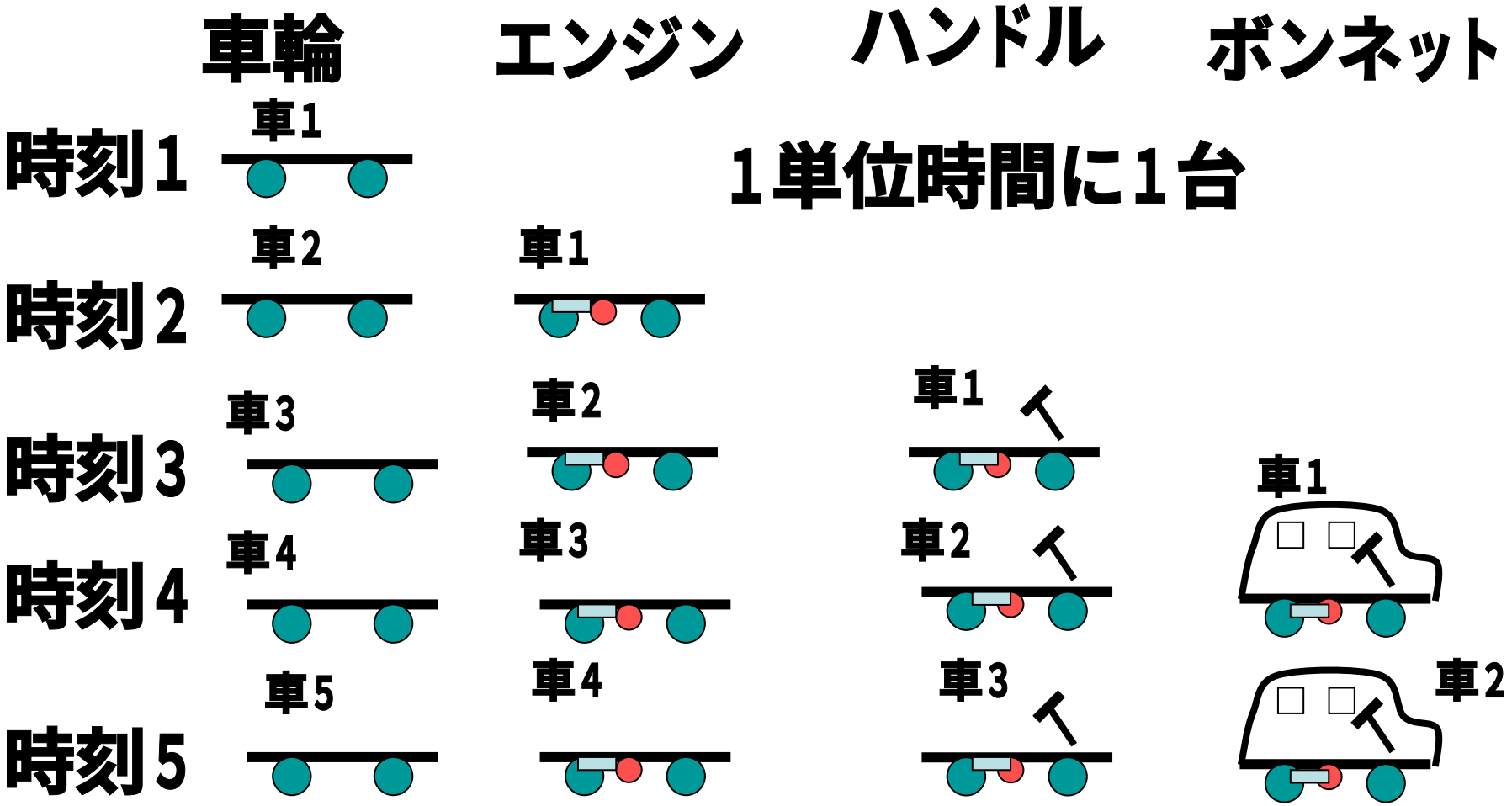
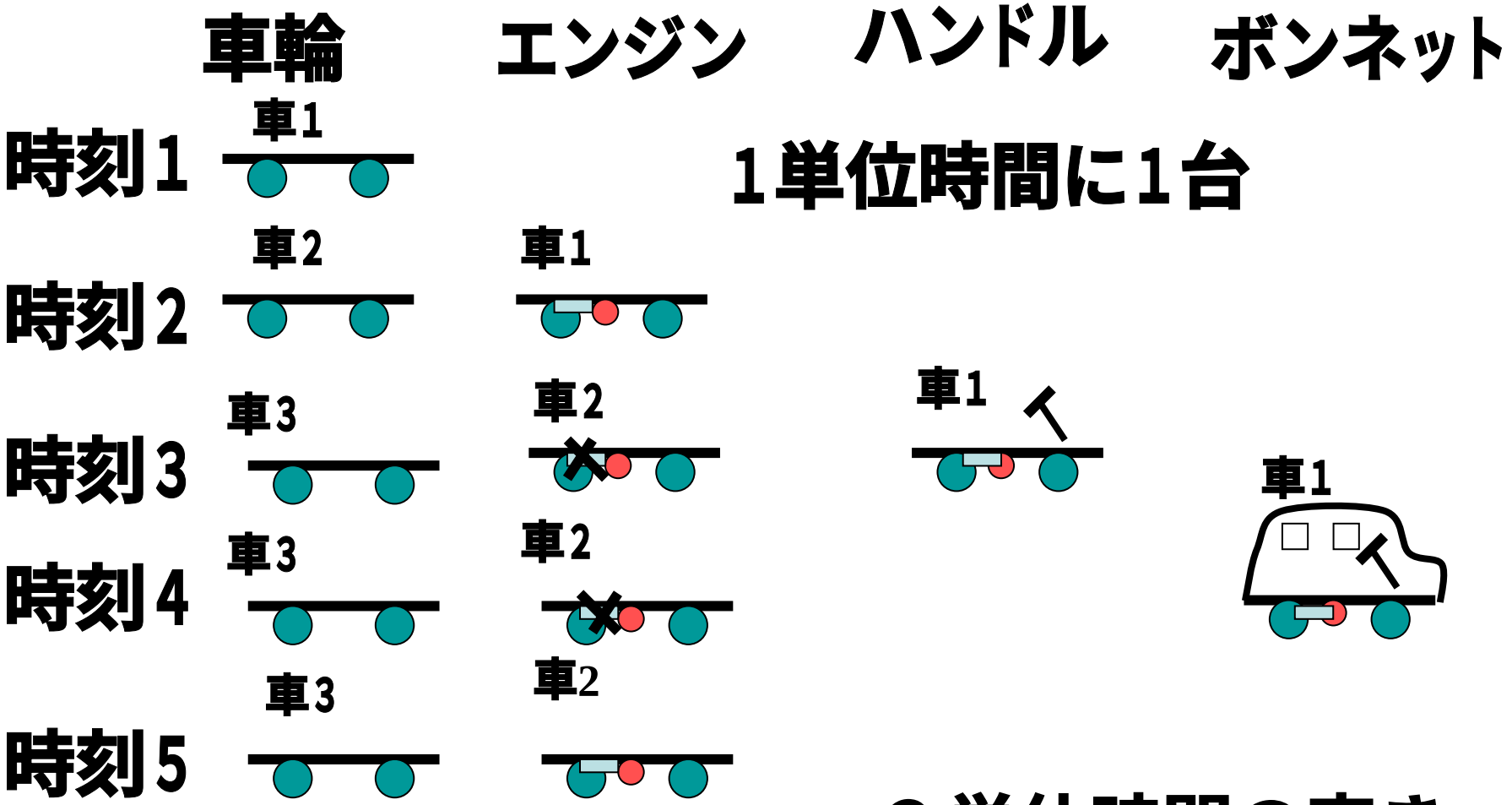
第4章　パイプライン方式

- スーパコンピュータ：　ベクトルコンピュータ
  数値計算向きの汎用コンピュータ
- プレステ2で採用
- 専用コンピュータ
  高速フーリエ変換、ＣＧ、画像処理、
  ソーティングなど
  ー＞第6章

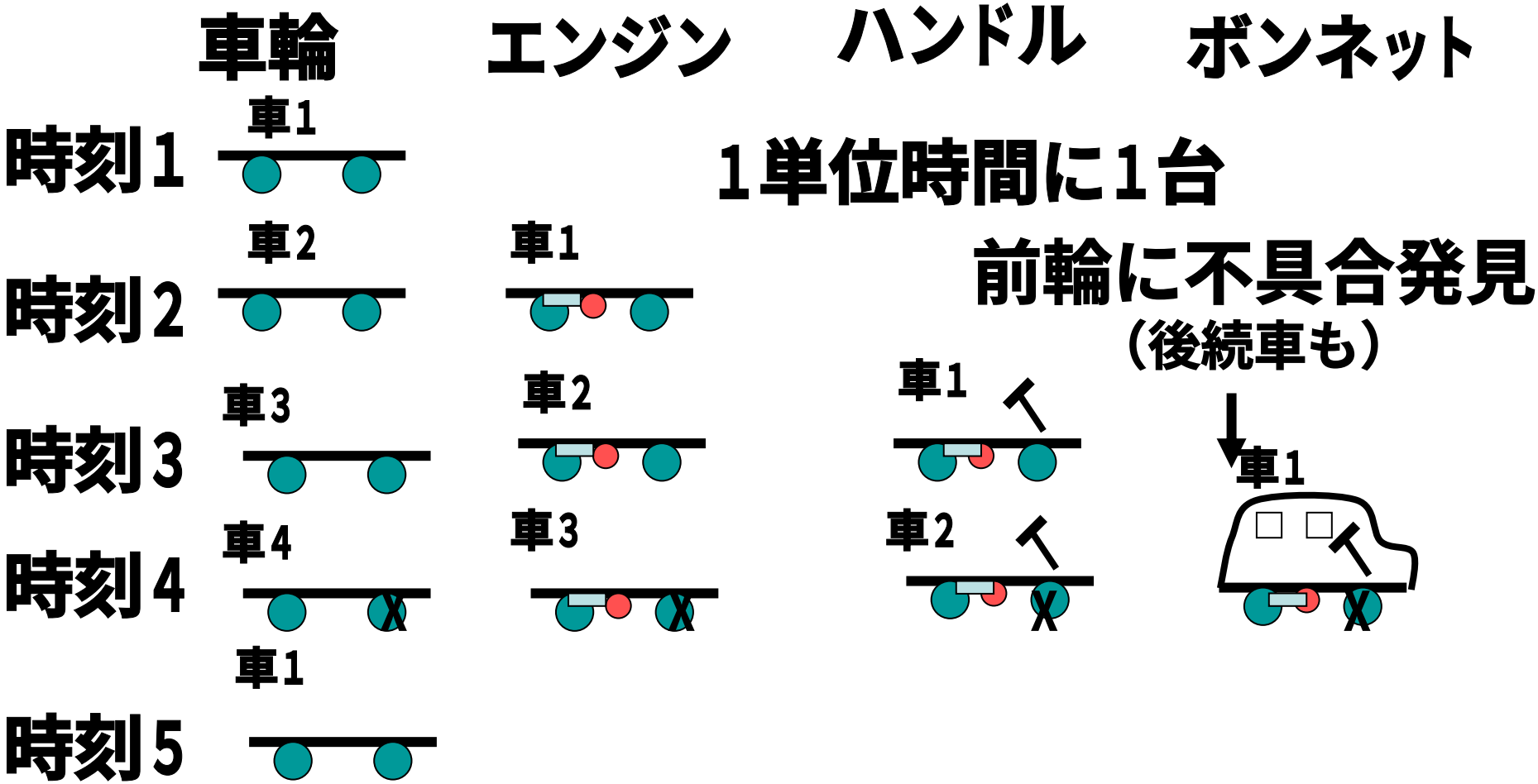
パイプライン＝流れ作業

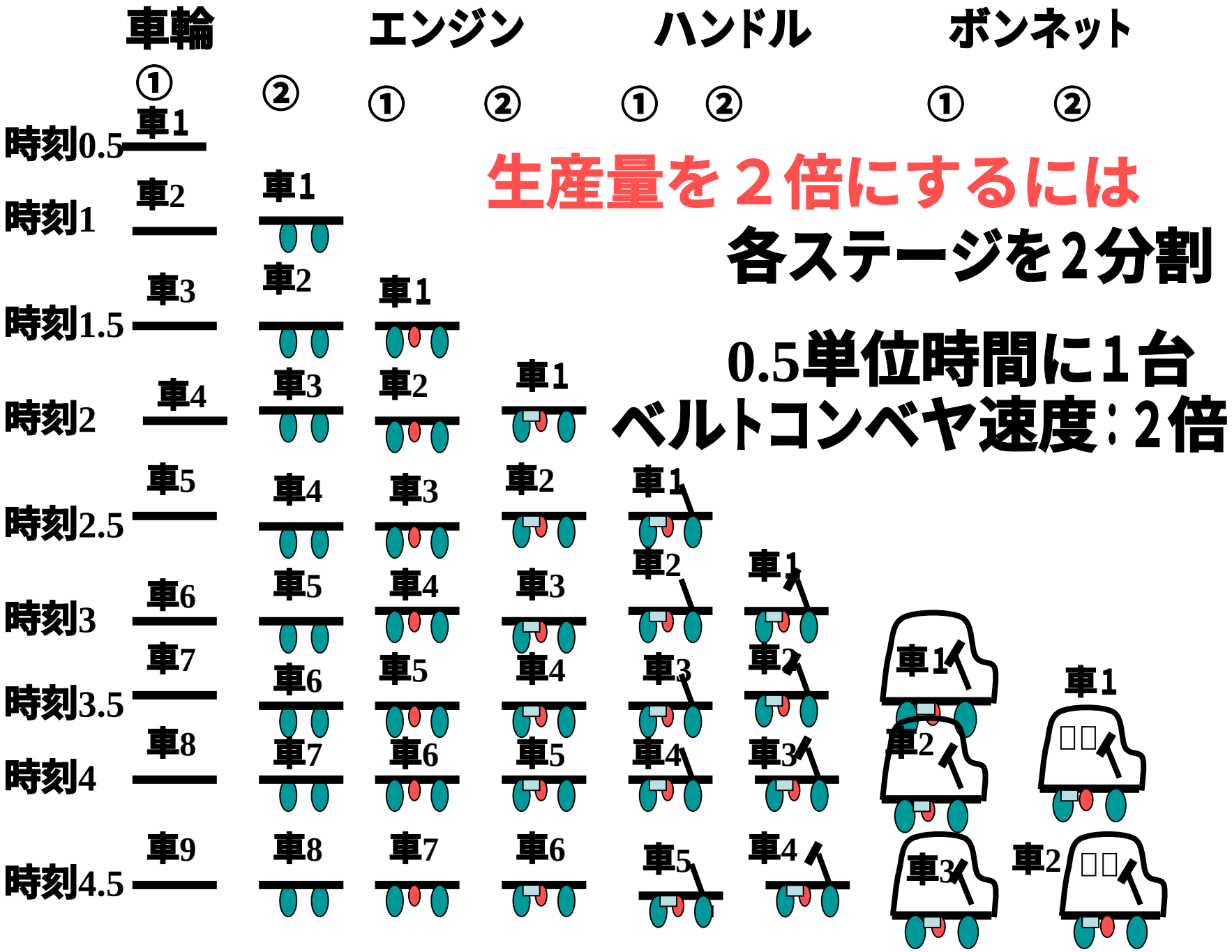

車輪
エンジン
ハンドル
ボンネット
時刻1
車1
1単位時間に1台
時刻2
車2
車1
時刻3
車3
車2
車1
時刻4
車4
車3
車2
車1
時刻5
車5
車4
車3
車2

流れ作業の乱れ
(1) 部品調達遅れ



流れ作業の乱れ
(2) 検査不良による再取付け




車輪
エンジン
ハンドル
ボンネット
①
②
①
②
①
②
①
②
時刻0.5
車1
時刻1
車2
車1
時刻1.5
車3
車2
車1
時刻2
車4
車3
車2
車1
時刻2.5
車5
車4
車3
車2
車1
時刻3
車6
車5
車4
車3
車2
車1
時刻3.5
車7
車6
車5
車4
車3
車2
車1
時刻4
車8
車7
車6
車5
車4
車3
車2
車1
時刻4.5
車9
車8
車7
車6
車5
車4
車3
車2
生産量を２倍にするには
各ステージを2分割
0.5単位時間に1台
ベルトコンベヤ速度：2倍

