

第六章 向量处理机

Dr. Feng Li

fli@sdu.edu.cn

<https://funglee.github.io>

向量处理机

- 向量处理机是具有向量表示的处理机
 - 向量流水处理机：以时间重叠途径开发
 - 阵列处理机：资源重复途径开发。设置大量重复的处理单元，按一定方式互连成阵列，在单一控制部件的控制下，对各自所分配的不同数据进行同一指令规定的操作，是操作级并行的SIMD计算机

向量处理方式

- 横向处理方式：向量计算是按行的方式从左至右横向进行
- 纵向处理方式：向量计算是按列的方式自上而下纵向进行
- 纵横处理方式：横向处理和纵向处理相结合的方式

举例

$\mathbf{D} = \mathbf{A} \times (\mathbf{B} + \mathbf{C})$, 其中 $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ 均为具有 N 个元素的向量

- 横向处理

- $a_i \times (b_i + c_i), a_{i+1} \times (b_{i+1} + c_{i+1}), \dots$
- 对于每条指令: $b_i + c_i \rightarrow k, k \times a_i \rightarrow d_i$
- 虽然只需要额外的一个单元 k 来保存中间结果, 但是会频繁发生“先写后读”的操作数相关, 流水线吞吐量下降
- 若流水线是静态的, 还需要进行功能切换, 流水线吞吐率比顺序执行还要低

- 纵向处理
 - $b_i + c_i \rightarrow k_i (i = 1, \dots, N), k_i \times a_i \rightarrow d_i (i = 1, \dots, N)$
 - 无相关发生，流水线能连续流动
 - 功能切换只发生一次
 - 需要同时保存多个中间结果，对存储器信息流量要求较高
- 纵横处理
 - 把长度为 N 的向量分成若干组，每组长度为 n ，组内按纵向处理方式，组间横向处理
 - 若 $N = K \cdot n + r$ ，则共有 $K + 1$ 组

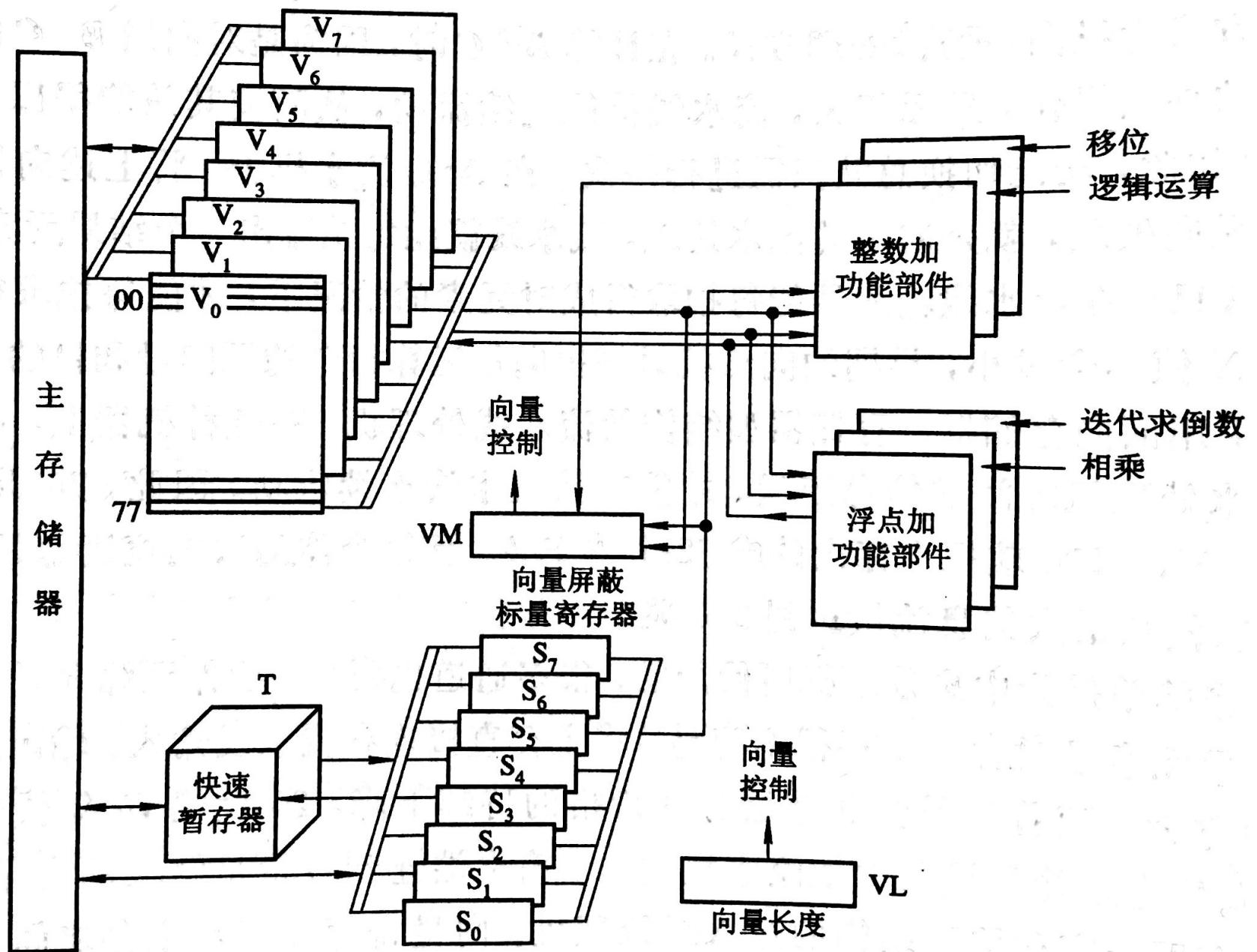
向量处理机的结构

- 早期向量机将向量运算指令的源向量和目的向量存放在主存中，采用面向“存储器—存储器”型结构的流水线处理机。由于主存速度的限制，很难保证源向量连续供应和结果向量元素及时存入主存
- 随着半导体存储器芯片价格的持续下降，可以将流水处理机改为把流水线输入、输出端连到容量足够大的向量寄存器组，采用面向“寄存器—寄存器”型结构的流水线处理机
- 若寄存器组个数不够，可采用纵横处理方式

向量流水处理机的结构举例：CRAY-1向量流水处理机

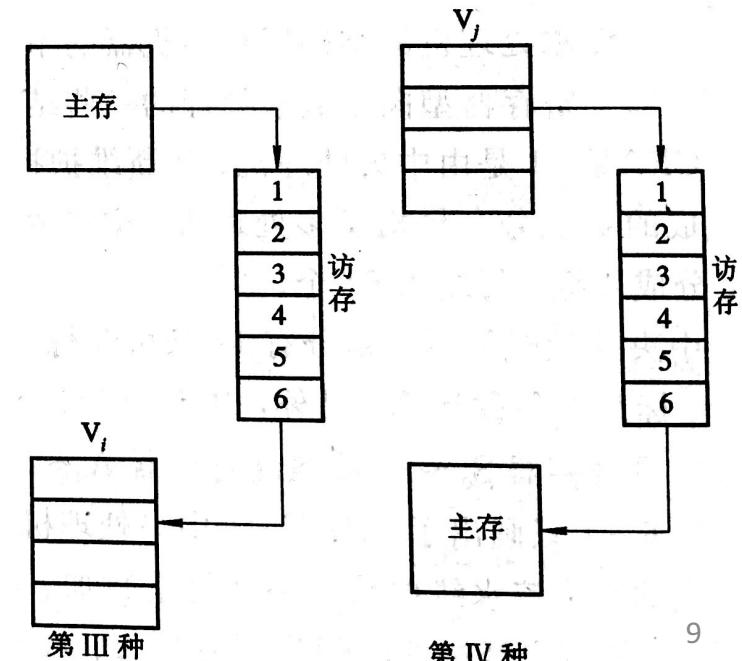
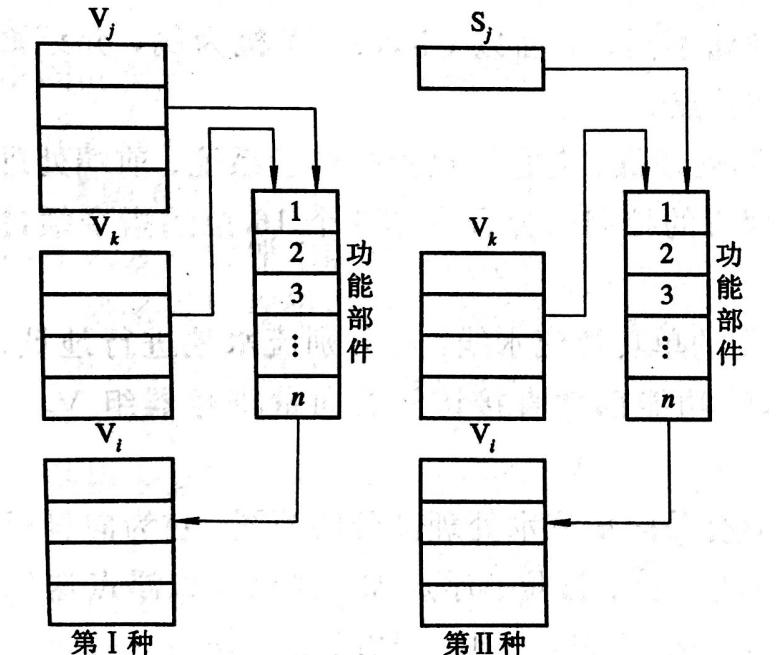
- 1972年首次交付使用CRAY-1向量流水处理机
- 分布异构型多处理机系统，由中央处理机、诊断维护控制处理机、大容量磁盘存储子系统、前端处理机组成
- 12条可并行工作的单功能流水线，其中6条向量运算流水线（整数加、逻辑运算、移位、浮点加、浮点乘和浮点迭代求倒数）
- 向量寄存器组 $V_0 \sim V_7$ ，标量寄存器 $S_0 \sim S_7$ 及地址寄存器 $A_0 \sim A_7$
- 向量寄存器由512个64位寄存器组成，分成8组。每个向量寄存器组 V_i 同时可存放最多64个分量（元素）的一个向量

向量寄存器组(8×64个)



四种向量指令

- 第一类指令从一个或二个向量寄存器取得操作数，并把结果送回到另一个向量寄存器中去
- 第二类指令从标量寄存器 S_j 取得一个标量操作数，又从向量寄存器 V_k 取得一个向量操作数，并把向量结果送到另一个向量寄存器 V_i
- 第三类和第四类分别把数据从存储器传送到一个向量寄存器中以及反过来把数据从向量寄存器传送到存储器中



通过并行、链接提高性能

- 加快临近向量指令执行
 - 多个流水线功能部件并行
 - 流水线链接
- 让循环向量化
 - 加快条件语句和稀疏矩阵处理
 - 加快向量归约操作

举例 (CRAY-1)

- 流水线链接
 - 向量寄存器组 V_i 在同一时钟周期内可接收一个结果分量并为下次运算提供一个源分量
 - 把寄存器组既作为结果寄存器组又作为源寄存器组的做法可以将两条或多条向量指令链接成一条
- 多个流水线功能部件并行
 - 向量寄存器组和功能部件之间有单独的I/O总线
 - 只要不出现 V_i 冲突和功能部件冲突，各个 V_i 和功能部件之间都能并行工作

$$\begin{array}{l} V_4 \leftarrow V_1 + V_2 \\ V_5 \leftarrow V_1 \wedge V_3 \end{array}$$

源向量冲突

$$\begin{array}{l} V_4 \leftarrow V_2 * V_3 \\ V_5 \leftarrow V_1 * V_6 \end{array}$$

功能部件冲突

举例 (CRAY-1)

