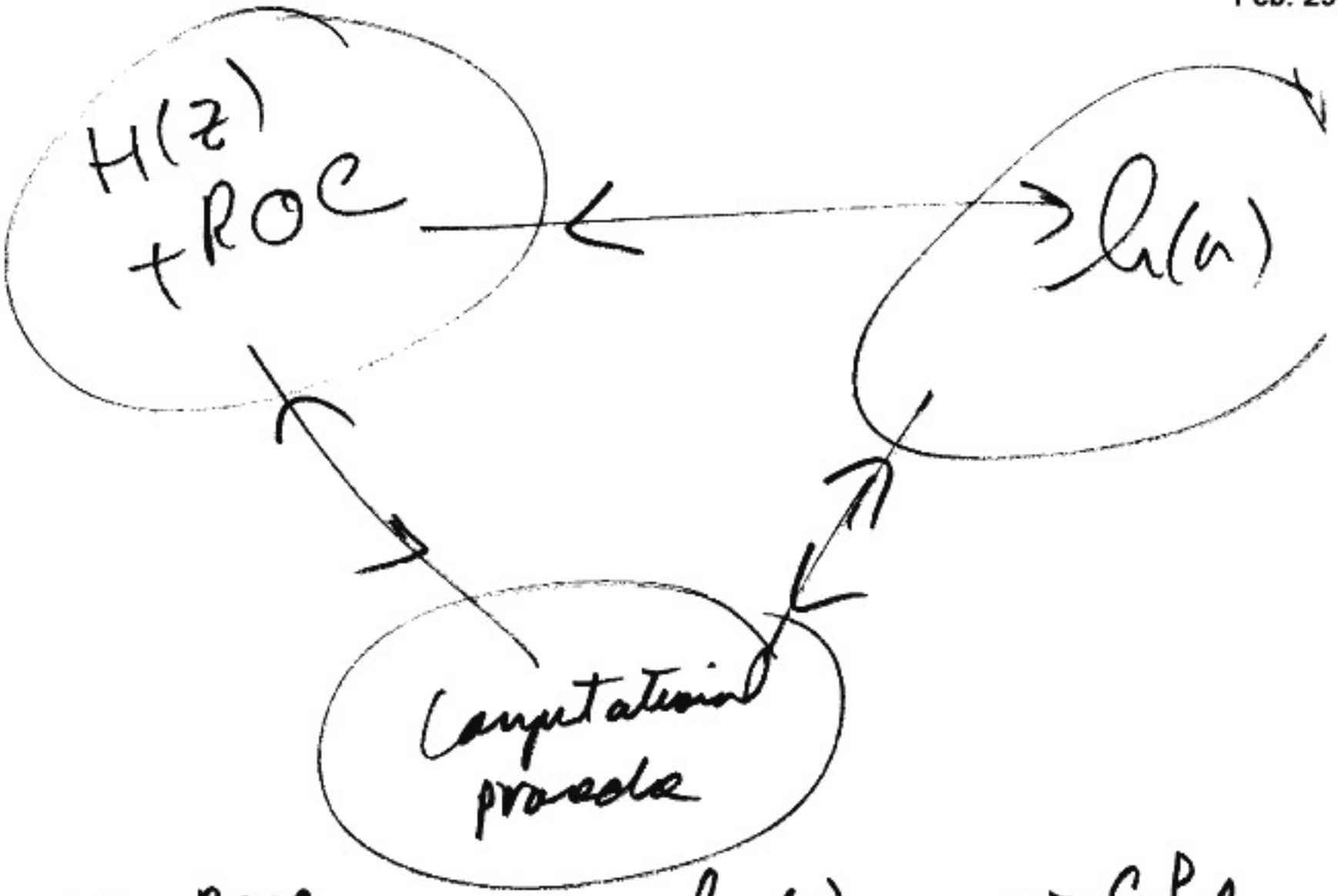


$H(z) + ROC1 \longrightarrow h_1(n) \longrightarrow C.P.1$

$H(z) + ROC2 \longrightarrow h_2(n) \longrightarrow CP.2.$

2.D. D.F.T

$x(n) \longrightarrow \boxed{\text{D.T.F.T.}} \longrightarrow X(\omega) = \sum_n x(n) e^{-j\omega n}$

discrete

continous variable real.

