並列計算機応用

信号処理

画像処理

３次元グラフィックス処理

VLSICAD

データベース処理

人工知能

数値計算

第６章　並列処理応用

６．１信号処理

6.1.1　FFTの原理

$$f'_k=1/N\ \Sigma_{j=0}^{N-1}\exp(-2\pi ijk/N)f_j$$

$$0\leqq k\leqq N-1 \qquad （1）$$

$$N\{6N+2(N-1)+2\}=8N^2 \qquad （2）$$

6.1.2　N点フーリエ変換のp点、q点フーリエ変換への分解

$$N=pq$$

$$h_l^r=\Sigma_{j=0}^{p-1}f_{l+jq}\exp(-2\pi ijr/p) \qquad （3）$$

$$f'_{r+pm}=\Sigma_{k=0}^{N-1}\exp\{-2\pi i(r+pm)k/N\}f_k \qquad （4）$$

ただし、$k=l+jq$

$0\leq r\leq p-1,\ 0\leq l\leq q-1$

$0\leq m\leq q-1,\ 0\leq j\leq p-1,$

$$0 \leqq l \leqq q-1$$

$$0 \leqq j \leqq p-1$$

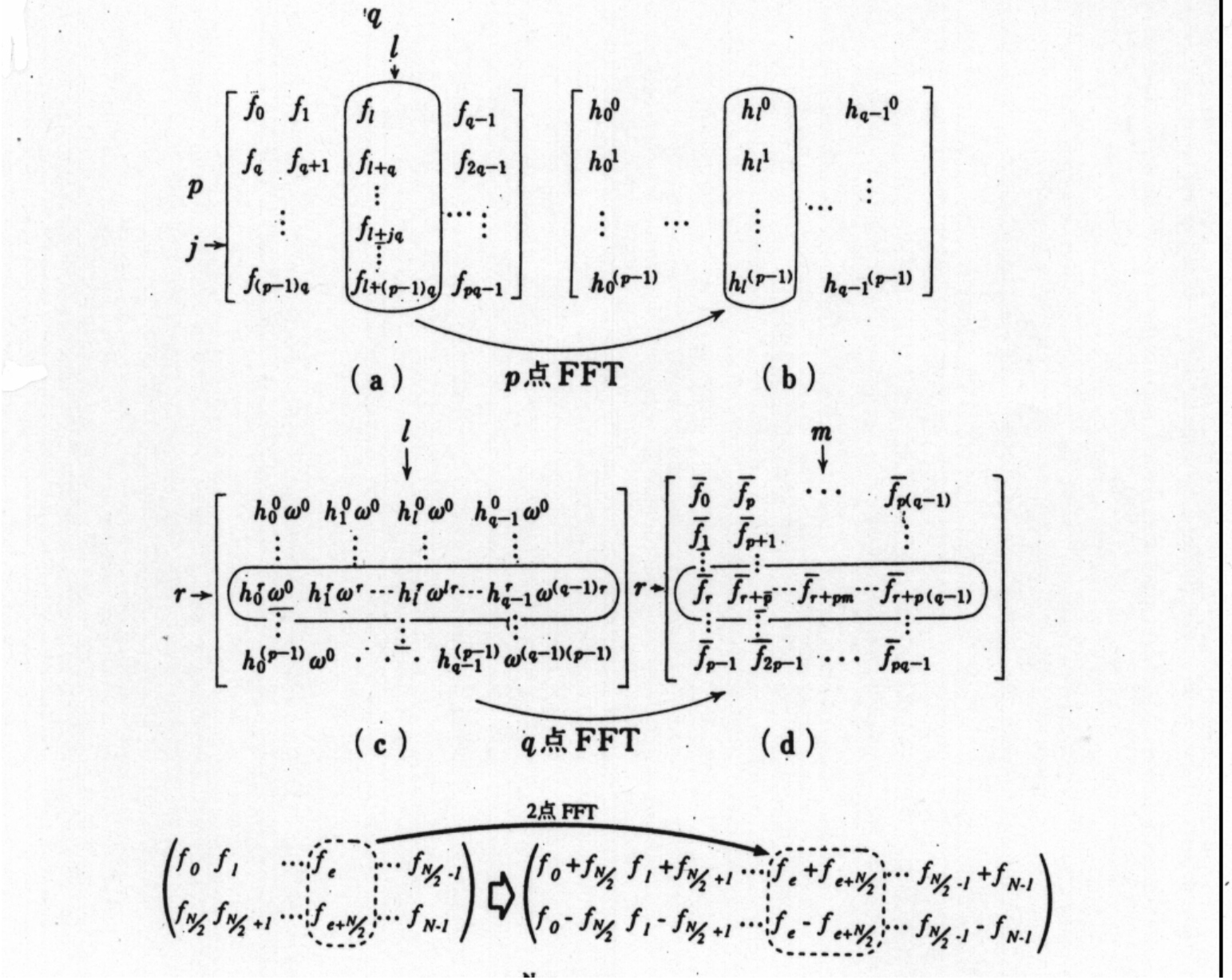
とおいて整理すると、

$$f'_{r+pm} = \sum_{l=0}^{q-1} \sum_{j=0}^{p-1} \exp\{-2\pi i(r+pm)(l+jq)/(pq)\} f_{l+jq}$$

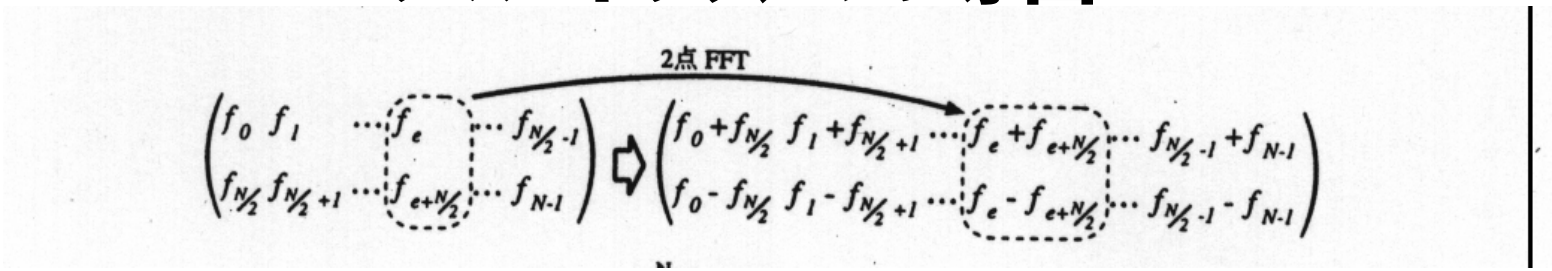
$$= \sum_{l=0}^{q-1} \{\exp(-2\pi i lm/q) \exp(-2\pi i rl/pq) x$$

$$\sum_{j=0}^{p-1} \exp(-2\pi i jr/p) f_{l+jq}\} \quad \textbf{（5）}$$

$$= \sum_{l=0}^{q-1} \exp(-2\pi i lm/q) \exp(-2\pi i lr/N) h_l{}^r$$


q
l
p
j
(a)
p点FFT
(b)
m
r
(c)
q点FFT
(d)
2点 FFT

6.1.3　FFTアルゴリズムの導出



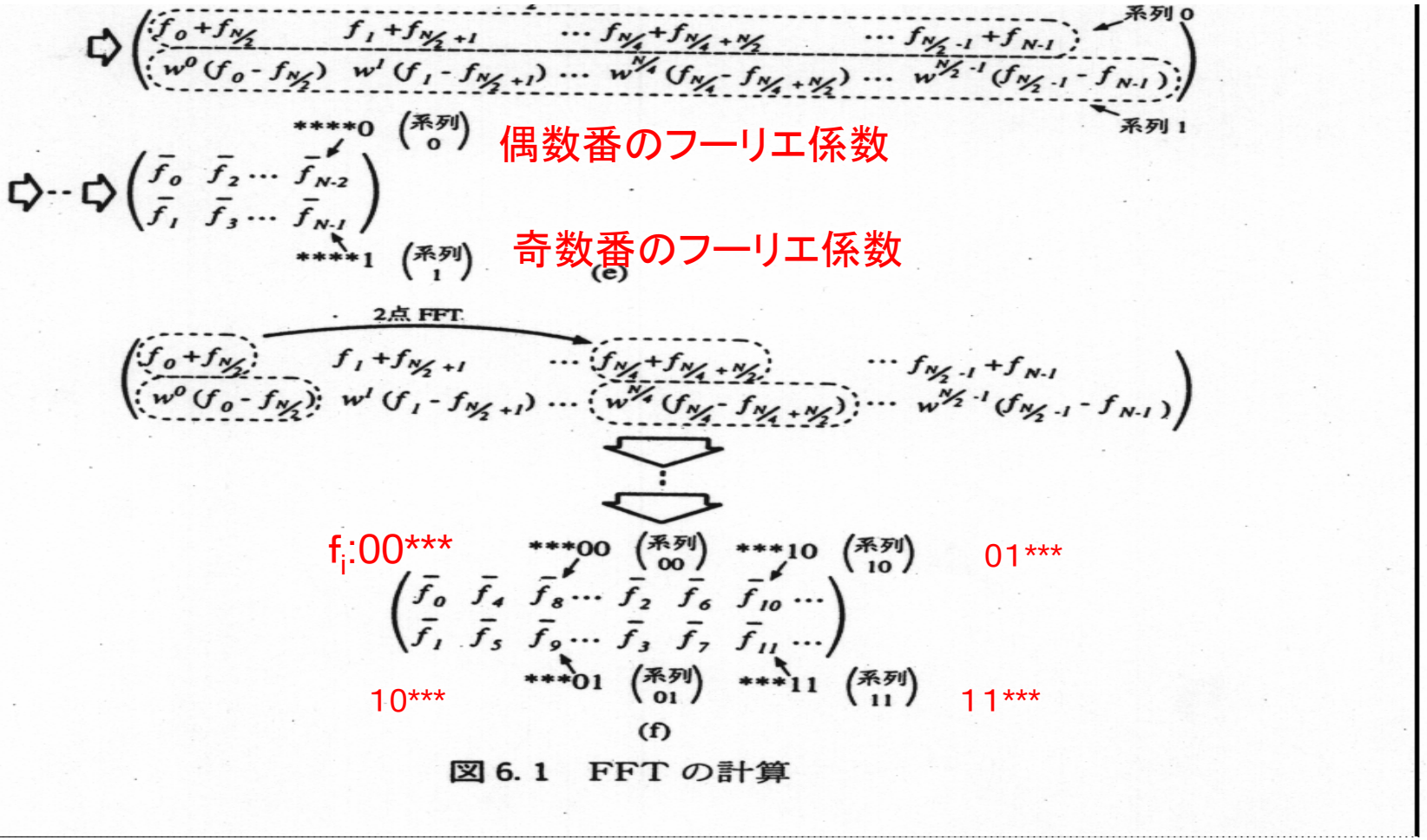
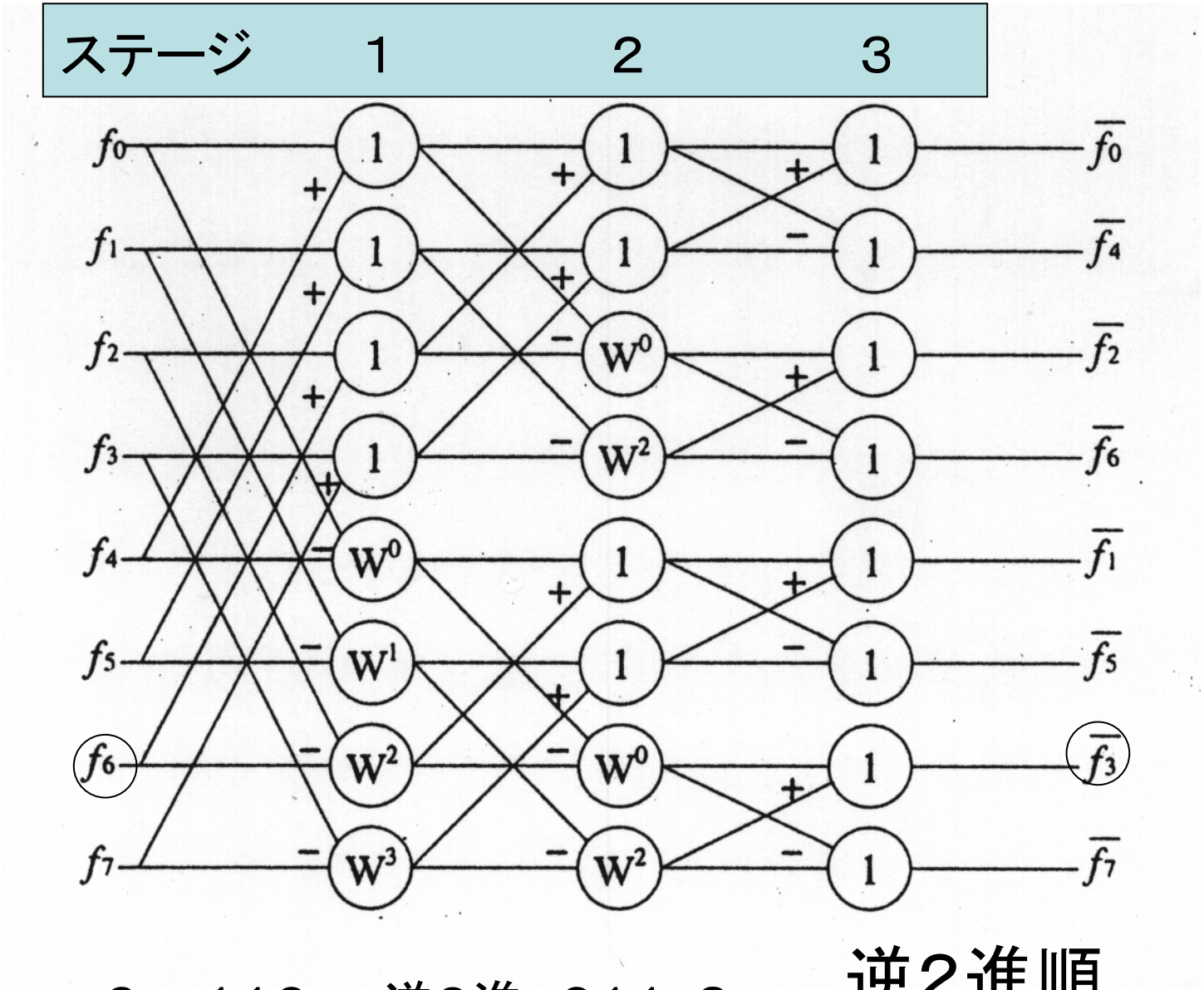


図 6.1　FFT の計算



6: 110 逆2進 011:3

逆2進順

6.1.3　FFTと並列処理

（１）逐次処理の場合

$$T=5N\alpha\log N \qquad （8）$$

半性能長

（２）パイプライン処理の場合

ベクトル長nのバタフライ演算：$10\alpha'(n+N_{1/2})$　　（9）

$$T_{p1}=10\alpha'\Sigma_{i=1}^{\log N}(N2^{-i}+N_{1/2})2^{i-1} \qquad （１０）$$

$$=5N\alpha'\log N+10\alpha'N_{1/2}(N-1)$$

$$T_{p2}=10\alpha'\sum_{i=1}^{\log N}N/2^{i}\cdot(2^{i-1}+N_{1/2}) \qquad （１１）$$

$$=T_{p1}$$

$$T_{p3}=10\alpha'\{\sum_{i=1}^{K}(N2^{-i}+N_{1/2})2^{i-1}+$$

$$\sum_{i=K+1}^{\log N}N/2^{i}\cdot(2^{i-1}+N_{1/2})$$

$$=10\alpha'(N_{1/2}2^{K}+NN_{1/2}2^{K}-2N_{1/2})+5N\alpha'\log N \qquad （１２）$$

$$dT_{p3}/dK=0 \qquad (13)$$

より、

$$K=1/2\log N \qquad (14)$$

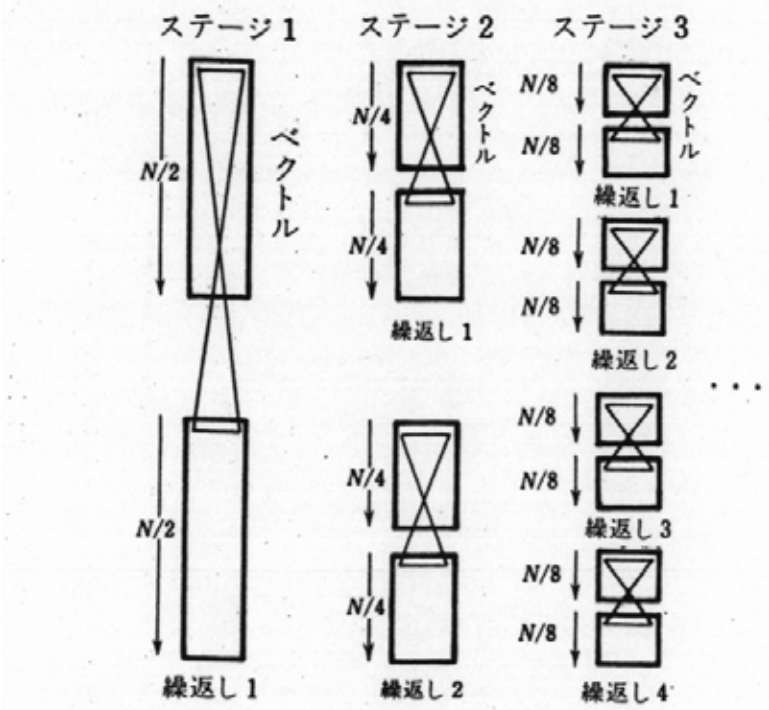
式（9）、（11）より、

$$T_{p3}=5N\alpha'\log N+20\alpha' N_{1/2}(\sqrt{N}-1) \qquad (15)$$

スピードアップ率S_pは、

$$S_p=\gamma\cdot 1/\{1+4N_{1/2}/(\sqrt{N}\log N)\} \qquad (16)$$

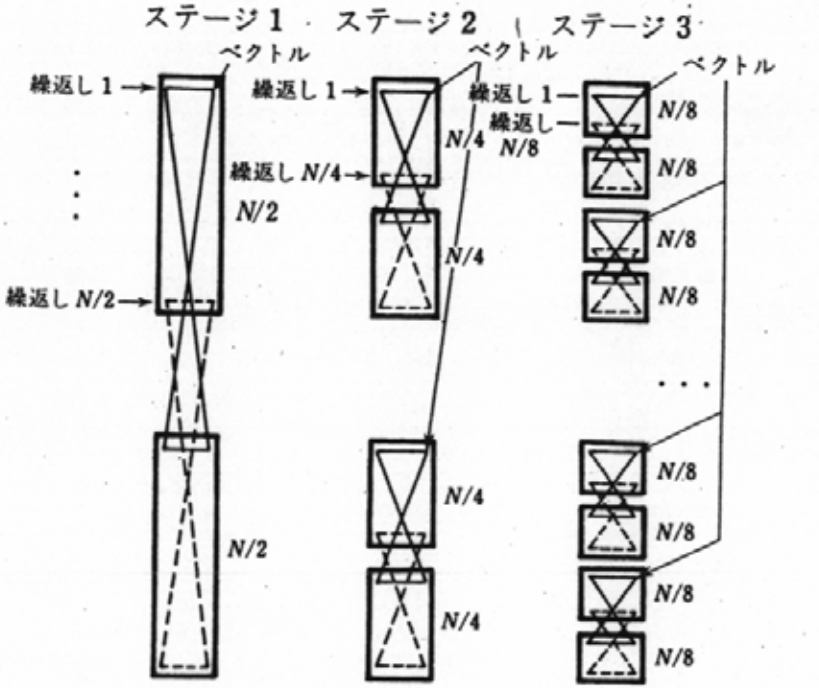
$$\gamma=\alpha/\alpha'$$



ベクトル長	$N/2$	$N/4$	$N/8$	$\cdots$	1
繰返し	1	2	4	$\cdots$	$N/2$

$$T_{p1}=10\alpha'\sum_{i=1}^{\log_2 N}(N\cdot 2^{-i}+N_{1/2})\,2^{i-1}$$
$$=10\alpha'\{N/2\log_2 N+N_{1/2}(N-1)\}$$

（a） 最内ループ i でベクトル化



ベクトル長	1	2	4	$\cdots$	$N/2$
繰返し	$N/2$	$N/4$	$N/8$	$\cdots$	1

$$T_{p2}=10\alpha'\sum_{i=1}^{\log_2 N}N\cdot 1/2^{i}\,(2^{i-1}+N_{1/2})$$
$$=10\alpha'\{N/2\log_2 N+N_{1/2}(N-1)\}$$

（b） 最内ループ k でベクトル化

ステージ　　1　2　3　logP

データ距離　N/2　N/4　N/8　…N/P

（３）並列処理の場合

logP段までの演算時間

$8(N\alpha/P)\ \log P$　　F+GまたはW(F-G)の計算をN/P回　**（１７）**

後半のlog(N/P)ステージ

$(5N\alpha/P)\log(N/P)$　　F+G,W(F-G)の計算をN/(2P)回　**（１８）**

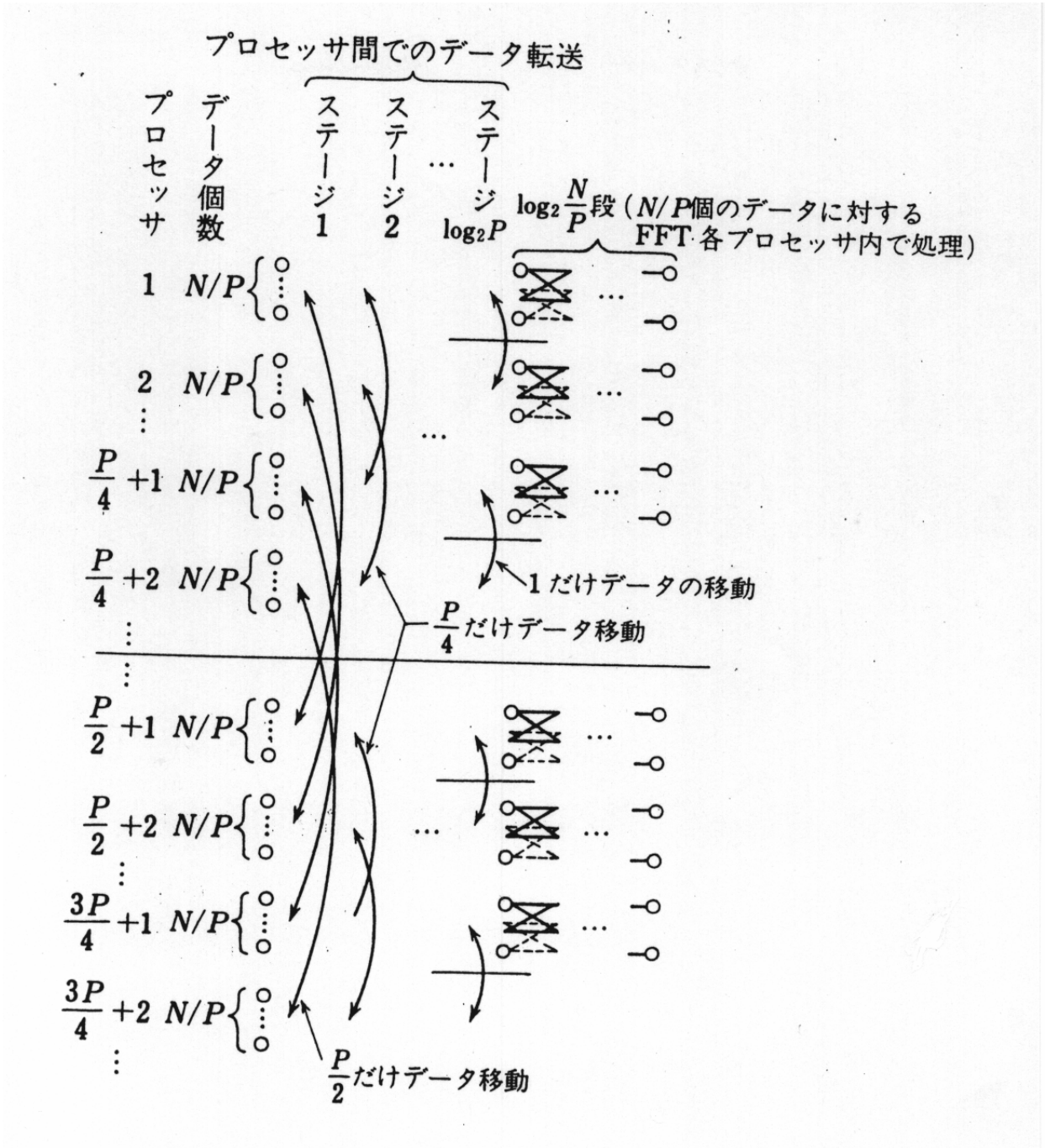
スピードアップ率S

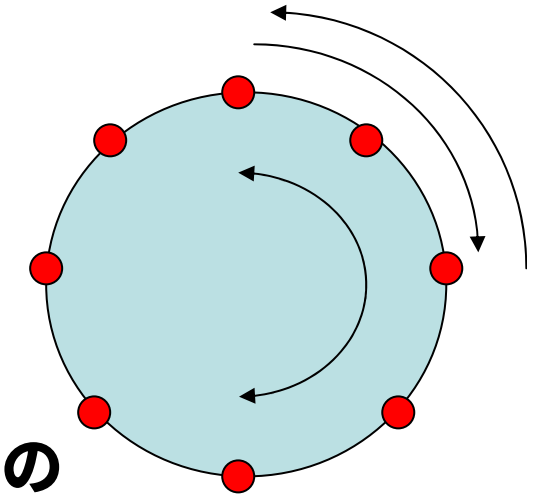
$S=5N\alpha\log N/\{(8N\alpha/P)\log P+(5N\alpha/P)\log(N/P)+$**転送時間**$\}$　**（１９）**

$=P/[1+\{3\log P+(\gamma P/N)$**転送回数**$\}/5\log N]$

ここに、$\gamma=\beta/\alpha$である。

P,$P^{1/2}$,logP


プロセッサ間でのデータ転送
プロセッサ
データ個数
ステージ 1
ステージ 2
ステージ log2P
log2 N/P 段（N/P個のデータに対する FFT 各プロセッサ内で処理）
1 N/P
2 N/P
P/4 +1 N/P
P/4 +2 N/P
P/2 +1 N/P
P/2 +2 N/P
3P/4 +1 N/P
3P/4 +2 N/P
1 だけデータの移動
P/4 だけデータ移動
P/2 だけデータ移動



①リング網

転送時間

１回目：(N/P)*(P/2)

２回目：P/4**だけ離れたプロセッサ間での**

データ交換　(N/P)*(P/4)*2

３回目からlogP**回までは同様に、**

(N/P)*(P/8)*2,...,(N/P)*1*2

全転送回数：　　**（２０）**

$(N/P)\{P/2+2(P/4+P/8+\cdots+P/2^{\log p})\}=N(3/2-2/P)$

スピードアップ率S_r

$S_r=P/[1+(3\log P+3/2\gamma P)/5\log N]$　　**（２１）**

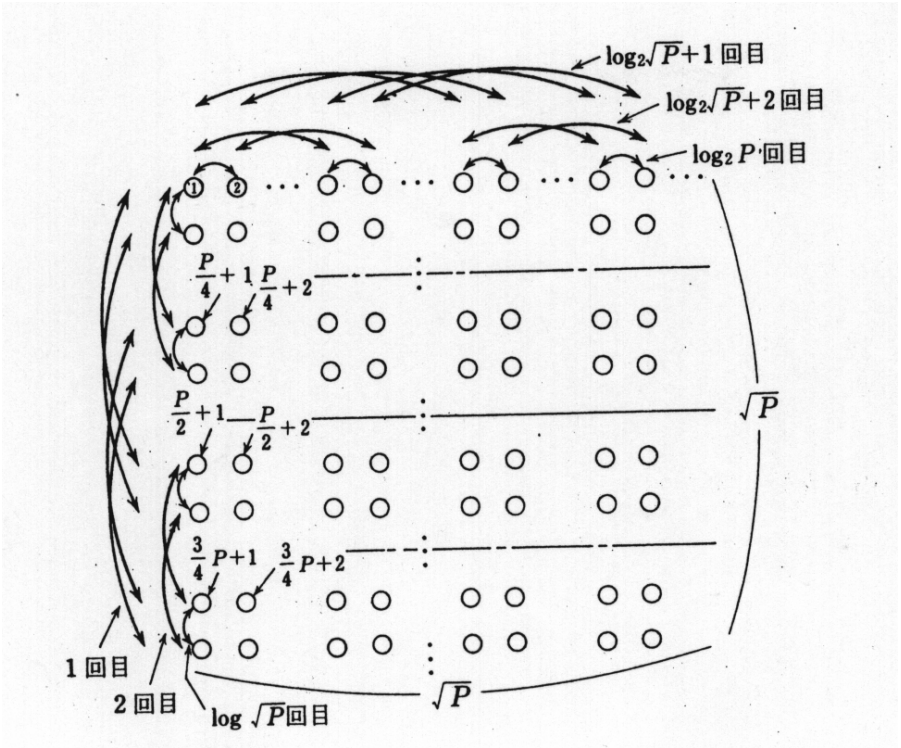
②トーラス

転送回数：

$$(3/\sqrt{P}-4/P)N \qquad (22)$$

スピードアップ率S_t

$$S_t=P/[1+(3\log P+3\gamma\sqrt{P})/5\log N] \qquad (23)$$



転送回数

1 回目 : $\frac{N}{P} \times \frac{\sqrt{P}}{2}$

2 回目 : $\frac{N}{P} \times \frac{\sqrt{P}}{4} \times 2$

⋮

$\log_2 \sqrt{P}$ 回目 : $\frac{N}{P} \times 2$

$\log_2 \sqrt{P}+1$ 回目 : $\frac{N}{P} \times \frac{\sqrt{P}}{2}$

$\log_2 \sqrt{P}+2$ 回目 : $\frac{N}{P} \times \frac{\sqrt{P}}{4} \times 2$

⋮

$\log_2 P$ 回目 : $\frac{N}{P} \times 2$

合計 $\left(\frac{3}{\sqrt{P}} - \frac{4}{P}\right) N$

③ハイパキューブ

通信回数は最初のlogP**段まで１段当たり**2N/P

全転送回数

$$(2N/P)\log P \qquad （２４）$$

スピードアップ率S_{bnc}

$$S_{bnc}＝P/[1+(3\log P+2\gamma\log P)/5\log N] \qquad （２５）$$

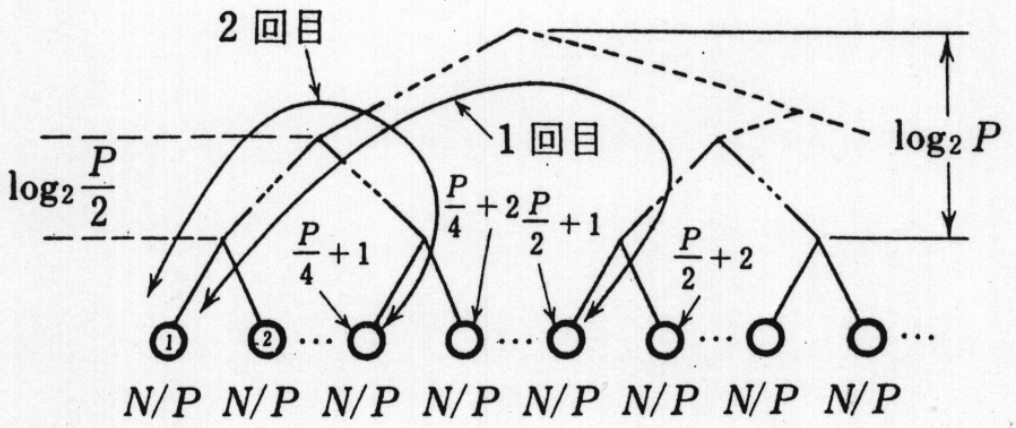
④トリー

ファットトリーの転送回数：

$$(2N/P)*(\log P)*(\log P+1) \qquad (26)$$

スピードアップ率S_{ft}

$$S_{ft}=P/[1+(3\log P+2\gamma(\log P)^2)/5\log N] \qquad (27)$$



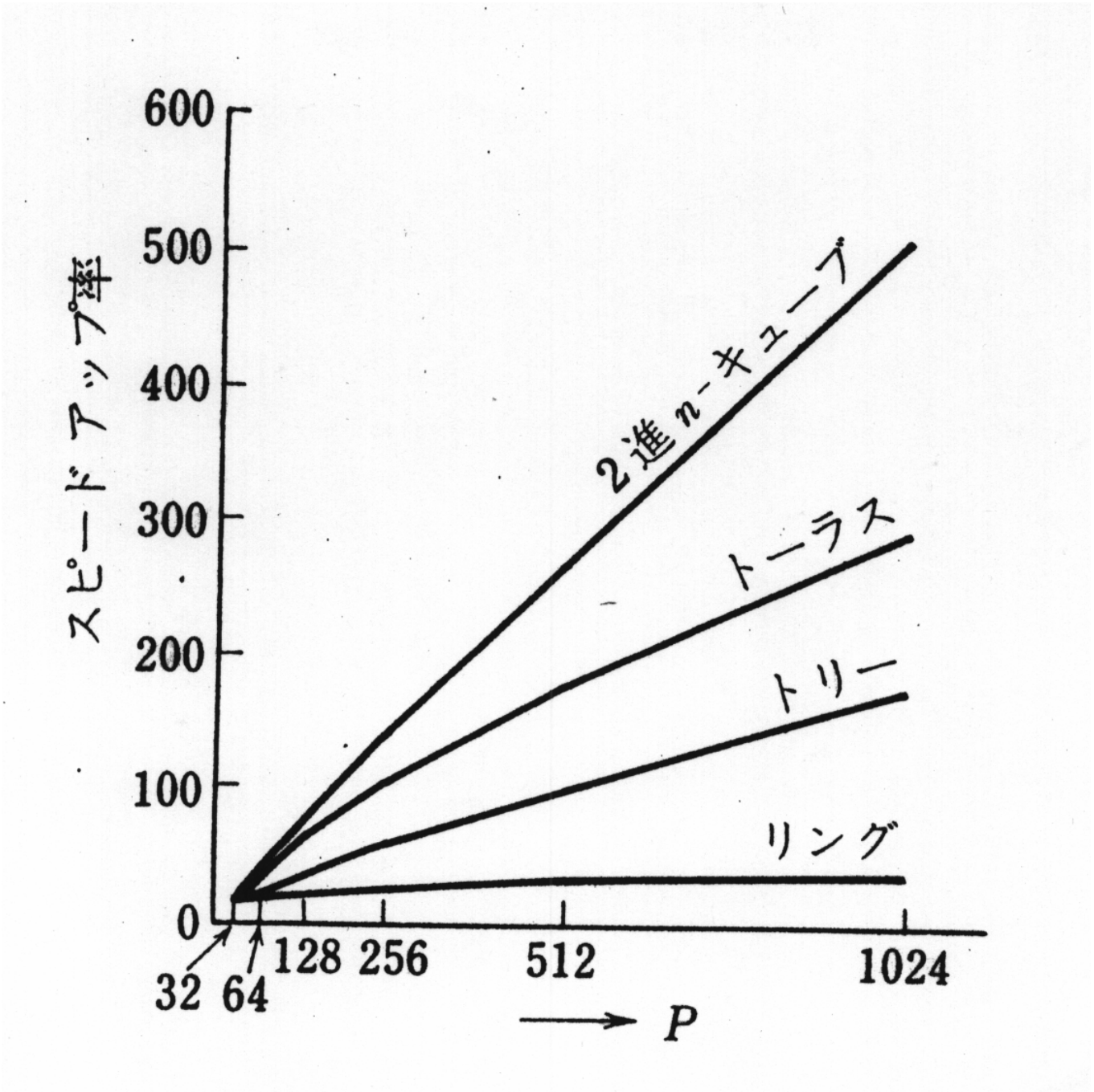
1 回目 : $\frac{N}{P}\times 2\log_2 P\times 2$

2 回目 : $\frac{N}{P}\times 2\log_2 \frac{P}{2}\times 2$

$\vdots$

$\log_2 P$ 回目 : $\frac{N}{P}\times 2\log_2 \frac{P}{2^{\log_2 P-1}}\times 2$

合計 : $\frac{2N}{P}(\log_2 P)(\log_2 P+1)$


600
500
400
300
200
100
0
スピードアップ率
2進n-キューブ
トーラス
トリー
リング
32
64
128
256
512
1024
P

（５）専用FFTコンピュータ

N点FFT（N=pq）：

パイプライン処理

p点FFT

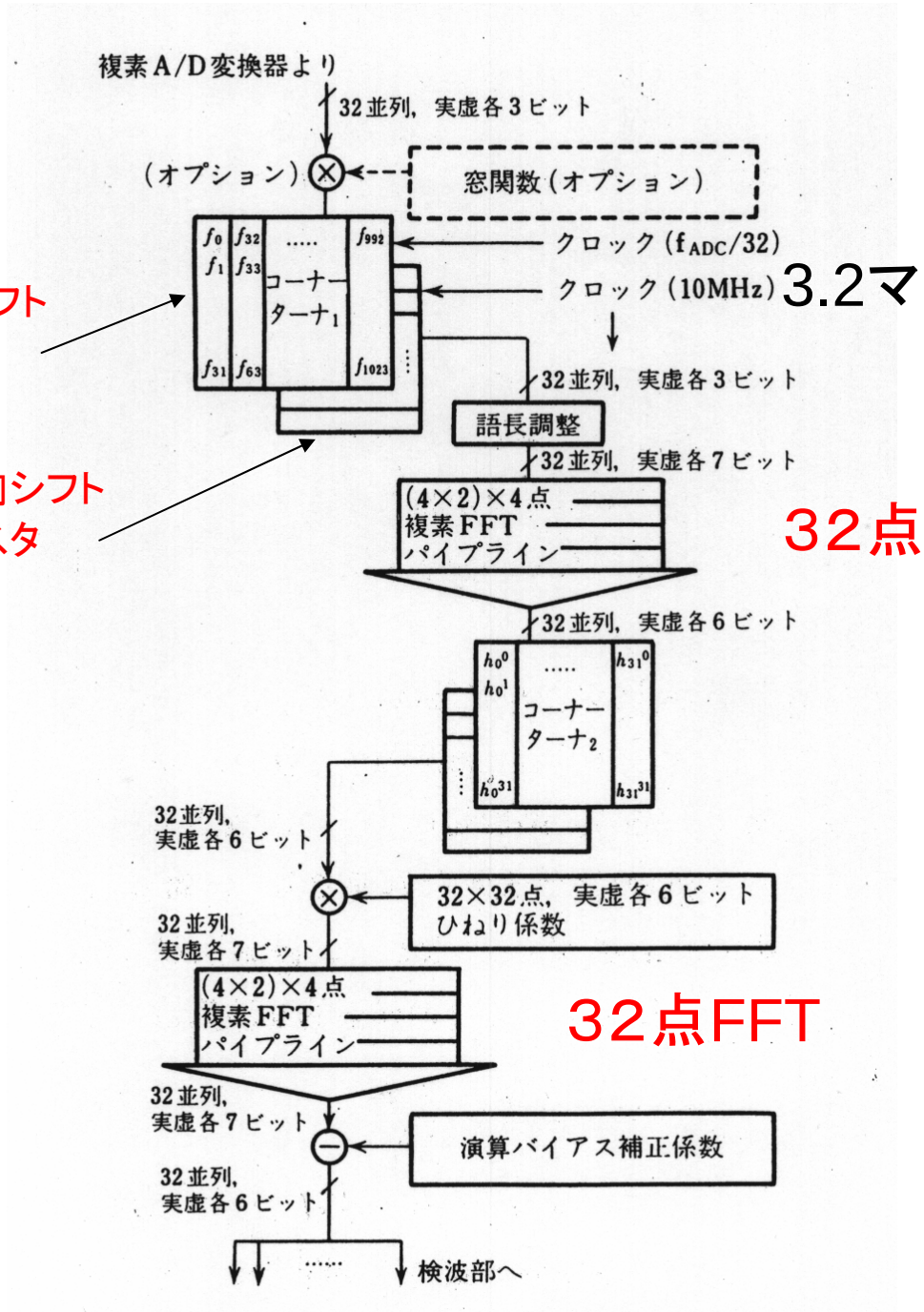
ひねり係数の乗算

q点FFT

東京天文台のFFTプロセッサ

N=1024、p=q=32

コーナターナ：２次元シフトレジスタ



X方向シフトレジスタ

Y方向シフトレジスタ

3.2マイクロ秒／1024点

32点FFT

32点FFT

６．２画像処理

6.2.1空間フィルタ処理

$$\nabla 2\phi = d^2\phi/dx^2 + d^2\phi/dy^2$$

$$= \phi(I+1,J) + \phi(I-1,J) +$$

$$\phi(I,J+1) + \phi(I,J-1) - 4\phi(I,J) \quad （２８）$$

①完全並列処理方式

ＭＰＰ、ＤＡＰ、ＣＡＰなど

②局所並列処理

東芝総合研究所のシステム（PPP）

フィルタ処理：輪郭線抽出など

ラプラシアンオペレータ

$$
\begin{aligned}
\nabla^2 \varphi &= \partial^2 \varphi / \partial x^2 + \partial^2 \varphi / \partial y^2 \\
&= \varphi(I+1, J) - \varphi(I, J) - \\
&\quad (\varphi(I, J) - \varphi(I-1, J)) + \\
&\quad \varphi(I, J+1) - \varphi(I, J) - \\
&\quad (\varphi(I, J) - \varphi(I, J-1)) \\
&= \varphi(I+1, J) + \varphi(I-1, J) \\
&+ \varphi(I, J+1) + \varphi(I, J-1) \\
&- 4\varphi(I, J)
\end{aligned}
$$

	1	
1	−4	1
	1	

MPP

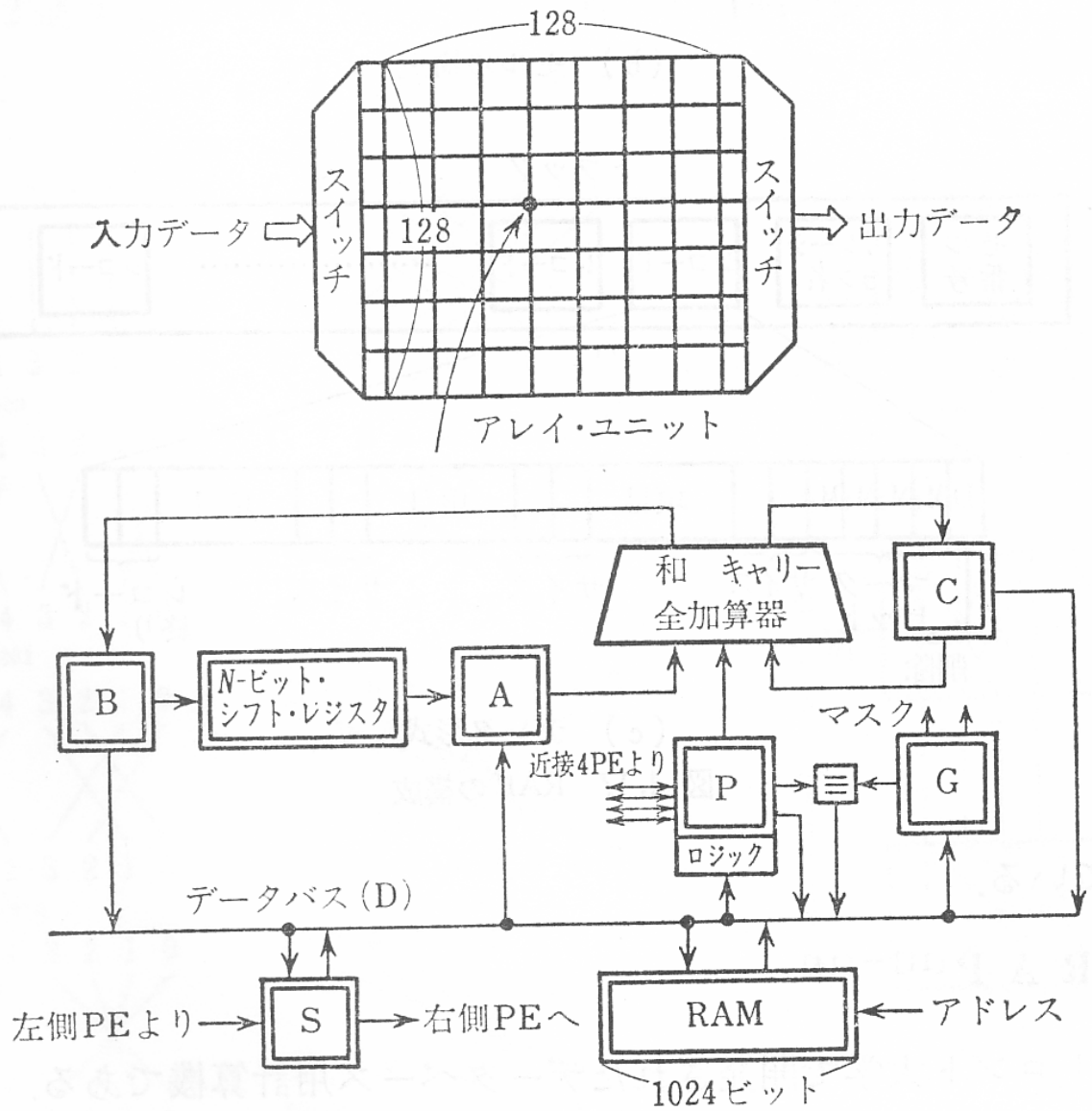


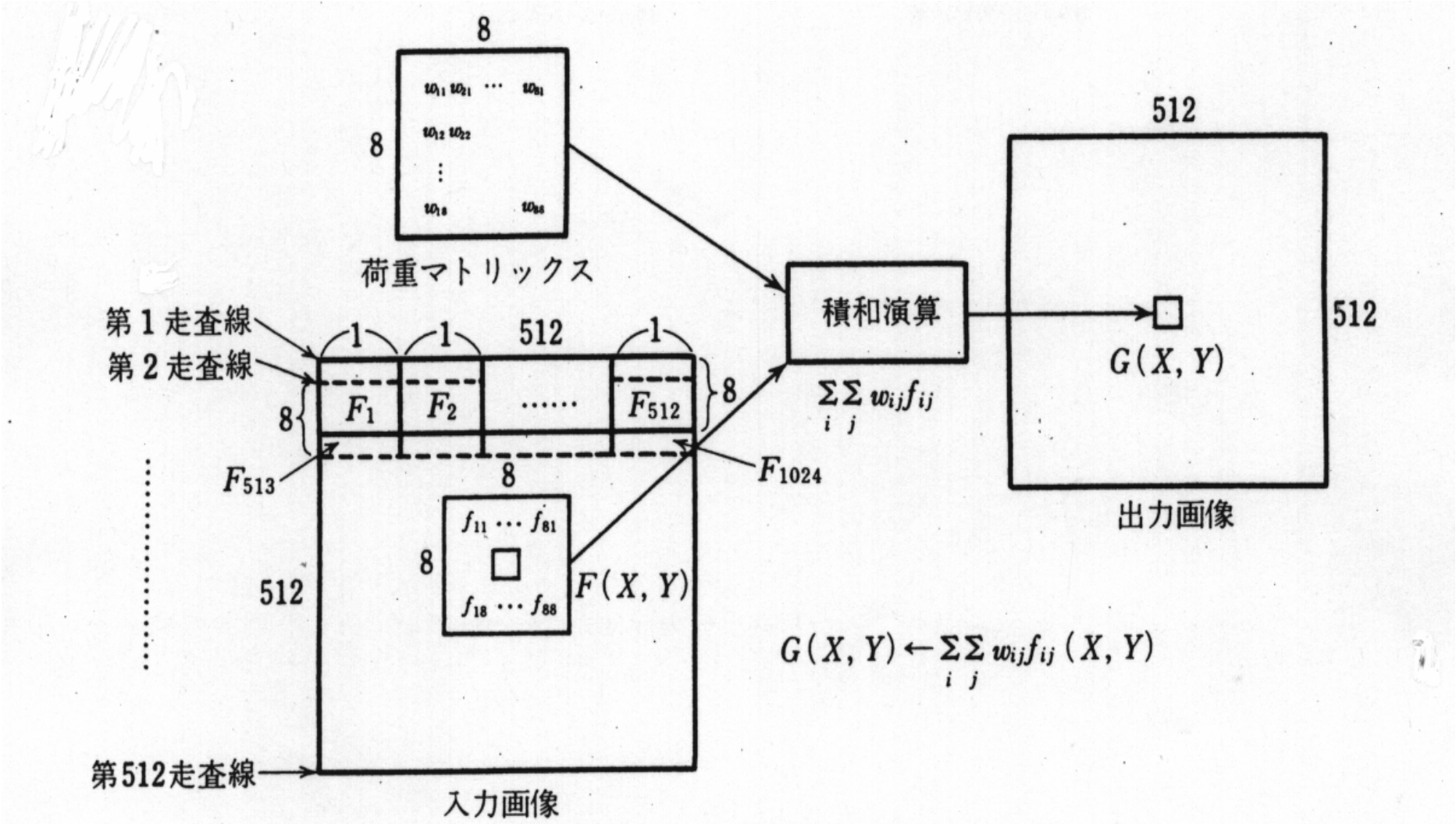
図 4.18 MPP の構成(K. E. Batcher: Design of a Massively Parallel Processor, *IEEE Trans.* C. Vol. 29, No. 9, 1980, pp. 836-840 による)

(a) Original photograph

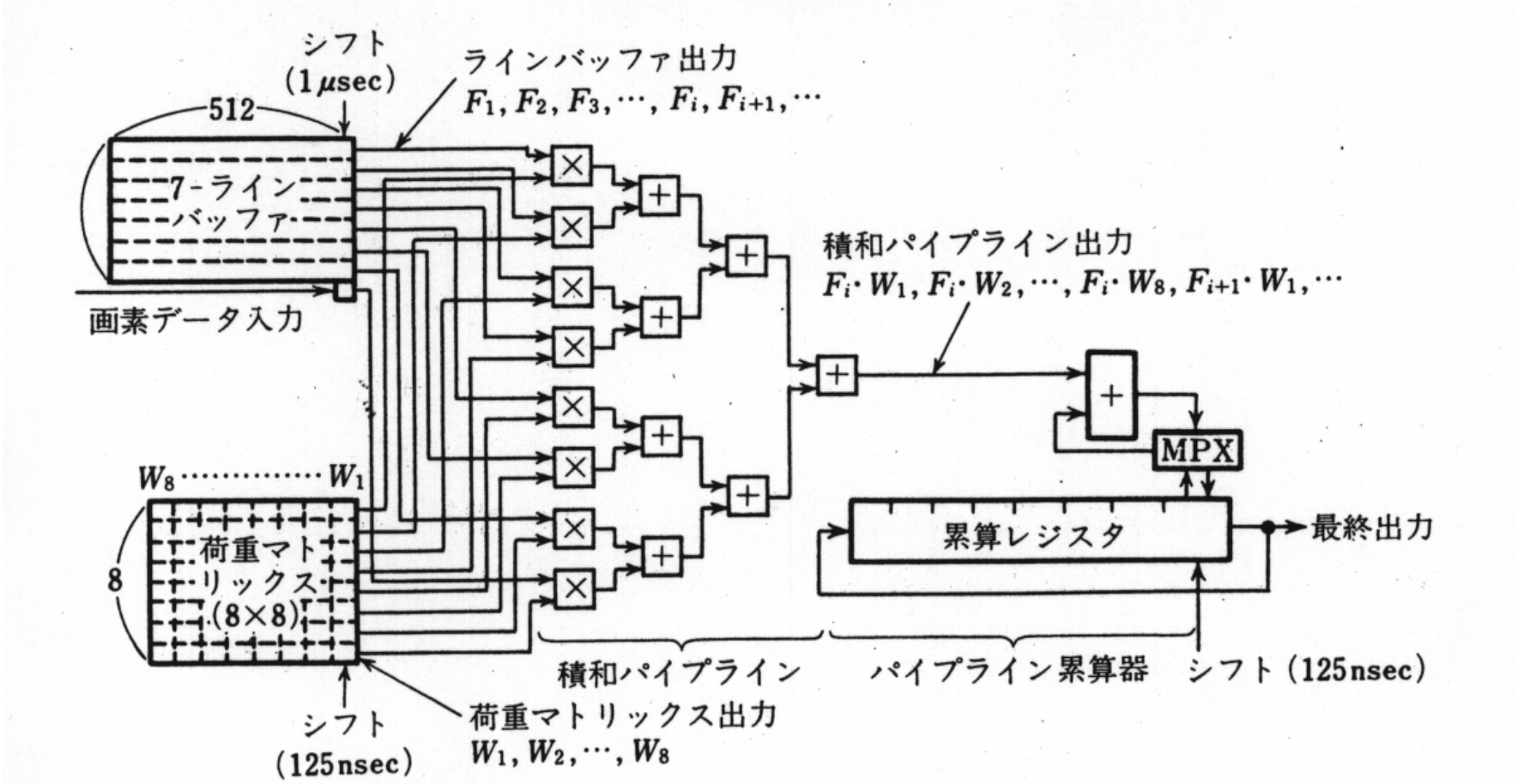
(b) Printout of the digital gray-level picture

(c) Binary picture

Figure 3-2 Picture input and line extraction. The dark horizontal line in the upper part is due to the burn in the CRT surface of the FSS used for digitization.