- 在不出现功能部件使用冲突和源向量寄存器使用冲突的情况下，通过链接机构可以使得有数据相关的向量指令仍然能够在大部分时间并行执行

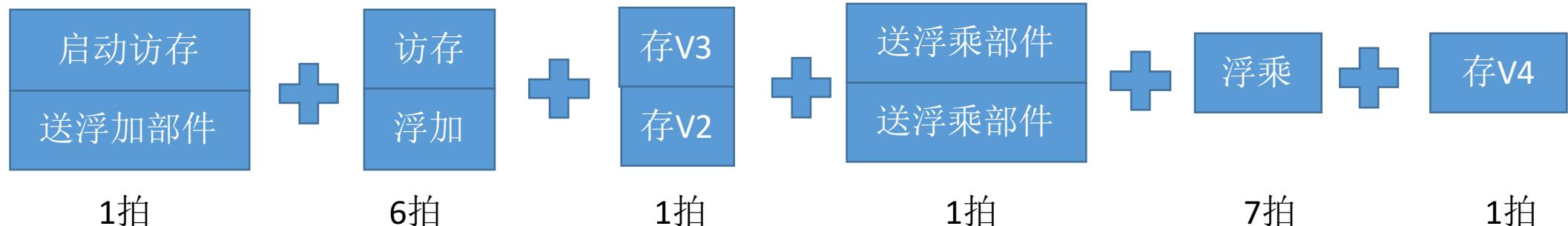
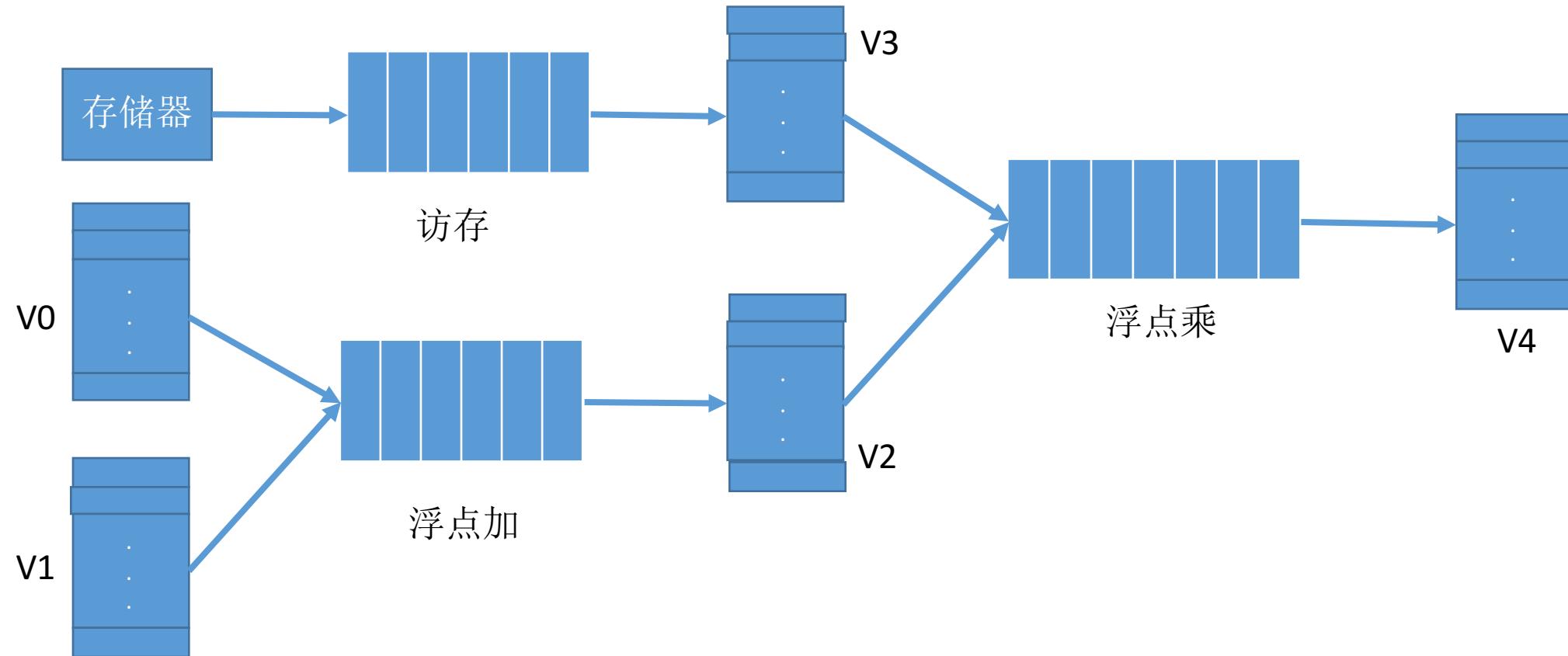
$$D = A \times (B + C)$$

$V_3 \leftarrow$ 存储器； 访存取A向量

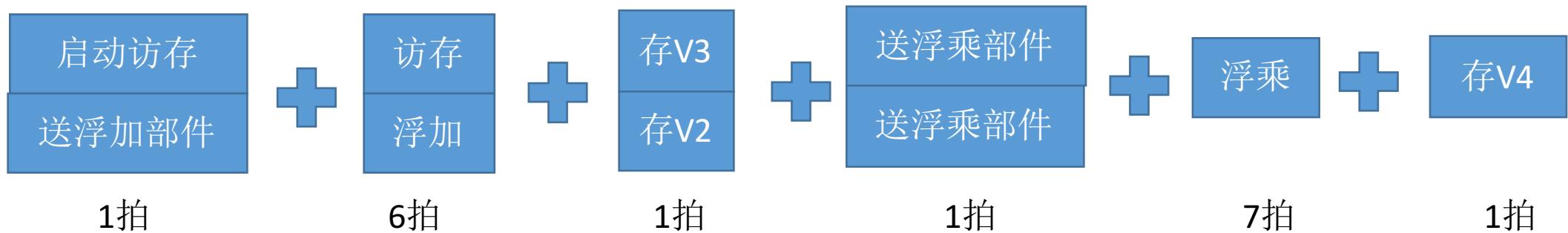
$V_2 \leftarrow V_0 + V_1$ ； B向量和C向量浮点加

$V_4 \leftarrow V_2 * V_3$ ； 浮点乘，存D向量

若将第一、二条指令的结果分量直接链接进第三条指令所用的功能部件，那么第三条指令可以与第一、二条指令在大部分时间内并行



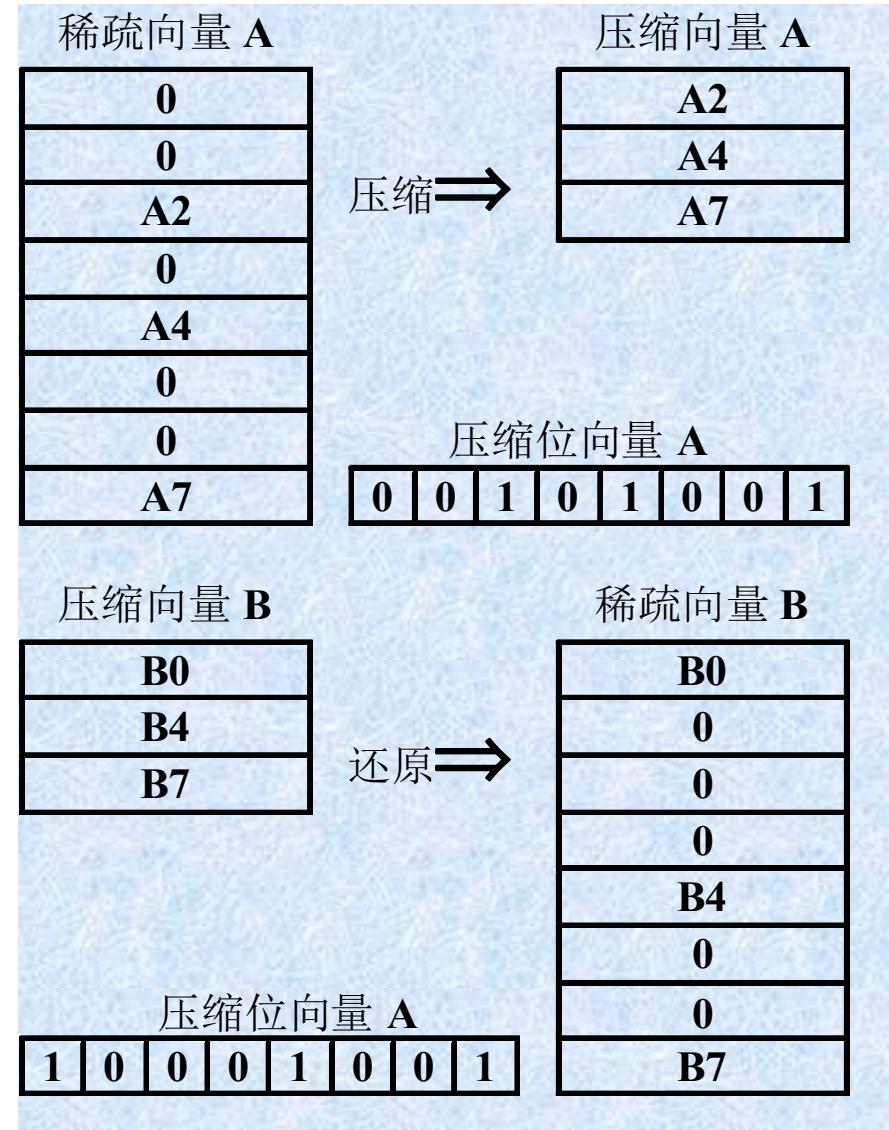
$$D = A \times (B + C)$$



- 共需要 $17 + (N - 1)$ 拍就可以执行完这3条向量指令，获得全部分量
- 若执行完所有第一、二条指令再执行第三条指令，需要 $15 + 2N$ 拍

- 稀疏矩阵的加速处理

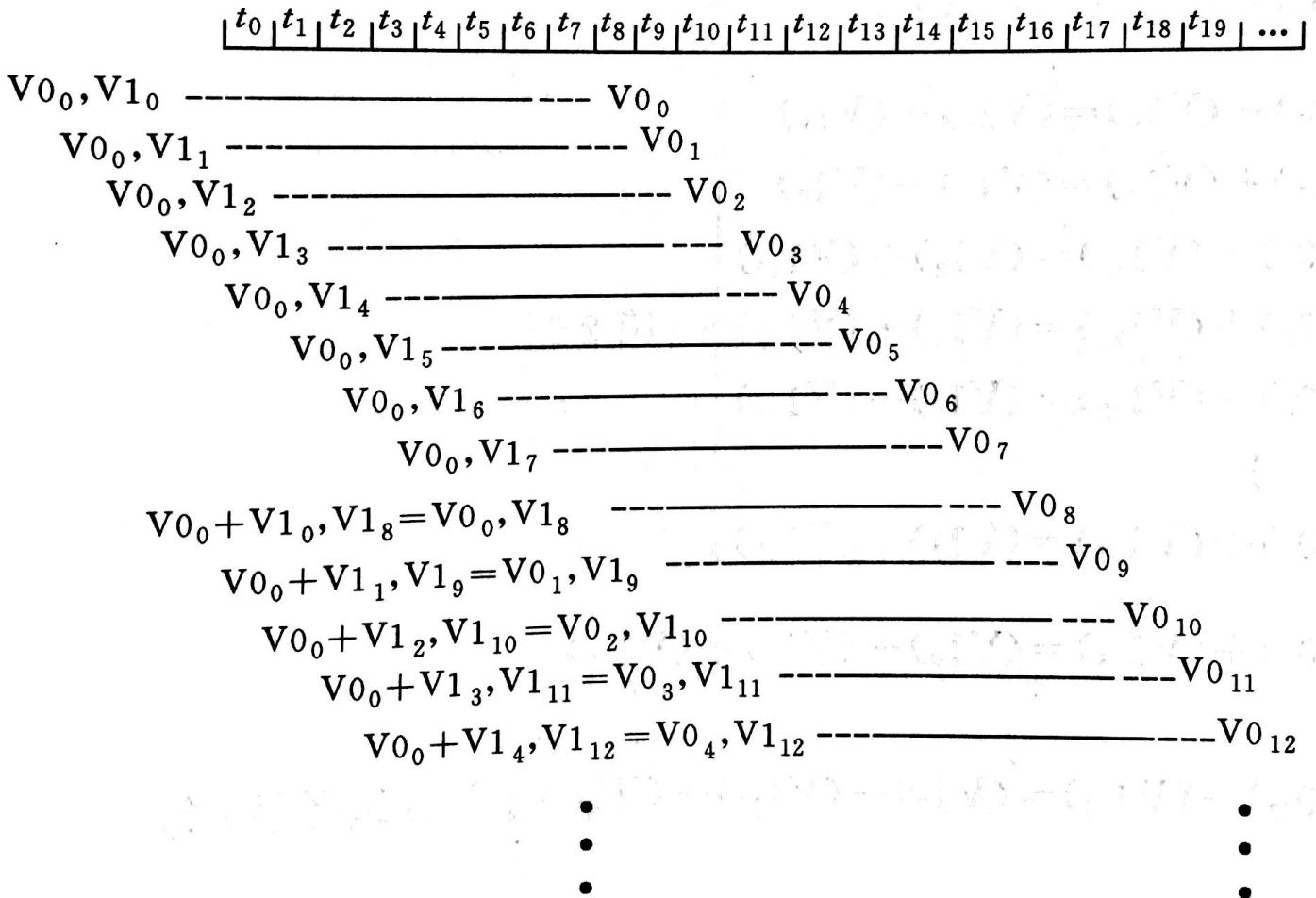
- 稀疏矩阵：非零元素较少的矩阵
- 一个稀疏向量由两个向量组成
 - 短向量仅包含向量的非零元素
 - 位向量的长度与稀疏向量的长度相等，“1”表示对应位置为非零元素，“0”表示对应位置为零元素
- 采用向量屏蔽技术，利用屏蔽码，可以将两个稀疏向量改成稠密向量存放，通过两个屏蔽码的与、或操作，可以控制对两个稀疏向量的聚合和散射，加快对稀疏矩阵的处理