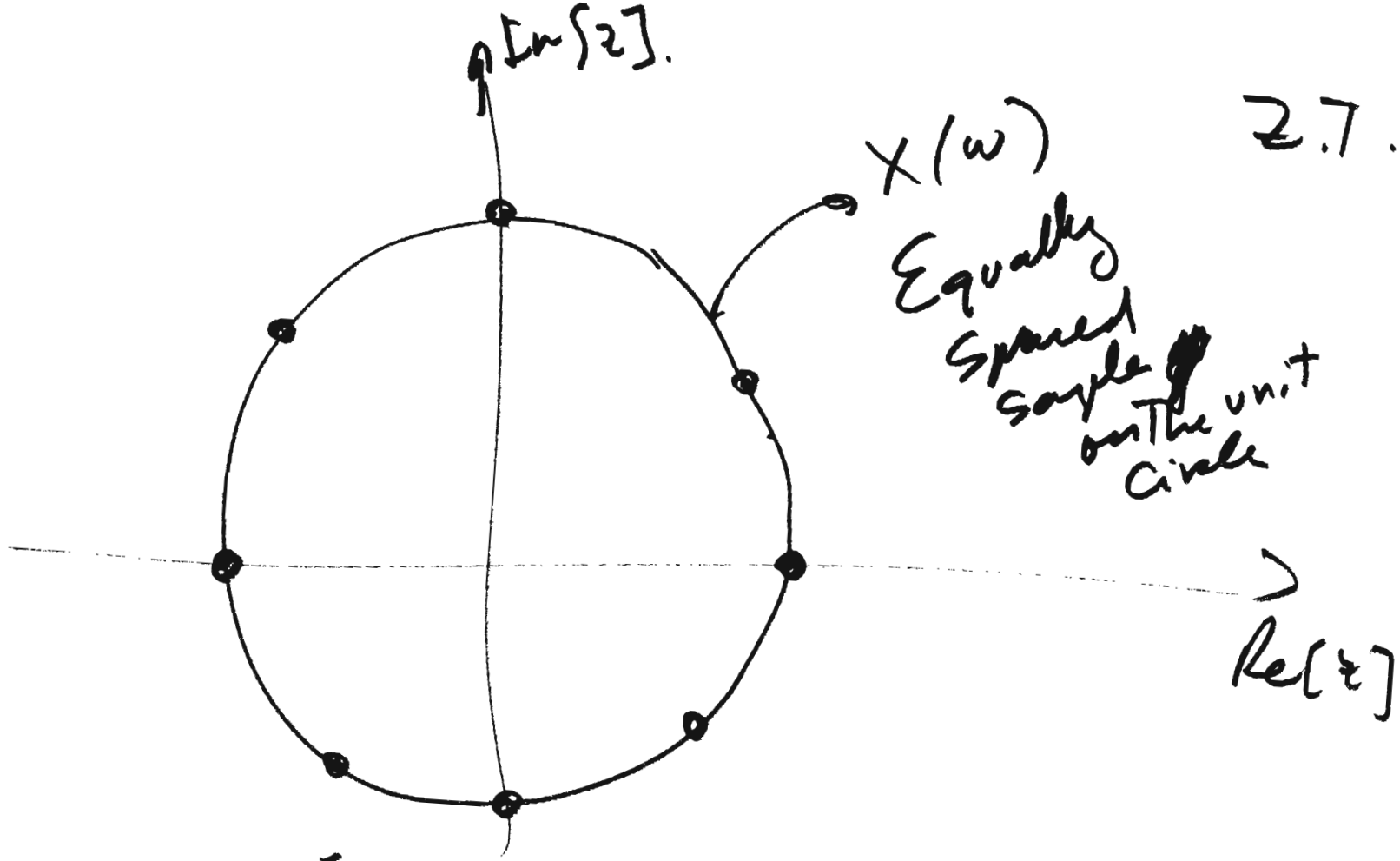
$x(n) \longrightarrow \boxed{\text{Z.T.}} \longrightarrow X(z) = \sum x(n) z^{-n}$

cont. variable complex.

$x(n) \longrightarrow \boxed{\text{DFT}} \longrightarrow X(k) = \sum_n x(n) e^{-j\frac{2\pi nk}{N}}$

integer discrete value

$\boxed{\text{FFT}}$ = Fast Fourier Transform



Z.T.

$$X(k) = \left[X(\omega) \right]_{\omega = \frac{2\pi k}{N}}$$

2.D. DFT

$N_1 \times N_2$

$$\mathrm{DFT}\{x(n_1,n_2)\} = \begin{cases} \displaystyle\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1,n_2)\, e^{-j2\pi\frac{n_1k_1}{N_1}}\, e^{-j2\pi\frac{n_2k_2}{N_2}} & 0 \le k_1 < N_1,\ 0 \le k_2 < N_2 \\ 0 & \text{otherwise} \end{cases}$$

$X(k_1,k_2)$

$$x(n_1,n_2) = \mathrm{IDFT}\{X(k_1,k_2)\} = \begin{cases} \displaystyle\frac{1}{N_1N_2}\sum\sum X(k_1,k_2)\, e^{j2\pi\frac{k_1n_1}{N_1}}\, e^{j2\pi\frac{k_2n_2}{N_2}} & 0 \le n_1 < N_1,\ 0 \le n_2 < N_2 \\ 0 & \text{otherwise} \end{cases}$$

Use of DFT in Convolution

$$x(n) \longrightarrow \boxed{h(n)} \longrightarrow y(n)$$

N pt

M point

$N+M-1$

Choose $P \geq N+M-1$

① Compute P point DFT of input $\longrightarrow X(K)$

② " " " " " h $\longrightarrow H(K)$

③ $X(K)\,H(K) = Y(K) = \text{DFT}\{y(n)\}$

④ $\text{IDFT}\{Y(K)\} = y(n)$

All of this applies in <u>2D</u>

How To Compute 2D DFT

3 ways.

1. Direct method
2. Row/Col. decomposition
3. Vector-Radix ← True FFT

Factors affect speed

- # of arithmetic ops
- complexity of program
  - memory for executible → footprint
  - memory usage as program runs.
- language / platform.

Direct Method

$N_1 = N_2 = N$

① For each (k_1, k_2) $\longrightarrow$ N^2 mult, N^2 adds. $\longrightarrow$

② how many (k_1, k_2)? N^2.