（a） 空間フィルタ処理の原理



(b) 空間フィルタ処理のためのパイプライン構成

ラインバッファ

F_i、F_{i+1}..,F_{i+7},...**の連続出力**

出力Pixel**値の計算**

$$F_i \cdot W_1 + F_{i+1} \cdot W_2 + \cdots + F_{i+7} \cdot W_8$$

$$F_{i+1} \cdot W_1 + F_{i+2} \cdot W_2 + \cdots + F_{i+8} \cdot W_8 \qquad (29)$$

パイプライン乗算器

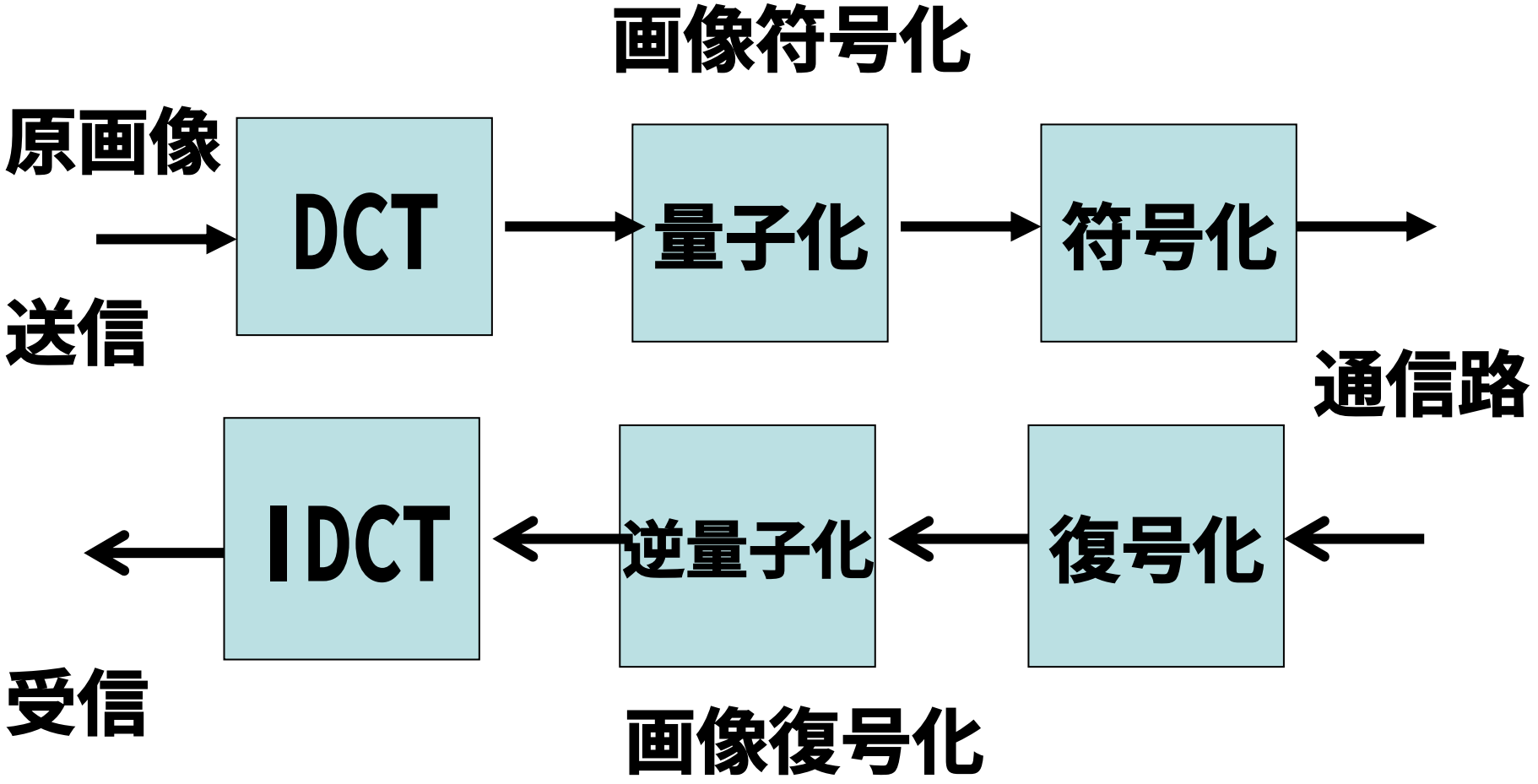
$$
\begin{array}{l}
F_i \cdot W_1,\quad F_i \cdot W_2,\quad F_i \cdot W_3\quad ,\ldots,F_i \cdot W_8 \\
F_{i+1} \cdot W_1, F_{i+1} \cdot W_2, F_{i+1} \cdot W_3, \ldots, F_{i+1} \cdot W_8 \\
F_{i+2} \cdot W_1, F_{i+2} \cdot W_2, F_{i+2} \cdot W_3, \ldots, F_{i+2} \cdot W_8 \\
F_{i+3} \cdot W_1, F_{i+3} \cdot W_2, F_{i+3} \cdot W_3, \ldots, F_{i+3} \cdot W_8 \\
F_{i+4} \cdot W_1, F_{i+4} \cdot W_2, F_{i+4} \cdot W_3, \ldots, F_{i+4} \cdot W_8 \\
F_{i+5} \cdot W_1, F_{i+5} \cdot W_2, F_{i+5} \cdot W_3, \ldots, F_{i+5} \cdot W_8 \\
F_{i+6} \cdot W_1, F_{i+6} \cdot W_2, F_{i+6} \cdot W_3, \ldots, F_{i+6} \cdot W_8 \\
F_{i+7} \cdot W_1, F_{i+7} \cdot W_2, F_{i+7} \cdot W_3, \ldots, F_{i+7} \cdot W_8 \\
F_{i+8} \cdot W_1, F_{i+8} \cdot W_2, F_{i+8} \cdot W_3, \ldots, F_{i+8} \cdot W_8
\end{array}
\qquad （３０）
$$

6.2.1**画像圧縮・復元処理**

MPEG2 （Moving Picture Image Coding Experts Group2）

画像符号化･復号化

DCT：離散コサイン変換



DCT変換（１次元）の証明

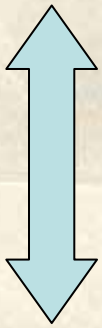
安田、藤原監訳

情報圧縮技術、bit別冊、共立、１９９７

正規化 DCT

$$X^{c2}(k) = \sqrt{\frac{2}{N}} c_k \sum_{n=0}^{N-1} x(n) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \qquad (5.11)$$

$$k = 0, 1, \ldots, N-1$$



正規化 IDCT

$$x(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} c_k X^{c2}(k) \cos\left[\frac{(2n+1)k\pi}{2N}\right],$$

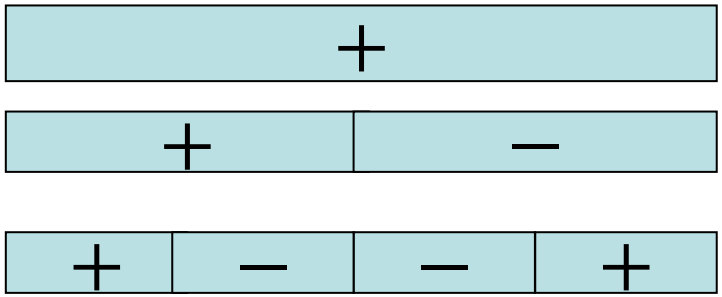
$$n = 0, 1, \ldots, N-1$$

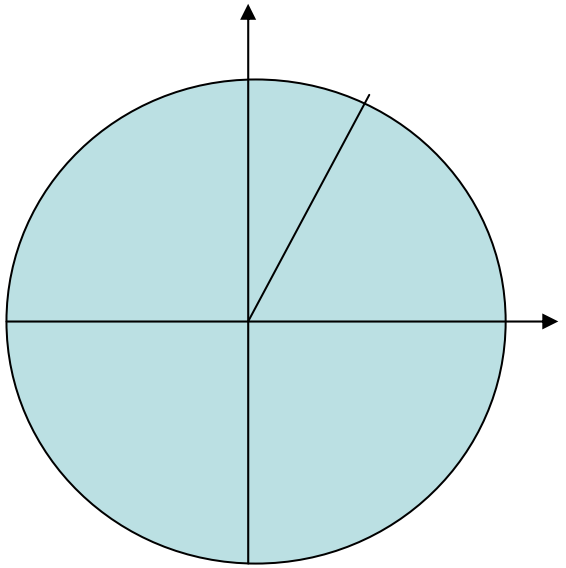
$\mathrm{COS}\{(2n+1)k\pi/(2N)\}$

	n=0	1	N/4-1	N/4	N/2-1	N/2	3/4N-1	3/4N	N-1
K=0	1	1	1	1	1	1	1	1	1
K=1	1/(2N)	3/(2N)・・・・・・・・・・・・・・・・・・・・・・・・			1/2-1/(2N)	1/2+1/(2N)・・・・・・・・・・・・・・・・・・・・・・・・			1-1/(2N)
K=2	1/N	3/N・・・(1/2-1/N)		（1/2＋1/N)・・・(1-1/N)		(1+1/N)・・・(3/2-1/N)		(3/2+1/N)・・・(2-1/N)	
・・・・・・・・・・・・・・・・									
K=N-1									

n=0	n=1	n=2	n=3
1/2-1/(2N)	3/2-3/(2N)	1/2-5/(2N)	3/2-7/(2N)

Kが小さいほど低周波領域





基底ベクトル

$$c(k,n)=1/\sqrt{N}:k=0$$

$$c(k,n)=\sqrt{2/N}\cos\frac{(2n+1)k\pi}{2N}:k\neq 0$$

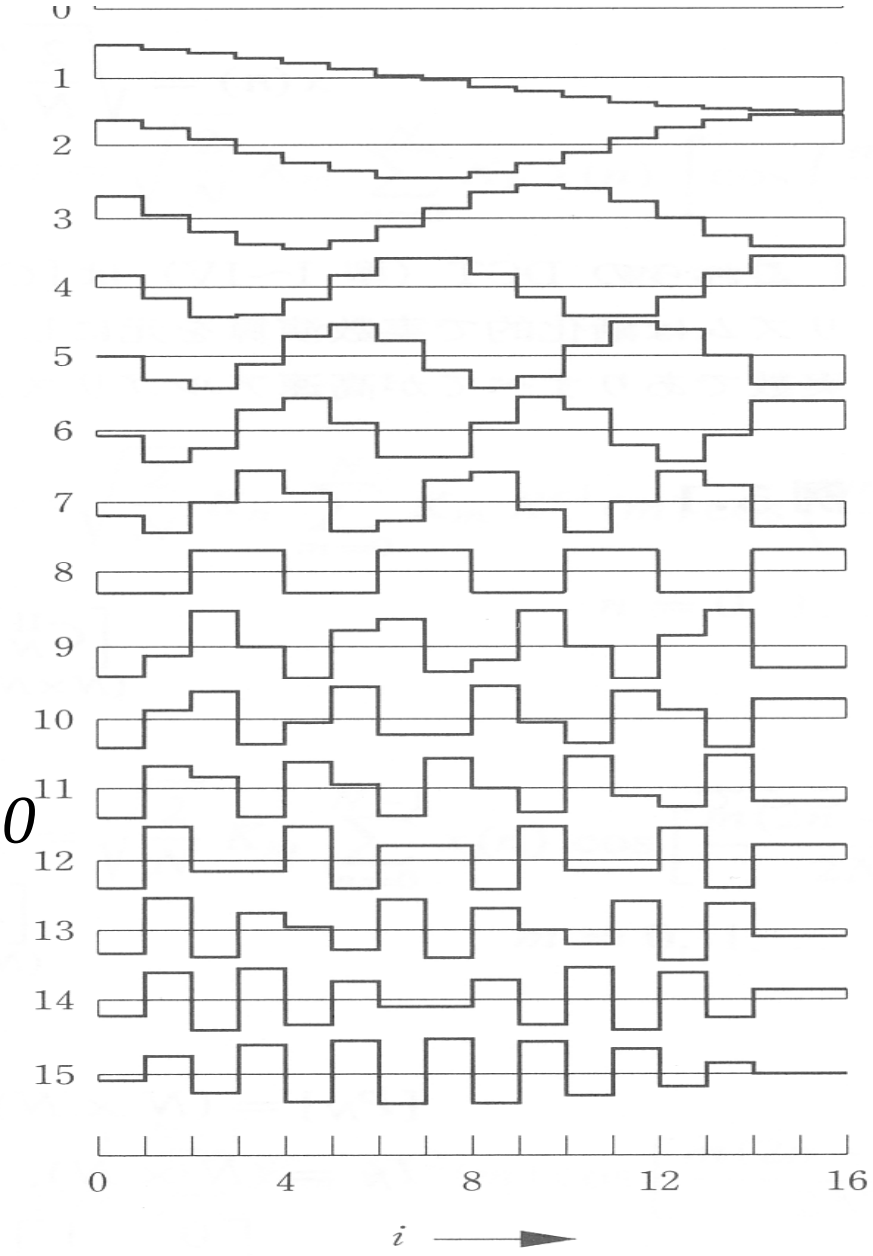


図 5.1　DCT-II の基底関数 $N = 16$

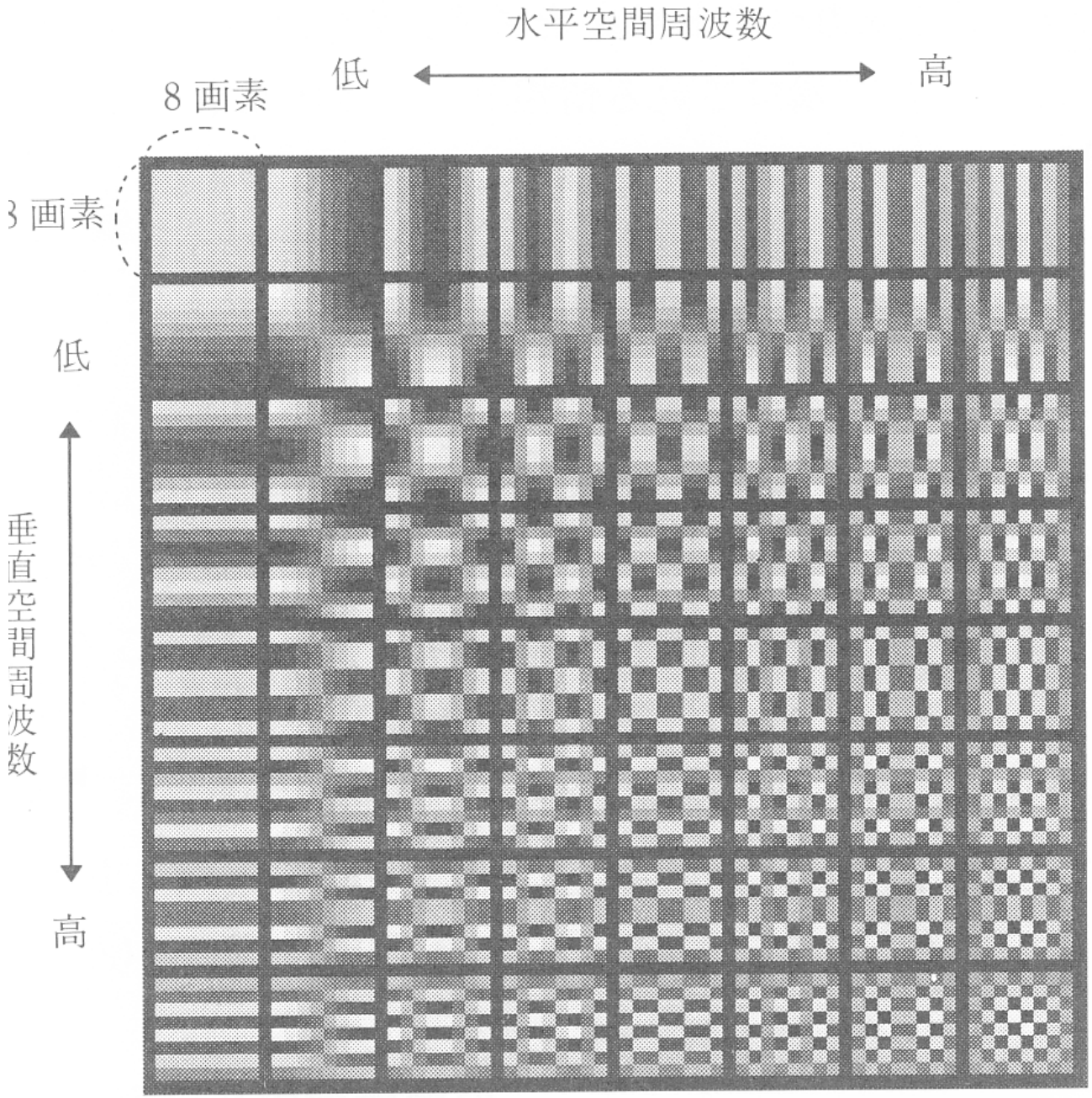


図 4.13　2 次元 DCT の基底画像パターン（8×8 の場合）

$$x_n = \frac{2}{N}\sum_{k=0}^{N-1} c_k^2 \sum_{m=0}^{N-1} x_m \cos\left[\frac{(2m+1)k\pi}{2N}\right]\cos\left[\frac{(2n+1)k\pi}{2N}\right]$$
$$= \frac{2}{N}\sum_{m=0}^{N-1} x_m \sum_{k=0}^{N-1} c_k^2 \cos\theta_m \cos\theta_n \qquad (5.25)$$

ただし，

$$\theta_m = \frac{(2m+1)k\pi}{2N}, \quad \theta_n = \frac{(2n+1)k\pi}{2N}$$
$$\cos\theta_m \cos\theta_n = \frac{1}{2}\left[\cos(\theta_m+\theta_n) + \cos(\theta_m-\theta_n)\right]$$
$$= \frac{1}{2}\left[\frac{e^{j(\theta_m+\theta_n)} + e^{-j(\theta_m+\theta_n)}}{2}\right] + \frac{1}{2}\left[\frac{e^{j(\theta_m-\theta_n)} + e^{-j(\theta_m-\theta_n)}}{2}\right]$$

ここで，$j = \sqrt{-1}$ とおくと，式(5.25) は次のようになる．

$$x_n = \frac{1}{2N}\sum_{m=0}^{N-1} x_m \sum_{k=0}^{N-1} c_k^2 \left(\exp\left[j\frac{2k\pi}{2N}(m+n+1)\right] + \exp\left[-j\frac{2k\pi}{2N}(m+n+1)\right]\right.$$
$$\left. + \exp\left[j\frac{2k\pi}{2N}(m-n)\right] + \exp\left[-j\frac{2k\pi}{2N}(m-n)\right]\right) \qquad (5.26)$$

$$c_k = \begin{cases} 1, & k \neq 0 \\ 1/\sqrt{2}, & k = 0 \end{cases}$$

$$= \frac{1}{2N} \sum_{m=0}^{N-1} x_m \sum_{k=0}^{N-1} c_k^2 \left(W_N^{-(m+n+1)\frac{k}{2}} + W_N^{(m+n+1)\frac{k}{2}} + W_N^{-(m-n)\frac{k}{2}} + W_N^{(m-n)\frac{k}{2}} \right) \tag{5.27}$$

ただし，$W_N = \exp\left(\frac{-j2\pi}{N}\right)$＝ユニタリ値の N 乗根，である．

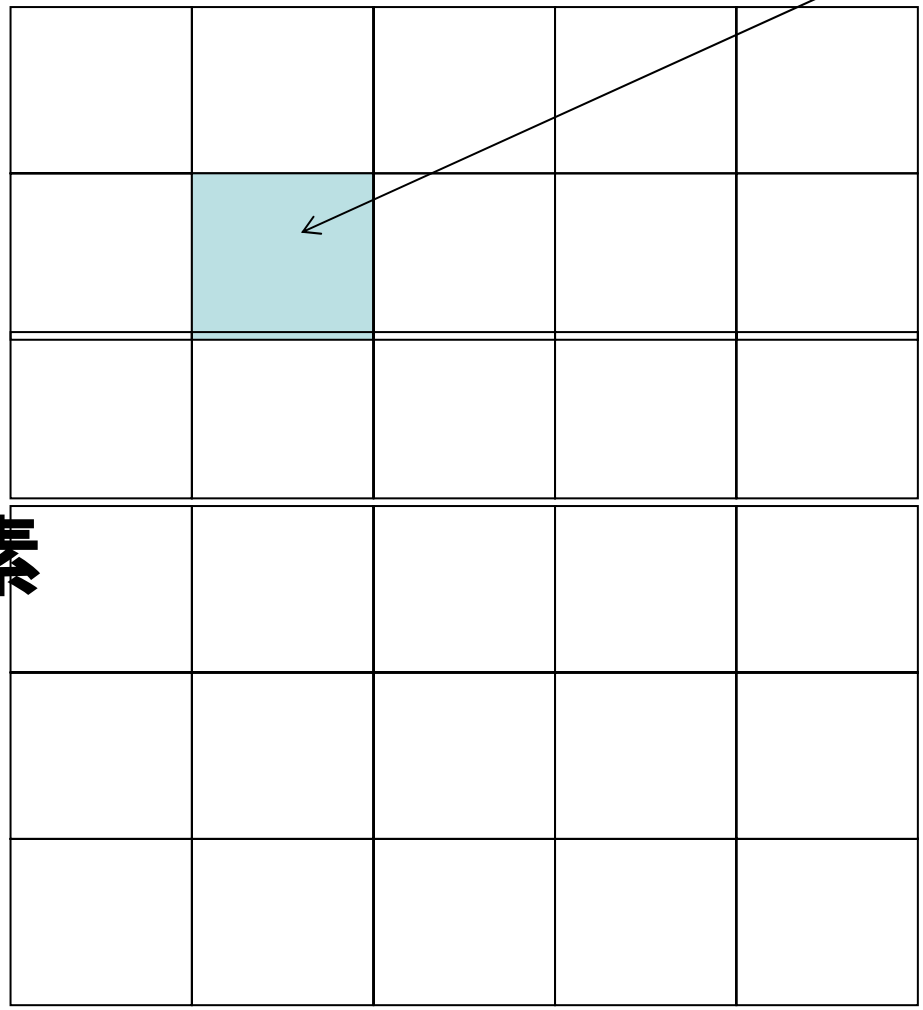
式(5.27) は以下のように変形出来る．

$m = n$ and $k \neq 0$ のとき $\left(\frac{1}{2N} x_n 2N\right) = x_n$

$m = n$ and $k = 0$ のとき $\left(\frac{1}{2N} x_n \left(\frac{1}{\sqrt{2}}\right)^2 4N\right) = x_n$

ただし $\sum_{k=0}^{N-1} W_N^{kl} = N\delta(l)$，ここで $\delta(l) = \begin{cases} 0, & l \neq 0 \\ 1, & l = 0 \end{cases}$

７２０画素



４８０画素

ブロック

８ｘ８画素

１画像：

９０ｘ６０
ブロック

正規化 2D-DCT

各ブロックに対して
DCT（N,M=8）

$$X_{u,v}^{c2} = c_u c_v \frac{2}{\sqrt{NM}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x_{n,m} \cos\left[\frac{(2n+1)u\pi}{2N}\right] \cos\left[\frac{(2m+1)v\pi}{2M}\right]$$

$$= \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} c_u \left[\sqrt{\frac{2}{M}} c_v \sum_{m=0}^{M-1} x_{n,m} \cos\frac{(2m+1)v\pi}{2M}\right] \cos\frac{(2n+1)u\pi}{2N}, \quad (5.38)$$

$$u = 0, 1, \ldots, N-1, \qquad v = 0, 1, \ldots, M-1, \qquad c_l = \begin{cases} 1/\sqrt{2}, & l = 0 \\ 1, & l \neq 0 \end{cases}$$

正規化 2D-IDCT

$$x_{n,m} = \frac{2}{\sqrt{NM}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} c_u c_v X_{u,v}^{c2} \cos\left[\frac{(2n+1)u\pi}{2N}\right] \cos\left[\frac{(2m+1)v\pi}{2M}\right], \quad (5.39)$$

$$n = 0, 1, \ldots, N-1, \qquad m = 0, 1, \ldots, M-1$$

- 演算量
- 各ブロック当たり:64x(63+64*4)=20480
- 毎秒当たりのブロック数
- 5400*30=162000
- 演算量:3.2G演算／秒

総和

X*cos*cos

cos：テーブル参照

各ブロックのDC係数Xuvに量子化

Xquv=Nearest Int (Xuv/Quv)

表8.1 輝度量子化マトリックス Q_{uv} [367]

v

u

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

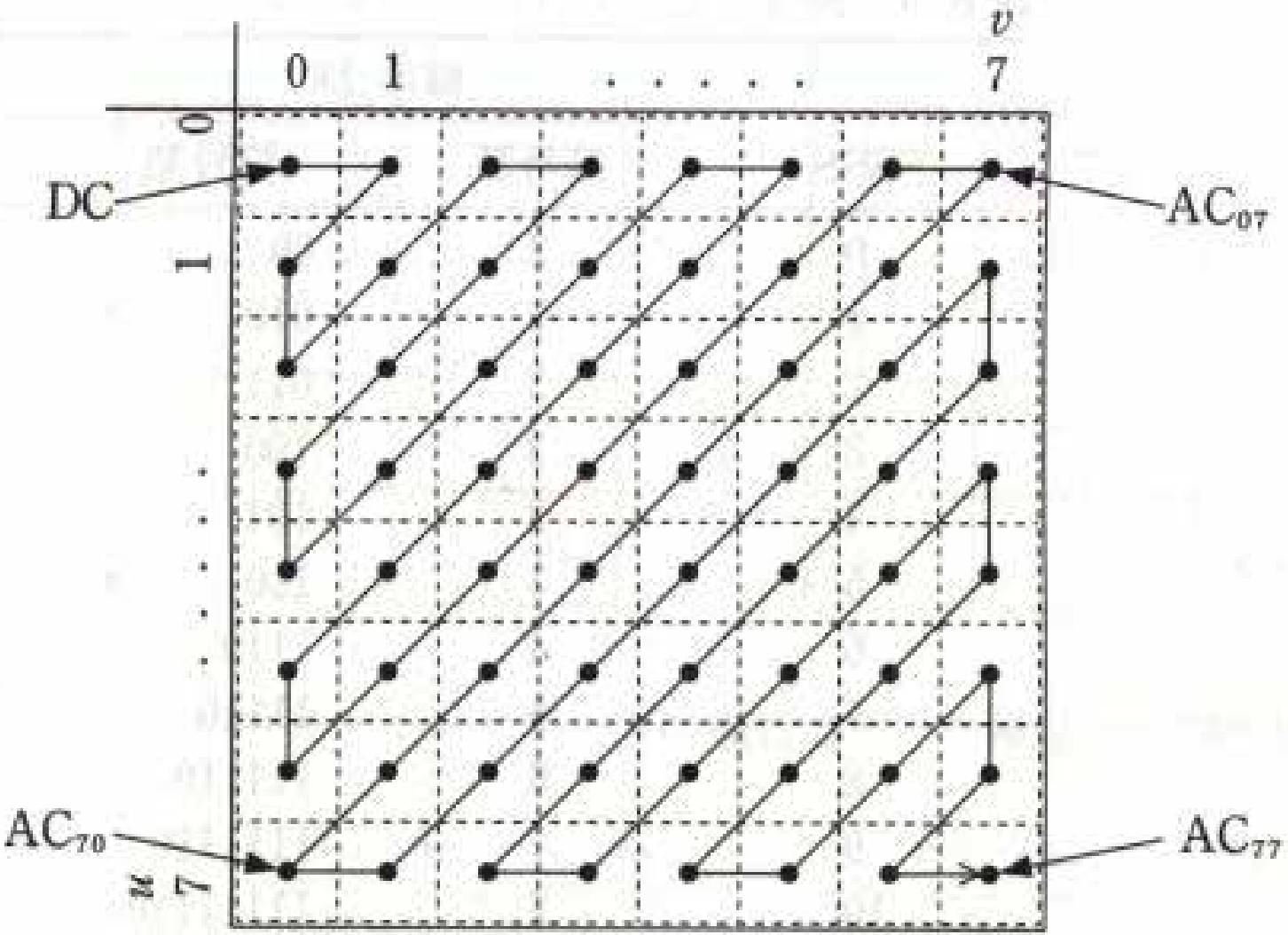
出典：© 1993 ITU-T.

表8.2 色差量子化マトリックス Q_{uv} [367]

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

出典：© 1993 ITU-T.

各ブロックDC係数をジグザクスキャンし符号化（ハフマン符号、ランレングス符号）



動き補償：フレーム間での圧縮

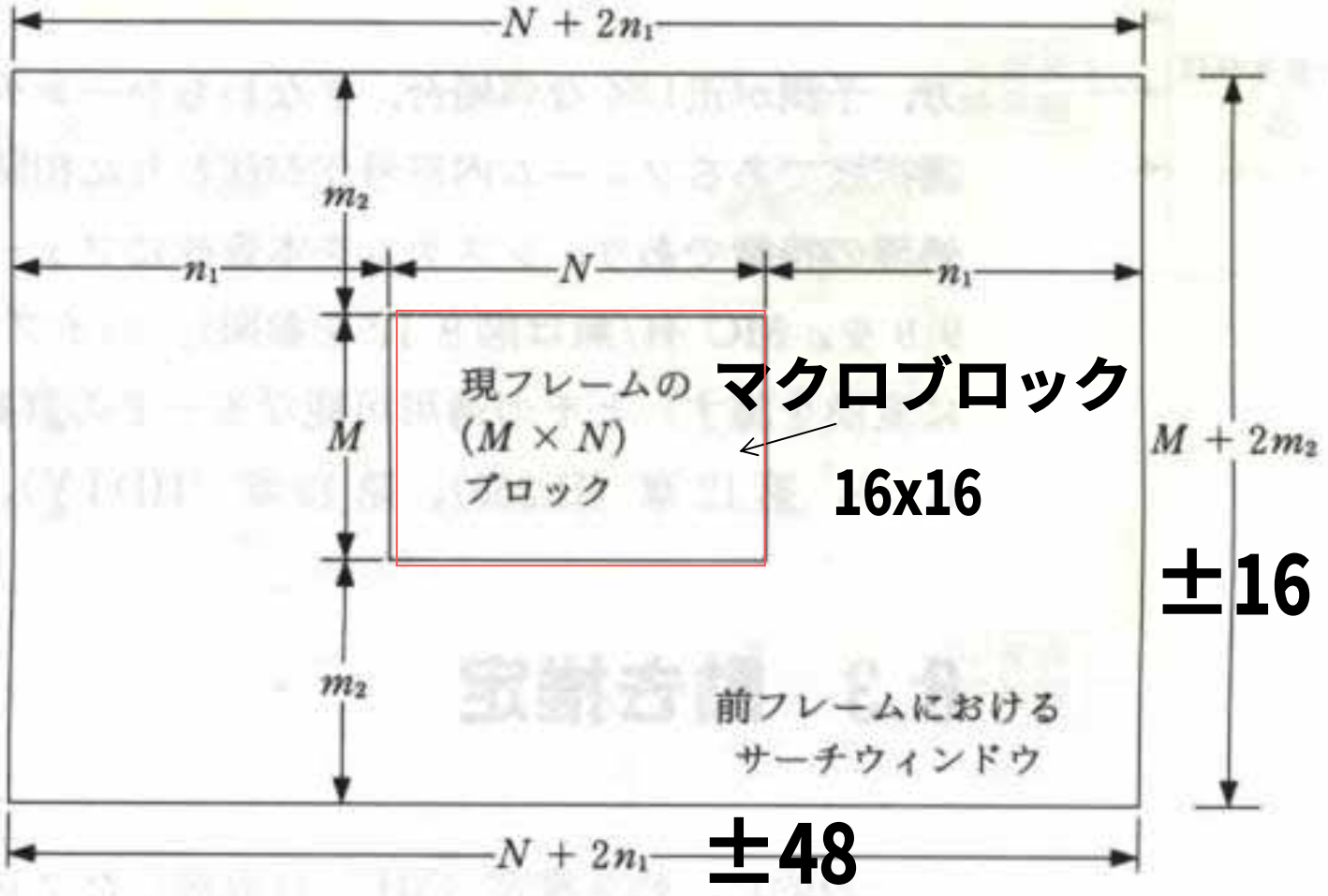
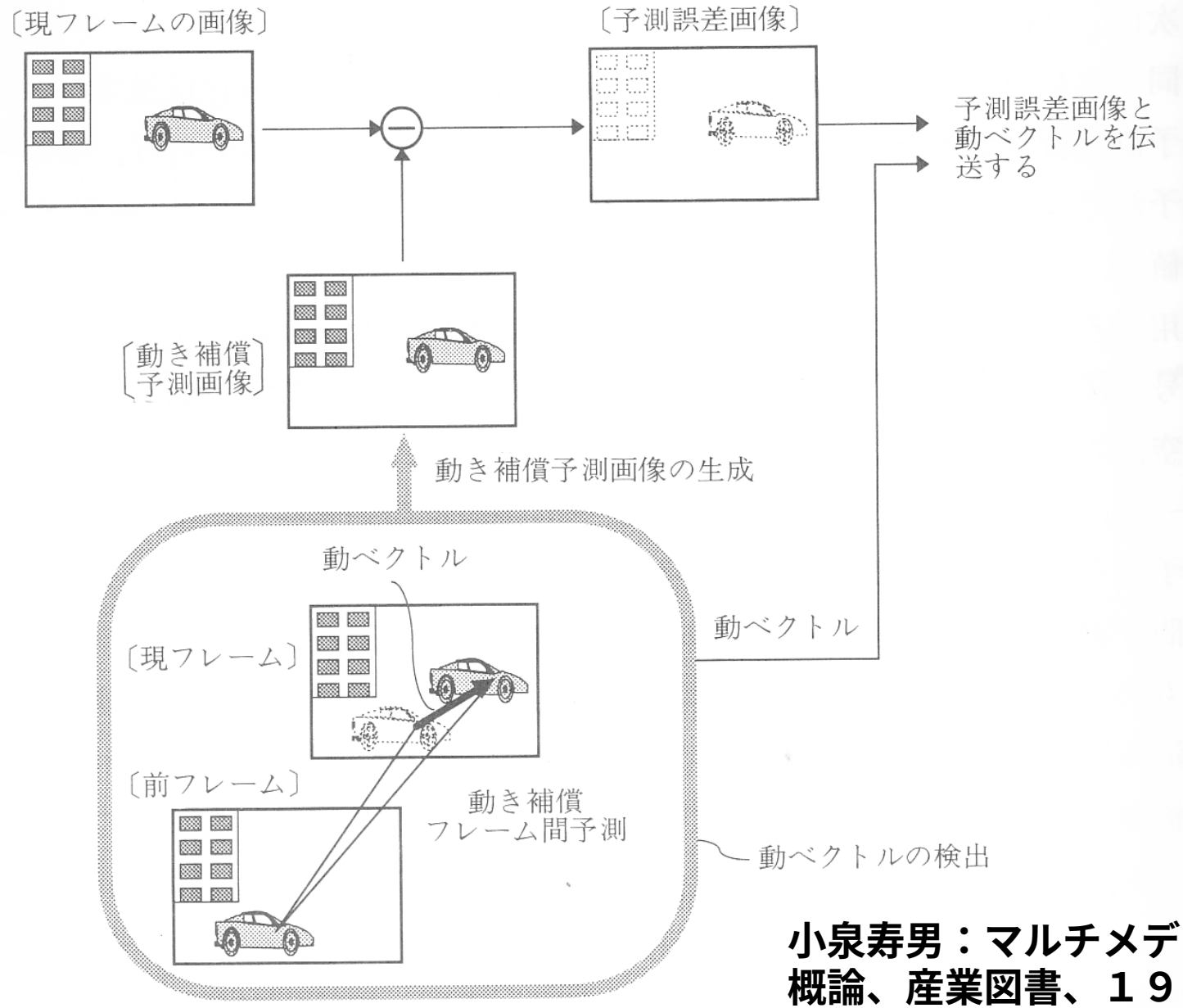
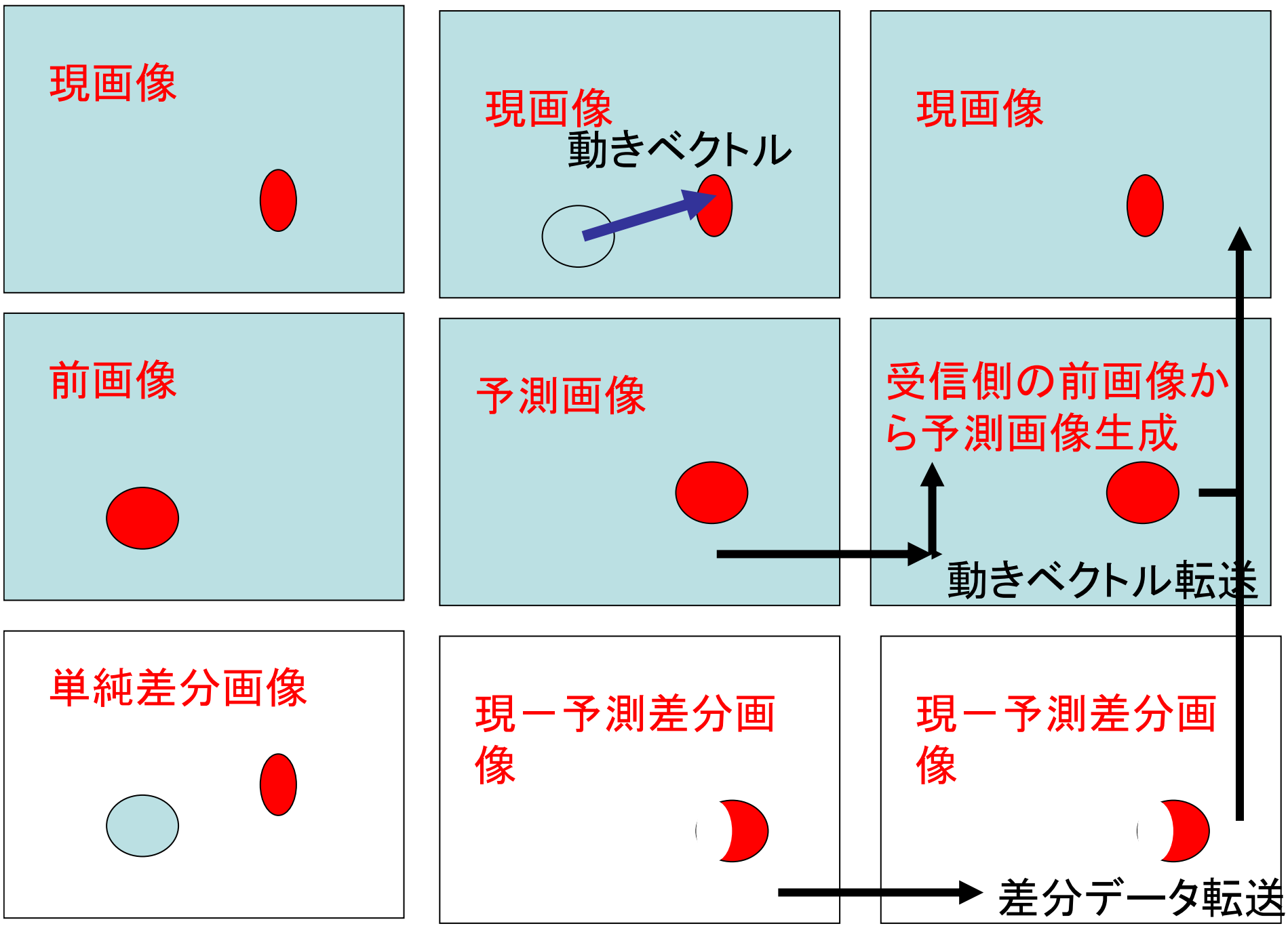


図6.5　前フレームの$(M + 2m_2) \times (N + 2n_1)$のサーチウィンドウを用いた，$(M \times N)$ブロックの動き推定（全探索）．動きベクトル範囲はフレーム間距離あたり水平 ± n_1 画素，垂直 ± m_2 ラインである．



小泉寿男：マルチメディア概論、産業図書、１９９７

図 4.10　動き補償フレーム間予測の原理


現画像
現画像
動きベクトル
現画像
前画像
予測画像
受信側の前画像から予測画像生成
動きベクトル転送
単純差分画像
現－予測差分画像
現－予測差分画像
差分データ転送

- 動き補償での演算量
- マクロブロック当たり：±48,16画素領域
- 16x16x97x33x3=2458368演算

マクロブロックサイズ　比較演算数

- 毎秒当たりのマクロブロック処理数：
- 45x30x30=40500
- 演算量：1000億/秒：100G演算/秒

フレーム数/秒

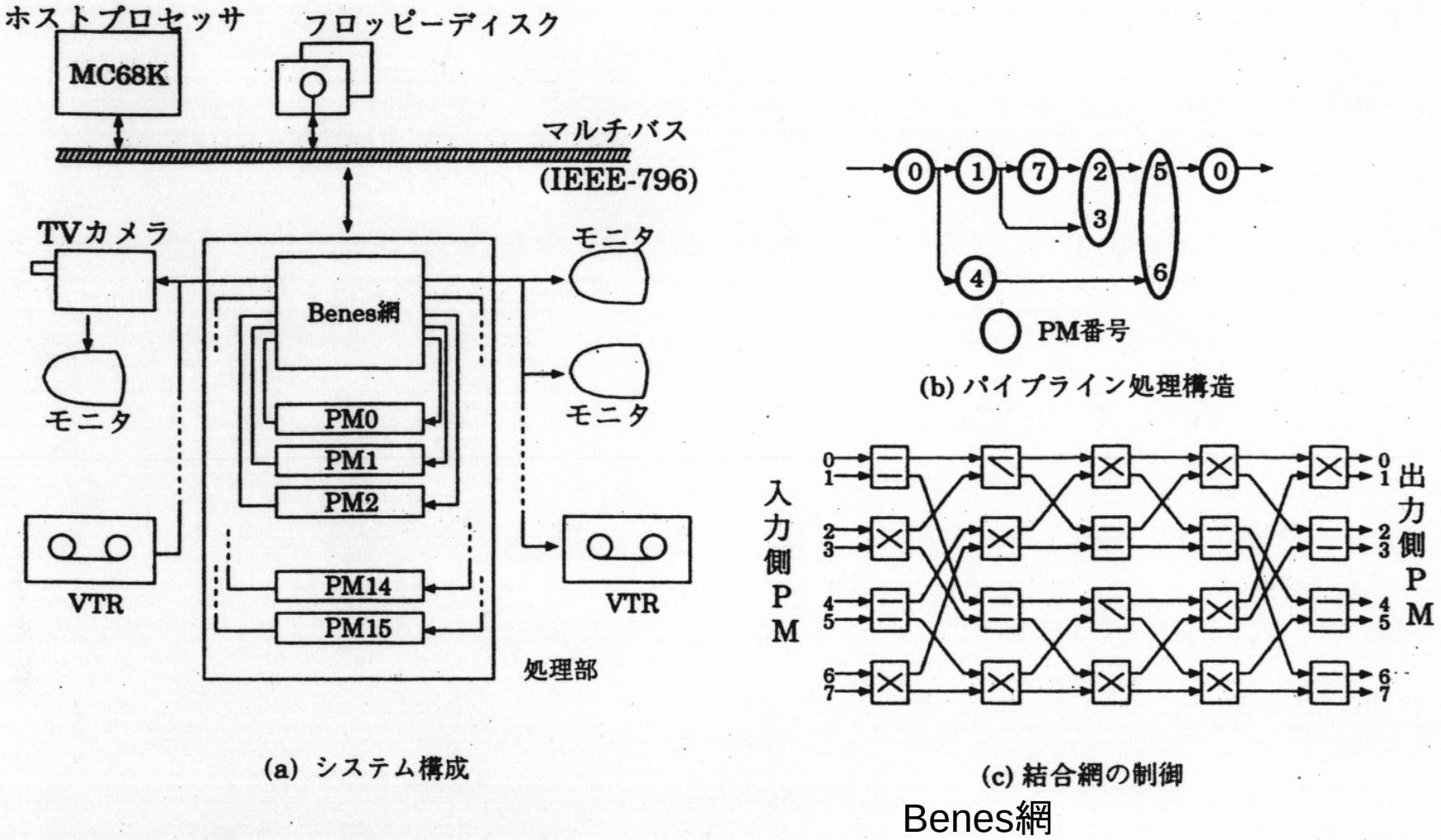


図 6. 11　韋駄天の構造

(佐々木ほか：構造可変型ビデオレート画像処理システム「韋駄天」，情報処理学会コンピュータビジョン研資，37-1(1985))