- 向量递归技术
 - 有些向量循环，其流水线功能部件的输出可能要回送到它的同一个源向量寄存器，即用同一个向量寄存器来存放源操作数和结果操作数
 - 每一个向量寄存器设置一个分量计数器，当一个操作数分量移出向量寄存器进入流水线功能部件时，一个结果分量可以在同一周期进入腾空的分量寄存器
 - 分量计数器必须跟踪移位操作，直到结果向量的所有64个分量都装入向量寄存器

递归向量求和 $V0 \leftarrow V0 + V1$

- V_1 保存要进行递归相加的向量，而 V_0 同时用做操作数寄存器和结果寄存器
- C_0 和 C_1 分别是对应的分量计数器
- 浮点加法流水线需要 6 个周期，寄存器和流水线之间的往返传送各需 1 个周期，因此一次加法计算共需 8 个周期
- t_8 之前， $C_0 = 0$ ；之后， C_0 每个周期都加 1
- C_1 每个周期都加 1



$$\left. \begin{array}{l} (V0_0) = (V0_0) + (V1_0) = 0 + (V1_0) \\ (V0_1) = (V0_0) + (V1_1) = 0 + (V1_1) \\ (V0_2) = (V0_0) + (V1_2) = 0 + (V1_2) \\ (V0_3) = (V0_0) + (V1_3) = 0 + (V1_3) \\ (V0_4) = (V0_0) + (V1_4) = 0 + (V1_4) \\ (V0_5) = (V0_0) + (V1_5) = 0 + (V1_5) \\ (V0_6) = (V0_0) + (V1_6) = 0 + (V1_6) \\ (V0_7) = (V0_0) + (V1_7) = 0 + (V1_7) \end{array} \right\}$$

第 1 组

$$\left. \begin{array}{l} (V0_8) = (V0_0) + (V1_8) = (V1_0) + (V1_8) \\ (V0_9) = (V0_1) + (V1_9) = (V1_1) + (V1_9) \\ (V0_{10}) = (V0_2) + (V1_{10}) = (V1_2) + (V1_{10}) \\ (V0_{11}) = (V0_3) + (V1_{11}) = (V1_3) + (V1_{11}) \\ (V0_{12}) = (V0_4) + (V1_{12}) = (V1_4) + (V1_{12}) \\ \vdots \\ (V0_{15}) = (V0_7) + (V1_{15}) = (V1_7) + (V1_{15}) \end{array} \right\}$$

第 2 组

$$(V0_{16}) = (V0_8) + (V1_{16}) = (V1_0) + (V1_8) + (V1_{16})$$



$$(V0_{55}) = (V0_{47}) + (V1_{55}) = (V1_7) + (V1_{15}) + (V1_{55}) : \text{第 3 组到第 7 组}$$



$$+ (V1_{31}) + (V1_{39}) + (V1_{47}) + (V1_{55})$$

$$\left. \begin{array}{l} (V0_{56}) = (V0_{48}) + (V1_{56}) = (V1_1) + (V1_8) + (V1_{16}) + (V1_{24}) \\ \quad + (V1_{32}) + (V1_{40}) + (V1_{48}) + (V1_{56}) \\ (V0_{57}) = (V0_{49}) + (V1_{57}) = (V1_1) + (V1_9) + (V1_{17}) + (V1_{25}) \\ \quad + (V1_{33}) + (V1_{41}) + (V1_{49}) + (V1_{57}) \\ (V0_{58}) = (V0_{50}) + (V1_{58}) = (V1_2) + (V1_{10}) + (V1_{18}) + (V1_{26}) \\ \quad + (V1_{34}) + (V1_{42}) + (V1_{50}) + (V1_{58}) \\ (V0_{59}) = (V0_{51}) + (V1_{59}) = (V1_3) + (V1_{11}) + (V1_{19}) + (V1_{27}) \\ \quad + (V1_{35}) + (V1_{43}) + (V1_{51}) + (V1_{55}) \\ (V0_{60}) = (V0_{52}) + (V1_{60}) = (V1_4) + (V1_{12}) + (V1_{20}) + (V1_{28}) \\ \quad + (V1_{36}) + (V1_{45}) + (V1_{53}) + (V1_{60}) \\ (V0_{61}) = (V0_{53}) + (V1_{61}) = (V1_5) + (V1_{13}) + (V1_{21}) + (V1_{29}) \\ \quad + (V1_{37}) + (V1_{45}) + (V1_{53}) + (V1_{61}) \\ (V0_{62}) = (V0_{54}) + (V1_{62}) = (V1_6) + (V1_{14}) + (V1_{22}) + (V1_{30}) \\ \quad + (V1_{38}) + (V1_{46}) + (V1_{54}) + (V1_{62}) \\ (V0_{63}) = (V0_{55}) + (V1_{63}) = (V0_7) + (V1_{15}) + (V1_{23}) + (V1_{31}) \\ \quad + (V1_{39}) + (V1_{47}) + (V1_{55}) + (V1_{63}) \end{array} \right\}$$

第 8 组 (结果)

阵列处理机

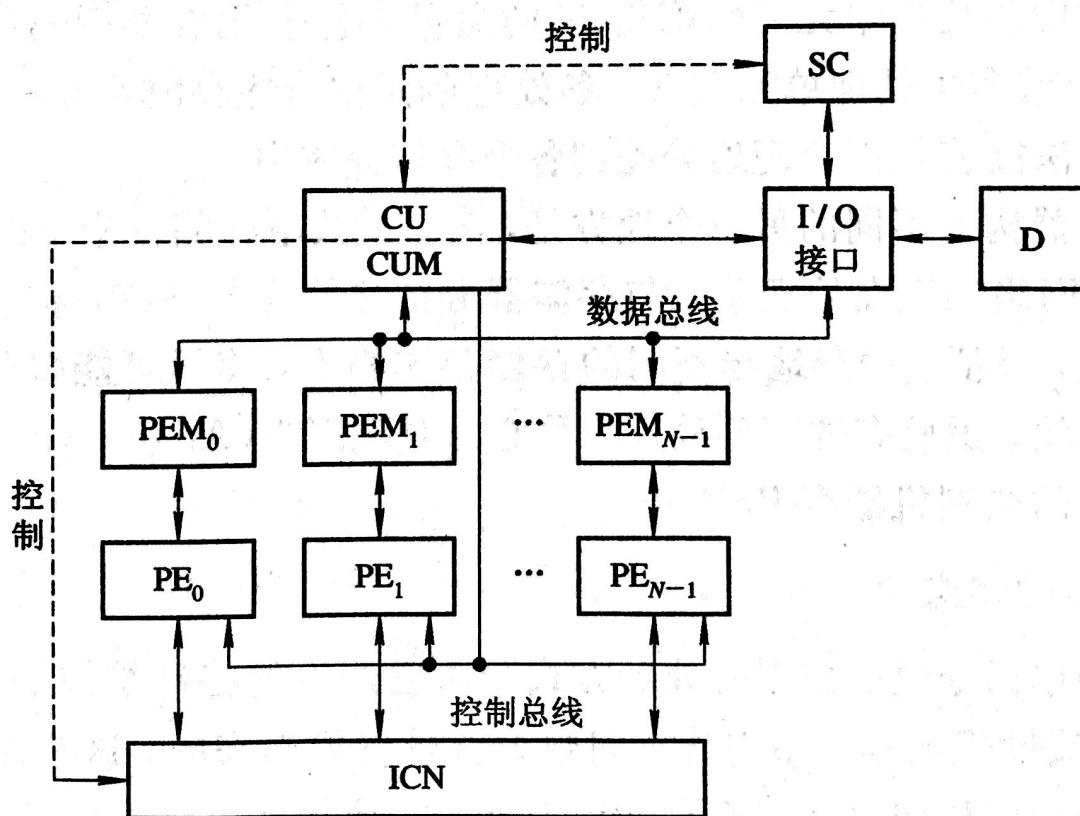
- 阵列处理机（Array Processor）也称并行处理器（Parallel Processor）通过重复设置大量相同的处理单元PE（Processing Element），将它们按一定方式互连成阵列，在单一控制部件CU（Control Unit）控制下，对各自所分配的不同数据并行执行同一组指令规定的操作。

阵列处理器的构形与特点

- 分布式存储器的阵列处理机构形
- 集中式共享存储器的阵列处理机构形

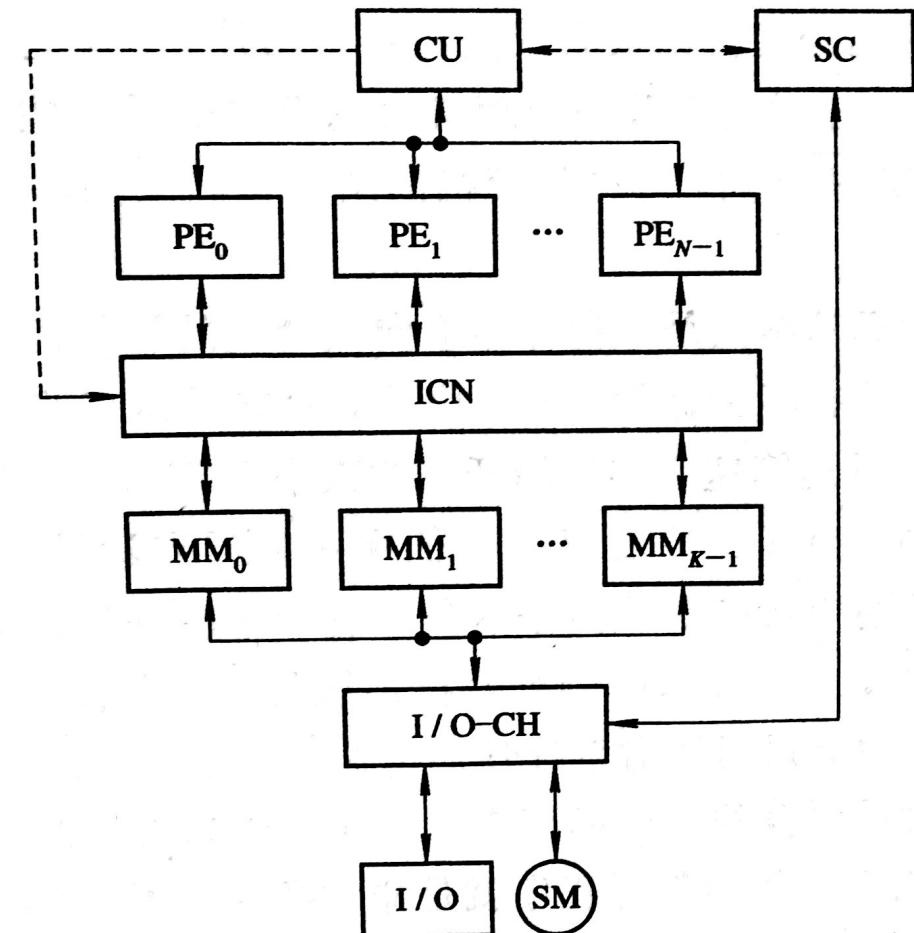
分布式存储器的阵列处理机

- 各处理单元设有局部存储器 PEM (Processing Element Memory), 存放被分布的数据; 只能被本处理单元直接访问
- 控制部件 CU 设有存放程序和数据的主存储器
- 整个系统在 CU 控制下运行用户程序和部分系统程序
- 处理单元之间可通过互连网络 ICN (Interconnection Network) 来交换数据, 其通路选择也由控制部件统一控制
- 在执行主存储器中的用户程序时, 所有指令都在控制部件中进行译码, 把适合串行处理的标量或控制指令留给控制部件自己执行, 而把适合于并行处理的向量类指令“播送”到各个 PE
- 目前的大部分阵列处理机是基于分布式存储器模型的系统