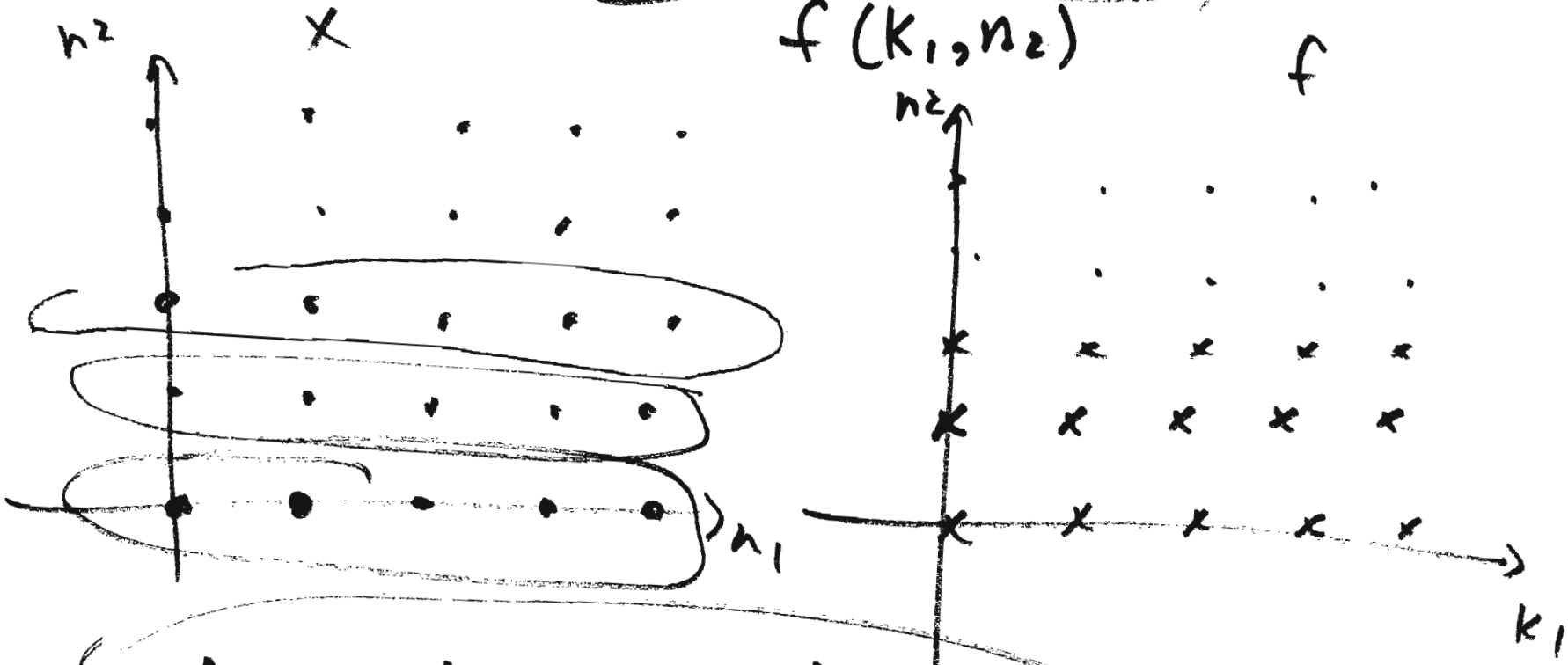
N^4 mult

N^4 adds

Row/

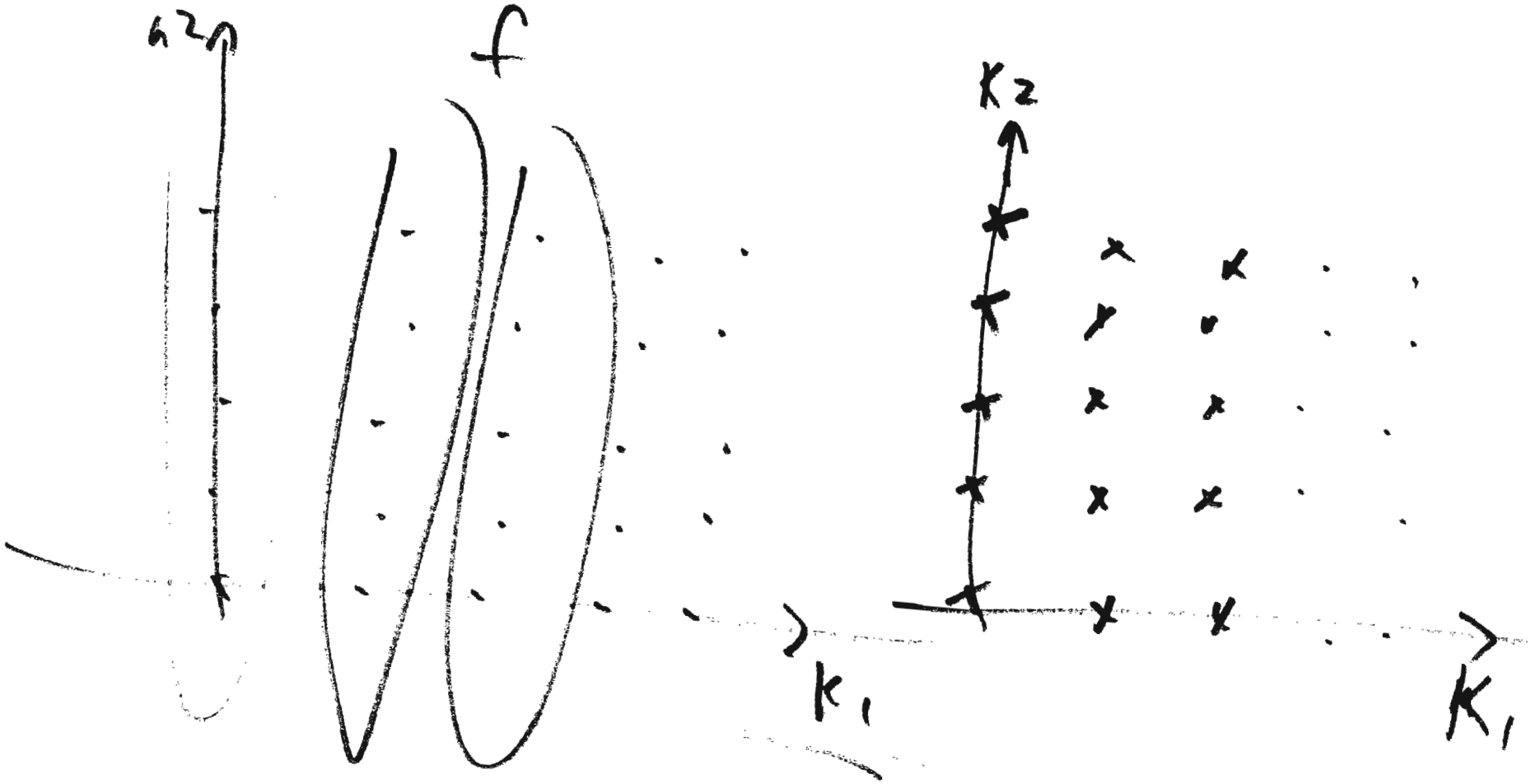
Row Column Decomposition

$$X(k_1,k_2) = \sum_{n_2=0}^{N-1} \underbrace{\left(\sum_{n_1=0}^{N-1} x(n_1,n_2)\, e^{-j\frac{2\pi n_1 k_1}{N}} \right)}_{f(k_1,n_2)} e^{-j\frac{2\pi n_2 k_2}{N}}$$



N - 1-D DFT of rows

each 1-D DFT is N point



N — 1D DFT.

each 1D DFT was N pt

1-D DFT $\longrightarrow$ FFT

$N \log_2 N$ mult

$\frac{N}{2} \log_2 N$ adds

R/C decomp $N^2 \log_2 N$ operations

LOT lower direct method

Computing DFT

$N \times N$ image.

Direct $\longrightarrow$ N^4 operations.

Row/col $\longrightarrow$ $N^2 \log_2 N$ operations

$N = 1000$

$$\left.\begin{array}{l} \text{Direct} \sim 10^{12} \text{ op.} \\ \text{Row/col} \approx 10^{6} \times 10 \approx 10^{7} \end{array}\right\} \Longrightarrow \frac{10^{12}}{10^{7}} \approx \underline{10^{5}}$$

$\cancel{\frac{10^{12}}{60} \text{ sec} \approx \frac{10^{3}}{60} \text{ min}}$ = 30 hours $\longrightarrow$ $1\frac{1}{4}$ days.