パイプライン型スーパコンピュータ

- 別名：ベクトルプロセッサ（コンピュータ）
- ベクトルデータの高速処理

  Z（I）＝X（I）＋Y（I）

  T＝Σ X（I）＊Y（I）：　内積

  T＝MAX（X（I））, MIN（X（I））

  X（I＋1）＝A（I）＊X（I）＋B（I）：　漸化式
- 線形代数：行列、ベクトル

  AX、　AB、　A^T、　A^{-1}

ユーザ側からの普及理由

①高速性：通常の計算機より１桁以上高速

②連続性：過去に作成されたFORTRANプログラムが
そのまま利用

③単純性：ベクトル演算で問題が定式化できれば
最大性能
ベクトル：高等学校数学

④汎用性：多様な数値計算分野に適応

⑤単一性：各社ハードウェアシステムの構造が同一
「よい」プログラムはどのスーパ
コンピュータでも高速実行

４．１　基本方式

（１）同期式パイプライン

半性能長：最大性能（$1/\tau_v$）の半分の
性能となるデータ要素数

演算時間

$$T=S+(N-1+L)\tau_v \qquad （１）$$

$$=(N+N_{1/2})\tau_v \qquad （２）$$

$$1/(2\tau_v)=N_{1/2}/(S+(N_{1/2}-1+L)\tau_v)$$

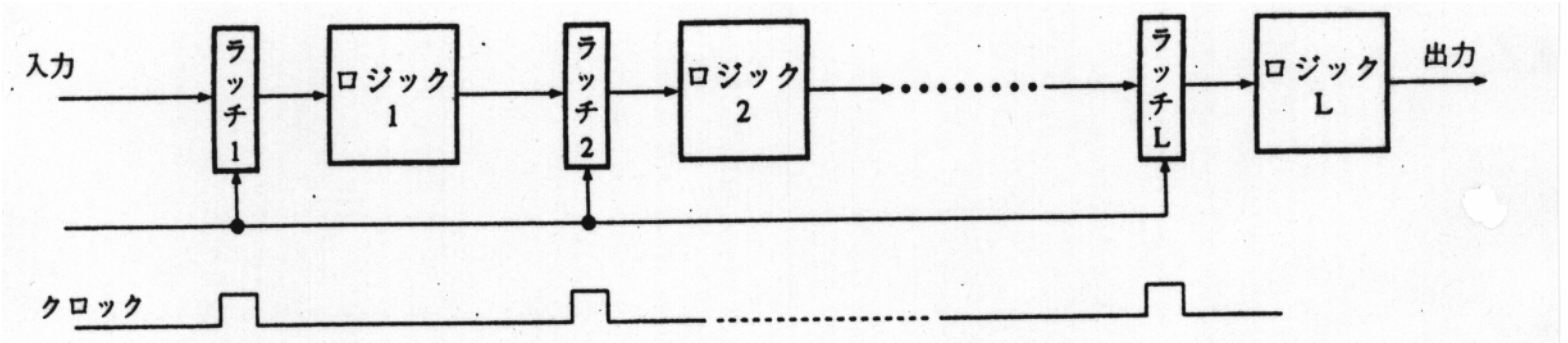
スカラ演算

$$T=N\tau_s \quad (3)$$

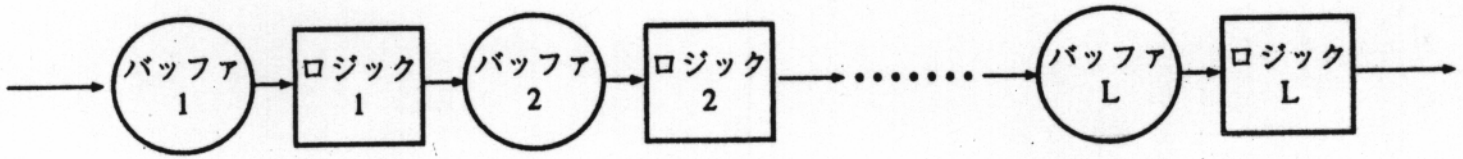
スカラ演算より高速

$$N \geqq N_{1/2}/(\tau_s/\tau_v-1) \quad (4)$$

（2）非同期式パイプライン



(a) 同期式パイプライン



(b) 非同期式パイプライン

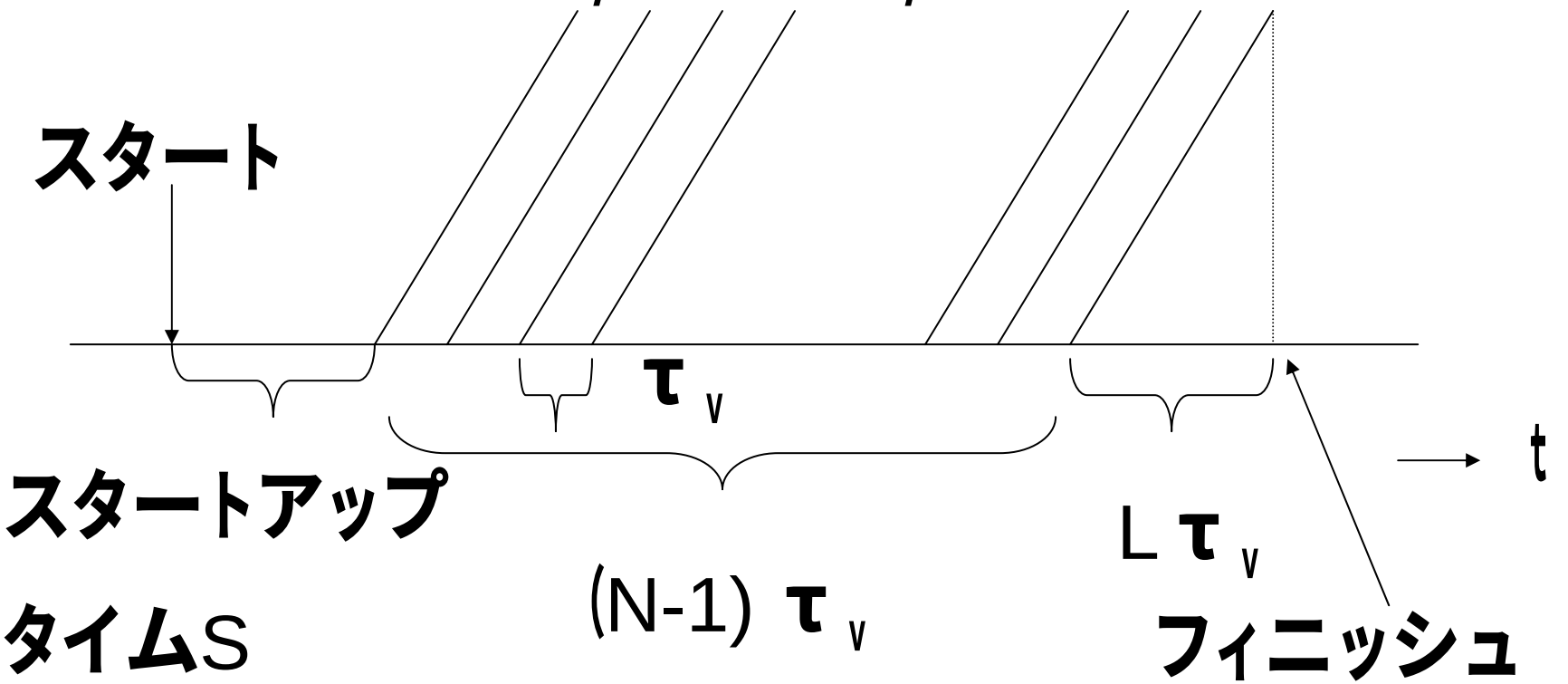
半性能長：最大性能（$1/\tau_v$）の半分の性能となるデータ要素数

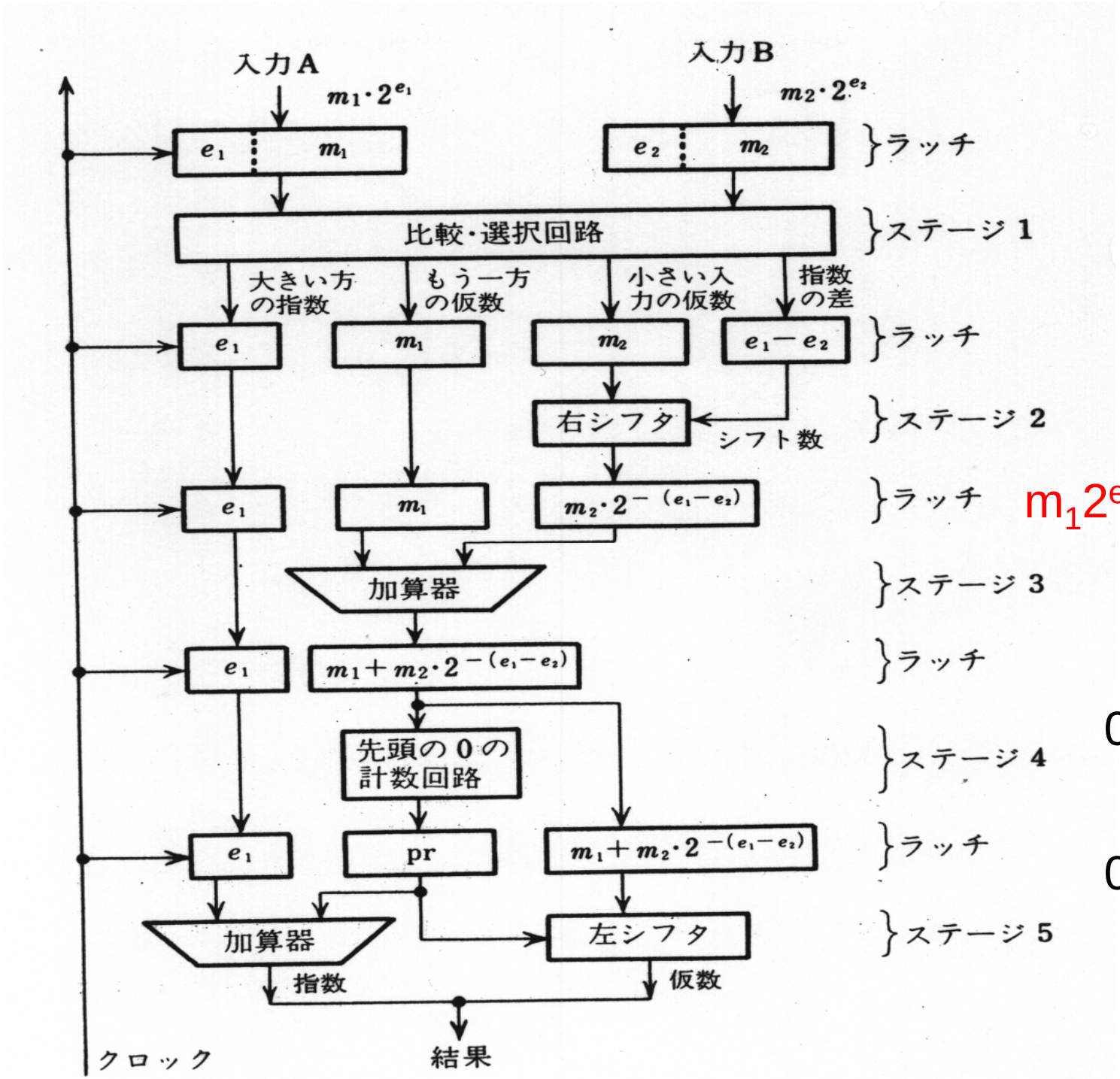
演算時間

$$T=S+(N-1+L)\tau_v$$

$$=(N+N_{1/2})\tau_v$$

$$1/(2\tau_v)=N_{1/2}/(S+(N_{1/2}-1+L)\tau_v)$$





$m_1 2^{e1} + m_2 2^{-(e1-e2)} 2^{e1}$

$0.00110..2^{10}$

$0.110\ldots\ \ 2^{8}$

４．２．１基本方式

（１）ベクトルレジスタ方式：　CRAYタイプ

（２）メモリーメモリ方式　：　CDCタイプ

CRAYタイプが席巻した理由

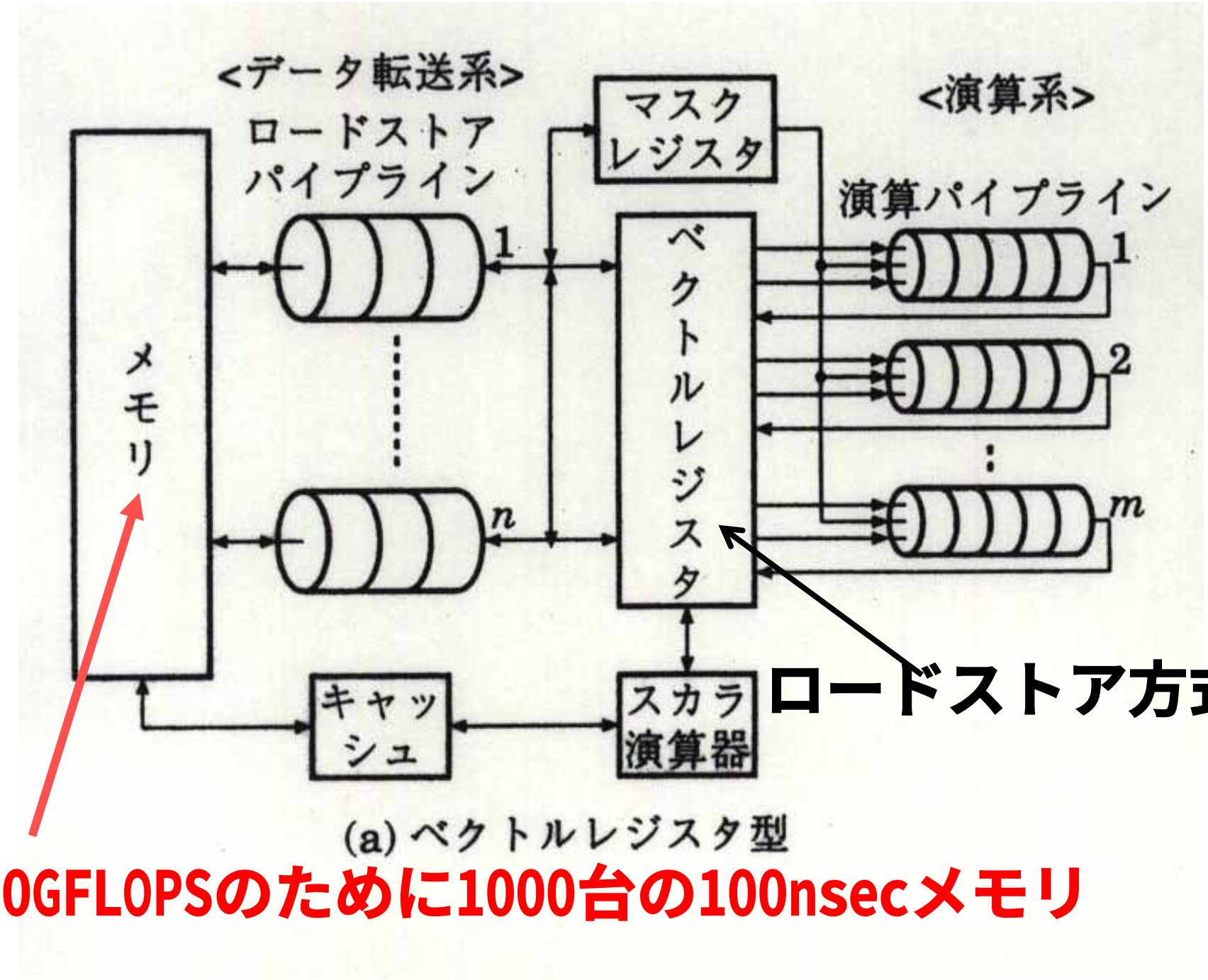
①メモリバンド幅

メモリーメモリ方式

$M=3P$　　（５）

ベクトルレジスタ方式：RISC方式

$M=P$　　（６）



10GFLOPSのために1000台の100nsecメモリ

ベクトル処理の高速化

Z(I)=X(I)*Y(I)

SET Vector Length 128

LOADV VR0 M(1000)

LOADV VR1 M(2000)

MULFV VR2 VR1 VR0

STOREV M(3000) VR2

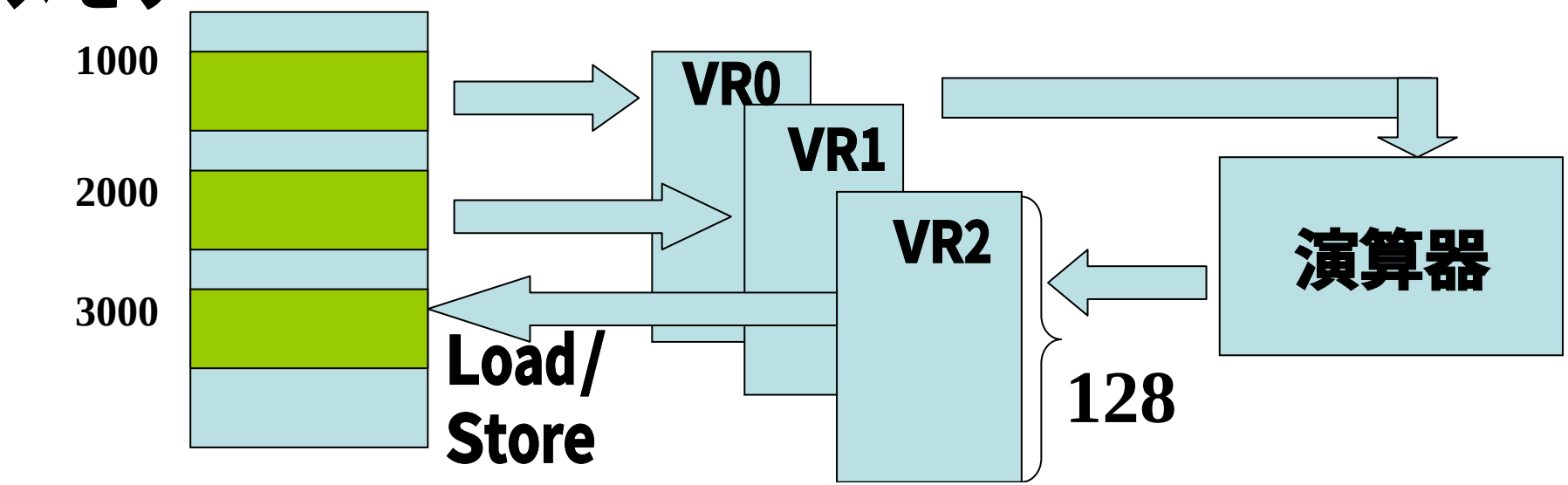


表 4.1　代表的なスーパコンピュータの諸元

機種			S-3800	VP-2600	SX-3	
マシンサイクル (nsec)			2	3.2	2.5	
最大性能 (GFLOPS)			32	5	25.6	
プロセッサ台数			4	1	4	
単一プロセッサ当りの性能	単体性能 (GFLOPS)		8	5	6.4	
	演算パイプライン	種類	加算/乗算 2台 除算 1台	加算/乗算 2台 除算 1台	加算 2台 乗算 1台	
		多重度	4 *3	4 *3	4	
	ベクトルレジスタ (KB)		128	128	144	
	ロードストアパイプライン	種類	ロード/ストア 1台 ロード 1台	ロード/ストア 2台	ロード 2台 ストア 1台	
		各パイプライン当りの最大転送速度	2GDW/sec	1.25GDW/sec	1.64GDW/sec	
		転送速度 (GDW/sec)				*1
		A(I)=B(I)	1.941	1.024	1.259	
		A(I)=B(L(I))	1.004	0.371	0.382	
		A(L(I))=1.0	0.583	0.472	0.265	
	メモリインタリーブ数		512	512	1024	
	初等関数計算性能					*2
		平方根 (メガ演算)	90.6MOPS	65.0MOPS	80.9MOPS	
		(半性能長)	181.3	131.8	105.1	
		SIN (メガ演算)	143.1MOPS	71.4MOPS	76.5MOPS	
		(半性能長)	242.0	139.5	116.9	

*1. DW/sec = 8B/sec , ベクトル長 : 2^{18}の場合 (T. Uehara, T. Tsuda : Benchmarking Vector Indirect Load / Store Instructions, Proc. WBPE, pp.16-25, 1993)

*2. MOPS = 毎秒当りの関数起動回数 (T. Nagai : Benchmarking Fortran Intrinsic Functions, 同上 , pp.26-32, 1993)

*3. 複合演算有り. S-3800 の場合, 2台の加算/乗算パイプライン, 2nsecで4多重であるので, 単体プロセッサでは4GFLOPSであるが, 加算・乗算の複合演算ができ, 8GFLOPS8となっている. VP-2600でも複合演算ができる.

②スタートアップタイムと半性能長

③データプリフェッチ

④長大ベクトル演算

４．２．４キャッシュメモリの非有効性

ベクトルレジスタの導入

①メモリに等間隔アドレスでアクセス

②必要なデータを予めプリフェッチ

キャッシュメモリの有効性

1024x1024の2次元配列Aとベクトルxの積B

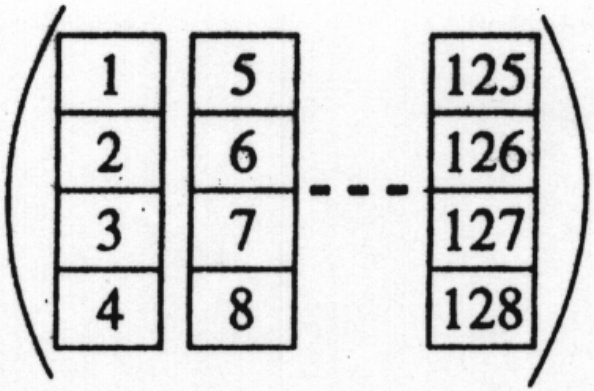
$B_i = \Sigma_j A_{ji} X_j$

各要素データは8B

キャッシュブロックのサイズは64B

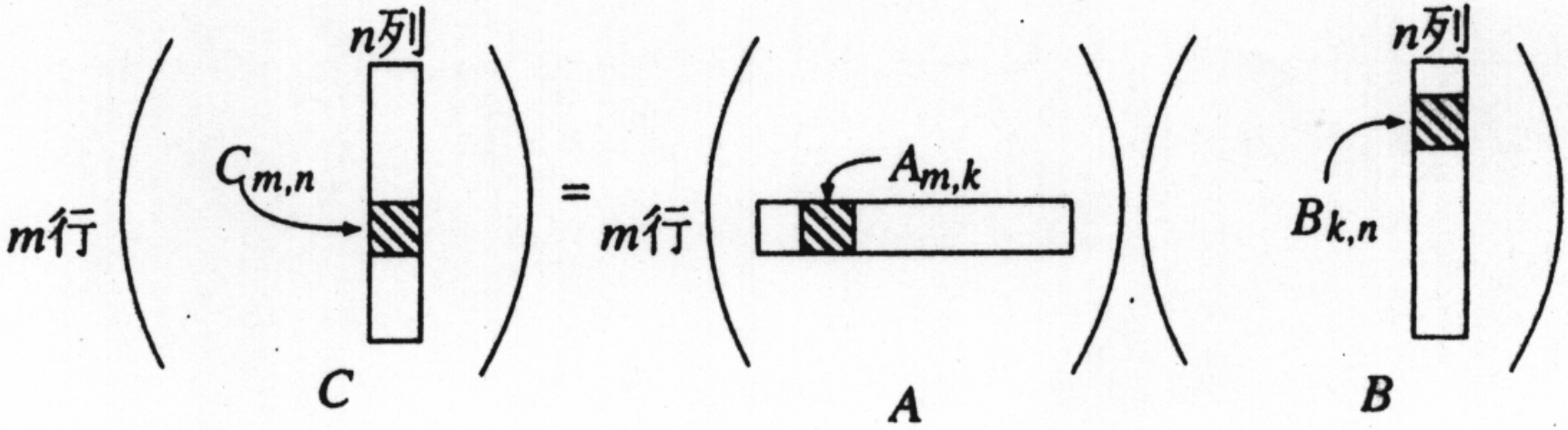
（すなわち8要素の格納が可能）

キャッシュ容量：18KB



番号:キャッシュブロック番号

(a) 列方向での行列要素写像



(b) 行列積

単純なプログラム

```
      DO 10 I=1,1024
      DO 10 J＝1, 1024
      B (I) ＝A (I, J) ＊X (J)
  10 CONTINUE
```

改良プログラム

```
    DO 10 I＝1, 1017, 8
    DO 10 K＝1, 4
    DO 10 J＝256 (K－1) ＋1, 256K
       B (I) ＝B (I) ＋A (I, J) ＊X (J)
       B (I＋1) ＝B (I＋1) ＋
                  A (I＋1, J) ＊X (J)
                      ・・・・・・
       B (I＋7) ＝B (I＋7) ＋
                  A (I＋7, J) ＊X (J)
10 CONTINUE
```

