \sum_{l=0}^{N/2-1} x[2l]\, W_N^{2lk} + \sum_{l=0}^{N/2-1} x[2l+1]\, W_N^{(2l+1)k} \\
&= \sum_{l=0}^{N/2-1} x[2l][\underbrace{W_N^2}_{W_{N/2}}]^{lk} + \underbrace{W_N^k}_{\text{twiddle factor}} \sum_{l=0}^{N/2-1} x[2l+1][\underbrace{W_N^2}_{W_{N/2}}]^{lk} \\
&= \underbrace{\sum_{l=0}^{N/2-1} x[2l]\, W_{N/2}^{lk} + W_N^k \sum_{l=0}^{N/2-1} x[2l+1]\, W_{N/2}^{lk}}_{\text{two } N/2\text{-point DFT's!!!}}
\end{aligned}
$$

$N+2(N/2)^2$ complex multiplications vs. N^2 complex multiplication

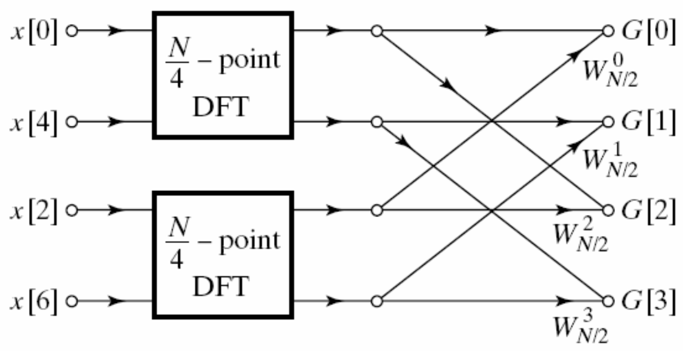
Using a briefer system of notation:

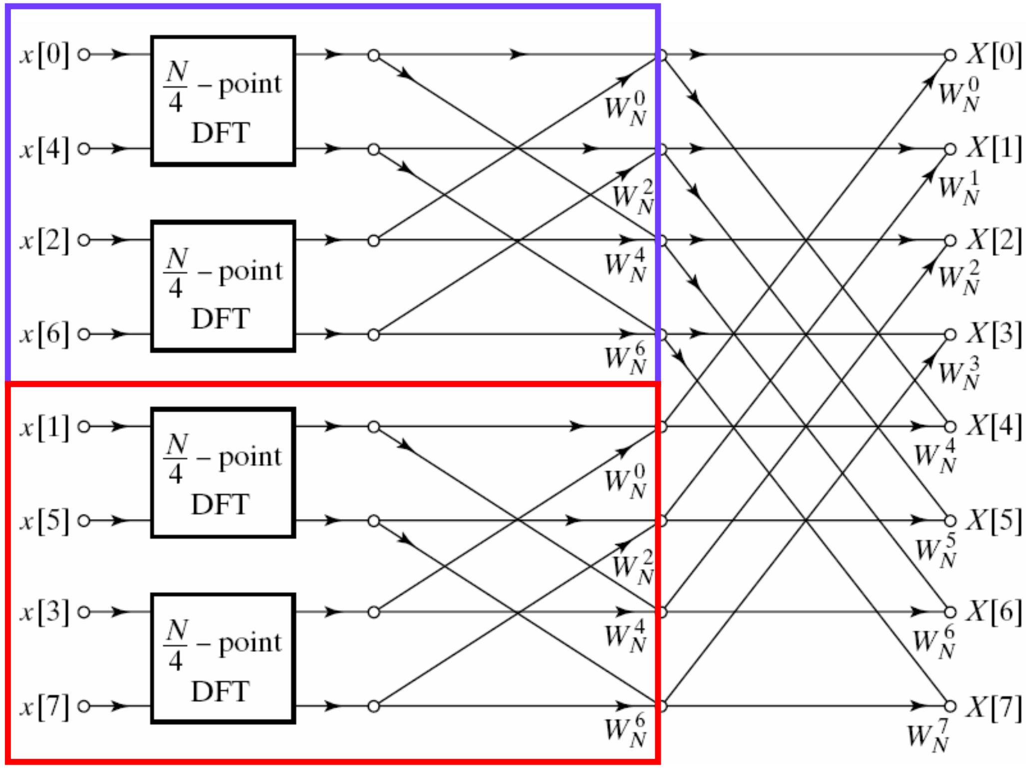
$$X_N[k] = G_{N/2}[k] + W_N^k H_{N/2}[k] \, ,$$

where $G_{N/2}[k]$ and $H_{N/2}[k]$ are the $N/2$-point DFTs involving $x[n]$ with even and odd n, respectively.