1 sec $\longrightarrow$

- Row/Col decomposition requires a matrix Transpose?

How do we do matrix Transpose in an I/O efficient way?

Eklunds Alg for Transposition

Recursive alg:

$N \times N \longrightarrow \log_2 N$ stages.

- Fast non I/O intensive way of doing matrix Transpose.

- Divide + Conquer..

Observation

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$N \times N$

$$A^T = \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}$$

simple example $A = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$ $A^T = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{bmatrix}$

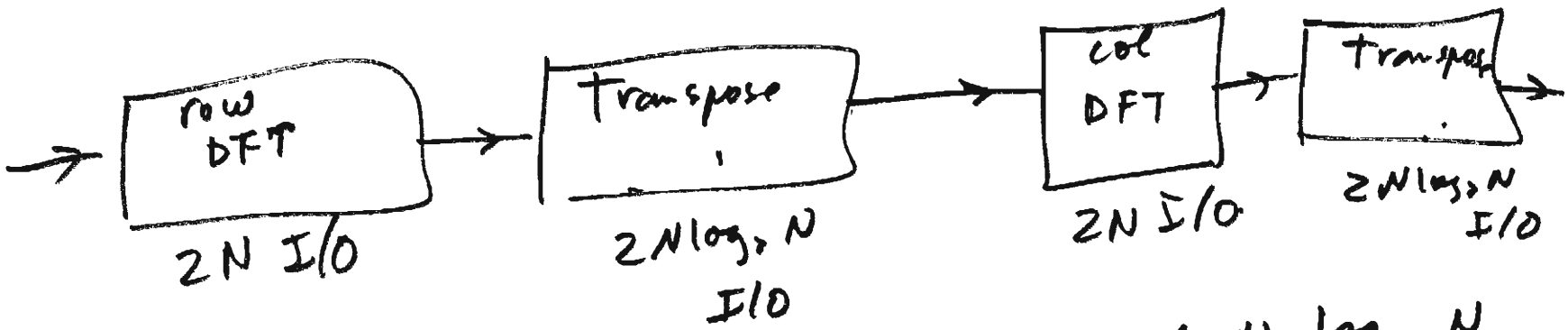
# of stages of decomposition $\log_2 N$ stage.

Each stage $2N$ I/O operation.

Example: first stage. : 4 I/O per row, $N/2$ rows $\rightarrow 4 \times \frac{N}{2} = 2N$ I/O per stage

total I/O cost : $2N \log_2 N$ stages.

Excludes Alg for matrix Transpo.



total I/O: $4N + 4N \log_2 N \approx 4N \log_2 N$.

see Fig 3.20 + 3.21 of J. lin.