集中式共享存储器的阵列处理器

- 存储器由K个存储体集中组成，经互连网络ICN为全部N个处理单元所共享
- 互连网络用于在处理单元与存储体分体之间进行转接而构成数据通路，使个处理单元能够高速、灵活、动态地与不同存储分体相连，使尽可能多的PE能无冲突地访问共享主存模块
- 对准网络（Alignment Network）



阵列处理机的特点

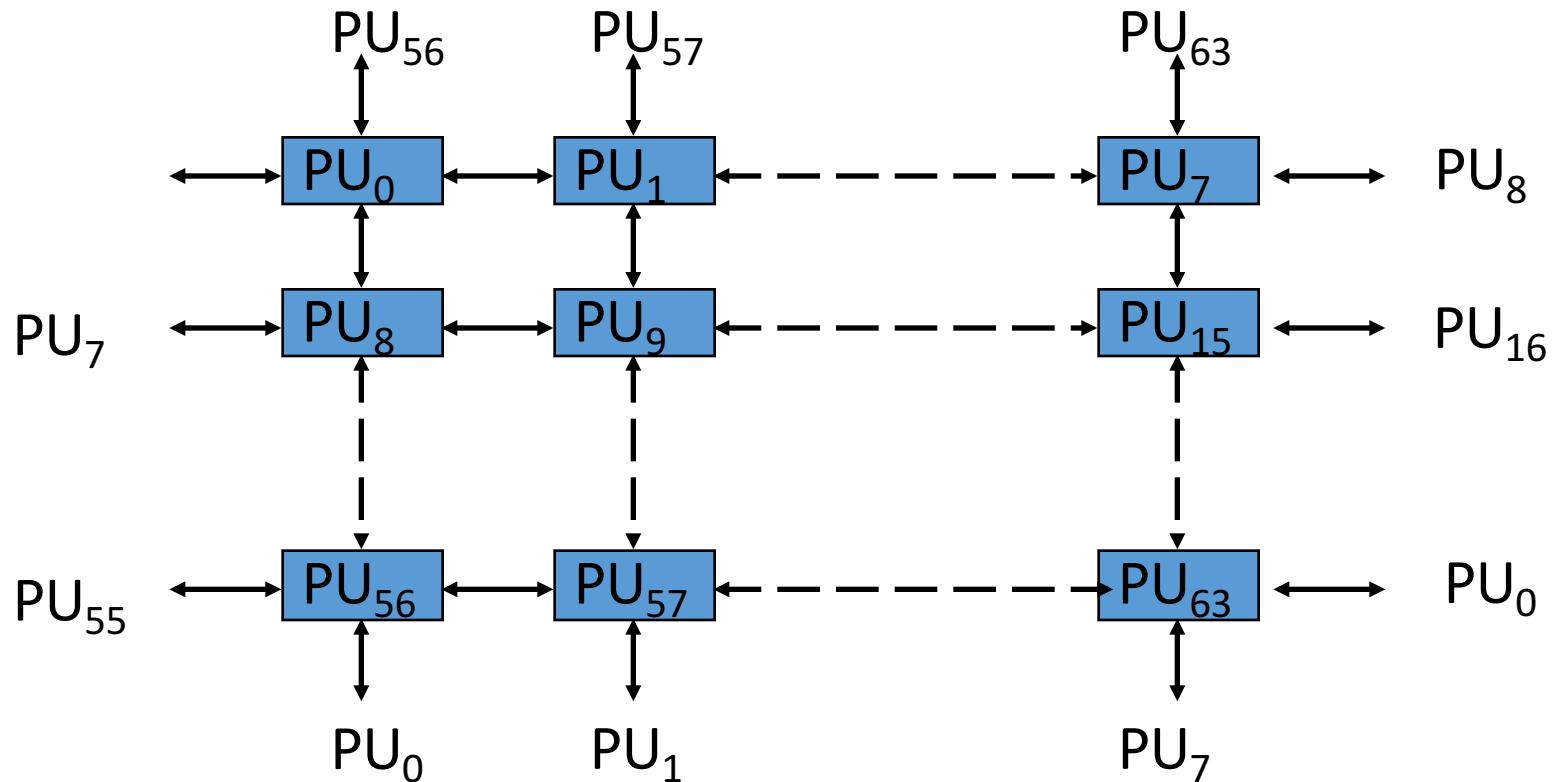
- 背景----科学计算
 - 有限差分、矩阵、信号处理、线性规划
 - 数组、向量处理
- 阵列处理机利用的是资源重复，而不是时间的重叠，利用并行性中的同时性，而不是并发性
- 阵列处理机使用简单而规整的互连网络来实现处理单元间的链接，互联网络的结构形式限定了阵列处理机可用的解题算法，会对系统的多种性能指标产生显著影响
- 阵列处理机在机间互连上比固定结构的单功能流水线灵活，因此在专门问题上的工作性能比流水线处理机高得多
- 与并行算法紧密联系，以提高阵列处理机解题算法的适应性和应用面

- 除了向量处理，阵列处理机还受到标量运算速度和编译开销的影响
 - 需要高性能、强功能的标量处理机，以减少标量运算对系统性能的影响
 - 建立具有向量化功能的高级语言编译程序
- 阵列处理机实质上是由
 - 专门对付数组运算的处理单元阵列组成的处理机
 - 专门从事处理单元阵列的控制及标量处理的处理机
 - 专门从事系统输入输出及操作系统管理的处理机

组成的一个**异构型多处理机系统**

ILLIAC IV的处理单元阵列结构

- 阵列处理机上并行算法的研究与结构紧密联系在一起
- 并行处理机处理单元阵列的结构又是适合于一定类型计算问题而专门设计的结构



闭合螺线阵列

特点

- 闭合螺线阵列
- 任意单元的最短距离不超过7步
- 一般来讲： $\sqrt{N} - 1$ 个处理单元组成的阵列中，任意两个处理单元之间的最短距离不会超过 $N = \sqrt{N} * \sqrt{N}$ 步
- 处理单元为通常的累加型运算器，把累加寄存器RGA中的数据和存储器来的数据进行操作

ILLIAC IV的并行算法举例

- 矩阵加
- 矩阵乘
- 累加和

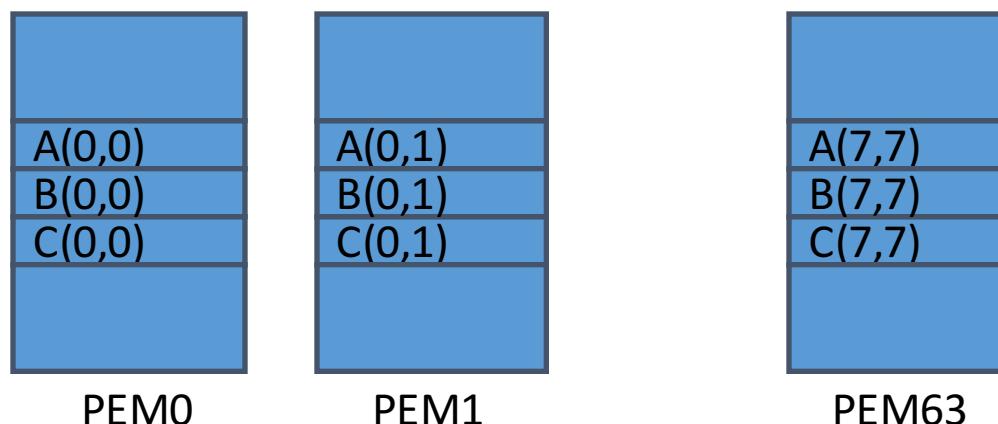
矩阵加

- 两个 8×8 矩阵相加，把分量放在每一个PEM内
- 算法：

LDA ALPHA ; 全部(α)由 PEM_i 送 PE_i 的累加器 RGA_i

ADRN ALPHA+1 ; 全部($\alpha+1$)与(RGA_i)浮点加，结果送(RGA_i)

STA ALPHA+2 ; 全部(RGA_i)由 PE_i 送 PEM_i 的 $\alpha+2$ 单元



矩阵乘

- 设A、B和C为三个8*8的二维矩阵

计算： $C = A * B$,

$$c_{ij} = \sum_{k=0}^7 a_{ik} * b_{kj}$$

矩阵乘 (续)

- SISD 算法:

```
DO 10 I=0,7
```

```
DO 10 J=0,7
```

```
C(I,J)=0
```

```
DO 10 K=0,7
```

```
10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

- 说明

- SISD 算法需 $8*8*8=512$ 次运算

矩阵乘 (续)

- SIMD算法：可用8个处理单元并行计算矩阵C(I,J)的某一行或一列

```
DO 10 I=0, 7
```

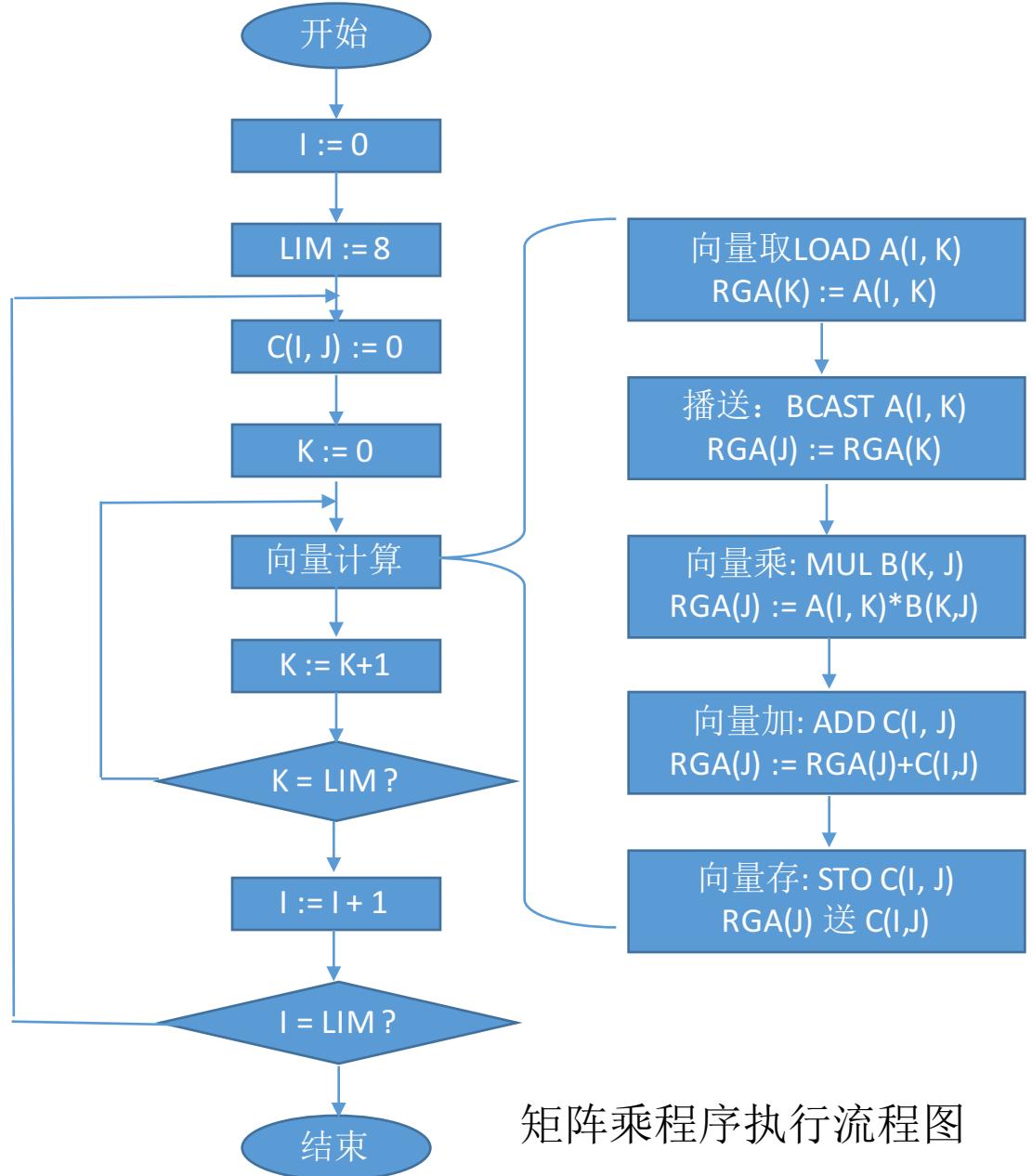
```
C(I,J)=0
```

```
DO 10 K=0,7
```

```
10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

- 说明

- 让 J = 0~7 各部分同时在PE0~PE7上运算，只需 $8*8=64$ 次运算



A(0,0)
A(1,0)
:
A(7,0)
B(0,0)
B(1,0)
:
B(7,0)
C(0,0)
C(1,0)
:
C(7,0)