６．３　３次元グラフィックス処理

グラフィックス処理の要件

①高速性

②扱える図形の複雑さ

③現実感表出の度合

レンダリング（rendering）技術

図形を「美しく」表出する技術

サーフィスレンダリング

ボリュームレンダリング

6.3.1　サーフィスレンダリング

（１）基本方式

物体表面：

多角形（ポリゴン）、２次曲面、自由曲面で近似

基本処理

①隠れ面消去

②シェーディング

グローシェーディング

多角形の各頂点の輝度値から多角形内部の輝

度値を補間

フォンシェーディング

各頂点の法線ベクトルから多角形内部の法線ベクトルを補間し、輝度計算

輝度：

面の法線ベクトルの方向、光源の位置、面の材質などで決定

- 光源の種類：並行光線、点光源、多光源、蛍光灯のような立体（面）光源
- 間接光：壁や床からの散乱光

・付影処理：光線があたらない場所での影付け処理

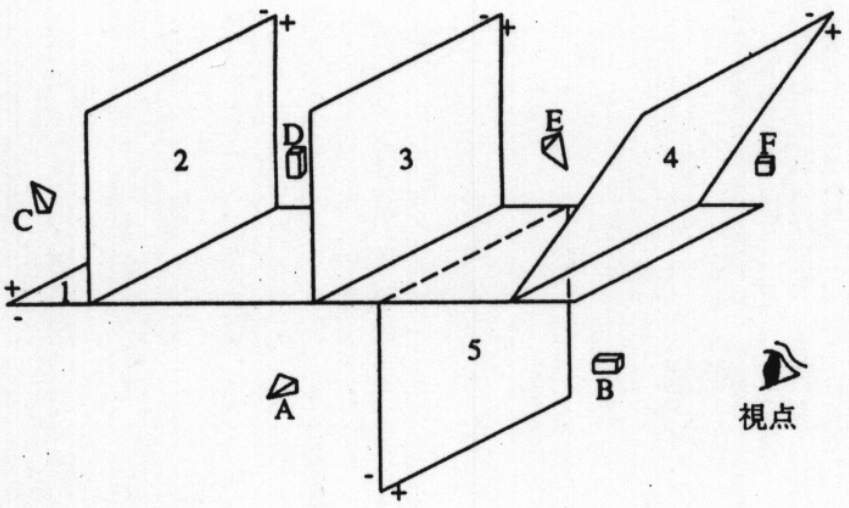
・アンチエイリアス処理：図形の境界線でギザギザが現れる
と見づらい

（２）隠面消去基本アルゴリズムと並列処理

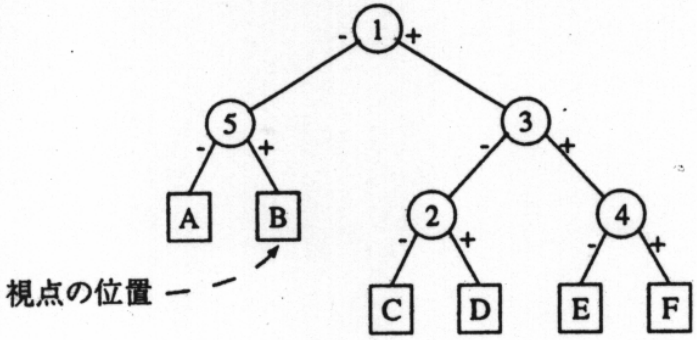
①優先順位リスト法

粗い図形集合の高速リアルタイム処理

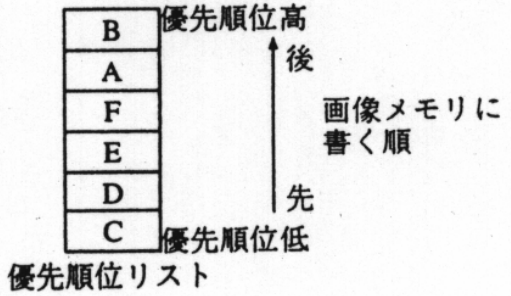
フライトシミュレータなど



(a) 図形と分割平面、および視点



(b) 分割平面についての 木構造



(c) 優先順位リスト

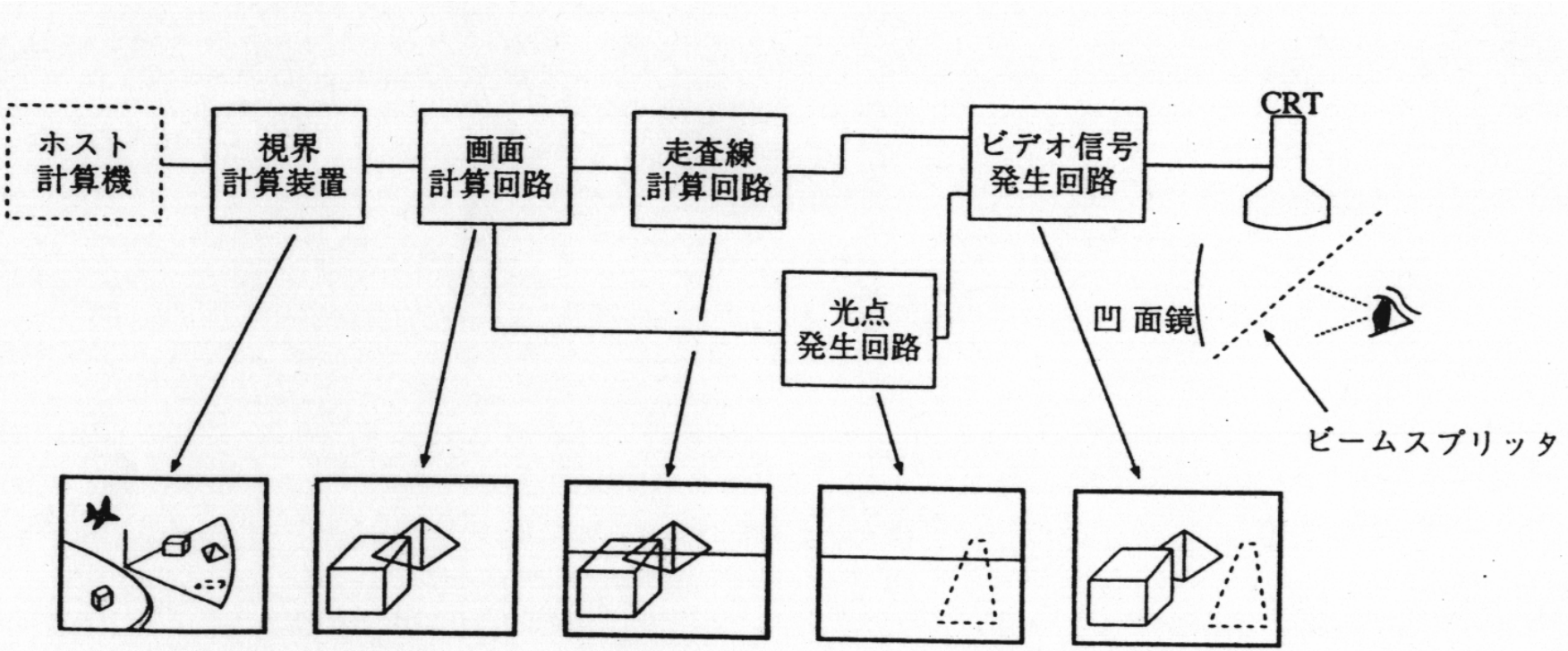


図 6. 13　フライトシミュレータ
（麻生和夫：フライトシミュレータ用計算機合成映像，テレビジョン学会技術報告，IPD 65-5（1982））

②スキャンライン（SLA）法

メモリが高価であったときの方式

インクレメンタル処理

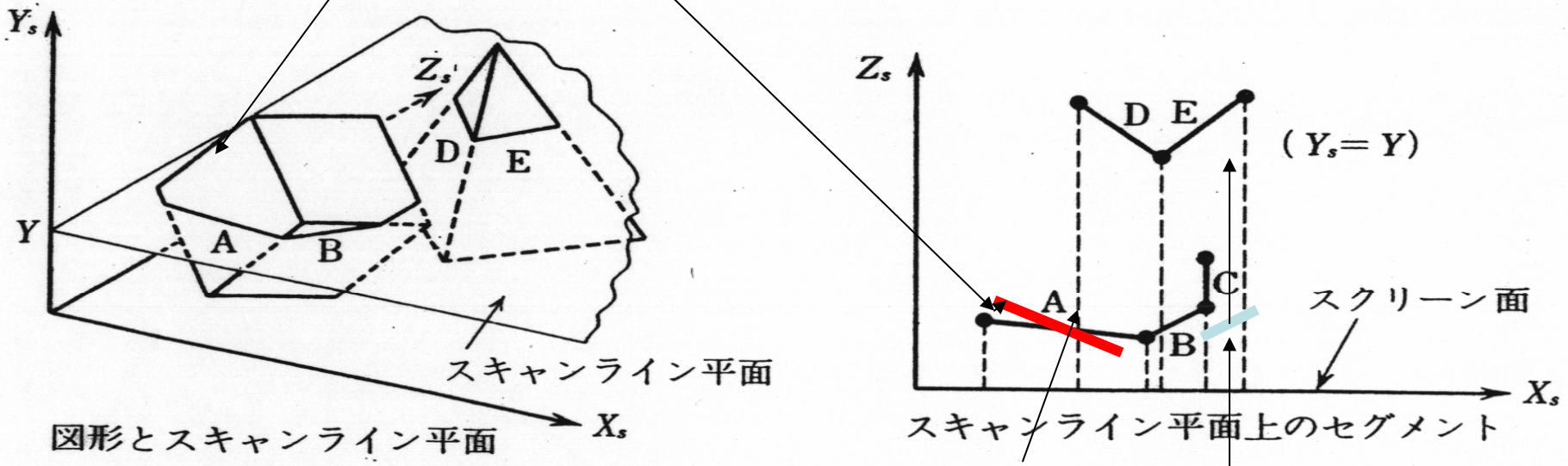
スキャンラインコヒーレンス利用

スキャンライン間の類似性

半透明物体の表示が可能

$(X-X_0)/a=(Y-Y_0)/b=(Z-Z_0)/c=t$

$dX=(a/b)dY, dZ=(c/b)dY$



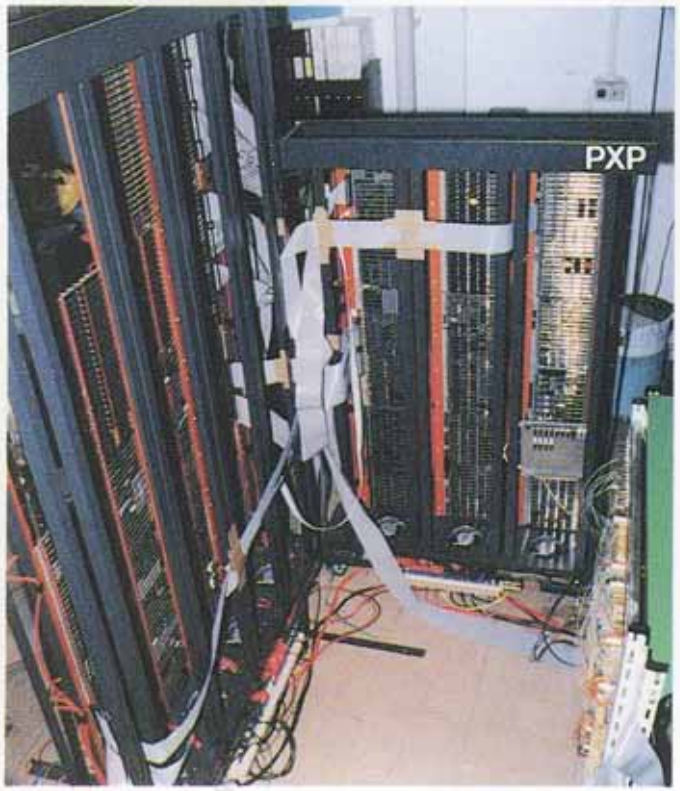
インクリメンタル処理

B半透明のときEとブレンド

図 6. 14　スキャンライン法

EXPERTS：SLP2台、PXP：4台

新実、富田、萩原他、電子情報通信学会論文誌、J-71D, 8,pp.1446-1453、1988



③Zバッファ法

単純なBrute　Force法：メモリ安価時代

半透明物体表示不可

Ｚバッファ：奥行き情報を１枚保持

３次元グラフィックス端末における

パイプライン構造

- 幾何プロセッサ（GP）
  ポリゴンの座標計算
- セグメントプロセッサ（ＳＰ）
  セグメント補間
- ピクセルプロセッサ（ＰＰ）
  Ｚバッファと比較

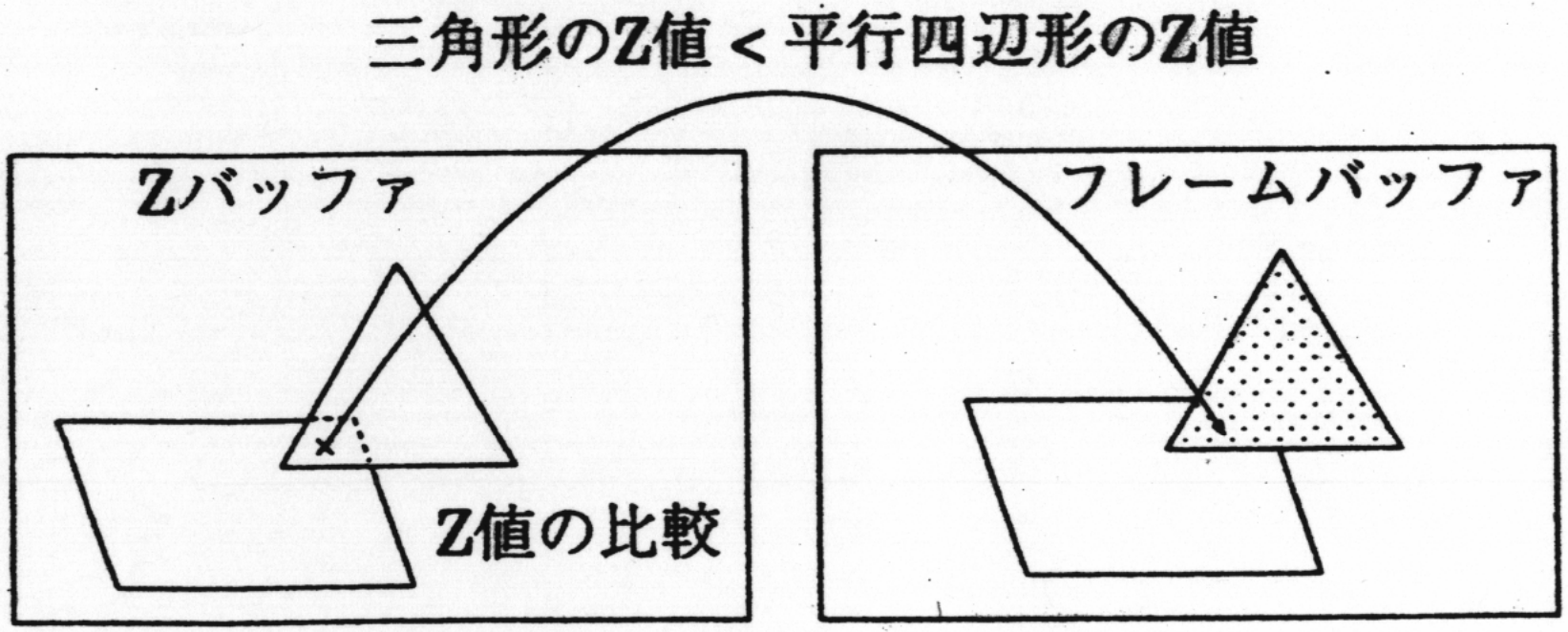


図 6. 15　Zバッファ法

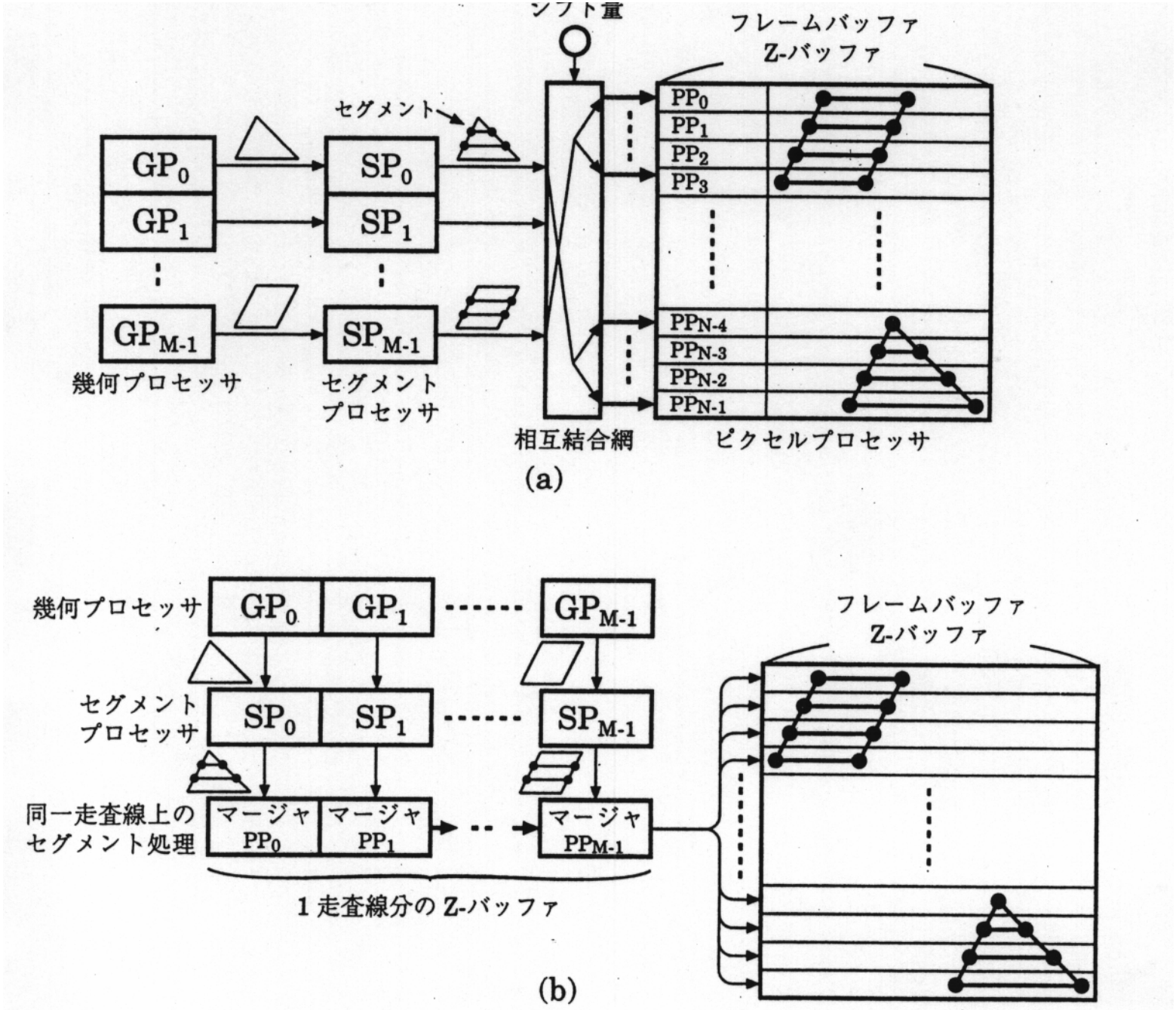


図 6.16　Zバッファ法による並列処理

④視線探索法

- **基本方式**

  １次視線と物体表面との交差判定

  透過視線と反射視線

  輝度計算

  $$I=I_a+k_d n \cdot L_i+k_s L_r+k_t L_t$$

  ただし、

  I_a **：周囲光**

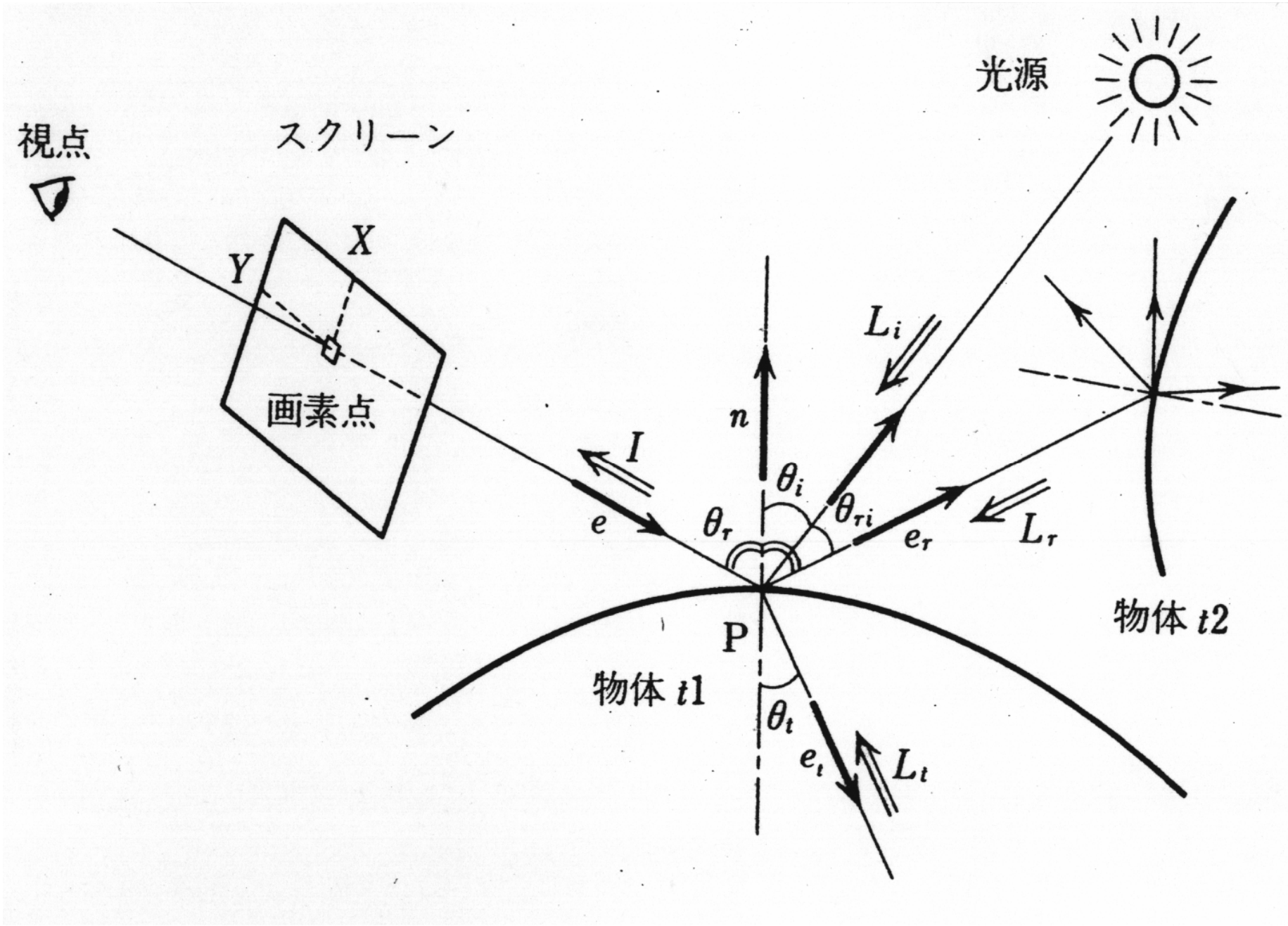
  n **：面の法線ベクトル**

・交差判定に時間がかかる

　　簡単な構造の外接物体の導入

・反射や屈折処理可能

・非常に美しいレンダリング可能


光源
視点
スクリーン
X
Y
画素点
L_i
n
I
θ_i
θ_{ri}
e
θ_r
e_r
L_r
物体 t2
P
物体 t1
θ_t
e_t
L_t

L_i：光源からの入射光の強度（ベクトル）

k_s：正反射率、k_t：透過率

L_r：正反射成分、L_t：透過成分

・並列処理

(1)ピクセル並列

各プロセッサ：データベースのコピー

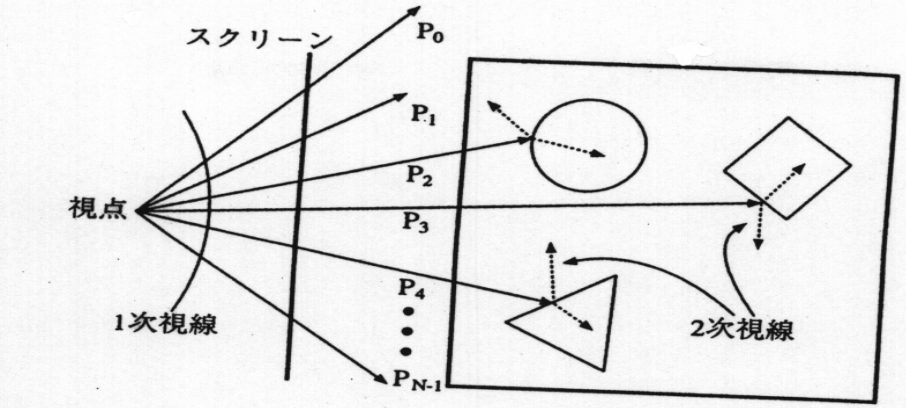
(2)空間並列

３次元空間の分割法

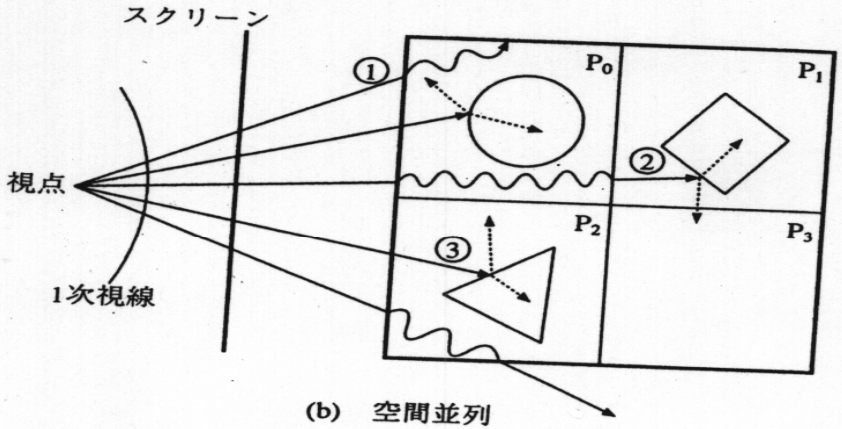
オクトリー法（octree）

空間等分割法

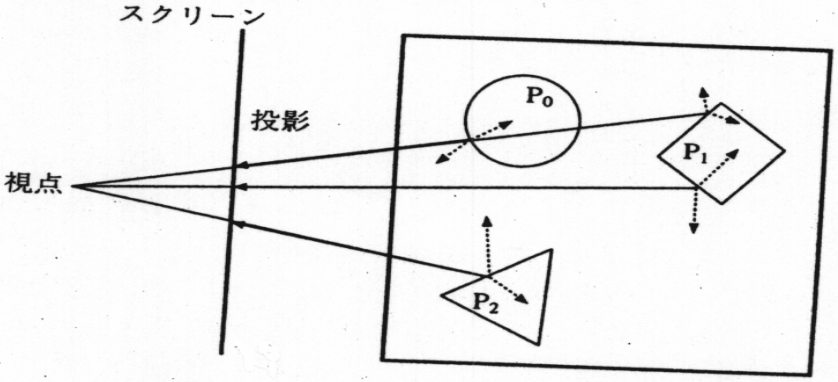
(3)物体並列



(a) ピクセル並列



(b) 空間並列



(c) 物体 並列

図 6.18 視線探索法の並列化

6.3.2ボリュームレンダリング

医療画像や科学技術計算結果の３次元表示

問題点

①ボリュームデータは３次元メモリが必要

処理時間が大

②３次元配列としてディジタル化

回転やズーム処理で歪

③法線ベクトルなど物体表面の関する情報なし

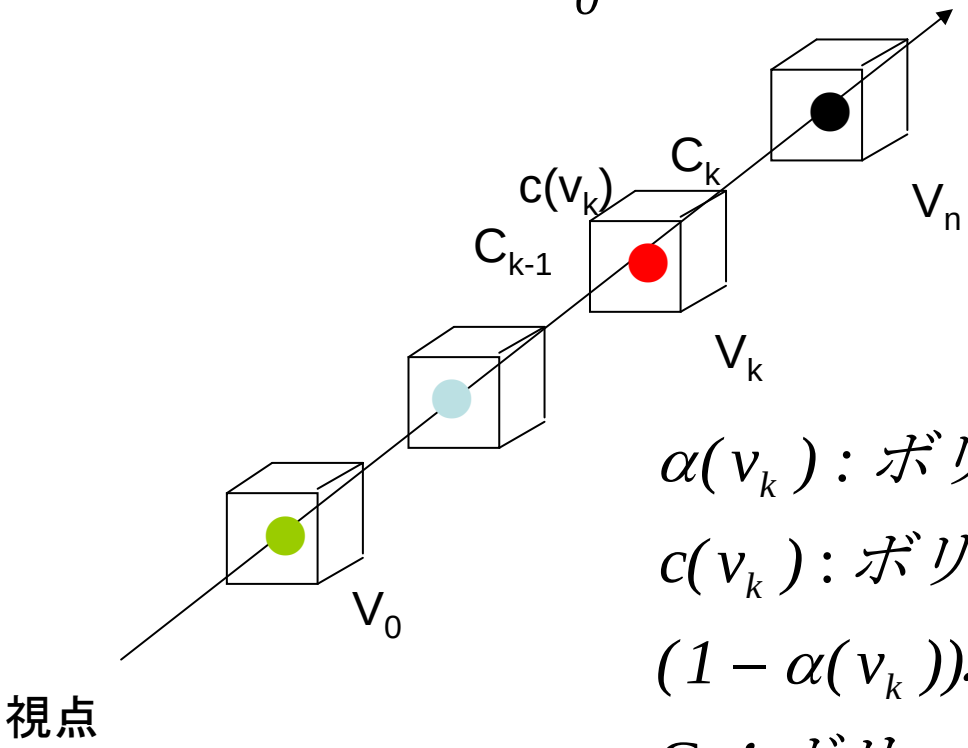
近接ボクセル値より計算する必要

アルゴリズム

$$P = \sum_{i=0}^{n} \alpha(v_i)c(v_i)\prod_{j=0}^{i-1}(1-\alpha(v_j))$$

$$C_{k-1} = \alpha(v_k)c(v_k) + (1-\alpha(v_k))C_k$$

$$C_0 = P$$



$\alpha(v_k)$: ボリューム k での不透明度

$c(v_k)$: ボリューム k での色

$(1-\alpha(v_k))$: ボリューム k での透明度

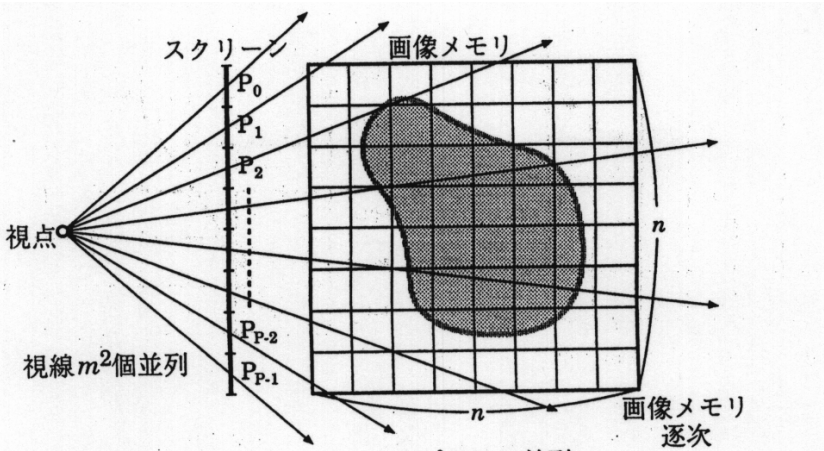
C_k: ボリューム k までの累積色

並列処理方式

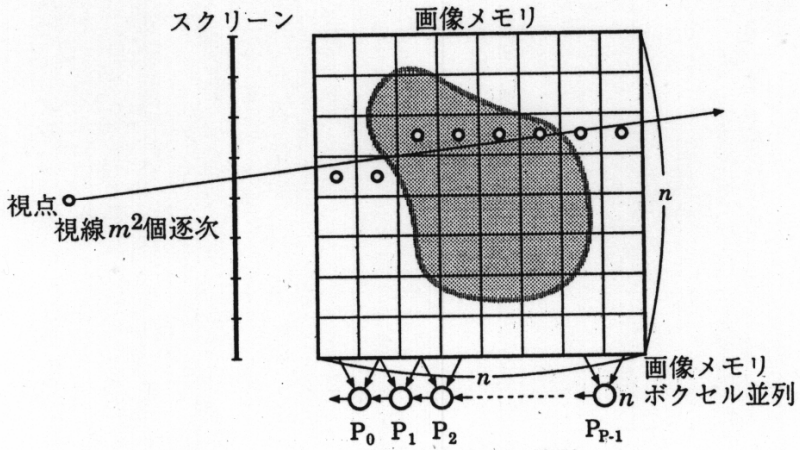
①ピクセル並列

②ボクセル並列

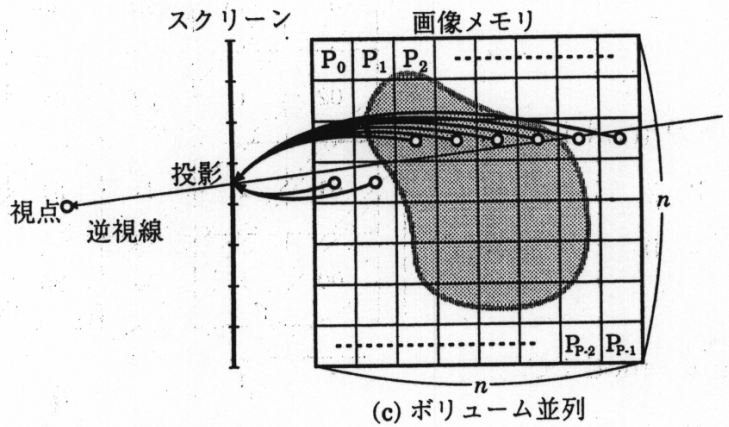
③ボリューム並列



(a) ピクセル並列



(b) ボクセル並列



(c) ボリューム並列

①ピクセル並列
ボリュームメモリをコピー：単純
分散共有メモリ（キャッシュメモリ）
リアルタイム性が保証できない
②ボクセル並列
３次元直交メモリが必要
リアルタイム性が保証できる

- ReVolver（京都大学）
  ボクセル並列
  ３重化３次元メモリ
  主軸サンプリング

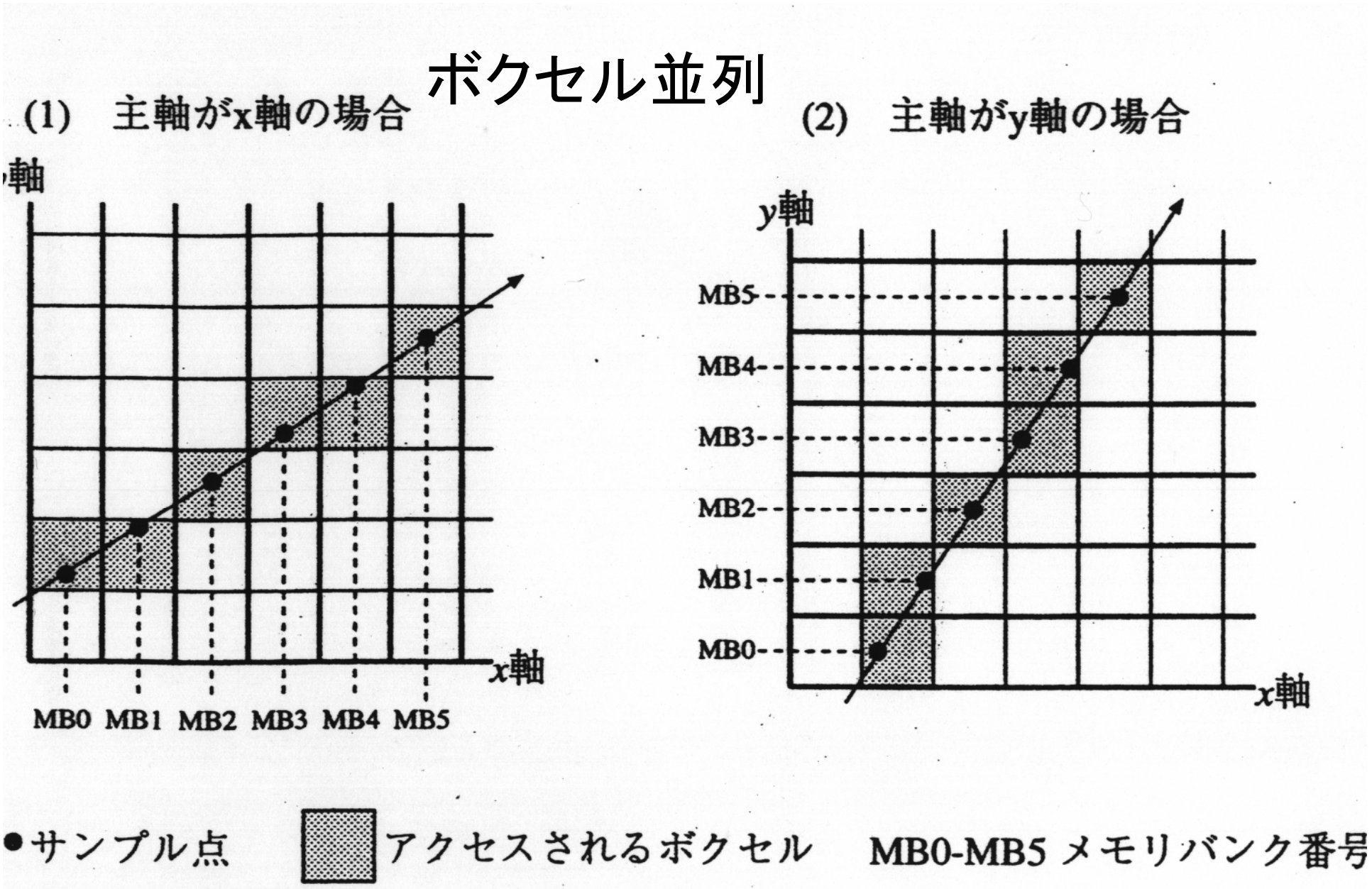
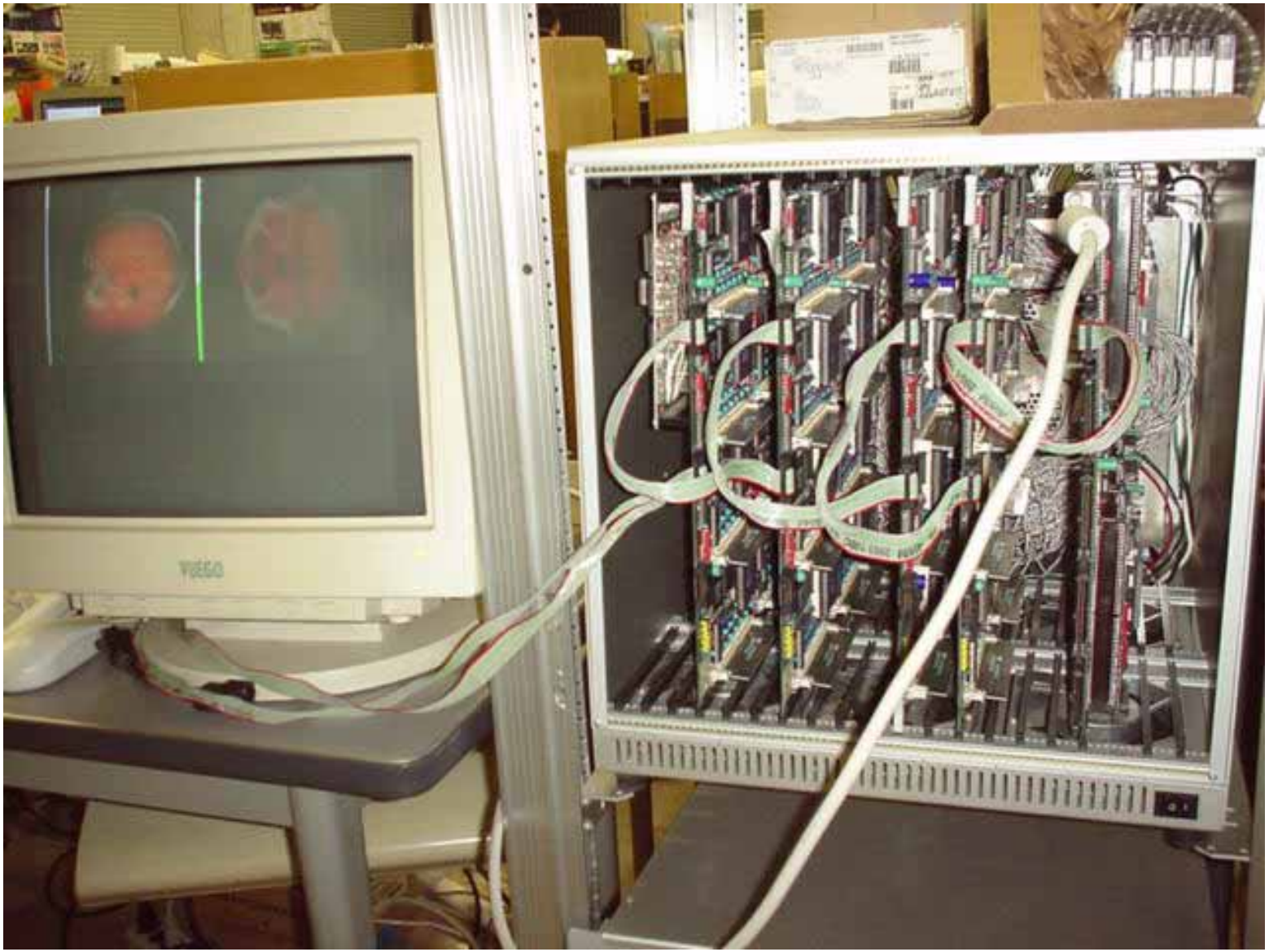


図 6.21 ReVolver のメモリ構成

ReVolver C-40：對馬、明石、富田ほか：情報処理学会論文誌、36,7,pp.1709-1718,1995



Cube（ニューヨーク大.Kaufman）

- Cube

$$k=(x+y) \bmod n \qquad （３４）$$

第１行（y=1**）**

k=(x+1)mod 4**で**

k	0	1	2	3
x	3	0	1	2

第2行（y=2）

k=(x+2)mod 4で

k	0	1	2	3
x	2	3	0	1

第1列（x=1）

k=(1+Y)mod 4で

k	0	1	2	3
y	3	0	1	2

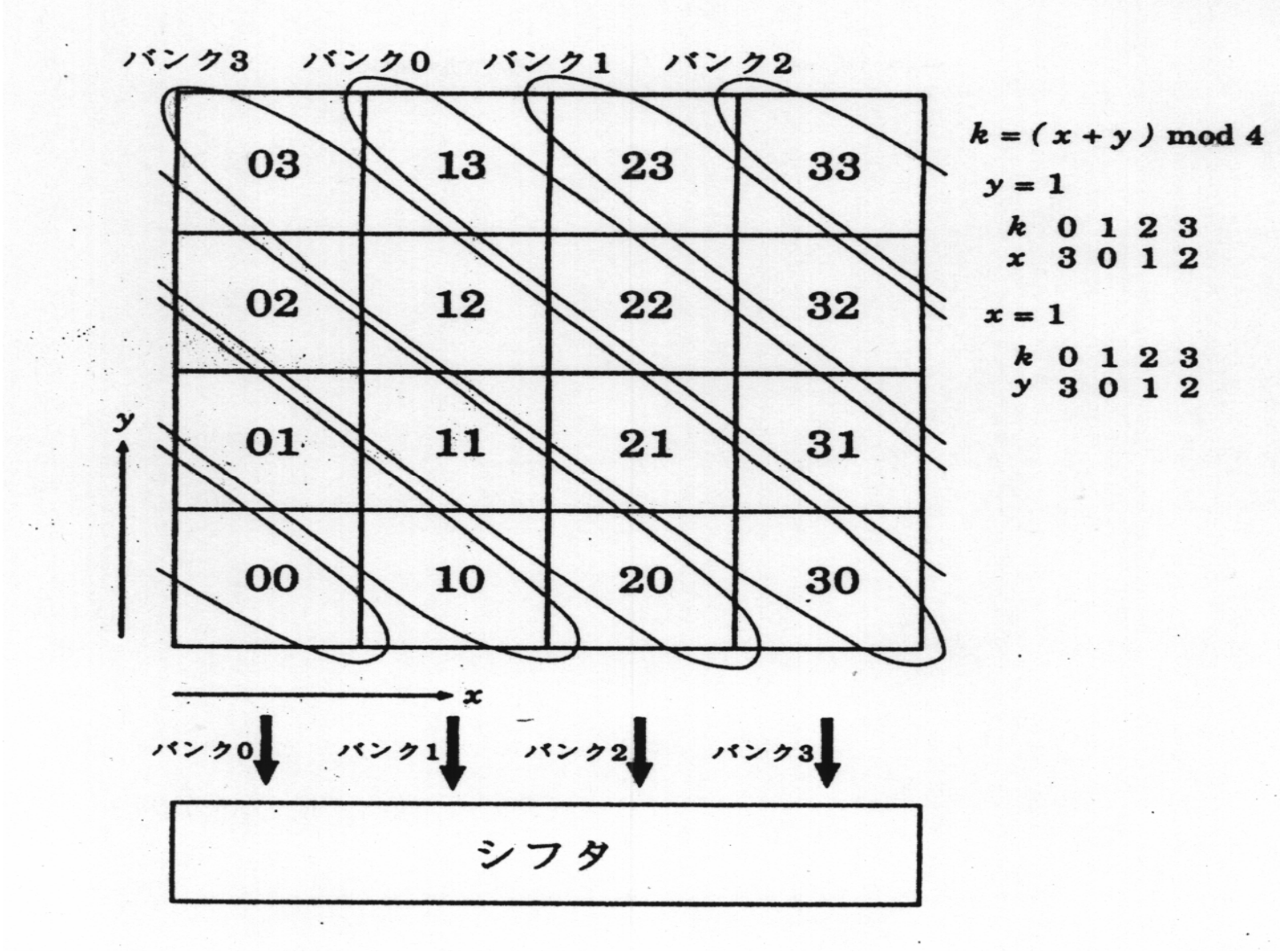
X方向、Y方向に水平な視線：同時読出し 直交メモリ

任意方向の視線：一旦２次元メモリの内容を回転

Cube：平行投影向き（視点が無限遠点にあり、すべて

の視線が平行となっている）

遠近法（視点からの視線が平行でない）は困難



Cubeの３次元メモリ

$$k=(x+y+z) \bmod n \qquad (35)$$

x,y,z方向に並行な視線データ：同時アクセス

nバンク構成

$0 \le x,y,z,k \le n-1$

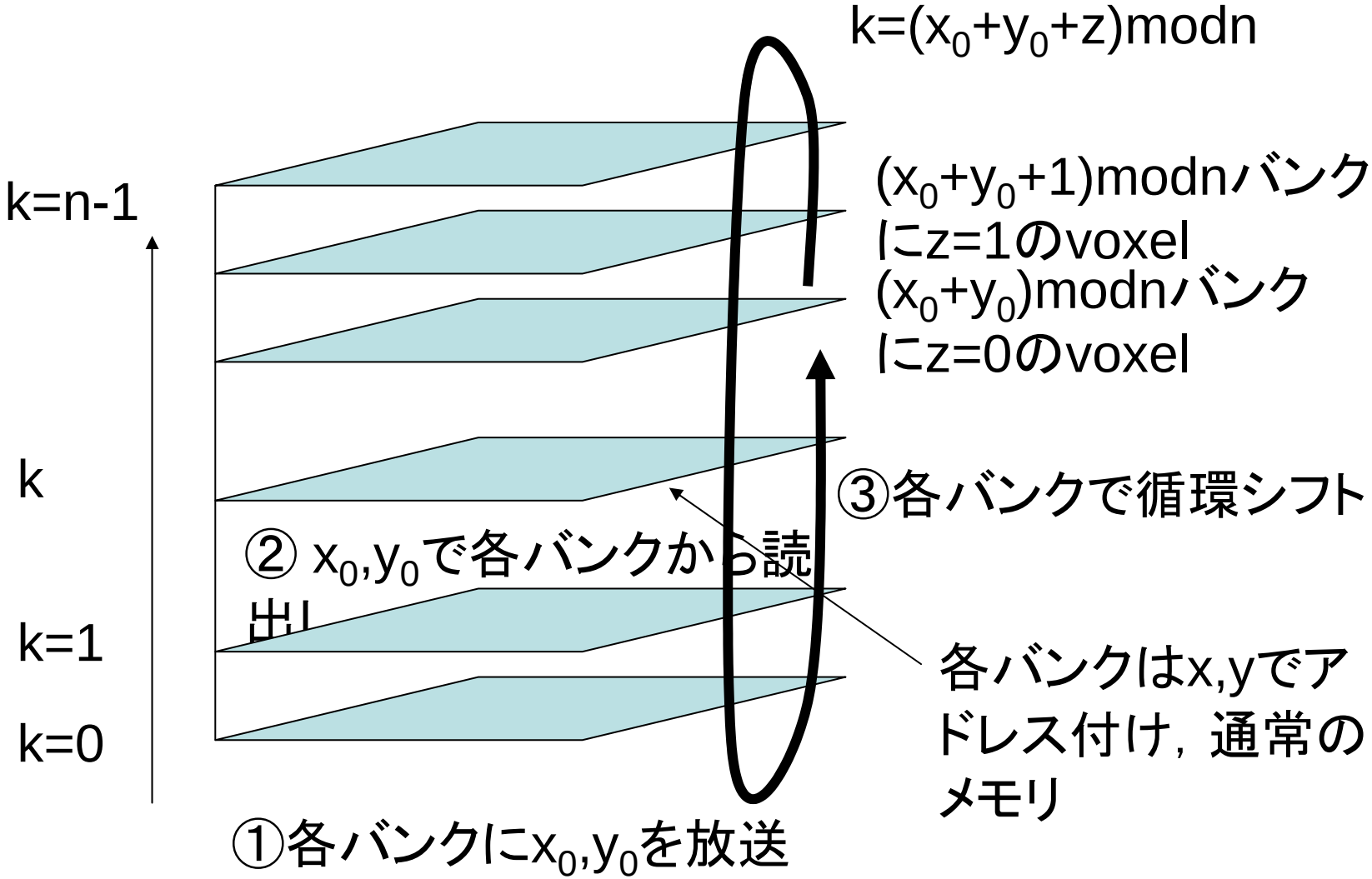
各バンク：

n^2voxelsの通常のメモリ

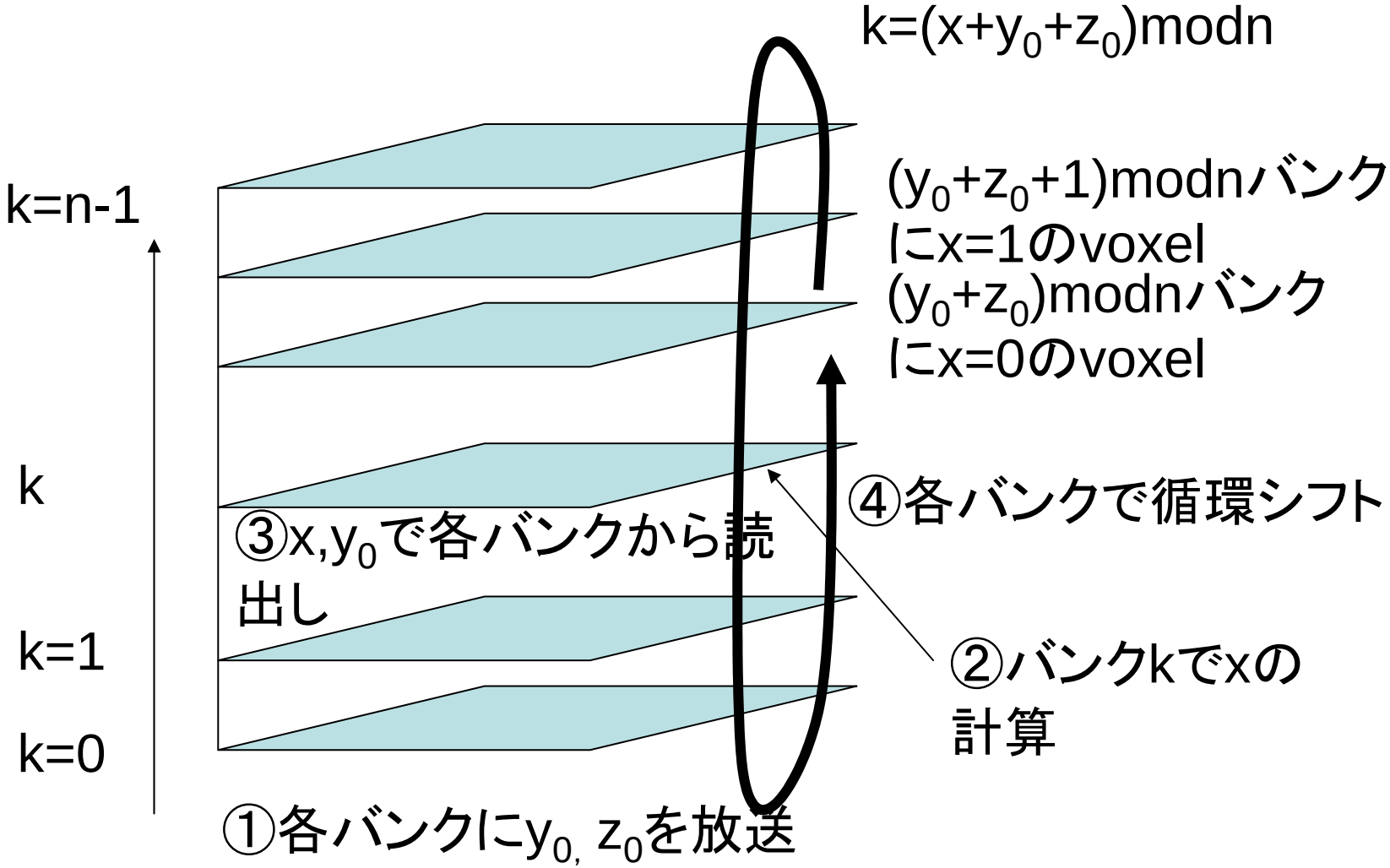
x,yでアドレス付け

k=(x+y+z)modnで決まるzが求まり，そのVoxel値が求まる

Z軸に平行なVoxelへの並列アクセス

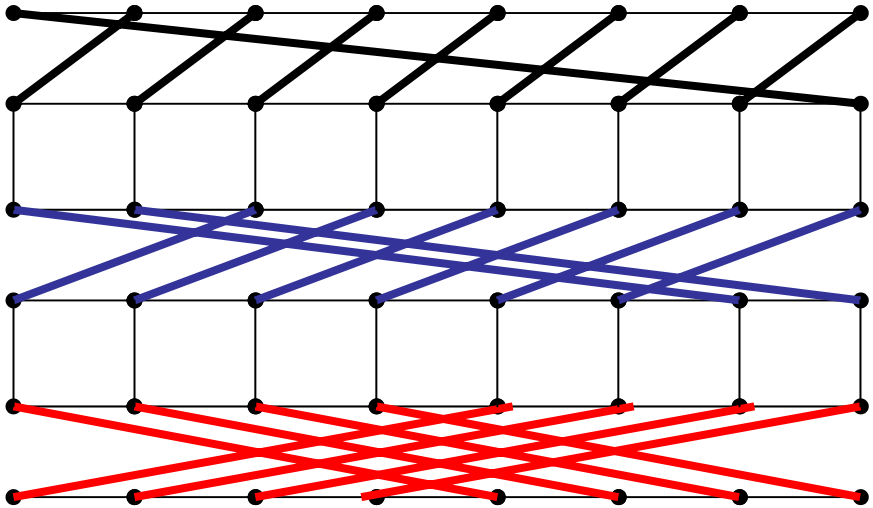


X軸に平行なVoxelへの並列アクセス



バレルシフタ

入力



出力

シフト量
（$a_2a_1a_0$）

a_0=1

a_1=1

a_2=1

Logn段

x,y,z方向に平行でない視線データ

物体データを3次元メモリ内で回転

x,y,z方向に平行な視線でみる

物体データの回転：x,y,z軸周りでの回転を合成

$$X=\cos\theta x+\sin\theta y \qquad (36)$$

$$Y=-\sin\theta x+\cos\theta y$$

（x,y）を固定したとき、

各バンクk：

式（３５）に対応したｚ値

式（３６）：ｚの値に無関係に定数

バンクｋの(x,y,z)に存在するデータの同時に読み出し

バンクk' {k'=(X+Y+z)modn} に一斉にバレルシフタで転送

シフト桁数（X+Yに対応する）：すべてのバンクについて同一

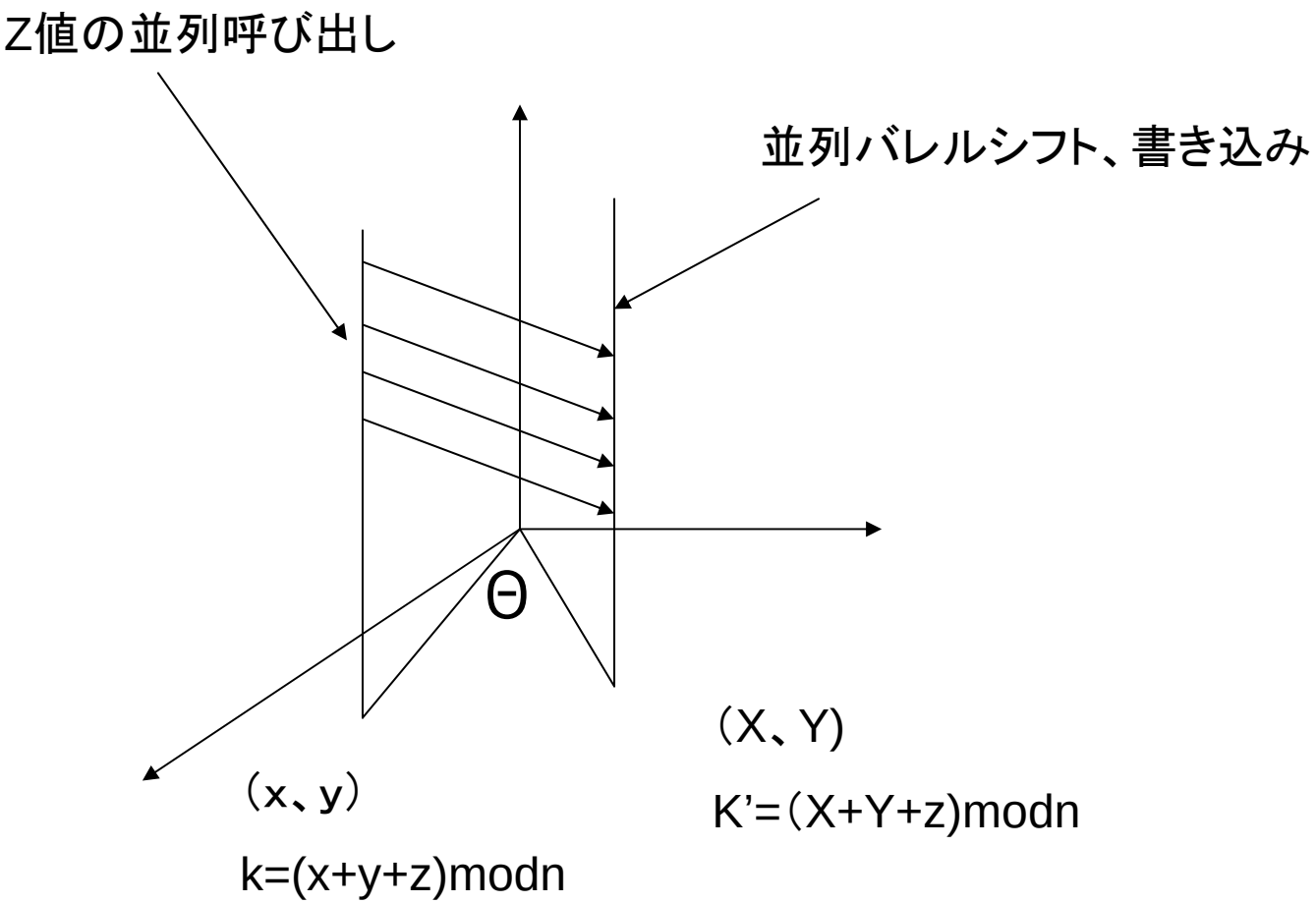
すべての（x,y）について実行

ｚ軸周りの回転

バレルシフタの時間：logn

１つの軸当り：n^2logn時間必要

z軸周りでのθ回転



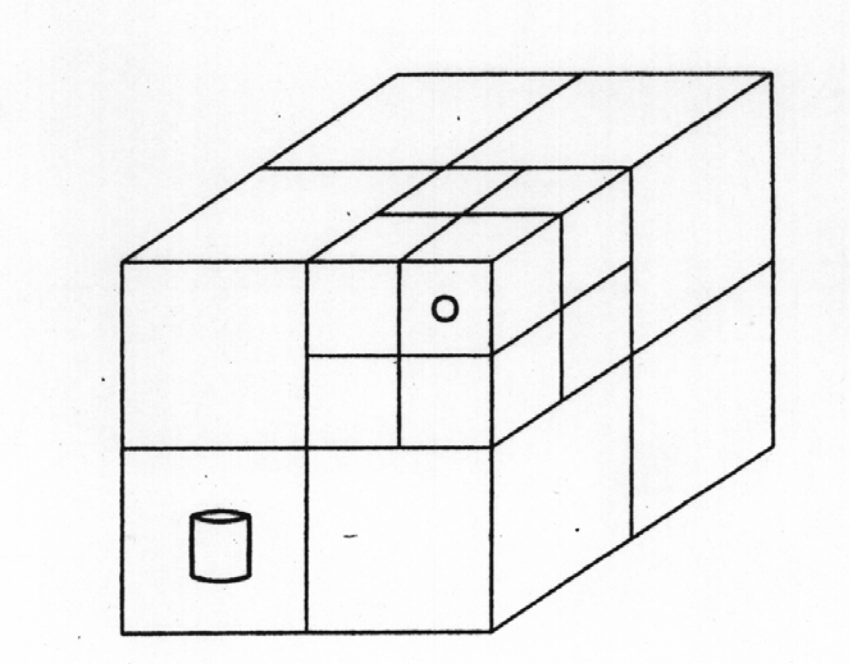
③ボリューム並列

視点から見たボクセルの奥行き情報

　によるソート

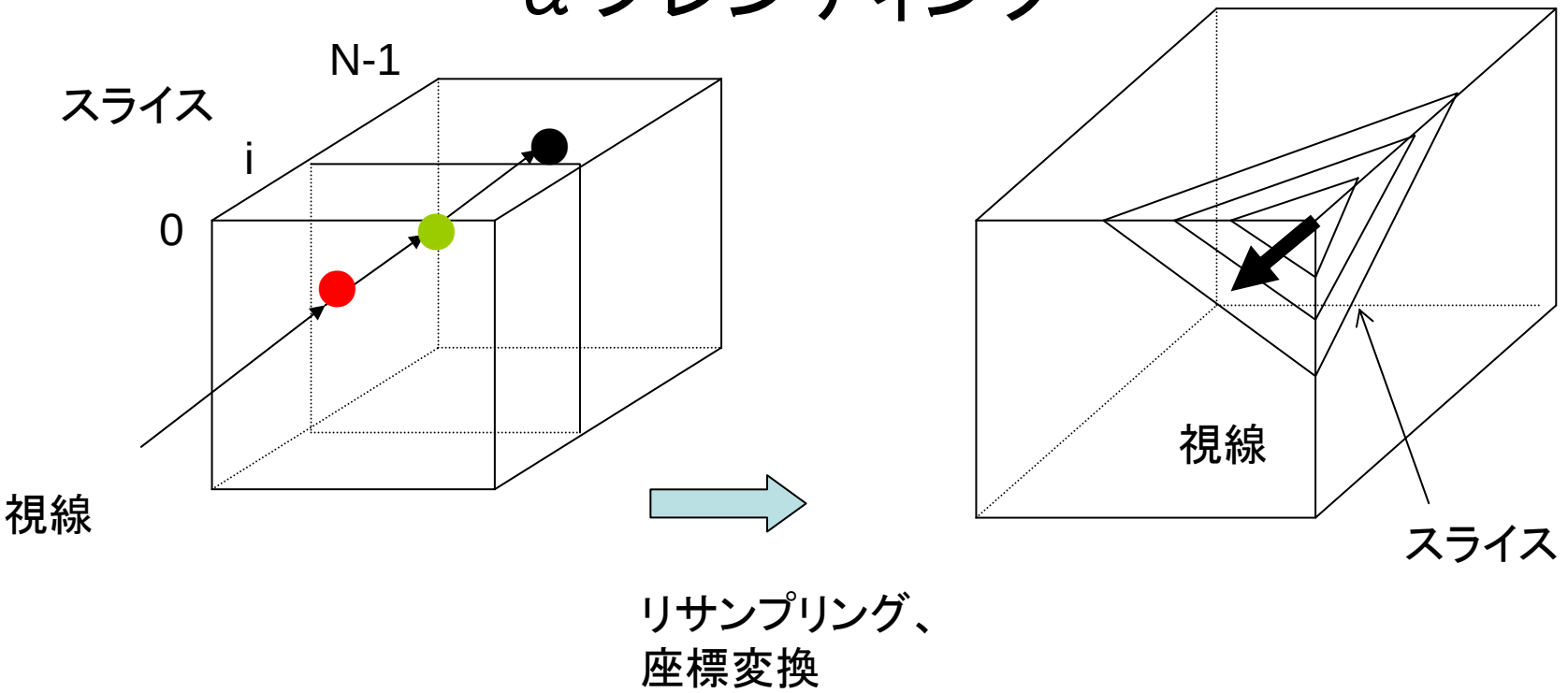
視点変更：ボクセル座標変換

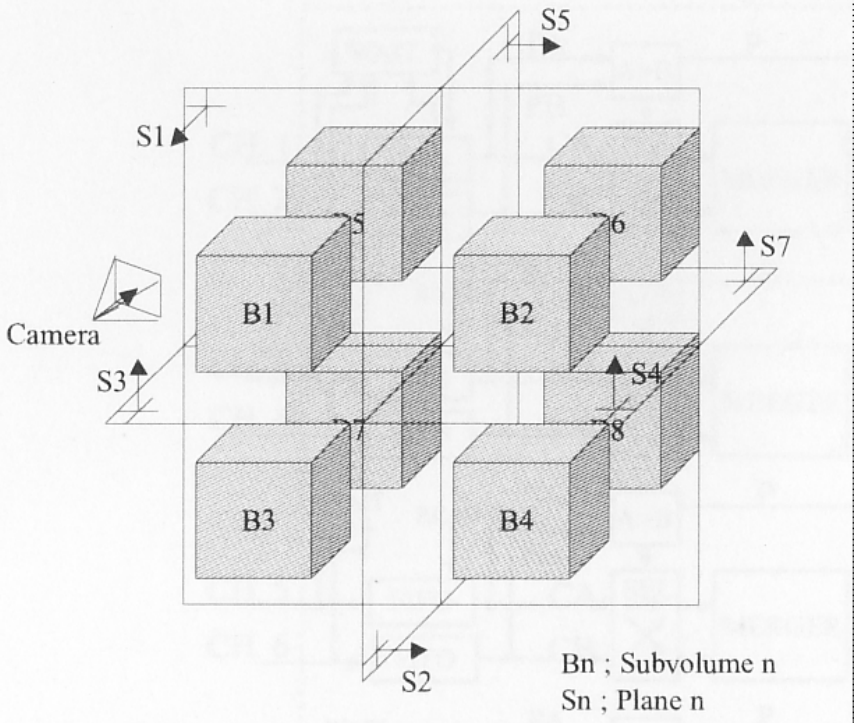
空間分割：空間等分割、オクトリー分割



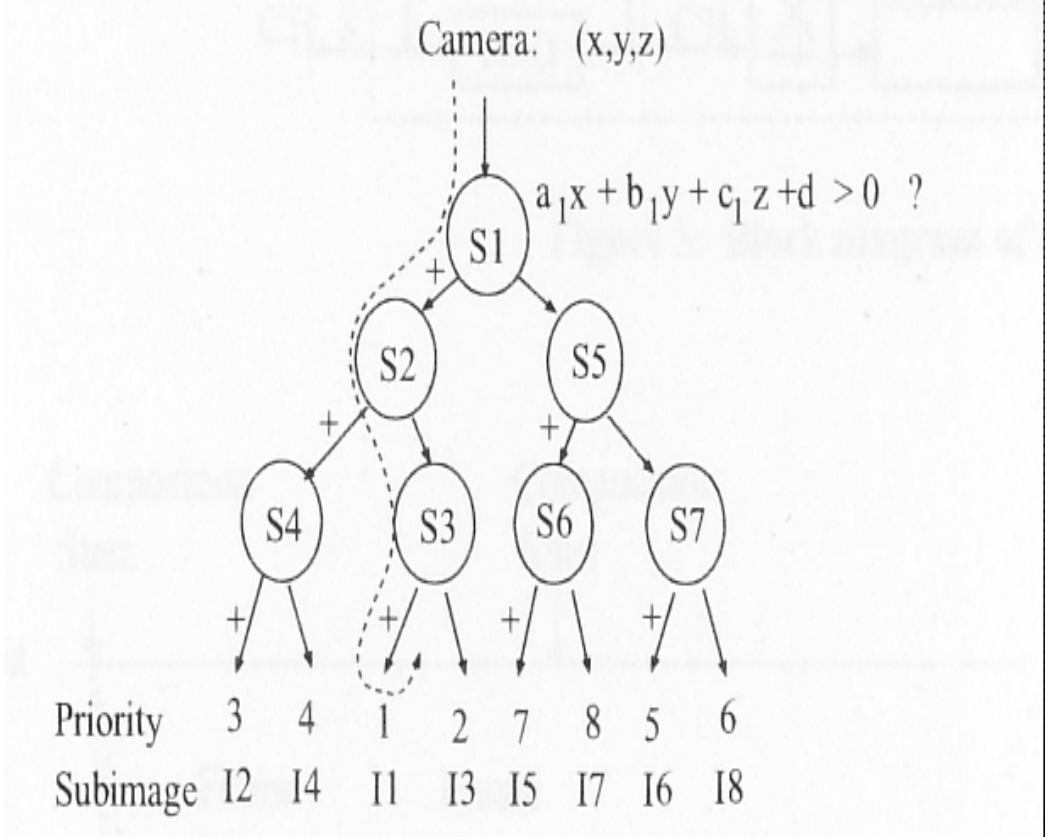
汎用グラフィックスカードを用いたボリュームレンダリング

αブレンディング

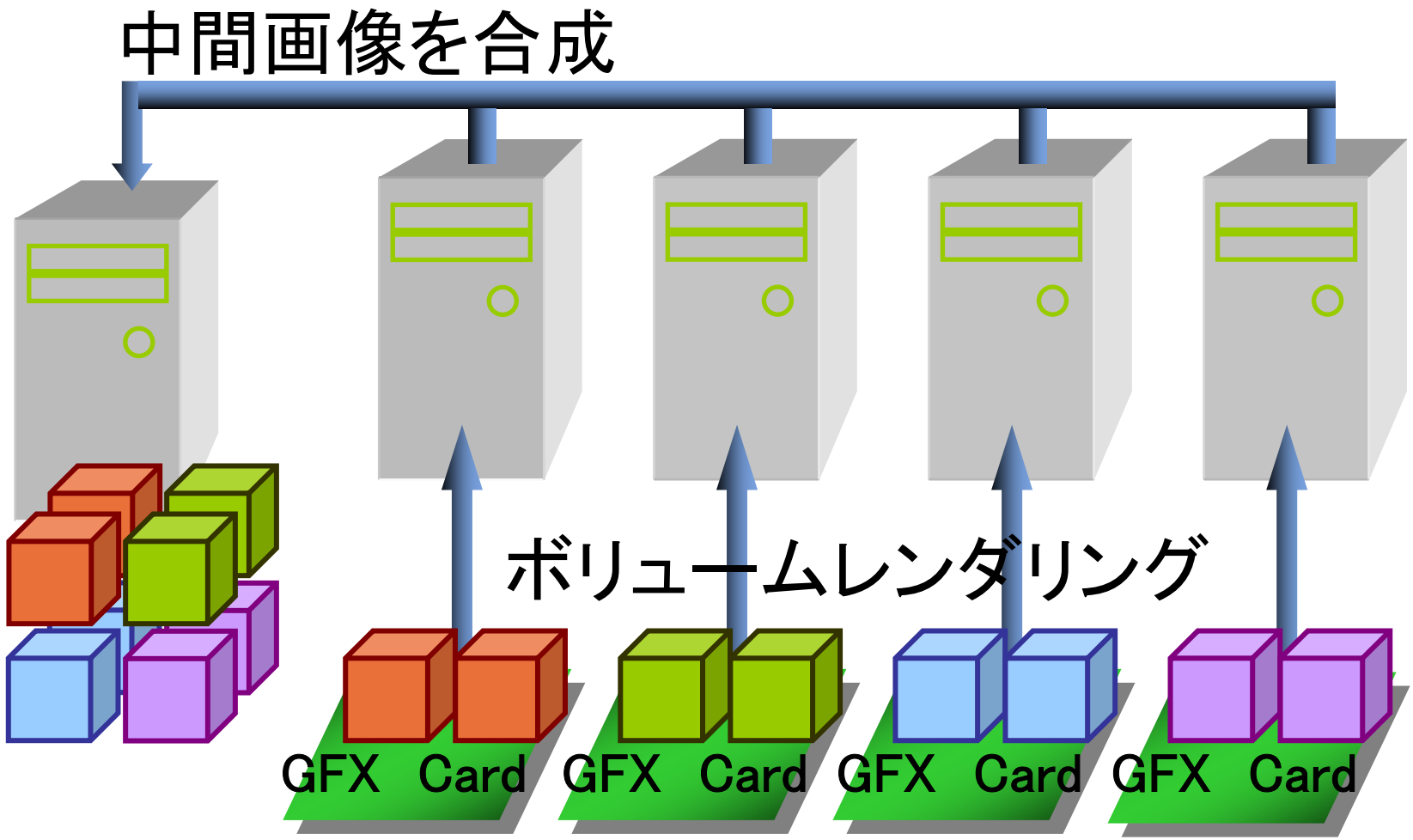




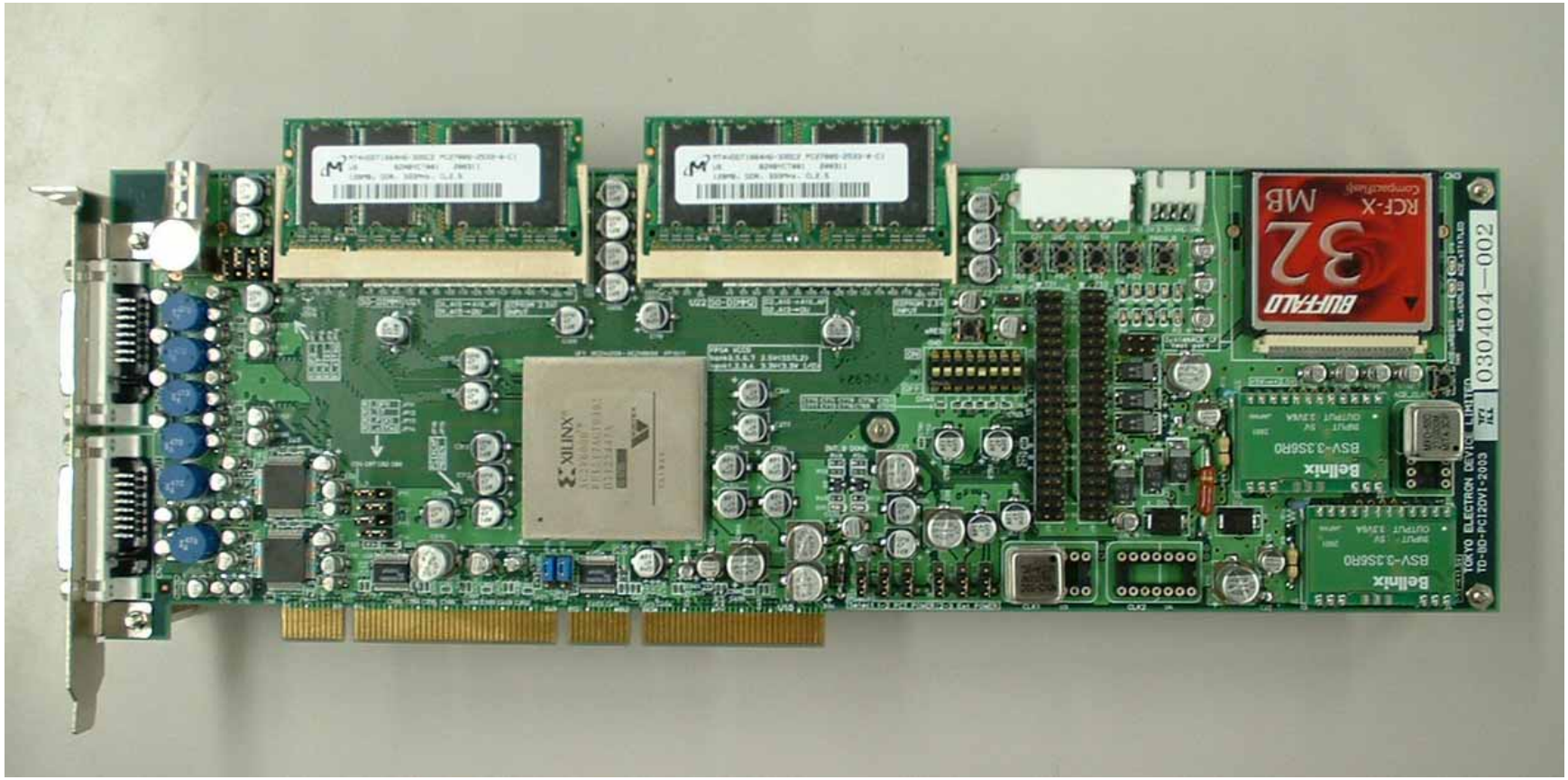
(a) Subdivide a volume data into subvolumes (B_n).