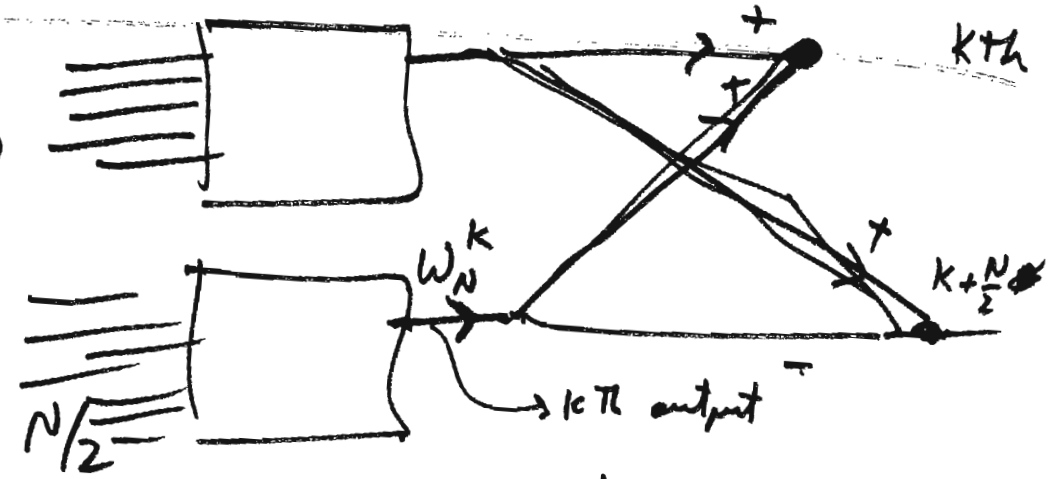
FFT

- Direct
- Row/Col.
- Vector Radix FFT. (True Translation of Divide + Conquer in 2D).

Recall 1-D.

$x(n)$ N pt. → even points $x_e(n)$ N/2.

→ odd points $x_o(n)$ N/2



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

Q how is DFT of g^x related to DFT of g_{eo}, g_{ee} g_{oe} g_{oo} ?

Show Fig 3.23, 3.24, 3.25 J.S. Lim

$$X(k_1,k_2)_{N\times N} = G_{ee}(k_1,k_2) + G_{eo}(k_1,k_2)\, e^{-j\frac{2\pi k_2}{N}} + G_{oe}(k_1,k_2)\, e^{-j\frac{2\pi k_1}{N}} + G_{oo}(k_1,k_2)\, e^{-j\frac{2\pi k_1}{N}}\, e^{-j\frac{2\pi k_2}{N}}$$

G_{ee}	:	$N/2 \times N/2$	pT	DFT	of	g_{ee}
G_{oe}		"	"	"	"	"	g_{oe}
G_{eo}		"	"	"	"	"	g_{eo}
G_{oo}		"	"	"	"	"	g_{oo}

$N \times N \longrightarrow 4 \quad \frac{N}{2} \times N/2.$

$N/2 \times N/2 \longrightarrow 4 \quad N/4 \times N/4$

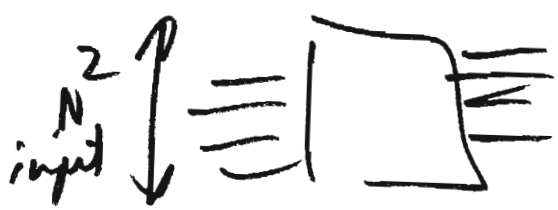
$N/4 \times N/4 \longrightarrow 4 \quad N/8 \times N/8$

$\vdots$

# of stages $N \times N$ array. $\log_2 N$

Each stage, how many butterflies:

each butterfly
4 input → 4 output.



$\Longrightarrow \frac{N^2}{4}$ butterflies/stage.

- how much computation per butterfly is needed.