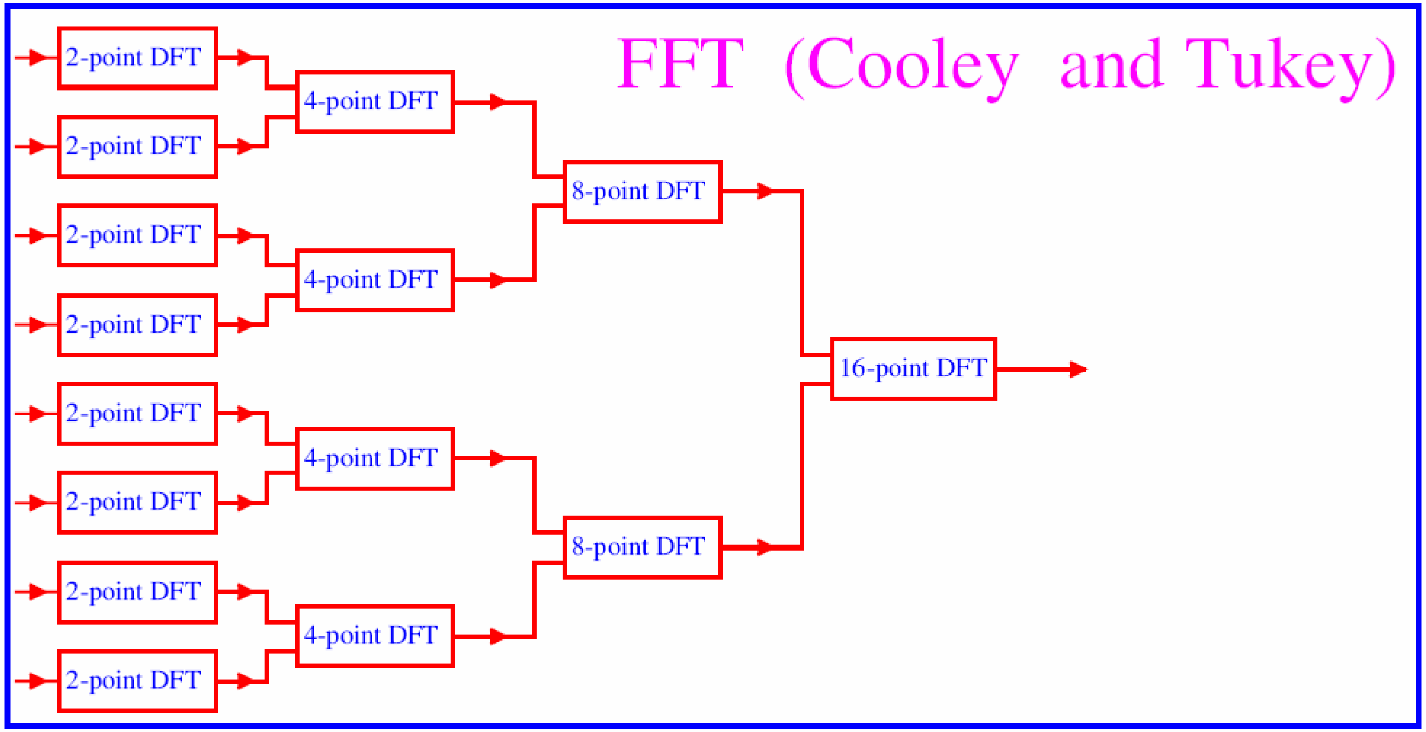
Flow Graph of the DIT FFT



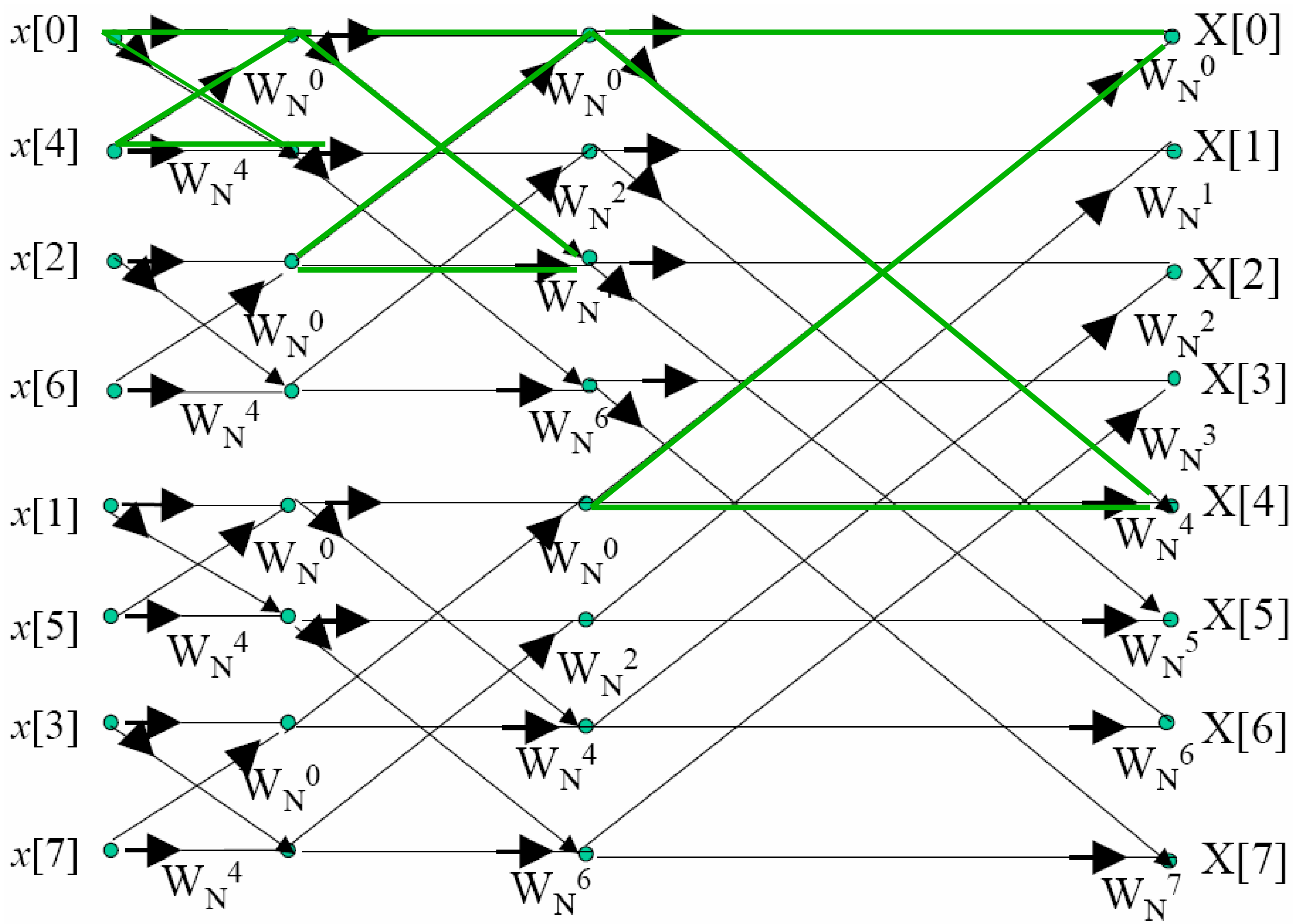


Corollary:

Any N-point DFT with even N can be computed via two $N/2$-point DFTs. In turn, if $N/2$ is even then each of these $N/2$-point DFTs can be computed via two $N/4$-point DFTs and so on. In the case of $N = 2^r$, all $N, N/2, N/4$ … are even and such a process of “splitting” ends up with all 2-point DFTs!



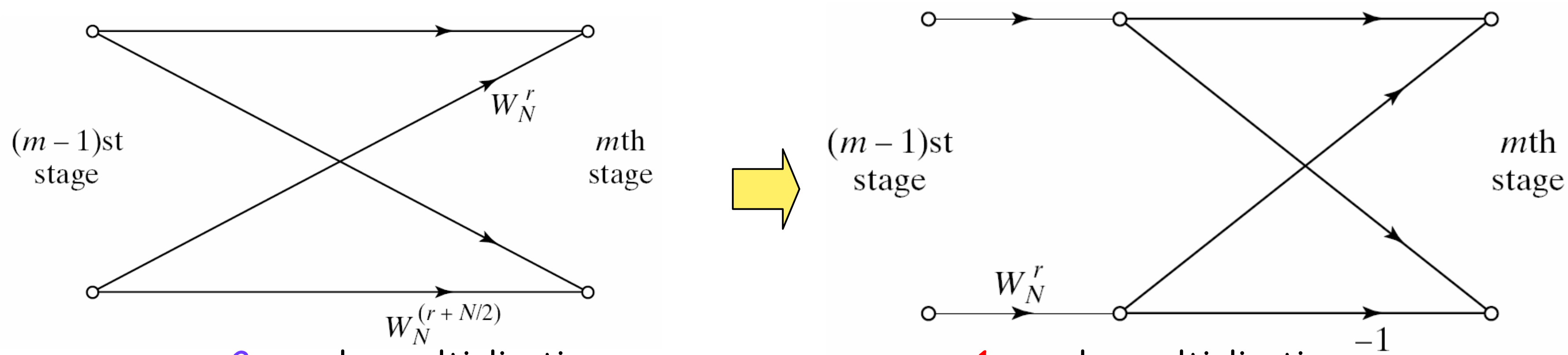
8-point DIT DFT



Remarks

- It requires $v=\log_2 N$ stages
- Each stage has N complex multiplications and N complex additions
- The number of complex multiplications (as well as additions) is equal to $N \log_2 N$
- By symmetry property, we have (butterfly operation)