並列化手法



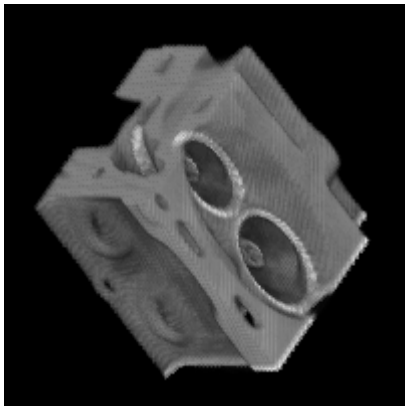
Visaカード


BUFFALO
32
MB
RCF-X
XILINX
Bellnix
BSV-3.3S6R0
TOKYO ELECTRON DEVICE LIMITED
TD-BD-PCI2DVI-2003
030404—002

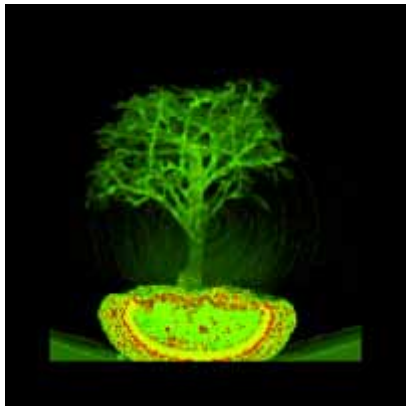
実装環境

Master
Pentium4 1.8GHz
512MB
GeForce4 Ti4600
128MB DDR
Slave(4台)
Pentium3 1.0GHz
512MB
RADEON9700PRO
325MHz Clock
Pipeline Unit 8
演算速度：2.6Gpixel/sec
128MB DDR
Red Hat Linux
100BASE Ethernet
PVM3.4

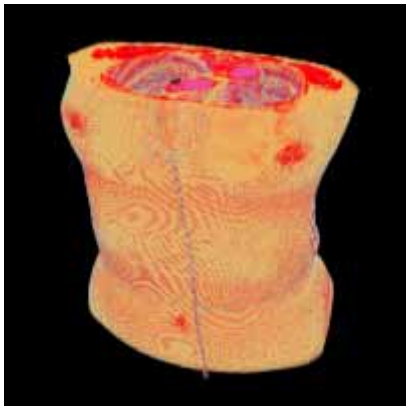
サンプルデータ



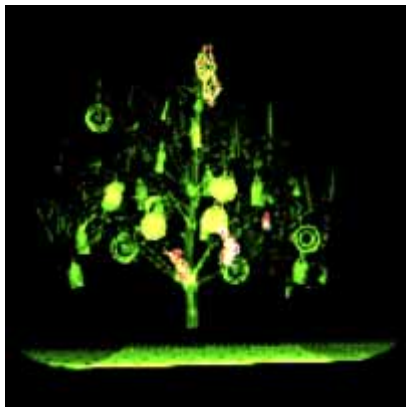
Engine : 256×256×128



Bonsai : 256×256×256



Chest : 512×512×512



Tree : 512×512×1024

並列化の効果

Data	描画速度[fps]			
	1台		4台	
	非分割	分割	非分割	分割
Engine (32MB)	102.4	39.9	17.7	18.0
Bonsai(64MB)	78.3	34.8	17.2	17.6
Chest (512MB)	-	0.066	1.30	13.2
Tree (1024MB)	-	0.007	-	0.77

分割：複数のテクスチャデータに分けて描画

サブブロックの数：4×4×4

６．４ＶＬＳＩＣＡＤ

ＣＡＤシステム

①VHDLなどの高水準論理記述言語

②論理シミュレータ

③自動配置配線システム

④回路シミュレーション

⑤故障シミュレーション

6.4.1論理シミュレーションの基本方式

①ゲートレベルシミュレータ

AND、ORなどのゲートを単位とした細かな

シミュレーション

②ブロックレベルシミュレータ

ゲートの集合体である機能レベルの高いブロックを対象

①全ゲート／ブロック方式

コンパイル方式:

前回と比較して入力信号に変化のないゲート／ブロックに対してもすべてシミュレーション

回路図はそれと等価な動作を行うプログラムにコンパイル

個々のゲート／ブロックに対応するプログラム部分は信号伝ぱん順に必ず実行

②イベント駆動方式

前回と比較して入力信号変化のあったゲート／ブロックのみシミュレーション

インタプリタ的な方式で処理

同期方式と非同期方式

・同期方式

時刻tでのシミュレーションが全回路について終了したのち、次の時刻t＋１のシミュレーションを開始

・非同期方式

時刻tでのシミュレーションが全回路について終了していないのに、部分的に時刻t+1のシミュレーションを開始

保守的方式：時刻t+1のシミュレーションを開始しても安全な（将来矛盾が起こらない）回路部分のみをシミュレーション。悲観的方式

タイムワープ方式：楽観的方式。　投機的な実行

安全でないような場合についてもシミュレーションを先に進める。

時刻tでの処理結果が、すでに開始された

時刻t+1のシミュレーションに影響を与える場合には、時刻t+1のシミュレーション結果は棄却され、時刻tからシミュレーションはやり直される。

6.4.2専用並列論理シミュレータ

IBMのYSE、
富士通のSP、
NECのHAL

HALの構成

29台の論理プロセッサ
２台のメモリプロセッサ
　メモリ装置をシミュレート
１台の制御プロセッサ
多段結合網

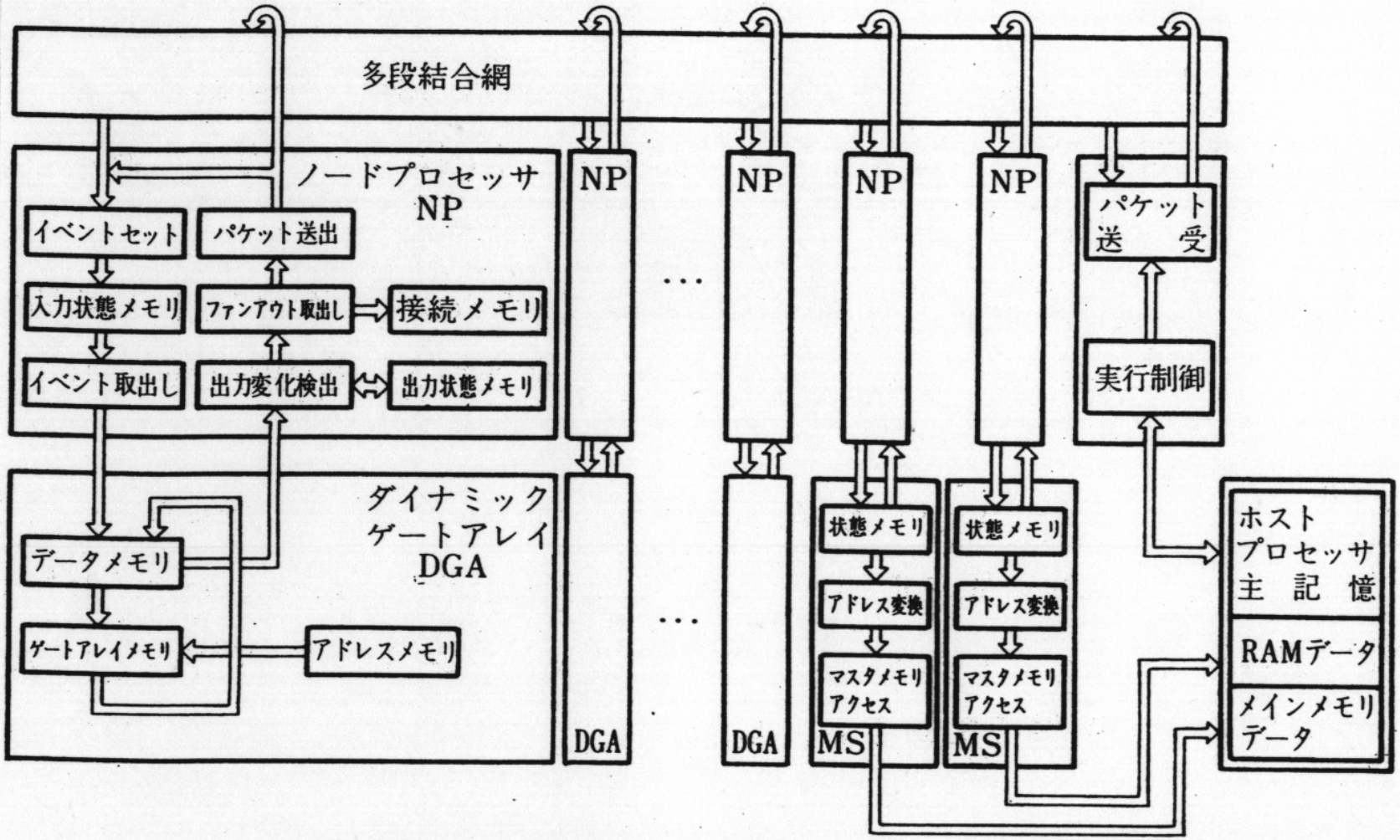
HAL：NECの並列コンピュータ
Cenjuの祖先

論理回路：

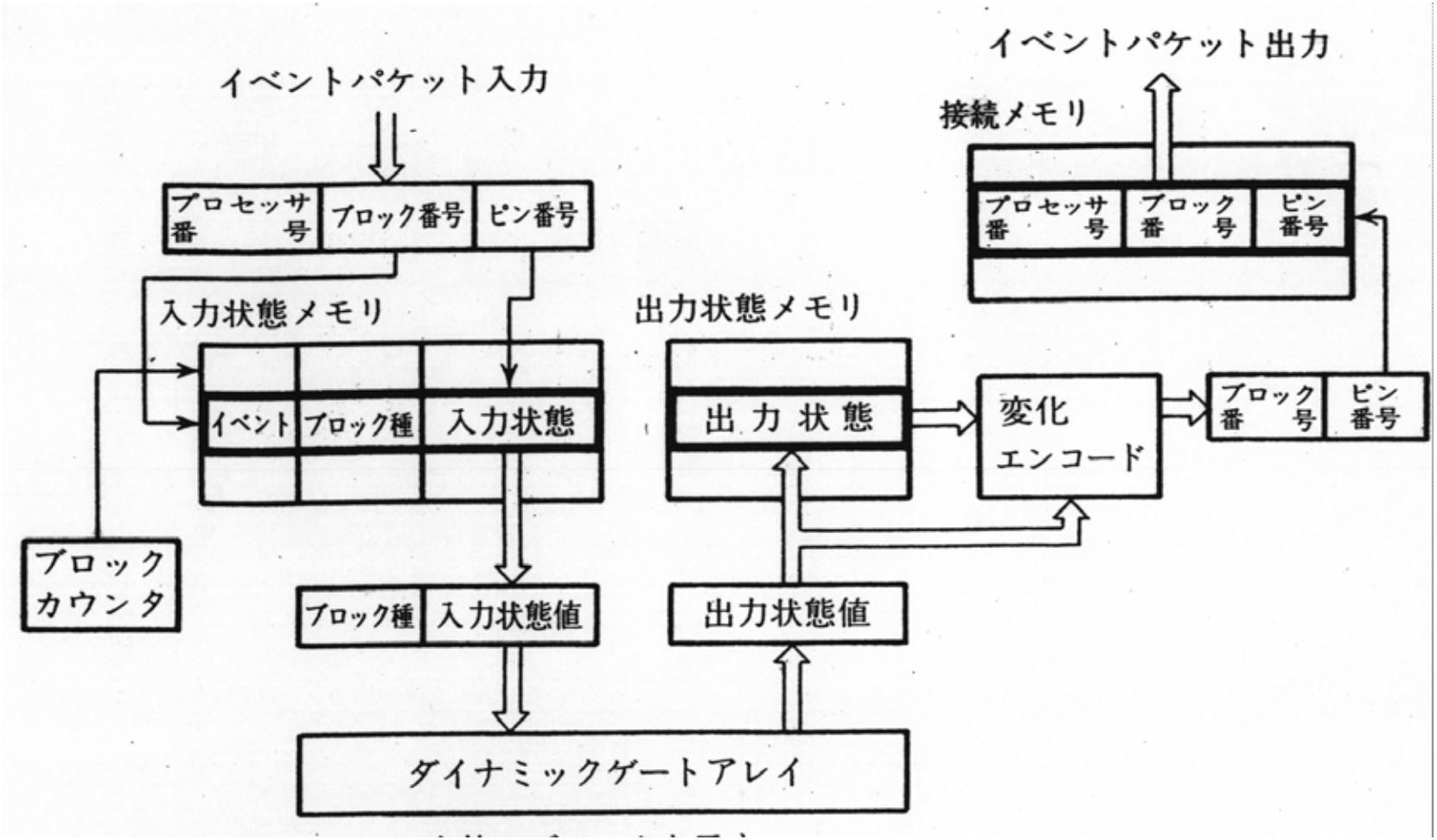
32入力32出力（数十～数百ゲートの論理回路に相当）を有する論理的にまとまりのある機能ブロックに分割

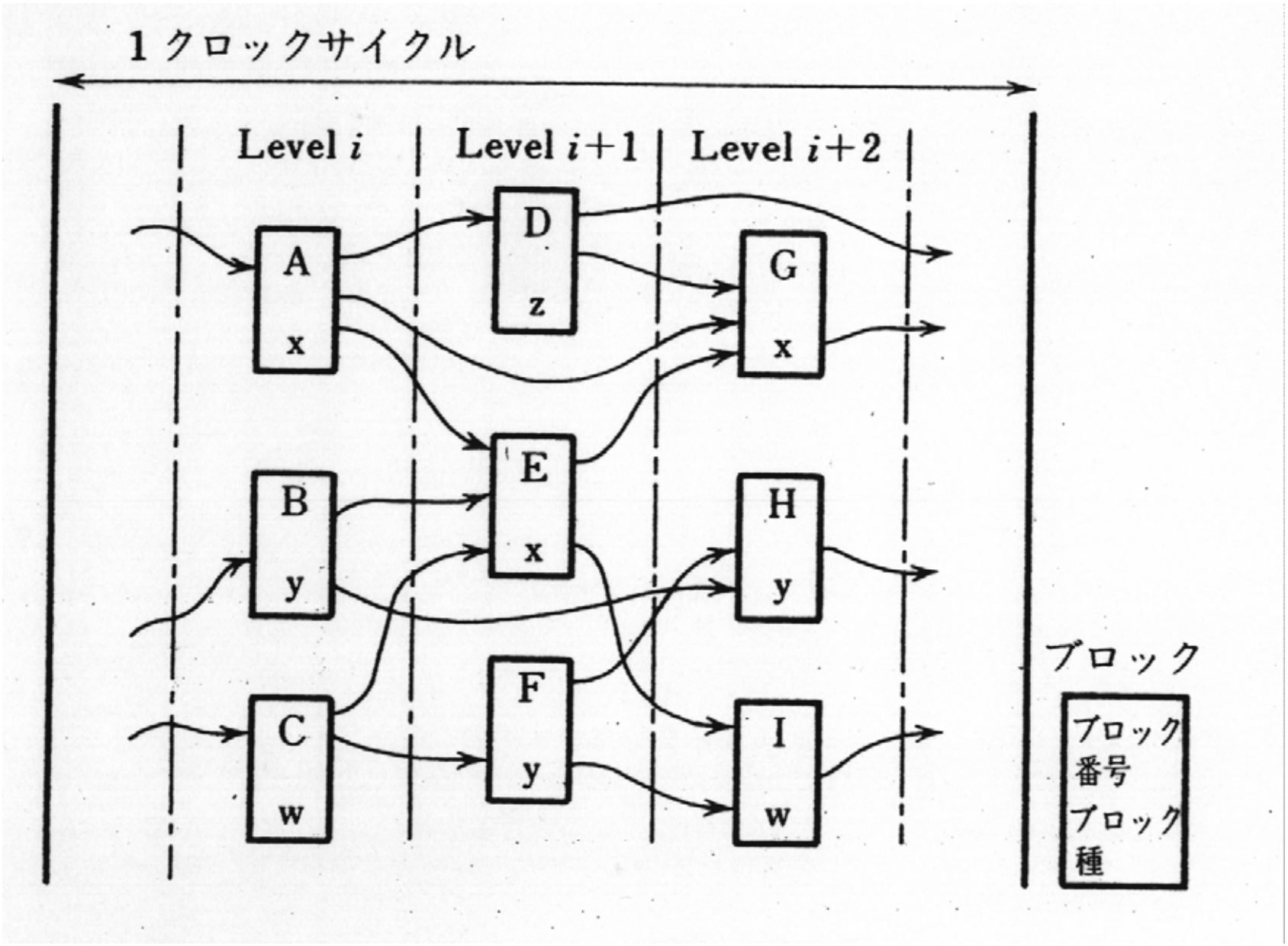
各ブロック：プロセッサの入出力状態メモリや接続メモリに格納

入力状態、出力状態、出力ピンの接続先（プロセッサ番号、ブロック番号、ピン番号）、および



NP：ノードプロセッサ　DGA：ダイナミックゲートアレイ


イベントパケット入力
プロセッサ番号
ブロック番号
ピン番号
入力状態メモリ
イベント
ブロック種
入力状態
ブロックカウンタ
ブロック種
入力状態値
ダイナミックゲートアレイ
出力状態メモリ
出力状態
出力状態値
変化エンコード
イベントパケット出力
接続メモリ
プロセッサ番号
ブロック番号
ピン番号
ブロック番号
ピン番号


1クロックサイクル
Level i
Level i+1
Level i+2
A
x
B
y
C
w
D
z
E
x
F
y
G
x
H
y
I
w
ブロック
ブロック番号
ブロック種

ブロック種（ブロックの論理機能を指示）

ブロックのシミュレーション：

ダイナミックゲートアレイ

真理値表：ブロック種を論理関数とし、入力状態を論理変数

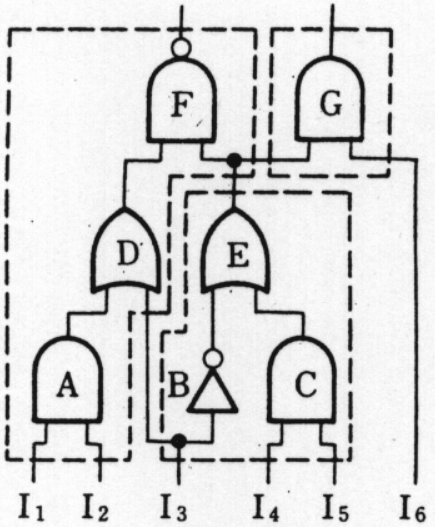
RAMによる真理値表を読出し

6.4.3スーパコンピュータの利用

日立のVELVET

信号線のベクトルデータの各々に対して

論理ゲートの操作を付随

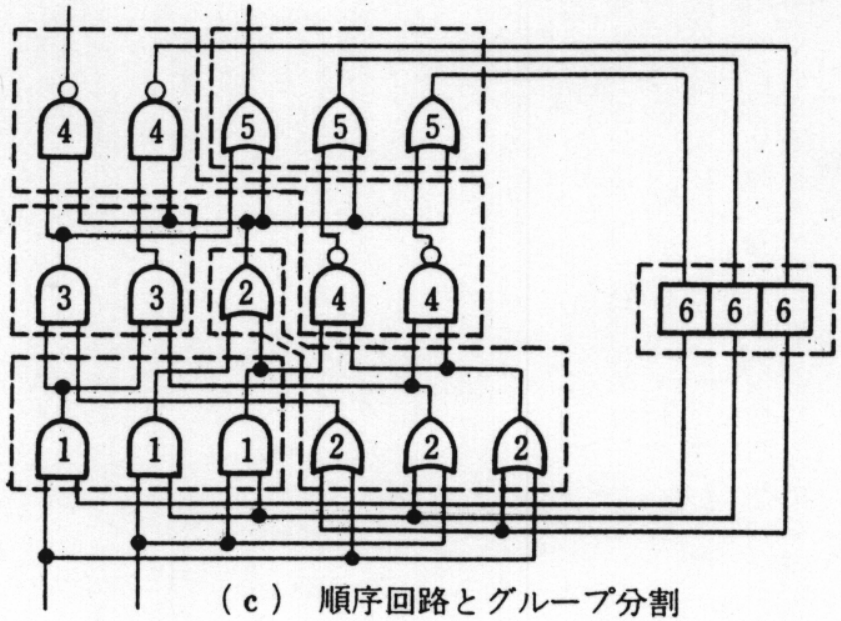


（a） 組合せ回路

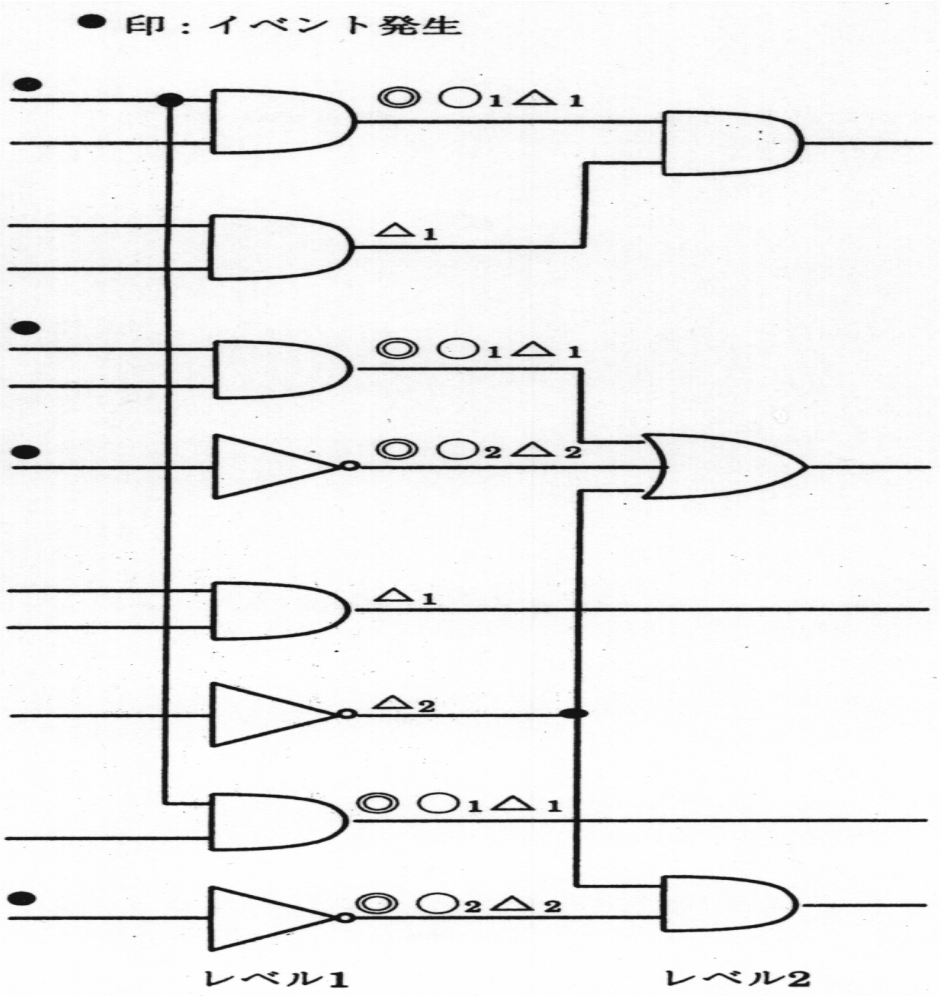
```
000001      READ(4,400) ((BUFF(J,I),J=1,LEN),I=1,6)
000002      DO  10 J=1,LEN
000003      BUFF(J,7)=IAND(BUFF(J,1),BUFF(J,2))
000004      BUFF(J,1)=IAND(BUFF(J,4),BUFF(J,5))
000005      BUFF(J,2)=IOR(BUFF(J,7),BUFF(J,3))
000006      BUFF(J,4)=NOT(BUFF(J,3))
000007      BUFF J,3)=IOR(BUFF(J,4),BUFF(J,1))
000008      BUFF(J,1)=NOT(IAND(BUFF(J,2),BUFF(J,3)))
000009   10 BUFF(J,2)=IAND(BUFF(J,3),BUFF(J,6))
```

1行目　外部入力I_1~I_6の読込み
3行目　ゲートAの評価
4行目　ゲートCの評価
5行目　ゲートDの評価
6行目　ゲートBの評価
7行目　ゲートEの評価
8行目　ゲートFの評価
9行目　ゲートGの評価

（b）（a）に対するFORTRANプログラム



（c） 順序回路とグループ分割



◎：VELVETで一つのベクトル命令での処理.
○1，○2　：イベント駆動方式で二つのベクトル命令で処理
△1，△2　：コンパイラ方式で二つのベクトル命令で処理.

図 6.27　スーパコンピュータによる論理シミュレーション

演算パイプラインの考え方を止揚

スーパコンピュータの機能の利用

①リストベクトル

・各レベル間での結線情報の処理

②マスクレジスタ

・イベントが生じたゲートのマーク

③ベクトルの収集・拡散

・イベントが生じたゲートのみへの処理

6.4.4マルチプロセッサ方式

並列論理シミュレーション：

一般的には並列離散的イベントシミュレーションの一問題

コンピュータシステムのシミュレーション例

①コンピュータシステム：２台装備され、

各々時分割（TSS）で動作

②ジョブ発生源：３つ

イベント：

ジョブ発生源からのジョブの発生

タイムクアンタムごとの処理の中断や処理の終了

シミュレーション時刻t

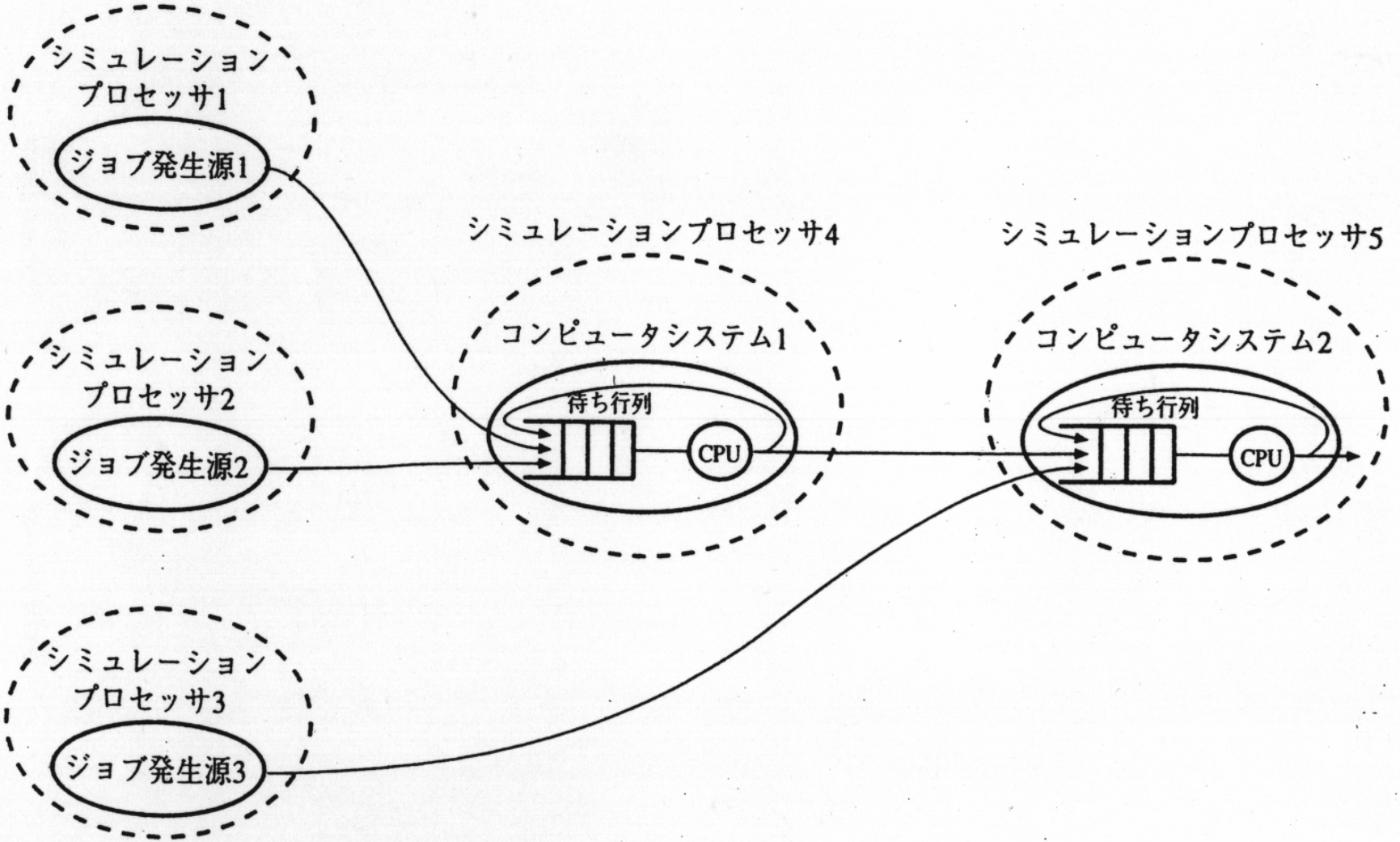


図 6.28　コンピュータシステムのシミュレーション

6.4.5逐次シミュレーション

タイムホイール表で管理

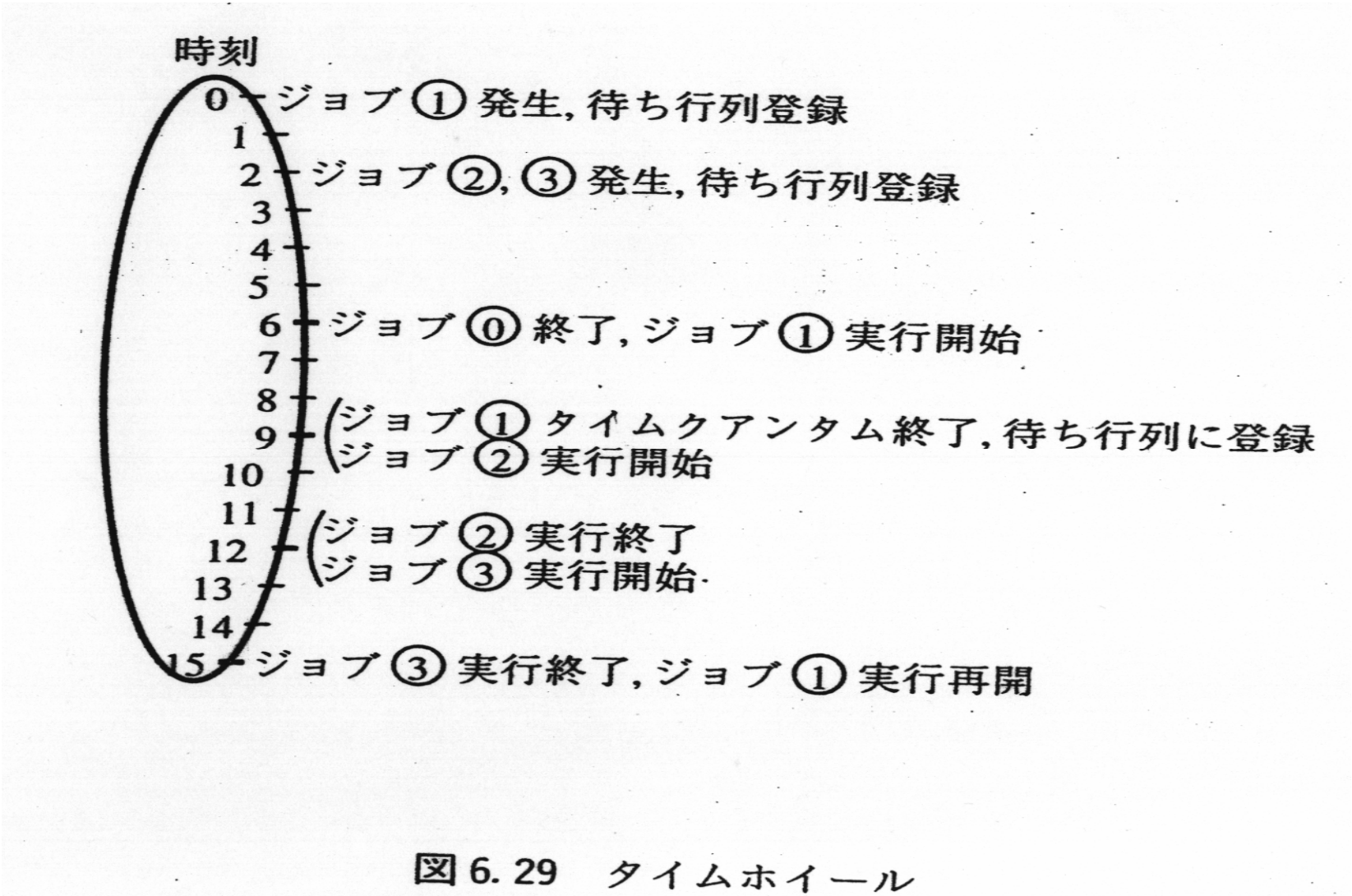


図 6. 29　タイムホイール

6.4.6並列シミュレーション

（１）同期方式

（２）保守的方式

- **タイムスタンプ：イベントの発生する**
  シミュレーション時刻
- **イベントの種類：どのイベントの種類とその処理内容**

・イベントの転送先

矛盾の生じない範囲でシミュレーション時刻を各プロセッサで独自に進める

無矛盾性：シミュレーションノード（プロセッサ）への制約で実現

①各ノードの入力リンク：タイムスタンプの小さなものから順に到着

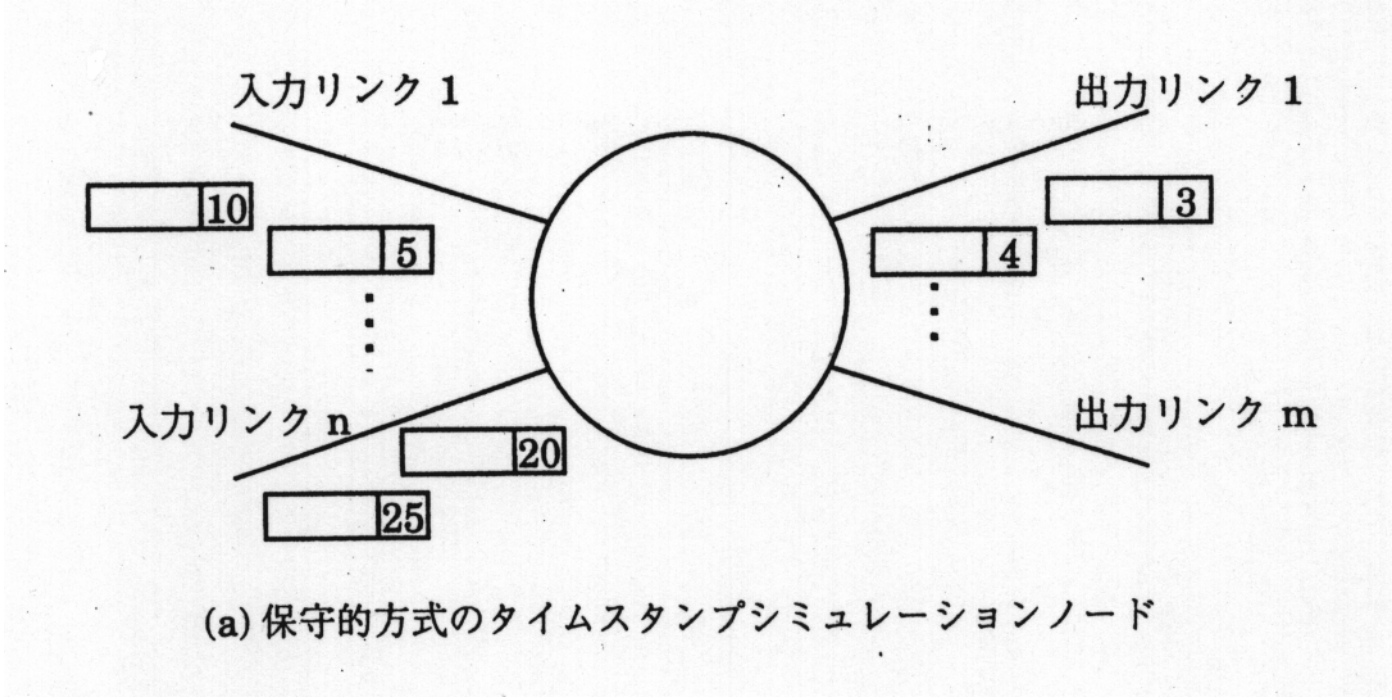
②各出力リンク：タイムスタンプの小さな順でイベントを生成

③すべての入力リンク上に少なくとも１つの
イベントが存在するとき
すべての入力リンク上の入力イベントのうち最も小さなタイムスタンプを有する入力イベントを取り出し、
そのノードでの出力時刻（入力イベントのタイムスタンプ値にノードでのイベント処理時間を加算した時間）をタイムスタンプとして、
出力リンクにイベントとして出力

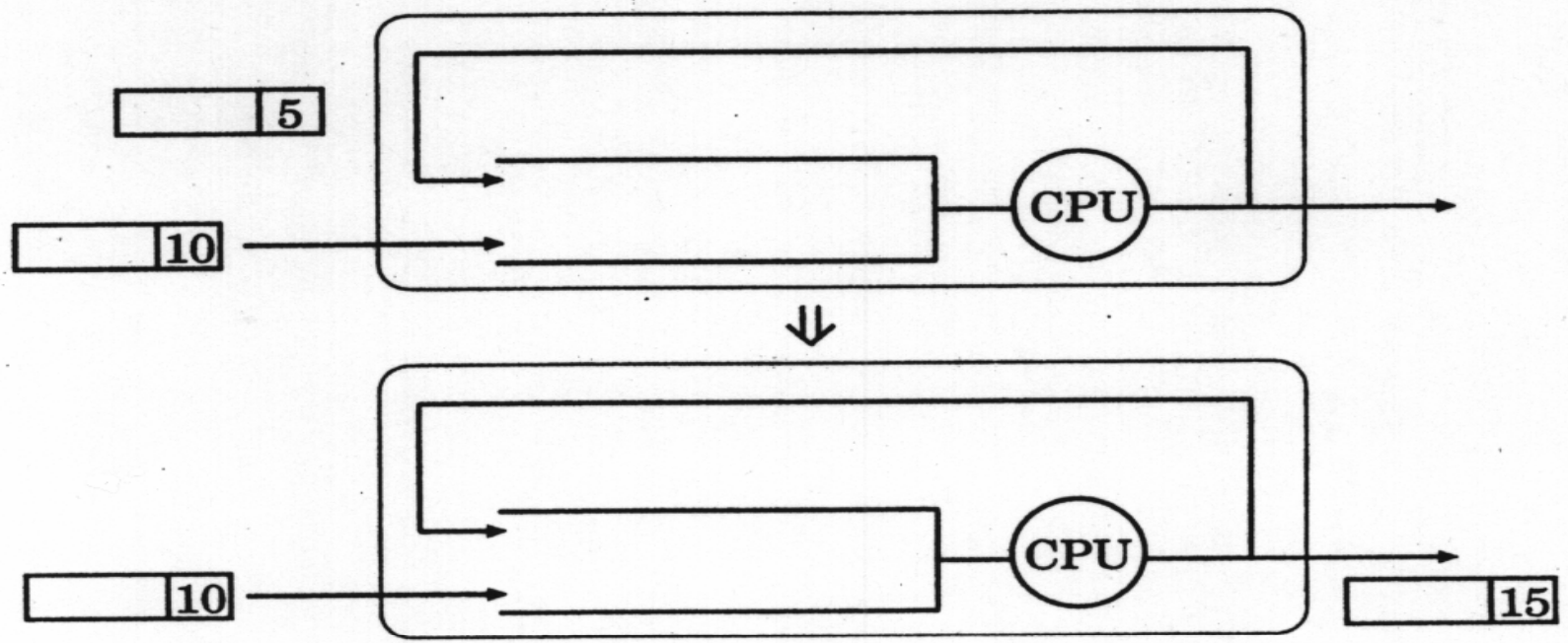
デッドロックの生起

ヌルメッセージの使用

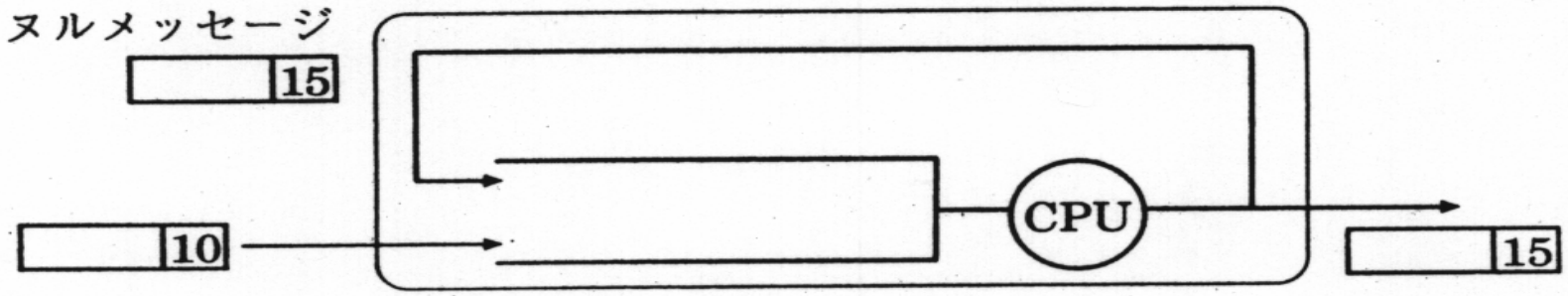
保守的な方式をマルチプロセッサで
　実現する際の問題点
①通信量の増大
　ヌルメッセージなどで通信量が増加
②低並列度
　イベントＡ、Ｂ
　　ＢはＡが生じなかった場合に起動
　　Ａが生ずるのはきわめて希
　　Ａが生じなかったことが判明するまで、Ｂを待た
　　すのは無駄



(a) 保守的方式のタイムスタンプシミュレーションノード



(b) デッドロック



(c) ヌルメッセージ

図 6. 30　保守的方式

シミュレーションにおける並列度を稼げない

並列度の向上

Ａ、Ｂのイベント処理を並列に実行

Ａが生じればＢのシミュレーションを無効化

（３）タイムワープ方式

先行したシミュレーションの無効化

アンチメッセージを送出

最も番号の小さなタイムスタンプ

以前の状態はすべて確定

履歴として記憶する必要はない

広域仮想時刻（global virtual time）

６．５　データベース処理

トランザクション処理

問い合わせ処理

関係データベース

基本演算

選択，射影，結合

6.5.1ソーティング

バブルソート（O(N²)）、

クイックソート（O(NlogN)）、

ヒープソート（O(NlogN)）

①一般にレコード数Ｎは極めて多いこと

②レコード数は大幅に変動すること

③レコードは通常ディスクに格納。

アクセスはヘッドやチャネルを通して逐次

表 6.1　並列ハードウェアソータ

演算器数	処理時間	例
$\log N$	N	パイプラインマージソート
$\log N$	N	パイプラインヒープソート
N	N	並列奇偶置換ソート
N	N	並列トリーソート
N	N	並列計数ソート
N	N	リバウンドソート
N	N	パイプラインバブルソート
N	$\sqrt{N}$	格子上パイトニックソート
$N(\log N)^2$	$(\log N)^2$	シャッフル結合バイトニックソート

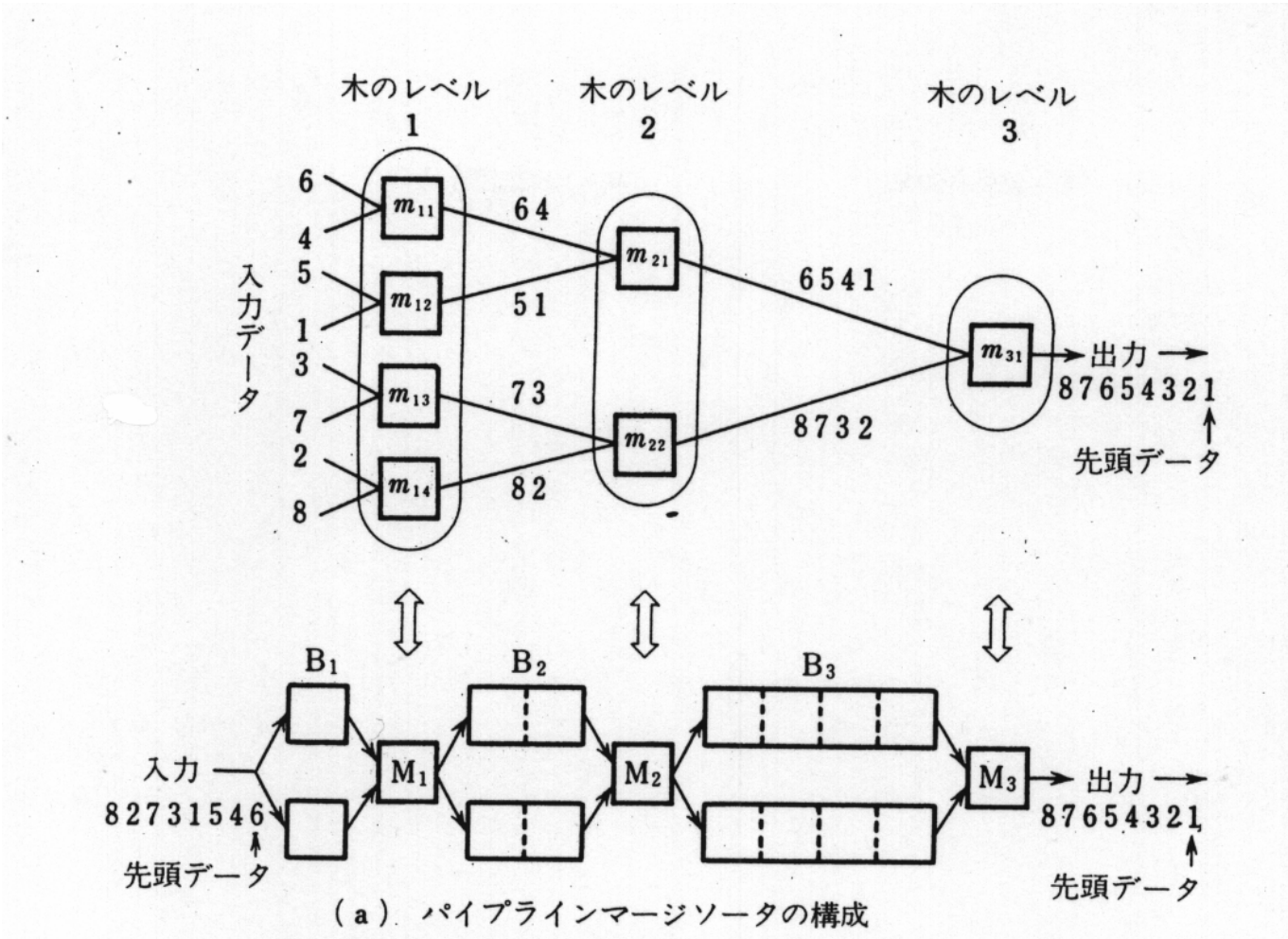
N：レコード数

（１）パイプラインマージソータ

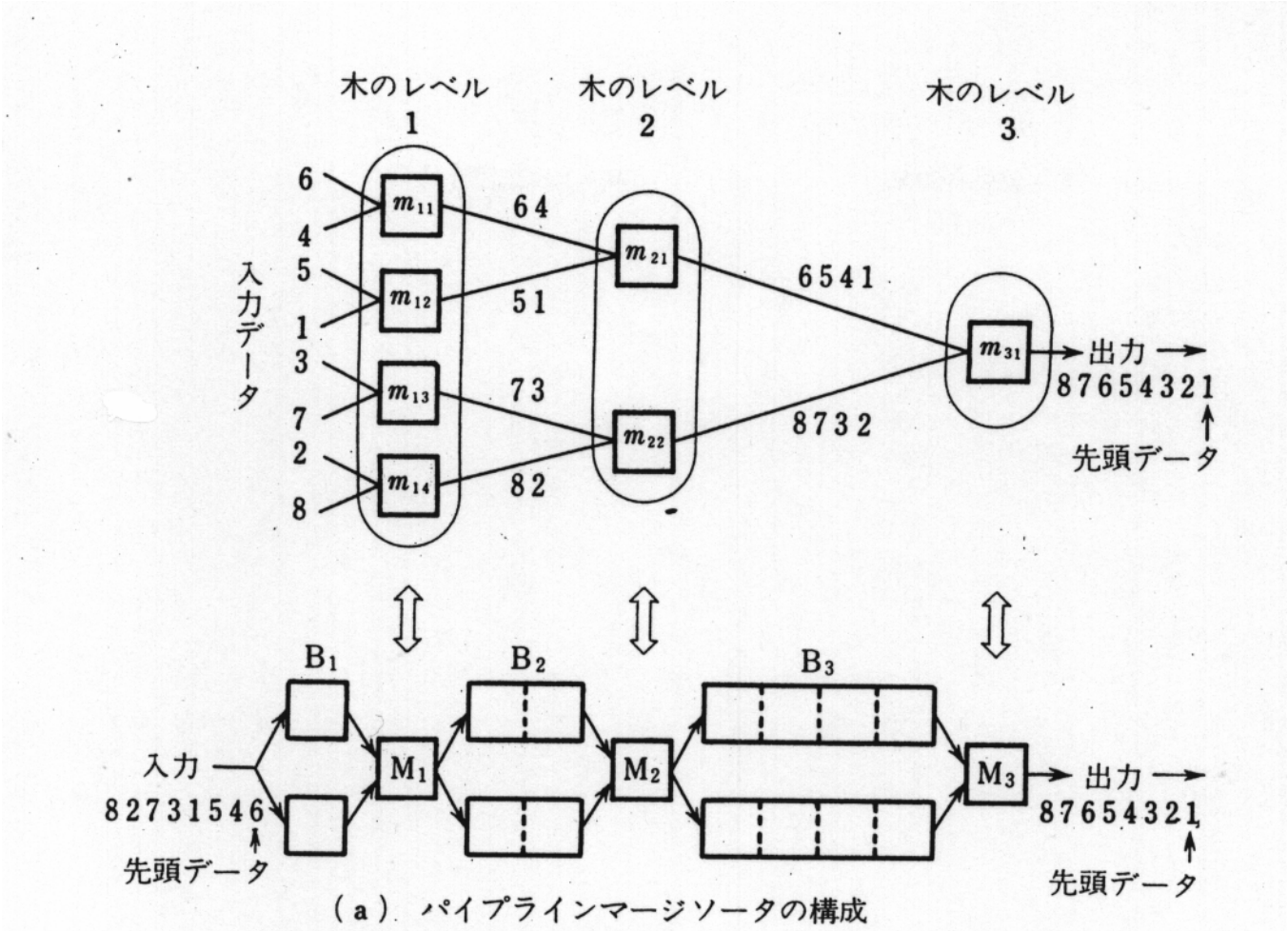
IBM**の**S.Todd**により提案された方式**

新世代コンピュータ開発機構（ICOT**）の**

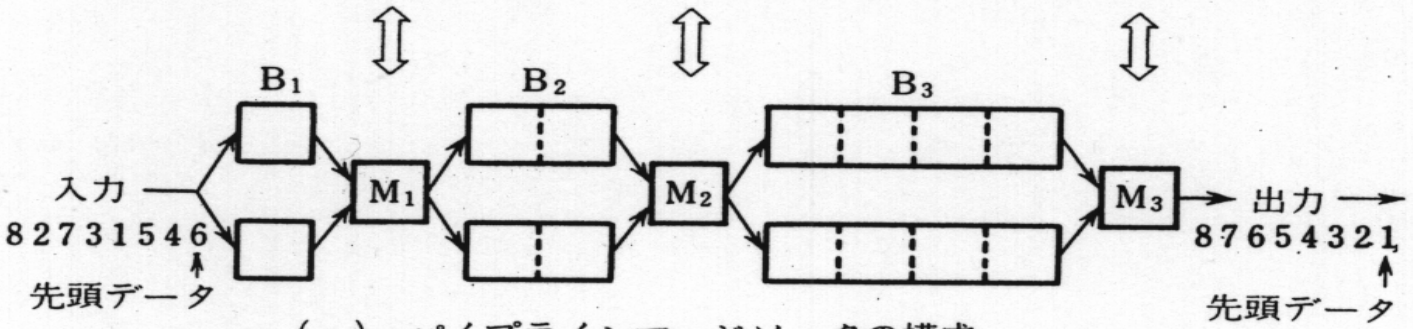
データベースマシンDelta**など**



（a） パイプラインマージソータの構成



(a) パイプラインマージソータの構成



（a） パイプラインマージソータの構成

時刻	B_1 上	B_1 下	M_1 起動	B_2 上	B_2 下	M_2 起動	B_3 上	B_3 下	M_3 起動	出力
1	6	○								
2		4	←○							
3	5			4						
4		1		6	○					
5	3				1	←○				
6		7			5		1			
7	2			3			4			
8		8		7			5	○		
9					2		6			
10					8			2	←○	
11								3		1
12								7		2
13								8		3
14										4
15										5
16										6
17										7
18										8

出力開始までの時間は
○印の項と←○印の項をすべて
加算すると得られる

出力開始時点

（b） 処理の流れ

（２）パイプラインヒープソータ

　北海道大学で考案されたソータ

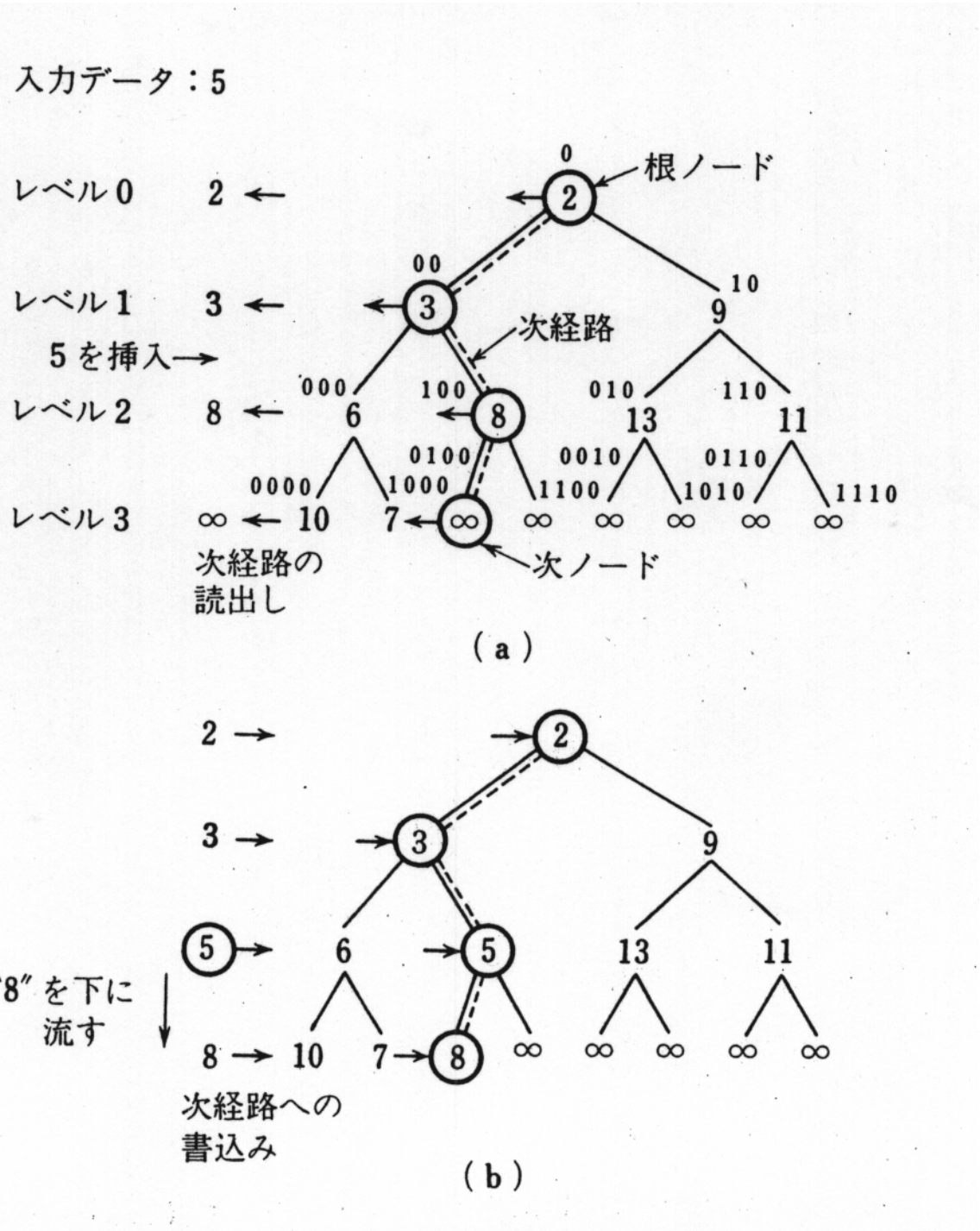
ヒープ：

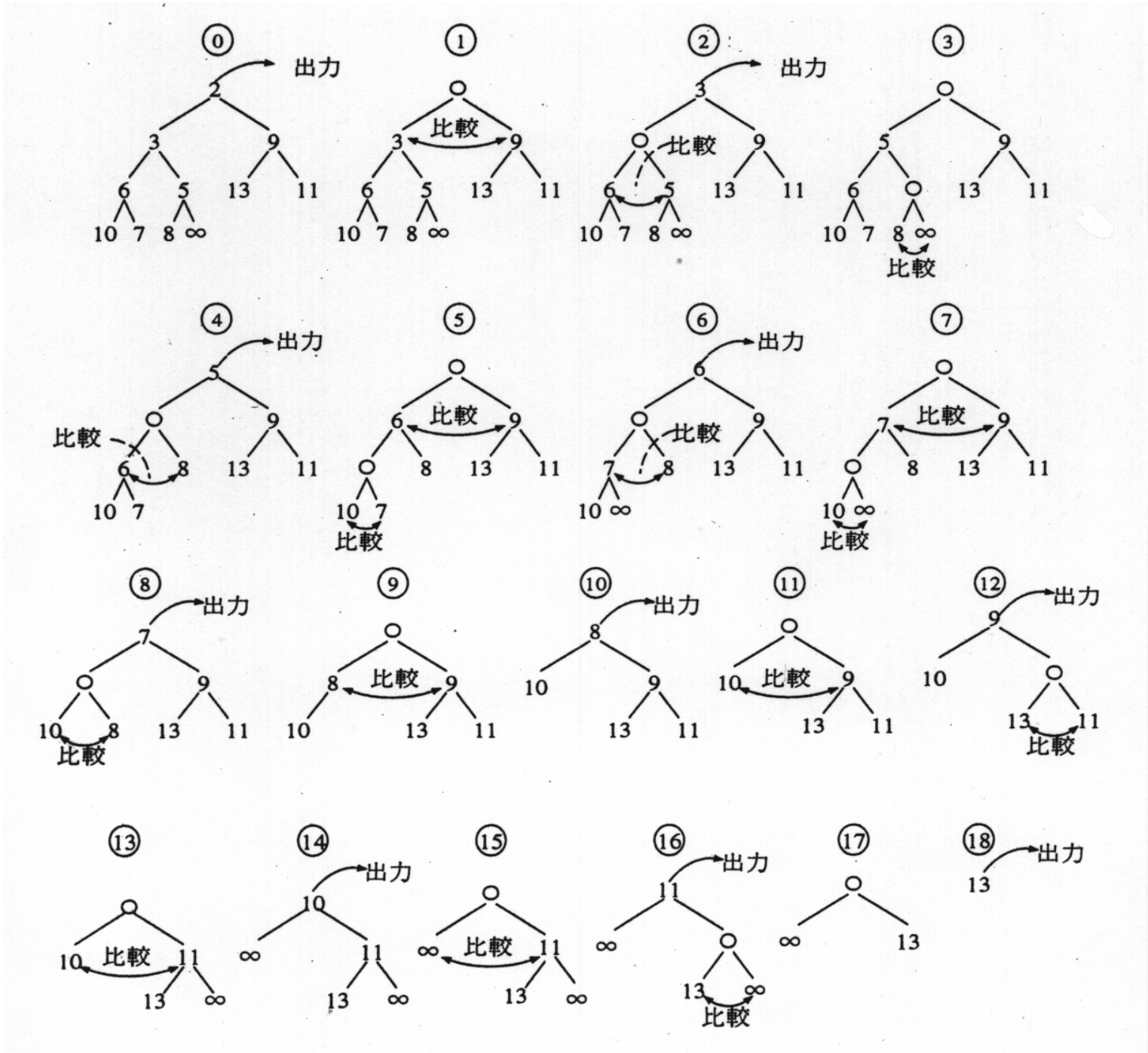
・最下位のノード（葉ノード）を除去すると完全２進木

・各ノードのデータはその子ノードのいずれよりも小さいか等しい。

①ヒープ生成過程

②ヒープ出力過程


入力データ：5
レベル0
2
レベル1
3
5を挿入
レベル2
8
レベル3
次経路の
読出し
根ノード
次経路
次ノード
(a)
"8"を下に
流す
次経路への
書込み
(b)


0
出力
2
3
9
6
5
13
11
10 7 8 ∞
1
比較
2
3
出力
比較
3
5
9
6
8
13
11
10 7
4
5
6
7
∞

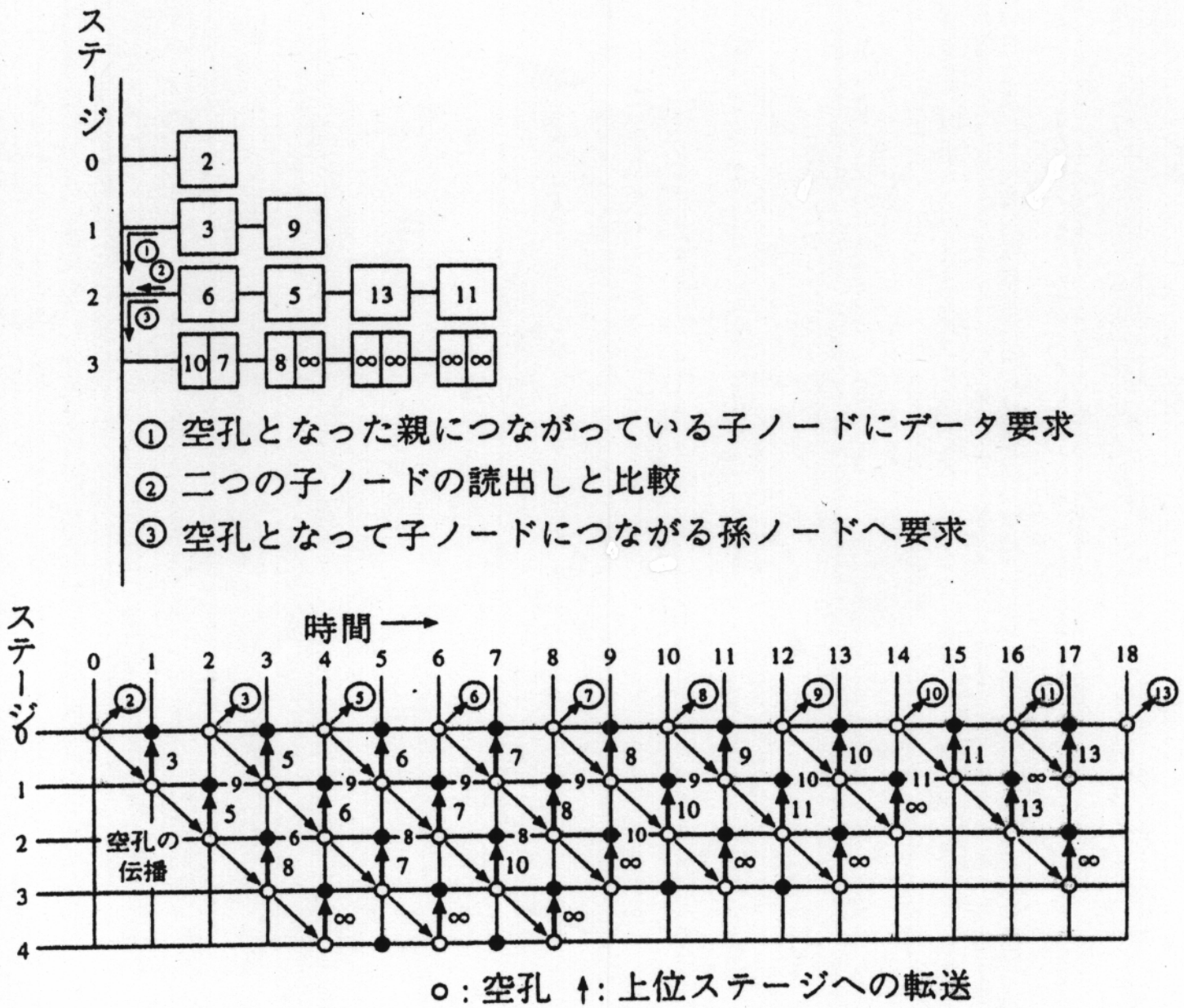


図 6.34　パイプライン方式によるヒープ出力

（３）バイトニックソータ

①双調列

順序系列$(a_1, a_2, a_3, \ldots, a_{2m})$が

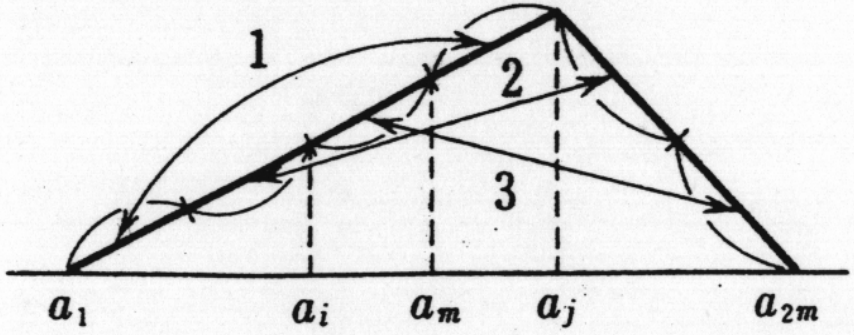
$$a_1 \leqq a_2 \leqq, \ldots, \leqq a_j \geqq, \ldots, \geqq a_{2m}$$