12 adds
3 mult per butterfly

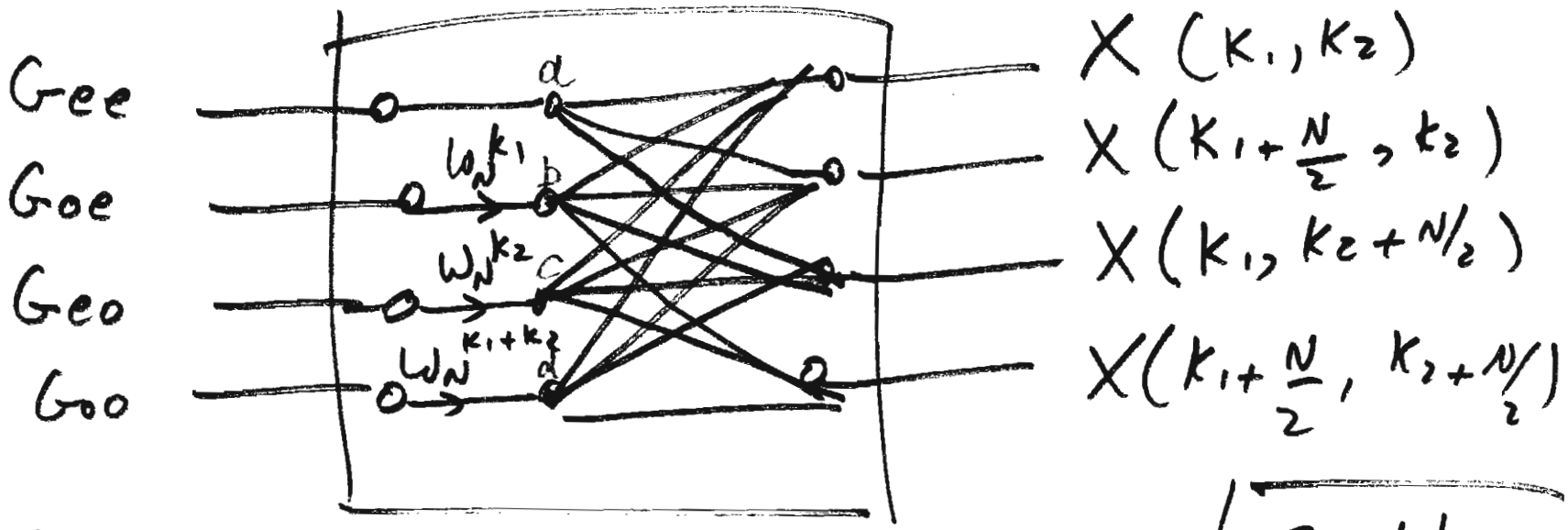
---

adds : $\frac{N^2}{4} \log_2 N \cdot 12 \cong 3N^2 \log_2 N$

mult $\frac{N^2}{4} \log_2 N \cdot 3 \cong \frac{3}{4} N^2 \log_2 N$

---

adds in Butterfly cargo. $12 \longrightarrow 8$



$\Longrightarrow$ $\boxed{8 \text{ adds}}$

Define new intermediate variables.

$$\boxed{\begin{aligned} A &= a+b \\ B &= a-b \\ C &= c+d \\ D &= c-d \end{aligned}}$$

then

$$\begin{aligned} X(k_1, k_2) &= A + C \\ X(k_1+\tfrac{N}{2}, k_2) &= B + D \\ X(k_1, \tfrac{N}{2}+k_2) &= A - C \\ X(k_1+\tfrac{N}{2}, k_2+\tfrac{N}{2}) &= B - D \end{aligned}$$

$\rightarrow$ adds. $12 \rightarrow 8$

$$\frac{N^2}{4} \cdot \log_2 N \cdot 8 = 2N^2 \log_2 N.$$

adds.

	# of mult	# of adds
Row/Col.	$N^2 \log_2 N$	$2N^2 \log_2 N$
Vector Radix	$\frac{3}{4} N^2 \log_2 N.$	$2N^2 \log_2 N$
Pined	~~N^4~~	~~N^4~~

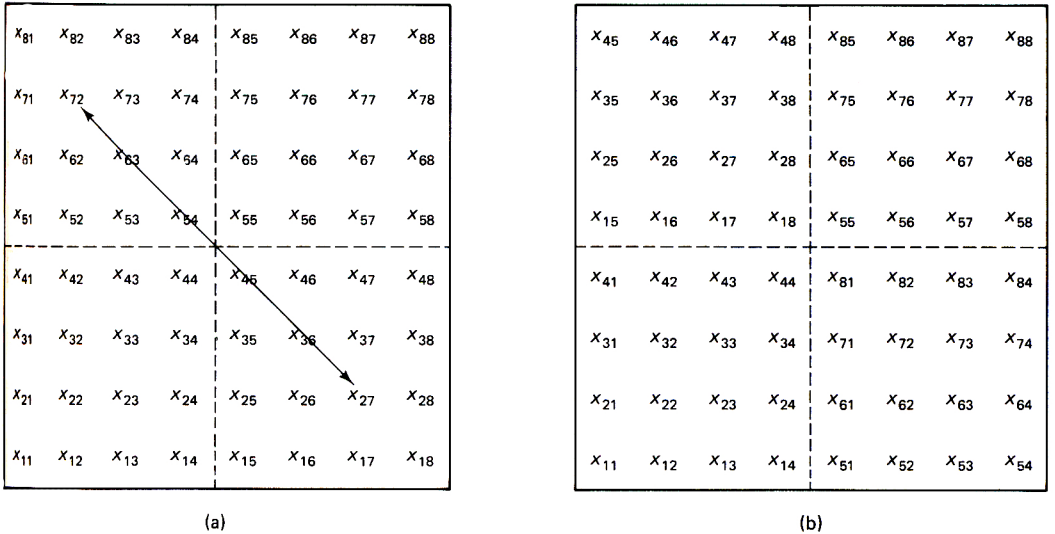


Figure 3.20 First stage of Eklundh's method. (a) Original sequence $x(n_1, n_2)$, showing exchanges; (b) output of the first stage.