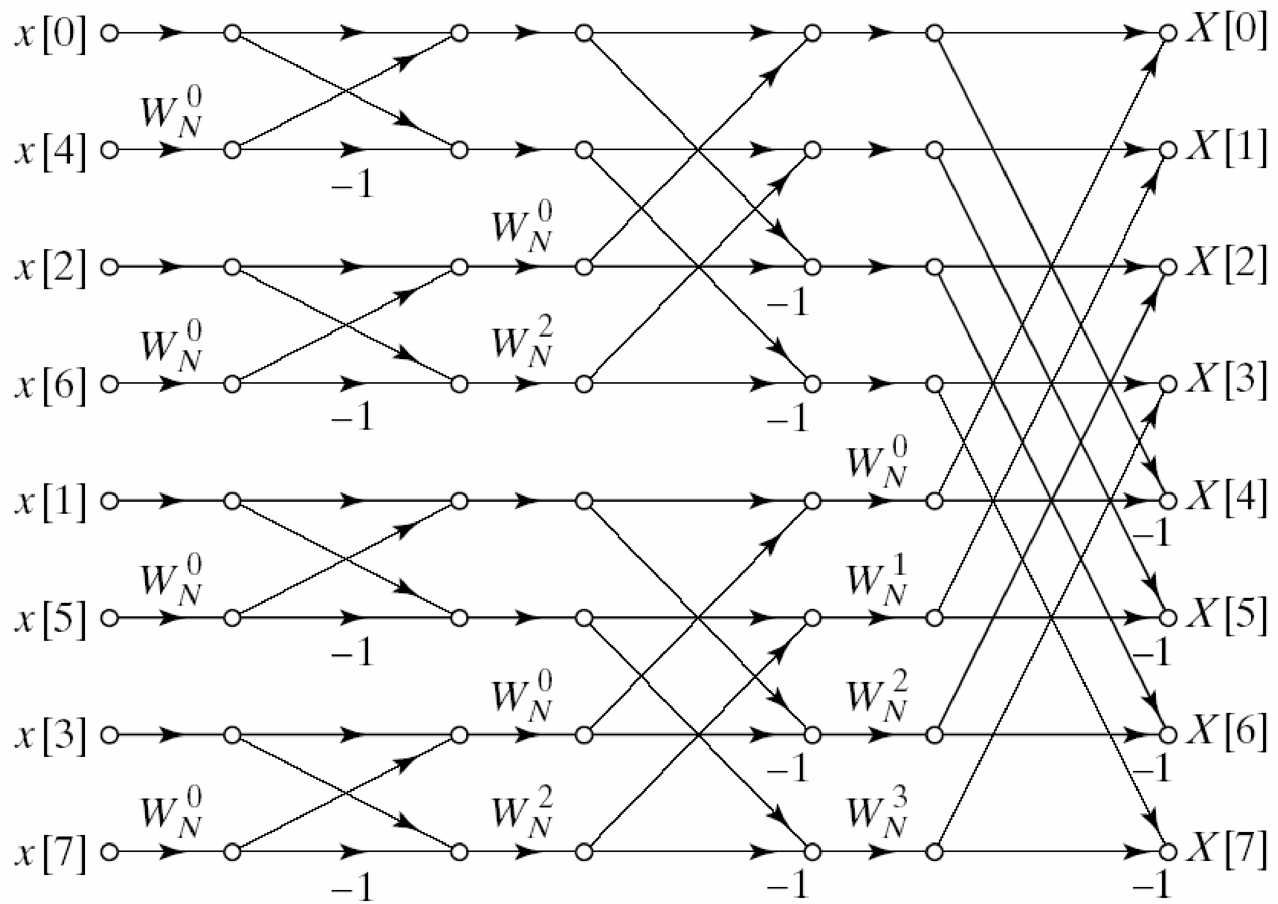
$$W_N^{r+N/2} = W_N^r W_N^{N/2} = W_N^r e^{-j\pi} = -W_N^{N/2}$$