（３７）

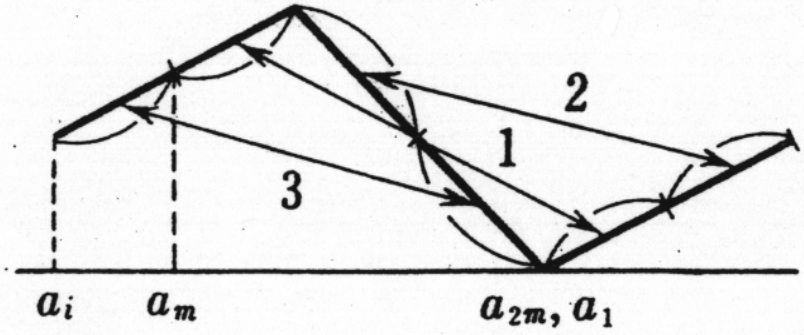
を満足するとき、または、この順序系列を巡回シフトして得られる系列、

$(a_i, a_{i+1}, \ldots, a_{2m}, a_1, a_2, \ldots, a_{i-1})$　（３８）

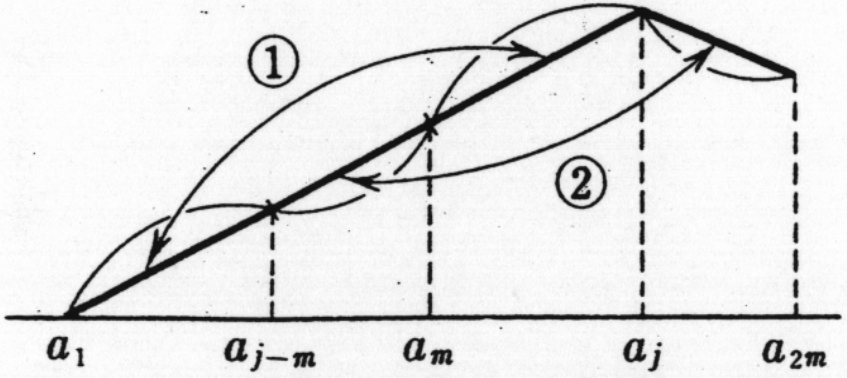
K.E.Batcher博士考案



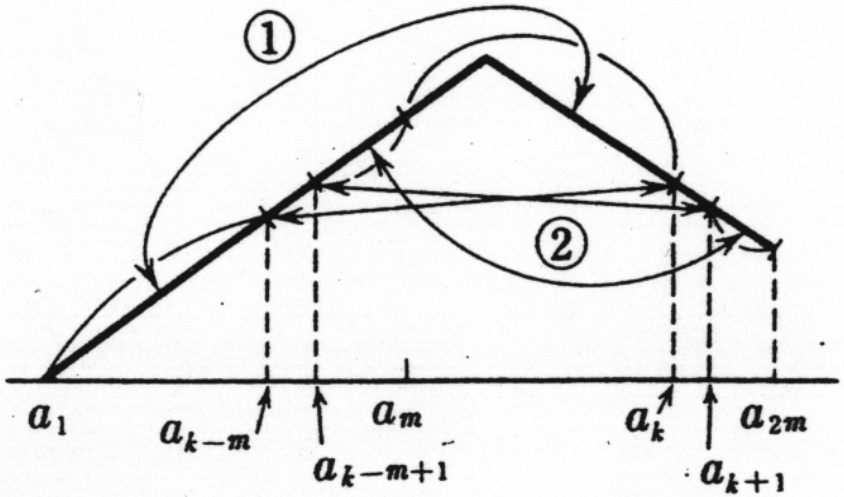
（a） $a_1 \leq a_2 \leq \cdots \leq a_j \geq \cdots \geq a_{2m}$



（b）（a）を巡回シフトしたもの



（c） $a_m \leq a_{2m}$の場合



（d） $a_m > a_{2m}$の場合

②「定理」

（$a_1, a_2, a_3, \ldots, a_{2m}$**）を双調列とし、**

$$d_i = \min(a_i, a_{i+m})$$

$$e_i = \max(a_i, a_{i+m}) \qquad \text{（３９）}$$

とするとき、

$$\{d_i\} = (d_1, d_2, \ldots, d_m)$$

$$\{e_i\} = (e_1, e_2, \ldots, e_m) \qquad (40)$$

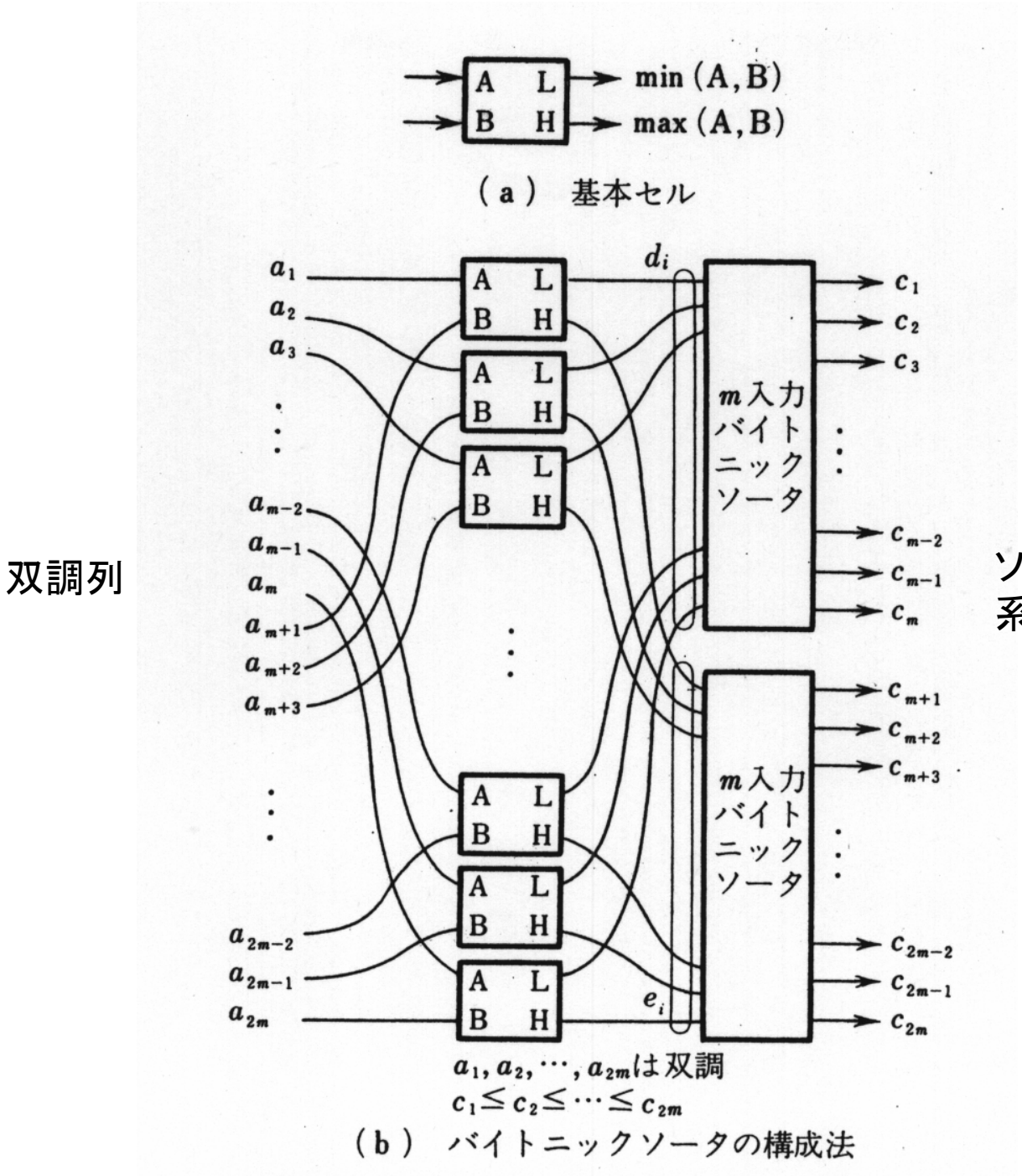
は双調列となり、

$$\max\{d_i\} \leqq \min\{e_i\} \qquad \text{（４１）}$$

③バイトニックソータの構成

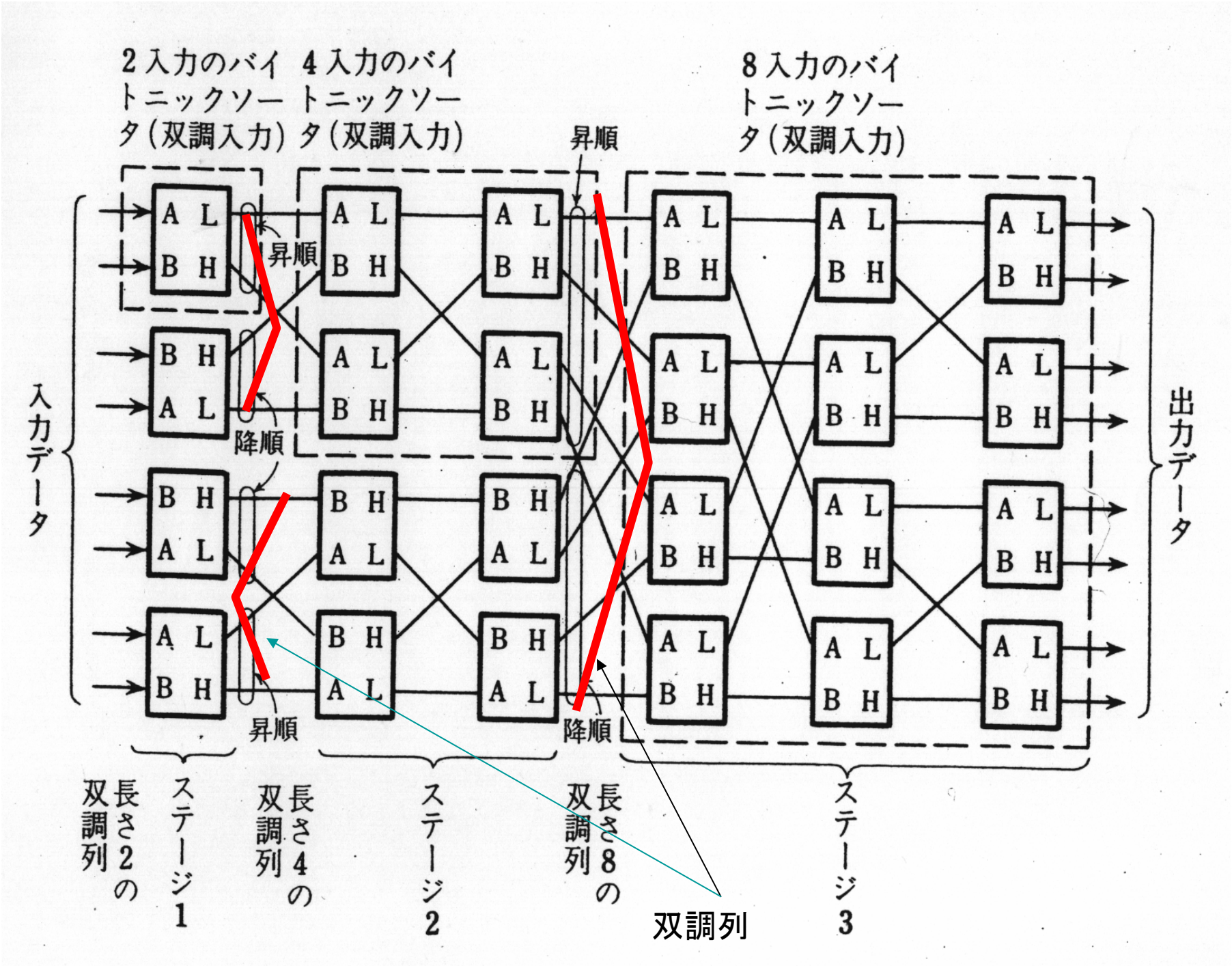
ステージ数

$$1+2+3+\cdots+\log N = \log N(\log N+1)/2 \qquad \text{（４９）}$$



双調列

ソートされた系列


2入力のバイトニックソータ(双調入力)
4入力のバイトニックソータ(双調入力)
8入力のバイトニックソータ(双調入力)
昇順
降順
入力データ
出力データ
A L
B H
ステージ1
長さ2の双調列
長さ4の双調列
ステージ2
長さ8の双調列
ステージ3
双調列

6.5.2データベースマシン

データベースの基本モデル

階層モデル、網モデル、

関係モデル、オブジェト指向モデル

（１）関係データベース

表形式で表現：「関係」

表の各項目：タプル（tuple）

氏名	成績	出身地	卒業年
A	80	大　阪	1990
B	90	京　都	1992
C	65	北海道	1993
D	100	島　根	1995
E	不	東　京	1994

(a) 学生データベース

企業名	所在地	業　　務	年商 (億)
イ	東　京	コンピュータ	100
ロ	北海道	海　産　物	20
ハ	長　野	精密機器	50
ニ	京　都	和　菓　子	10
ホ	島　根	瓦	5

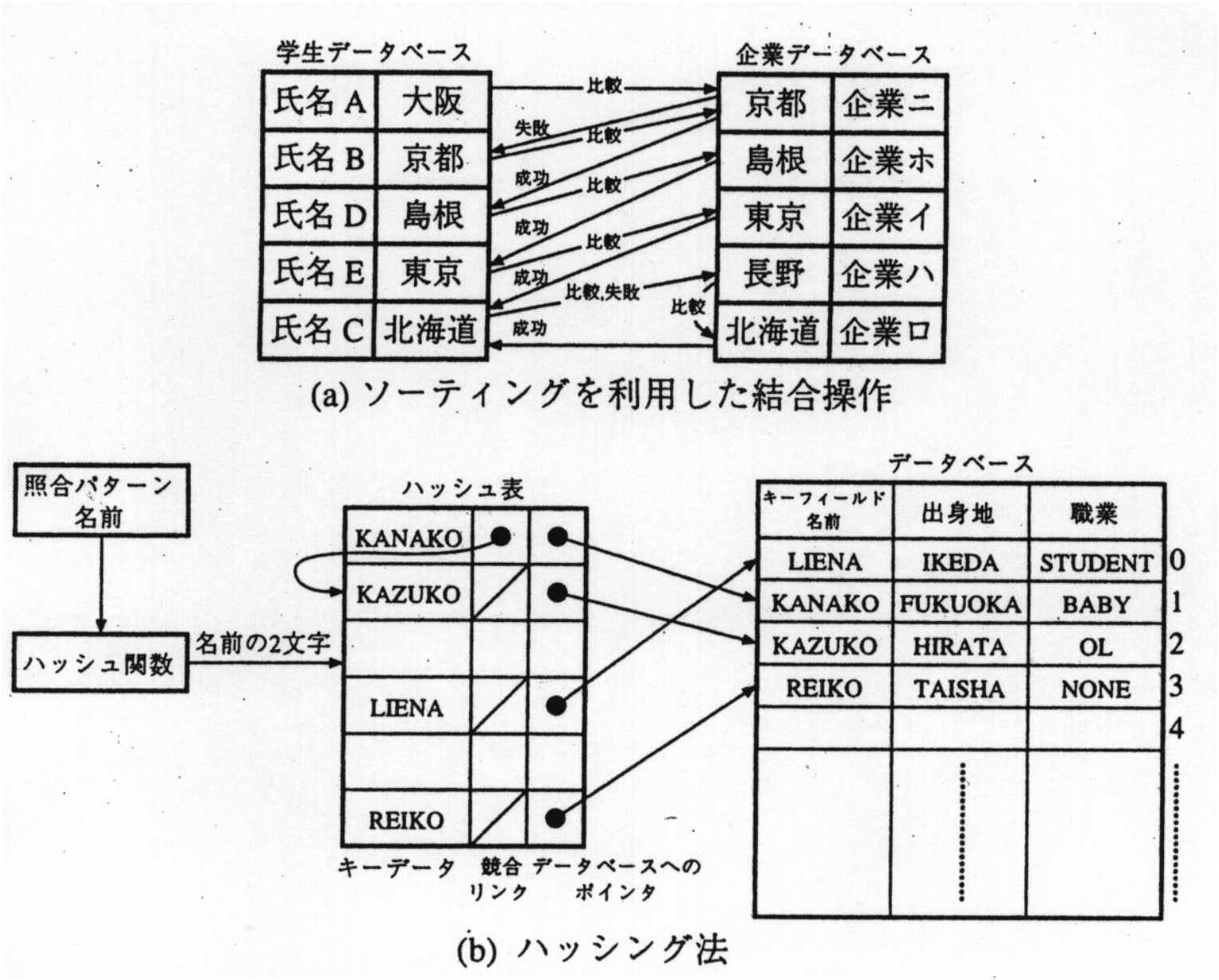
(b) 企業データベース

都道府県名	氏名	成績	卒業年	企業名	業　　務	年商 (億)
島　根	D	100	1995	ホ	瓦	5
東　京	E	不	1994	イ	コンピュータ	100
北海道	C	65	1994	ロ	海　産　物	20
京都	B	90	1992	ニ	和菓子	10

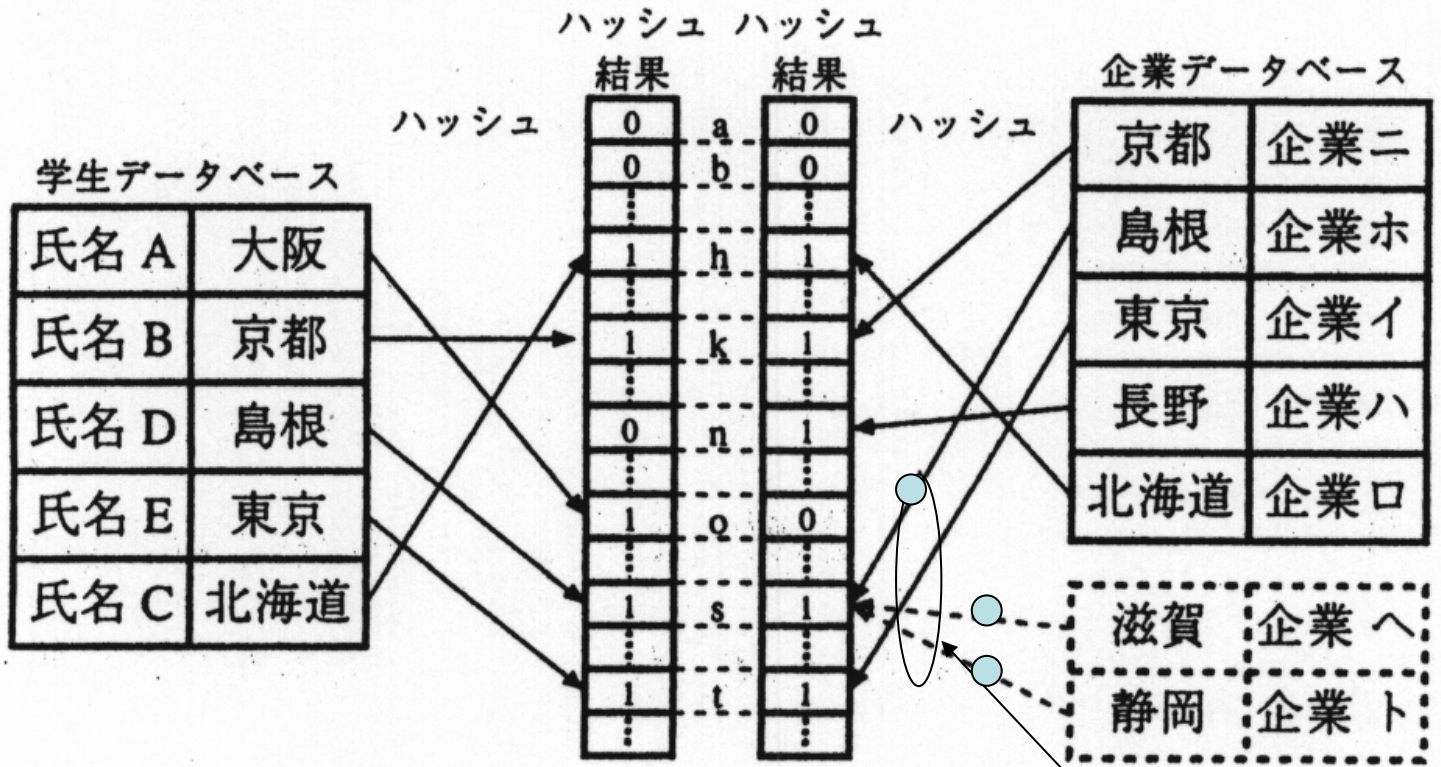
(c) U-ターン就職データベース

教科書訂正

図 6.38　関係データベースの例


学生データベース
氏名 A
大阪
氏名 B
京都
氏名 D
島根
氏名 E
東京
氏名 C
北海道
企業データベース
京都
企業ニ
島根
企業ホ
東京
企業イ
長野
企業ハ
北海道
企業ロ
比較
失敗
比較
成功
比較
成功
比較
成功
比較,失敗
比較
成功
(a) ソーティングを利用した結合操作
照合パターン
名前
ハッシュ関数
名前の2文字
ハッシュ表
KANAKO
KAZUKO
LIENA
REIKO
キーデータ
競合リンク
データベースへのポインタ
データベース
キーフィールド
名前
出身地
職業
LIENA
IKEDA
STUDENT
0
KANAKO
FUKUOKA
BABY
1
KAZUKO
HIRATA
OL
2
REIKO
TAISHA
NONE
3
4
(b) ハッシング法

地名の最初のアルファベットでハッシュ



(c) ハッシュを用いた結合操作

図 6.39　結合操作の高速化

基本操作

①選択（selection）

②射影（projection）

③結合（join）

学生データベースと企業データベースの２つのデータベース

Ｕ－ターン就職データベースの作成

結合操作：時間のかかる処理

データベースのタプル数：各々N_1、N_2

処理時間：$O(N_1 x N_2)$

・ソータの利用

処理時間：$O(N_1+N_2)$

・ハッシュの利用

（２）単一プロセッサによるデータベースマシン

単一プロセッサに多数のディスクが

結合されたシステム

データベースマシン：

当初、選択操作を高速化する手法

D.Slotnick教授の提唱したLogic/track

トロント大学のＲＡＰ、

テキサス大学のＣＡＳＳＭ

情報フィルタの機構を実現

結合操作の高速化は実現できない

商用の関係データベースマシン

情報フィルタ機構とソータ機構を組み合わせた方式

ソータは多くの場合、マージソータ

NTTのRINDA、日立RDSP、三菱GREO、東芝DBE、富士通DBA

（３）マルチプロセッサによるデータベースマシン

データベースマシンにおける並列処理

①検索・問合わせごとでの並列処理

②関係データベース演算間での並列・パイプライン処理

③関係データベース演算内での並列処理

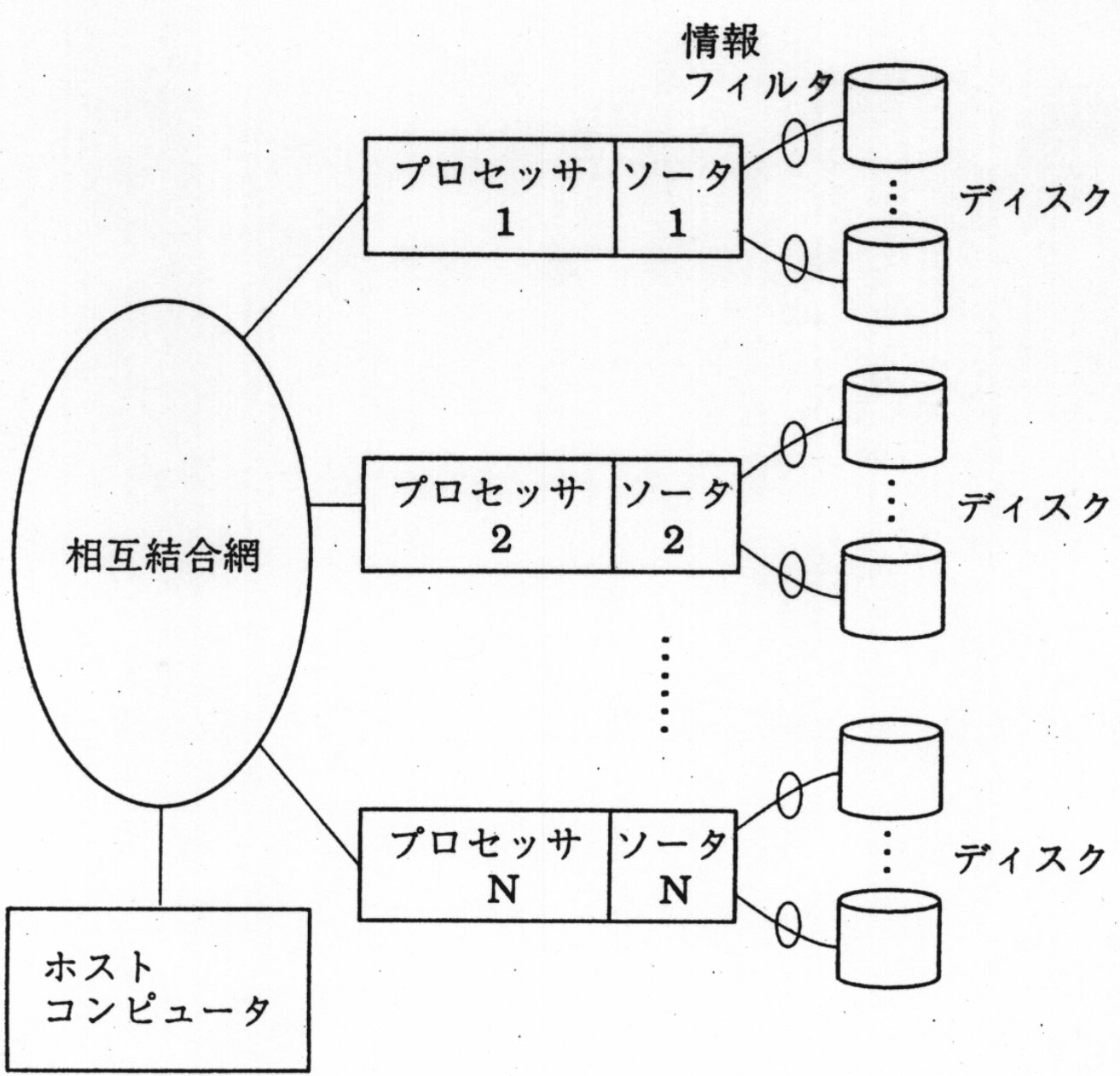


図 6.40　データベースマシンの構成

関係データベース演算内での並列処理

①結合網内でのソーティング処理

大規模な結合操作ではクラスタ内でのソートされたデータがクラスタ間でさらにソートされる必要

相互結合網内でのソート機能が必要

NCR社のNCR3600の相互結合網Y-net

各ノードにソータが装備

NCR3600：

最大256台のマイクロプロセッサi486で構成

20台のディスクアレイからなるディスクサブシステム

（容量21.7GB）が48台接続できる。

NCR5380（2004年現在）

最大512台のIntel Xeon 3.06GHz HyperThreading

BYNET：Banyan Network 120MB/s（node-node）

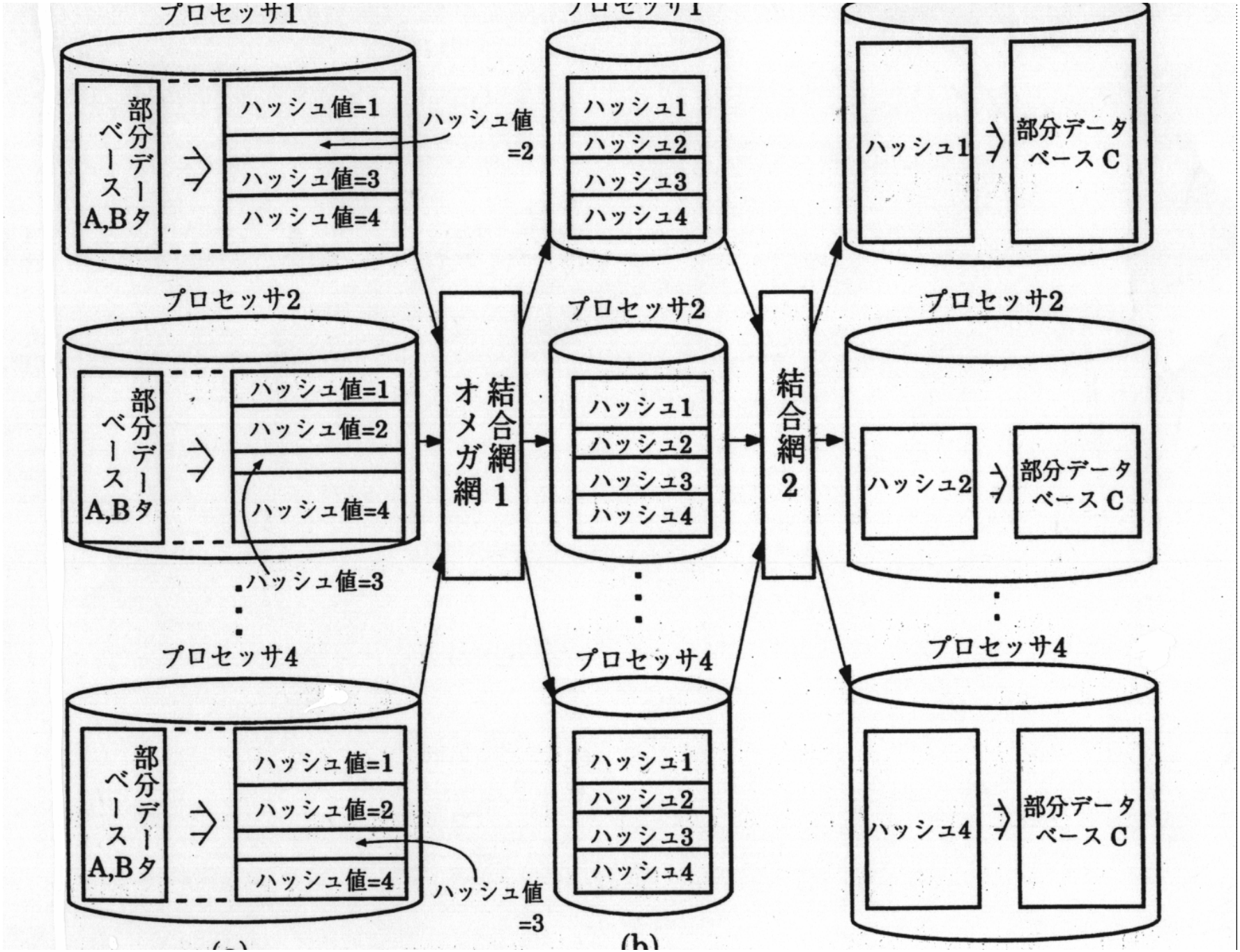
マージ機能付き

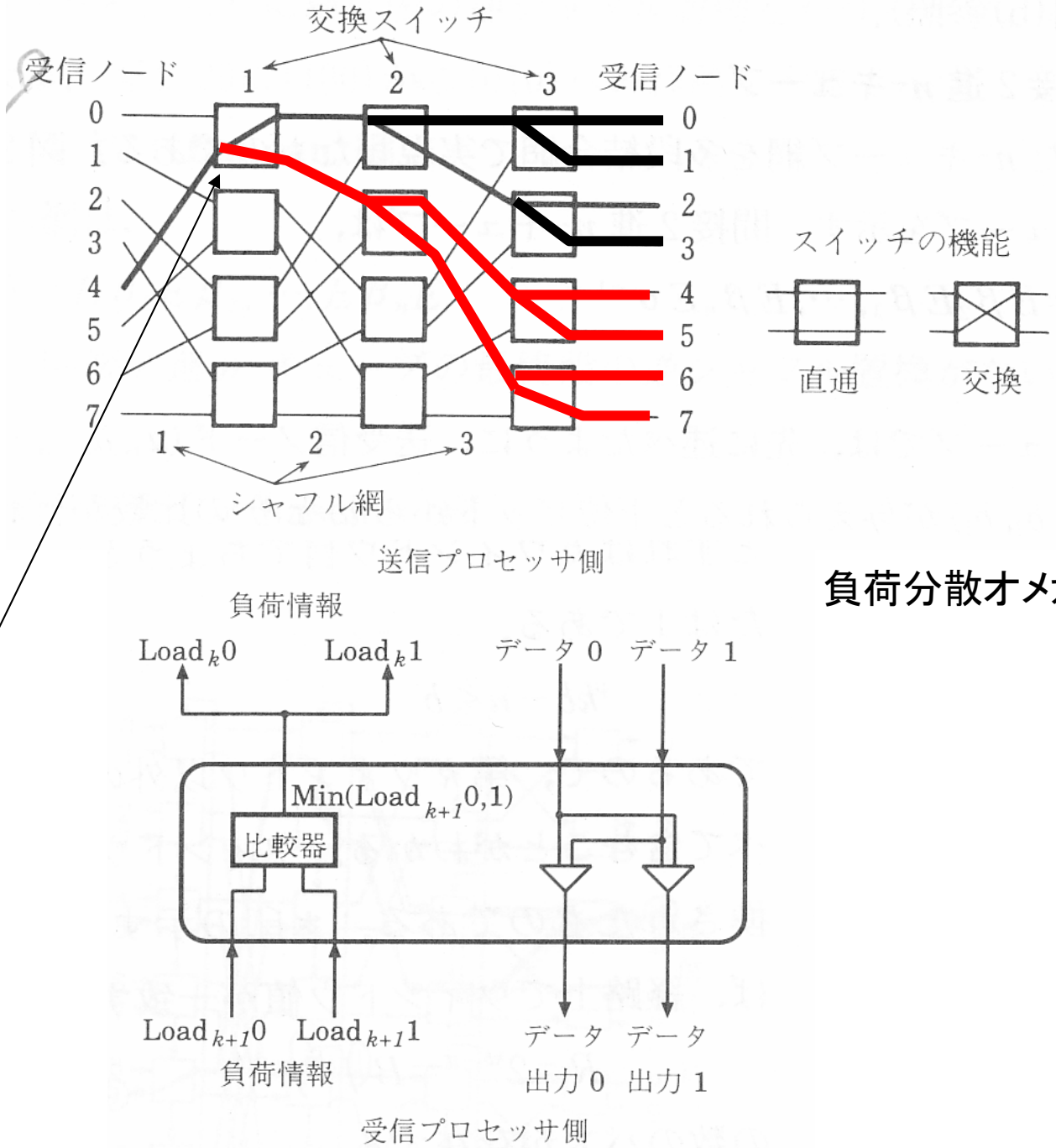
最大200TB

②東大のSDC

ハッシュを利用したデータベースマシン

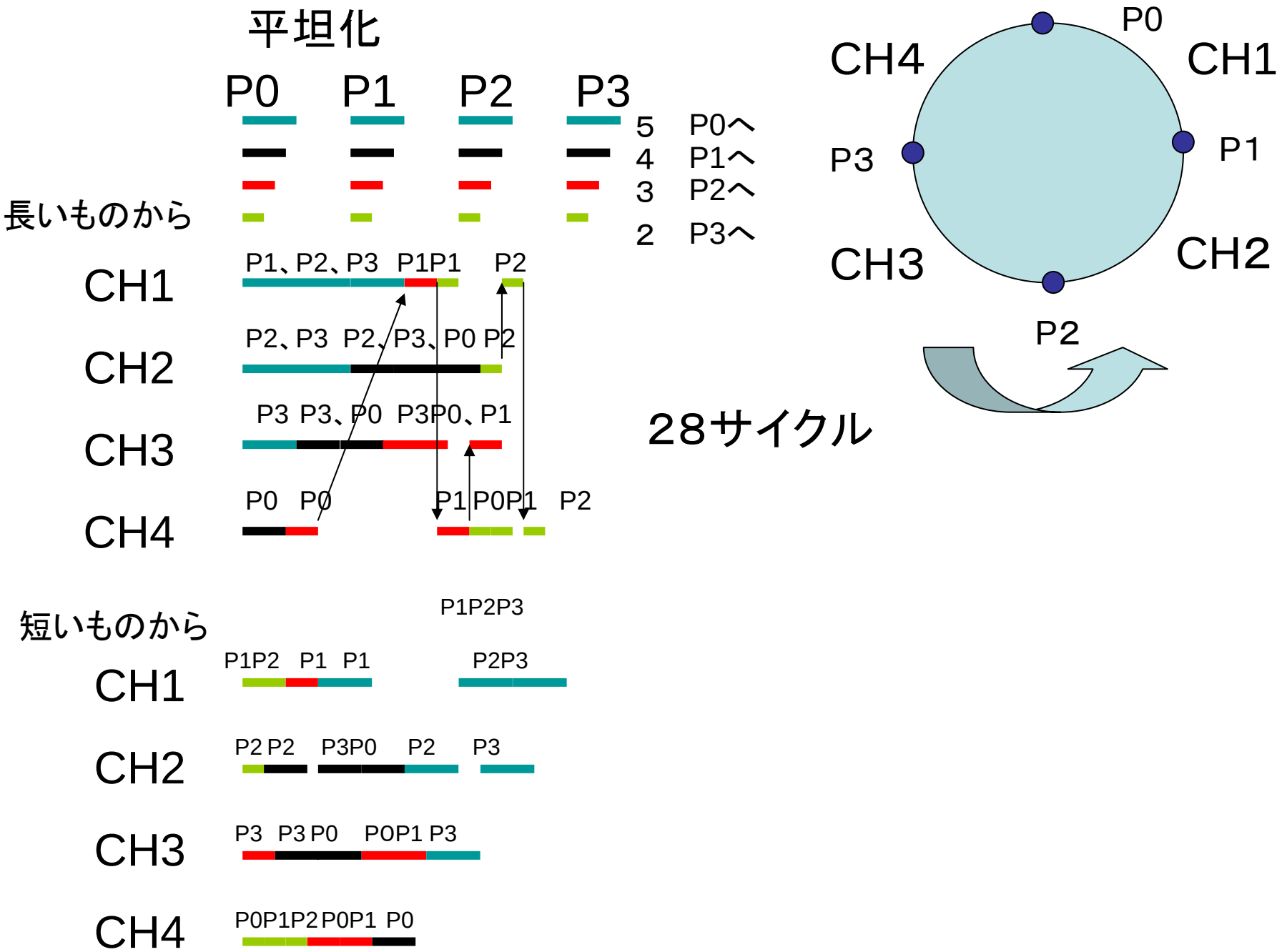
平坦化ネットワーク付き

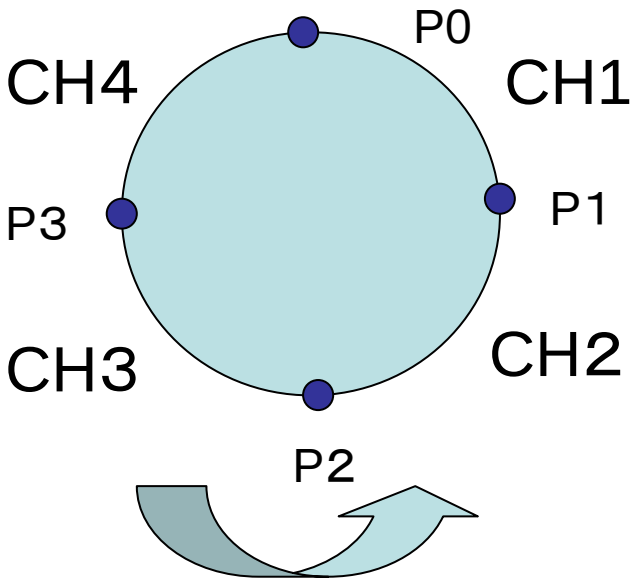

プロセッサ1
部分データベースA,B
ハッシュ値=1
ハッシュ値=3
ハッシュ値=4
ハッシュ値=2
プロセッサ2
部分データベースA,B
ハッシュ値=1
ハッシュ値=2
ハッシュ値=4
ハッシュ値=3
プロセッサ4
部分データベースA,B
ハッシュ値=1
ハッシュ値=2
ハッシュ値=4
ハッシュ値=3
結合網1
オメガ網
プロセッサ1
ハッシュ1
ハッシュ2
ハッシュ3
ハッシュ4
プロセッサ2
ハッシュ1
ハッシュ2
ハッシュ3
ハッシュ4
プロセッサ4
ハッシュ1
ハッシュ2
ハッシュ3
ハッシュ4
(b)
結合網2
ハッシュ1
部分データベースC
プロセッサ2
ハッシュ2
部分データベースC
プロセッサ4
ハッシュ4
部分データベースC



このスイッチで
行き先が2分

負荷分散オメガネット


平坦化
P0
P1
P2
P3
5 P0へ
4 P1へ
3 P2へ
2 P3へ
長いものから
CH1
CH2
CH3
CH4
P1、P2、P3 P1P1 P2
P2、P3 P2、P3、P0 P2
P3 P3、P0 P3P0、P1
P0 P0 P1 P0P1 P2
短いものから
P1P2P3
P1P2 P1 P1 P2P3
P2 P2 P3P0 P2 P3
P3 P3 P0 P0P1 P3
P0P1P2 P0P1 P0
P0
CH4
CH1
P3
P1
CH3
CH2
P2
28サイクル



5 P0へ
4 P1へ
3 P2へ
2 P3へ

バラツキがある場合

P0
P1
P2
P3

CH1 P1 P2 P3 P1

CH2 P2 P3 P3 P0

CH3 P3 P3 P0 P3 P0

CH4 P0 P0 P1 P1

34サイクル

（４）RAID技術

RAID（Redundant Array of Inexpensive Disks）

レベル１から５

レベル１、３、５のシステム

①レベル１

ディスク：単純に２重化、３重化

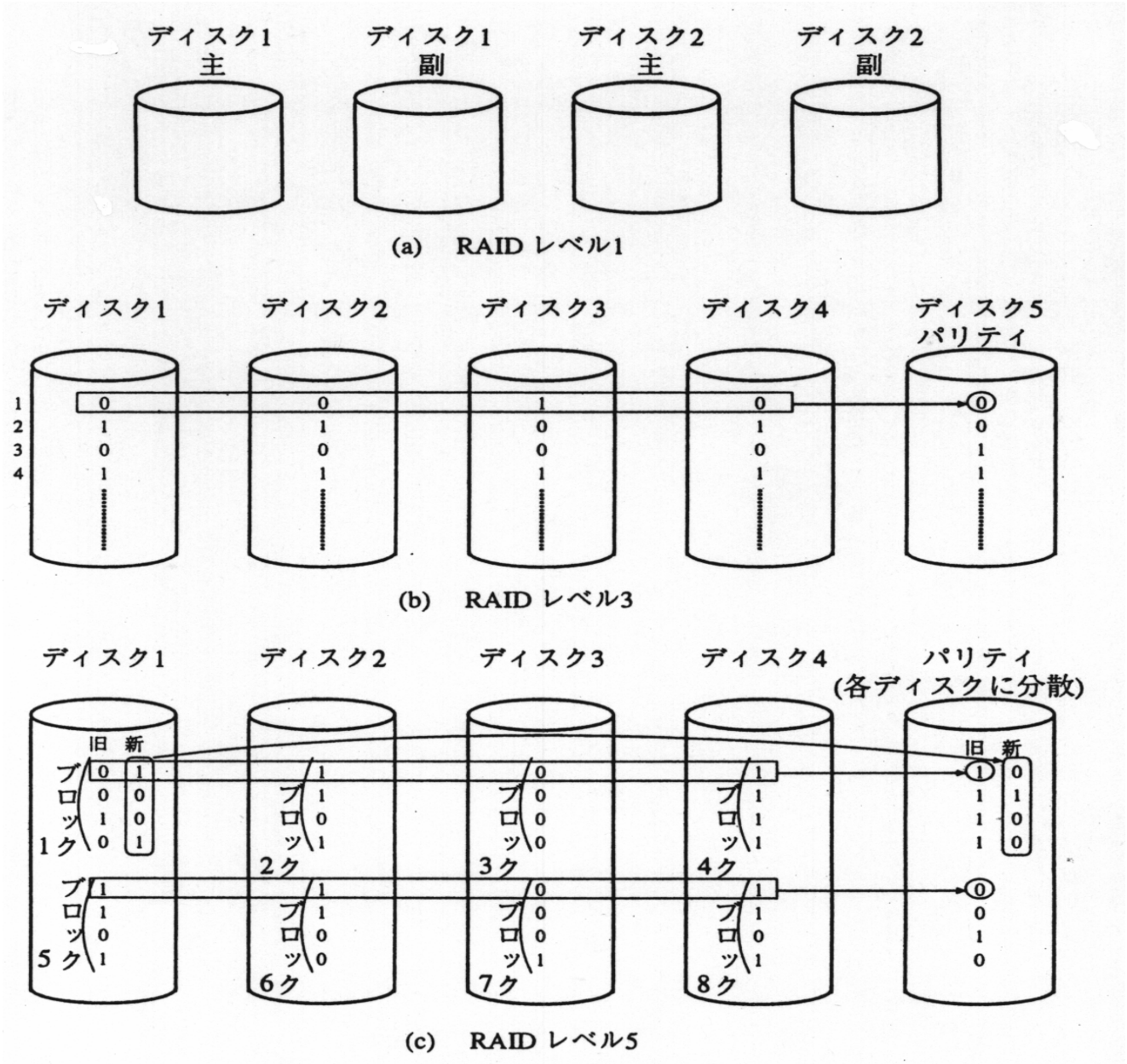
②レベル３

誤り検出方式：パリティ

元のデータ列：	0010	1101	0000	1111
排他的論理和：	1	1	0	0
パリティ列：	0	0	1	1

ディスクを５台用いる場合

パリティ：４ビットごとに計算


ディスク1
主
ディスク1
副
ディスク2
主
ディスク2
副
(a) RAID レベル1
ディスク1
ディスク2
ディスク3
ディスク4
ディスク5
パリティ
1
2
3
4
(b) RAID レベル3
ディスク1
ディスク2
ディスク3
ディスク4
パリティ
(各ディスクに分散)
旧
新
ブロック1
ブロック2
ブロック3
ブロック4
ブロック5
ブロック6
ブロック7
ブロック8
(c) RAID レベル5

・元のビットデータ列：４台のディスクにインタリーブして記憶

**　　ディスクｉ：先の４ビット小片の第ｉビット目のみ記憶**

・パリティ列：５番目のディスクにまとめて記憶

ディスク２が固定的故障した場合

元のデータ列：	0*10	1*01	0*00	1*11
パリティ列：	0	0	1	1
排他的論理和：	1	0	1	0
ディスク２：	0	1	0	1

レベル３

・読出し、書込みに際して各ディスクを１度アクセス

・あるディスクが故障したとき、その他のディスクを１度だけ読み出す

・１つのデータを格納するのに、すべてのディスクを使用する必要

非常に長いデータの格納には都合がよい

短いデータの格納の際には全ディスク専有問題

③レベル５

インタリーブの単位：ブロック

ブロック：ディスクの数セクタ（数百バイト程度）分に対応した長さ

ブロック：４ビット仮定

ディスク：４台装備

各パリティ：異なる４つのブロックの対応ビットごとに付ける

旧ブロック１： 0010

旧パリティ： 1111

新ブロック１： 1001

新パリティの計算

ブロック２、３、４を読み出し、

新ブロック１と排他的論理和をとる

↓↓↓

時間がかかる

旧パリティ＝ブロック２⊕ブロック３

⊕ブロック４⊕旧ブロック１　（５０）

旧ブロック１⊕旧パリティ＝ブロック２⊕

ブロック３⊕ブロック４　（５１）

新パリティ＝新ブロック１⊕ブロック２⊕

ブロック３⊕ブロック４　（５２）

＝新ブロック１⊕旧ブロック１⊕

旧パリティ

新パリティの計算：

ディスクアクセスを３回、

新ブロック１の書込みにディスクアクセス１回

レベル５

・インタリーブの単位が大

短いデータ：複数のディスクにまたがってデータ格

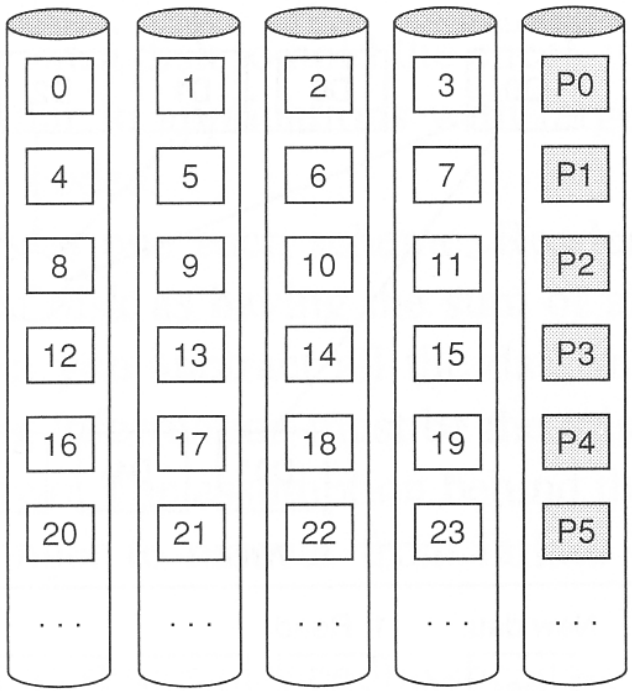
納する必要なし

２つのディスクにアクセス

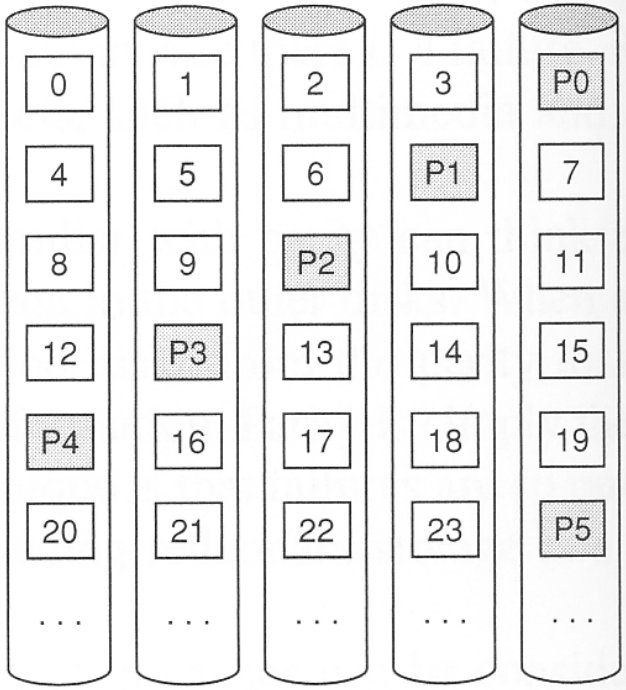
多数の短いデータに対して同時にディスクアク

セス

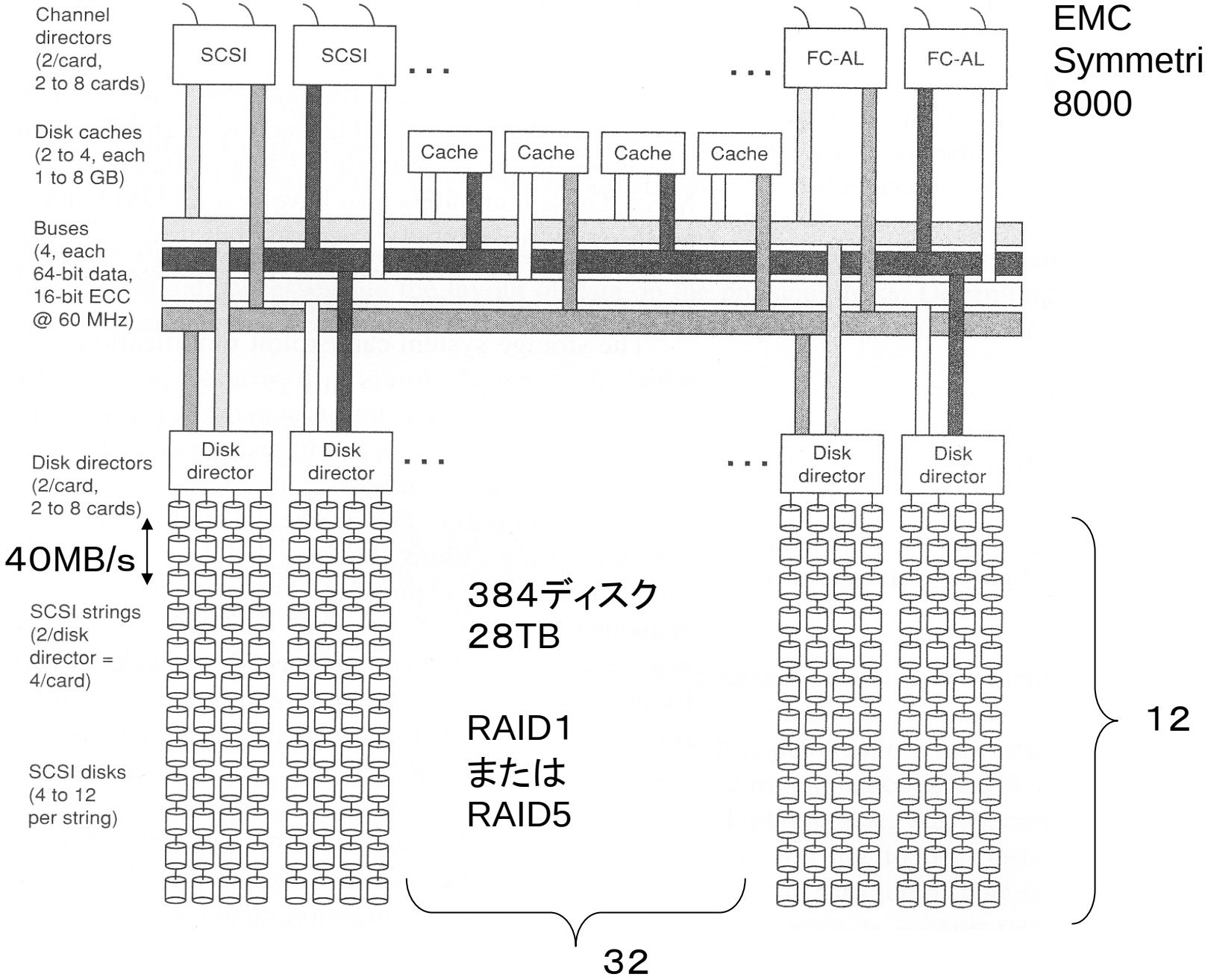
・１回の書込ごとに４回のディスクアクセスが必要


0
4
8
12
16
20
...
1
5
9
13
17
21
...
2
6
10
14
18
22
...
3
7
11
15
19
23
...
P0
P1
P2
P3
P4
P5
...

RAID 4

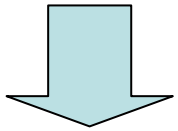

0
4
8
12
P4
20
...
1
5
9
P3
16
21
...
2
6
P2
13
17
22
...
3
P1
10
14
18
23
...
P0
7
11
15
19
P5
...