タイリング法

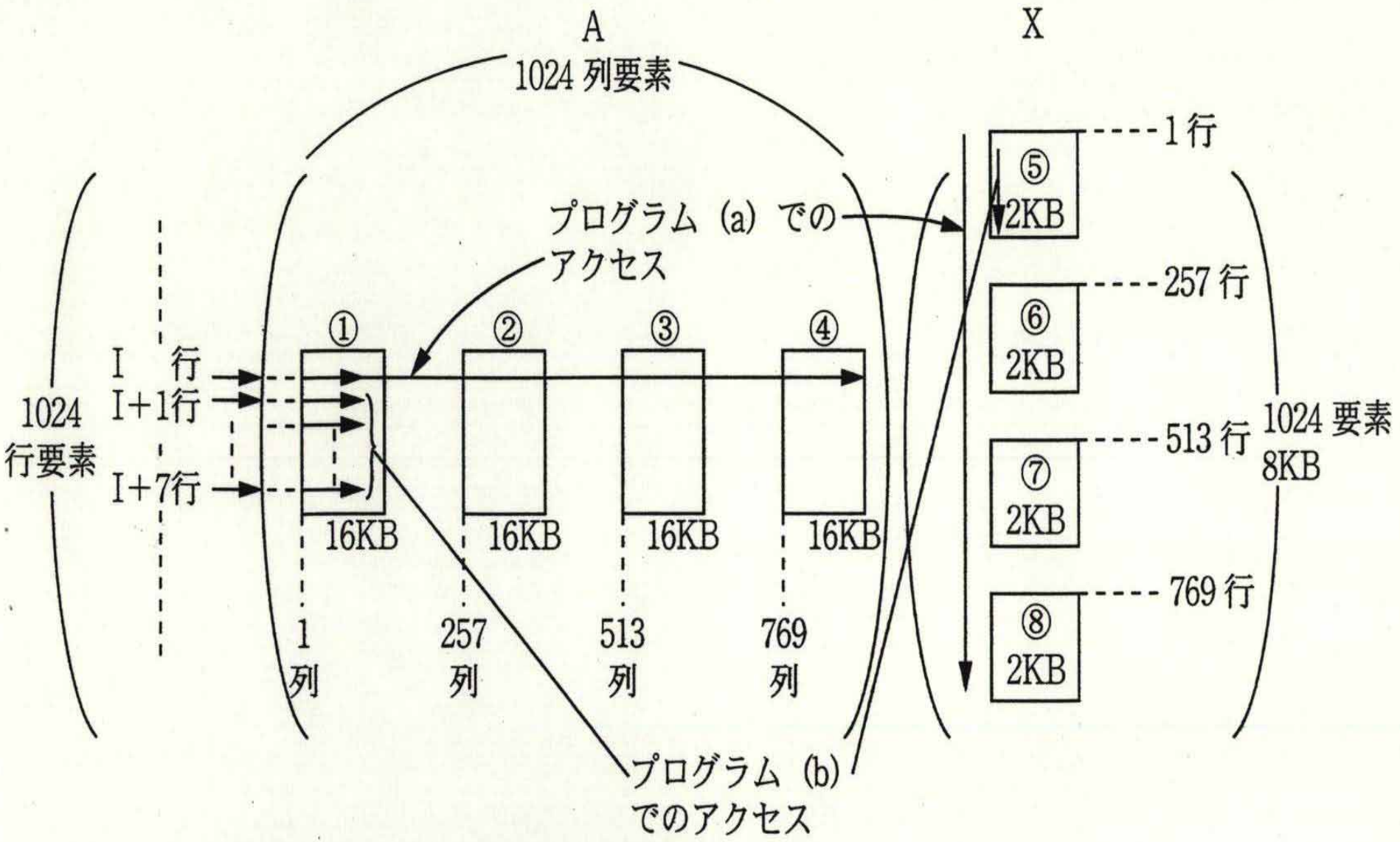


図 4.23 行列ベクトル積

４.２.５各種手法

高速化、 汎用化、 特殊化

（１）高速化手法

①マシンサイクルの短縮化

１９７６年のCRAY-1:12.5nsec

２００１年発表のSX-6:2nsec

②パイプラインの並列化

③パイプラインでの多重化

４～８多重

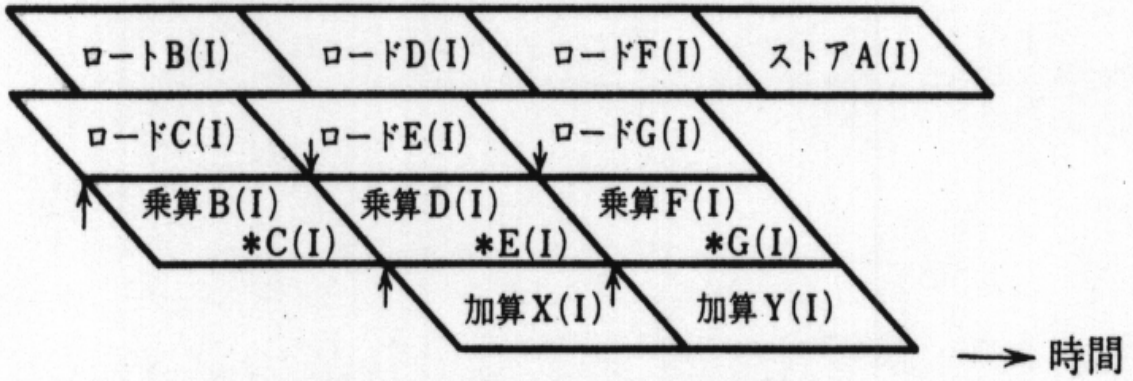
④チェイニング機能

⑤パイプラインの複合化

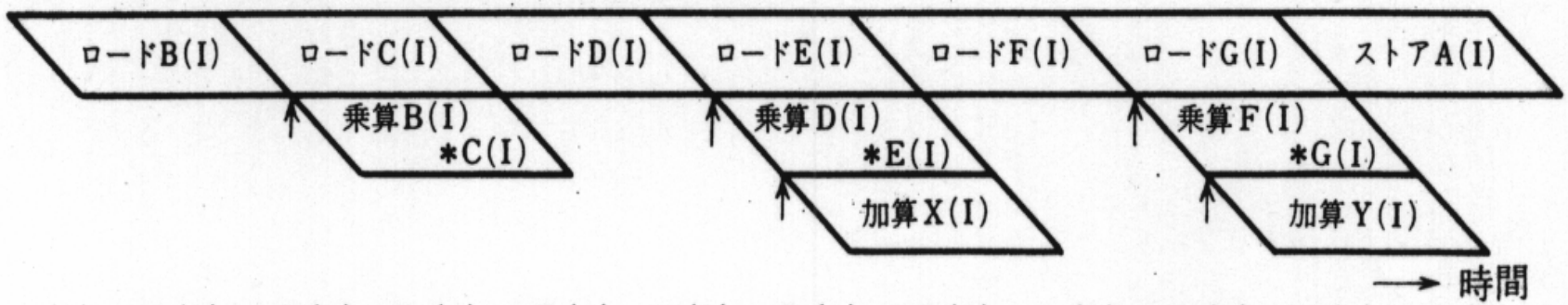
積和演算

```
DO 10 I = 1, N
10 A (I) = B (I) *C (I) + D (I) *E (I) + F (I) *G (I)
```

(a) ロード/ストアパイプラインが2本の場合



(b) ロード/ストアパイプラインが1本の場合



X(I) = B(I) *C(I) + D(I) *E(I), Y(I) = B(I) *C(I) + D(I) *E(I) + F(I) *G(I)

→：チェイニング操作の開始点

⑥スカラユニットの高速化

アムダールの法則

$K/\{\alpha+(1-\alpha)K\}$　　（７）

$\alpha=90\%$、$K=\infty$で性能向上率は１０倍

ＶＬＩＷ方式、ＲＩＳＣ方式

⑦パイプラインの共用

ベクトル化率：９０％

パイプライン稼働率：５０％

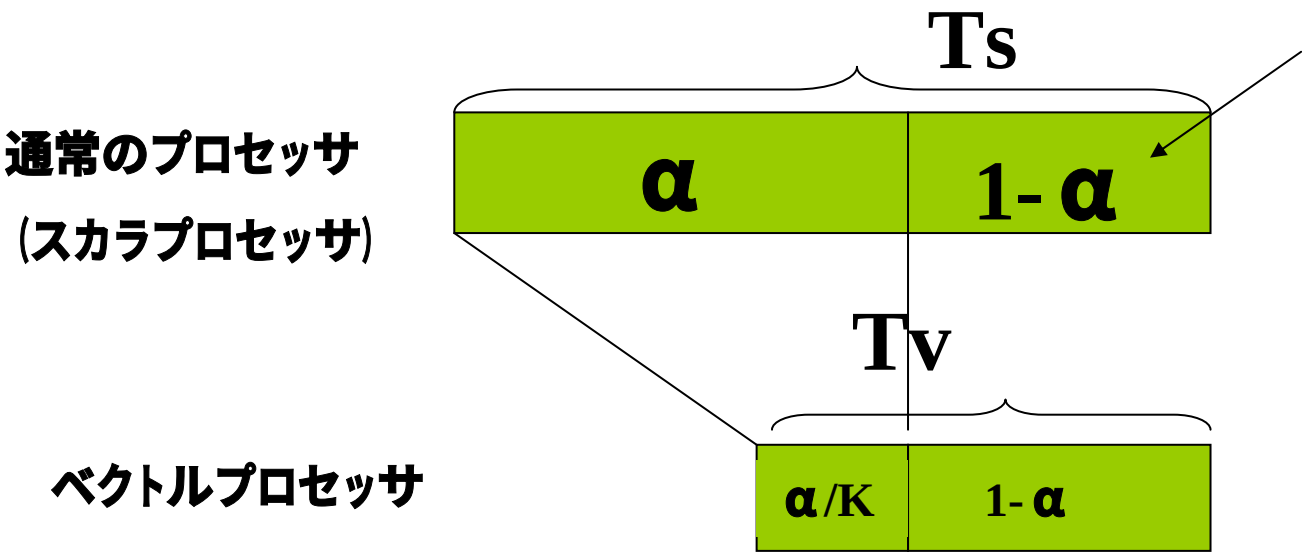
・デュアルスカラープロセッサ方式

・マシンサイクルレベルでのパイプラインの

時分割利用：漸化式などの回帰演算

7.4.2ベクトルプロセッサの泣き所：アムダールの法則

①ベクトル化率（α）が高くないとだめ

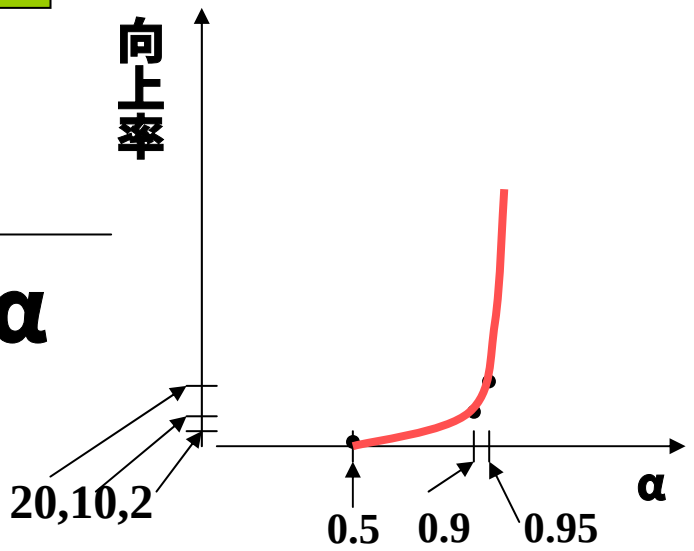


α90%で高々10倍

性能向上率

$$Ts/Tv = \frac{1}{1-\alpha+\alpha/K}$$

$$\frac{1}{1-\alpha}$$



通常のプロセッサも高速の必要：二重苦

⑧メモリの高速化

インタリーブ方式：５１２、１０２４バンク

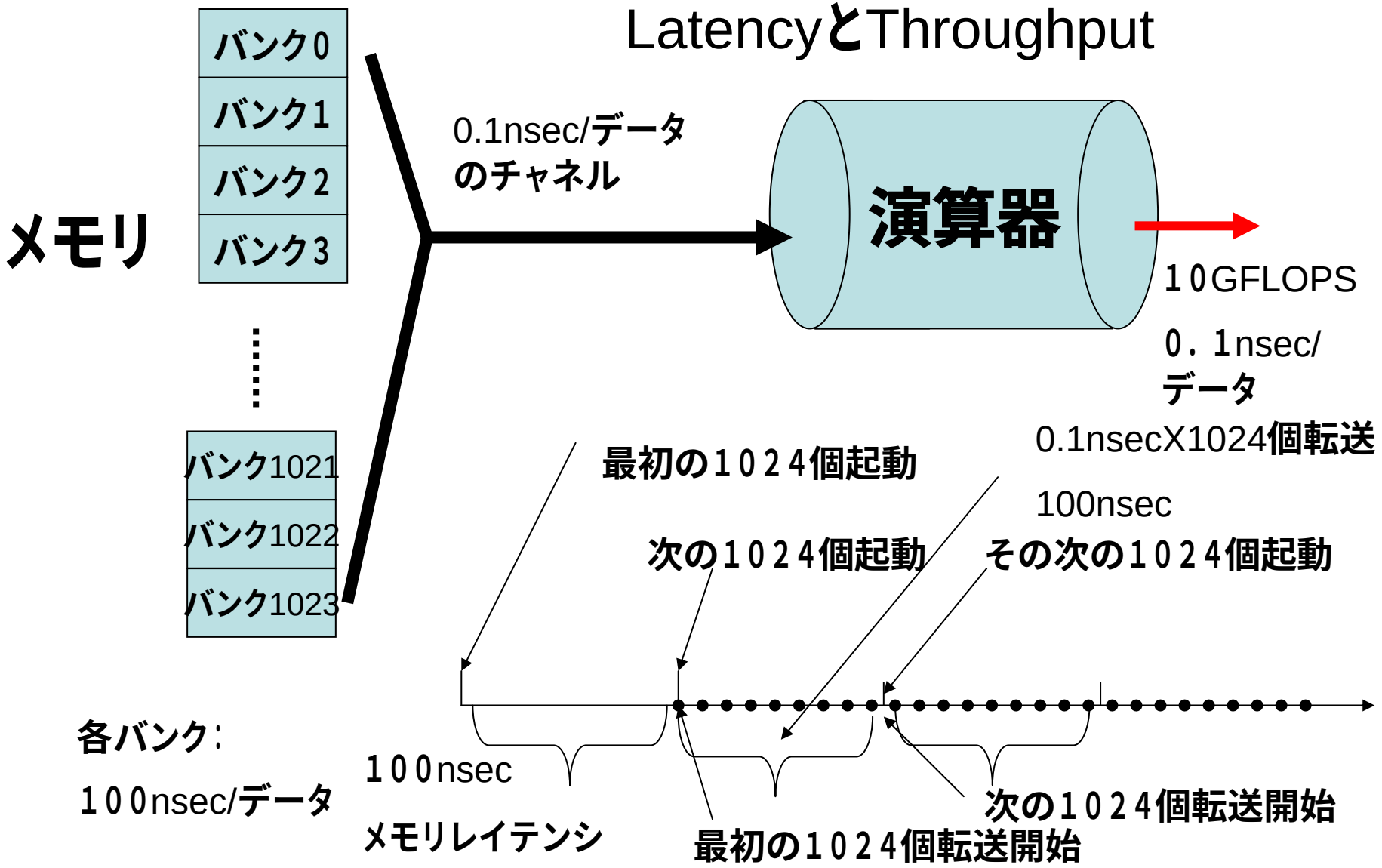
拡張メモリ

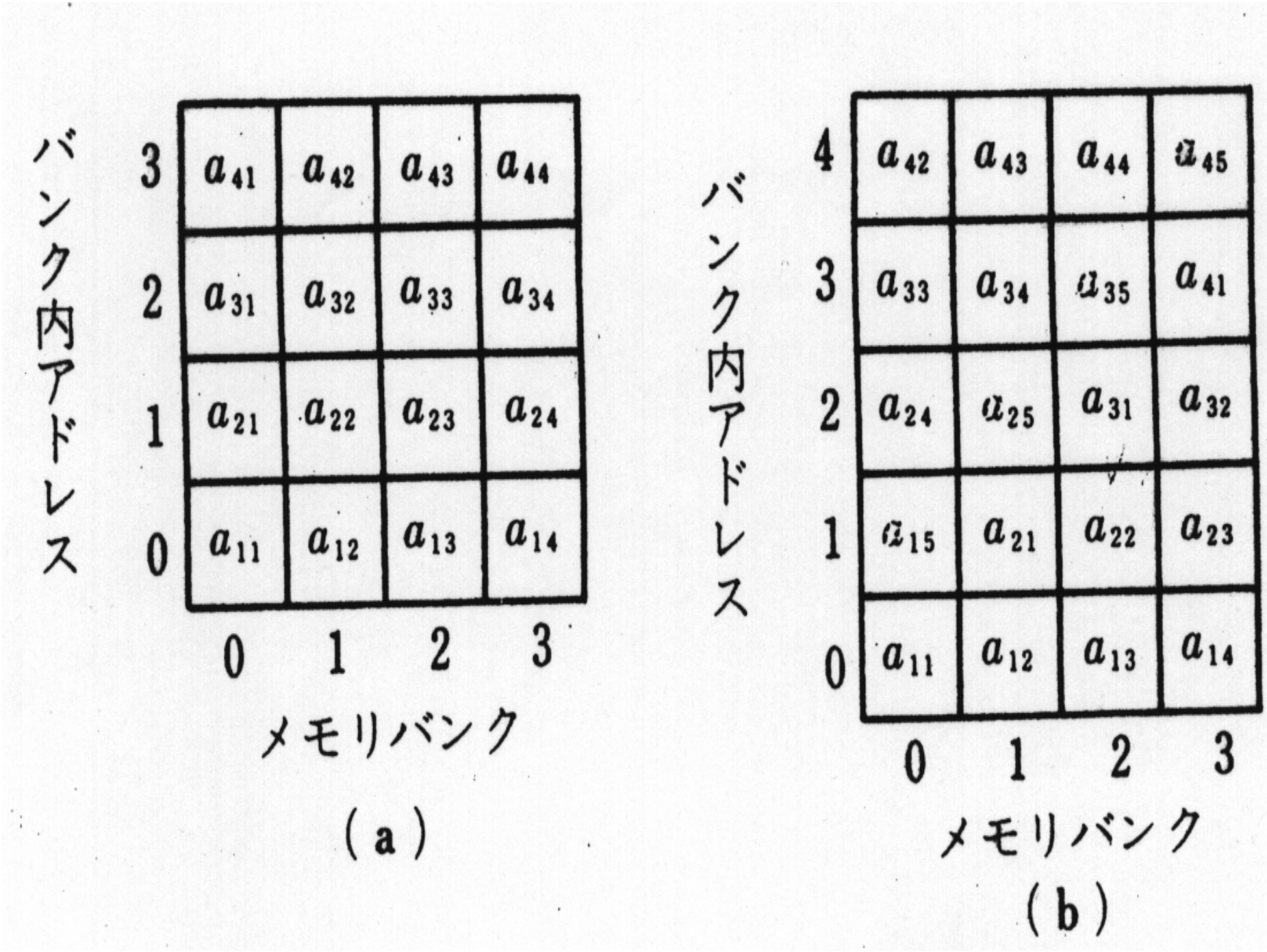
⑨マルチプロセッサ化

ノード：数PEで構成、共有メモリ化

システム：複数ノードで構成、メッセージパッシング

スーパコンピュータ：１０２４バンク構成
（ベクトルプロセッサ）




バンク内アドレス
3
2
1
0
a_{41} a_{42} a_{43} a_{44}
a_{31} a_{32} a_{33} a_{34}
a_{21} a_{22} a_{23} a_{24}
a_{11} a_{12} a_{13} a_{14}
0 1 2 3
メモリバンク
(a)
バンク内アドレス
4
3
2
1
0
a_{42} a_{43} a_{44} a_{45}
a_{33} a_{34} a_{35} a_{41}
a_{24} a_{25} a_{31} a_{32}
a_{15} a_{21} a_{22} a_{23}
a_{11} a_{12} a_{13} a_{14}
0 1 2 3
メモリバンク
(b)