PEM0

A(0,1)
A(1,1)
:
A(7,1)
B(0,1)
B(1,1)
:
B(7,1)
C(0,1)
C(1,1)
:
C(7,1)

PEM1

A(0,7)
A(1,7)
:
A(7,7)
B(0,7)
B(1,7)
:
B(7,7)
C(0,0)
C(1,7)
:
C(7,7)

PEM7

矩阵乘程序执行流程图

矩阵乘存储器分配举例

累加和

- 将N个数按顺序相加

$$C = \sum_{i=0}^7 a_i$$

- SISD算法（8次加法）：

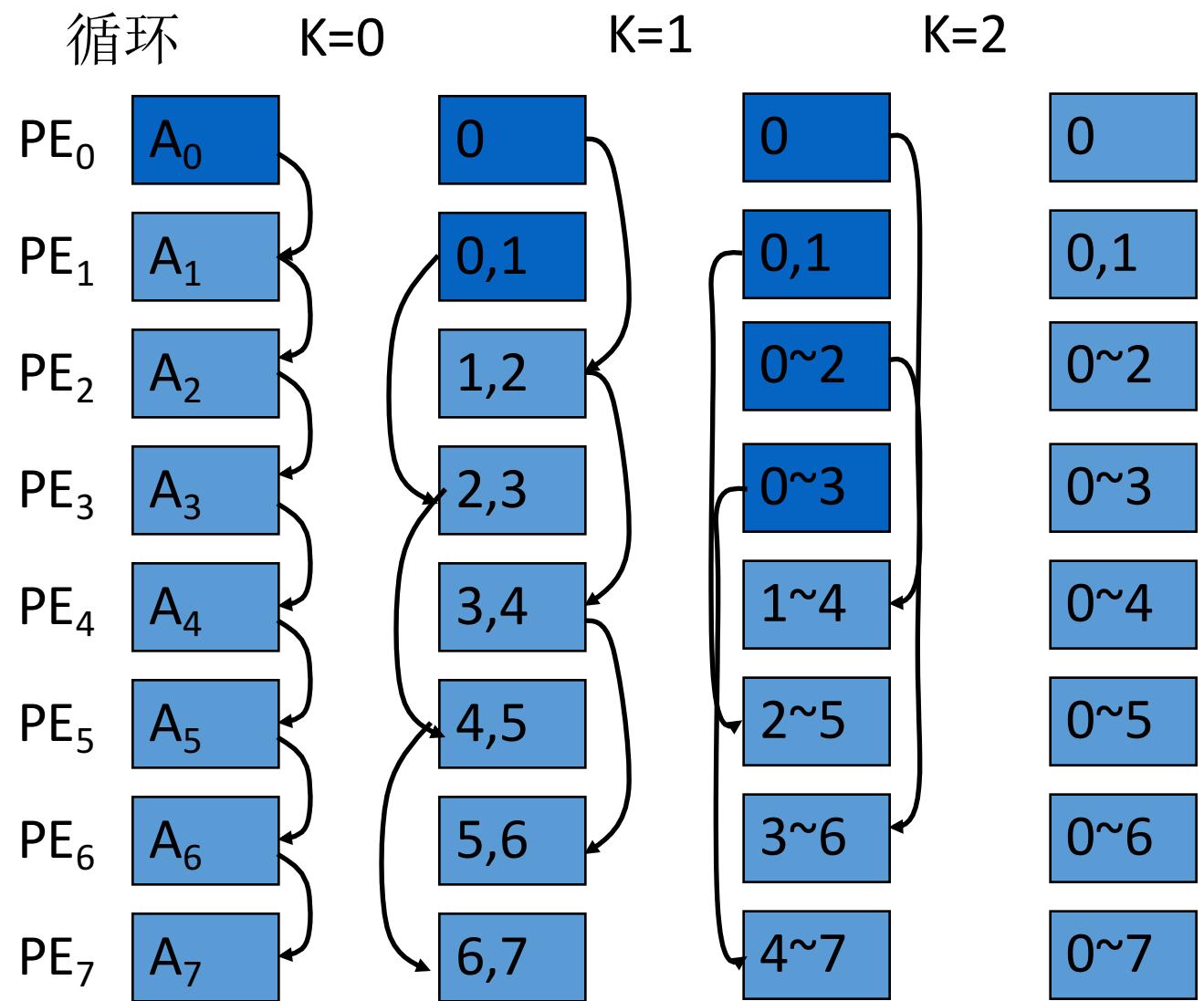
C=0

DO 10 I=0,7

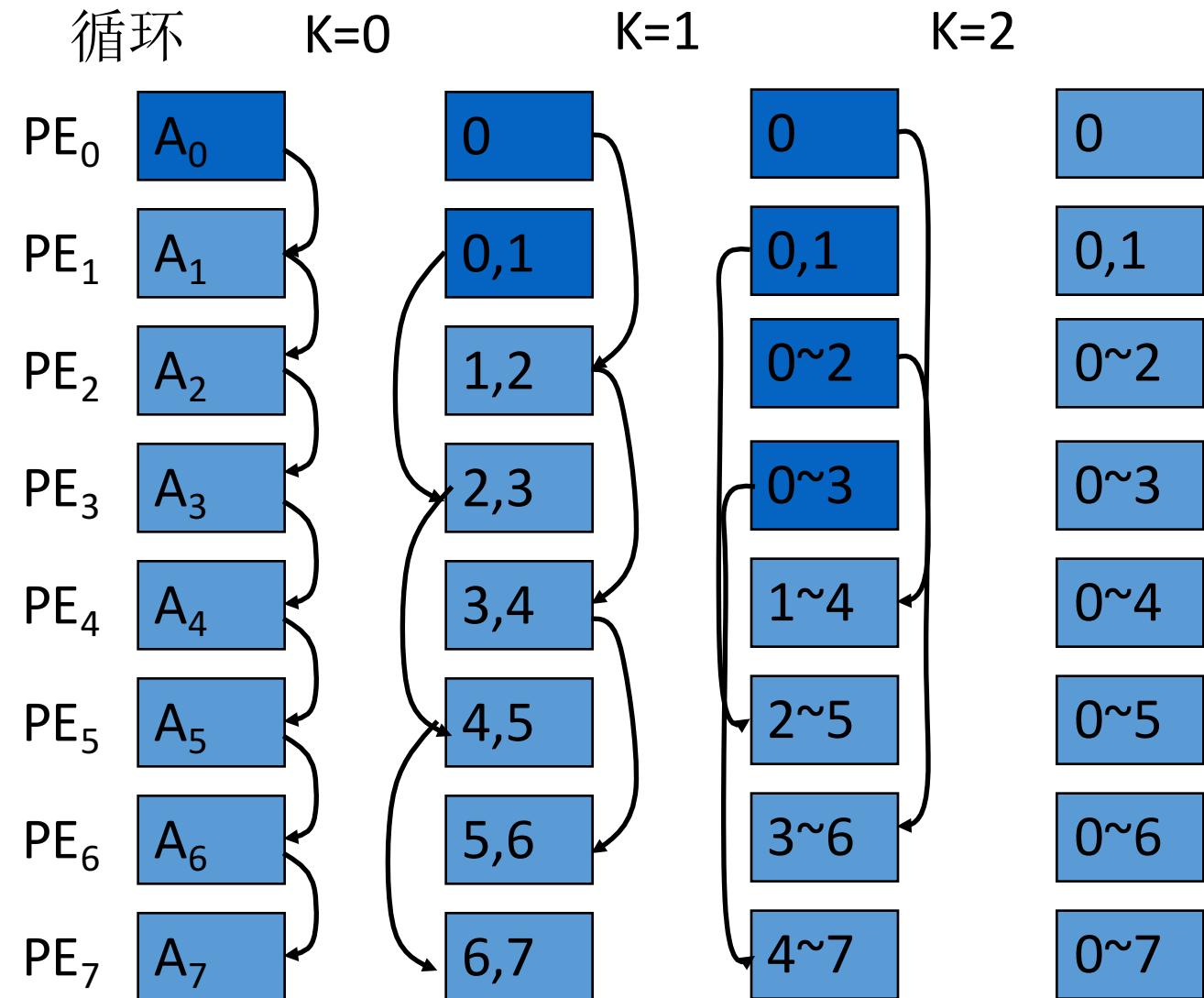
10 C=C+A(I)

- SIMD算法：递归相加

- 1) 置全部 PE_i 为活跃状态， $0 \leq i \leq 7$
- 2) 全部 $A(I)$ 从 PEM_i 的 α 单元读到相应 PE_i 的累加寄存器 RGA_i 中， $0 \leq i \leq 7$
- 3) 令 $k = 0$ ；
- 4) 将全部 PE_i 的 (RGA_i) 转送到传送寄存器 RGR_i ， $0 \leq i \leq 7$ ；
- 5) 将全部 PE_i 的 (RGR_i) 经过互连网络向右传送 2^k 步距， $0 \leq i \leq 7$ ；



- 6) 令 $j=2^{k-1}$
- 7) 置 PE_0 至 PE_j 为不活跃状态
- 8) 处于活跃状态的所有 PE_i 执行 $(RGA_i) := (RGA_i) + (RGR_i)$, $j < i \leq 7$;
- 9) $k := k+1$
- 10) 如 $k < 3$, 则转回第四步, 否则往下继续执行;
- 11) 置全部 PE_i 为活跃状态, $0 \leq i \leq 7$
- 12) 将全部 PE_i 的累加寄存器内容 (RGA_i) 存入相应 PEM_i 的 $\alpha+1$ 单元中, $0 \leq i \leq 7$ 。



- 需要 $\log_2 N$ 次加法, 速度提高 $N / \log_2 N$

互连网络的设计目标

- SIMD计算机中的处理单元之间，处理单元和存储分体之间，要通过互连网络进行信息交换
- 由于处理单元规模较大 ($2^{14} - 2^{16}$)，无法在任意处理单元之间设置数据通路，因此一般只让相邻单元之间有有限的直连方式，利用多跳传送，实现信息的传递
- 设计目标：
 - 结构不要过分复杂，以降低成本
 - 互连要灵活，以满足算法和应用需要
 - 处理单元之间的传送步数要少，以提高速度
 - 能用规整单一的基本构件组合而成，或者经多次通过，或者经多级链接来实现复杂功能，以提高模块性

互连函数

- $N = 2^n$ 个输入端, $N = 2^n$ 个输出端 ($0, 1, \dots, N - 1$)
- 互连函数可以定义为出端号和入端号的一一对应关系 ($j \rightarrow f(j)$)
- 互联网络的入端和出端实际上是同一组处理单元的出端和入端
- 互连函数可以用节点间的连线图表示, 但是显得繁琐, 也难以体现连接上的内在规律
- 函数式表示
 - x 为输入端变量, 用 n 位二进制形式来表示, $x_{n-1}x_{n-2} \cdots x_1x_0$
 - 互连函数表示为 $f(x_{n-1}x_{n-2} \cdots x_1x_0)$