2 complex multiplications
2 complex additions

1 complex multiplications
2 complex additions

8-point FFT

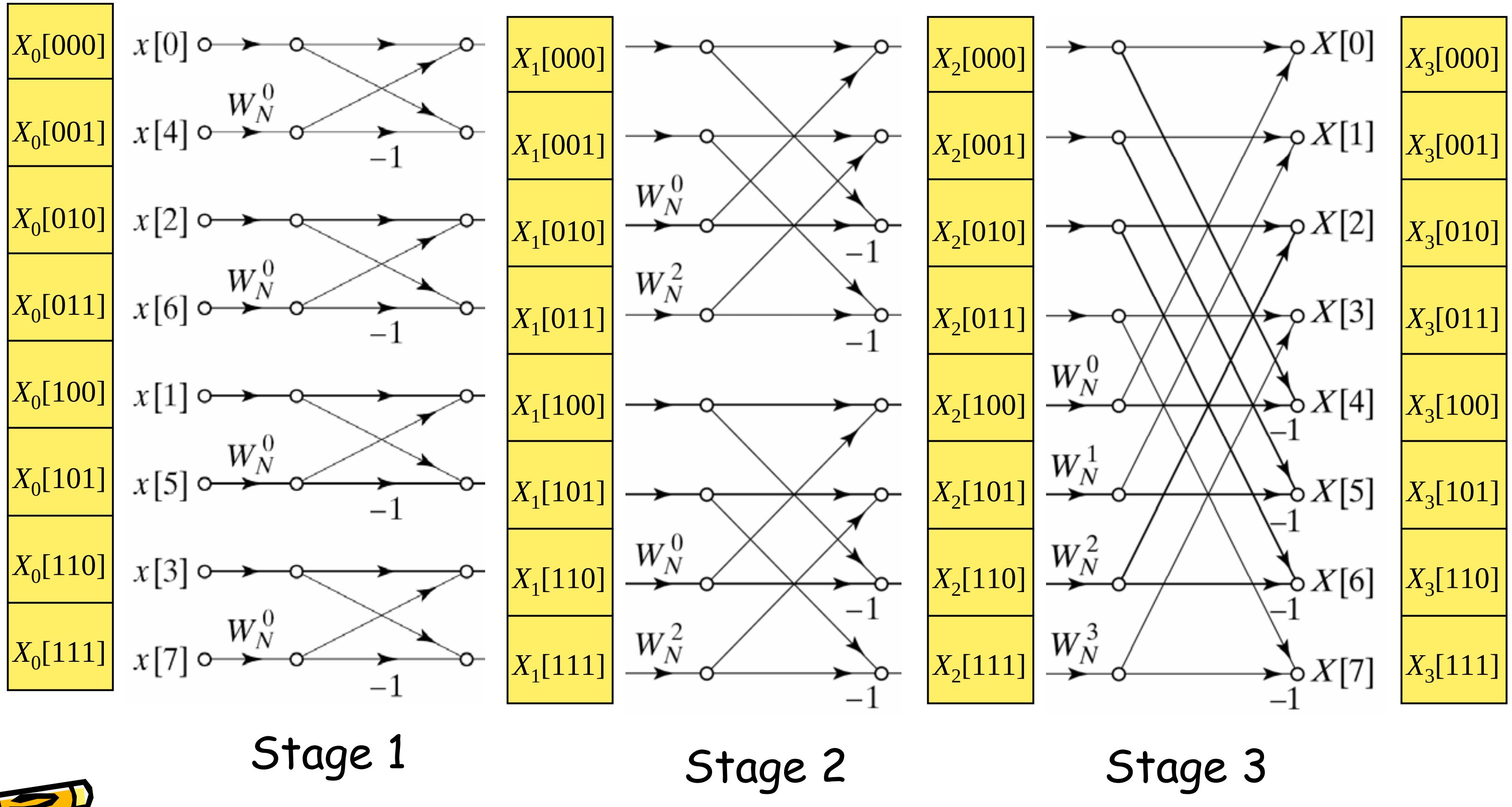


Bit-Reversed order

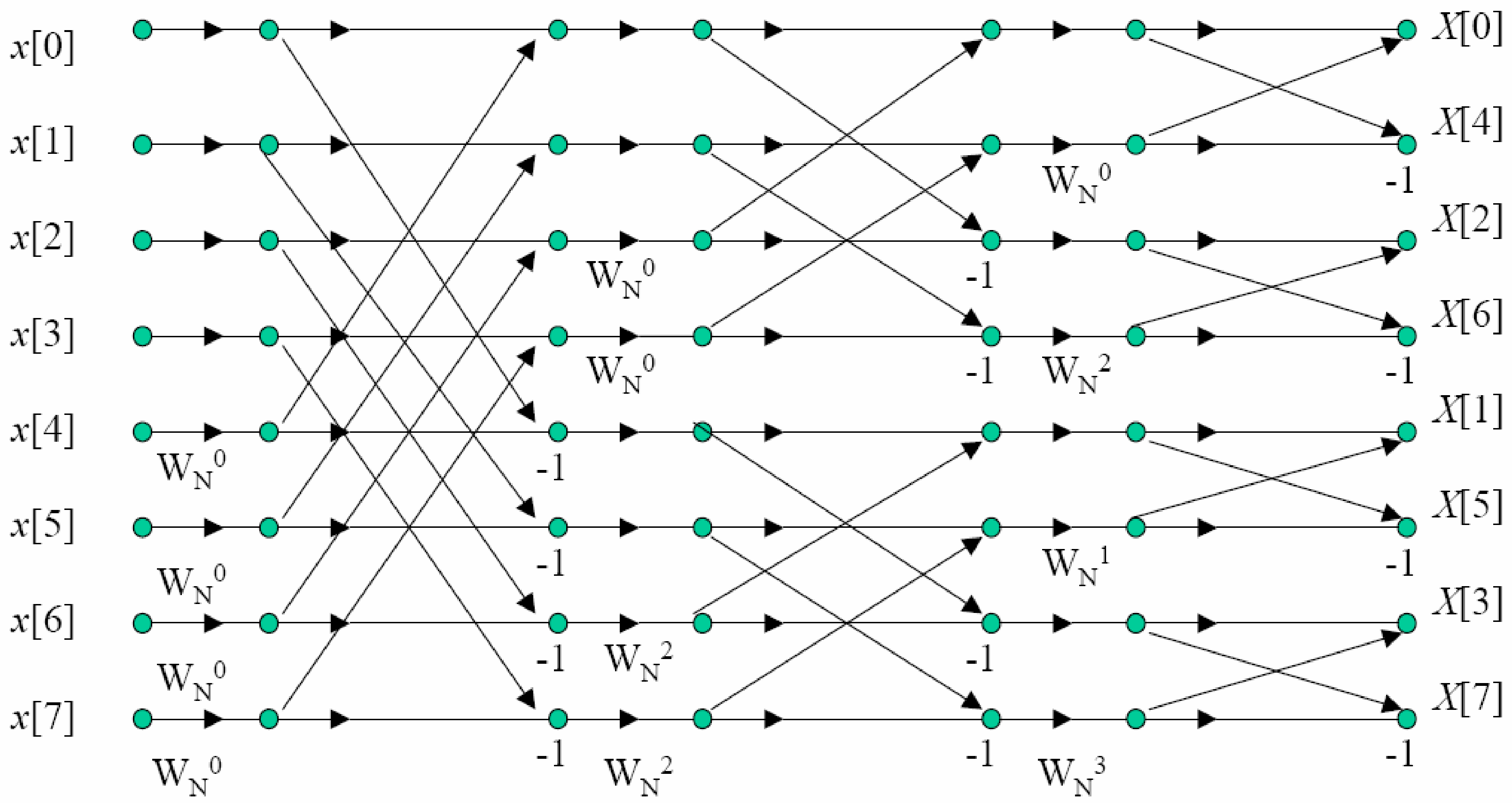
Normal order

In-Place Computation

The same register array can be used in each stage



8-point FFT

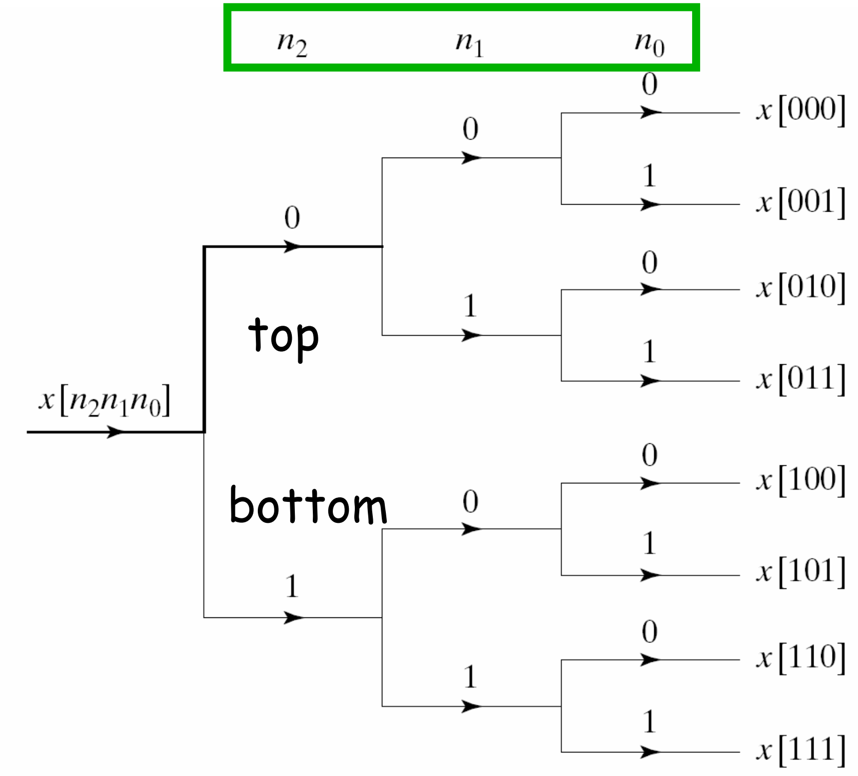


Normal order

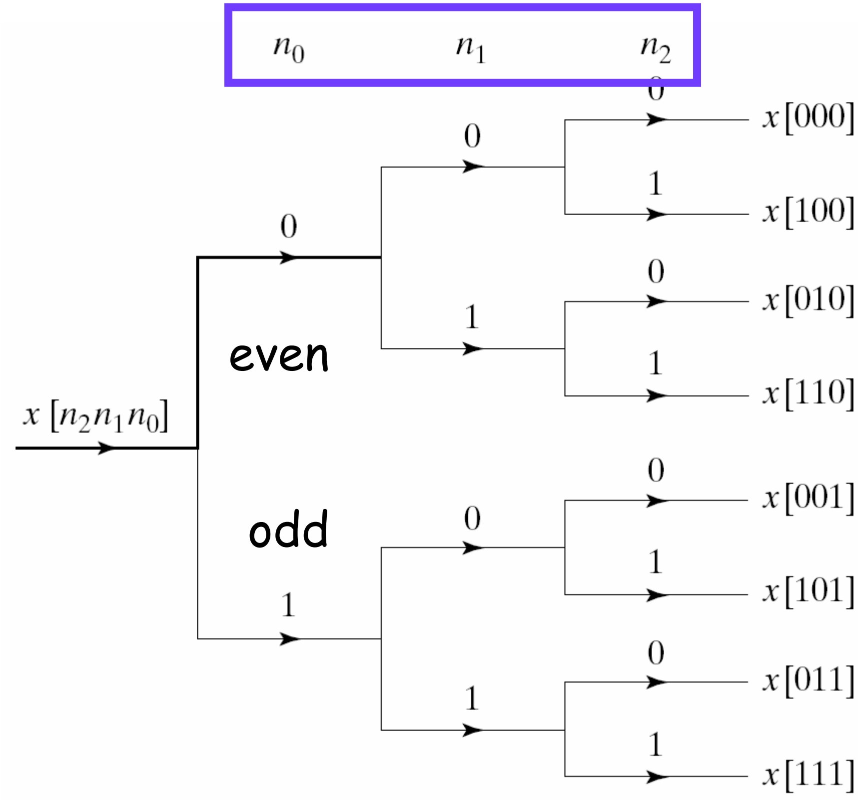