（２）汎用化手法

ベクトル化支援機構

等間隔ベクトル

スキューードメモリ方式

ベクトルの収集、拡散

マスク付き演算

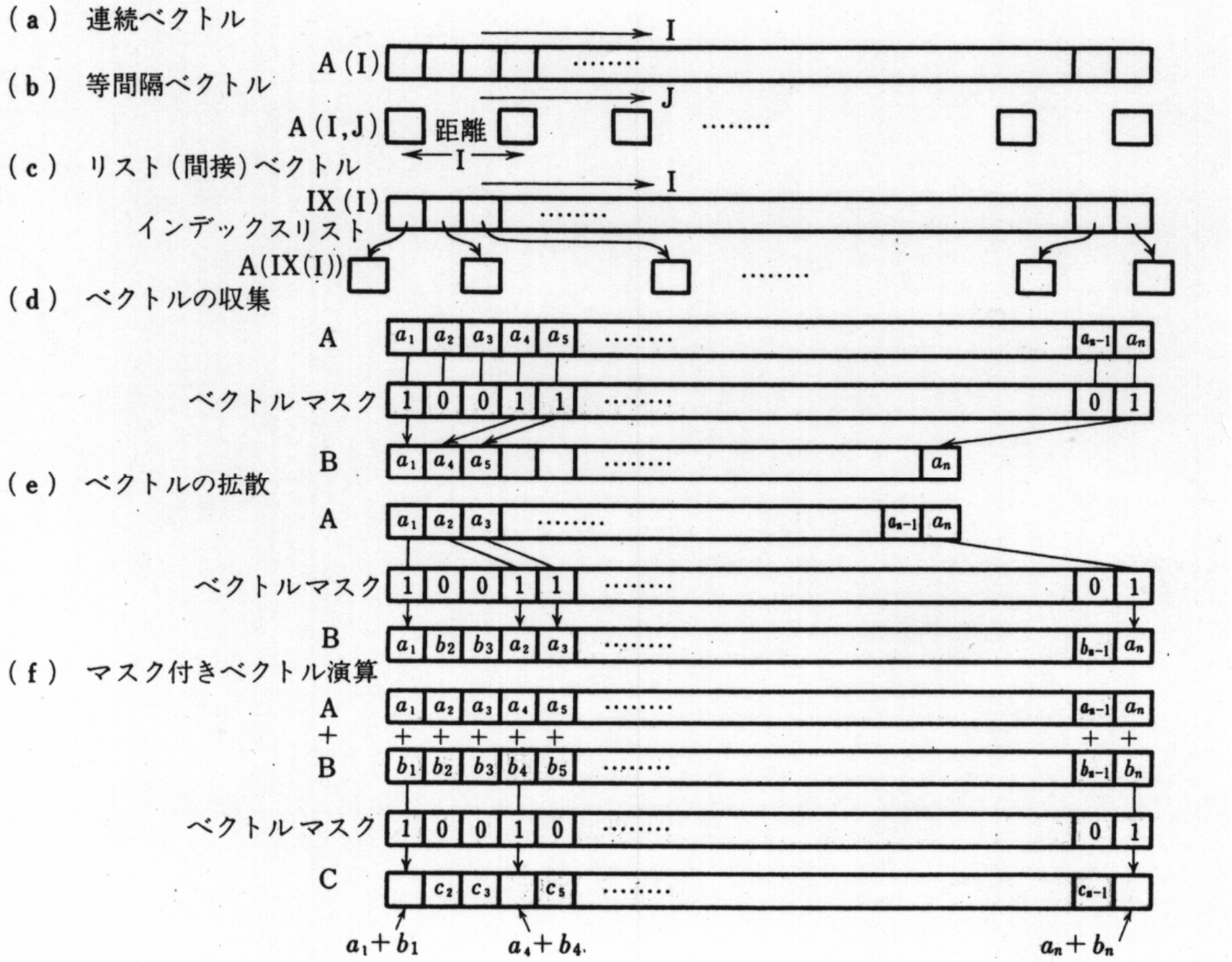
リストベクトル

・疎行列の処理

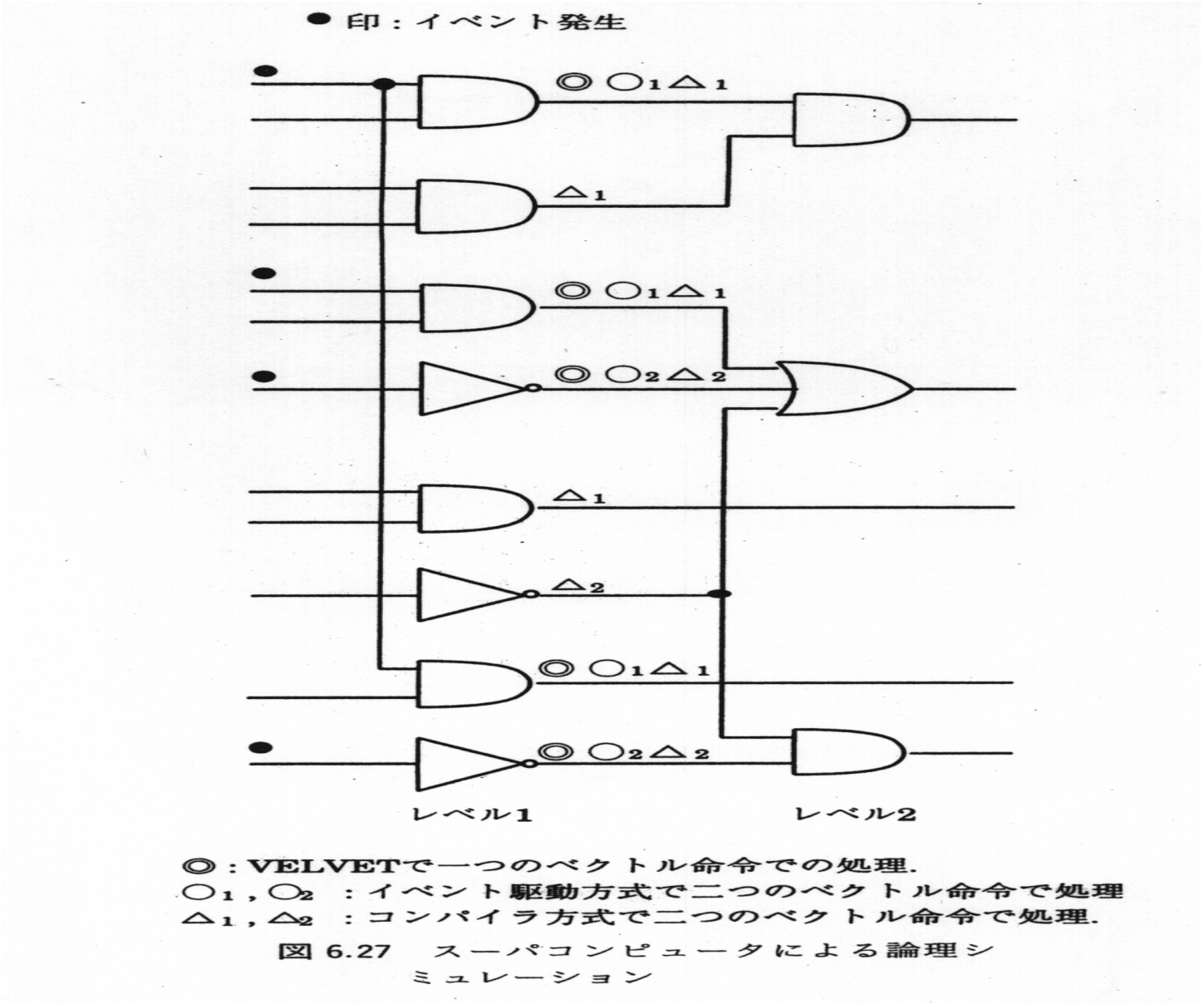
・リスト処理

・クラス分けや部分集合に対する演算

・多重ループの一重化


(a) 連続ベクトル
I
A(I)
(b) 等間隔ベクトル
J
A(I,J)
距離
I
(c) リスト(間接)ベクトル
I
IX(I)
インデックスリスト
A(IX(I))
(d) ベクトルの収集
A
ベクトルマスク
B
(e) ベクトルの拡散
A
ベクトルマスク
B
(f) マスク付きベクトル演算
A
+
B
ベクトルマスク
C
a_1+b_1
a_4+b_4
a_n+b_n

(3) 特殊化手法:論理シミュレーション



◎：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

日立：　Velvet

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

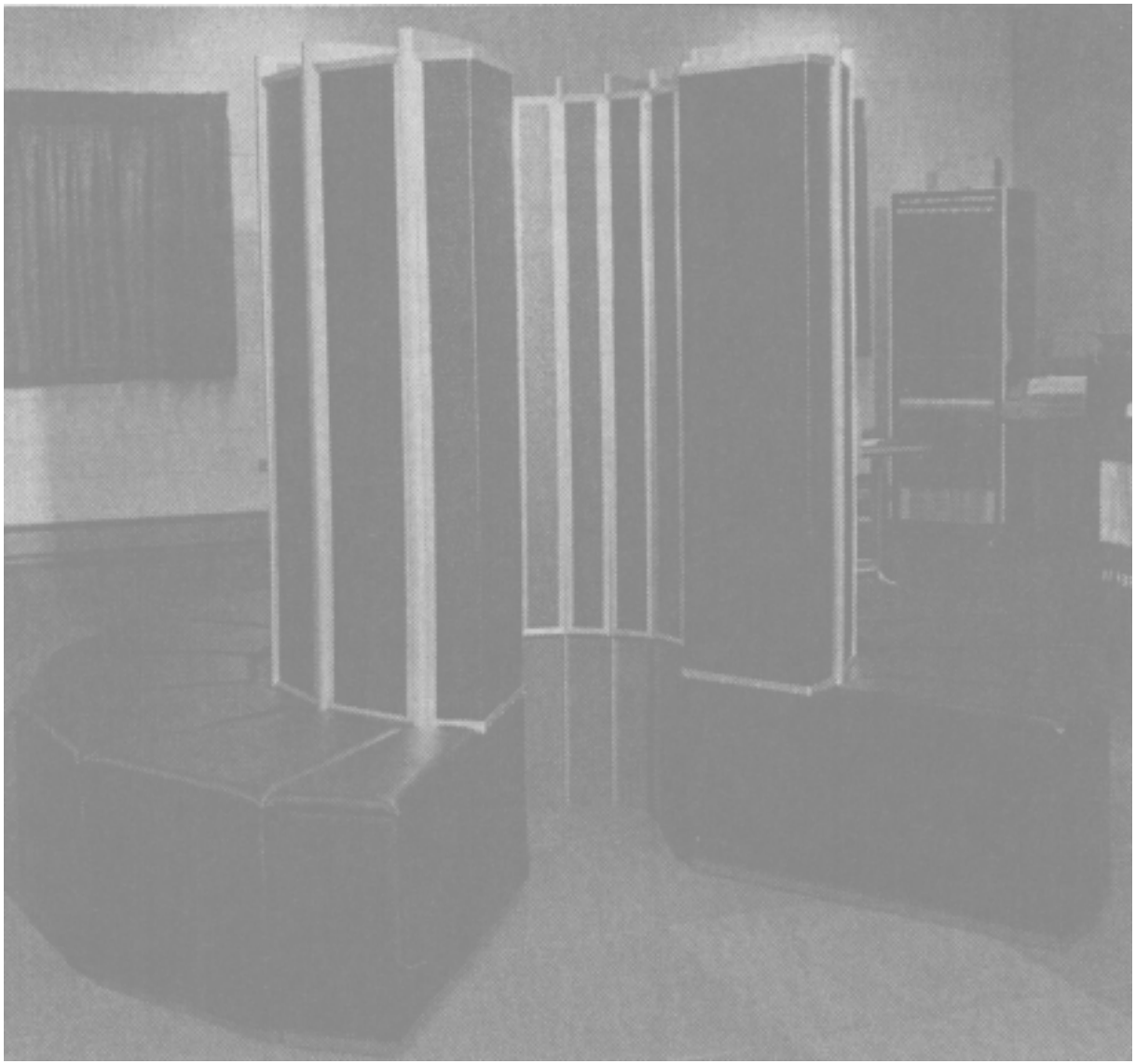
②マスクレジスタ

・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

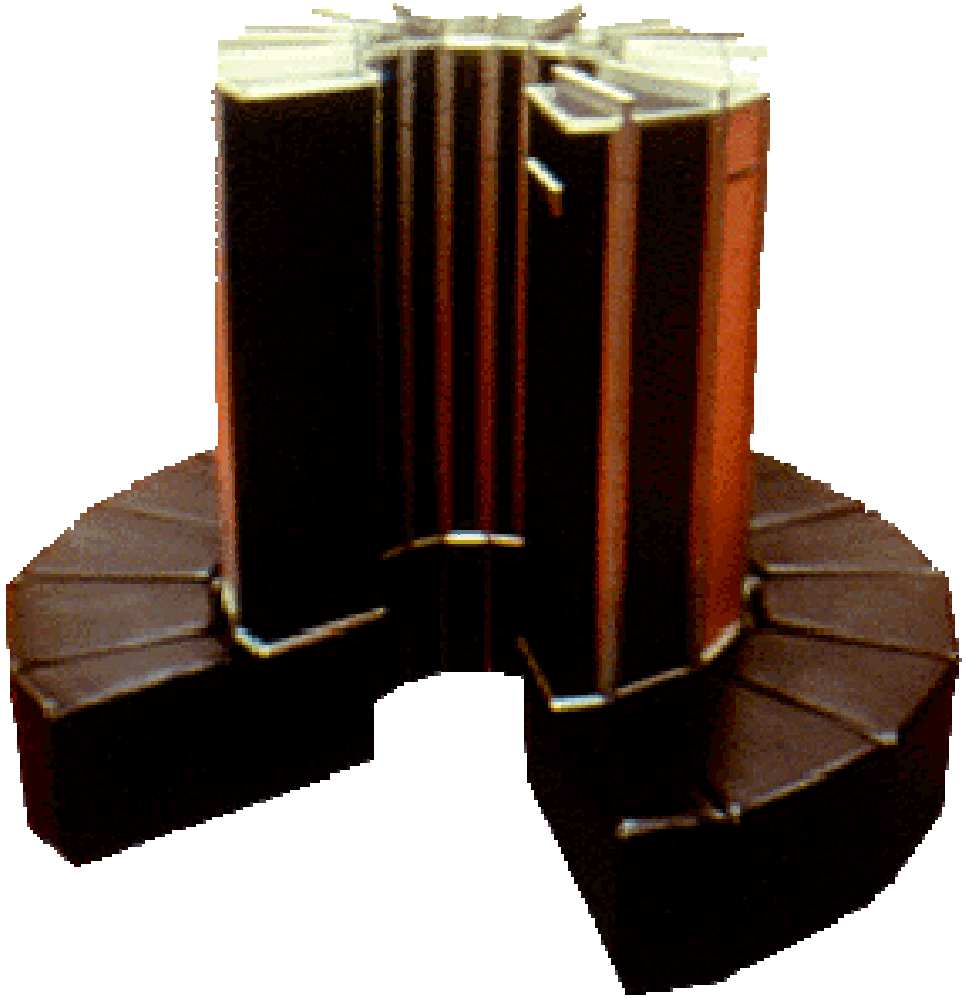
Cray-1　1976　12.5nsec　160MFLOPS



The World's Most Expensive Love-seat

CACM Vol21,No.1,1978,pp.63－72

パイオニア：Cray-1 **１９７６**、**マシンサイクル１２．５** nsec、**１６０**MFLOPS



The World's Most Expensive Love-seat

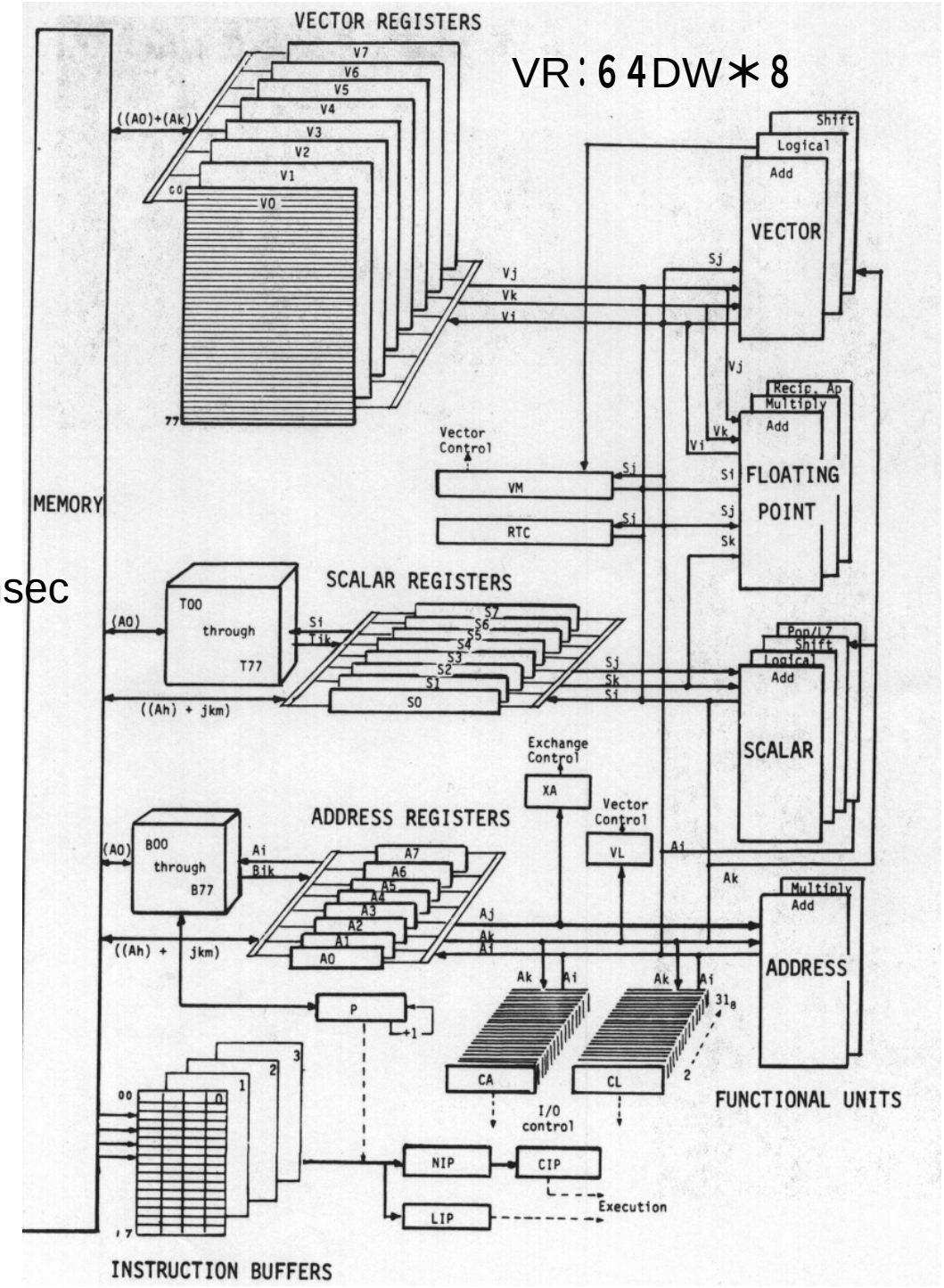
CACM Vol21,No.1,**１９７８**,**ｐｐ.６３－７２**

メモリ

8MB

16バンク

1DW/12．5nsec



1976年

12．5nsec

160MFLOPS

115KW

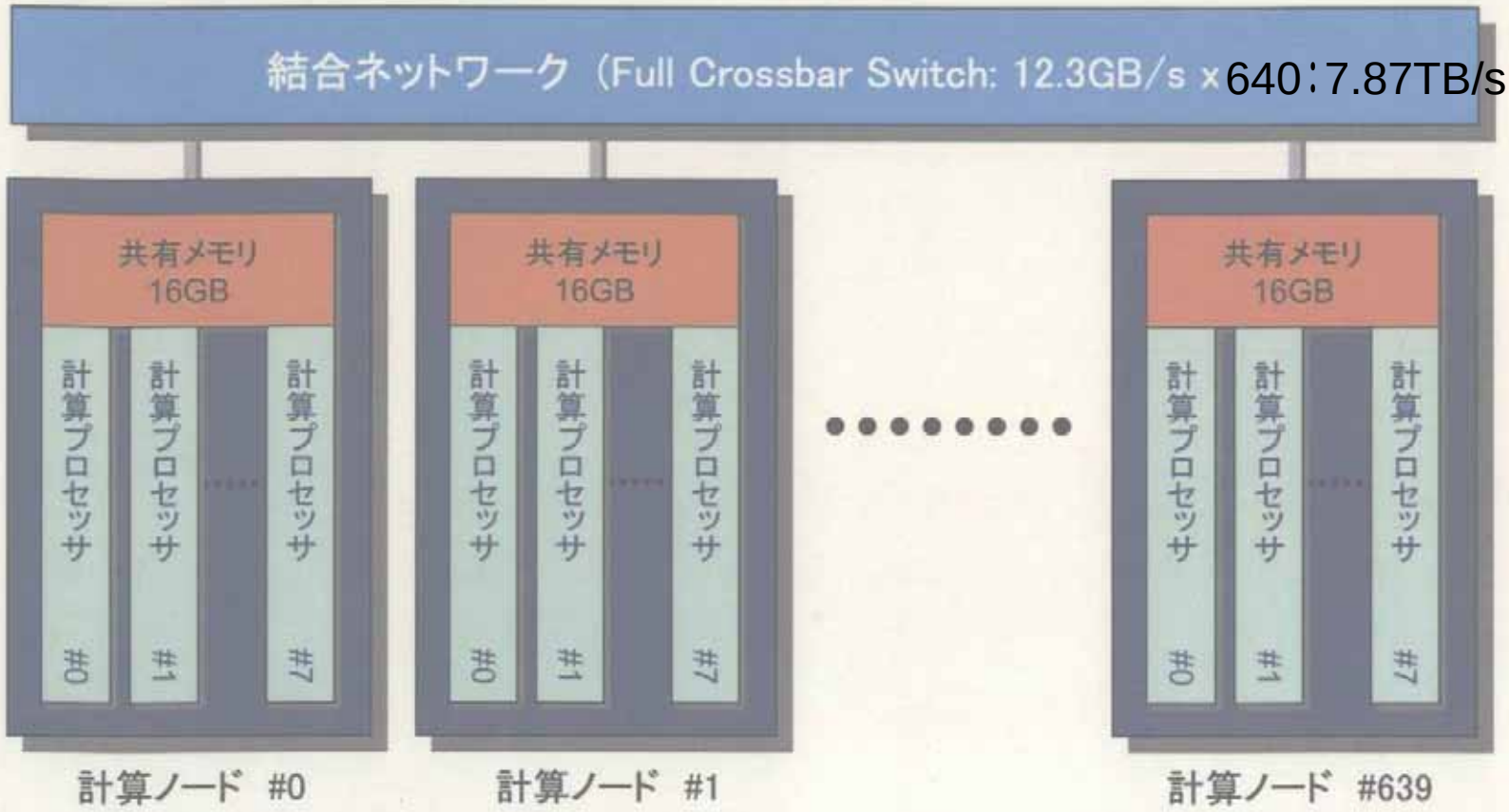
ECLロジック

NECスーパコンピュータ

- 機種　年　サイクル単体性能　最大性能　台数
- Cray-1　1976　12.5ns　160MF　160MF　1台
- SX-1/2　1984　6ns　1.3GF　1.3GF　1台
- SX-3　1989　2.9ns　5.5GF　22GF　4台
- SX-4　1994　8ns　2GF　1TF　512台
- SX-5　1998　4ns　8GF　4TF　512台
- SX-6　2001　2ns　8GF　8TF　1024台
- (CMOSシングルチップ、8PE/1ノード、最大128ノード、0.15μm)
- SX-7　2002　1.8ns　11.4GF　23TF　2048台
- (32PE/1ノード、最大64ノード、0.15μm)
- SX-8　2004　0.5ns　16GF　65TF　4096台
  (8PE/1ノード、最大512ノード、0.09μm)
- SX-9　2007　0.3ns　102.4GF　839TF　8192台
  (3.2GHz、8-16PE/1ノード、最大512ノード、0.065μm、11層銅配線)