RAID 5


Channel directors (2/card, 2 to 8 cards)
Disk caches (2 to 4, each 1 to 8 GB)
Buses (4, each 64-bit data, 16-bit ECC @ 60 MHz)
Disk directors (2/card, 2 to 8 cards)
40MB/s
SCSI strings (2/disk director = 4/card)
SCSI disks (4 to 12 per string)
SCSI
SCSI
FC-AL
FC-AL
Cache
Cache
Cache
Cache
Disk director
Disk director
Disk director
Disk director
384ディスク
28TB
RAID1
または
RAID5
32
12
EMC
Symmetrix
8000

データマイニング

関係データベース：SQLで問合せ



規則の発見

経営戦略の立案

ごぼうとこんにゃくを買う人は牛肉を買う場合が多い

データウェアハウス

相関ルールの定義

アイテム：属性

アイテムの集合 $I = \{i_1, i_2, ..., i_m\}$

データベース $D = \{t_1, t_2, ..., t_n\}$

アイテム集合 $t_i \subseteq I$

相関ルール $X \Rightarrow Y$

$X, Y \subset I, X \cap Y = \phi$

X: ごぼう、こんにゃく

Y: 牛肉

例

T_1=(1,3,4)

T_2=(1,2,3,5)

T_3=(2,4)

T_4=(1,2)

T_5=(1,3,5)

アイテム：1，2，3，4，5

アイテムの集合I:{1，2，3，4，5}

アイテム集合t_i：Iの部分集合

データベース：T_1,T_2,T_3,T_4,T_5

長さkのアイテム集合ti：要素数がk個

最小サポート値、確信値：40%

サポート値(X)：Dの中でのXの割合

確信値(X ∪ Y): Xを含むものの内でX ∪ Yを含むものの割合

$$\text{確信値} = \frac{\text{サポート値(X} \cup \text{Y)}}{\text{サポート値(X)}}$$

X ：ごぼう、こんにゃく

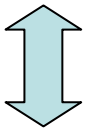
X ∪ Y：ごぼう、こんにゃく、牛肉

アイテム集合の要素の数が増大するにつれ、

サポート値は単調非増大する

サポート値（X∪Y）≦サポート値（X）

相関ルールを見つけること



ユーザ指定のサポート値、確信値の最小値を満たす相関ルールをすべて見出すこと

ステップ１：最小サポート値を満たすアイテム集合（頻出アイテム集合）を見つける

ステップ２：頻出アイテム集合から確信値の最小値を満たすものを見つける

逐次Aprioriアルゴリズム（IBM）

- 初期ステップ：生起確率＞最小サポート値であるアイテムを長さ1の頻出アイテム集合とする
- ステップ1：長さk-1の頻出アイテム集合から長さkの候補アイテム集合を作成する。

　長さk-1の頻出アイテム集合で最初のk-2個のアイテムが共通なものを選ぶ。

①$(i_1,i_2,\ldots,i_{k-2}、i'_{k-1})$、$(i_1,i_2,\ldots,i_{k-2}、i''_{k-1})$をジョインした$(i_1,i_2,\ldots,i_{k-2}、i'_{k-1},i''_{k-1})$が候補に入るか？

②$(i_2,\ldots,i_{k-2}、i'_{k-1},i''_{k-1})$ $(i_1,i_3,\ldots,i_{k-2}、i'_{k-1},i''_{k-1})$ ，…，$(i_1, i_2, i_3,\ldots,i_{k-3}、i'_{k-1},i''_{k-1})$がk-1頻出アイテム集合にないものは排除して長さkの候補アイテム集合$(i_1,i_2,\ldots,i_{k-2}、i'_{k-1},i''_{k-1})$を作成

- ステップ２

データベースを検索し、候補アイテム集合のサポート値を求める。最小値より大きなものを長さkの頻出アイテム集合とする。

例

T_1=(1,3,4)

T_2=(1,2,3,5)

T_3=(2,4)

T_4=(1,2)

T_5=(1,3,5)

最小サポート値、確信値：40%

①長さ1の最頻アイテム集合
　{1}、{2}、{3}、{4}、{5}
②長さ2の候補アイテム集合
　{1，2}、{1，3}、{1，4}、{1，5}、{2，3}、
　{2，4}、{2，5}、{3，4}、{3，5}、{4，5}
③データベース検索
④長さ2の最頻アイテム集合
　{1，2}、{1，3}、{1，5}、{3，5}
⑤長さ3の候補アイテム集合
　{1，2}、{1，3}ジョイン{1，2，3}：
　　{2，3}長さ2最頻アイテムになし
　{1，2}、{1，5}ジョイン{1，2，5}
　　{2，5}長さ2最頻アイテムになし
　{1，3}、{1，5}ジョイン{1，3，5}
　　{3，5}長さ2最頻アイテムにあり
　{1，3，5}が唯一の長さ3の候補アイテム集合
⑥データベース検索
⑦長さ3の最頻アイテム集合
　{1，3，5}

T1=(1,3,4)
T2=(1,2,3,5)
T3=(2,4)
T4=(1,2)
T5=(1,3,5)

並列Apriori法1：NPA
（IBM　Count　Distribution）

- 候補アイテム集合のコピーを全プロセッサが有する
- データベースは各プロセッサに分割

１　各プロセッサの登録表に長さk-1の候補アイテム集合全体を設定する

２　データベース読み出し、候補表に登録されておればカウント＋１

３　カウント値を交換。長さk-1の頻出アイテム集合を作成

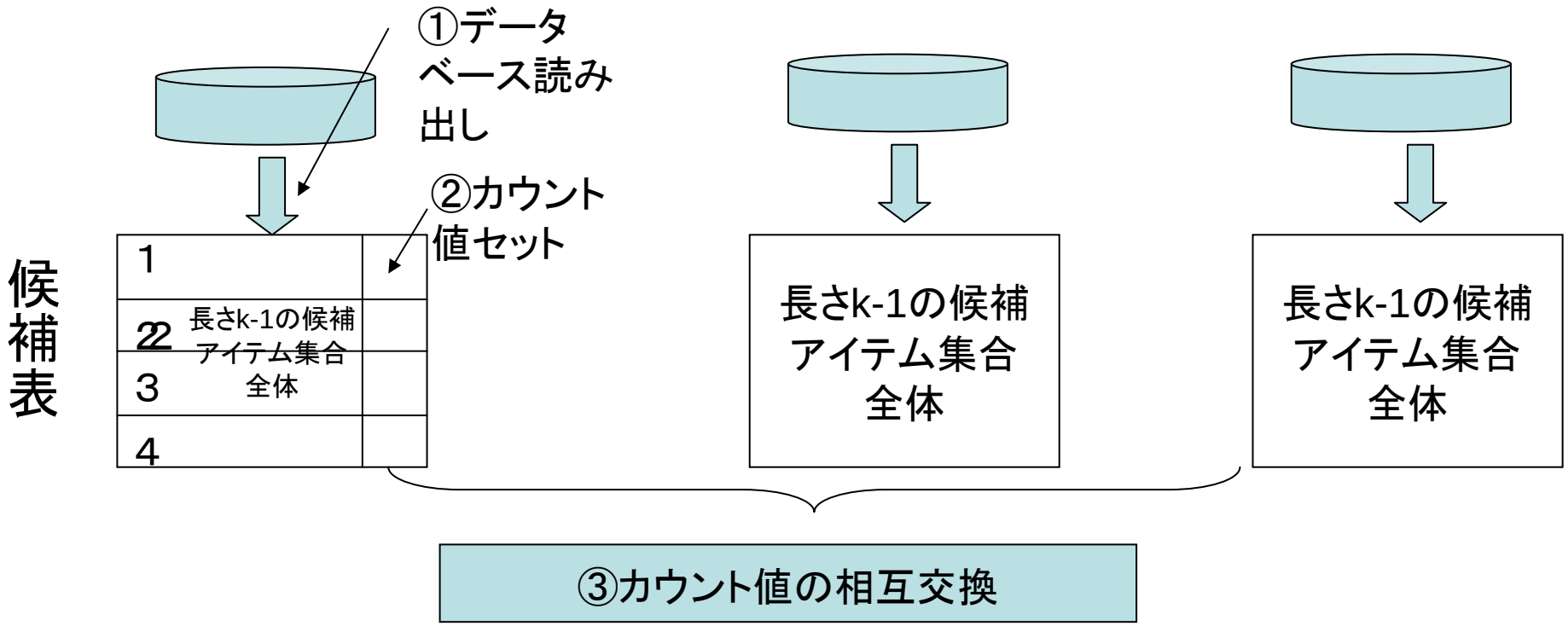
４　各プロセッサで長さk-1頻出アイテム集合から長さkの候補アイテム集合全体を作成し、各プロセッサの候補表に登録

データベースはプロセッサ間で転送なし

長さ2の候補アイテム集合のサイズが大で溢れる。

例：長さ1の頻出アイテム集合の数：10^4

　　長さ2の候補アイテム集合の数：　10^8

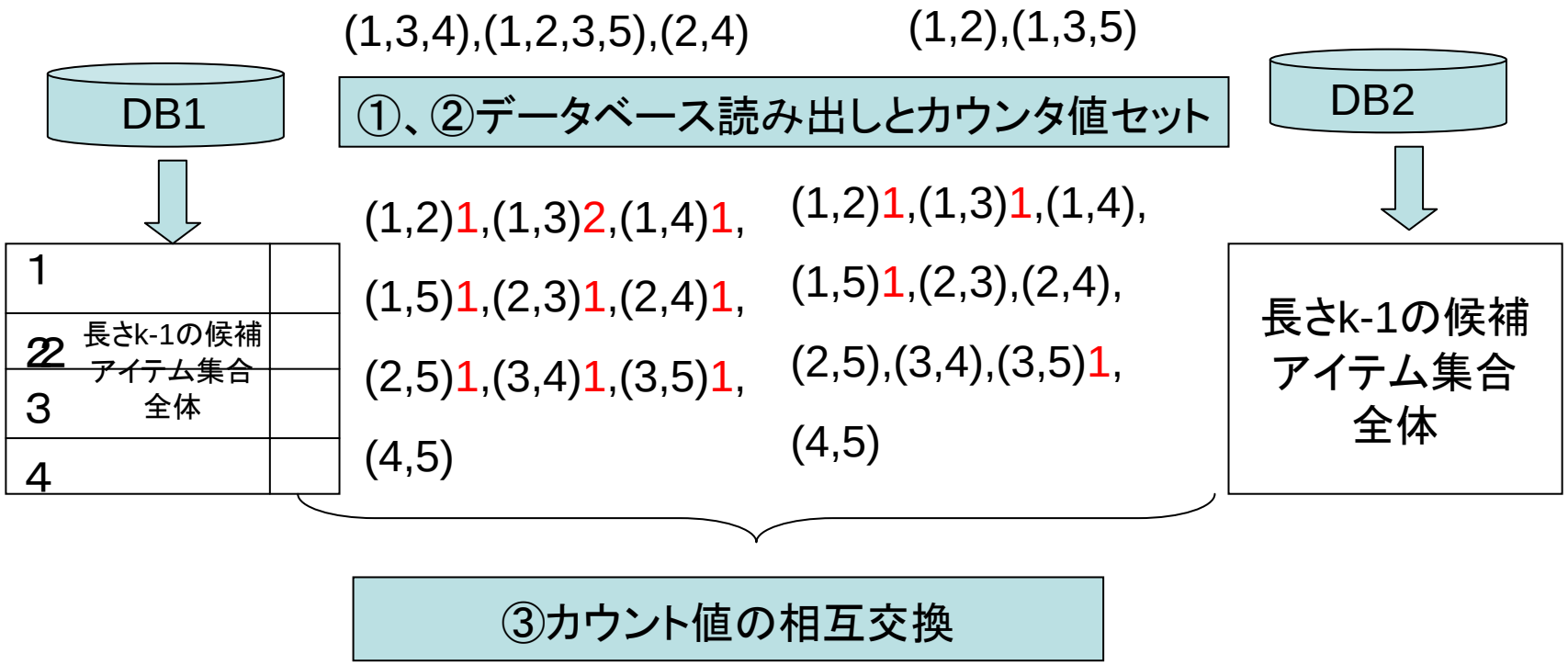


④長さK-1頻出アイテム集合決定　長さk-1頻出アイテム集合決定　長さk-1頻出アイテム集合決定

⑤長さkの候補アイテム集合の作成　長さkの候補アイテム集合の作成　長さkの候補アイテム集合の作成

同一コピー作成　同一コピー作成　同一コピー作成

同一処理


(1,3,4),(1,2,3,5),(2,4)
(1,2),(1,3,5)
DB1
①、②データベース読み出しとカウンタ値セット
DB2
(1,2)1,(1,3)2,(1,4)1,
(1,5)1,(2,3)1,(2,4)1,
(2,5)1,(3,4)1,(3,5)1,
(4,5)
(1,2)1,(1,3)1,(1,4),
(1,5)1,(2,3),(2,4),
(2,5),(3,4),(3,5)1,
(4,5)
1
2
3
4
長さk-1の候補アイテム集合全体
長さk-1の候補アイテム集合全体
③カウント値の相互交換
④長さk-1の頻出アイテム集合の決定
(1,2),(1,3),(1,5),(3,5)
長さk-1の頻出アイテム集合の決定
⑤長さkの候補アイテム集合の作成
(1,3,5)
長さkの候補アイテム集合の作成
同一コピー作成
(1,3,5)
同一コピー作成
同一処理

並列Apriori法２：SPA
（IBM　Data Distribution)

- 候補アイテム集合を各プロセッサの候補表に分散配置
- データベース分割

１　長さk-1の候補アイテム集合の一部を候補表に登録

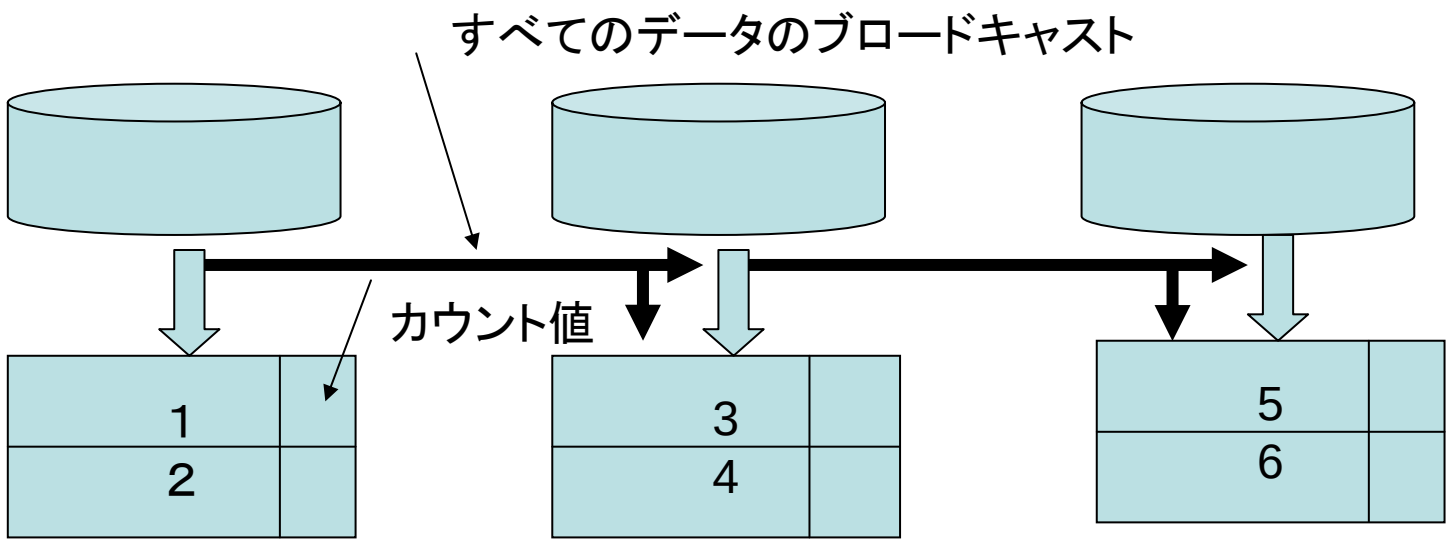
２　各プロセッサでデータベースを読み、候補表に登録されていればカウント＋１

３　他のプロセッサからのデータベースに1，2を実行

４　長さk-1の頻出アイテム集合を作成し、ブロードキャスト

５　長さkの候補アイテム集合を作成し、各プロセッサの候補表に分割配置

データベースを他プロセッサにブロードキャスト：オーバヘッド



長さk-1候補アイテム集合（部分）

長さk-1候補アイテム集合(部分）

長さk-1候補アイテム集合(部分）

長さk-1頻出アイテム集合（部分）作成とブロードキャスト

長さk-1頻出アイテム集合（部分）作成とブロードキャスト

長さk-1頻出アイテム集合（部分）作成とブロードキャスト

長さk-1頻出アイテム集合（全）

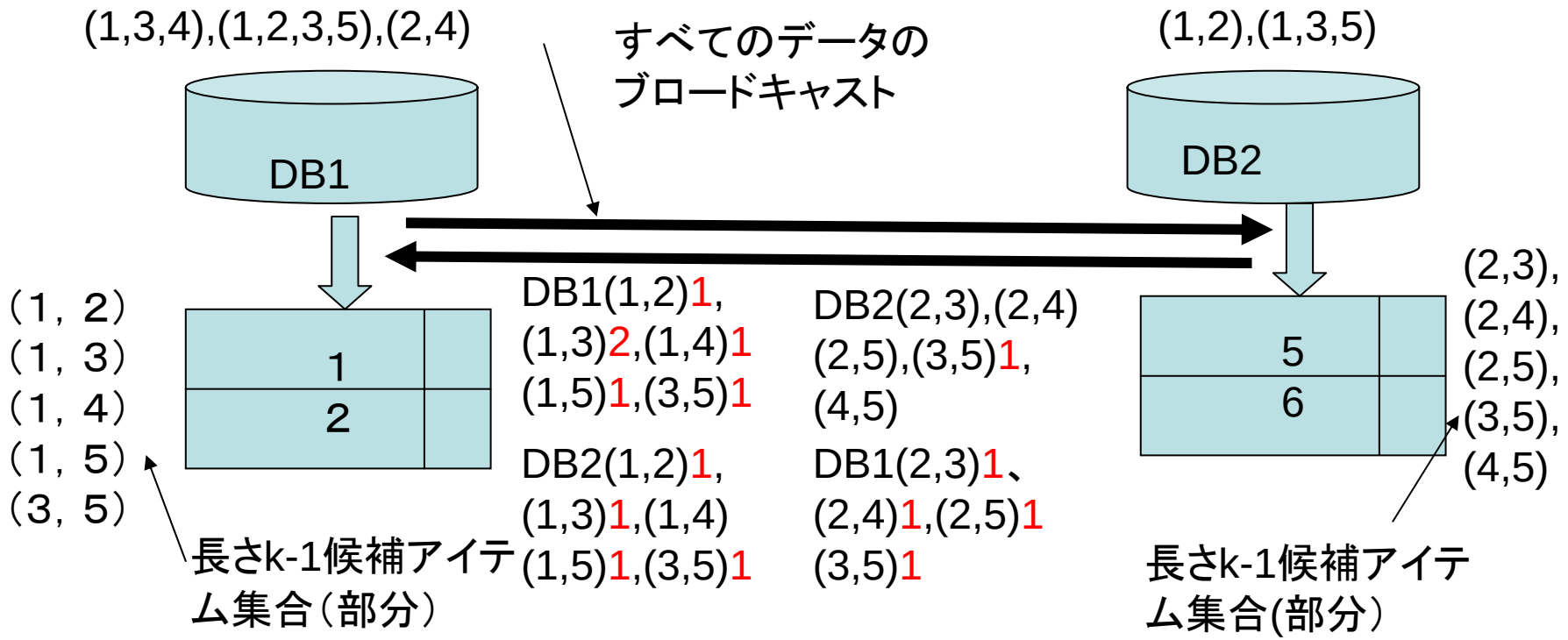
長さk-1頻出アイテム集合（全）

長さk-1頻出アイテム集合（全）

長さk候補アイテム集合作成（部分）

長さk候補アイテム集合作成（部分）

長さk候補アイテム集合作成（部分）



長さk-1頻出アイテム集合（部分）作成とブロードキャスト

(1,2),(1,3),
(1,5),(3,5)

(3,5)

長さk-1頻出アイテム集合（全）

(1,2),(1,3),(1,5),(3,5)

(1,3,5)

長さk候補アイテム集合作成（部分）

長さk-1頻出アイテム集合（部分）作成とブロードキャスト

長さk-1頻出アイテム集合（全）

長さk候補アイテム集合作成（部分）

並列Apriori法３：HPA
(ハッシュ法：東大）

- 候補アイテム集合をハッシュで各プロセッサに分割
- データベース分割

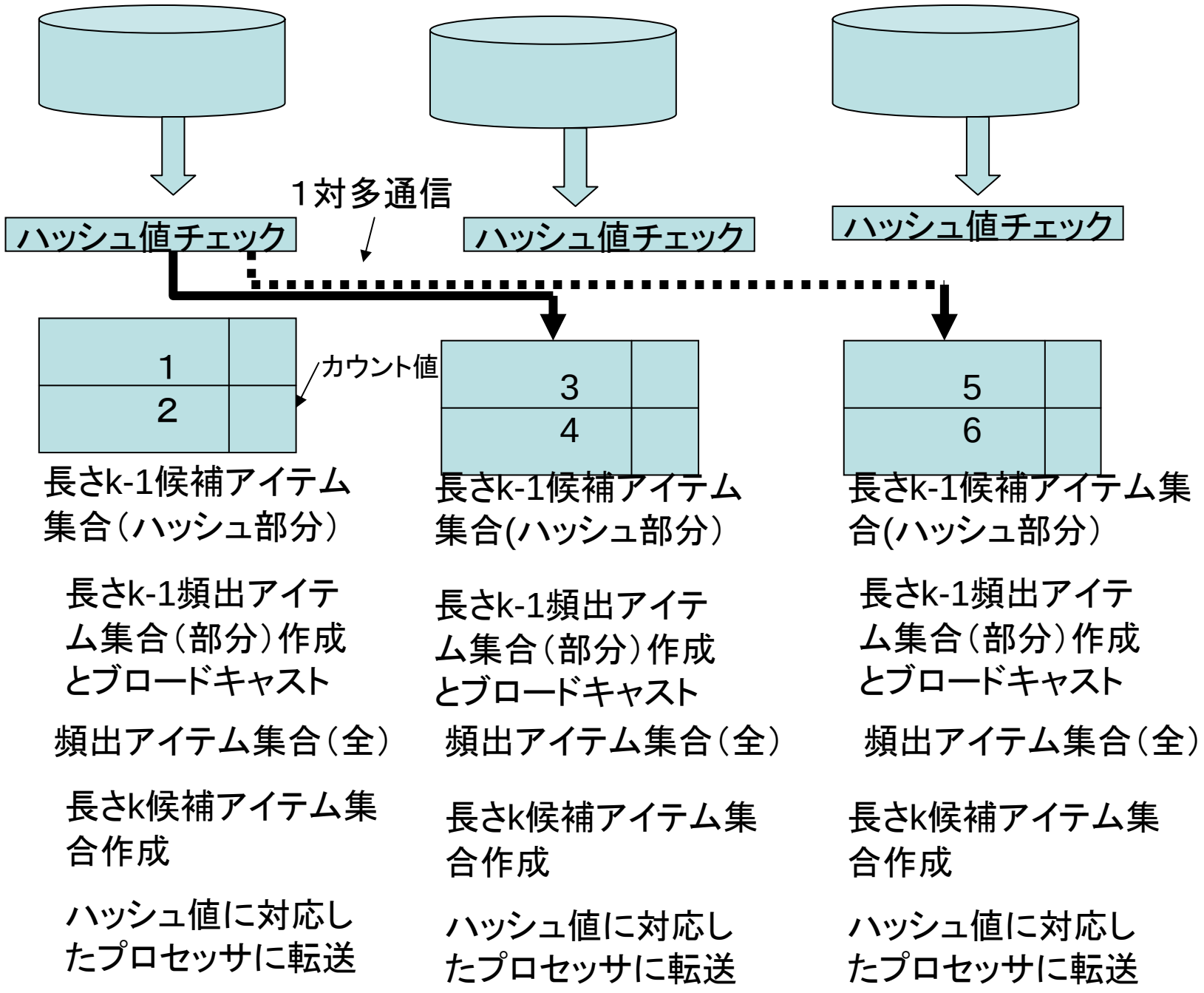
１　長さk-1の候補アイテム集合をハッシュで各プロセッサに分割配置　：
　　例アイテム3を含むもの　　ー＞P1へ
　　含まないものー＞P2へ

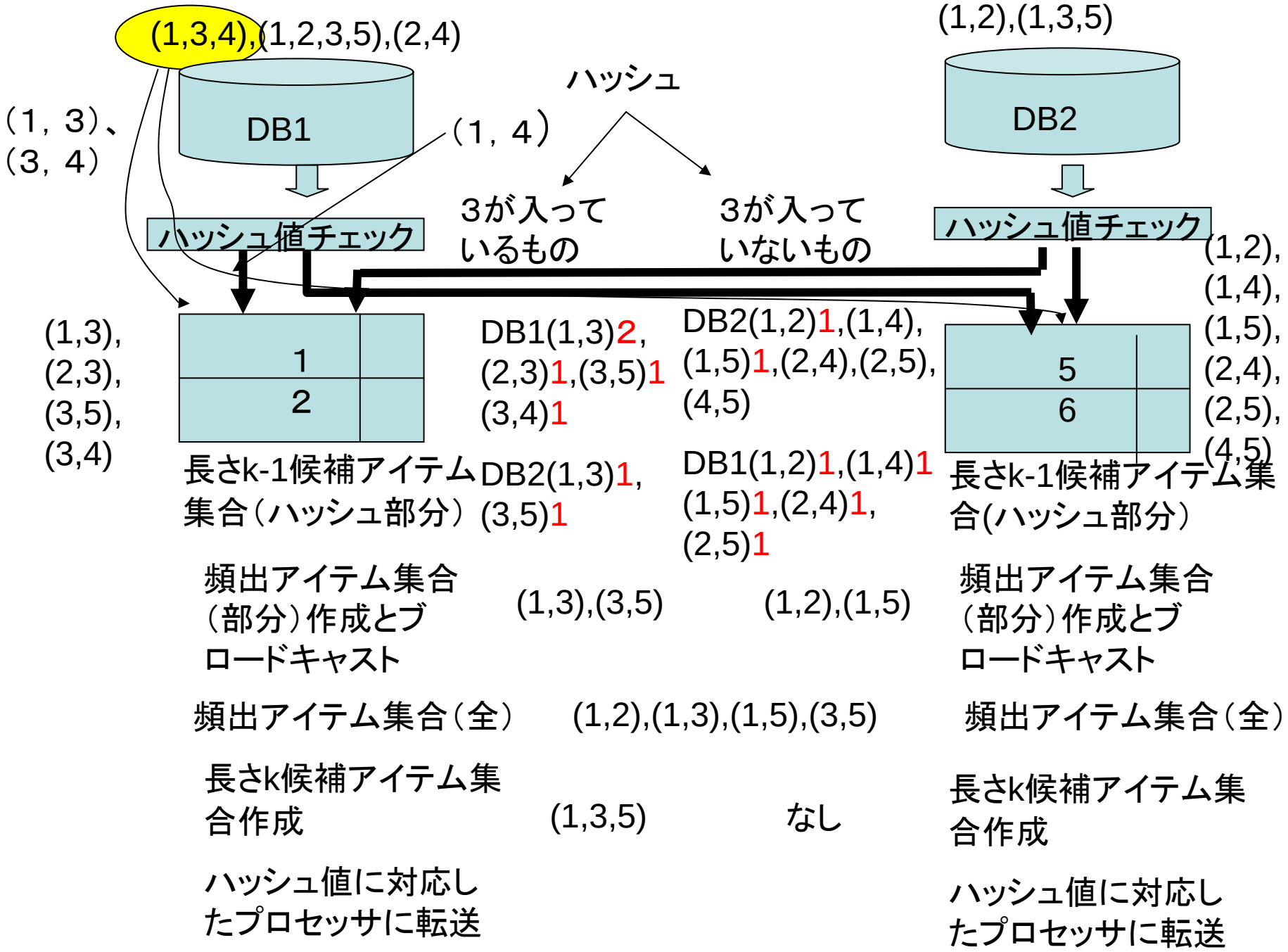
２　データベースを読み、長さk-1のアイテム集合すべてについて、1と同じハッシュにより、プロセッサ番号をもとめ、そこに転送し、登録表にあればカウント＋１
　　例データ（１，２，３，４）
　　（１，２）、（１，４）、（２，４）－＞P2へ
　　（１，３）、（２，３）、（３，４）－＞P1へ

３　長さk-1の頻出アイテム集合を各プロセッサで求め、ブロードキャスト

４　長さkの候補アイテム集合を作成し、ハッシュ値に対応するプロセッサに分割配置

HPA-ELD：　アイテム集合に偏りがあり、特定のプロセッサが過負荷
　　他のプロセッサの候補表にコピーを置き、負荷の分散化
　　例：アイテム3の生起確率が大極めて大のとき、
　　P1：過負荷　－＞　P2にも回すようにする


1対多通信
ハッシュ値チェック
ハッシュ値チェック
ハッシュ値チェック
1
2
カウント値
3
4
5
6
長さk-1候補アイテム集合（ハッシュ部分）
長さk-1頻出アイテム集合（部分）作成とブロードキャスト
頻出アイテム集合（全）
長さk候補アイテム集合作成
ハッシュ値に対応したプロセッサに転送
長さk-1候補アイテム集合(ハッシュ部分）
長さk-1頻出アイテム集合（部分）作成とブロードキャスト
頻出アイテム集合（全）
長さk候補アイテム集合作成
ハッシュ値に対応したプロセッサに転送
長さk-1候補アイテム集合(ハッシュ部分）
長さk-1頻出アイテム集合（部分）作成とブロードキャスト
頻出アイテム集合（全）
長さk候補アイテム集合作成
ハッシュ値に対応したプロセッサに転送


(1,3,4),(1,2,3,5),(2,4)
DB1
(1，3)、
(3，4)
(1，4)
ハッシュ
3が入っているもの
3が入っていないもの
(1,2),(1,3,5)
DB2
ハッシュ値チェック
ハッシュ値チェック
(1,3),
(2,3),
(3,5),
(3,4)
1
2
DB1(1,3)2,
(2,3)1,(3,5)1
(3,4)1
DB2(1,2)1,(1,4),
(1,5)1,(2,4),(2,5),
(4,5)
5
6
(1,2),
(1,4),
(1,5),
(2,4),
(2,5),
(4,5)
長さk-1候補アイテム集合（ハッシュ部分）
DB2(1,3)1,
(3,5)1
DB1(1,2)1,(1,4)1
(1,5)1,(2,4)1,
(2,5)1
長さk-1候補アイテム集合(ハッシュ部分）
頻出アイテム集合（部分）作成とブロードキャスト
(1,3),(3,5)
(1,2),(1,5)
頻出アイテム集合（部分）作成とブロードキャスト
頻出アイテム集合（全）
(1,2),(1,3),(1,5),(3,5)
頻出アイテム集合（全）
長さk候補アイテム集合作成
(1,3,5)
なし
長さk候補アイテム集合作成
ハッシュ値に対応したプロセッサに転送
ハッシュ値に対応したプロセッサに転送

データ数:2048K

平均アイテム数:15

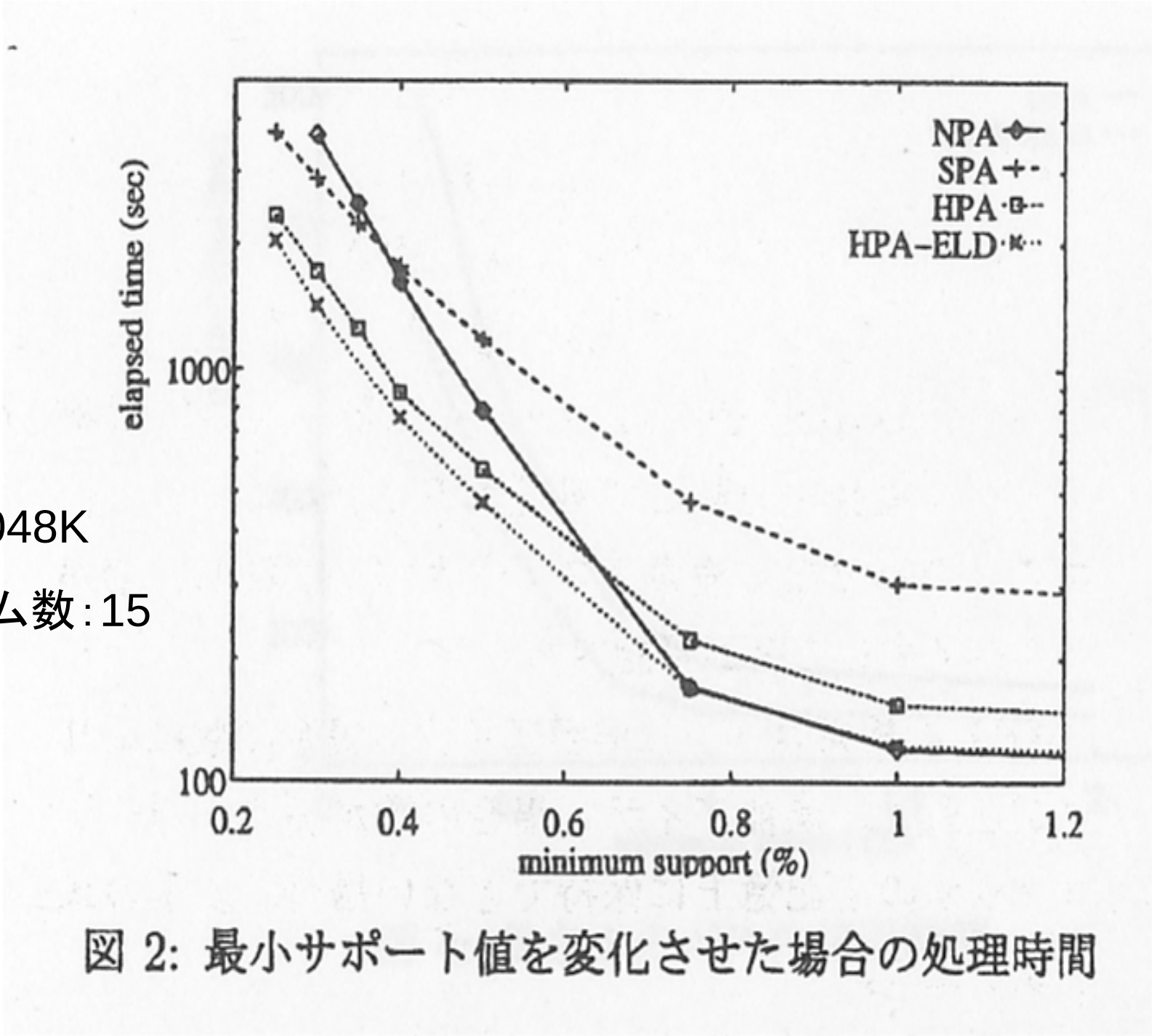


図 2: 最小サポート値を変化させた場合の処理時間

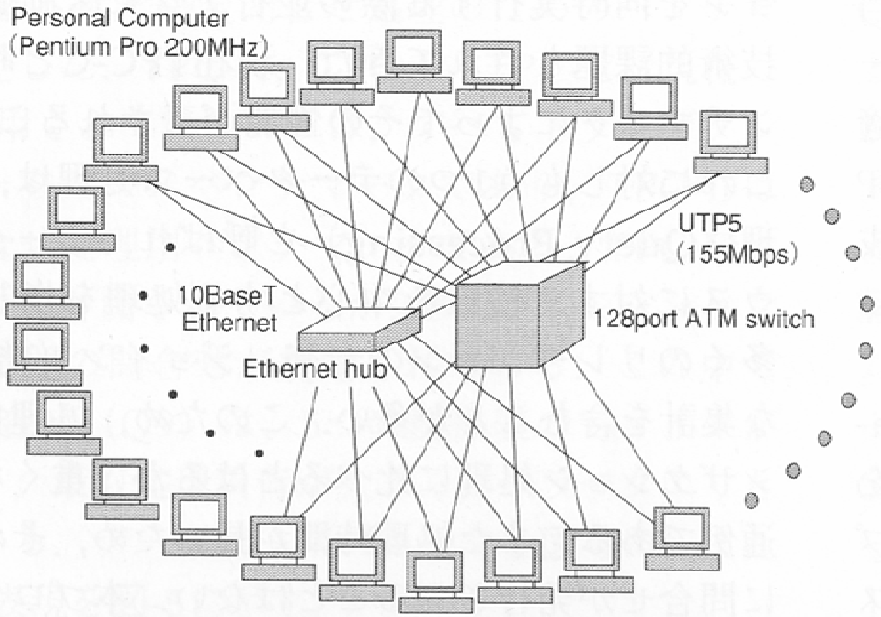
文献

1　喜連川優：データベースとデータマイニングにおける並列処理、情報処理、39巻、11号、pp.1089-1094、1998

2　新谷隆彦、喜連川優：データマイニングにおける相関関係抽出の並列処理方式、JSPP,pp.97-104、平成8年

3　R.Agrawal,J.C.Shafer: Parallel Mining of Association Rules,IEEE Trans.Data Engineering,8,6,pp.962-969,1996

4　R.Agrawal,R.Srikant: Fast Algorithms for Mining Association Rules,Proc of 20th VLDB Conf,487-499、1994



CPU	Intel Pentium Pro 200MHz
Chip Set	Intel 440FX
Main Memory	64MB
Disk Drives	For OS Western Digital Caviar 32500 (EIDE, 2.5GB)
	For databases Seagate Barracuda (Ultra SCSI, 4.3GB, 7200RPM)
SCSI HBA	Adaptec AHA 2940 Ultra Wide
ATM NIC	Interphase 5515 PC ATM Adapter
OS	Solaris2.5.1 for x86

図-1　NEDO-100 PCクラスタの構成

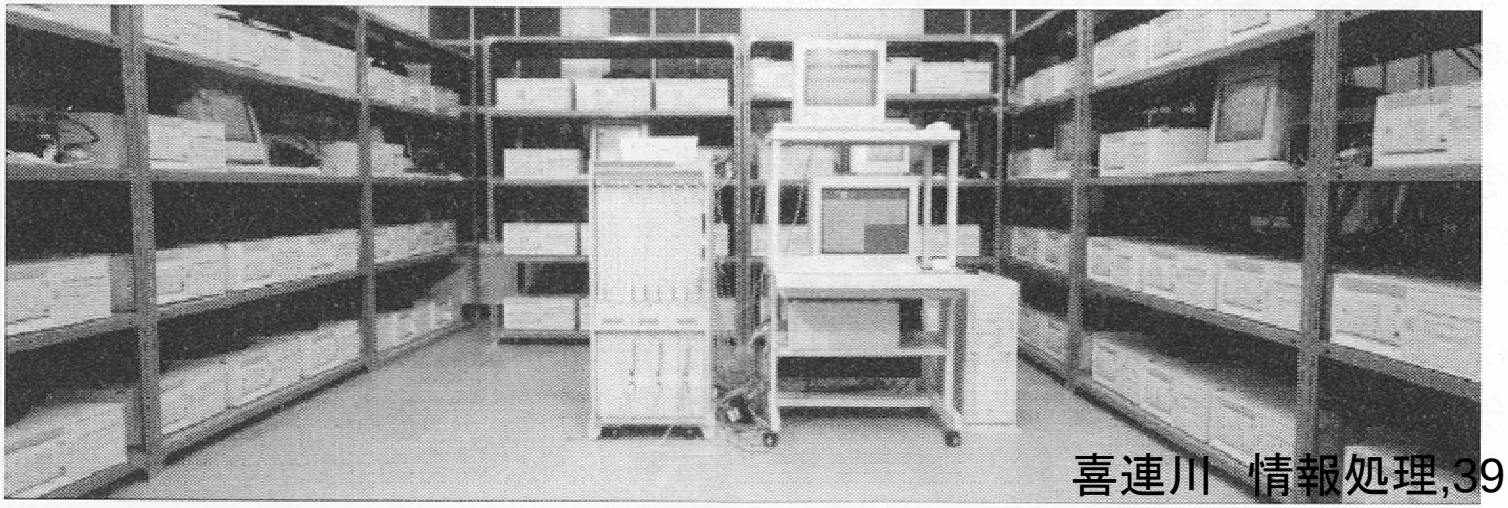
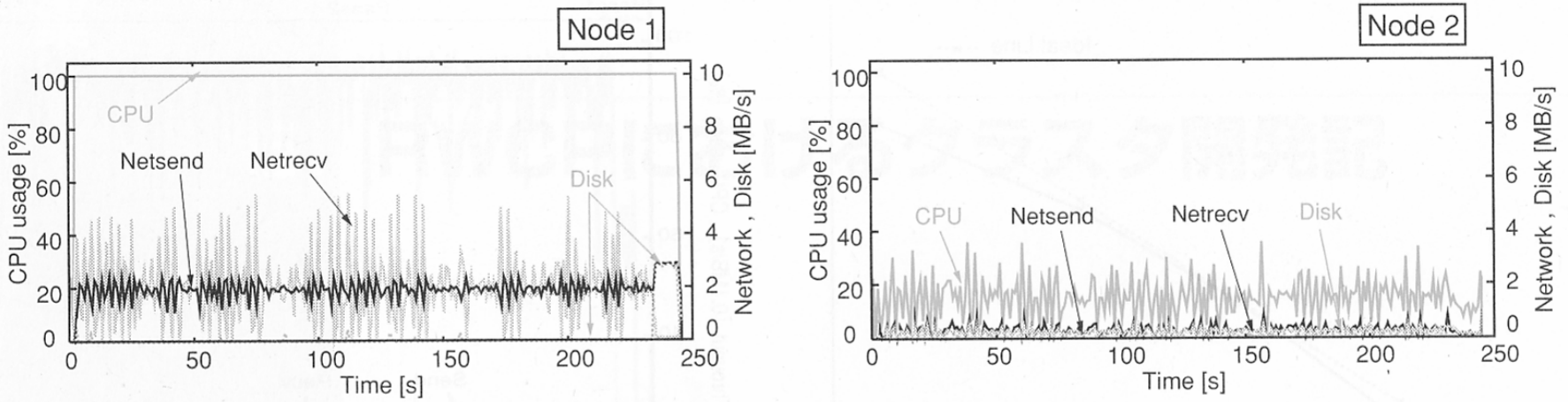
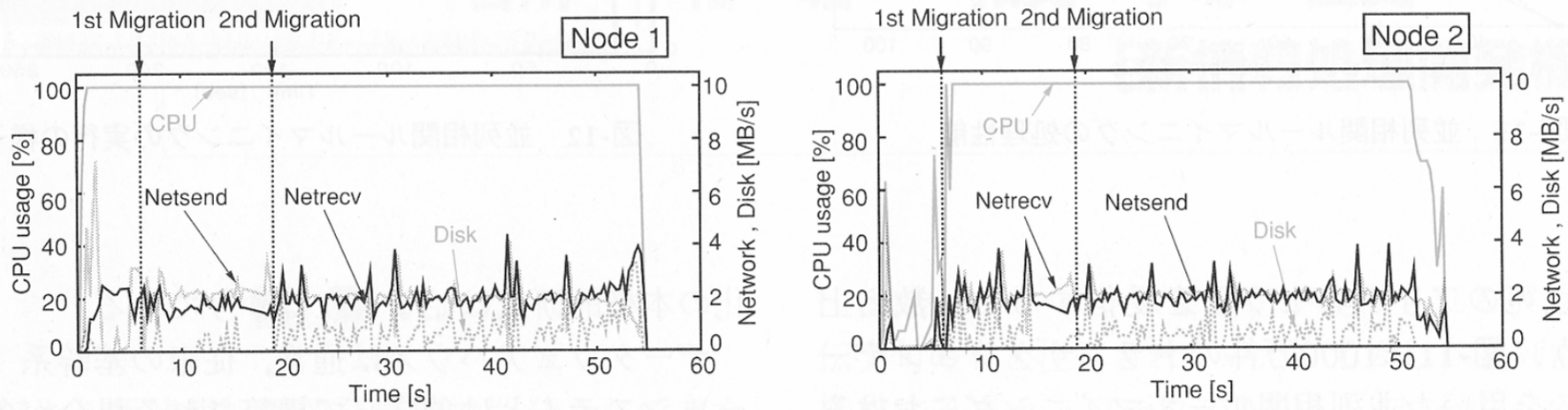


図-2　NEDO-100概観

喜連川　情報処理,39,
11,p.1089, 1998



(a) Without Dynamic Load Balancing



(b) With Dynamic Load Balancing

図-10　実行時動的負荷分散機構の効果

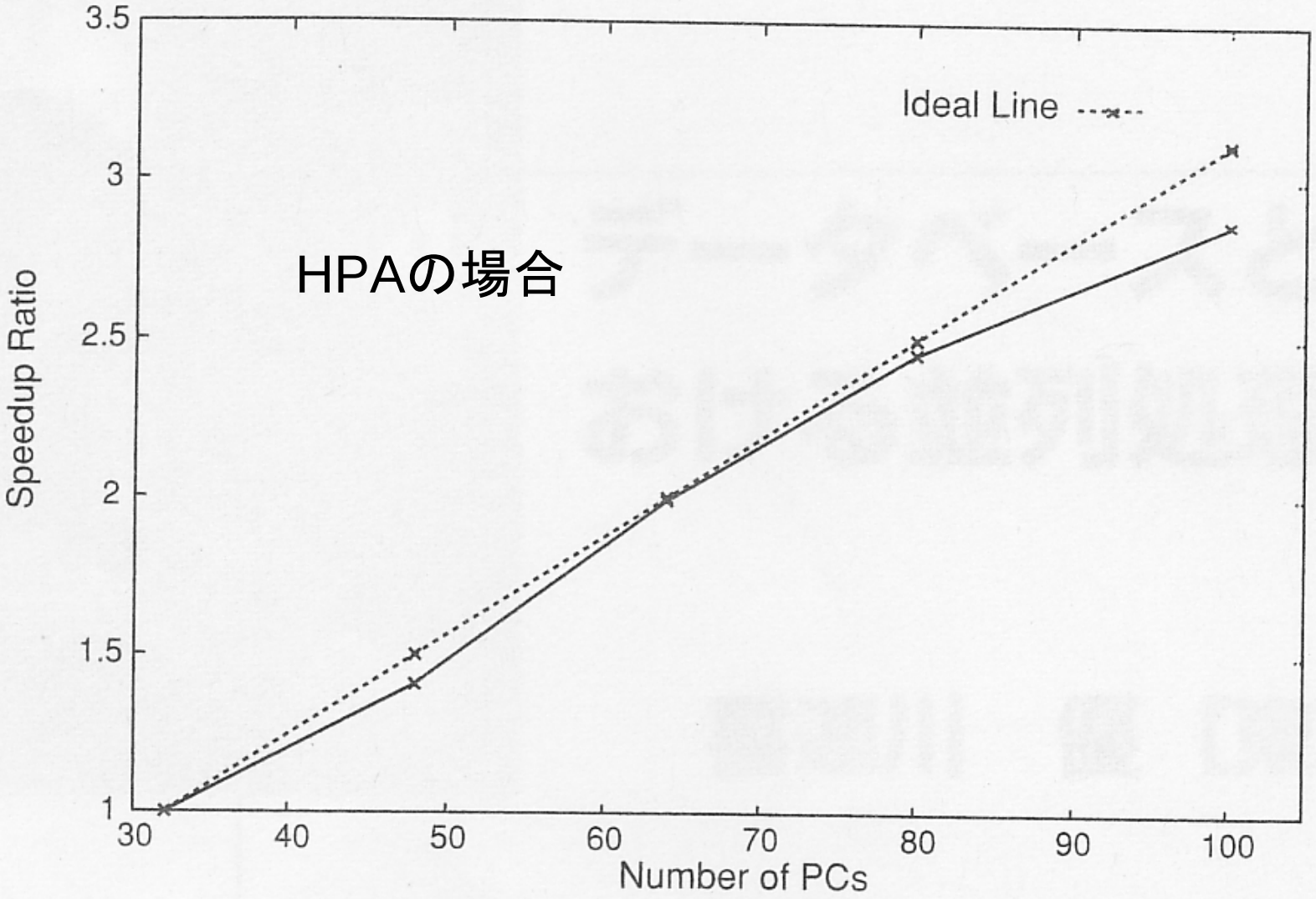


図-11　並列相関ルールマイニングの処理性能

６．６人工知能

①情報検索の並列化

知識データベースの構築とそれに対する

情報検索を高速化

②推論規則の並列化

プロダクションシステム、論理型言語PROLOG

③新しい探索アルゴリズムの並列化

ニューロコンピュータ、

シミュレーティッド・アニーリング、

遺伝的アルゴリズム

④分散協調処理

6.6.1情報検索の並列処理

意味ネットワーク

①キーワード検索：連想メモリ

②マーカ伝播（○,●,※）

③ビット演算

並列処理

通信を非同期化したSIMD方式

（たとえば、Connection Machine）

MBR：Memory Based Reasoning

Connectionist

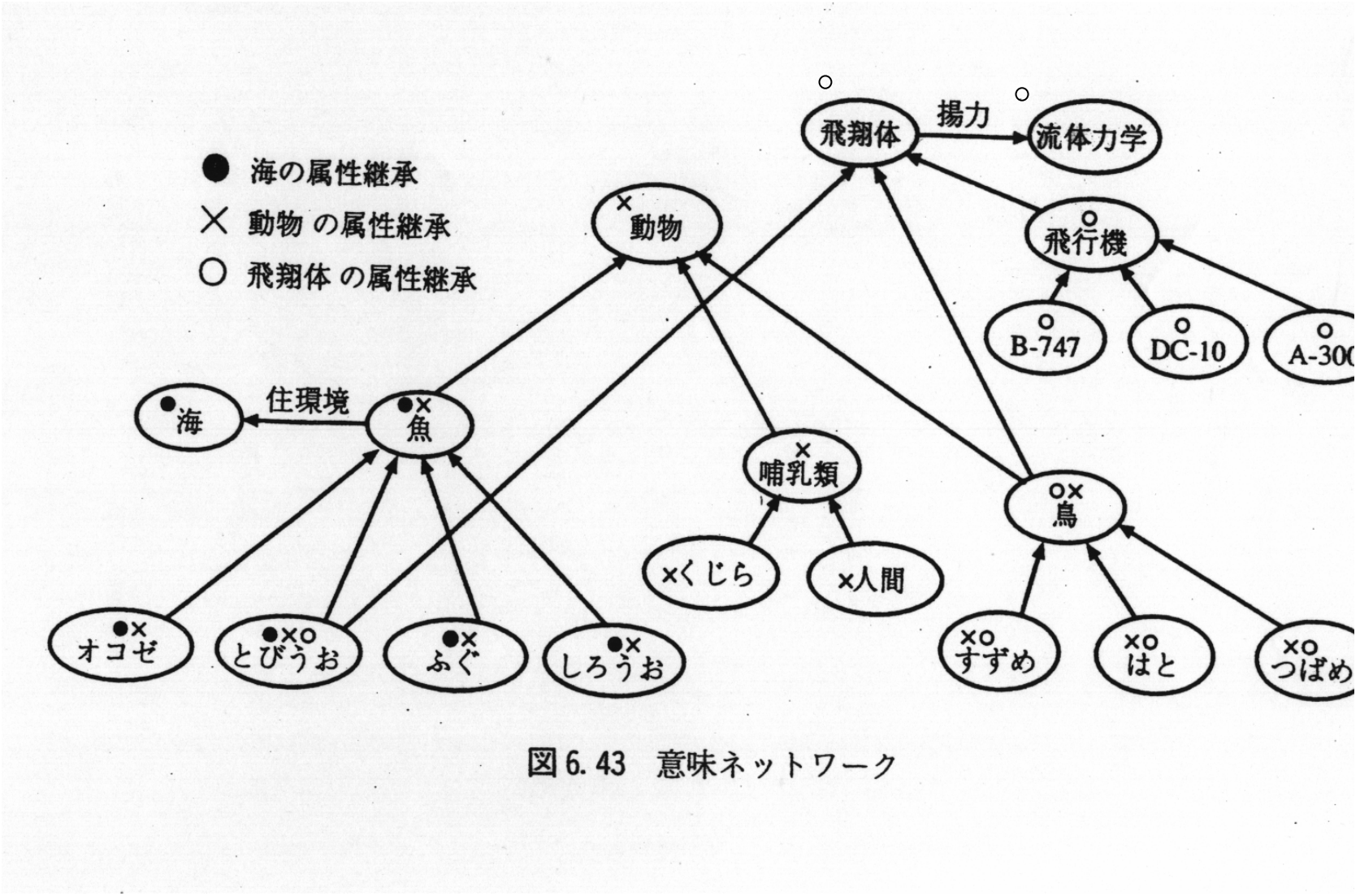


図 6.43　意味ネットワーク

マルチプロセッサ方式

電子技術総合研究所のIXM

**　マルチプロセッサと連想メモリを有機的結合**

6.6.2　プロダクションシステムの並列化

エキスパートシステム構築用の基本メカニズム

（１）OPS5の基本動作

プロダクション：条件文の集合

プロダクションの左辺（LHS）：条件要素が連接

右辺（RHS）：動作記述（追加と削除）

プロダクション：プロダクションメモリに格納

データ：ワーキングメモリ（WM）に格納

（その要素をWME）

(P make-possible-trip

(city ↑name<x> ↑state New-York)

-(weather-forcast ↑place<x> ↑date tomorrow ↑

weather rainy)

→

(make possible-trip ↑place<x> ↑date tomorrow))

P：**プロダクションの開始記号、**

make-possible-trip：**プロダクション名、**

city,weather-forecast：クラス名、↑：属性記号、
New-York,tomorrowなど：値
上記のプロダクション
［もし、NewYork州の市を表すWMEが存在し、かつ、明日その市の天気が雨である　ことを示すWMEが存在しなければ、
明日その市へ旅行することが可能であることを示す新しいWMEをWM中に追加せよ。」

WM中に、

- (city↑name Buffalo↑state New-York)

・(weather-forecast ↑place Buffalo ↑
date tomorrow ↑weather fine)

がWMEとして存在すると、上記のプロダクションを実行後

・(possible-trip ↑place Buffalo ↑date tomorrow)

がWMに追加される。

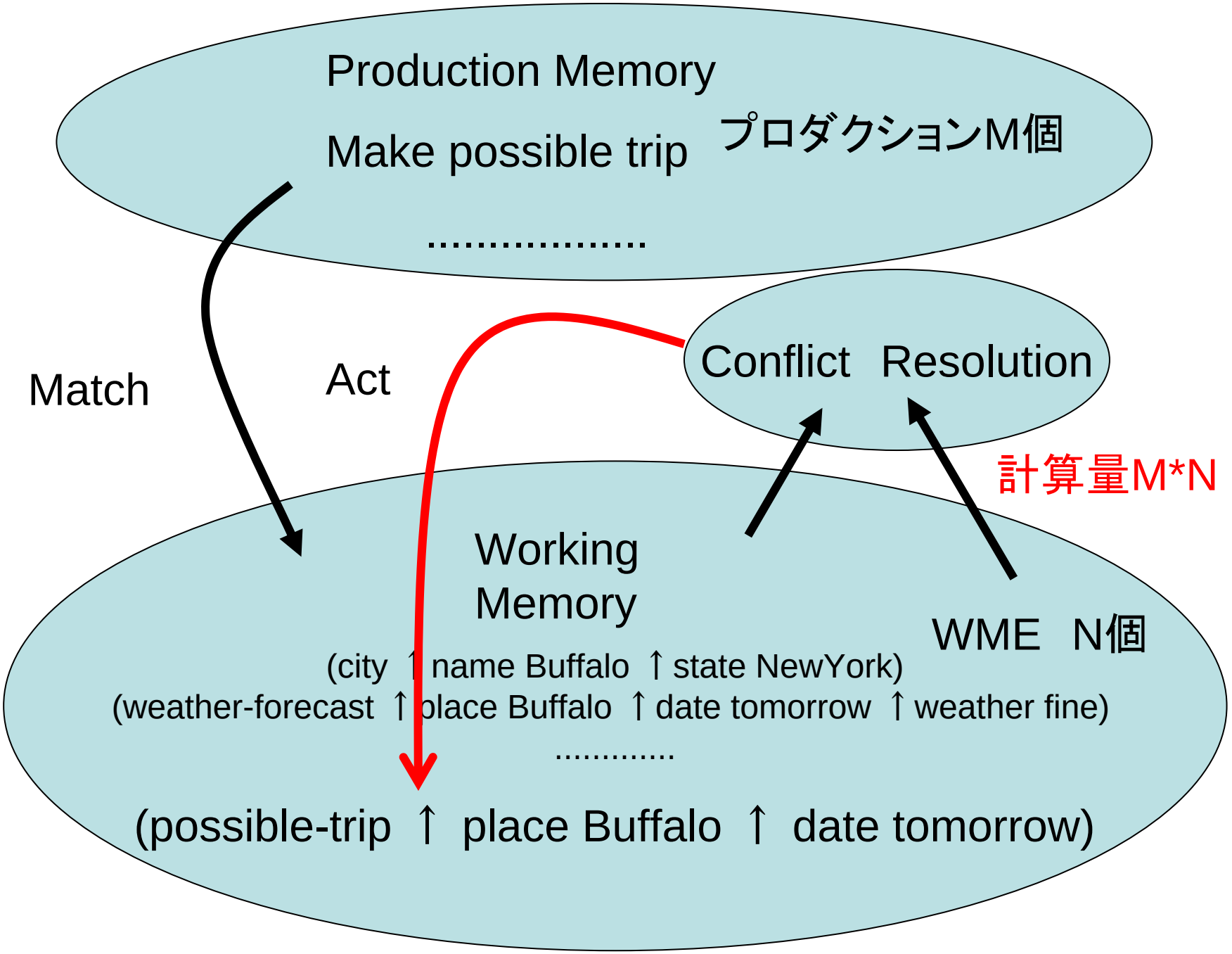
OPS5**の実行過程**

①Match:**すべてのプロダクションについて**LHS**と**WM**中のWMEの組とのマッチング**

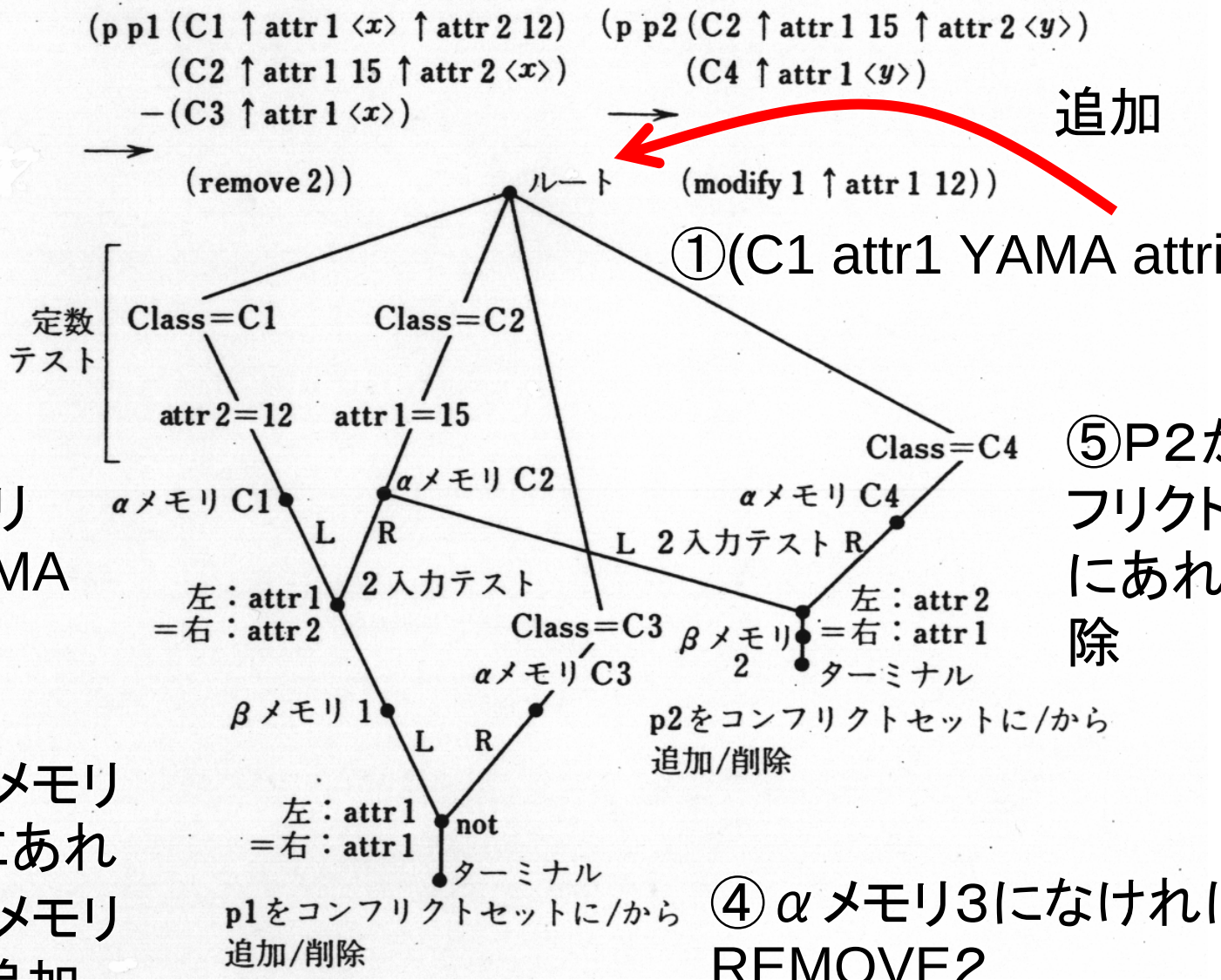
②Conflict Resolution:**マッチしたプロダクショ**

ンの中から、定められた戦略によって、プロダクションを一つ選択

③Act:選択されたプロダクションを実行し、プロダクションのRHSの実行としてWMEの追加や削除の実行


Production Memory
Make possible trip
プロダクションM個
.....
Match
Act
Conflict Resolution
計算量M*N
Working
Memory
WME N個
(city ↑ name Buffalo ↑ state NewYork)
(weather-forecast ↑ place Buffalo ↑ date tomorrow ↑ weather fine)
.....
(possible-trip ↑ place Buffalo ↑ date tomorrow)

(2)RETEアルゴリズム：ACTで変化したWMEのみ処理



6.6.3論理型言語PROLOGの並列処理

（1）PROLOGの実行過程

PROLOG：Programming in Logicの略

ホーン節（Horn Clause）の集合

$H_1 :- B_{11}, B_{12}, . . , B_{1k}.$

・・・

$H_i :- B_{i1}, B_{i2}, . . , B_{il}.$ **（５４）**

・・・

$H_n :- B_{n1}, B_{n2}, . . , B_{nm}.$

H：**頭部リテラル（head literal）、**

B：**本体（body）リテラル**

頭部リテラルのないホーン節：ゴール節；

例?-B_1, B_2.