The original from given by Cooly & Tukey (DIT FFT)

Bit-reversed order

Normal-Order Sorting v.s. Bit-Reversed Sorting



Normal Order



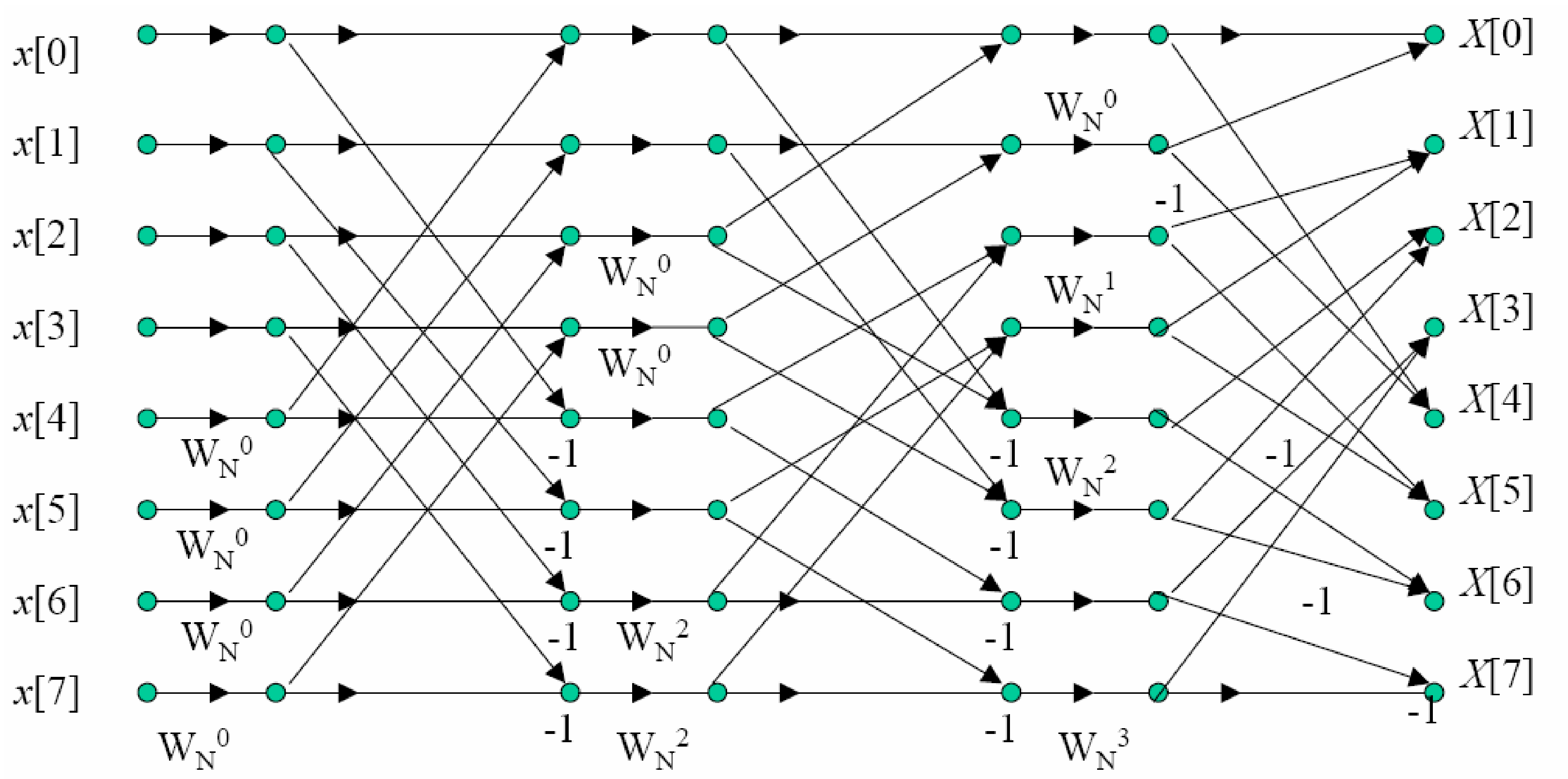
Bit-reversed Order

DFT v.s. Radix-2 FFT

- DFT: N^2 complex multiplications and $N(N-1)$ complex additions
- Recall that each butterfly operation requires one complex multiplication and two complex additions
- FFT: $(N/2)\log_2 N$ multiplications and $N\log_2 N$ complex additions