互连网络的设置

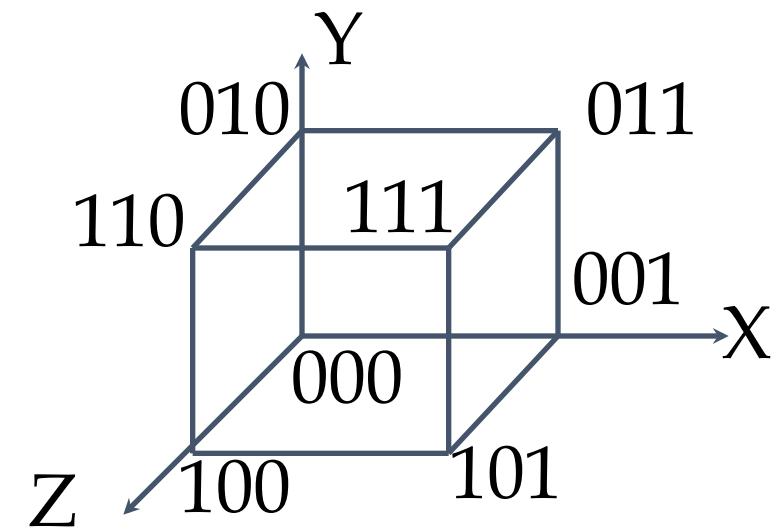
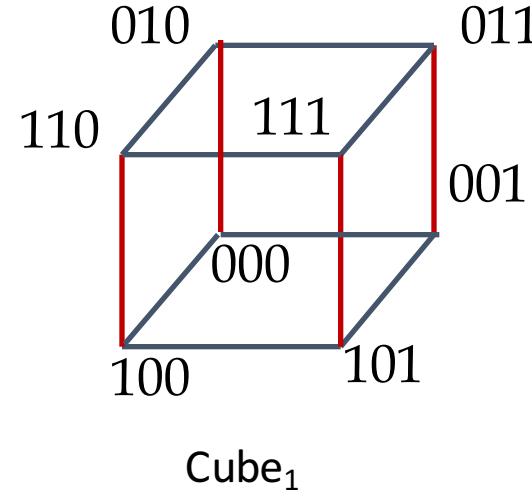
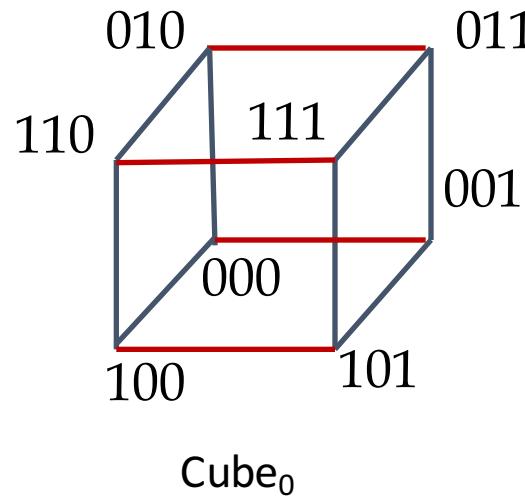
- 操作方式（一般采用同步方式）
 - 同步、异步、同步与异步的组合
- 控制策略（一般采用集中控制）
 - 互连网络由开关单元和互连线组成，互连通路的路径选择通过置定开关单元的工作状态来控制
 - 集中控制、分布式控制
- 交换方法（一般采用硬连的线路交换）
 - 线路交换：在源和目的之间建立实际连接通路
 - 包交换：将数据置于包内传送
 - 线路与包的组合交换

互连网络的设置

- 网络拓扑结构（互连网络入、出端可以连接的模式）
 - 静态： PE之间的连接是固定的
 - 动态： PE之间的连接是可以重新配置的
 - 单级（循环网络）： 只有有限几种连接， 需要经多次循环才能实现处理单元之间的信息传送
 - 多级： 由多个单级网络串联组成， 可实现实意两个处理单元之间的连接
 - 将多级互连网络循环使用， 可实现复杂互连， 称为循环多级网络或多级循环网络

立方体单级网络 (Cube)

- 每个顶点代表一个处理单元
- 使用 xyz 三位二进制编码
- 互连函数: $Cube_i$ 第*i*位互为反码
 - $Cube_i(p_{n-1}, \dots, p_i, \dots, p_1, p_0) = p_{n-1} \dots \bar{p}_i \dots p_1 p_0$
 - 共有 $n = \log_2 N$ 种互连函数
 - 最大传送距离 n
 - 任意两个节点之间至少有 n 条不同路径
 - $n > 3$ 时, 称为超立方体 (Hyper Cube) 网络



PM2I单级网络

- “加减 2^i ” (Plus-Minus 2^i) 单级网络的简称： j 号处理单元直接相连的是 $j \pm 2^i$ 号处理单元
- 互连函数：

$$PM2_{+i}(j) = (j + 2^i) \bmod N$$

$$PM2_{-i}(j) = (j - 2^i) \bmod N$$

- 说明
 - 共有 $2n$ 个互连函数，其中 $n = \log N$
 - 由于 $PM2_{+(n-1)} = PM2_{-(n-1)}$ ，因此只有 $2n - 1$ 种互连函数是不同的
 - 最大距离是 $[N/2]$

PM2I单级网络（续）

- 当N=8时，有 $n=\log_2 N$, 2n=6个互联函数

PM2₊₀: (0 1 2 3 4 5 6 7)

PM2₋₀: (7 6 5 4 3 2 1 0)

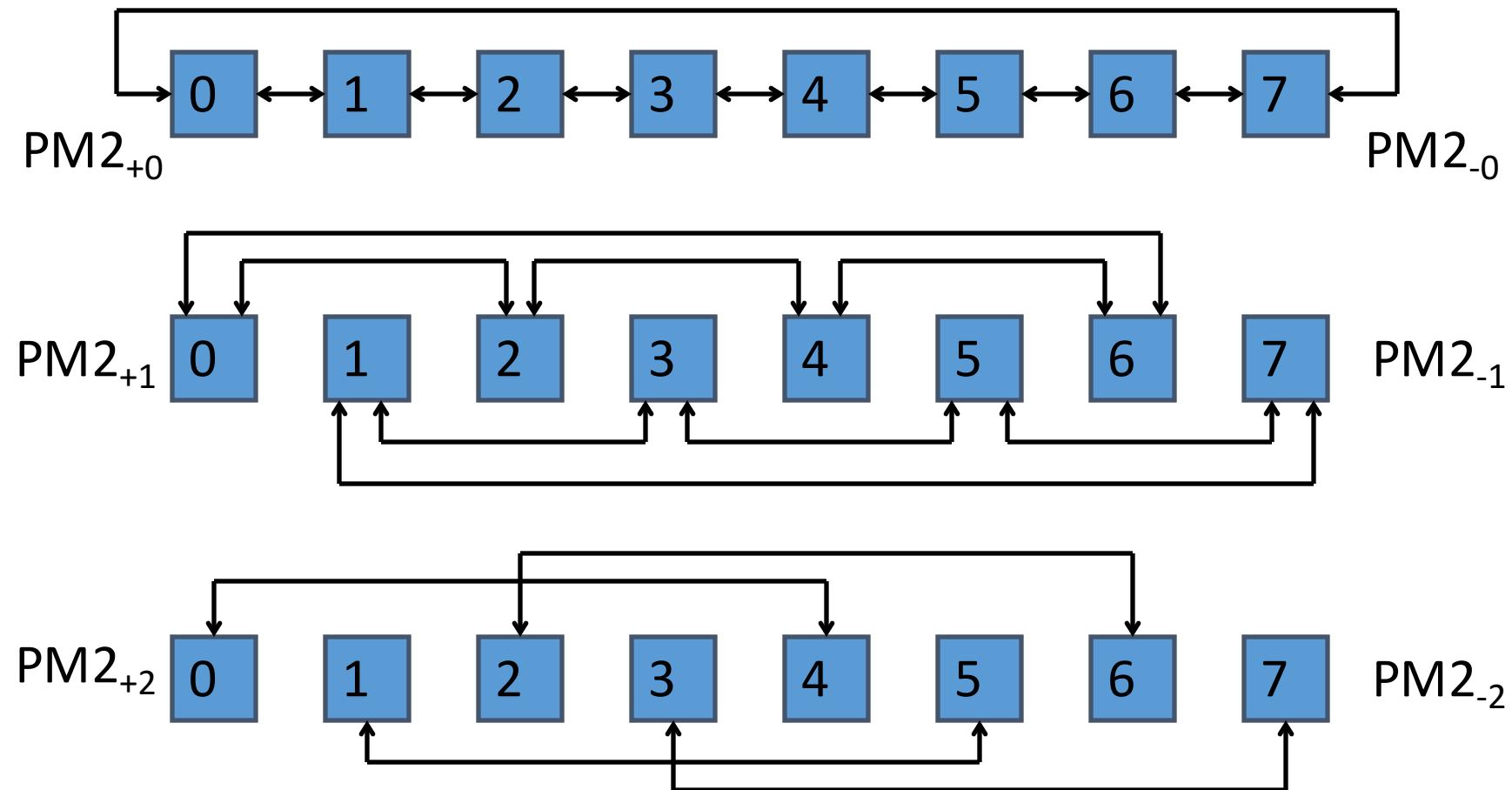
PM2₊₁: (0 2 4 6)(1 3 5 7)

PM2₋₁: (6 4 2 0)(7 5 3 1)

PM2₊₂: (0 4)(1 5)(2 6)(3 7)

PM2₋₂: (4 0)(5 1)(6 2)(7 3)

PM2I单级网络（续）



混洗交换单级网络

- 包含两个函数：混洗、交换

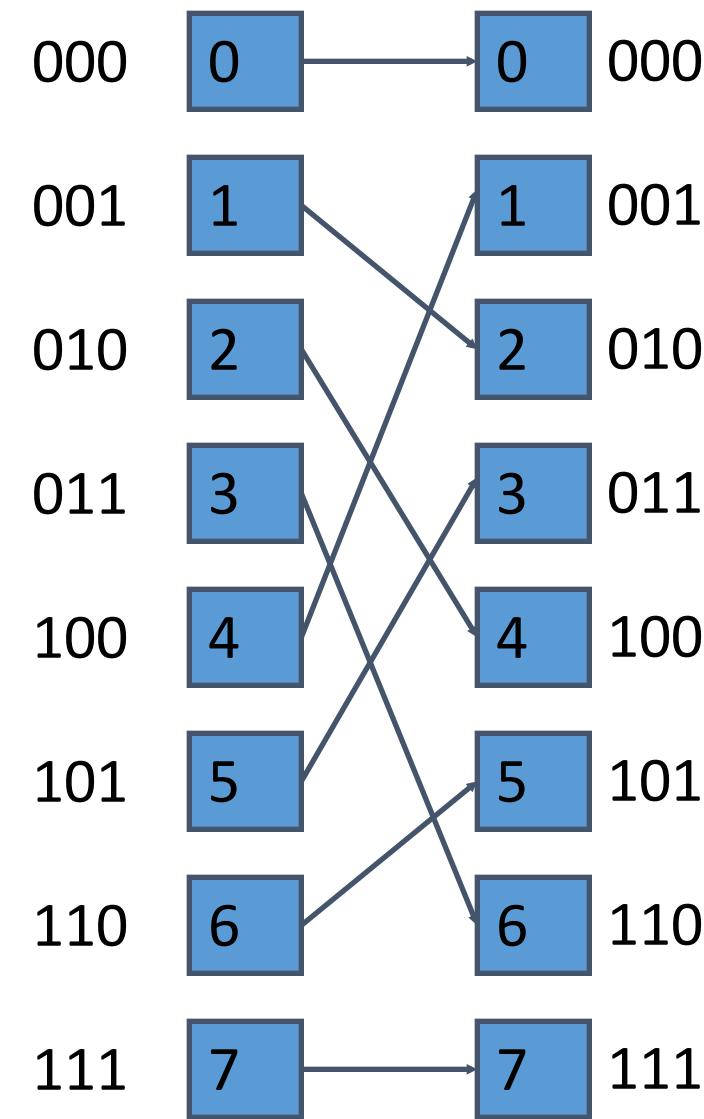
$$Shuffle(P_{n-1}P_{n-2} \cdots P_1P_0) = P_{n-2} \cdots P_1P_0P_{n-1}$$

- 说明：

- 不是可逆函数
- 特性：作 n 次后，恢复到原来，因此多次混洗后，每个处理器都会遇到与其他处理器连接的机会（除全0和全1）；
- 增加交换函数，得到全混交换单级网络；
- 全混连接与立方体连接存在对应关系，此性质便于构成多级连接，并与立方体具有相似的关系；
- 最大距离为 $2n - 1$