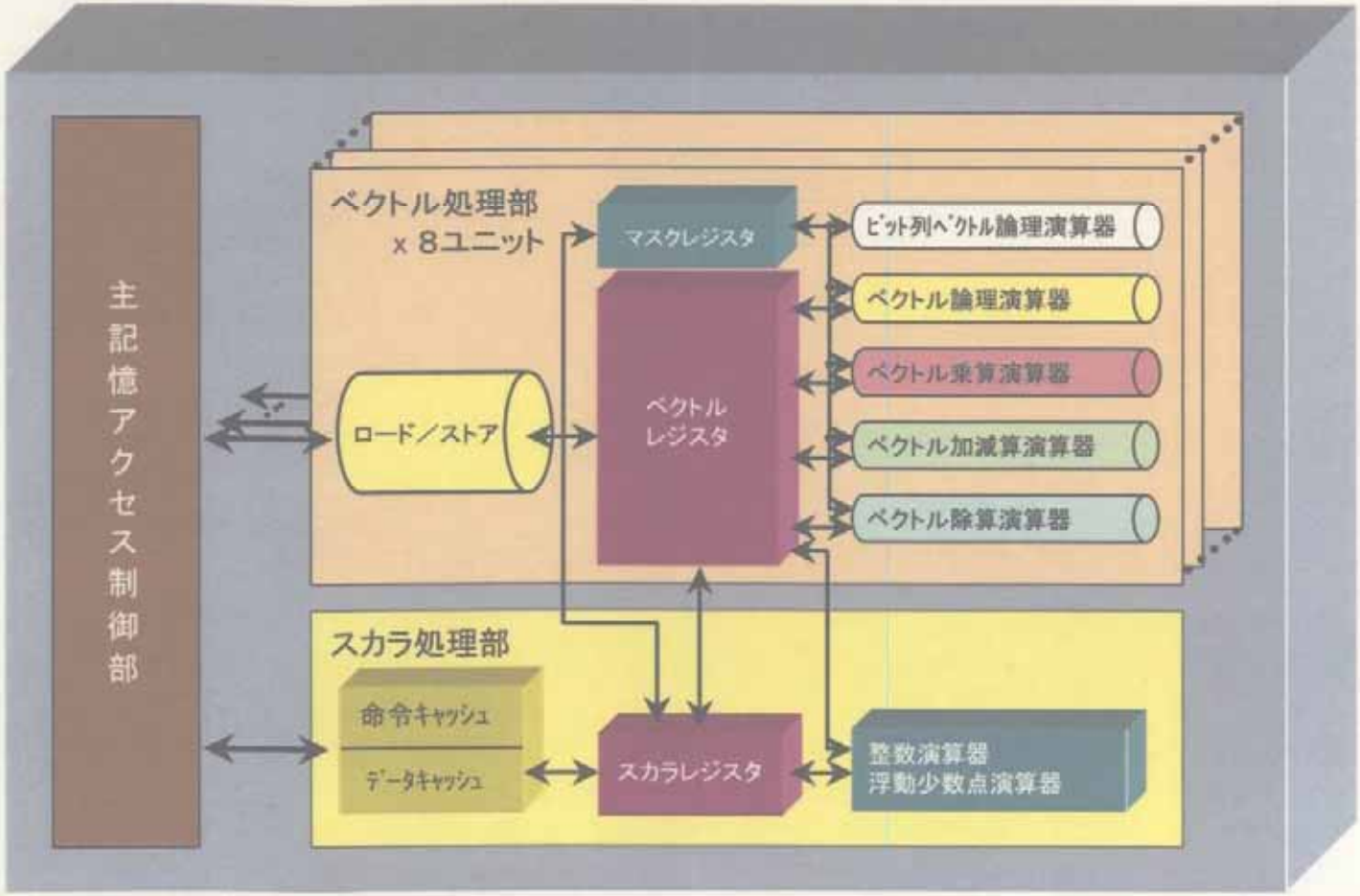
地球シミュレータの全体構成

- 総計算ノード数: 640
- ピーク性能: 40TFLOPS
- 主記憶容量: 10TB
- 総プロセッサ数 : 5120
- 計算プロセッサのピーク性能: 8GFLOPS
- 計算ノードのピーク性能: 64GFLOPS
- 計算ノードの主記憶容量: 16GB



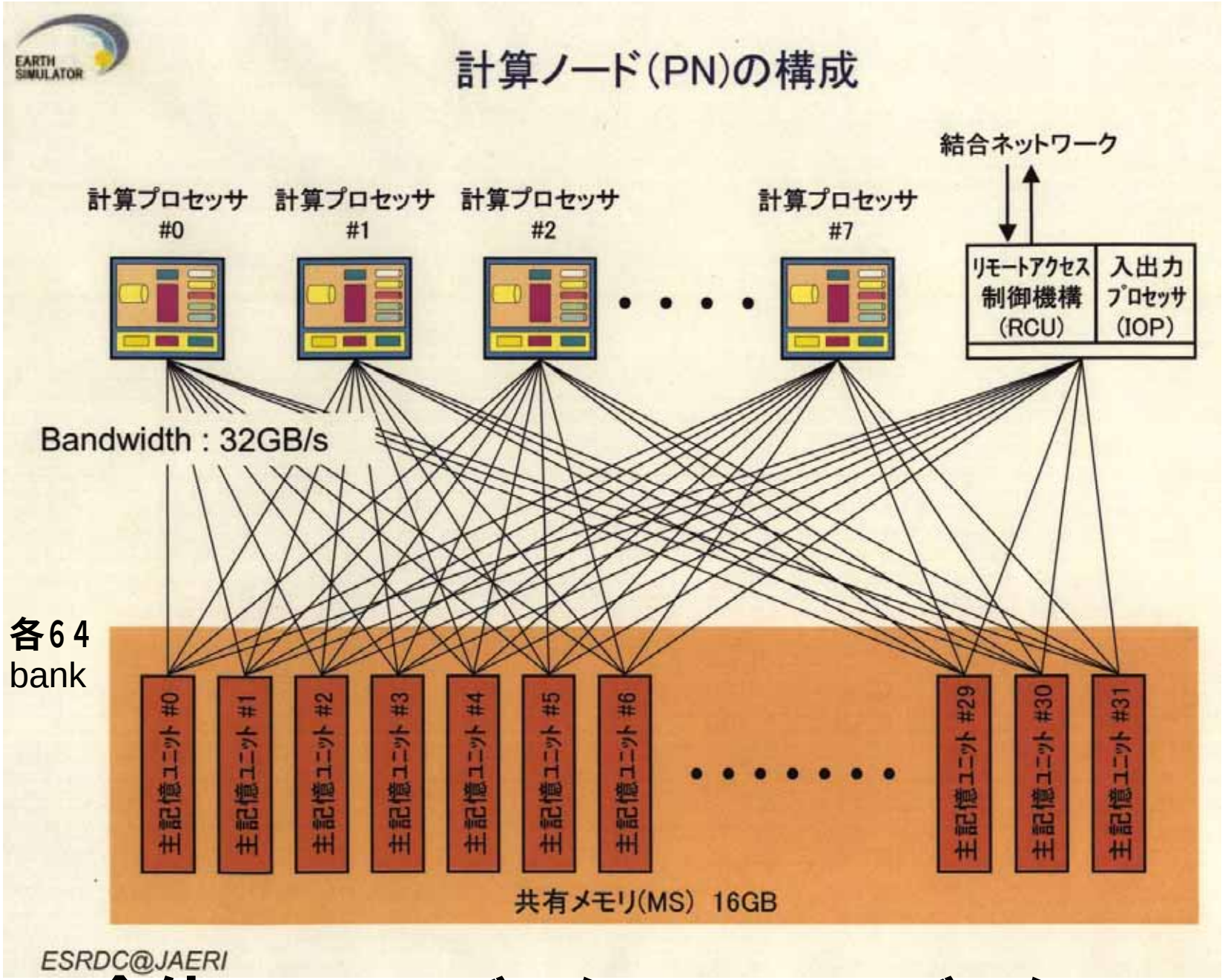
計算プロセッサ(AP)の構成

- ベクトルユニット:8セット
  - 6種のベクトルパイプライン
  - 256要素のベクトルレジスタ: 72個
  - 256ビットのマスクレジスタ: 17個
- 主記憶アクセス制御部
- スカラユニット
  - 4-ウェイ スーパースカラ
  - 64KB 命令キャッシュ
  - 64KB データキャッシュ
  - 128個の汎用レジスタ



1チップLSI: 8Gflop

- 0.15μm CMOSテクノ
  + 銅配線
- 20.79mm x 20.79mm
- 5,700万トランジスタ
- 5185 ピン
- クロック周波数
  500MHz(1GHz)
- 消費電力
  135W(Typ.)



全体で2048バンク、24nsec/バンク

４．３コンパイラ

４.３.１データ参照関係解析

ＤＯループ内にm個の文、

$S_1, S_2, \ldots, S_m$

n回$(I_1, .., I_n)$**反復**

逐次実行では、

$(S_1, I_1), (S_1, I_2), .., (S_1, I_n),$

↓ ↓ ↓

$(S_2, I_1), (S_2, I_2), .., (S_2, I_n),$

↓ ↓ ↓

・・・・・・

$(S_m, I_1), (S_m, I_2), .., (S_m, I_n)$

```
      DO 10 I 1,N
        S1
        S2
10    CONTINUE
          ↓
      DO 10 I 1,N
        S1
10    CONTINUE
      DO 20 I 1,N
        S2
20    CONTINUE
```

ベクトル実行では、

$(S_1, I_1) \rightarrow (S_1, I_2) \rightarrow , , \rightarrow (S_1, I_n)$

$(S_2, I_1) \rightarrow (S_2, I_2) \rightarrow , , \rightarrow (S_2, I_n)$

...

$(S_m, I_1) \rightarrow (S_m, I_2) \rightarrow , , \rightarrow (S_m, I_n)$

次プログラム：ベクトル化可能

```
      DO 20 I=1,N
      C(I)=A(I)+B(I) S1                    （8）
      A(I)=C(I)-D(I) S2
20 CONTINUE
```

データ参照関係

逐次処理のとき、

C(1)=A(1)+B(1)

A(1)=C(1)-D(1)

C(2)=A(2)+B(2)　(9)

A(2)=C(2)-D(2)

ベクトル処理のとき

```
C(1)=A(1)+B(1)
C(2)=A(2)+B(2)
      ---                          (10)
A(1)=C(1)-D(1)
A(2)=C(2)-D(2)
```

次プログラム：ベクトル化不可能

```
   DO 20 I=1,N
   C(I)=A(I)+B(I)   S1
   A(I+1)=C(I)-D(I)S2
20 CONTINUE
```

データ参照関係

逐次処理のとき、

```
C(1)=A(1)+B(1)
A(2)=C(1)-D(1)    (12)
C(2)=A(2)+B(2)
A(3)=C(2)-D(2)
```

ベクトル処理のとき

```
C(1)=A(1)+B(1)
C(2)=A(2)+B(2)
      ---                    (13)
A(2)=C(1)-D(1)
A(3)=C(2)-D(2)
```

次プログラム：Ｓ１とＳ２が逆とすると、

ベクトル化可能

```
      DO 20 I=1,N
      C(I)=A(I)+B(I) S1(14)
      E(I)=C(I+1)-D(I) S2
   20 CONTINUE
```

（１）データ依存

①フロー依存

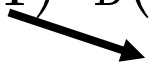
②出力依存

リネーミング（renaming）

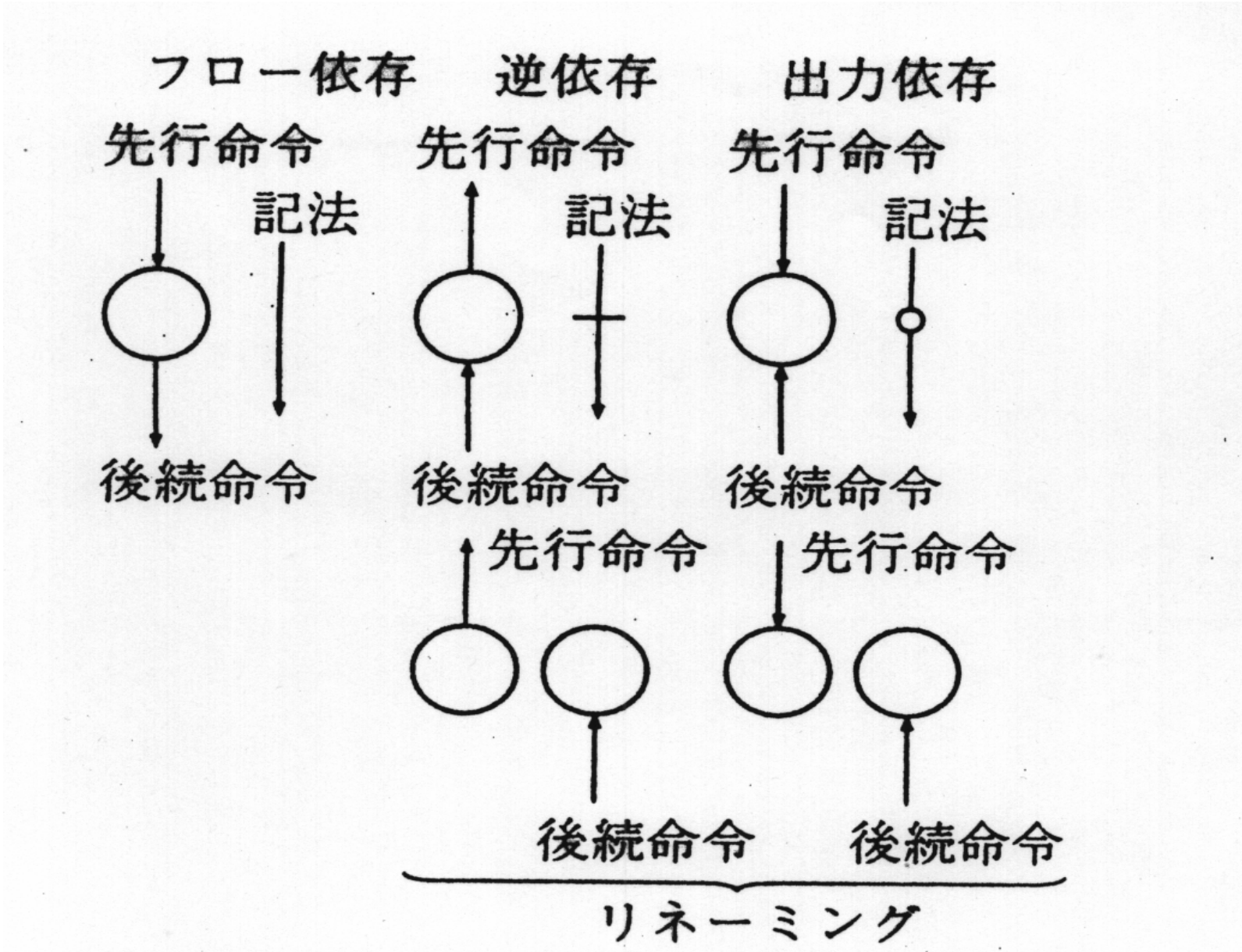
③逆依存

（ａ）同一反復内でのデータ依存（ループ独立依存）

```
   DO 10 I=1,N
   A(I)=B(I)+C(I)   S1
   D(I)=A(I)+B(I)   S2
10 CONTINUE
```



Ａ：ループ独立のフロー依存


フロー依存
先行命令
記法
後続命令
逆依存
先行命令
記法
後続命令
出力依存
先行命令
記法
後続命令
先行命令
先行命令
後続命令
後続命令
リネーミング
○：同一レジスタや同一メモリアドレス

（ｂ）異なった反復間でのデータ依存

（ループ運搬依存）

```
   DO 10 I=1,N
   A(I+1)=B(I)+C(I)   S1
   D(I)=A(I)+B(I)     S2
10CONTINUE
```

反復Ｉの Ｓ１で書込まれたデータ：

反復（Ｉ＋１）のＳ２で使用

Ａにループ運搬のフロー依存

```
   DO 10 I=1,N
   A(I-1)=B(I)+C(I)   S1
   D(I)=A(I)+B(I)     S2
10CONTINUE
```

反復ＩでのＳ１の書込みは

**　反復（Ｉ－１）のＳ２の読出し後**

ループ運搬の逆依存

（２）データ依存関係と添字式

（ａ）Diophantine方程式

```
  DO 10 I=1,N
  A(I+N+1)=B(I)+C(I) S1
  D(I)=A(I)+B(I)     S2
10CONTINUE
```

Ｓ１でのＡの書込み領域：

Ｓ２でのＡの読み込み領域は重ならない

フロー依存なし

$M_{S1}=I+N+1$

$M_{S2}=I$ (19)

$M_{S1}(I1)=M_{S2}(I2)$

```
      DO 10 I=1,20
      A(9*I+16)=B(I)+C(I)S1
      D(I)=A(6*I-11)+B(I)S2
10CONTINUE
```

$9*I1+16=6*I2-11$

（ｂ）Banerjee法など、最大公約数法

(I1、I2)=(1、15)、(3、18)解

（３）ベクトル化を可能とするＤＯループの変換

依存グラフ上に依存関係が全くない場合

単方向の依存関係のみ存在する場合

ベクトル化が可能

依存グラフにループ構造が存在

ベクトル化は一般的には不可能

①ループ分割（loop distribution）

```
      DO 20 I=1,N
      C(I)=A(I)+B(I)      S1
      A(I)=C(I)-D(I)      S2
      G(I)=E(I)+F(I)      S3
      E(I+1)=G(I)-H(I)    S4
   20 CONTINUE
```

②添字の分割

B(添字小)：S2で書かれた後S1で読み出し

B（添字大）：S1で呼び出された後S2で書き込まれる

```
      DO 10 I=1,100
      A(I)=B(100-I)+T     S1
      B(I)=C(I)           S2
   10 CONTINUE
```

添字を分割して、

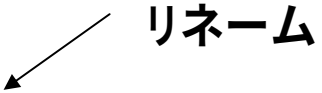
```
      DO 10 I=1,50
      A(I)=B(100-I)+T    S1
      B(I)=C(I)          S2
10    CONTINUE

      DO 20 I=51,100
      A(I)=B(100-I)+T    S1
      B(I)=C(I)          S2
20    CONTINUE
```

③ノードスプリット

```
      DO 10 I=1,N
      A(I)=B(I)+C(I)        S1
      D(I)=A(I)+A(I+1)      S2
10 CONTINUE
```

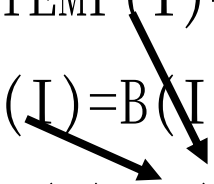
Ｓ１からＳ２にＡ（Ｉ）でフロー依存、

Ｓ２からＳ１にＡ（Ｉ＋１）で逆依存




```
T(I)=B(I)+C(I)
D(I)=T(I)+A(I+1)
A(I)=T(I)
```

```
      DO 10 I=1,N
      TEMP(I)=A(I+1)              S0
    A(I)=B(I)+C(I)        S1
      D(I)=A(I)+TEMP(I)   S2
10 CONTINUE
```



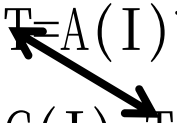
Ｓ０からＳ２へのフロー依存、

Ｓ１からＳ２へのフロー依存

ループ消滅

④スカラエクスパンジョン

```
    DO 10 I=1,N
    T=A(I)*B(I)       S1
    C(I)=T+B(I)       S2
10  CONTINUE
```

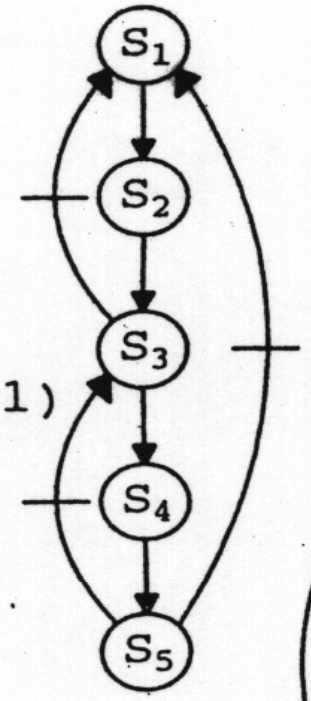


Ｓ１からＳ２へのＴに関するフロー依存、

Ｓ２からＳ１へのＴに関する逆依存

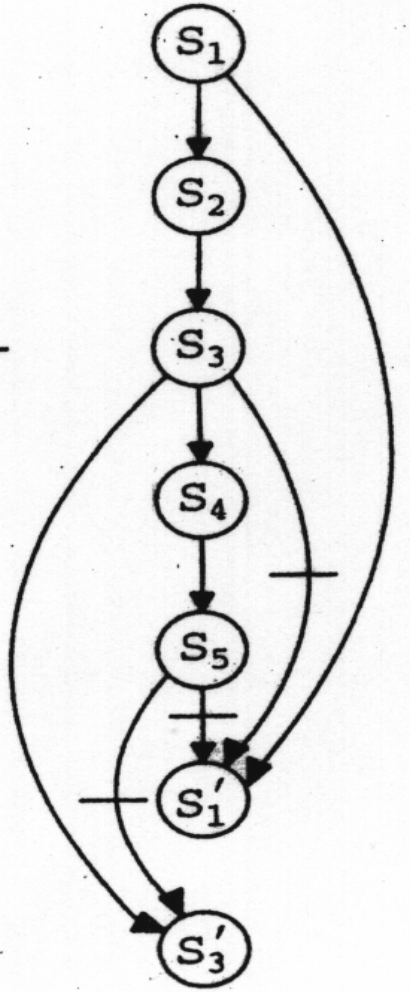
```
    DO 10 I=1,N
    TEMP(I)=A(I)*B(I) S1
    C(I)=TEMP(I)+B(I) S2
10  CONTINUE
```

```
      DO 10 I=1, N
S1        A(I)=B(I)+C(I)
S2        D(I)=A(I)+1.0
S3        E(I)=D(I)+A(I+1)
S4        F(I)=E(I)+1.0
S5        G(I)=F(I)+E(I+1)+A(I+1)
  10  CONTINUE
```



(a)

```
      DO 10 I=1, N
S1        T1(I)=B(I)+C(I)
S2        D(I)=T1(I)+1.0
S3        T2(I)=D(I)+A(I+1)
S4        F(I)=T2(I)+1.0
S5        G(I)=F(I)+E(I+1)+A(I+1)
S1'       A(I)=T1(I)
S3'       E(I)=T2(I)
  10  CONTINUE
```



(b)