- In-place computations: the input and the output nodes for each butterfly operation are horizontally adjacent ➔ only one storage arrays will be required

Alternative Form



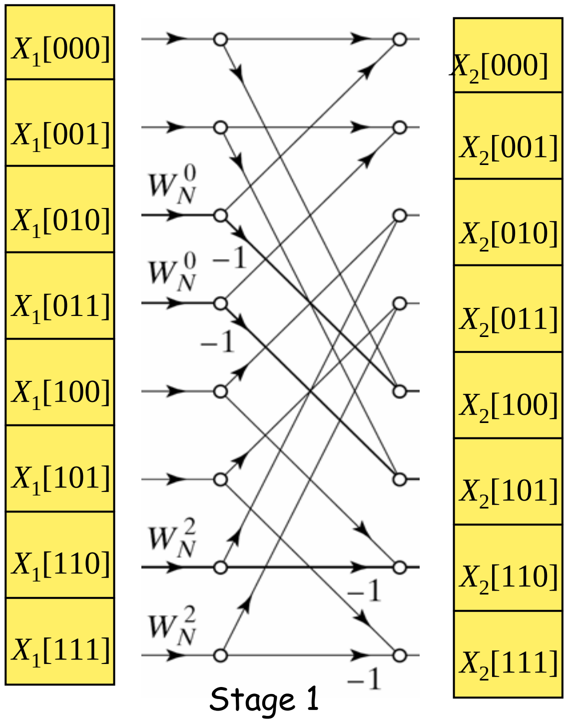
Normal order

Normal order

Two complex storage arrays are necessary !!

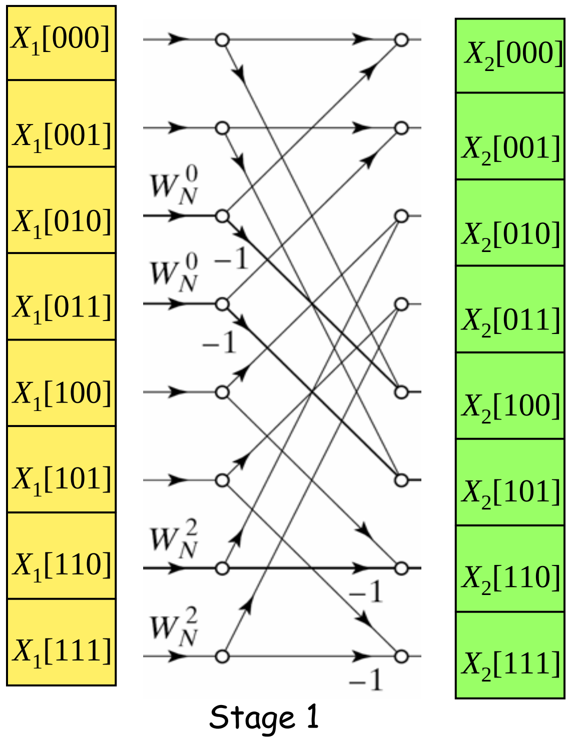
Alternative Form

Parallel processing:
4 BF units



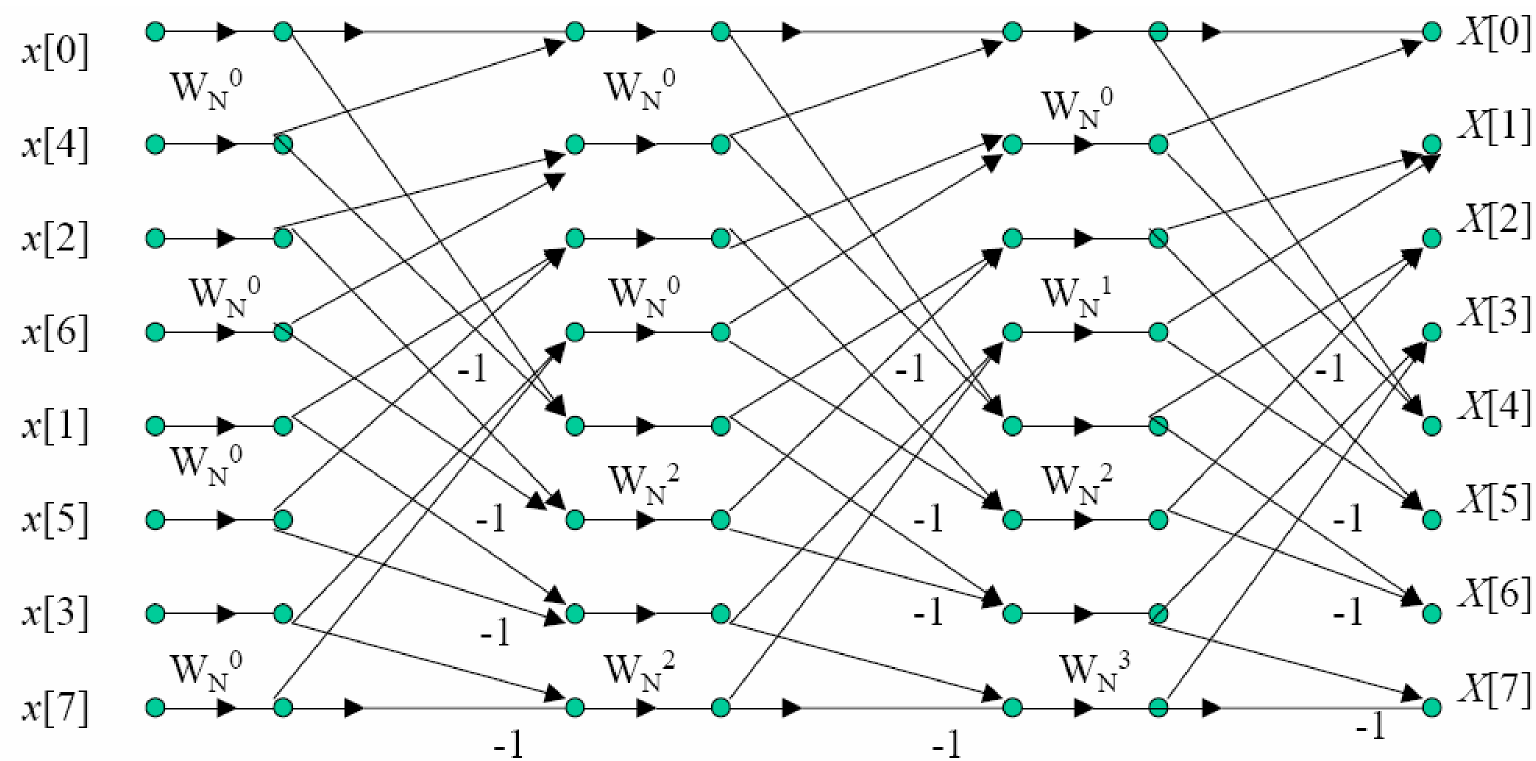
The same register array can be used

Sequential processing:
1 BF unit



Two register arrays are required

Alternative Form 2



Bit-reversed order

Normal order

The geometry of each stage is identical

Decimation in Frequency (DIF)