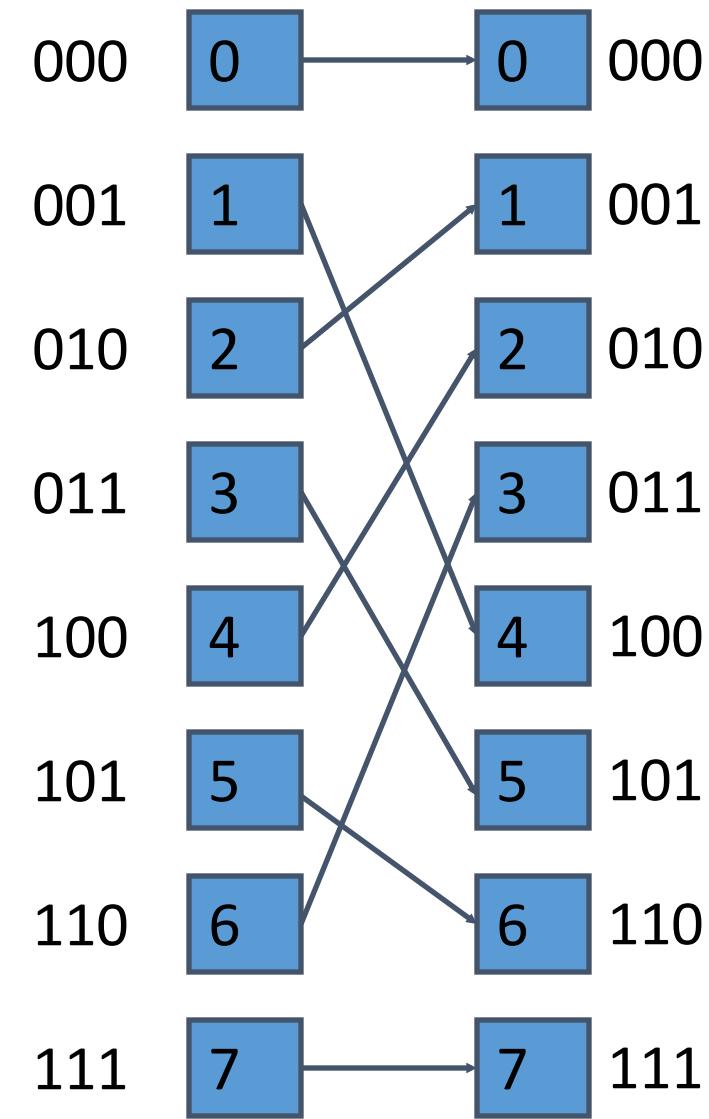
- 0: 000 -> 0: 000
- 1: 001 -> 2: 010
- 2: 010 -> 4: 100
- 3: 011 -> 6: 110
- 4: 100 -> 1: 001
- 5: 101 -> 3: 011
- 6: 110 -> 5: 101
- 7: 111 -> 7: 111

一次混淆



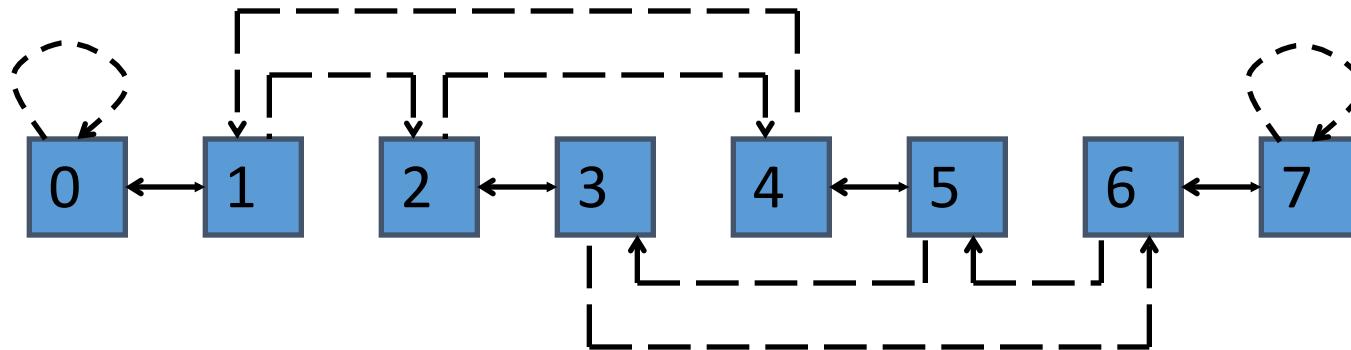
- 0: 000 -> 0: 000 -> 0: 000
- 1: 001 -> 2: 010 -> 4: 100
- 2: 010 -> 4: 100 -> 1: 001
- 3: 011 -> 6: 110 -> 5: 101
- 4: 100 -> 1: 001 -> 2: 010
- 5: 101 -> 3: 011 -> 6: 110
- 6: 110 -> 5: 101 -> 3: 011
- 7: 111 -> 7: 111 -> 7: 111

二次混淆



全混交换互连网络

- 单纯的全混互连网络不能实现“全0”和“全1”的处理单元与其他处理单元的连接，因此还需要增加 $Cube_0$ 交换函数

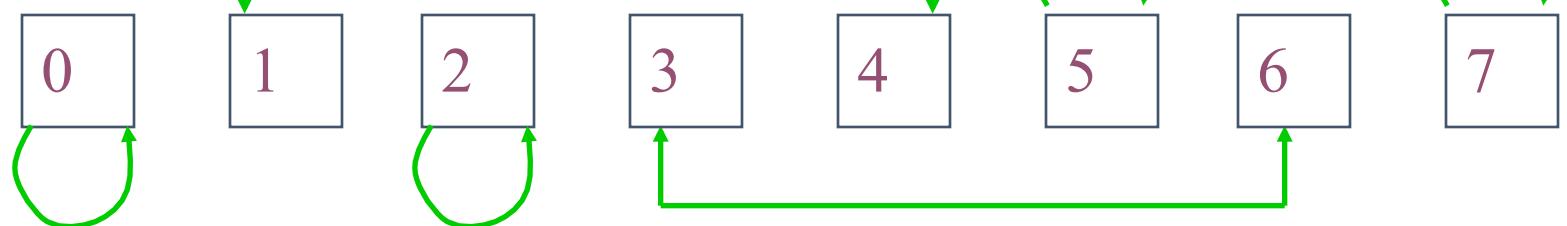
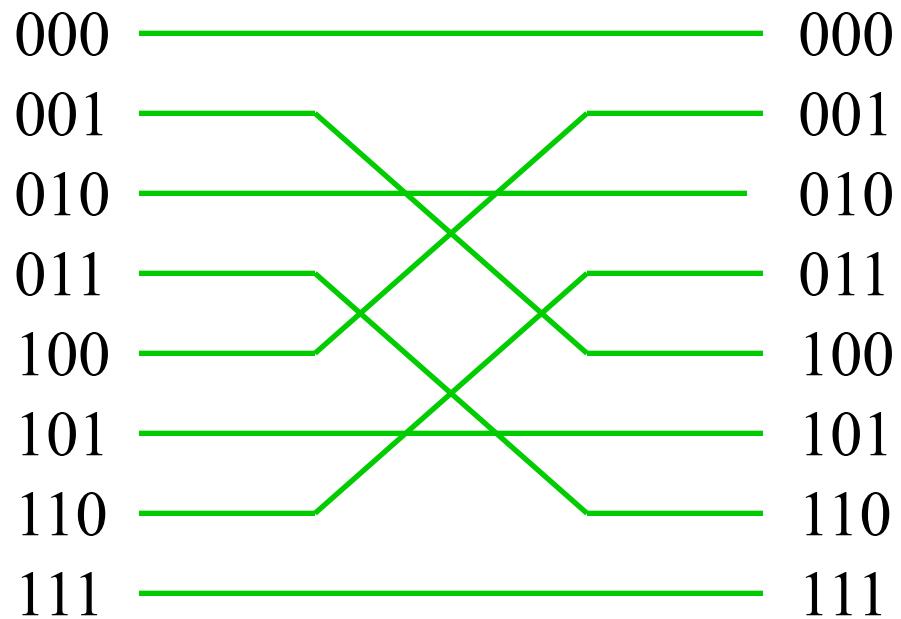
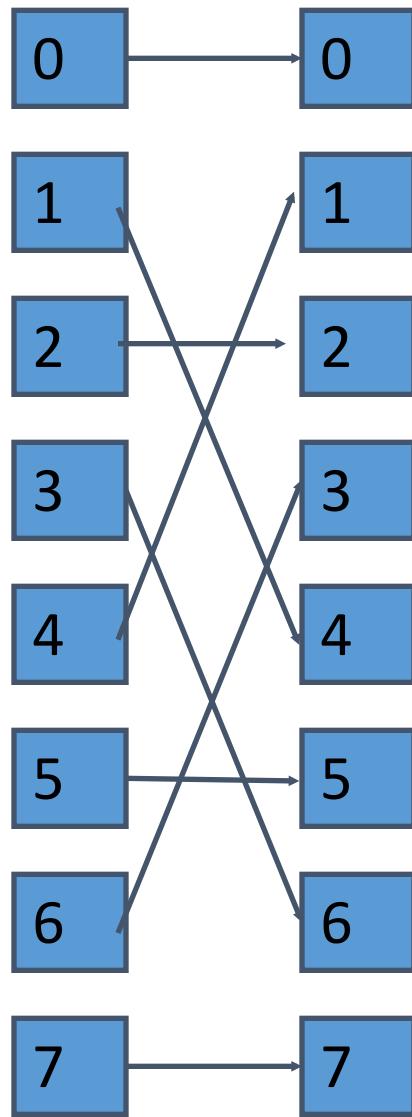


蝶形单级网络 (Butterfly)

- 互连函数

$$Butterfly(P_{n-1}P_{n-2} \cdots P_1P_0) = P_0P_{n-2} \cdots P_1P_{n-1}$$

- 即将二进制的最高位和最低位相互交换位置。

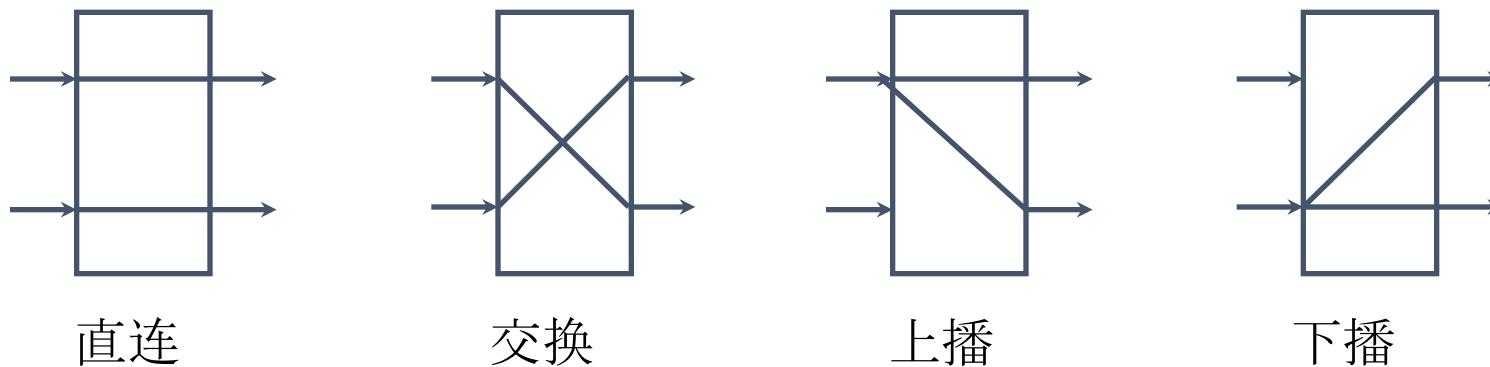


基本多级互连网络

- 能够实现结点到结点之间的任意互连是互连网络的一种基本功能。
- 多级互连网络采用多个相同的或不同的互连网络直接连接起来。属于组合逻辑线路，一个时钟周期就能够实现任意结点到结点之间的互连。
- 多级互连网络采用的关键技术：
 - 交换开关
 - 交换开关之间的拓扑连接
 - 对交换开关的不同控制方式