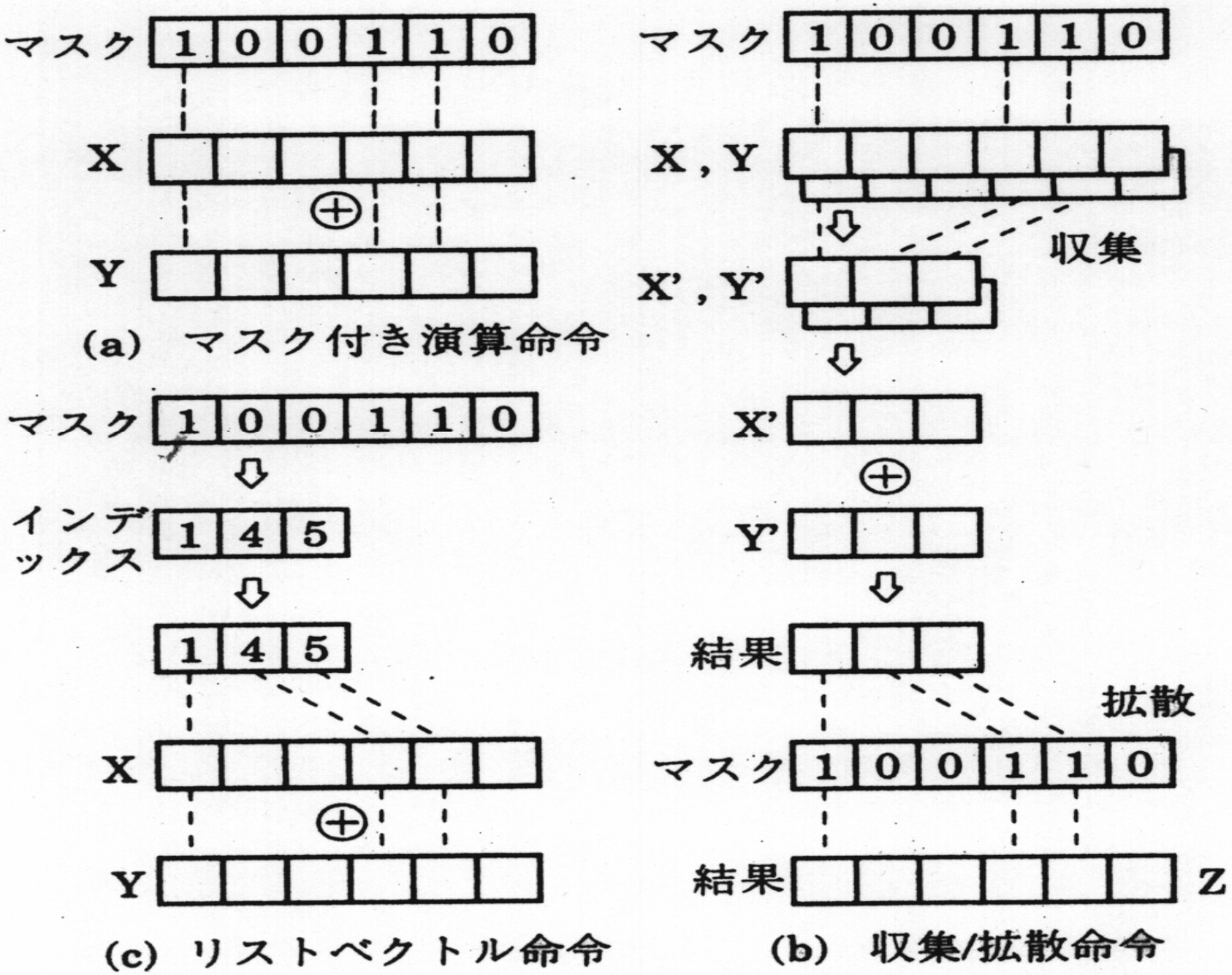
４．３．２制御依存

ＩＦ文を含むＤＯループのベクトル化

マスク付き演算、収集拡散、リストベクトルなど

```
      DO  10 I  1,N
        IF A(I)=B(I)
        Z(I)=X(I)+Y(I)
   10 RETURN
```

```
      DO 10 I=1,N
      IF (A(I).EQ.B(I)) Z(I)=X(I)+Y(I)
   10 CONTINUE
```



マスク ： A(I) . EQ. B(I)の時1

4.3.3マクロ演算のベクトル化
フロー依存のサイクルがある場合

```
      DO 20 I=1,N
       C(I)=A(I)+B(I)       S1
       A(I+1)=C(I)-D(I)     S2
 20 CONTINUE
```

(1) マクロ演算の種類

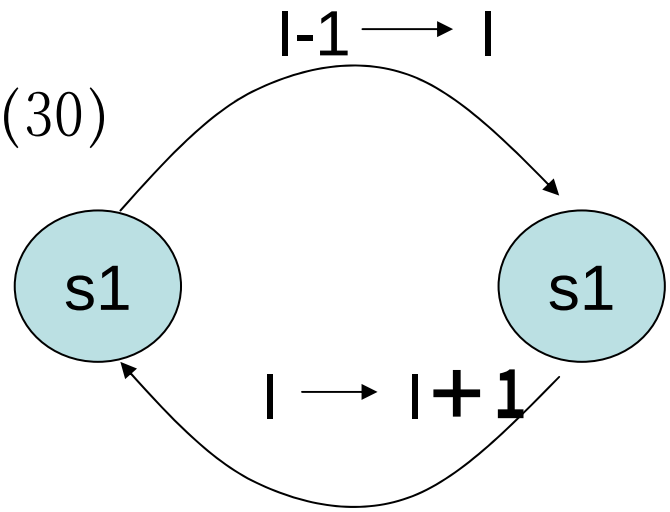
- **内積型演算**($S=S+A_i \times B_i$)
- **総和型演算**($S=S+A_i$)
- **最大値最小値演算**
- **漸化式**($X_i=A_i+X_{i-1} \times B_i$)

総和型演算

```
      S=0
      DO 10 I=1,N          (29)
      S=S+A(I)          S1
10 CONTINUE
```

部分和（プレフィックス演算）

```
      S(0)=0
      DO 10 I=1,N                (30)
      S(I)=S(I-1)+A(I) S1
10 CONTINUE
```



（ａ）総和演算器のないコンピュータ

$\log N$**回のベクトル演算**

N**個のデータの隣りあうデータ同士の加算**

（ベクトル長$N/2$**）**

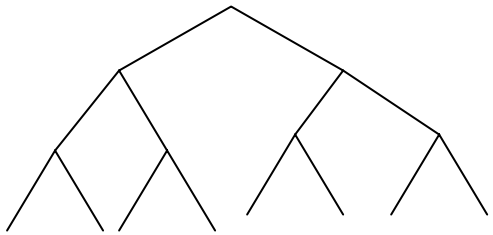
その結果の隣あうデータ同士の加算

（ベクトル長$N/4$**）**

$$T=\tau(N/2+N_{1/2})+\tau(N/4+N_{1/2})+\cdots+\tau(2+N_{1/2})$$

$$=\tau(N_{1/2}\log N+N(1-1/N)$$

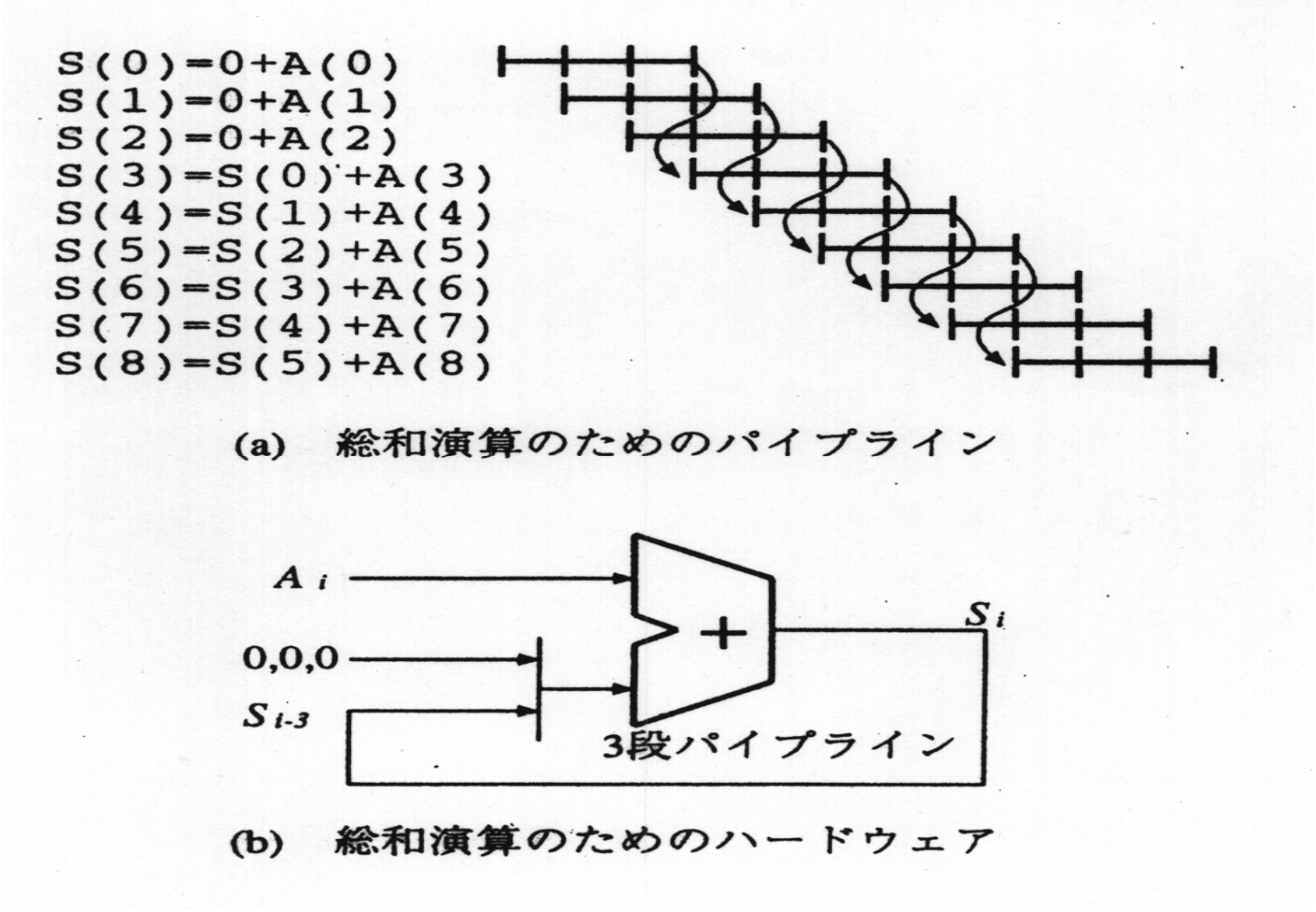
$N=1024$**、**$N_{1/2}=100$**の場合：**



最大性能の約半分の性能

（b）総和演算装置を装備したベクトルコンピュータ

$$T=\tau(N+N_{1/2})$$



（２）サイクリックリダクション法

①並列処理向きアルゴリズム

漸化式

$$X_i=A_iX_{i-1}+D_i \qquad （３３）$$

ただし、$2 \leqq i \leqq N$**、**X_1**は既知（３３）より、**

$$X_{i-1}=A_{i-1}X_{i-2}+D_{i-1} \qquad （３４）$$

（３３）に代入して、

$$X_i=A_iA_{i-1}X_{i-2}+D_i+A_iD_{i-1} \quad (35)$$

これより、

$$X_i=A_iA_{i-1}(A_{i-2}A_{i-3})X_{i-4}+$$

$$D_i+A_iD_{i-1}+A_iA_{i-1}(D_{i-2}+A_{i-2}A_{i-3}) \quad (36)$$

一般に、

$$X_i=A_i^{(l)}X_{i-2^l}+D_i^{(l)} \quad (37)$$

とすると、（３７）より、

$$X_{i-2^l}=A^{(l)}_{i-2^l}X_{i-2^{l+1}}+D^{(l)}_{i-2^l} \quad (38)$$

（３７）、（３８）より、

$i-2^l \geq 1,$

$i \leq N$より

$2^l+1 \leq i \leq N$

$2 \leq i \leq 2^l$で

$X_{i-2^l}=0,$

$X_i=D_i^{(l)}$

$$X_i=A_i^{(l)}A^{(l)}_{i-2^l}X_{i-2^{l+1}}+D^{(l)}_i+A^{(l)}_iD^{(l)}_{i-2^l} \quad (39)$$

したがって、

$$A^{(l+1)}_i=A^{(l)}_iA^{(l)}_{i-2^l} \quad (40)$$

$$D^{(l+1)}_i=D^{(l)}_i+A^{(l)}_iD^{(l)}_{i-2^l}$$

ただし、

$A^{(0)}_i=A_i$、$D^{(0)}_i=D_i$、$2 \leqq i \leqq N$

この過程を$\log N$**回繰り返すと、式（３７）より、**

$$X_i=A^{(\log N)}_iX_{i-N}+D^{(\log N)}_i \quad (42)$$

$2 \leqq i \leqq N$**であるので、**

$$X_{i-N}=0,\ (i \leqq N)$$

$i-2^l \geq 1,$

$i \leq N$より

$2^l+1 \leq i \leq N$

$2 \leq i \leq 2^l$で$X_i=D_i^{(l)}$

よって式（４２）より、

$$X_i = D^{(\log N)}{}_i \qquad （４３）$$

（４３）より、X_iを求めるためには、（４０）の漸化式を解けばよいことが分かる。

$A^{(l)}{}_i$、$D^{(l)}{}_i$の計算$(l=0,1,2,\cdots,(\log N)-1)$：

ベクトル長$N-2^l$

したがって、

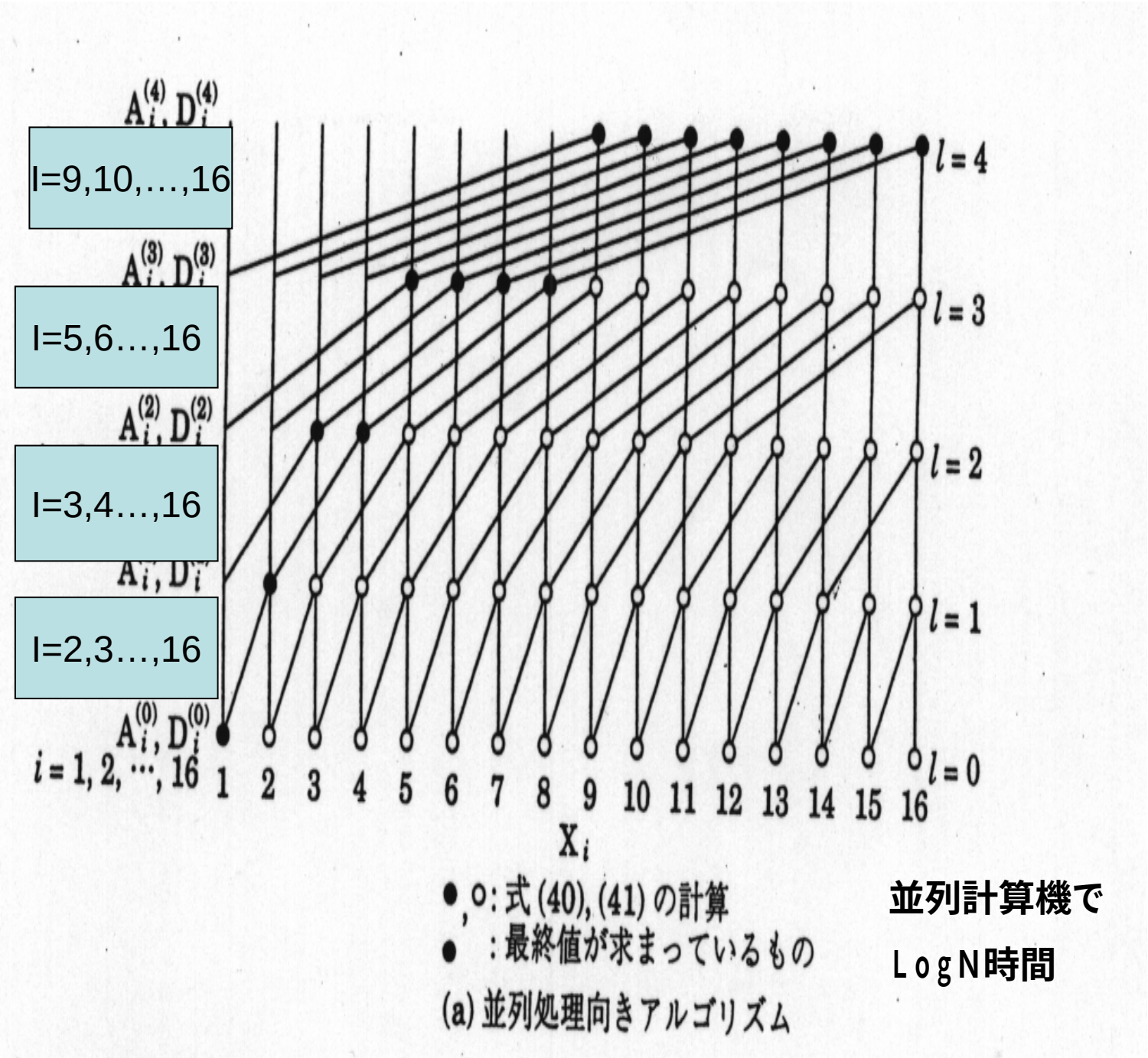
$$T=3\tau_v\Sigma_{l=0}{}^{(\log N)-1}(N+N_{1/2}-2^l)$$

$$=3\tau_v(\log N)(N+N_{1/2}) \qquad (44)$$

汎用コンピュータ

１ループの実行時間：τ_s

$$T=N\tau_s \qquad （４５）$$



並列計算機で

ＬｏｇＮ時間

②ベクトル処理向きアルゴリズム

前半の処理

式（４０）、（４１）より、$i=2,4,8,\cdots,N$の

値のみ計算。

ベクトル長は$N-1,N/2-1,\ldots,1$

$$T=3\tau_v\{(N+N_{1/2}-1)+(N/2+N_{1/2}-1)+\cdots+(1+N_{1/2}-1)\}$$

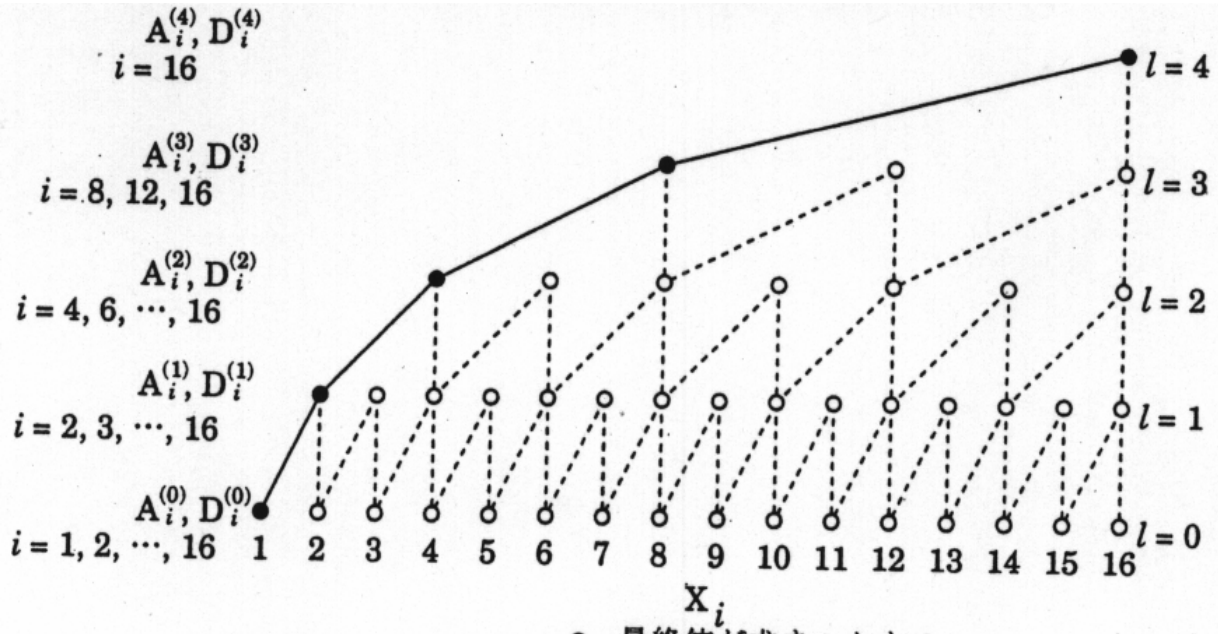
$$=3\tau_v(2N+N_{1/2}\log N) \qquad (46)$$

後半の処理

- **前半の処理：X_8、X_4、X_2の値が確定**
- **X_{12}とX_8の関係は前半で求まっているので、**
  X_{12}の値が確定
- **X_4、X_8、X_{12}とX_6、X_{10}、X_{14}の関係も前半で**
  求まっているので、これらの値が定まる。
- **同様にして、X_3、X_5、X_7、X_9、X_{11}、X_{13}、X_{15}の値が定まる**