ゴール節以外の節：定義節

本体リテラルのない節：

単位節（unit clause）、事実節（fact clause）

頭部及び本体リテラルを含む節：

非単位節、規則節（rule clause）

「:-」：含意、「,」：論理積

$H:-B_1,B_2.$

「B_1かつB_2ならばHである」、あるいは、

対偶をとって

「HでなければB_1でないかまたはB_2でない」

（２）逐次型PROLOGの実行過程

単一化(unification)

バックトラック

ゴール節から空節を導く過程

① $?\text{-}H_1.$

② $H_1\text{:-}H_2,H_3.$

③ $H_2.$

（５９）

④ $H_3.$

$?\text{-}H_1.$ → $?\text{-}H_2,H_3.$ → $?\text{-}H_3.$ → **（空節）**

PROLOGの論理的解釈：背理法の論理で実行

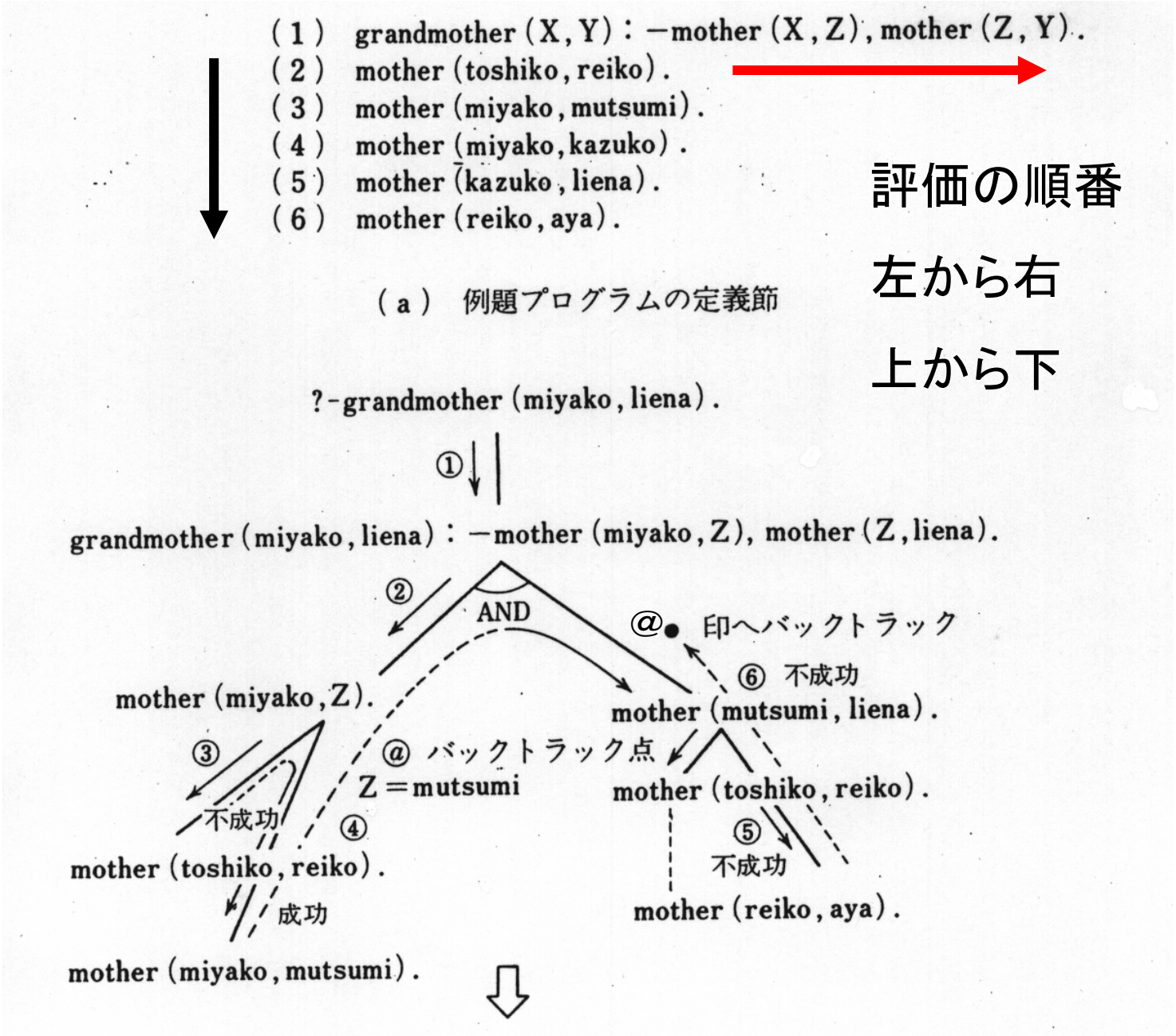
「ゴール節は成立しないと過程すると矛盾する。よってHは成立する。」

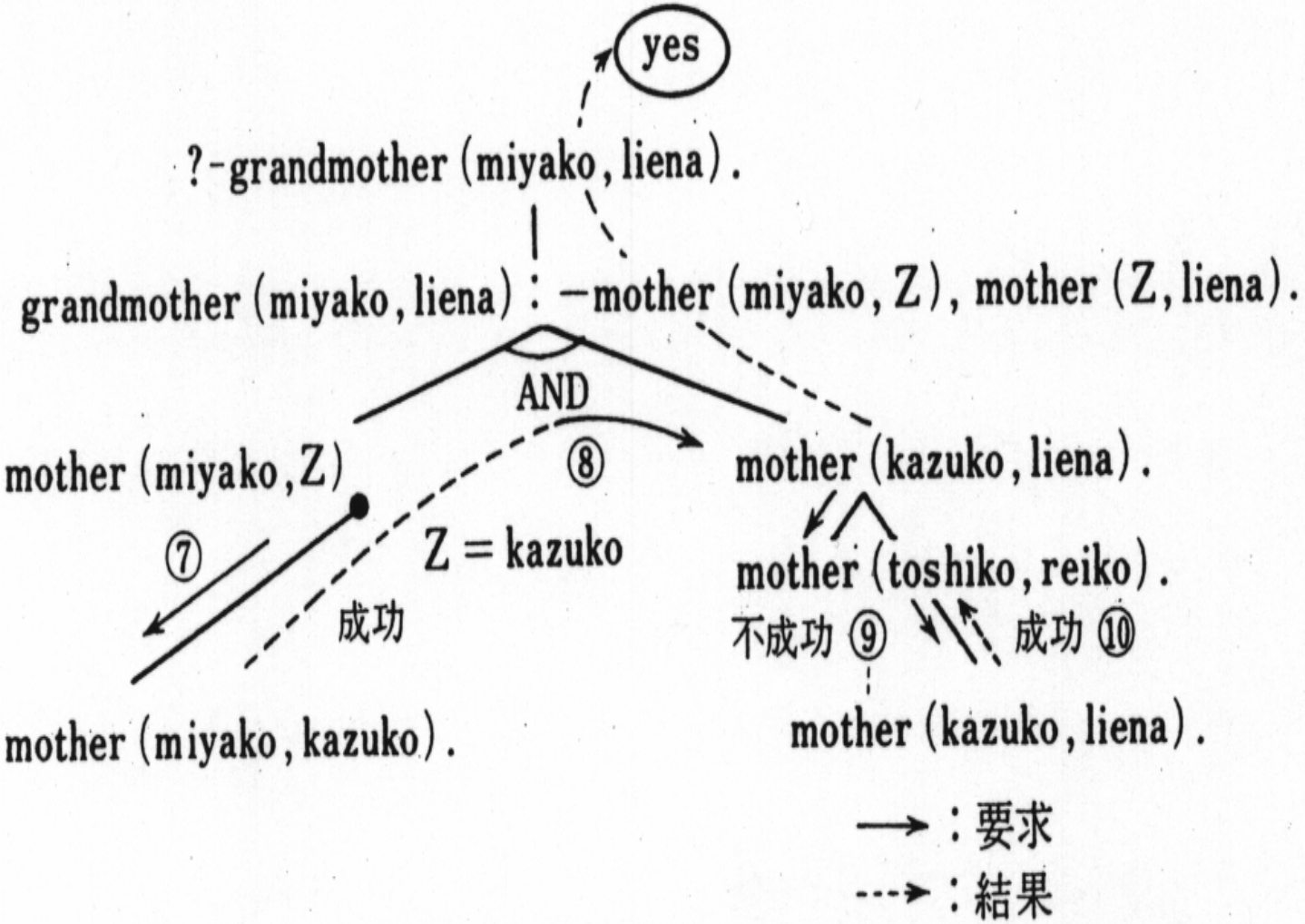
「H_1が成立しない」とすると、②によって、

「H_2が成立しないかまたはH3が成立しない」となる。

しかし、これは、③、④で矛盾する。よって、H_1は成立する。

逐次型PROLOG


(1) grandmother (X, Y) : −mother (X, Z), mother (Z, Y).
(2) mother (toshiko, reiko).
(3) mother (miyako, mutsumi).
(4) mother (miyako, kazuko).
(5) mother (kazuko, liena).
(6) mother (reiko, aya).
評価の順番
左から右
上から下
(a) 例題プログラムの定義節
?-grandmother (miyako, liena).
①
grandmother (miyako, liena) : −mother (miyako, Z), mother (Z, liena).
②
AND
@● 印へバックトラック
⑥ 不成功
mother (miyako, Z).
mother (mutsumi, liena).
③
ⓐ バックトラック点
Z = mutsumi
mother (toshiko, reiko).
不成功
④
⑤
不成功
mother (toshiko, reiko).
成功
mother (reiko, aya).
mother (miyako, mutsumi).


yes
?-grandmother (miyako, liena).
grandmother (miyako, liena) : −mother (miyako, Z), mother (Z, liena).
AND
mother (miyako, Z)
⑧
mother (kazuko, liena).
⑦
Z = kazuko
mother (toshiko, reiko).
成功
不成功 ⑨
成功 ⑩
mother (miyako, kazuko).
mother (kazuko, liena).
→ : 要求
⇢ : 結果

（b） 実行過程

単一化操作、環境の保存、引き数の受渡しとバックトラックなどの高速化

ICOTのPSI

インタプリタ方式

コンパイラ方式

PROLOG向き機械命令セット

（WAM: Warren Abstract Machine）

（3）PROLOGの実行過程における並列処理

？－H_1,H_2,…,H_m
H_1:-B_{11},B_{12}
H_1:-B_{21}
H_1:-B_{31},B_{32},B_{33}

①AND並列
ゴール節：?-H_1,H_2,..,H_m.

$H_1, H_2, ..., H_m$**の単一化処理を同時に起動**

解の整合性チェック(consistensy check)**が必要**

②ストリーム並列

?-$H_1, H_2, ..., H_m$.

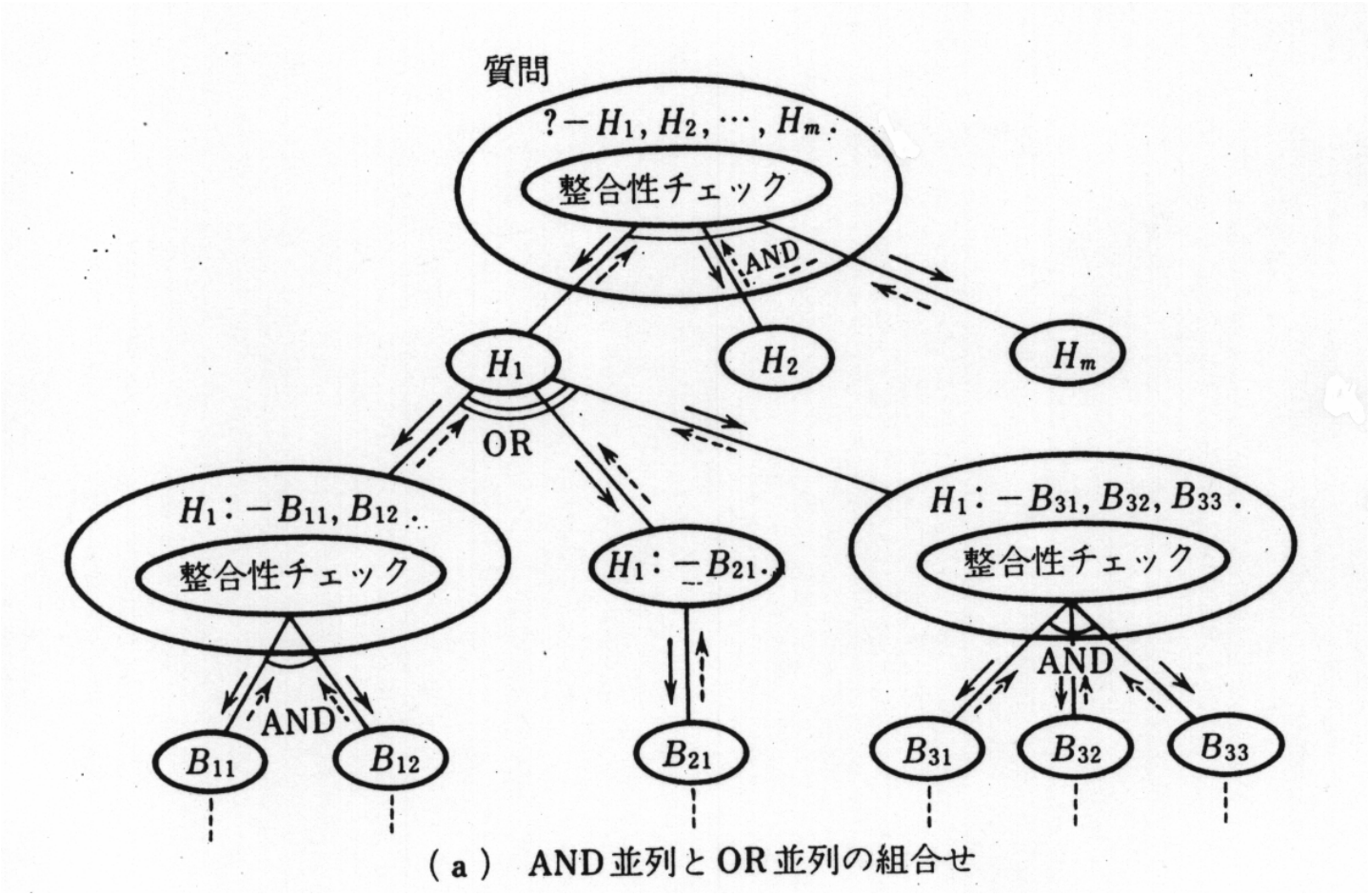
ゴールのパイプライン的実行

③OR**並列**

H_1:-B_{11}, B_{12}.

H_1:-B_{21}.

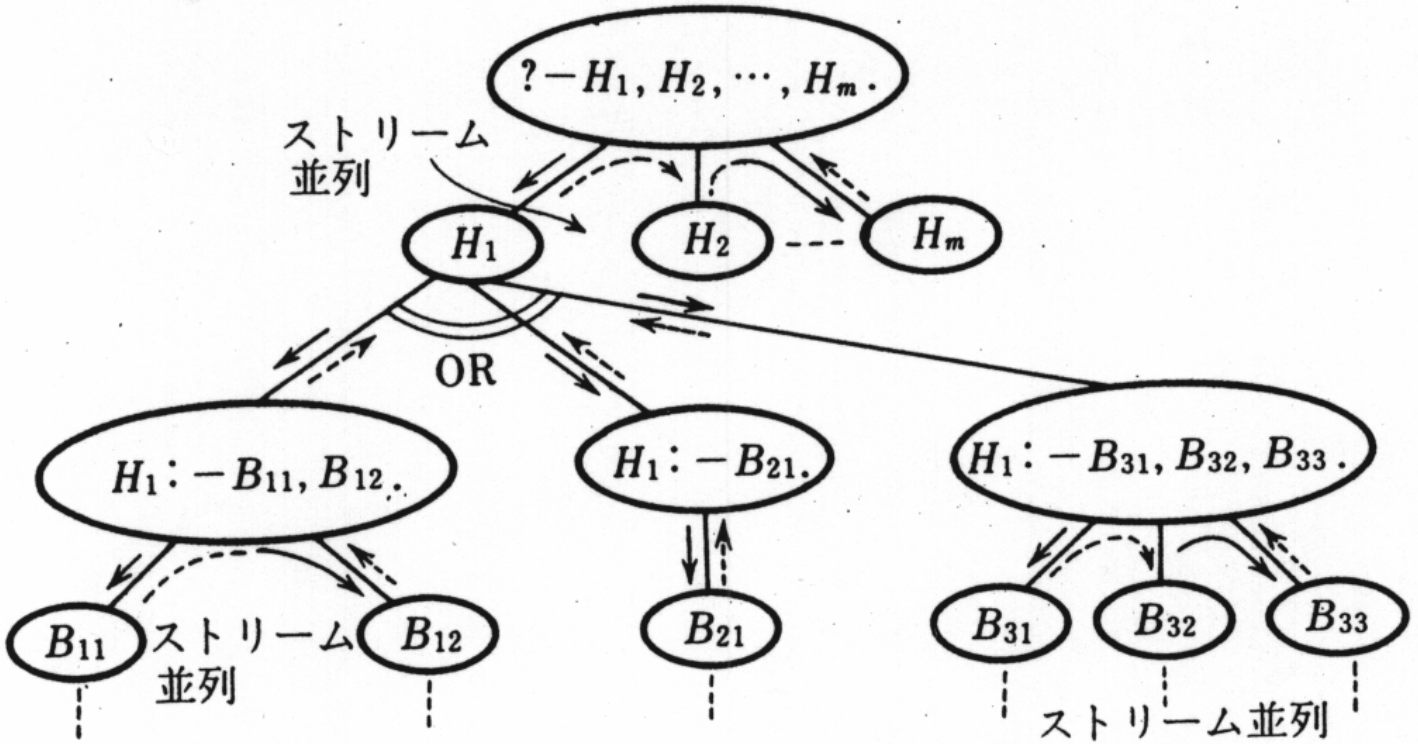
H_1:-B_{31}, B_{32}, B_{33}.



（a） AND並列とOR並列の組合せ

AND並列とOR並列の組み合わせ

ストリーム並列とOR並列の組み合わせ



?-H1•

単一化可能なものは３つ存在

④低レベルでの並列処理

単一化操作自体：

基本的には記号列のパターンマッチング

（サーチ並列）

引数多数の場合：各引き数ごとに単一化処理を並列実行

（引き数間並列）

環境の受渡しや構造体の処理

（４）PROLOG並列コンピュータ

・プロセスの適切な機能レベルの設定

・プロセスの生成、管理の高速化

・効率のよい負荷分散と通信量削減

・相互結合網自体の高速化

・ガーベッジコレクションの高速化[323]

（５）第五世代コンピュータ

1982より11年の歳月をかけた大プロジェクト

大規模システム：

PIM/p（富士通、512PE）、

PIM/m（三菱、256PE）、

PIM/c（日立、256PE）

小規模システム

PIM/k（東芝、16PE）、

PIM/i（沖、16PE）

並列論理型言語KL-1の設計とその処理系の開発

ガード付きホーン節

$H:-G_1,G_2,...,G_n|B_1,B_2,...,B_m.$

ゴールHとの単一化の際：そのガード部が満たされた節が１つのみ選択

KL-1処理系の特徴

①分散メモリ環境での効率のよい単一化
（ユニフィケーション）の実現

②マルチプロセッサ上での新しいガーベッジ
コレクション方式
（輸出入表やMRB（Multiple Reference Bit）
方式などの導入）の提案

③ユーザ指定（プラグマの指定）による負荷分
散方式の導入

④荘園とよぶメタ管理機能の導入

⑤遅延通信、一括通信など通信オーバヘッド削減手法の提案

6.6.4ニューロコンピュータ

（1）歴史

①新しい計算モデルの必要性

②様々なニューロモデルの提唱と
有効性の実証

ニューロコンピュータの利用目的

①学習機械

②最適化機械

ユーザ問題：

**　エネルギー関数の最小化の形に定式化**

システム：

**　エネルギー関数を最小化するように動作**

**　平衡点が一つの解**

ニューロンモデル

$$f(x) = \frac{1}{1 + exp(-x + \theta)}$$

$$f'(x) = f(x)(1 - f(x))$$

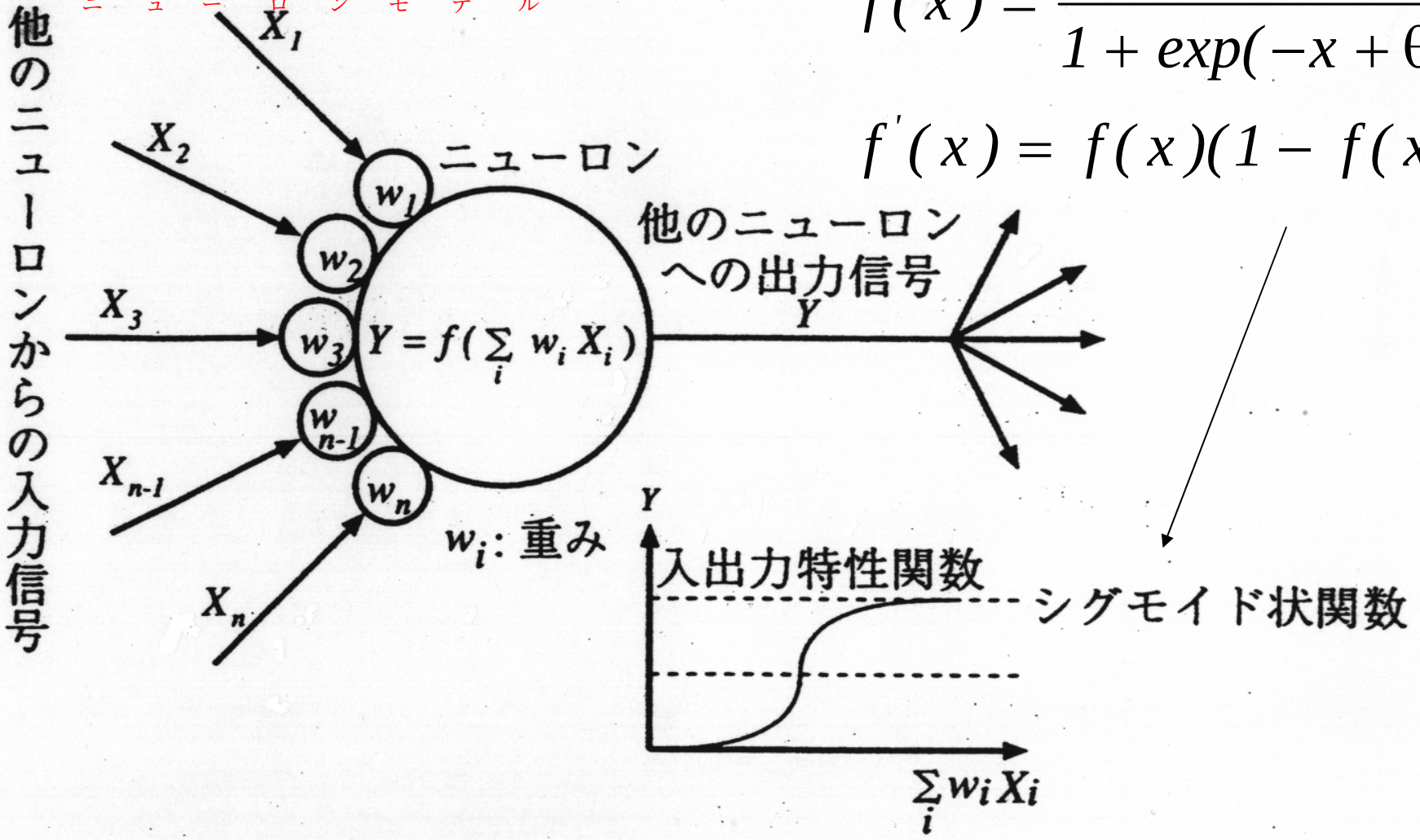


図 6.47　ニューロンモデル

（２）ニューロコンピュータ

(a)階層型モデル

Rumelhartのバックプロパゲーションアルゴリズム

(b)相互結合型モデル

ホップフィールドモデル

。

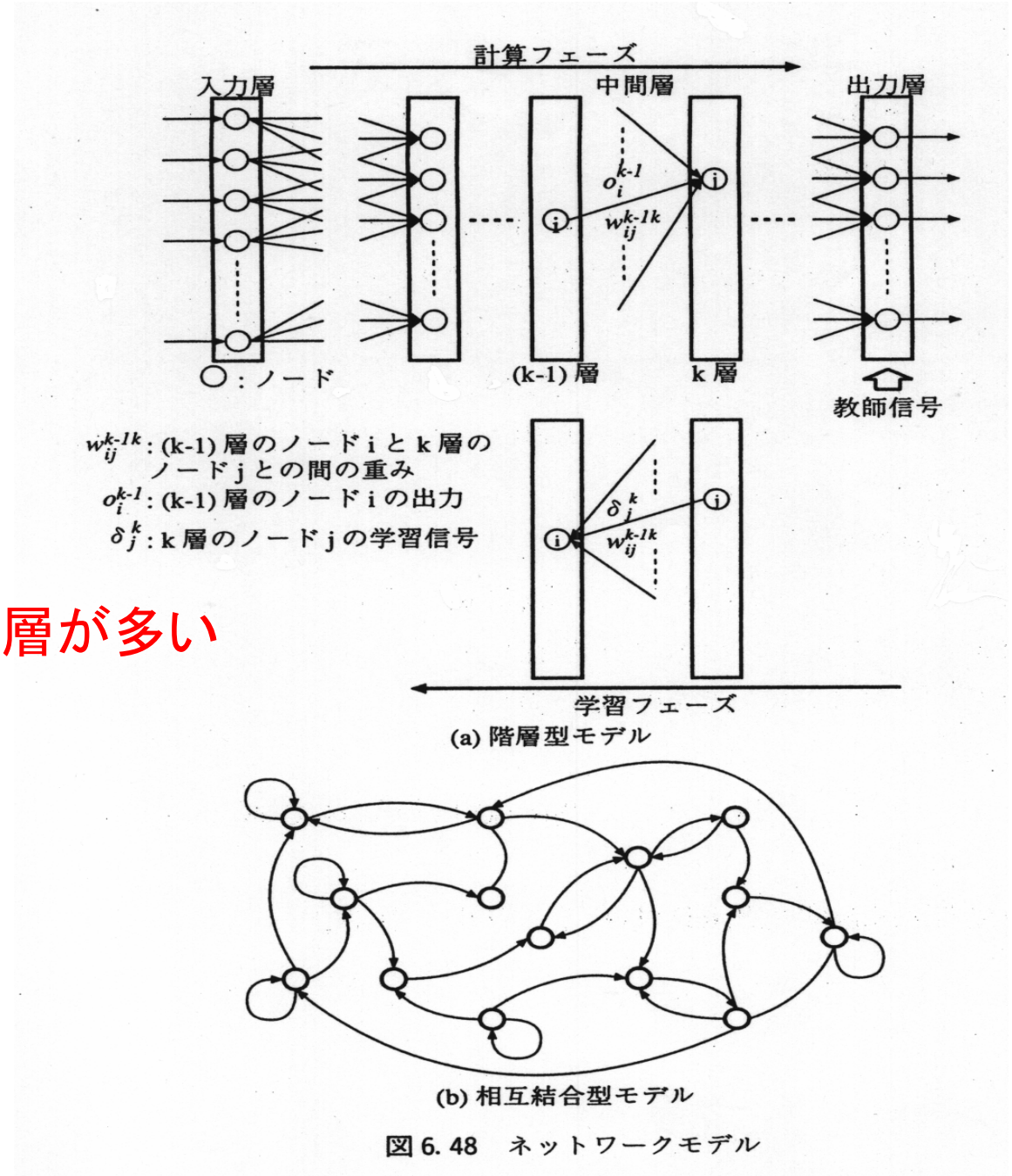


図 6. 48　ネットワークモデル

3層が多い

バックプロパゲーション

① 入力層に、入力信号を入力する

② 入力されたデータは、中間層（隠れ層）を経て、出力層へと伝わる。

③ 出力層Mのノードiから得られる出力をo^M_iとする。

①で入力した信号に対応する教師信号をd_iとすると、出力層の学習信号δ^M_iは次式で求められる。

$$\delta^M_i=(d_i-o^M_i)f'(i^M_i) \qquad (62)$$

ただし、$f'(x)$は特性関数の微分値である。

④ 第$k-1$層の学習信号は、k層の学習信号を使って次式で計算される

最急降下法

$$\delta^{k-1}_i=f'(i^{k-1}_i)\Sigma_j\delta^k_j W^{k-1}_{ij} \qquad (63)$$

$o^{k-1}{}_i$はk-1層のノードiにおける出力値

$\delta^k{}_j$はK層のノードjにおける学習値

⑤ ノード間のリンクの重みを次式に従って更新する。

$$\Delta w^{k-1}{}_i{}^k{}_j(n+1)=\eta\delta^k{}_j o^{k-1}{}_i+\alpha\Delta w^{k-1}{}_i{}^k{}_j(n)$$

（６４）

ただし、nは学習のステップ

ηは学習係数

αは安定化定数

③以上の①から⑤までのステップを、種々の入力信号とそれに対応する教師信号のセットに対し、繰り返す。

(b)**相互結合型モデル**

ネットワークの平衡状態

ホップフィールドモデル

各ノードの動作（状態変化規則）は次式で与えられる。

$\Sigma_j W_{ij} u_j - \theta_i > 0$　**ならば**　$u_i = 1$

$\Sigma_j W_{ij} u_j - \theta_i < 0$　**ならば**　$u_i = 0$　　**（６５）**

$\Sigma_j W_{ij} u_j - \theta_i = 0$　**ならば**　u_i **は変化しない**

ただし、u_i **はノード**i**の状態**

θ_i はノード i のしきい値

エネルギ関数

$$E=-1/2\Sigma_i\Sigma_j W_{ij}u_iu_j+\Sigma_i\theta_iu_i\ (i\neq j) \qquad (66)$$

研究課題

① ローカルミニマム問題：

② 評価関数の設定：

巡回セールスマン問題（TSP）の近似解

$E=1/2[\Sigma_i\Sigma_j\Sigma_k d_{ij}u_{ik}(u_{jk+1}+u_{jk-1})$ **：都市間の距離**

$+A(\Sigma_i(\Sigma_k u_{ik}-1)^2$ **：②の制約条件**

$+\Sigma_k(\Sigma_i u_{ik}-1)^2)]$ $(i \neq j)$ **：①の制約条件**

①：同時刻には一つの都市しか通らず

②：一度通った都市は二度と通らず

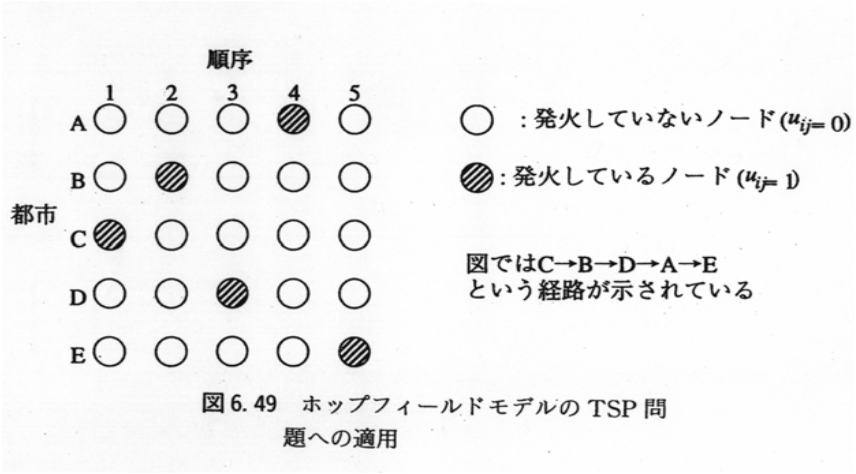


図 6.49　ホップフィールドモデルの TSP 問題への適用

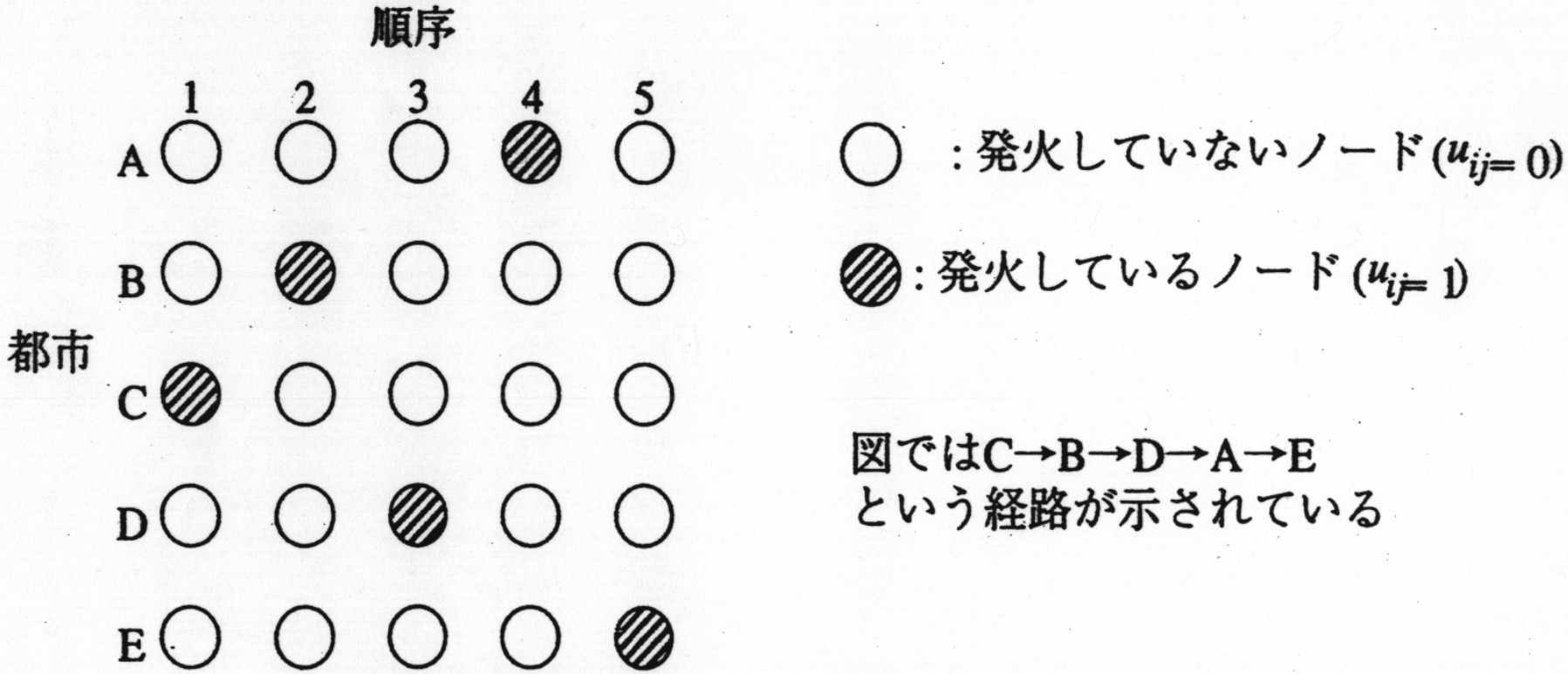


図 6.49　ホップフィールドモデルの TSP 問題への適用

$$E=1/2[\Sigma_i\Sigma_j\Sigma_k d_{ij}u_{ik}(u_{jk+1}+u_{jk-1})$$
$$+A(\Sigma_i(\Sigma_k u_{ik}-1)^2$$
$$+\Sigma_k(\Sigma_i u_{ik}-1)^2)] \quad (i\neq j)$$

$$E=-1/2\Sigma_i\Sigma_k\Sigma_j\Sigma_l W_{ikjl}u_{ik}u_{jl}+\Sigma_i\Sigma_k\theta_{ik}u_{ik}$$
$$(i\neq j \text{ and } k\neq l)$$

ただし、u_{ik}はi行k列にあるノードの値

d_{ij}は都市iとjの距離

Aは距離と制約条件との重要度を決めるパラメータ

このとき、ノード$<i,k>$、$<j,l>$間の重みW_{ikjl}、

および、ノード$<i,k>$のしきい値θ_{ik}を

$$W_{ikjl}=-d_{ij}(\delta_{lk+1}+\delta_{lk-1})$$
$$-A(\delta_{ij}(1-\delta_{kl})+\delta_{kl}(1-\delta_{ij})) \quad (68)$$

$$\theta_{ik}=-A$$

$u_{ij}^2+(1-u_{ij})^2=1$

$$\sum_i \left(\sum_k u_{ik} - 1 \right)^2 =$$

$$\sum_i \left[\sum_k u_{ik}^{\ 2} + \sum_{k,j,l} \delta_{ij}(1-\delta_{kl})u_{ik}u_{jl} - 2\sum_k u_{ik} + 1 \right]$$

$$= \sum_{i,k,j,l} \delta_{ij}(1-\delta_{kl})u_{ik}u_{jl} - \sum_{i,k} u_{ik} + n$$

ただし、$\delta = 1 \quad (i=j)$

$0 \quad (i \neq j)$

で与えることにより、ネットワークのエネルギー関数E**は**

$$E=-1/2\Sigma_i\Sigma_k\Sigma_j\Sigma_l W_{ikjl}u_{ik}u_{jl}+\Sigma_i\Sigma_k\theta_{ik}u_{ik} \qquad (69)$$

$$(i \neq j \ \text{and} \ k \neq l)$$

（３）並列処理

ニューロコンピュータの実現機構

① ノード間の結線、

② 各ノードの総入力を計算するための積和演算機構、

③各ノードの特性関数のための非線型演算機構

(a)**アナログ方式**

(b)**光方式**

①相互結線

②積和演算

光変調（SLM:Spatial Light Modulator**）機能と受光素子機能を合わせ持った感度可変型受光素子（**VSPD:Variable Sensitivity Photodector**）**

③非線型演算機構

(c)**ベクトルプロセッサの利用**

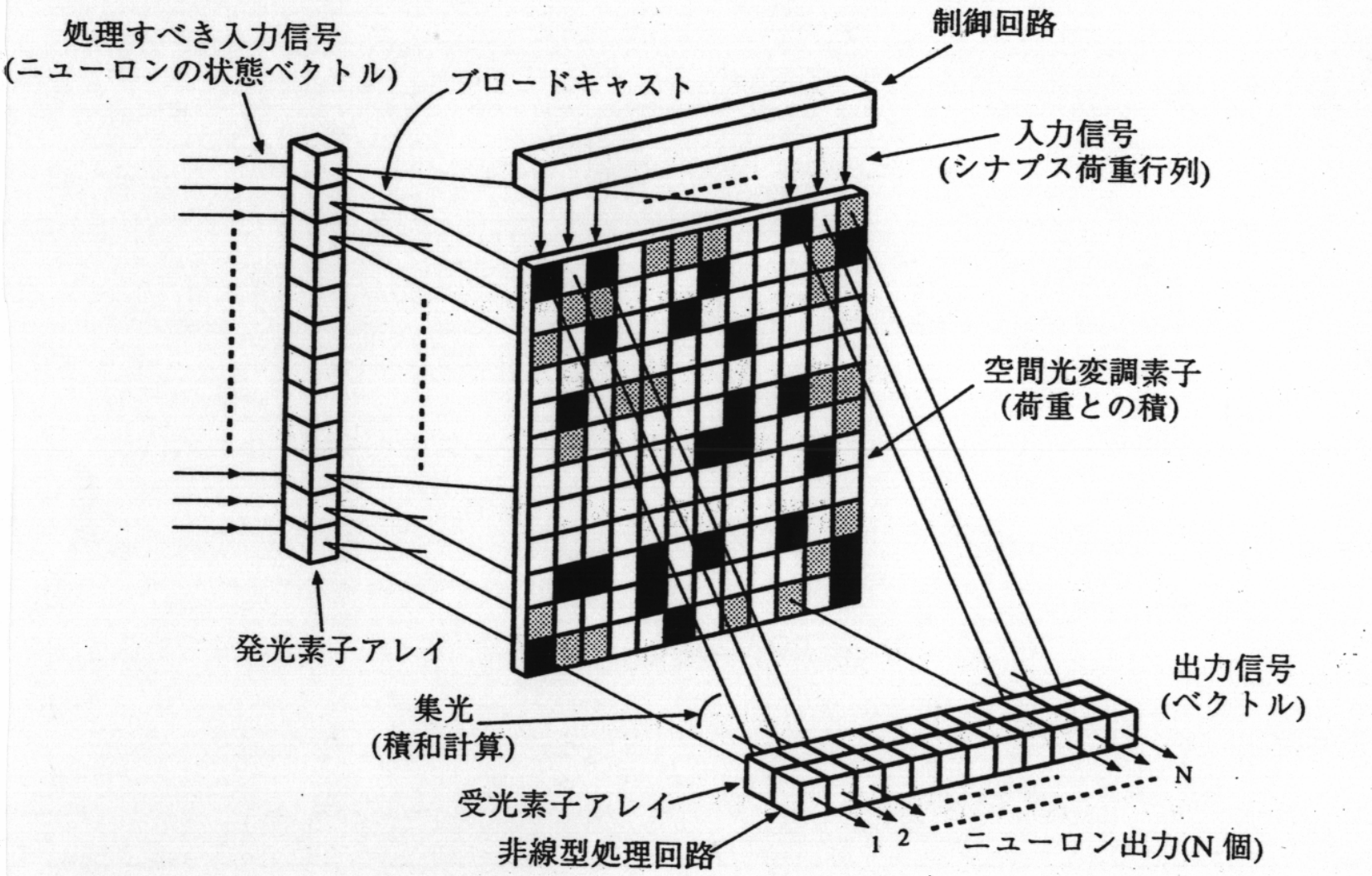


図 6. 50　光方式によるニューラルネットワーク（太田，新田，久間：「光電子共存型ニューラルネットワーク」，情報処理，Vol. 35, No. 6, pp. 543-545 (1994)）

(d)ニューロチップ

(e)マルチプロセッサ

マルチプロセッサの利用

・テストパターン並列

m個のテストパターンをPp台の大プロセッサに均等に分割

一定回数のテストパターンを実行の後、各大プロセッサは重みの交換

重み交換の周期：テストパターンによる収束速度に大きな影響

・層内並列処理

１台の大プロセッサで層をPs台の小プロセッサに分割し、並列処理

各層間：同一の小プロセッサで分担

・層間パイプライン

各層にＰs台の小プロセッサを割り付け、各層間でパイプライン処理

①層内並列処理のみ（NP）

K層をN台の小プロセッサで並列に処理

１つのテストパターンごとに重みの変更

②テストパターン並列のみ（TSP）

N組にテストパーンを分割

一定周期で各プロセッサの重みを平均

③テストパターン並列と層内並列の組み合せ（2APC）

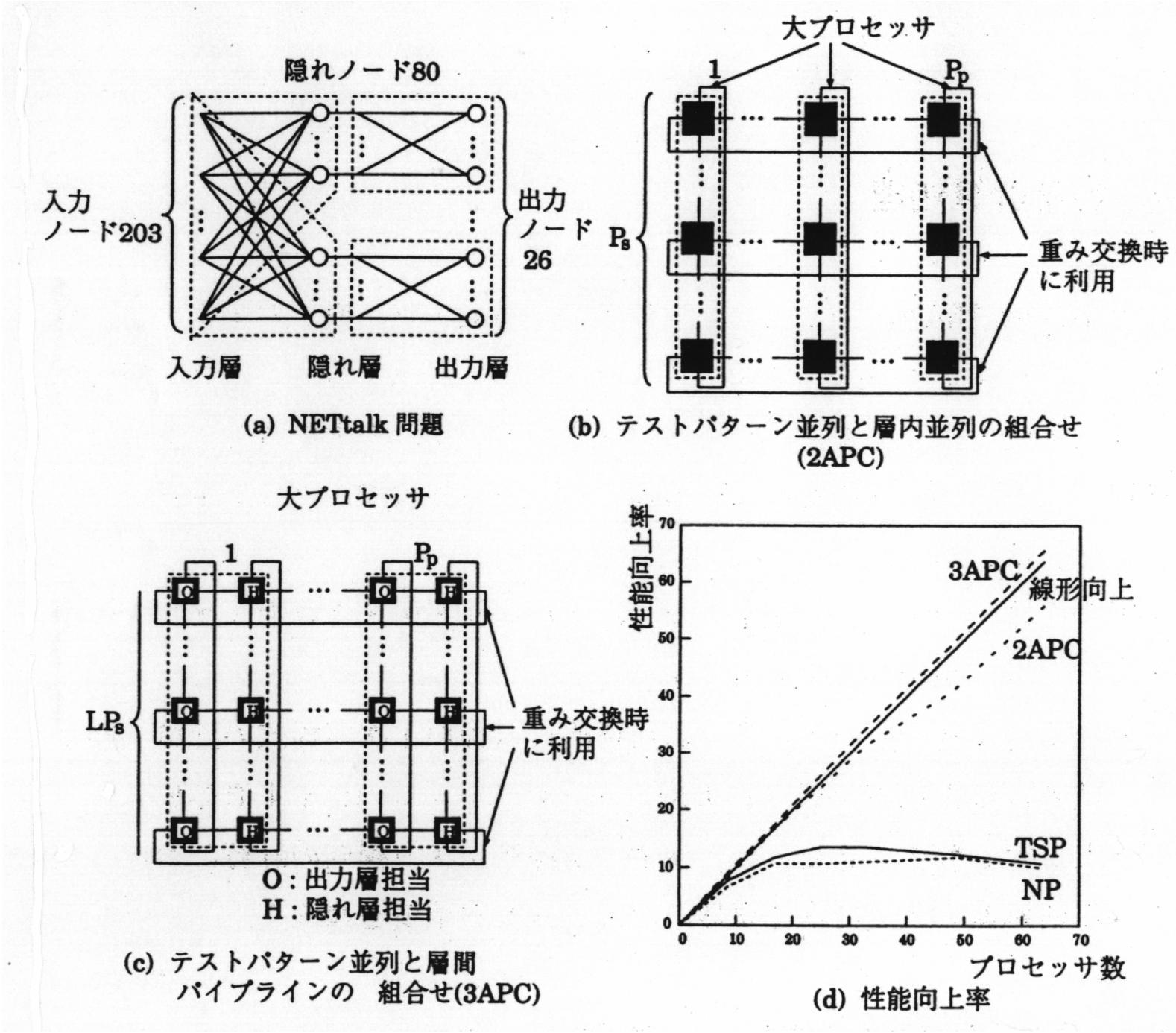
$N=P_p x P_s$**であり、最適な**P_p**、**P_s**の組み合せが期待できる。**

④テストパターン並列、層間パイプラインの

組合せ（3APC）

$N=P_p x P_s x L$**であり、最適な**P_p**、**P_s**の組み合せが**

期待できる。


隠れノード80
入力
ノード203
出力
ノード
26
入力層
隠れ層
出力層
(a) NETtalk 問題
大プロセッサ
1
Pp
Ps
重み交換時
に利用
(b) テストパターン並列と層内並列の組合せ
(2APC)
大プロセッサ
1
Pp
LPs
O
H
重み交換時
に利用
O : 出力層担当
H : 隠れ層担当
(c) テストパターン並列と層間
パイプラインの 組合せ(3APC)
性能向上率
3APC
線形向上
2APC
TSP
NP
0
10
20
30
40
50
60
70
プロセッサ数
(d) 性能向上率

６．７数値計算処理

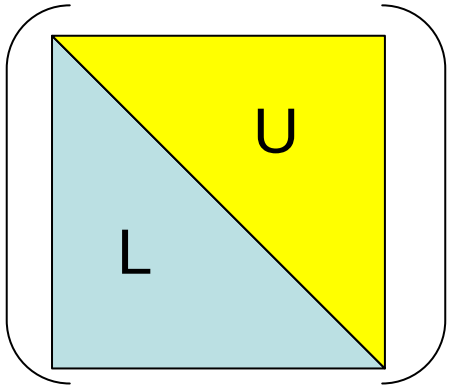
６.７.１基本的なベクトル演算と行列計算

A:行列　a_{ij}

下三角行列　L

上三角行列　U

転置行列　A^T　a_{ji}



単位行列　I

逆行列　A^{-1}:　AA^{-1}=I、A:正則行列

列ベクトル　a

行ベクトル　a^T

直交ベクトル： $a_i^T a_j = o$

ベクトル和：$z = ax + y$

行列とベクトル積： $z = Ax$

行列積： $C = AB$

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

I,J,Kの３重ループ構造

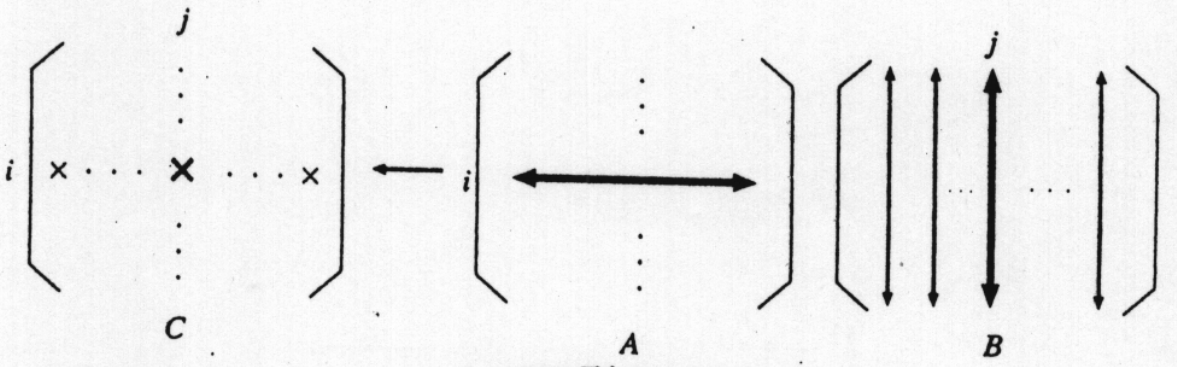
IJK,JIK,KIJ,KJI,IKJ,JKIの６通り

ば，IJK では，

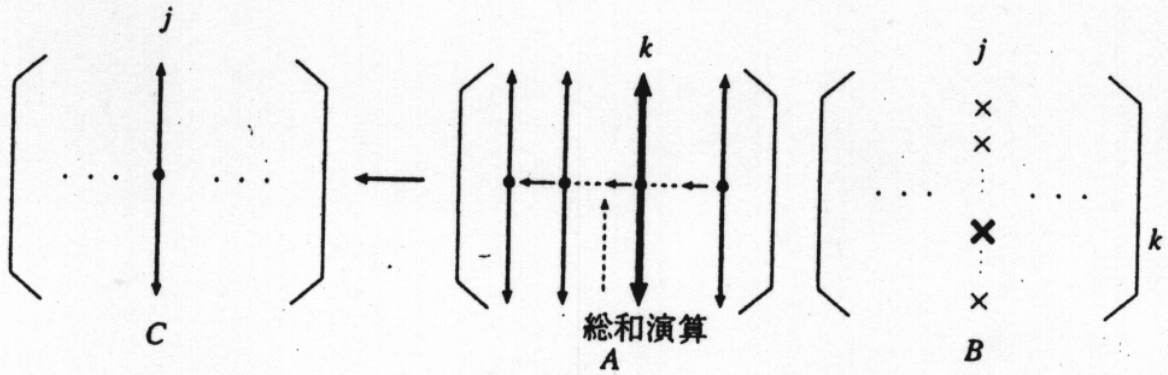
```
      DO   10   I=1, N
        DO   20   J=1, N
        C(I, J)=0. 0
          DO   30   K=1, N                                    (6. 74)
          C(I, J)=C(I, J)+A(I, K)*B(K, J)
        30   CONTINUE
      20   CONTINUE
    10   CONTINUE
```

また，JKI では

```
      DO   10   J=1, N
        DO   20   I=1, N
        C(I, J)=0, 0
      20   CONTINUE                                           (6. 75)
        DO   30   K=1, N
          DO   40   I=1, N
          C(I, J)=C(I, J)+A(I, K)*B(K, J)
        40   CONTINUE
      30   CONTINUE
    10   CONTINUE
```



(a) IJK型の $C = AB$



(b) JKI型の $C = AB$

⟷ : ベクトル
× : スカラ
太線 : 最内ループ

図 6.52　$C=AB$ 計算におけるメモリアクセスパターン

６．７．２１次方程式の直接解法

$Ax=b$

（１） ガウス消去法

前進消去

計算量：n**が大きいとき、**$n^3/3$**の乗算、**$n^3/3$**の加算**

後退代入：加算、乗算$n^2/2$

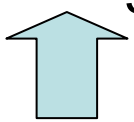
$$X_1+X_2+X_3=6$$
$$2X_1+3X_2+4X_3=20$$
$$X_1+4X_2-2X_3=3$$

$$X_1+X_2+X_3=6$$
$$X_2+2X_3=8$$
$$3X_2-3X_3=-3$$

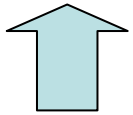
$$X_1+X_2+X_3=6$$
$$X_2+2X_3=8$$
$$9X_3=27$$

前進消去

$$X_1+2+3=6$$
$$X_2=2$$
$$X_3=3$$



$$X_1+X_2+3=6$$
$$X_2+6=8$$
$$X_3=3$$



$$X_1+X_2+X_3=6$$
$$X_2+2X_3=8$$
$$X_3=3$$

後退代入

$$
\begin{aligned}
a^1_{1\,1}x_1 &+ a^1_{1\,2}x_2 &&+\cdots+ a^1_{1\,n}x_n &&= b^1_1\\
a^1_{2\,1}x_1 &+ a^1_{2\,2}x_2 &&+\cdots+ a^1_{2\,n}x_n &&= b^1_2\\
&&&\cdots\\
a^1_{n-1\,1}x_1 &+ a^1_{n-1\,2}x_2 &&+\cdots+ a^1_{n-1\,1}x_n &&= b^1_{n-1}\\
a^1_{n\,1}x_1 &+ a^1_{n\,2}x_2 &&+\cdots+ a^1_{n\,n}x_n &&= b^1_n
\end{aligned}
$$

$$
\begin{aligned}
a^1_{1\,1}x_1+\ a^1_{1\,2}x_2\ \ &+\cdots+a^1_{1\,n}x_n\ \ &=b^1_1\\
a^2_{2\,2}x_2\ \ &+\cdots+a^2_{2\,n}x_n\ \ &=b^2_2\\
&\vdots\\
a^2_{n-1\,2}x_2&+\cdots+a^2_{n-1\,n}x_n\ &=b^2_{n-1}\\
a^2_{n\,2}x_2\ \ &+\cdots+a^2_{n\,n}x_n\ \ &=b^2_n
\end{aligned}
$$

上式で $i,j\geqq 2$ に対して

$$a^2_{i\,j}=a^1_{i\,j}-a^1_{i\,1}a^1_{1\,j}/a^1_{1\,1}$$

$$b^2_i=b^1_i-a^1_{i\,1}b^1_1/a^1_{1\,1}$$

である．一般に分解$(k-1)$

$$
\begin{aligned}
a^1_{1\,1}x_{\,1}+a^1_{1\,2}x_{\,2}+\cdots \qquad & +a^1_{1\,n}x_{\,n} \quad =b^1_1\\
a^2_{2\,2}x_{\,2}+\cdots \qquad & +a^2_{2\,n}x_{\,n} \quad =b^2_2\\
\vdots & \\
a^{k-1}_{k-1\,k-1}x_{\,k-1}+\cdots & +a^{k-1}_{k-1\,n}x_{\,n} \quad =b^{k-1}_{k-1}\\
a^{k-1}_{k\,k-1}x_{\,k-1}+\cdots \quad & +a^{k-1}_{k\,n}x_{\,n} \quad =b^{k-1}_k\\
\vdots & \\
a^{k-1}_{n\,k-1}x_{\,k-1}+\cdots \quad & +a^{k-1}_{n\,n}x_{\,n} \quad =b^{k-1}_n
\end{aligned}
$$

を用いて，分解 k $(i, j \geqq k)$

$$a^k_{i\,j}=a^{k-1}_{ij}-a^{k-1}_{i\,k-1}\,a^{k-1}_{k-1\,j}/a^{k-1}_{k-1\,k-1}$$

$$b^k_i=b^{k-1}_i-a^{k-1}_{i\,k-1}\,b^{k-1}_{k-1}/a^{k-1}_{k-1\,k-1}$$

を得る．これを $k=n$ まで繰り返すと，

を得る．これを $k=n$ まで繰り返すと，

$$
\begin{aligned}
a^1_{1\,1}x_1+a^1_{1\,2}x_2+\cdots &\qquad +a^1_{1\,n}x_n &&= b^1_1\\
a^2_{2\,2}x_2+\cdots &\qquad +a^2_{2\,n}x_n &&= b^2_2\\
&\ \ \vdots\\
a^{n-1}_{n-1\,n-1}x_{n-1}+a^{n-1}_{n\ \ n}x_n &&&= b^{n-1}_n\\
a^n_{n\,n}x_n &&&= b^n_n
\end{aligned}
$$

$$\sum_{l=1}^{n-1} l^2 = n(n-1)(2n-1)／6$$

（２）　ＬＵ分解

A=LU

Ax=b-->Ly=b、Ux=y

①外積形式ガウス法

②内積形式ガウス法

③クラウト法

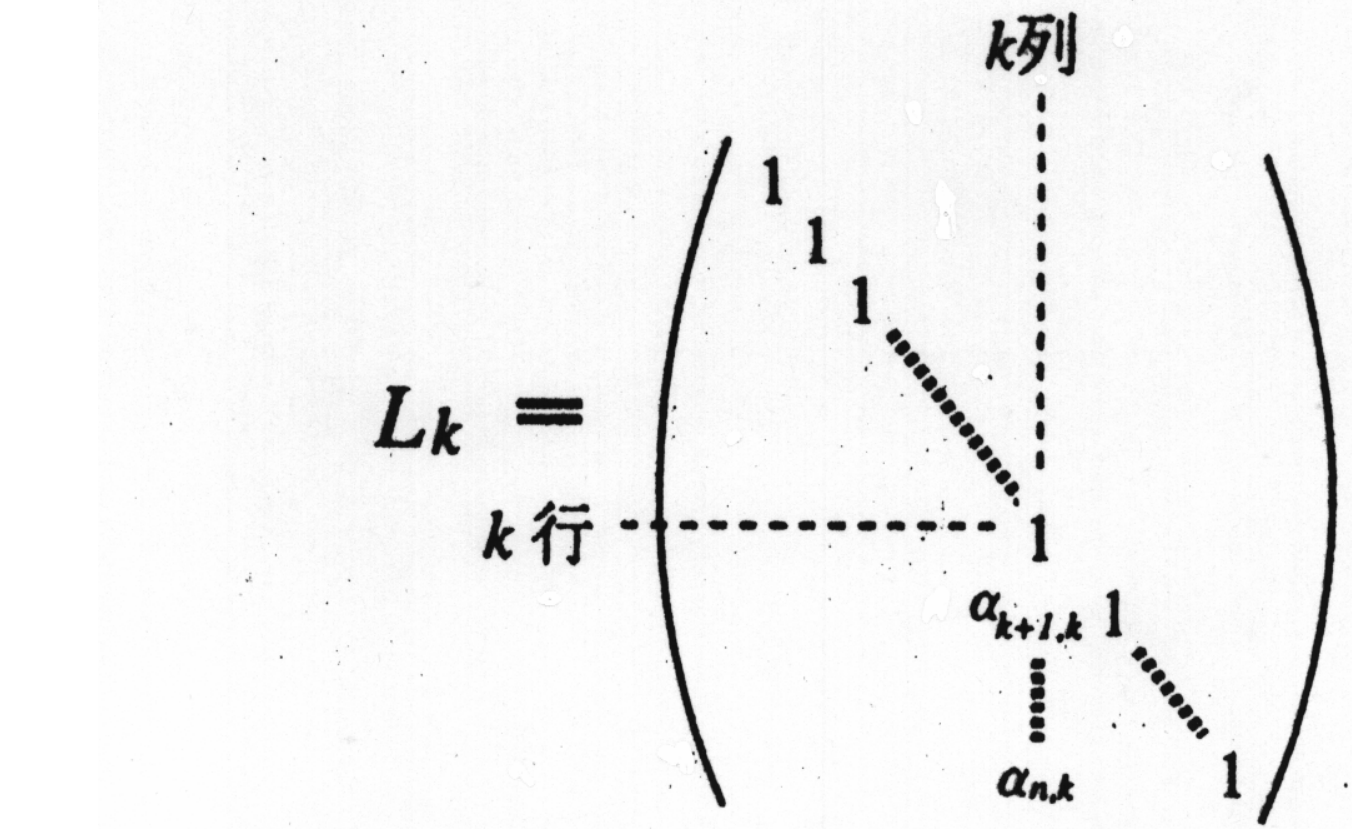
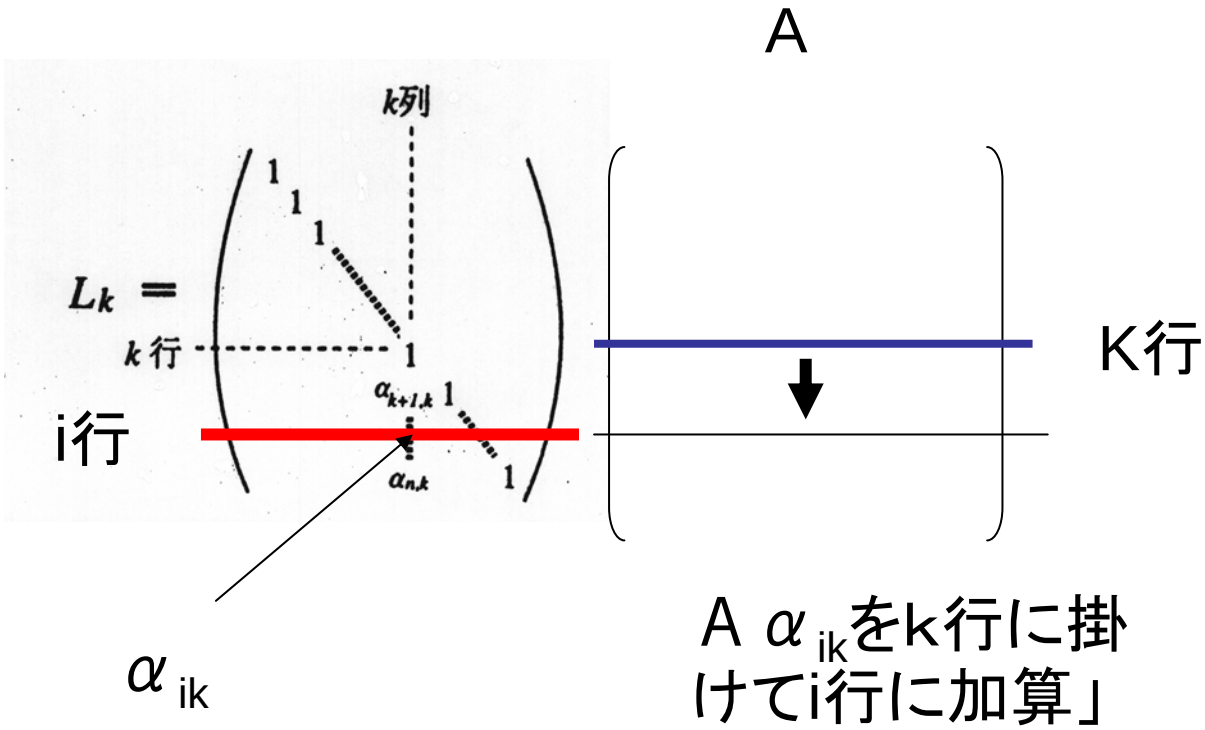