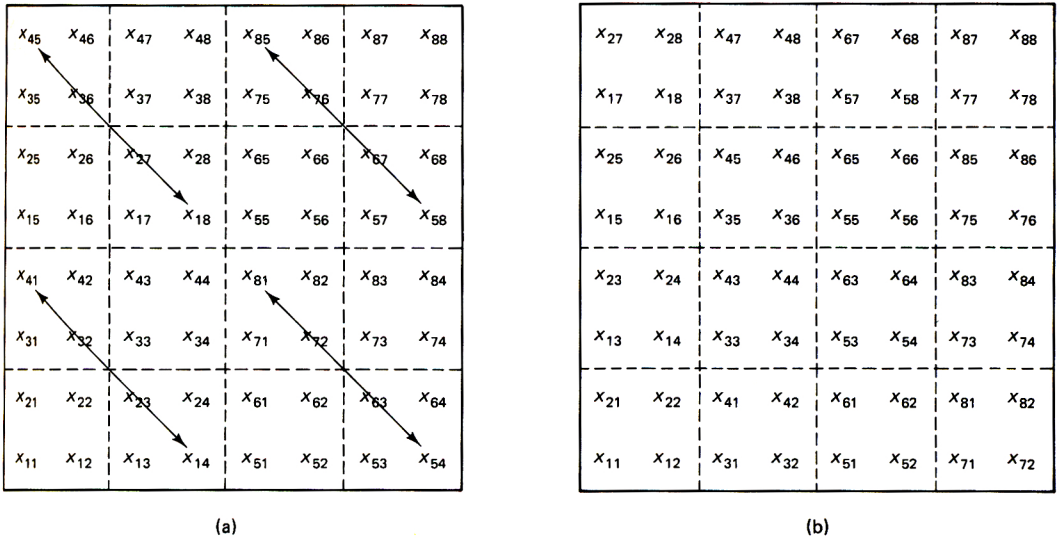


Figure 3.21 Second stage of Eklundh's method. (a) Input to the second stage, showing exchanges; (b) output of the second stage.

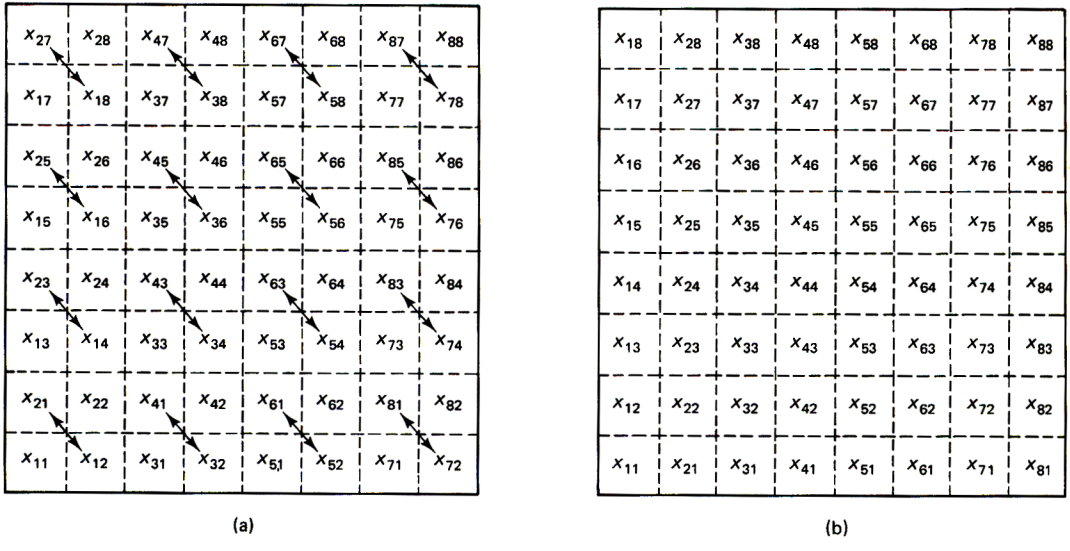


Figure 3.22 Third and final stage of Eklundh's method. (a) Input to the third stage, showing exchanges; (b) output $x^T(n_1, n_2) = x(n_2, n_1)$.