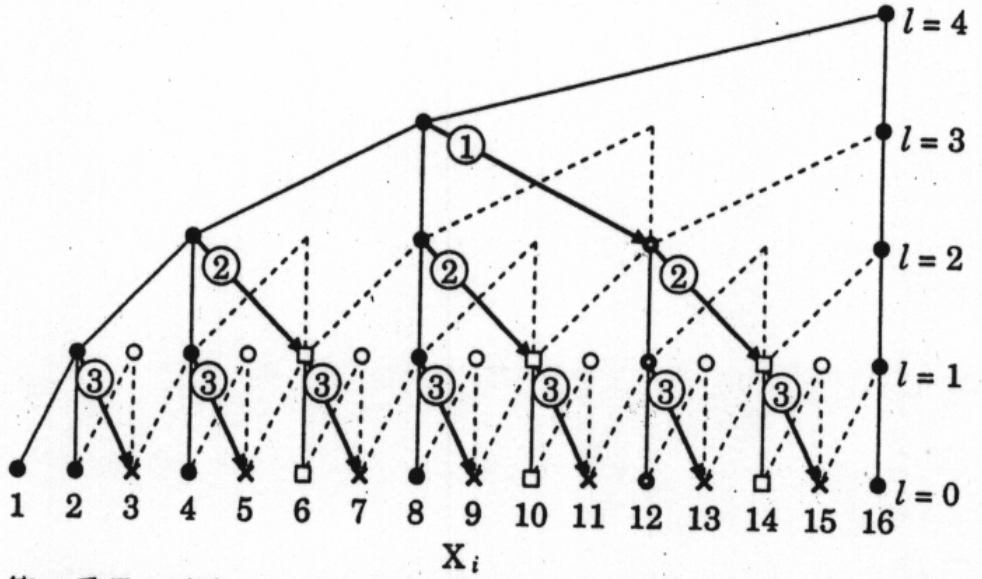
計算時間：$l=3$**で**1、$l=2$**で**3、
$l=1$**で**7**のベクトル長**

lステージ：ベクトル長$N/2^l-1$
（ただし、$l=\log N-1$、・・・、2、1）



●：最終値が求まったもの

(b) ベクトルプロセッサ向きアルゴリズム － 前半



●: 第 1 番目に確定, ⊙: 第 2 番目に確定, □: 第 3 番目に確定, ×: 第 4 番目に確定

(c) ベクトルプロセッサ向きアルゴリズム － 後半

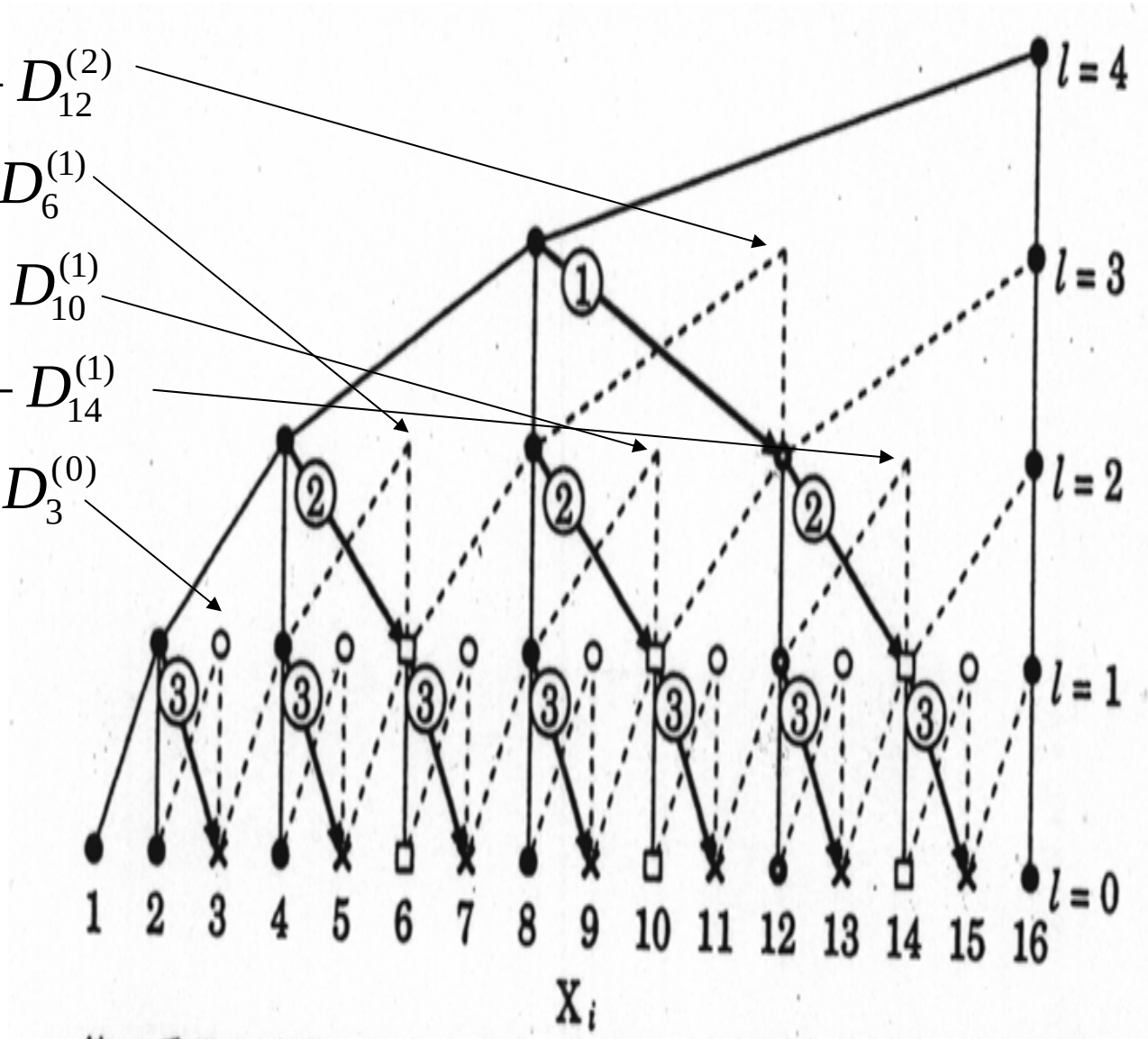
$$X_{12} = A_{12}^{(2)} X_8 + D_{12}^{(2)}$$

$$X_6 = A_6^{(1)} X_4 + D_6^{(1)}$$

$$X_{10} = A_{10}^{(1)} X_8 + D_{10}^{(1)}$$

$$X_{14} = A_{14}^{(1)} X_{12} + D_{14}^{(1)}$$

$$X_3 = A_3^{(0)} X_2 + D_3^{(0)}$$



●: 第 1 番目に確定, ●: 第 2 番目に確定, □: 第 3 番目に確定, ×: 第 4 番

(c) ベクトルプロセッサ向きアルゴリズム − 後半

$$T=2\tau_v\Sigma_{l=1}^{(\log N)-1}(N/2^l-1+N_{1/2})$$

$$=2\tau_v(N+N_{1/2}\log N)$$

$$T=\tau_v(8N+5N_{1/2}\log N) \qquad (48)$$

（3）漸化式の高速実行のためのハードウェア

$$X_i=A_iX_{i-1}+D_i \qquad （49）$$

乗算：４ステージ、加算：３ステージ

$$T=7\tau_v\left(N+N_{1/2}\right)$$

方式１

X_i**と**X_{i-7}**の関係**

$$X_i=A_iA_{i-1}\cdots A_{i-7}X_{i-7}+B_i \qquad （51）$$

X_1、. . .X_7を初期値として解く

方式２

式（４９）より、

$$X_i=X_{i-2}\cdot A_i\cdot A_{i-1}+A_i\cdot D_{i-1}+D_i$$

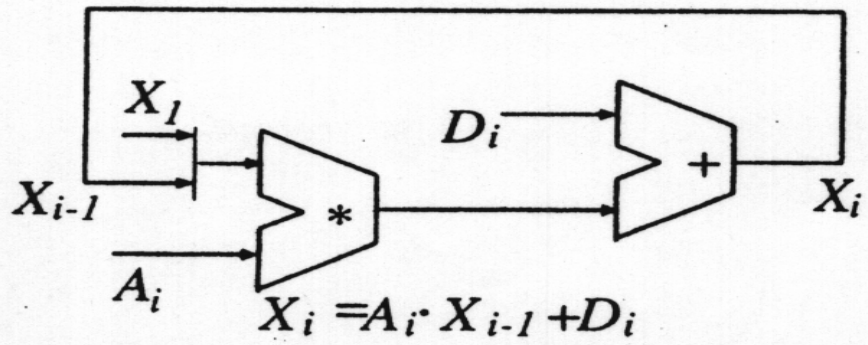
$$=X_{i-2}\cdot E_i+F_i \qquad （５２）$$

をX_1、X_2を初期値として解く。

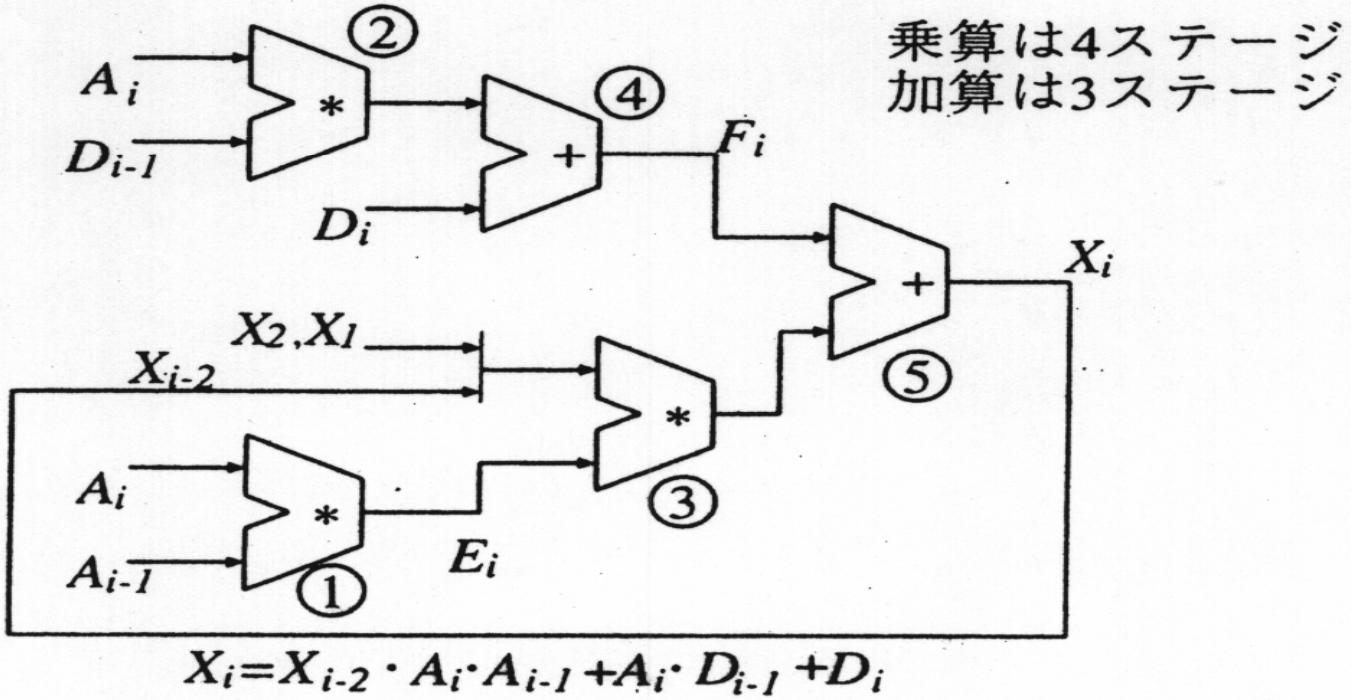
物理的な乗算器への演算投入：

①の演算、②の演算、①の演算、

②の演算、③の演算、一回休、③の演算



(a) X_i を逐次的に 1つずつ 求める方式



(b)X_i, X_{i-1} を連続的 に求める方式

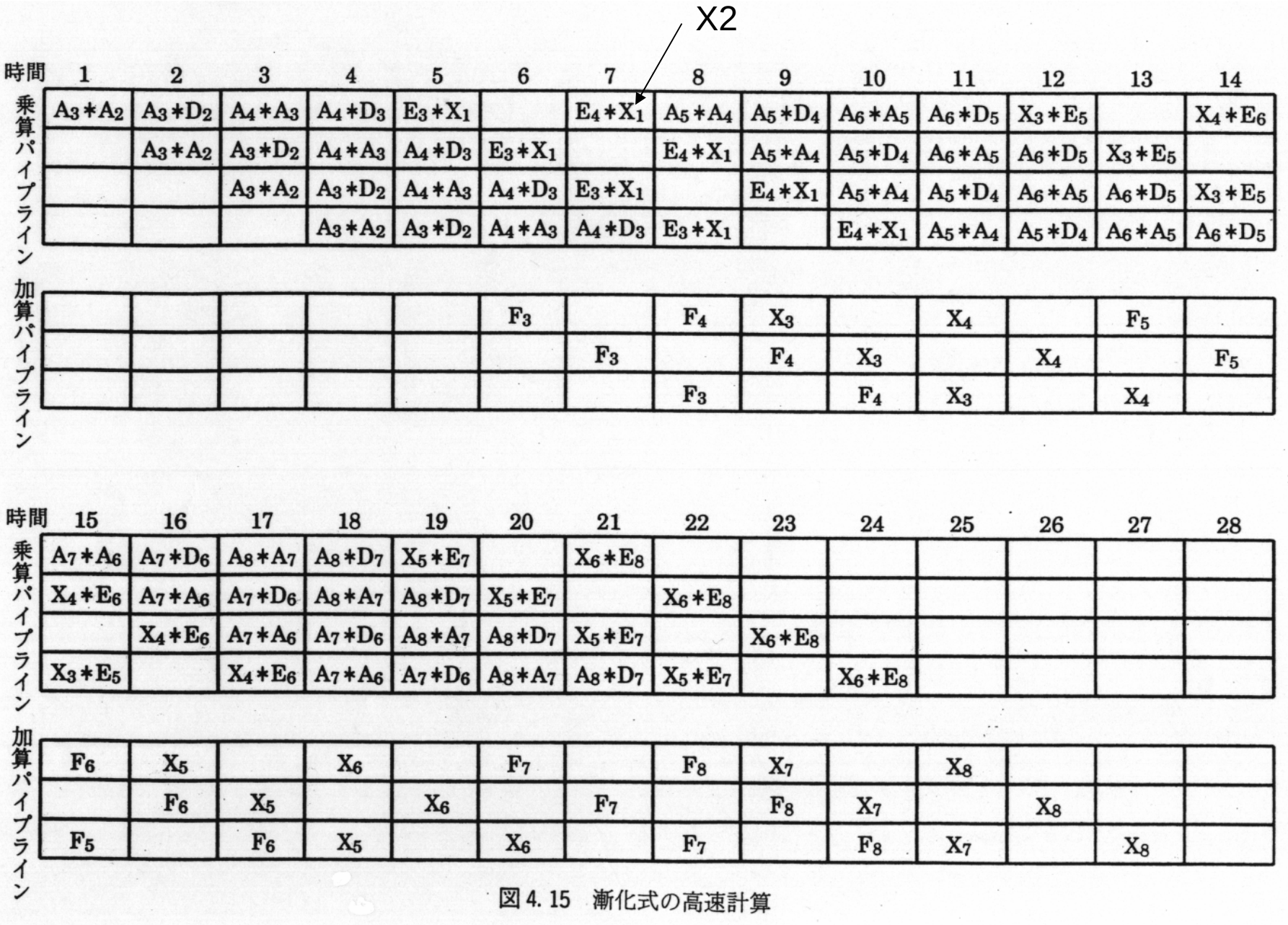
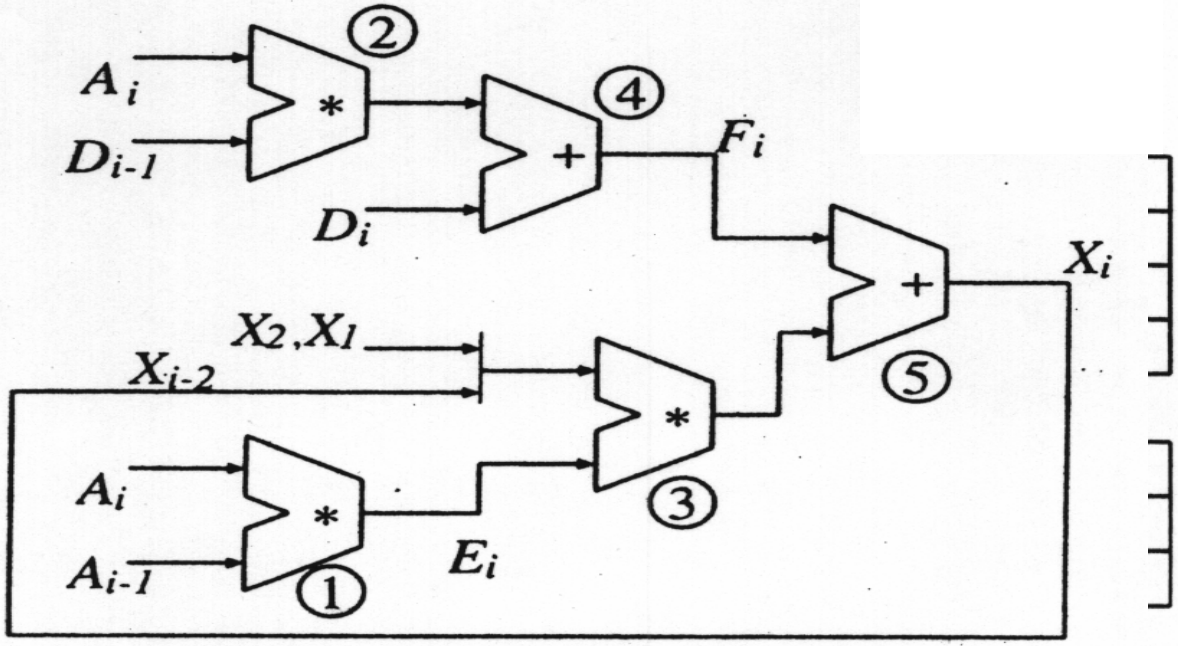


図 4. 15　漸化式の高速計算

X2

時間	1	2	3	4	5	6	7	8	9	10	11	12	13	14
乗算パイプライン	A_3*A_2	A_3*D_2	A_4*A_3	A_4*D_3	E_3*X_1		E_4*X_1	A_5*A_4	A_5*D_4	A_6*A_5	A_6*D_5	X_3*E_5		X_4*E_6
		A_3*A_2	A_3*D_2	A_4*A_3	A_4*D_3	E_3*X_1		E_4*X_1	A_5*A_4	A_5*D_4	A_6*A_5	A_6*D_5	X_3*E_5	
			A_3*A_2	A_3*D_2	A_4*A_3	A_4*D_3	E_3*X_1		E_4*X_1	A_5*A_4	A_5*D_4	A_6*A_5	A_6*D_5	X_3*E_5
				A_3*A_2	A_3*D_2	A_4*A_3	A_4*D_3	E_3*X_1		E_4*X_1	A_5*A_4	A_5*D_4	A_6*A_5	A_6*D_5
加算パイプライン						F_3		F_4	X_3		X_4		F_5	
							F_3		F_4	X_3		X_4		F_5
								F_3		F_4	X_3		X_4	

時間	15	16	17	18
乗算パイプライン	A_7*A_6	A_7*D_6	A_8*A_7	A_8*D_7
	X_4*E_6	A_7*A_6	A_7*D_6	A_8*A_7
		X_4*E_6	A_7*A_6	A_7*D_6
	X_3*E_5		X_4*E_6	A_7*A_6
加算パイプライン	F_6	X_5		X_6
		F_6	X_5	
	F_5		F_6	X_5



$$X_i = X_{i-2} \cdot A_i \cdot A_{i-1} + A_i \cdot D_{i-1} + D_i$$

４．３．４多重ループの
ベクトル化

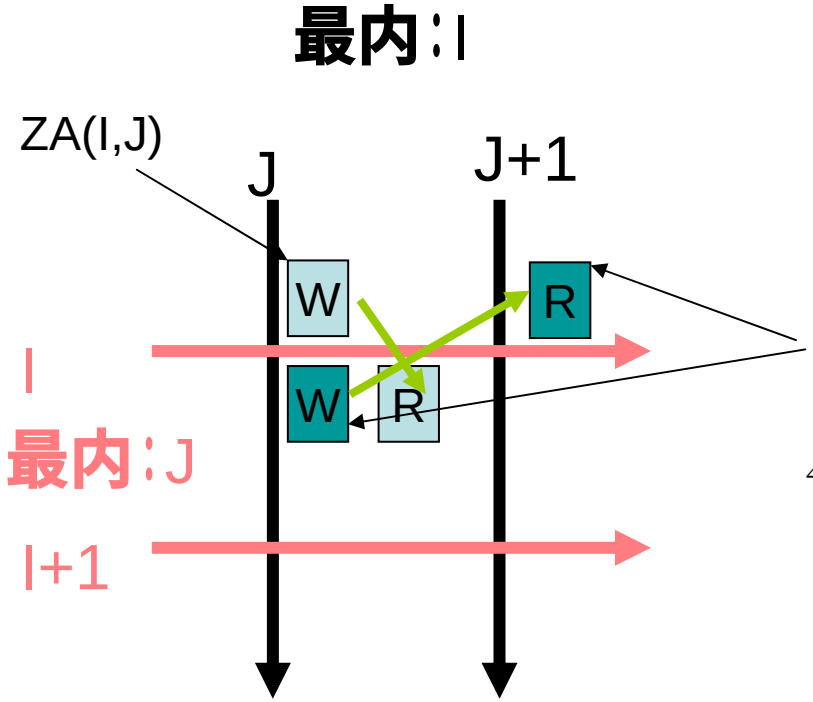
（１）ＤＯループの入替え（ループエクスチェンジ）

```
      DO 10 I=1,N
      DO 10 J=2,N
      ZA(I,J)=PA(I,J-1)+PB(I,J)     (53)
      PA(I,J)=ZA(I,J)xPC(J)
10 CONTINUE
```