- Recall that the DFT is $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk},\ 0 \le k \le N-1$
- DIT FFT algorithm is based on the decomposition of the DFT computations by forming small subsequences in time domain index "n": n=2ℓ or n=2ℓ+1
- One can consider dividing the output sequence X[k], in frequency domain, into smaller subsequences: k=2r or k=2r+1:

$$X[k] \left\langle \begin{array}{l} X[2r] \\ X[2r+1] \end{array} \right. \qquad 0 \le r \le \tfrac{N}{2} - 1$$

Substitution of variables

$$X[2r] = \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_N^{2nr} + \sum_{n=\frac{N}{2}}^{N-1} x[n]\, W_N^{2nr} = \sum_{n=0}^{N/2-1} x[n]\, W_N^{2nr} + \sum_{n=0}^{N/2-1} x[n+\tfrac{N}{2}]\, W_N^{2r(n+\frac{N}{2})}$$

$$= \sum_{n=0}^{N/2-1} (x[n] + x[n+\tfrac{N}{2}]) W_{\frac{N}{2}}^{nr}$$

$$\boxed{W_N^{2r(n+\frac{N}{2})} = W_N^{2rn} W_N^{rN} = W_N^{2rn}}$$

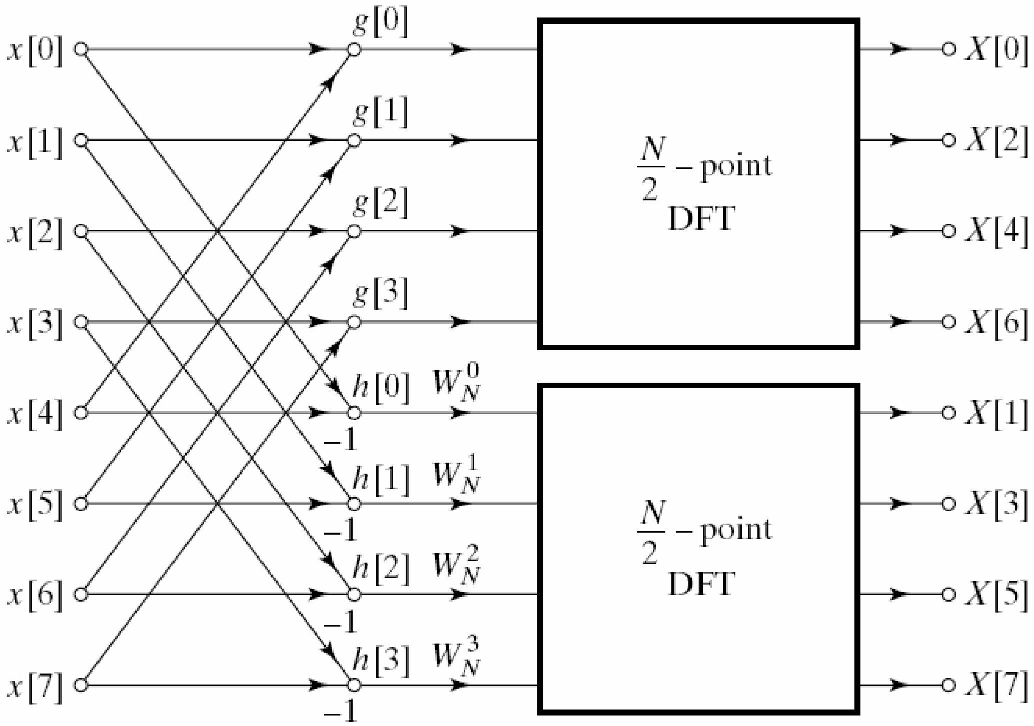
DEPT. OF ELECTRONICS ENGINEERING & INST. OF ELECTRONICS NCTU

DIF FFT Algorithm (1)

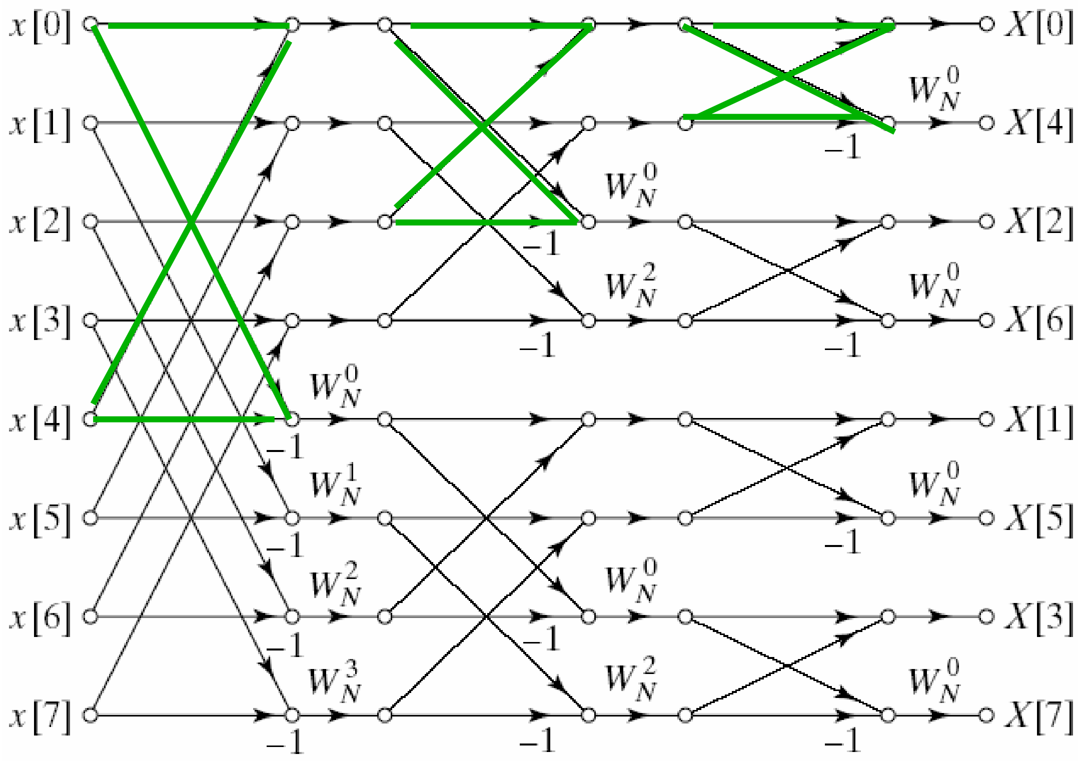
$$\boxed{0 \le r \le \frac{N}{2} - 1}$$

$$X[2r] = \sum_{n=0}^{N/2-1} \underline{(x[n] + x[n+\tfrac{N}{2}])} W_{\frac{N}{2}}^{nr}$$ is just N/2-point DFT. Similarly,