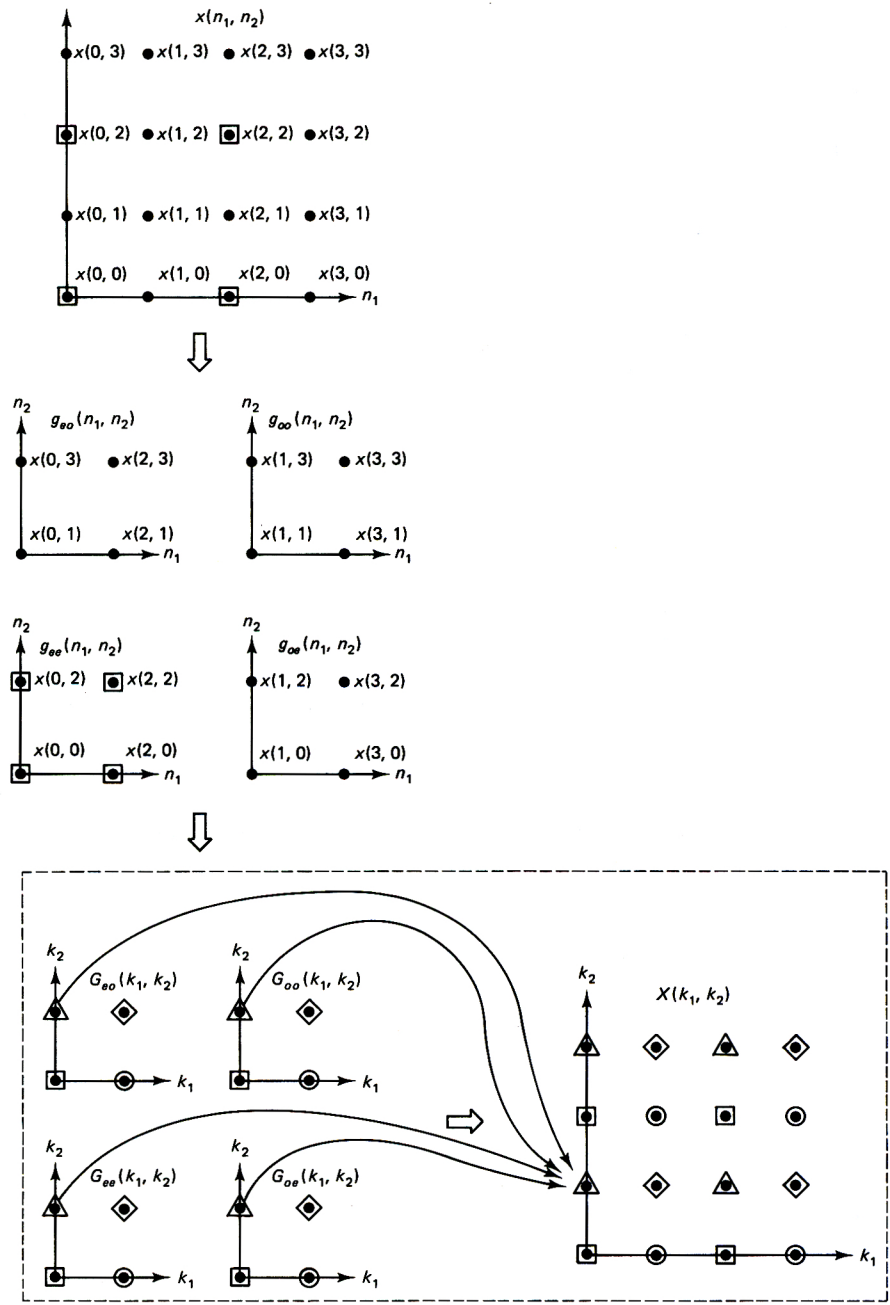


Figure 3.23 Development of one stage of decimation-in-space FFT algorithm.

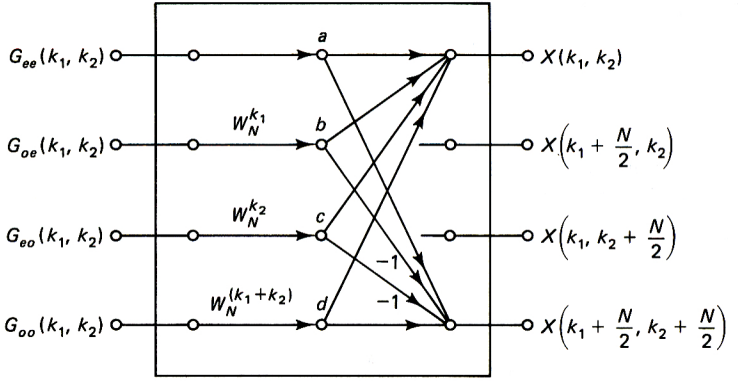


Figure 3.24 Vector radix 2 × 2 butterfly computation. For clarity, not all arrows are shown.

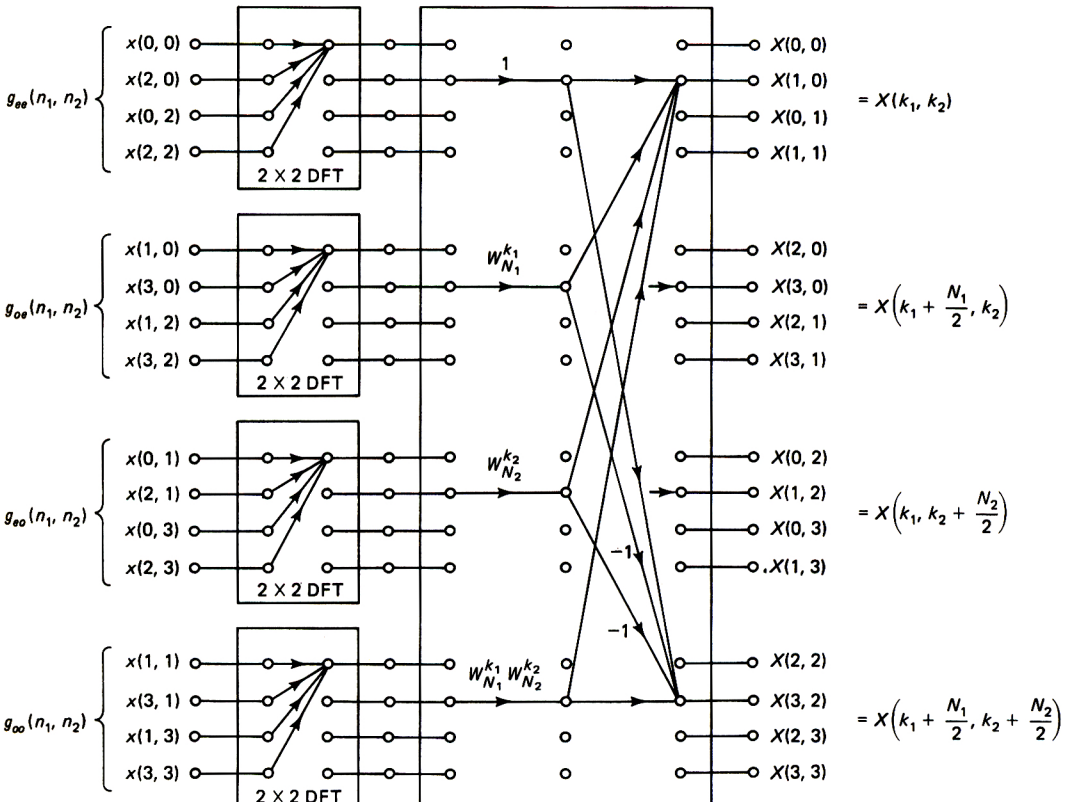


Figure 3.25 Decimation-in-space 2×2 vector radix FFT, for the case $N_1 = N_2 = 4$. For clarity, many values are omitted.