最内ループ：Ｊに関してＰＡで

データ参照関係不適

ループの入れ替え



最内Ｊではベクトル化不可

PA(I,J)：(I,J)で定義され，(I,J+1)で引用

4.3.4多重ループのベクトル化

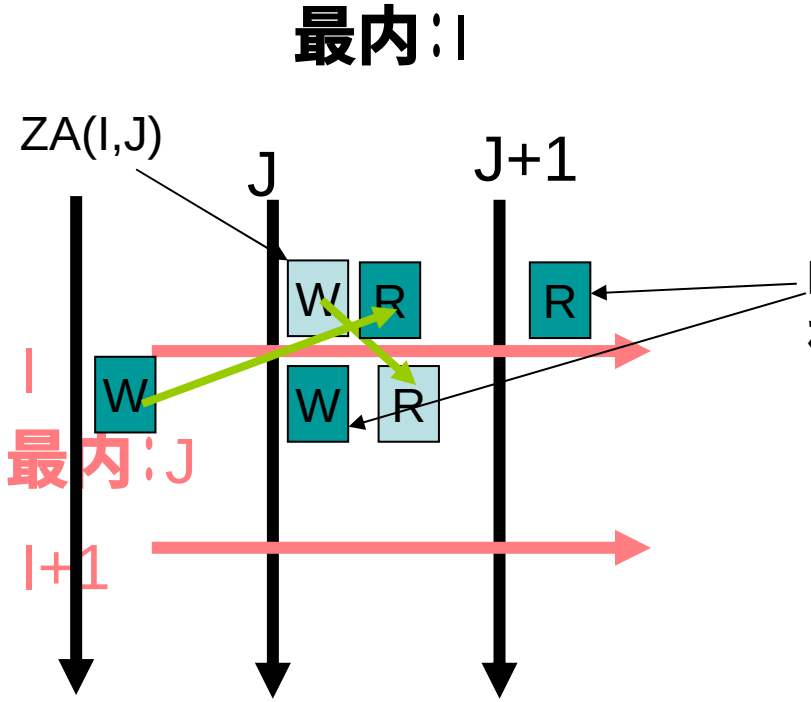
（1）DOループの入替え（ループエクスチェンジ）

```
    DO 10 I=1,N
    DO 10 J=2,N
    ZA(I,J)=PA(I,J-1)+PB(I,J)    (53)
    PA(I,J)=ZA(I,J)xPC(J)
10 CONTINUE
```

最内ループ：Ｊに関してＰＡで

データ参照関係不適

ループの入れ替え



最内Iではベクトル化可能

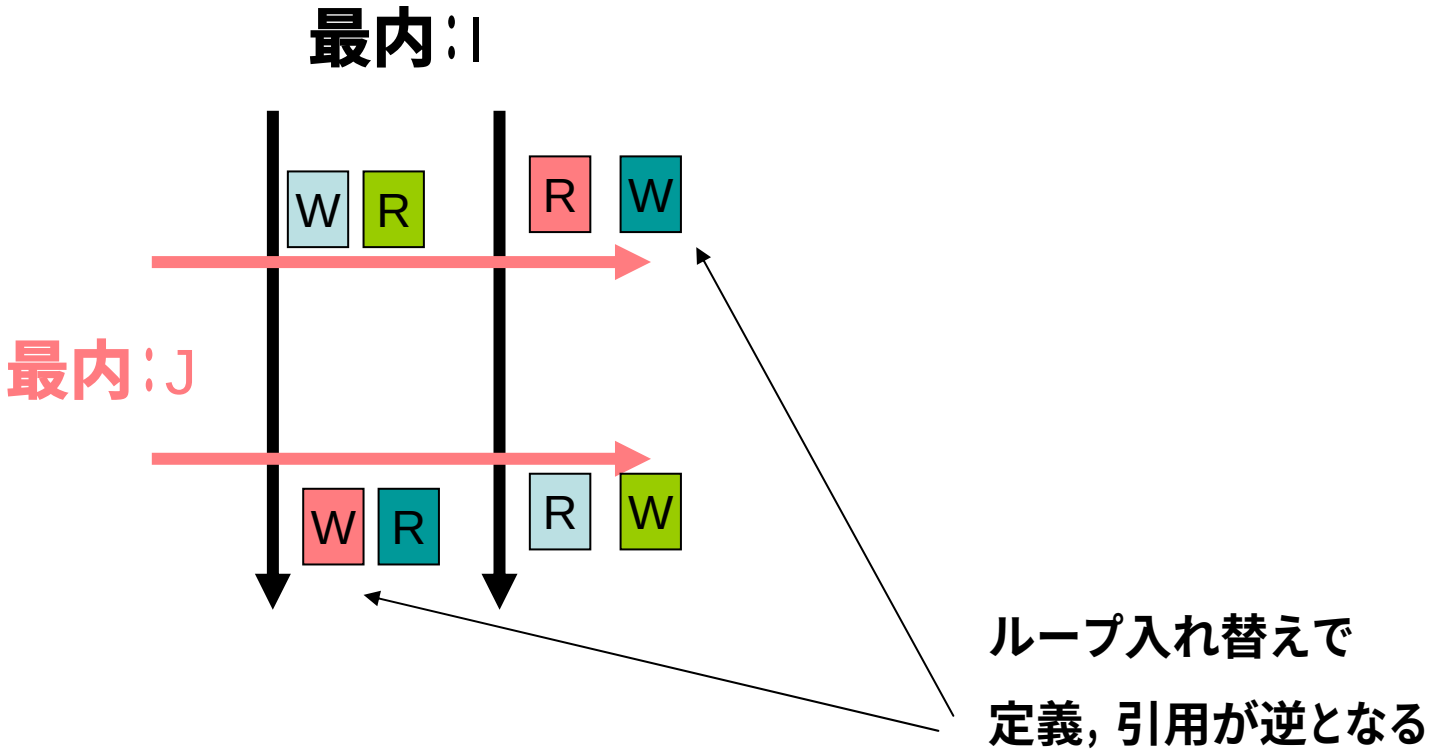
PA(I,J)：(I,J)で定義され，(I,J+1)で引用

ループの入れ換え

```
      DO 10 J=2,N
      DO 10 I=1,N
      ZA(I,J)=PA(I,J-1)+PB(I,J)   (54)
      PA(I,J)=ZA(I,J)xPC(J)
10    CONTINUE
```

最内ループ：ベクトル化可能

ループの入れ替え



ループの入れ換え

```
      DO 10 J=2,N
      DO 10 I=1,N
      ZA(I,J)=PA(I,J-1)+PB(I,J)   (54)
      PA(I,J)=ZA(I,J)xPC(J)
10 CONTINUE
```

最内ループ：ベクトル化可能

（2）多重ループの一重化：

ループ崩壊（loop collapsing**）**

```
    DO 10 I=1,4                (55)
    DO 10 J=1,4
    A(I,J)=B(I,J)
10   CONTINUE
    DO 10 K=1,16
    A(K)=B(K)
10   CONTINUE
```

```
      DO 10 I=1,100          (56)
      DO 10 J=1,I
      A(I,J)=B(I,J)+C
10 CONTINUE
```

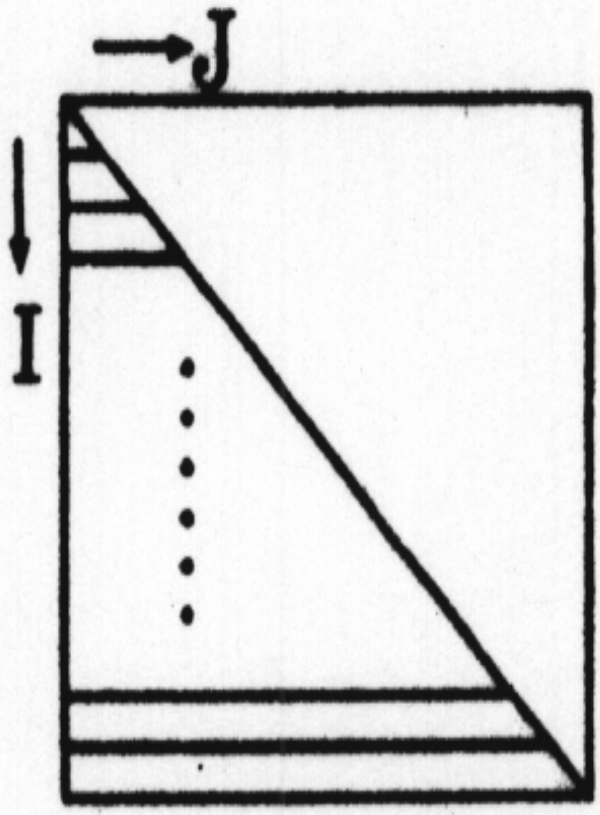
Ａ，Ｂの下三角部分に対してのみ処理

Ａ，Ｂの下三角部分の各要素アドレス：

リストベクトル

$IX=(AD_{11},AD_{21},AD_{22},AD_{31},..,AD_{100\ 1}\ ,..,AD_{100100})$ (57)

```
      DO 10 I=1,5050
      A(IX(I))=B(IX(I))+C                    (58)
 10 CONTINUE
```



A(I , J) , B(I , J) のアクセス領域

（３）一重ループの多重化（ストリップマイニング）

ベクトルプロセッサの並列処理

非常に長いベクトルの分割

（４）多重ループのアンローリング

```
    DO 10 J=1,N
    DO 10 I=1,M                (59)
10  A(I)=A(I)+B(I,J)/X(J)


    DO 10 J=1,N-1,2
    DO 10 I=1,M
10  A(I)=A(I)+B(I,J)/X(J)+B(I,J+1)X(J+1)
```

(60)

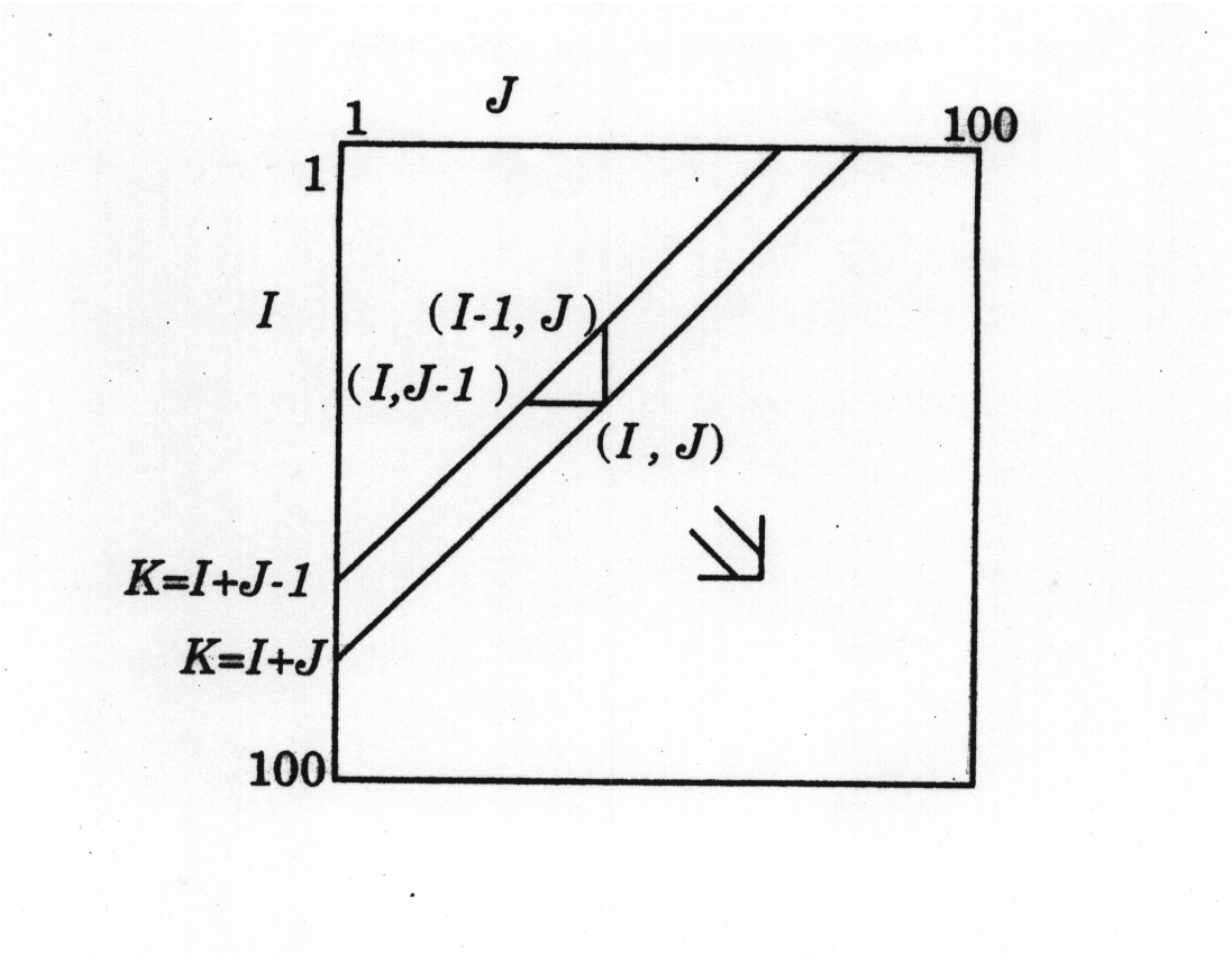
（５）ウェーブフロント化

```
      DO 10 I=2,100
      DO 20 J=2,100                                    (61)
      A(I,J)=A(I,J-1)+A(I-1,J)
20 CONTINUE
10 CONTINUE
```

I+1行：I行の値を用いた回帰演算で計算

ウェーブフロント（波頭）

```
        K=I+J-1
```


J
1
100
1
I
(I-1, J)
(I,J-1)
(I , J)
K=I+J-1
K=I+J
100

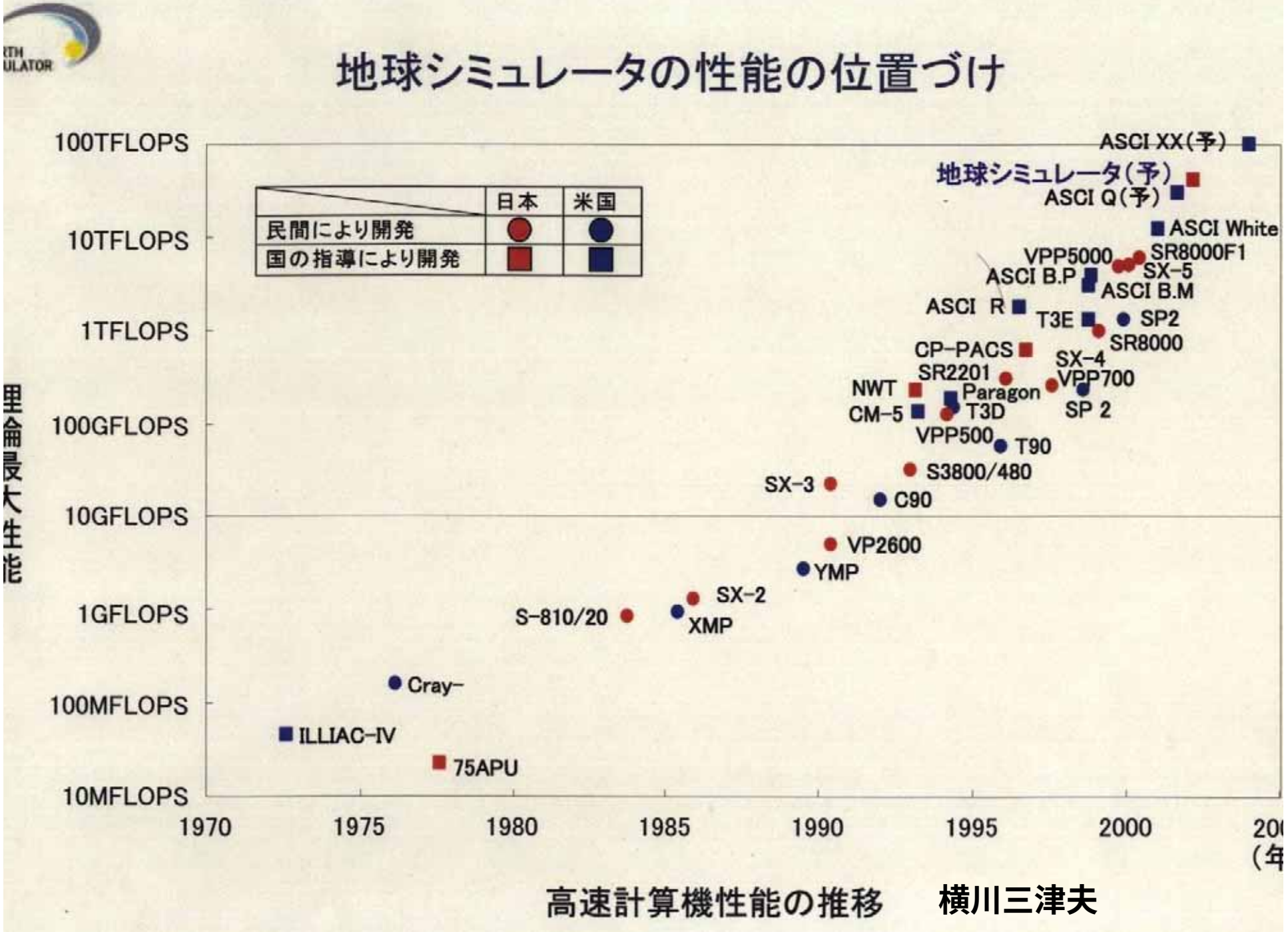
ベクトルコンピュータの将来

- CRAY-1 以降 １９００年代半ばまで席巻
- ゲーム機プレステ2で採用
- さまざまな技術の集積
- ベクトルコンピュータ技術：

  あまり売れない

  ドライビングフォースー＞汎用機の高速化

地球シミュレータの性能の位置づけ



高速計算機性能の推移　　横川三津夫

表 2.5　ASCI　プラットフォームの概要

名称	Red	Blue Pacific	Blue Mountain	White	T30
設置研究所	Sandia	Lawrence Livermore	Los Alamos	Lawrence Livermore	Los Alamos
メーカ	Intel	IBM	SGI	IBM	未定
使用 MPU	9,536×Pentium Ⅱ Xeon	5,856×Power PC	Origin2000 MIPS R10000	8,192×Power 3-Ⅱ	未定
目標性能	1.8Tflops メモリ 606GB Disk 容量 40TB	3.1Tflops メモリ 2.6TB Disk 容量 75TB	3.1Tflops メモリ 2.5TB Disk 容量 75TB	10.2Tflops メモリ 2.5TB Disk 容量 75TB	30 + Tflops
実績	3.2Tflops ('99/10 月)	3.9Tflops ('98/10 月)	3.1Tflops ('98 年)	― ― ―	― ― ―

(注意)　性能はピーク性能値である。

京都新聞　2003年（平成15年）11月23日

日米スパコン計算最速でしのぎ

用途拡大で躍起

毎秒35兆回→360兆回…1000兆回へ

科学技術計算に利用されるスーパーコンピューター（スパコン）で、日米間の世界最速争いに拍車が掛かっている。スパコンはヒトのゲノム（全遺伝情報）を活用した医薬研究などに用途が広がってきた上、国防上も重要な役割を果たすためだ。最速マシンを日本製に奪われた米国側がナンバーワン奪回へ躍起となっている。

「日本は科学技術計算で新時代を切り開いた」とで称賛されるべきだが、米国が新時代に遅れてはならない」―。エーブラハム米エネルギー長官は今月十日、ワシントンでの講演で世界最速を誇る日本の「地球シミュレータ」（ES）を引き合いにし、最速の座を取り戻すと強調した。

二〇〇二年に完成したESは、海洋科学技術センターなどが開発し、NECが製造に当たった。平和利用を目的とする用途は気候変動予測、地殻変動の解明などで、実測値で毎秒三五・八六〇兆回の計算能力を持っている。それまでの最速は、核兵器の備蓄管理などを目的に米国で用いられるIBM製マシンの毎秒七・二二六兆回だっただけに、「衝撃的な数字」（米ニューヨーク・タイムズ紙）だ。

米国の科学者らが今月中旬にまとめた最新のスパコン上位五百機リストでも、ESは登場以来の首位を堅持、二位の米ヒューレット・パッカード製マシンの三倍近い能力となっている。

ただ、IBMは遺伝子からつくられる、たんぱく質の構造解析などに利用するため、毎秒一〇〇〇兆回規模の計算能力を備えたスパコン開発計画を一九九九年から推進中。〇一年には計画の一環として、米エネルギー省と共同で「ブルー・ジーン（青い遺伝子）／L」と呼ぶスパコン開発を打ち出している。目指す能力は毎秒三百六十兆回。既に小型試作機が完成し、IBMは最新ランキングで試作機が七十三位に入ったとアピール。「〇五年の完成時にはトップになる」と首位奪還を予告している。

システム設計の立場からの性能限界と衰退理由

①パイプラインのピッチの高速化限界

②メモリ：集中制御

５１２～１０２４ウェイのインタリーブ方式

（メモリを多数のバンクで構成する方式）

③ECL（Emitter Coupled Logic）素子の使用

１０GFLOPS性能のシステム：数百から数千ｋW

④マルチプロセッサの性能向上

ベクトルプロセッサ開発の二重投資

NECのみ奮闘

システム設計者からみた要因

①マイクロプロセッサの驚異的高速化と低価格化

１GFLOPSの性能を有するプロセッサ

１０００個でピーク性能で１TFLOPS

プロセッサ１０万円、１０００個で１億円

②低電力

CMOS素子利用、消費電力：１台当り１０W程度

１０００台のマルチプロセッサ：１０ｋW程度

③必要悪としてのユーザへの押付け

１TFLOPSの要求：

実現可能システムはマルチプロセッサ方式

以外ない？？