$$X[2r+1] = \sum_{n=0}^{N/2-1} (x[n] - x[n+\tfrac{N}{2}])\, W_N^{n(2r+1)} = \sum_{n=0}^{N/2-1} \underline{\{x[n] - x[n+\tfrac{N}{2}]\} W_N^n}\, W_{N/2}^{nr}$$



DIF FFT Algorithm (2)



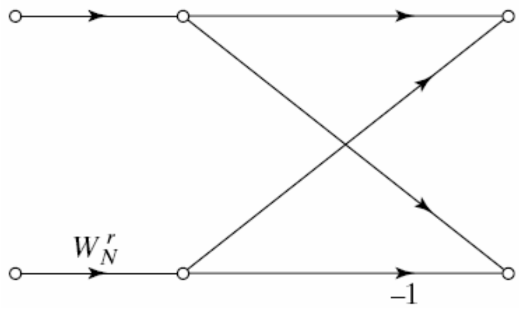
$v=\log_2 N$ stages, each stage has N/2 butterfly operation.

$(N/2)\log_2 N$ complex multiplications, N complex additions

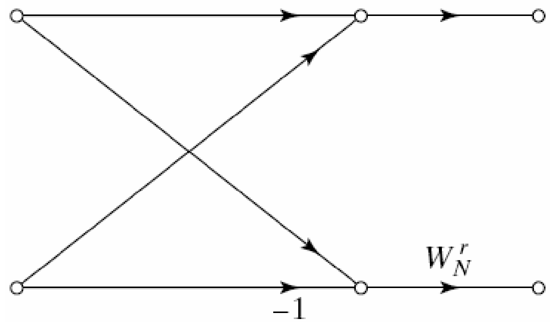
cwliu@twins.ee.nctu.edu.tw

Remarks

- The basic butterfly operations for DIT FFT and DIF FFT respectively are transposed-form pair.



DIT BF unit



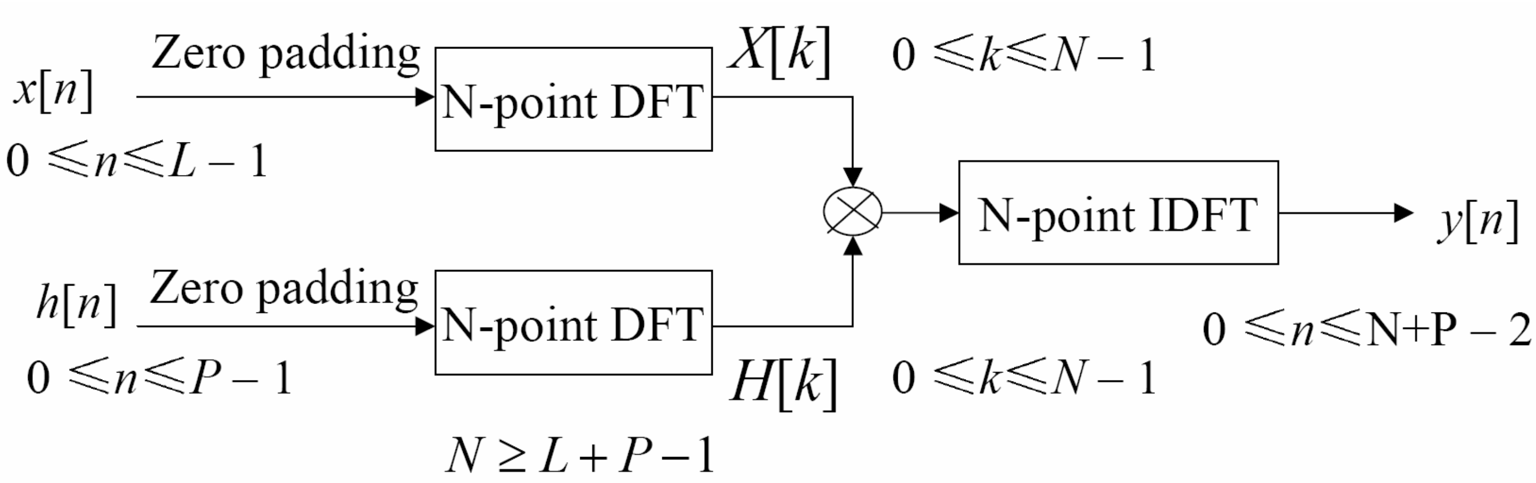
DIF BF unit

- The I/O values of DIT FFT and DIF FFT are the same
- Applying the transpose transform to each DIT FFT algorithm, one obtains DIF FFT algorithm

Implementation Issues

- Radix-2, Radix-4, Radix-8, Split-Radix,Radix-2^2, …,
- I/O Indexing
- In-place computation
  - Bit-reversed sorting is necessary
  - Efficient use of memory
  - Random access (not sequential) of memory. An address generator unit is required.
  - Good for cascade form: FFT followed by IFFT (or vice versa)
    - E.g. fast convolution algorithm
- Twiddle factors
  - Look up table
  - CORDIC rotator

Recall… Linear Convolution



Fast Convolution with the FFT

- Given two sequences x_1 and x_2 of length N_1 and N_2 respectively
  - Direct implementation requires N_1N_2 complex multiplications
- Consider using FFT to convolve two sequences:
  - Pick N, a power of 2, such that $N \geq N_1+N_2-1$
  - Zero-pad x_1 and x_2 to length N
  - Compute N-point FFTs of zero-padded x_1 and x_2, one obtains X_1 and X_2
  - Multiply X_1 and X_2
  - Apply the IFFT to obtain the convolution sum of x_1 and x_2
  - Computation complexity: $2(N/2)\log_2 N + N + (N/2)\log_2 N$

Other Fast Algorithm for DFT

- Goertzel Algorithm
  - By reformulating DFT as a convolution
  - it is not restricted to computation of the DFT but any desired set of samples of the Fourier transform of a sequence
- Winograd Algorithm
  - An efficient algorithm for computing short convolutions
  - The number of multiplication complexity is of order O(N), however the number of addition complexity is significantly increased.
- Chirp Transform Algorithm