図 6.53　k 行要素の他行への加算操作


A
k列
L_k =
k行
i行
K行
α_{ik}
A α_{ik}をk行に掛けてi行に加算」

$$L_{n-1}L_{n-2}\cdots L_2L_1Ax=Ux \tag{6.84}$$

$$L_{n-1}L_{n-2}\cdots L_2L_1=L_{n-1}+L_{n-2}+\cdots+L_2+L_1-(n-2)E=L^{-1}$$

となる．また，

$$\begin{aligned}L&=(L_{n-1}L_{n-2}\cdots L_1)^{-1}=L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}\\&=L_1^{-1}+L_2^{-1}+\cdots+L_{n-1}^{-1}-(n-2)E\end{aligned} \tag{6.85}$$

L_i^{-1}：L_iの非対角要素の符号反転したもの

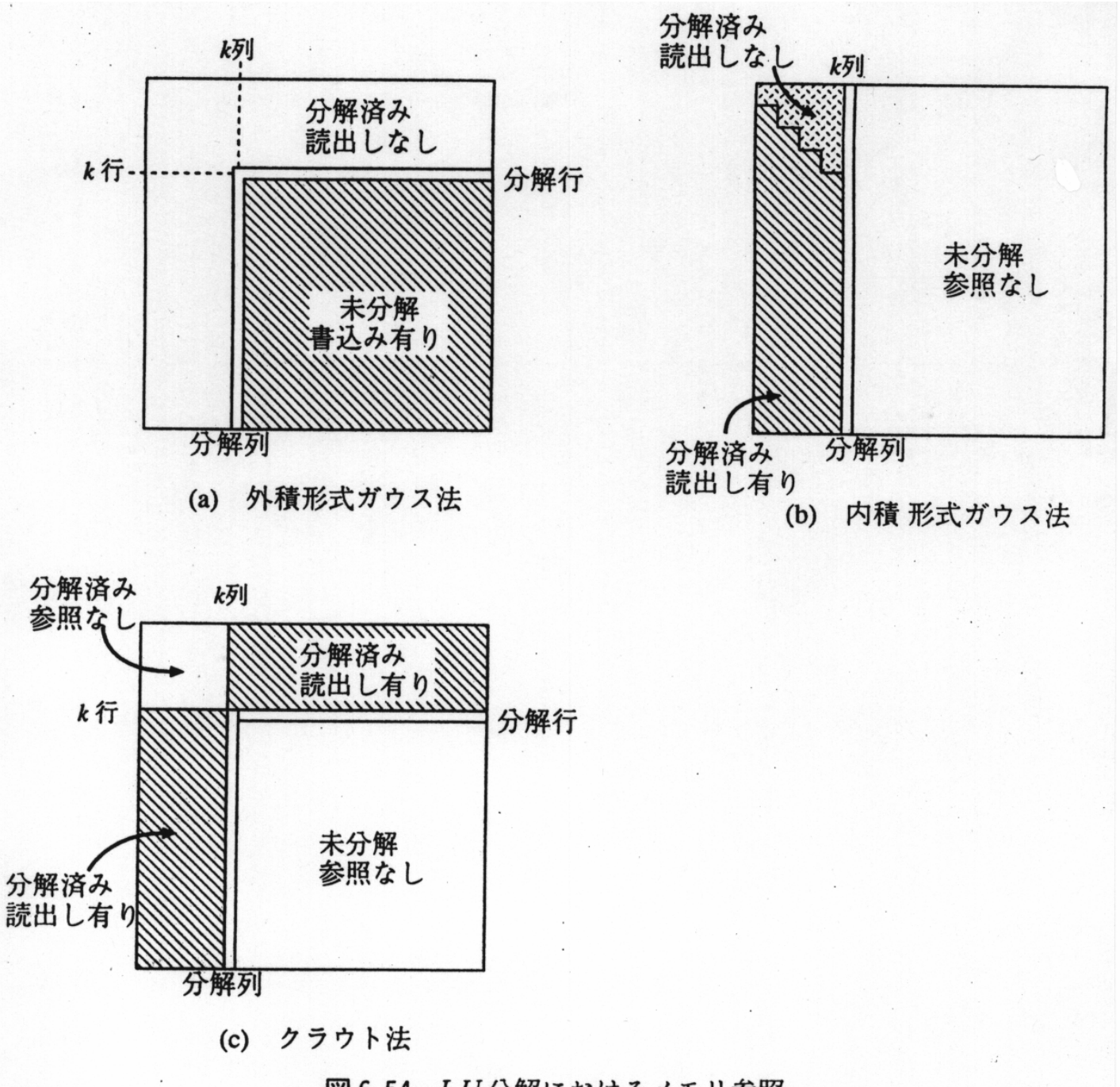


図 6.54 LU分解におけるメモリ参照

(b)　内積形式ガウス法

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ & & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}} = \underset{L}{\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & 0 \\ l_{31} & l_{32} & 1 & \\ & \cdots & & \ddots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}} \underset{U}{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & & \\ 0 & & \ddots & \\ & & & u_{nn} \end{bmatrix}}$$

①$u_{11}=a_{11}$

②$l_{21}u_{11}=a_{21}, l_{31}u_{11}=a_{31}, \ldots, l_{n1}u_{11}=a_{n1}$　並列

③$u_{12}=a_{12}, l_{21}u_{12}+u_{22}=a_{22}$　逐次

④$l_{31}u_{12}+l_{32}u_{22}=a_{32}, l_{41}u_{12}+l_{42}u_{22}=a_{42}, \ldots, l_{n}1u_{12}+l_{n2}u_{22}=a_{n2}$　並列

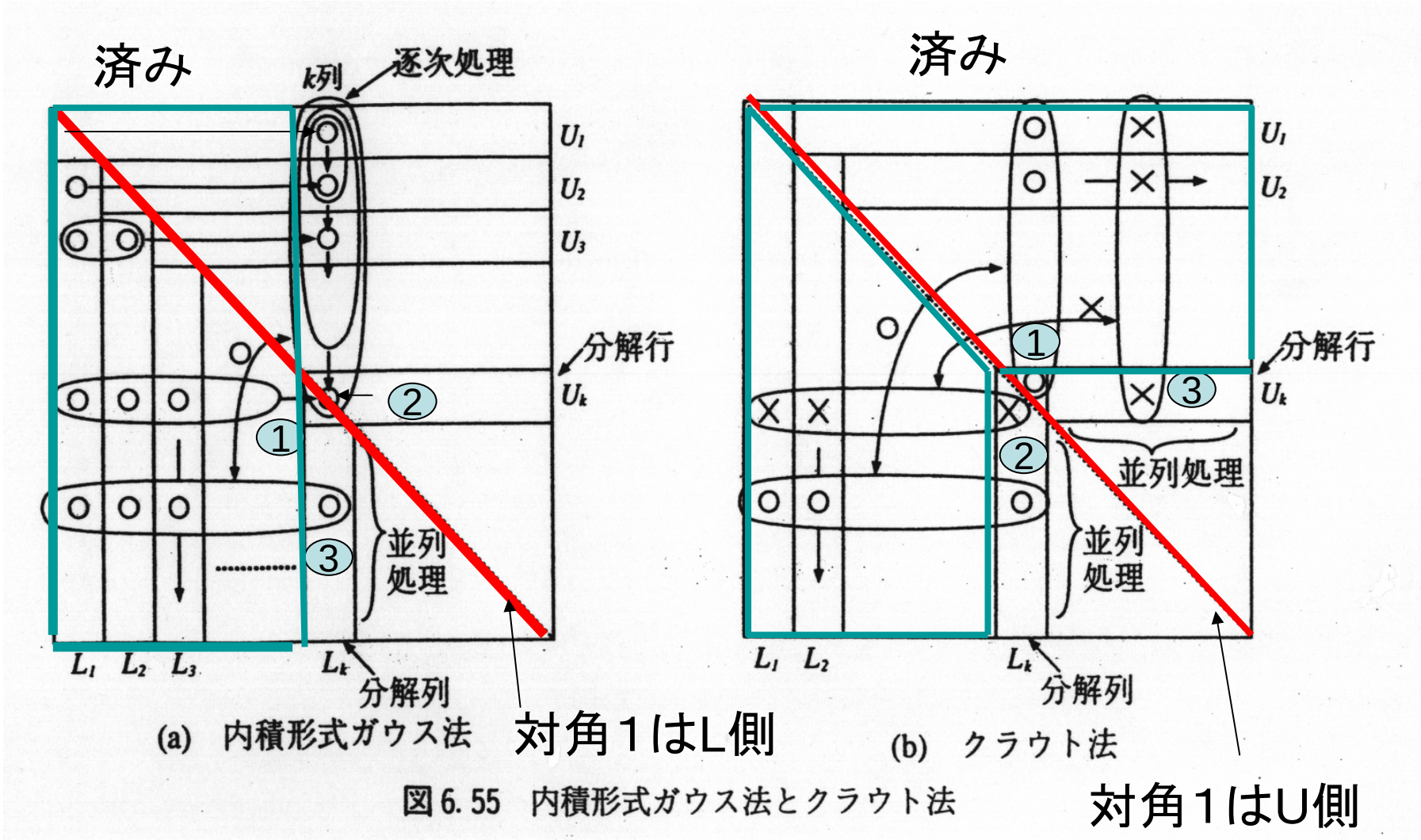


図 6.55 内積形式ガウス法とクラウト法

(c) クラウト法

クラウト法では U の対角要素が 1 に設定されている.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ & & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & 0 \\ l_{31} & l_{32} & l_{33} & \\ & \cdots & & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ & 1 & u_{23} & \cdots & u_{2n} \\ & & & & \\ & 0 & & \ddots & \\ & & & & 1 \end{bmatrix}$$

A L U

(6.90)

$l_{11}=a_{11}, l_{21}=a_{21}, \ldots, l_{n1}=a_{n1}$ 第1列

$l_{11}ul_{12}=a_{12}, l_{11}u_{13}=a_{13}, \ldots, l_{11}u_{1n}=a_{1n}$ 第1行

ガウス消去法の計算時間

逐次処理で計算量：$O(n^3)$

① ベクトルプロセッサの場合

式（８１）より$k=1$**から**$n-1$**に対してベクトル長**$(n-k+1)$**の演算を**$(n-k)$**回行なえばよい。**

$$\Sigma_{k=1}^{n-1} k^2=\Sigma_{k=1}^{n-1}(n-k)^2=n(n-1)(2n-1)/6 \qquad （９４）$$

$$T_{vf}=2\tau_v\Sigma_{k=1}^{n-1}(N_{1/2}+n-k+1)(n-k) \quad =\tau_v(N_{1/2}n^2+2n^3/3)$$

②マルチプロセッサ（Block，＊）

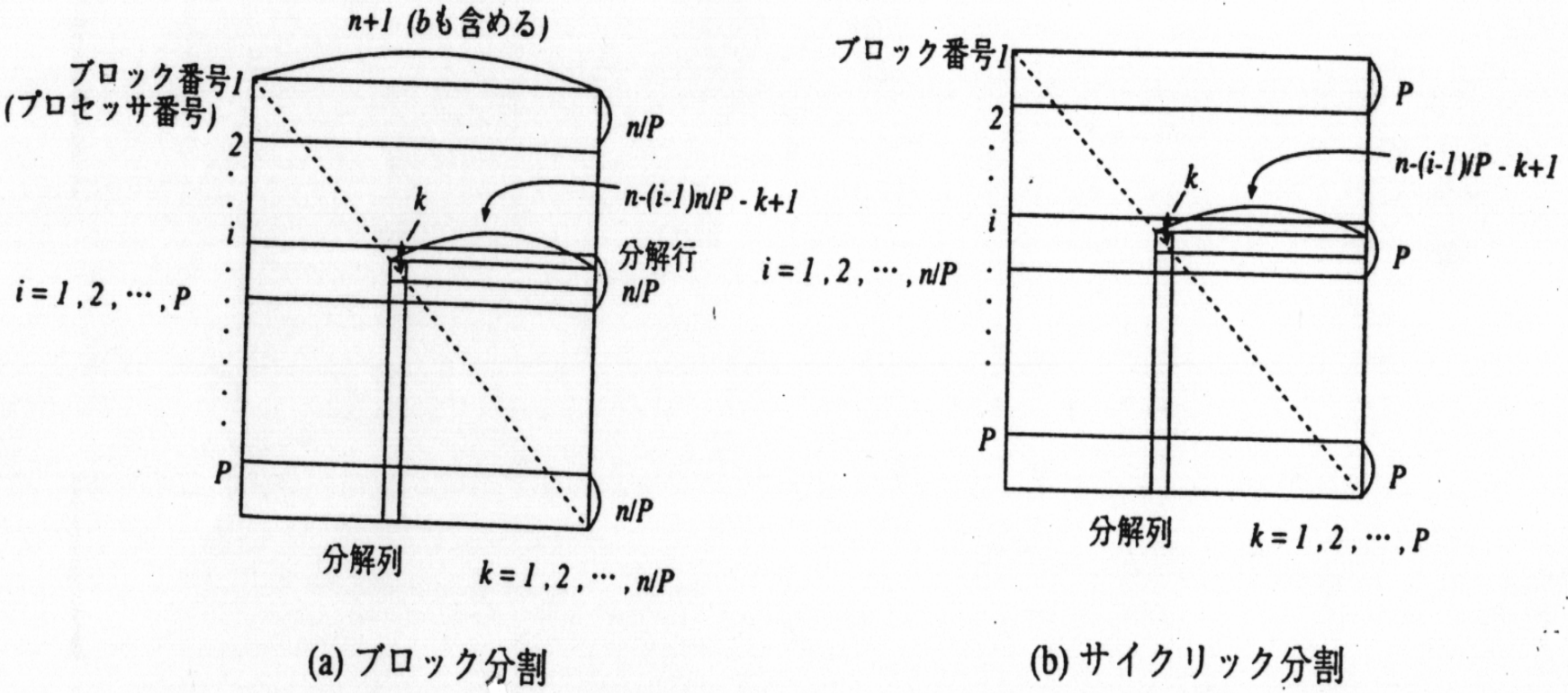


図 6.56　並列ガウス消去法

A：行方向にＰ個のブロックに分割

n、P：大

- 粗粒度の場合

n/P：大

各プロセッサ：n/Pの行ブロックが格納

P_iに割り当てられたk番目の行を分解することを考える。手順は、

(i)P_i（1≦i≦P-1）のk番目の行要素（bも含める）をP_k（k＞i）のプロセッサにブロードキャストする。転送量は(n-(i-1)n/P-k+1)

である。

(ii)P_kではP$_i$から転送された各要素を用いて

式（８１）の計算をｎ／Ｐ行分行なう。計算量は

$$2(n/P)\Sigma_{i=1}^{P-1}\Sigma_{k=1}^{n/P}(n-(i-1)n/P-k+1)$$

$$=(n/P)^2(P-1)(n+n/P+1) \qquad （９６）$$

$$n^3(1/P+1/P^2)$$

(iii)分解行がP_pとなると、他のプロセッサは処理をすべて終えているので、P_pの処理時間は

$$2\Sigma_{k=1}^{n/P-1}(n-(P-1)n/P-k+1)(n/P-k)$$

$$2/3(n/P)^3 \qquad （９７）$$

式（９６）と式（９７）より計算時間は

$$n^3(1/P+1/P^2+2/(3P^3)) \qquad （９８）$$

また、ブロードキャスト時間T_bを

$$T_b=aT_L+bm \qquad （９９）$$

としよう。ここで、T_Lはブロードキャストされたデータの最初の要素が到着するまでの時間（レイテンシ）であり、mはデータ転送量である。a、bは演算時間を１としたときの相対値である。

転送時間：

$$\Sigma_{i=1}^{P-1}\Sigma_{k=1}^{n/P}(aT_L+b(n-(i-1)n/P-k+1))$$

$anT_L+bn^2/2$ （１００）

逐次処理の場合（計算量は$2/3n^3$）と比較したスピードアップは

$2P/(3(1+P(aT_L/n^2+b/(2n))))$ （１０１）

・細粒度の場合

極端な場合：$n=P$の場合

式（９６）と式（１００）より、$n=P$として、

計算 $2\Sigma_{i=1}^{P-1}(P-i+1)=P^2$

通信 $\Sigma_{i=1}^{P-1}(aT_L+b(P-i+1))$

$aPT_L+(b/2)P^2$

（１０２）

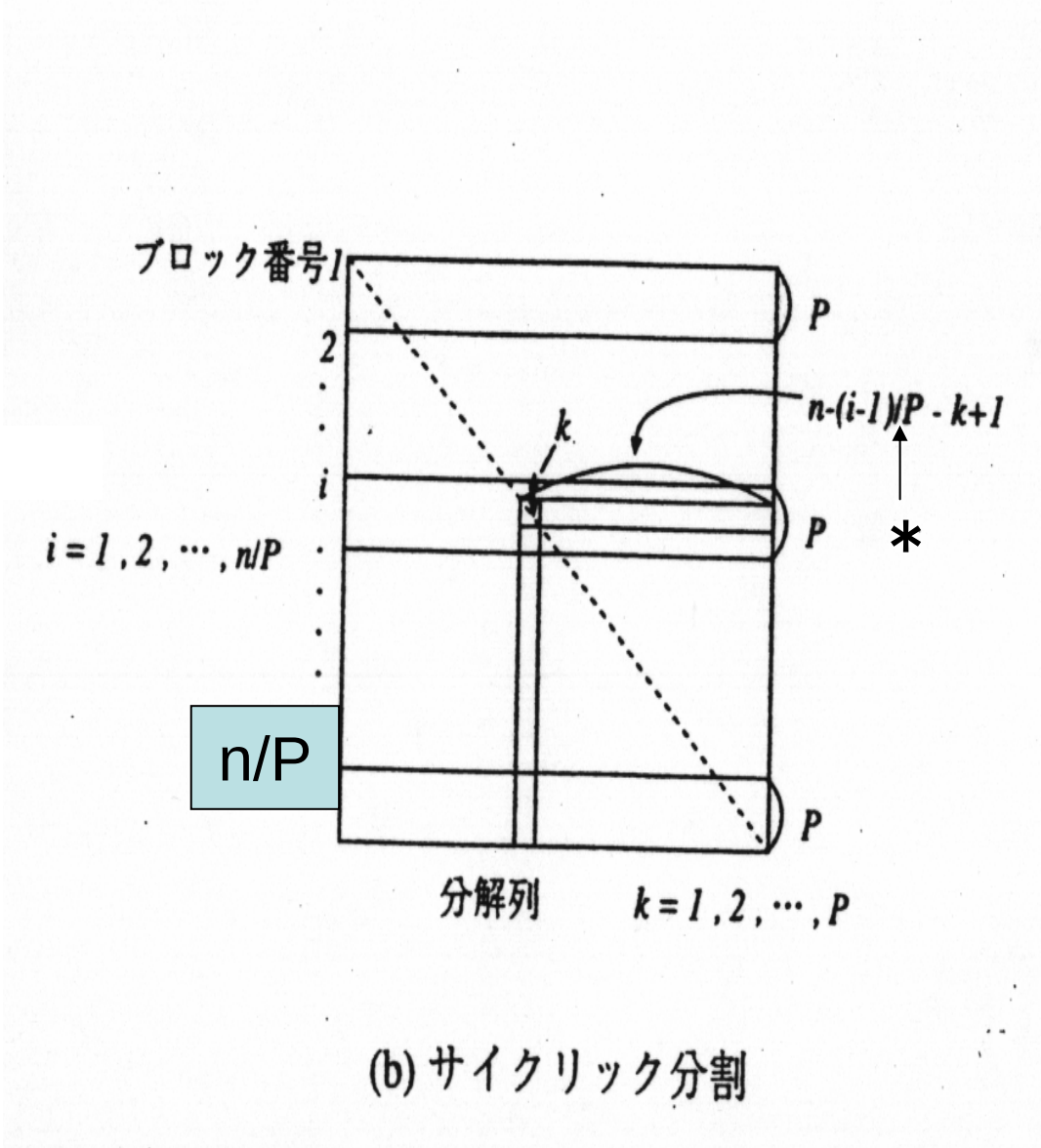
スピードアップ率は

$$(2/3)P^3/\{P^2(1+b/2)+aPT_L\}$$

$$=\{4/(3(2+b))\}P/\{1+2(aT_L/P)/(2+b)\}$$

（１０３）

③（Cyclic，＊）
分割の場合
・粗粒度の場合
　A:行方向に
　n/Pのブロックに分割
各ブロック:P行



(b) サイクリック分割

各行を各プロセッサが担当

第iブロックのプロセッサP_kに割り当てられた行を分解する場合

手順：

(i)P_kの行要素をすべてのプロセッサにブロードキャストする。転送量は

$$(n-(i-1)P-k+1)$$

である。ただし、$i=1,2,\ldots,n/P,k=1,2,\ldots,P$である．

(ii)各プロセッサではP_kから転送された各要素を用いて式（８１）の計算を行なう。各プロセッサでは行

の分解をする必要があるので、計算量は

$$2\Sigma_{i=1}^{n/P}\Sigma_{k=1}^{P}(n-(i-1)P-k+1)(n/P-i+1)$$

$$2n^3/3P \qquad （１０４）$$

データ転送量は式（１００）と同一であるので、スピードアップ率は

$$P/(1+3P/2(aT_L/n^2+b/(2n))) \qquad （１０５）$$

・細粒度の場合

（Block,*）の場合と同一となる。

６．７．３反復法

$$Ax = b$$

緩和法

陽解法　左辺計算後、右辺に代入

$$x^{k \cdot 1} \cdot Sx^{k} \cdot g$$

陰解法　右辺計算後、直接法で $x^{k \cdot 1}$ 解く

$$A_0 x^{k \cdot 1} \cdot Bx^{k} \cdot b$$

共役勾配法

汎関数の最小化問題に置換え

（１）ヤ　コ　ビ　法

$$x^{k+1} = D^{-1}\{(D-A)x^{k} + b\}$$

並　列　処　理　可　能

x^{k}を放送

収束が遅く実用的にはX

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij}\, x_j\right)$$

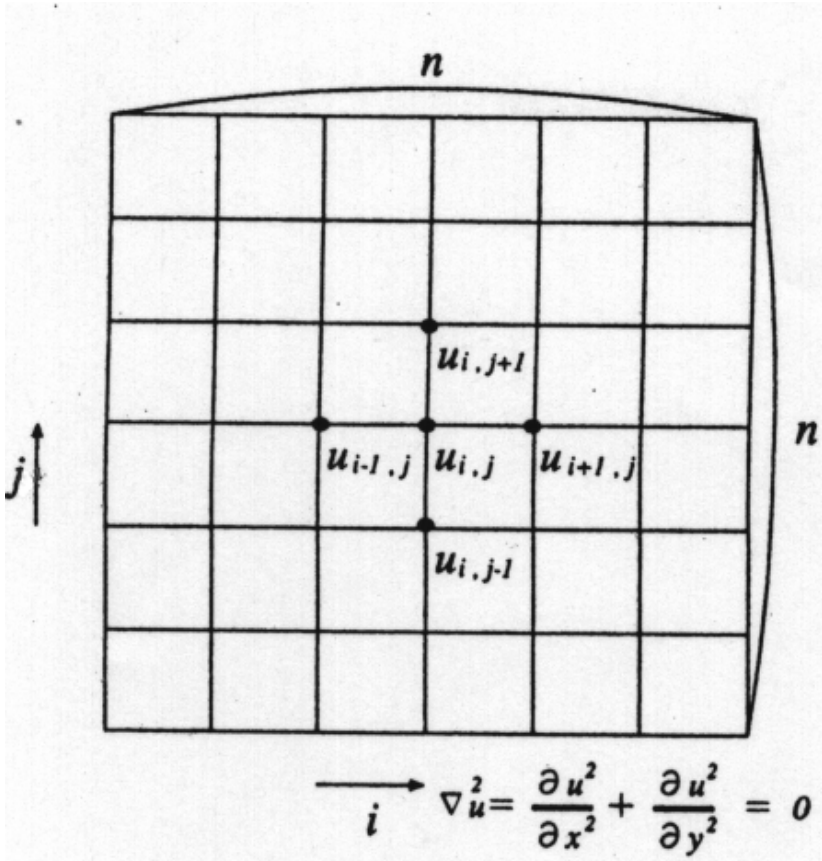
これより，

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij}\, x_j^{k}\right)$$

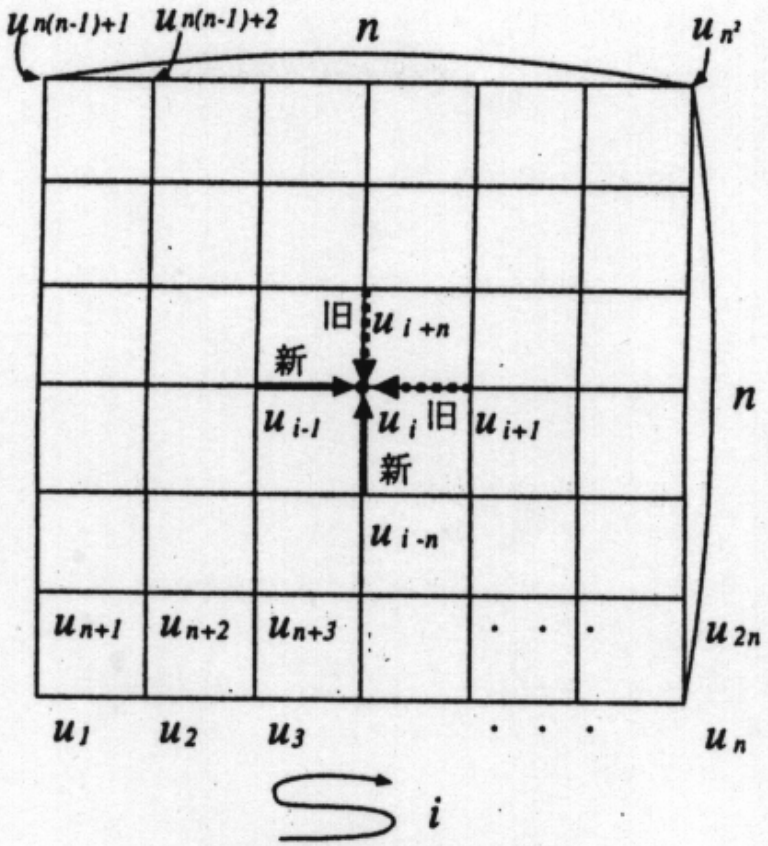
ラプラス方程式

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

$$u_{ij} = (u_{i+1\,j} + u_{i-1\,j} + u_{ij+1} + u_{ij-1}) / 4$$



(a) 2次元番号付け



(b) 1次元番号付け

(2)ガウスーザイデル法

(a)基本アルゴリズム

$$\begin{pmatrix} x_1 \\ x_2 \\ x_i \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_i \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_i \\ b_n \end{pmatrix}$$

新しい値

古い値

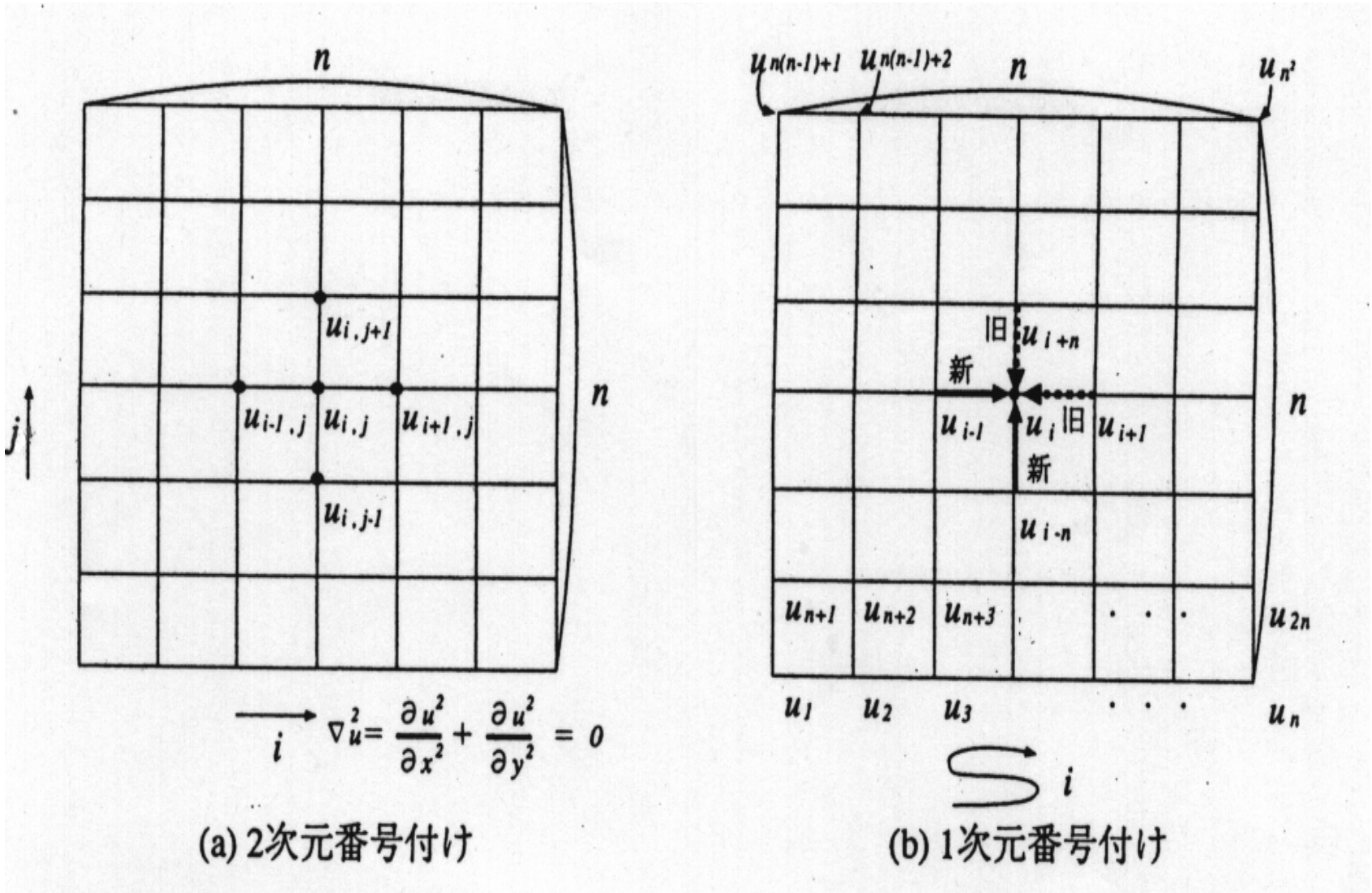
$$A = L + D + U$$

$$Dx = b - Lx - Ux$$

$$x^{k+1} = D^{-1}(b - Lx^{k+1} - Ux^{k})$$

収束速いが逐次的

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\, x_j^{k+1} - \sum_{j>i} a_{ij}\, x_j^{k}\right)$$



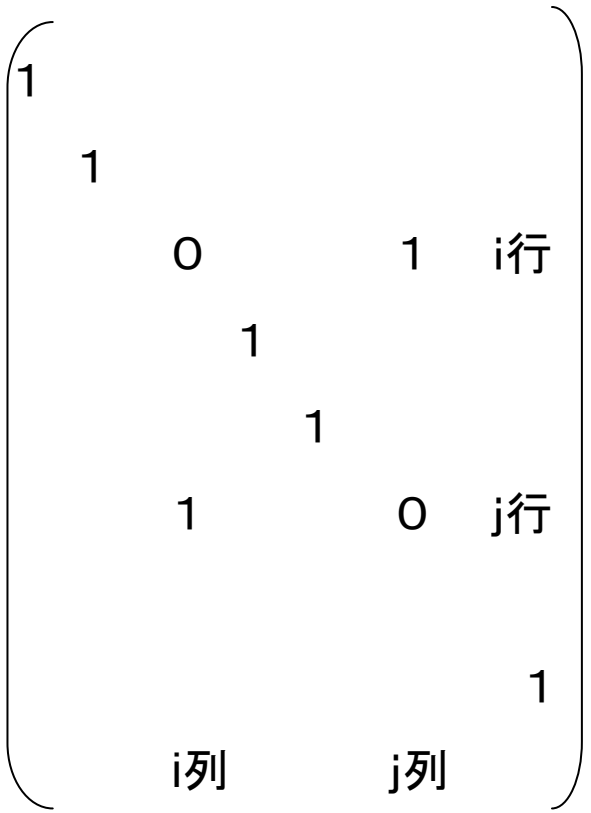
(a) 2次元番号付け

(b) 1次元番号付け

(b) Red-Black法

要素の入れ替えで並列処理できる場合あり

xの要素x_iとx_jを入れ替える行列P_{ij}



入替え行列P

$$PP^T = I, P^T = P^{-1}$$

$$PAP^{-1}Px = Pb$$

$$Px = y, P_b = c$$

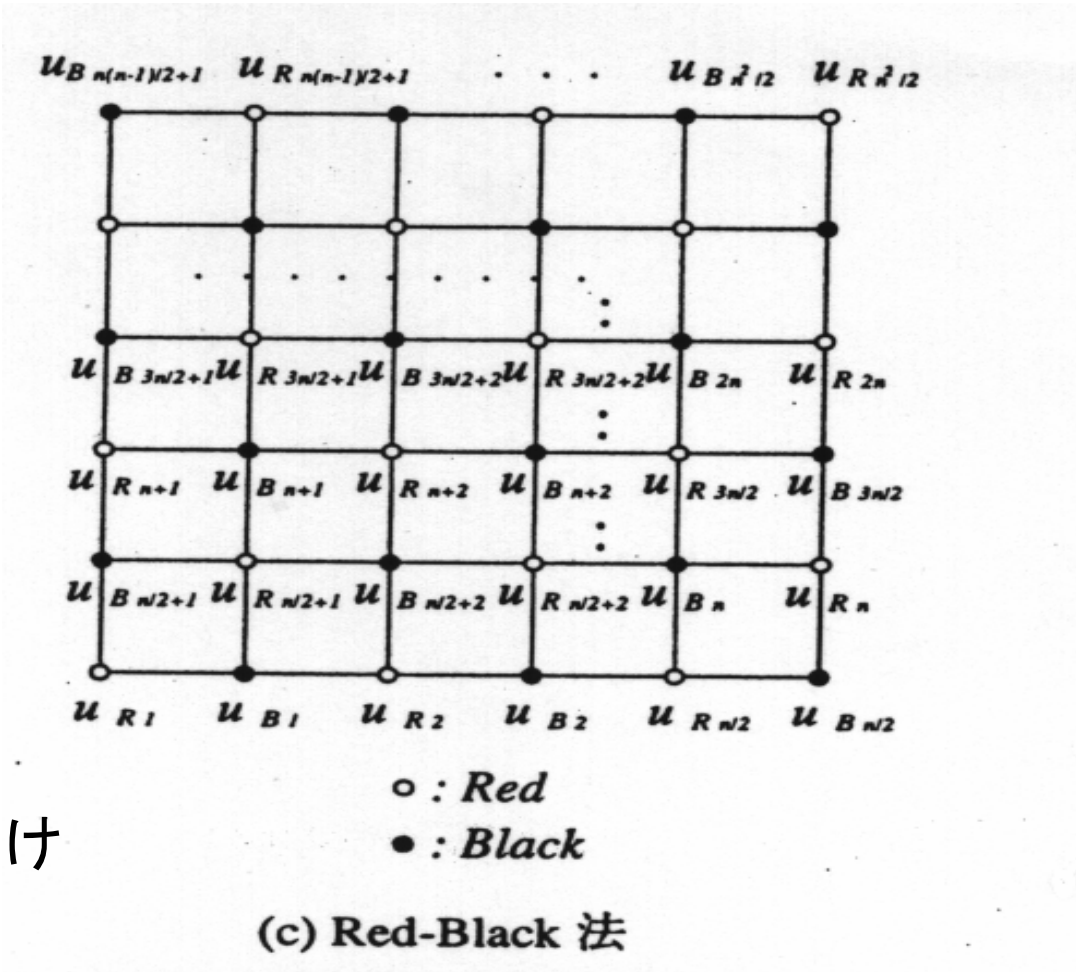
$$PAP^T = \begin{pmatrix} D_R & E \\ F & D_B \end{pmatrix}$$

とすると

$$\begin{pmatrix} D_R & E \\ F & D_B \end{pmatrix} \quad \begin{pmatrix} \mathbf{y_R} \\ \mathbf{y_B} \end{pmatrix} = \begin{pmatrix} \mathbf{C_R} \\ \mathbf{C_B} \end{pmatrix}$$

$$D_R y_R^{k+1} = C_R - E y_B^k$$

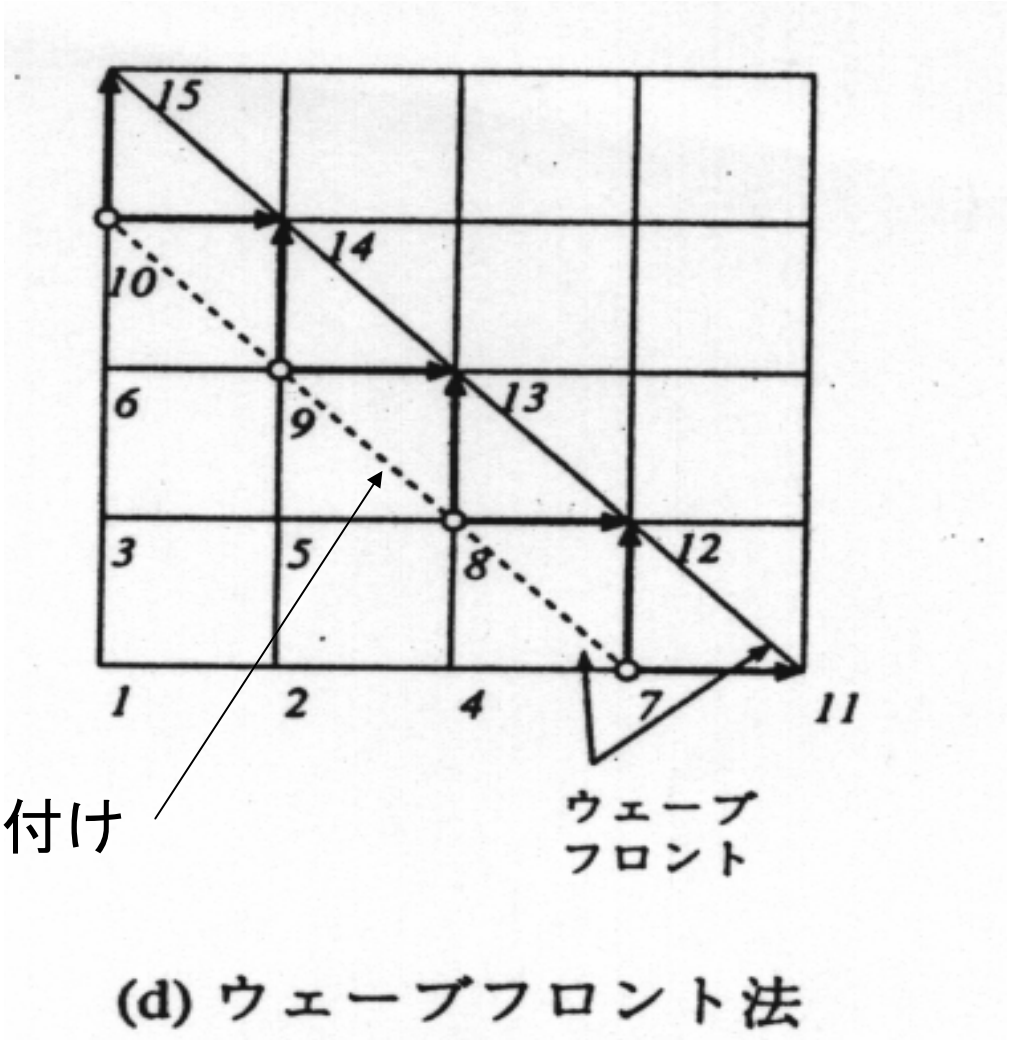
$$D_B y_B^{k+1} = C_B - F y_k^{k+1}$$



番号付け

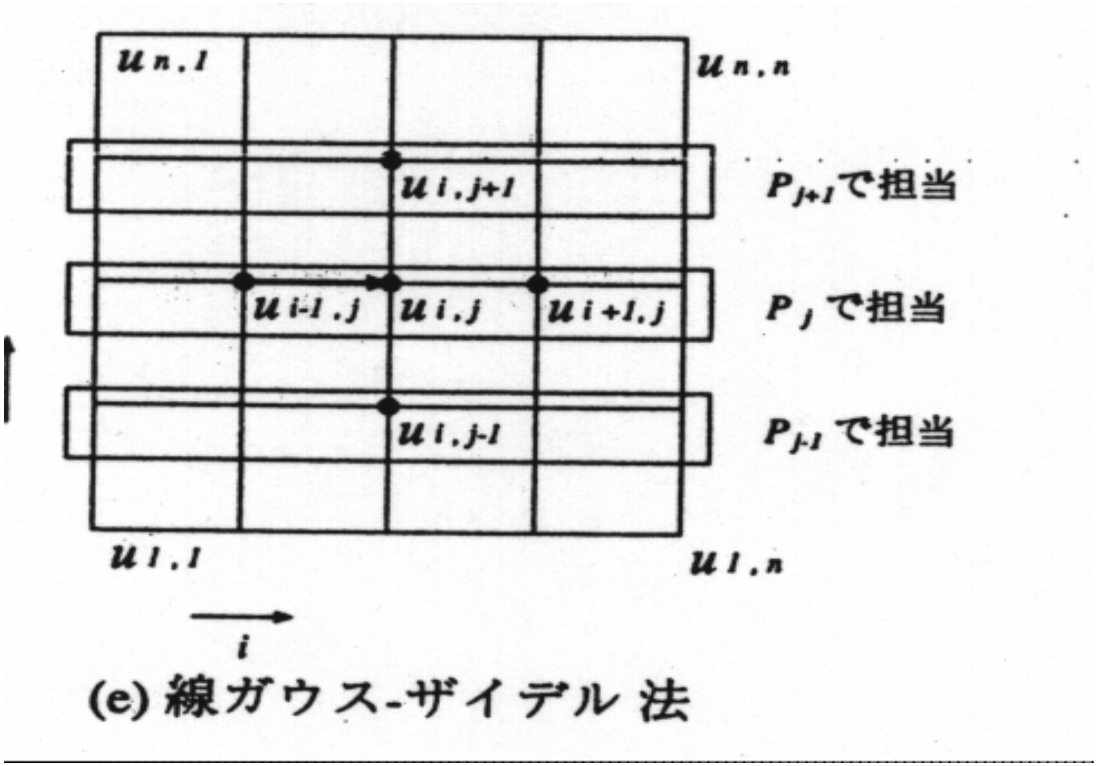
(c) Red-Black 法

（c）ウェーブフロント方式



(d) ウェーブフロント法

（d）線ガウスーザイデル
方式



(e) 線ガウス-ザイデル 法

（3） SOR法

$$x^{k+1} = (1-\omega)x^{k} + \omega D^{-1}(b - Lx^{k+1} - Ux^{k})$$

（4） ADI法： 陰解法

（4） ＡＤＩ法[349]

図 6.57 に示すラプラス方程式を考える．ADI (Alternate Direction Implicit)法では，１回の反復を X 方向(I 方向)と Y 方向(J 方向)に分割して，それぞれを陰解法で解く．ラプラスの方程式を

$$u_{ij}^{k+1/2}=u_{ij}^{k}+(1/\alpha)\,(u_{i+1\,j}^{k+1/2}+u_{i-1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i\,j+1}^{k}+u_{i\,j-1}^{k}-2u_{ij}^{k})$$

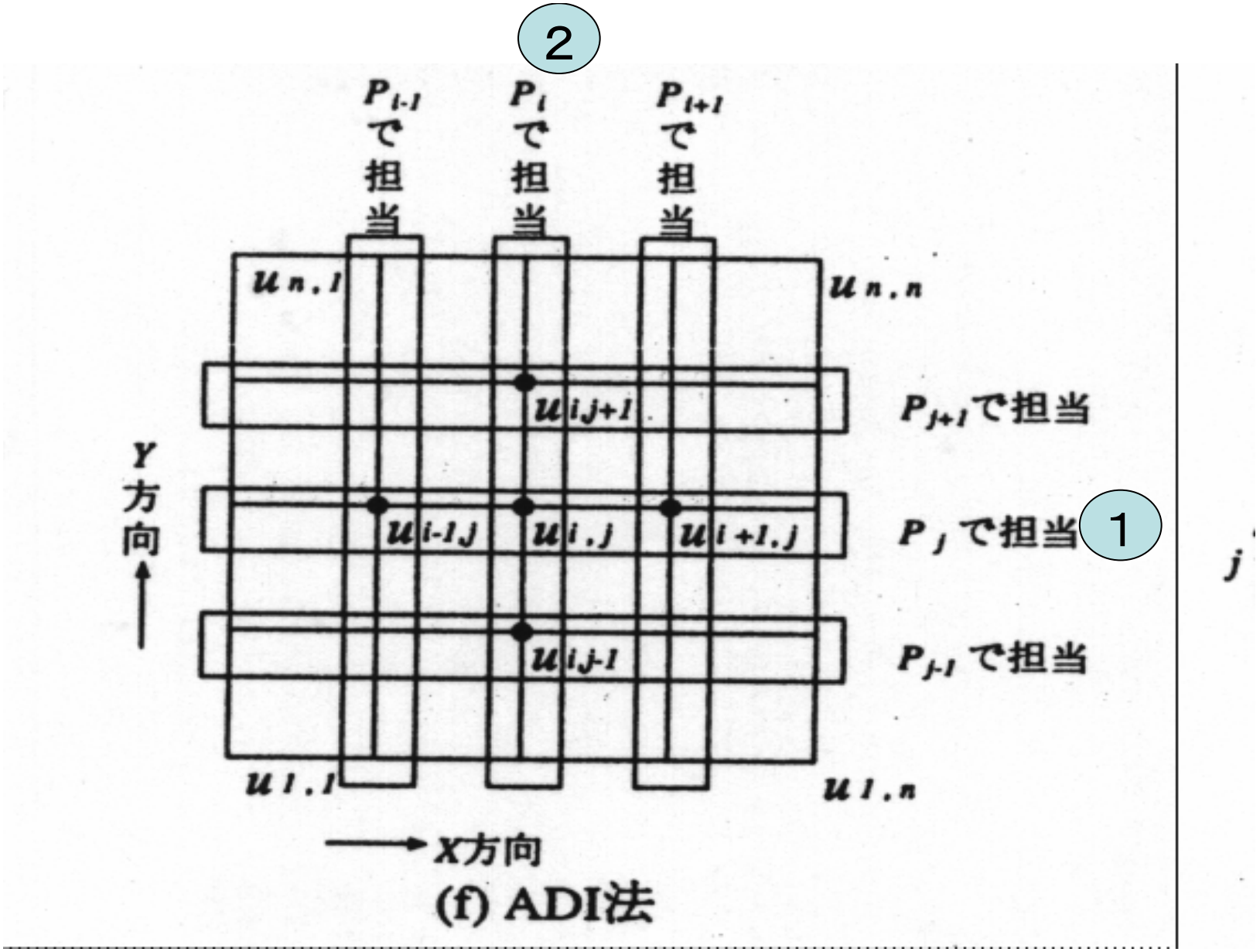
$$u_{ij}^{k+1}=u_{ij}^{k+1/2}+(1/\alpha)\,(u_{i\,j+1}^{k+1}+u_{i\,j-1}^{k+1}-2u_{ij}^{k+1/2}+u_{i+1\,j}^{k+1/2}+u_{i-1\,j}^{k+1/2}-2u_{ij}^{k+1/2}) \qquad (6.124\,\mathrm{a})$$

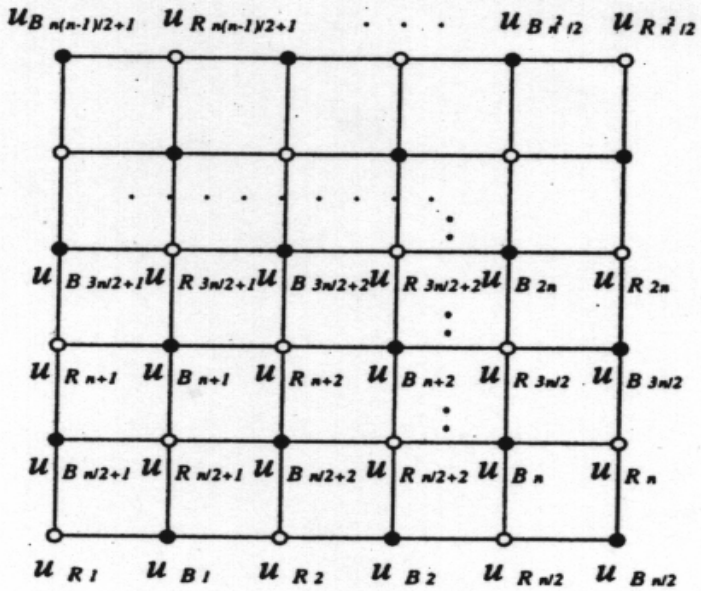
で反復して解く．これより，

$$(\alpha+2)u_{ij}^{k+1/2}-u_{i+1\,j}^{k+1/2}-u_{i-1\,j}^{k+1/2}=(\alpha-2)u_{ij}^{k}+u_{i\,j+1}^{k}+u_{i\,j-1}^{k} \qquad (6.124\,\mathrm{b})$$

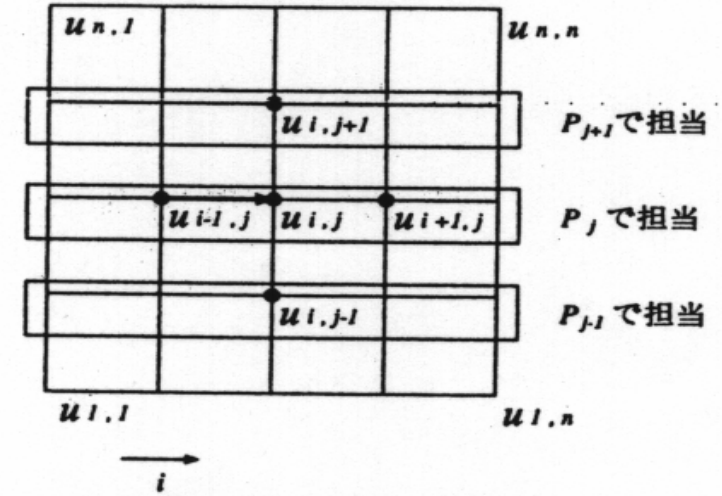
$$(\alpha+2)u_{ij}^{k+1}-u_{i\,j+1}^{k+1}-u_{i\,j-1}^{k+1}=(\alpha-2)u_{ij}^{k+1/2}+u_{i+1\,j}^{k+1/2}+u_{i-1\,j}^{k+1/2} \qquad (6.124\,\mathrm{c})$$

式(6.124)において，式(1)

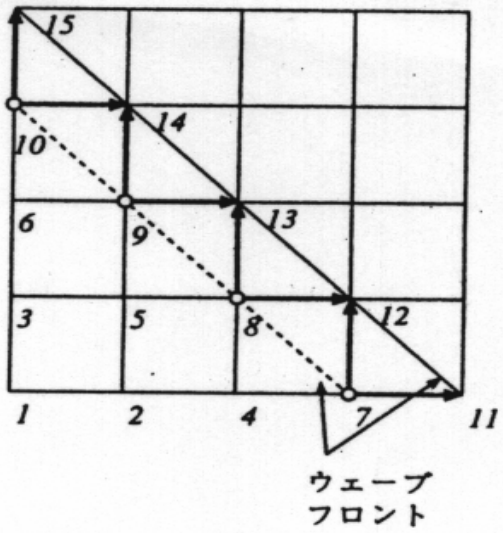

2
P_{i-1} で担当
P_i で担当
P_{i+1} で担当
$u_{n,1}$
$u_{n,n}$
$u_{i,j+1}$
P_{j+1}で担当
Y方向
$u_{i-1,j}$
$u_{i,j}$
$u_{i+1,j}$
P_j で担当
1
$u_{i,j-1}$
P_{j-1} で担当
$u_{1,1}$
$u_{1,n}$
X方向
(f) ADI法



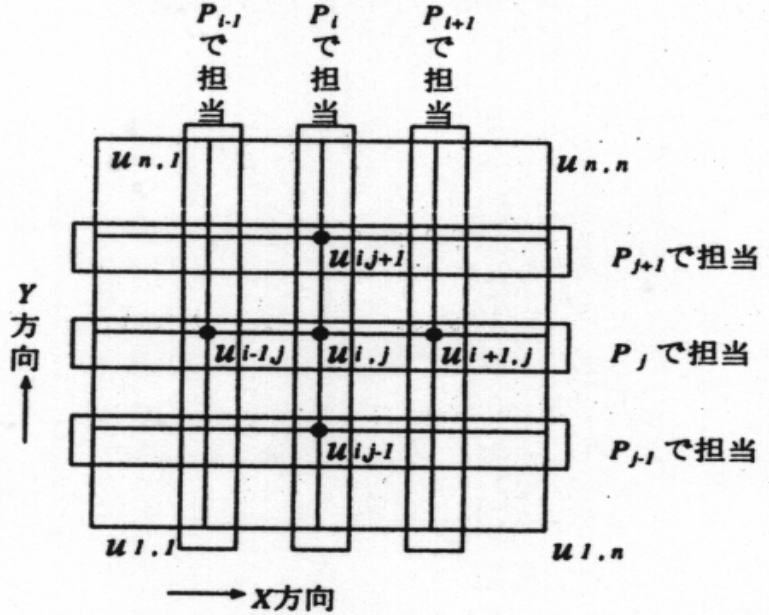
(c) Red-Black 法



(e) 線ガウス-ザイデル 法



(d) ウェーブフロント法



(f) ADI法

（5）共役勾配法

行列A：対称、正定値行列

$$x^T Ax > 0$$

汎関数の最小化：解を求めること

$$F(x) = x^T Ax - 2x^T b$$

$$F(x)=x^TAx-2x^Tb \tag{6.125}$$

を考える．共役勾配法では，$F(x)$を最小化するxを求めることが$Ax=b$の解を求めることと同一になる，ことを利用する．まずこれを証明しよう．

[証明]　任意のベクトルhに対して

$$\begin{aligned}F(x+h)&=(x+h)^TA(x+h)-2(x+h)^Tb\\&=F(x)+2h^T(Ax-b)+h^TAh\end{aligned} \tag{6.126}$$

である．Aが対称行列であり，$x^TAh=(Ax)^Th=h^TAx$である．$Ax=b$なら，任意のベクトルに対して，$h^TAh>0$より，

$$F(A^{-1}b+h)>F(A^{-1}b) \tag{6.127}$$

であるので，$x=A^{-1}b$でFは最小である．

また，$F(x)$が最小なら$Ax=b$であることは，その対偶をとって，$Ax\neq b$なら，$F(x)$は最小でないことを示せばよい．

$r=Ax-b$の非ゼロ要素をr_k，$h^T=(0,0,\cdots,h_k,\cdots,0)$とすると，

$$F(x+h)-F(x)=2h_kr_k+a_{kk}h_k^2<0$$

となるようなh_kを採ることができる．したがって，$F(x)$は最小ではない．

Q. E. D.

さて，$F(\boldsymbol{x})$の最小化問題を考えよう．xの反復を与える式として，

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^{k}+\alpha_k \boldsymbol{p}_k \tag{6.128}$$

を考える．これを式(6.125)に代入すると，$A=A^T$であるので，

$$F(\boldsymbol{x}^{k+1})=F(\boldsymbol{x}^{k})-2\alpha_k \boldsymbol{p}_k^T \boldsymbol{r}_k+\alpha_k{}^2 \boldsymbol{p}_k^T A\boldsymbol{p}_k \tag{6.129}$$

ここで，

$$\boldsymbol{r}_k=\boldsymbol{b}-A\boldsymbol{x}^{k} \tag{6.130}$$

である．式(6.129)をα_kについての2次式と見れば，これを最小化するのは

$$\alpha_k=\boldsymbol{p}_k^T \boldsymbol{r}_k/\boldsymbol{p}_k^T A\boldsymbol{p}_k \tag{6.131}$$

の場合である．

次に$\boldsymbol{p}_k$を下記のように定める．まず，相異なる$\boldsymbol{p}_i$と$\boldsymbol{p}_j$について，A直交，すなわち，

$$\boldsymbol{p}_i^T A\boldsymbol{p}_j=0 \tag{6.132}$$

となるようにする．

$$F(x^{k+1})-F(x^{k})<p_k{}^T Ap_k\left[\alpha_k-p_k{}^T r_k/(p_k{}^T Ap_k)\right]^2-\left(\frac{p_k^T r_k}{p_k^T Ap_k}\right)^2$$

$$<-\left(\frac{p_k^T r_k}{p_k^T Ap_k}\right)^2$$

また，

$$p_{k+1}=r_{k+1}+\beta_k p_k \tag{6.133}$$

とすると，A 直交性より

$$p_{k+1}^T A p_k=r_{k+1}^T A p^k+\beta_k p_k^T A p_k=0 \tag{6.134}$$

であるので，

$$\beta_k=-r_{k+1}^T A p_k / p_k^T A p_k \tag{6.135}$$

また，式(130)より

$$\begin{aligned} r_{k+1} &= b-Ax^{k+1} \\ &= b-A(x^k+\alpha_k p_k) \\ &= r_k-A\alpha_k p_k \end{aligned} \tag{6.136}$$

したがって以上より初期値を x^0 とすると，$p_0=r_0=b-Ax^0$ であるので，プログラムは

$$\begin{aligned} &\text{While} \quad |r_{k+1}|>\varepsilon \\ &\qquad \alpha_k=p_k^T r_k / p_k^T A p_k && \text{(i)} \\ &\qquad x^{k+1}=x^k+\alpha_k p_k && \text{(ii)} \\ &\qquad r_{k+1}=r_k-\alpha_k A p_k && \text{(iii)} \\ &\qquad \beta_k=-r_{k+1}^T A p_k / p_k^T A p_k && \text{(iv)} \\ &\qquad p_{k+1}=r_{k+1}+\beta_k p_k && \text{(v)} \end{aligned} \tag{6.137}$$

となる．逐次処理では，(i)には $2n^2+4n$，(ii)から(v)にはそれぞれ $2n$ 時間必要となる．また，共役勾配法は反復回数 n で確実に収束することが理論的に保証されている．

①逐次処理

(ｉ)には$2n^2+4n$、(ｉｉ)から(v)にはそれぞれ2n時間必要となる。また、共役勾配法は反復回数nで確実に収束することが理論的に保証されている。

②並列処理

Aが密行列であり、粗粒度（n/Pは大）の場合を考える。各プロセッサには行列A、ベクトルx^k、p_k、r_kの行ブロック（行数はn/p。Ｐはプロセッサ台数）を担当させる場合(Block,*)を考える。上記プログラムの各文((ｉ)ー(v)）について考える。

（ｉ）

・Ap_kの処理

　行列とベクトルの積である。各プロセッサから他のプロセッサに長さn/Pのデータのブロードキャストが必要である。ＨＰＦの項で述べたように通信と処理のオーバラップができれば理想であるが、ここではオーバラップはないものとする。式（９９）より、通信時間は

$$(aT_L+bn/P)P \qquad （１３８）$$

計算時間は

$$2n^2/P \qquad (139)$$

である。

・$p_k^T r_k$と$p_k^T A p_k$の処理

これは内積計算である。各プロセッサでのｎ／Ｐ個のデータの積和時間は

$$4n/P \qquad (140)$$

各プロセッサ内の結果のプロセッサ間での積和計算時間はトリー状に$\log P$段で計算されるとし、

$$cT_N \log P \qquad (141)$$

と表されるとしよう。T_Nはプロセッサ間の平均通信時

間であり、logP（ハイパキューブ）、√P（トーラス）、P（リング）などと表されよう。

・α_kのブロードキャスト

転送時間は

(aT_L+b) （１４２）

（ｉｉ）、（ｉｉｉ）、（ｖ）

各プロセッサで独立に処理できる。

計算時間は

$6n/p$ （１４３）

（IV）

- $r_{k+1}{}^{T}Ap_k$の処理

  これはは内積計算である。

  各プロセッサでのｎ／Ｐ個のデータの積和時間は

  $2n/P$ （１４４）

  各プロセッサ内の結果の積和計算の時間は

  $cT_N \log P$ （１４５）

- β_kのブロードキャスト

  転送時間は

  (aT_L+b) （１４６）

逐次処理の場合には$(2n^2+12n)$の時間がかかる。式（１３８）から（１４５）より、

並列処理によるスピードアップは

$$P/\{1+(aT_L/2)(P/n)^2+(bP/2)/n+cT_NP\log P/n^2\} \quad (147)$$

である。ＣＧ法は並列処理に向いた方式といえる。