交换开关

- 一个 $a \times b$ 交换开关有 a 个输入和 b 个输出。
- 最常用的二元开关: $a=b=2$ 。



- 具有直通和交换两种功能的交换开关称为二功能开关，或交换开关。用一位控制信号控制。
- 具有所有四种功能的交换开关称为四功能开关，用两位控制信号控制。

拓扑结构

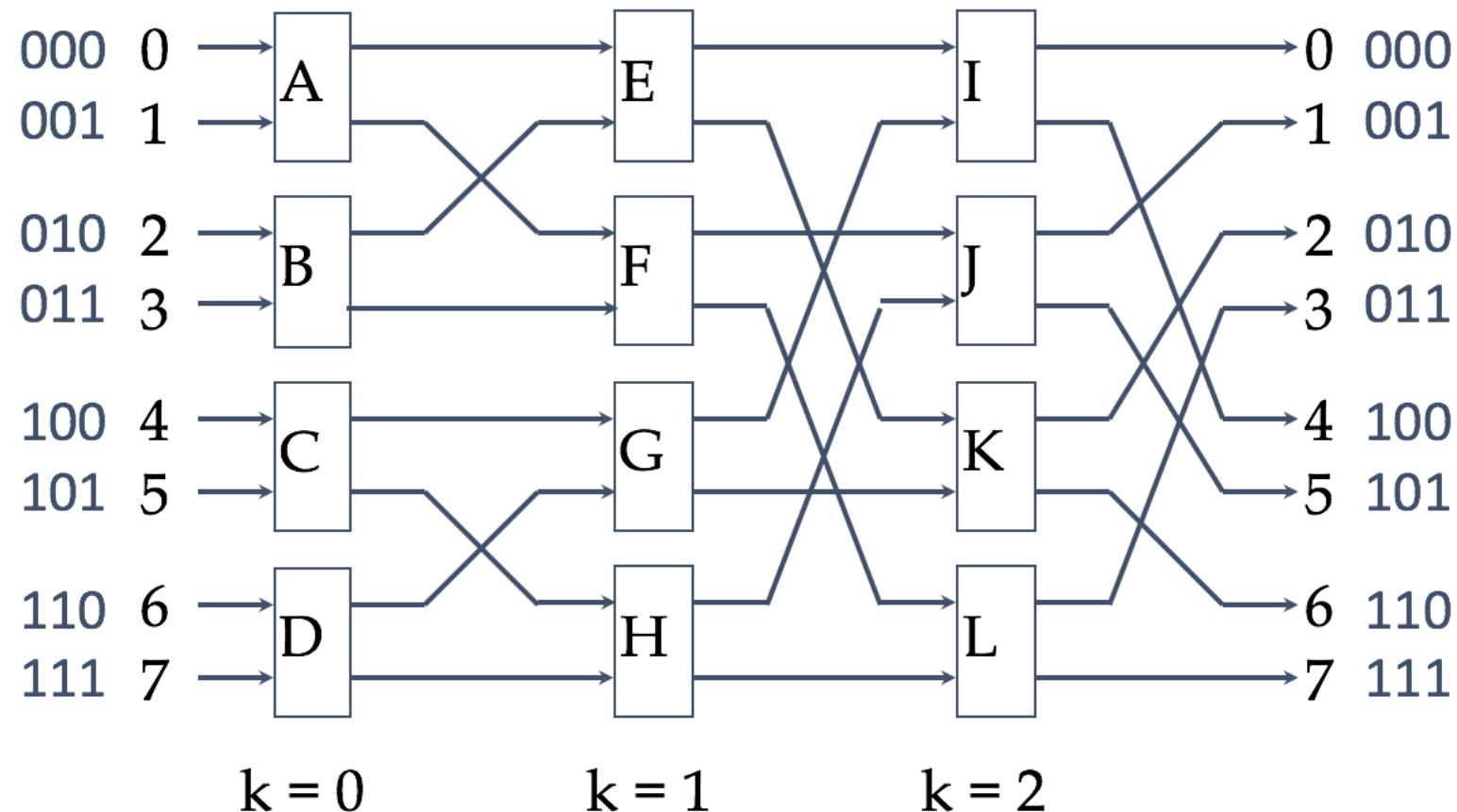
- 拓扑结构是各级间出端与入端互连的模式
- 单级互连网络的连接模式可以用来组合成不同的多级互连网络
- 控制方式是对各个交换开关进行控制的方式
 - 多级立方体的控制方式：
 - 1) 级控制：同级别的所有开关只用一个控制信号控制，同时只能处于同一种状态
 - 2) 单元控制：每一个开关都由独立的控制信号控制，可各自处于不同的状态
 - 3) 部分级控制：第*i*级别的所有开关分别用*i+1*个信号控制
- 利用交换开关、拓扑结构和控制方式三个参量，可以描述各种多级互连网络结构

基本多级互连网络

- 多级立方网络(Single Stored Cube Network)
- 多级混洗交换网络
- 多级PM2I网络(Plus-minus 2^i)
- 基准网络
- 多级交叉开关网络
- 多级碟式网络

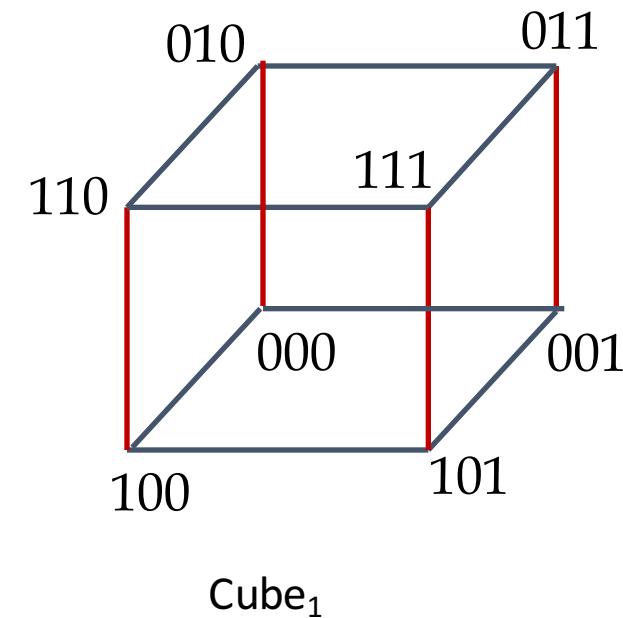
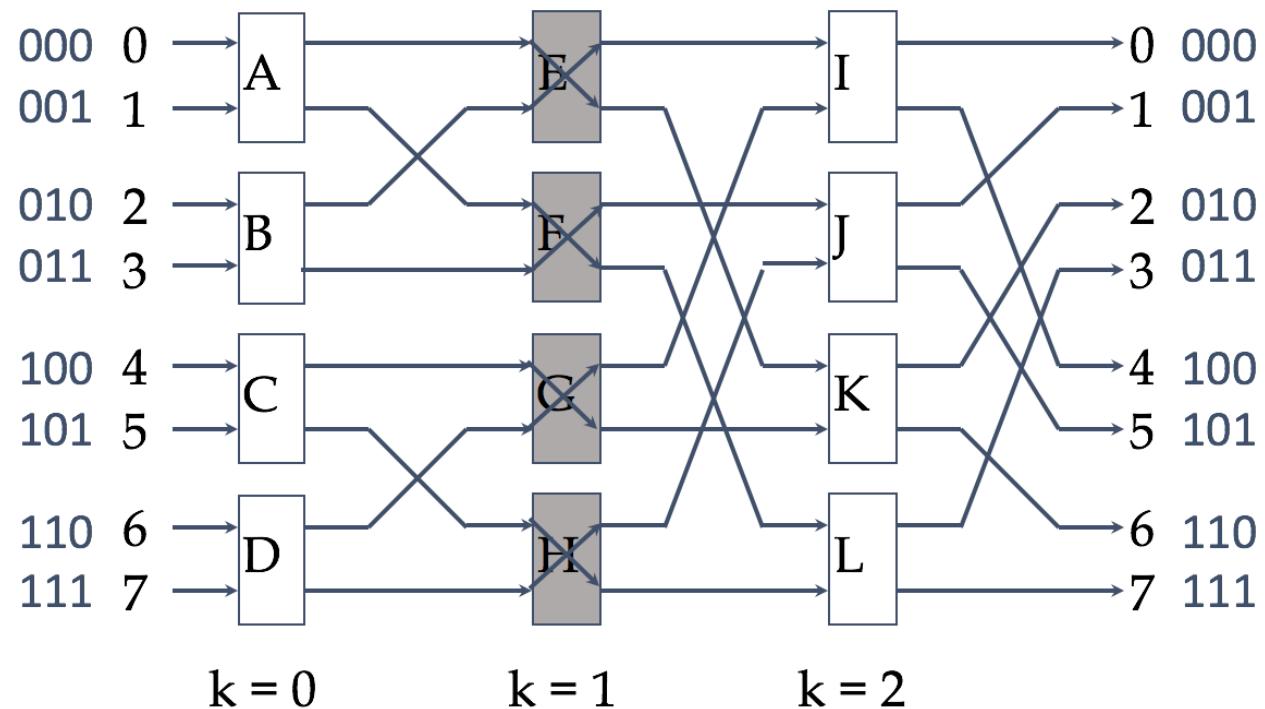
多级立方体网络

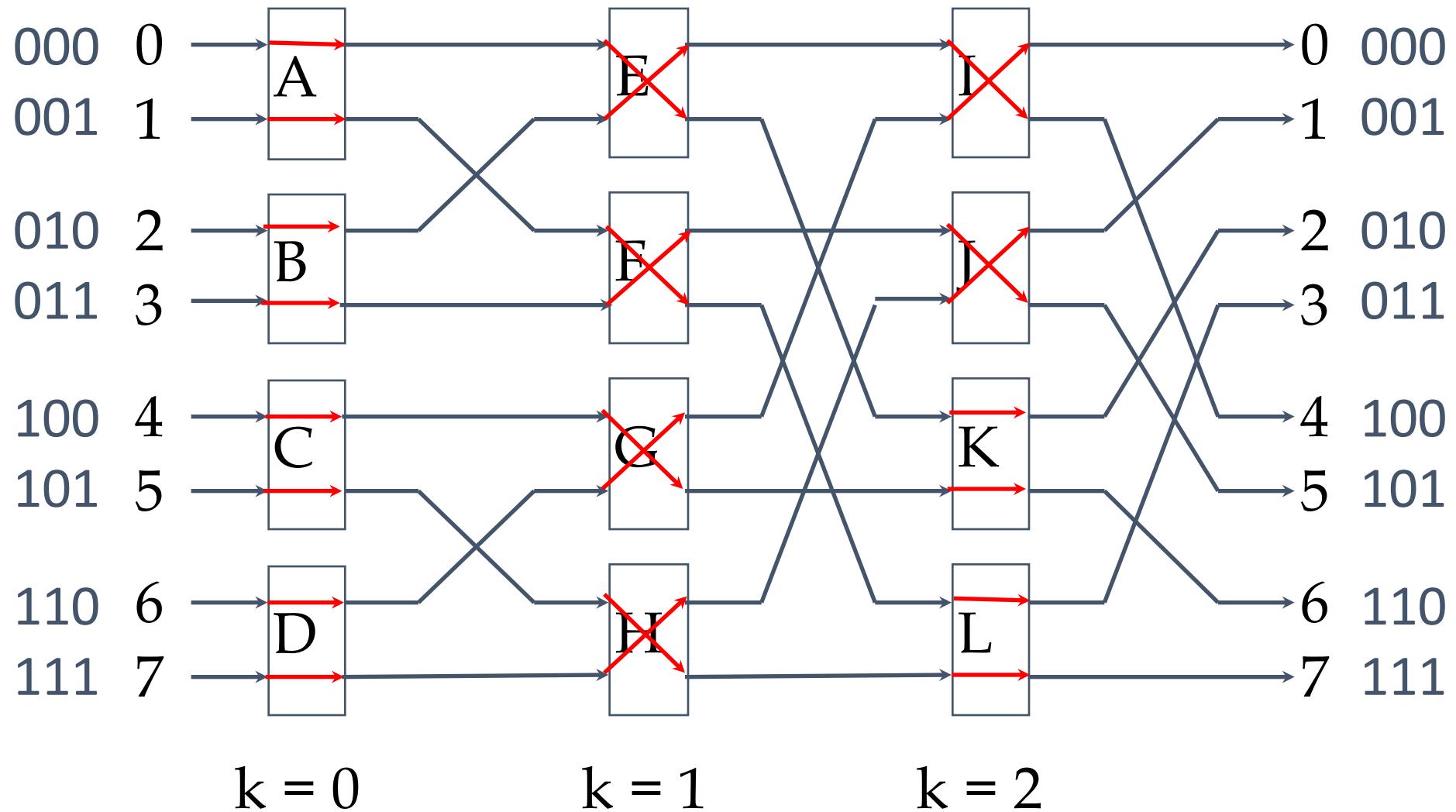
- 多级立方体互连网络有STARAN网络、间接二进制n立方体网络等



多级立方体网络

- STARAN网络采用级控制和部分级控制，而间接二进制 n 立方体网络使用单元控制，后者的连接灵活性更强
- 第*i*级交换单元处于交换状态时，实现的是Cube_{*i*}互连函数





多级立方体网络（续）

- 采用三种不同的控制方式，可以构成三种不同的互连网络。
 - 采用级控制可以构成STARAN交换网。
 - 采用部分级控制，可以构成STARAN移数网。
 - 采用单元控制可以构成间接二进制 n 方体网。

交换功能

		级控制信号 ($k_2k_1k_0$)							
		000	001	010	011	100	101	110	111
入端	0	0	1	2	3	4	5	6	7
	1	1	0	3	2	5	4	7	6
	2	2	3	0	1	6	7	4	5
	3	3	2	1	0	7	6	5	4
	4	4	5	6	7	0	1	2	3
	5	5	4	7	6	1	0	3	2
	6	6	7	4	5	2	3	0	1
	7	7	6	5	4	3	2	1	0
功能	i	Cube ₀	Cube ₁	Cube ₀	Cube ₂	Cube ₀	Cube ₁	Cube ₀	
				+		+	+	+	
				Cube ₁		Cube ₂	Cube ₂	Cube ₁	
								+	
								Cube ₂	

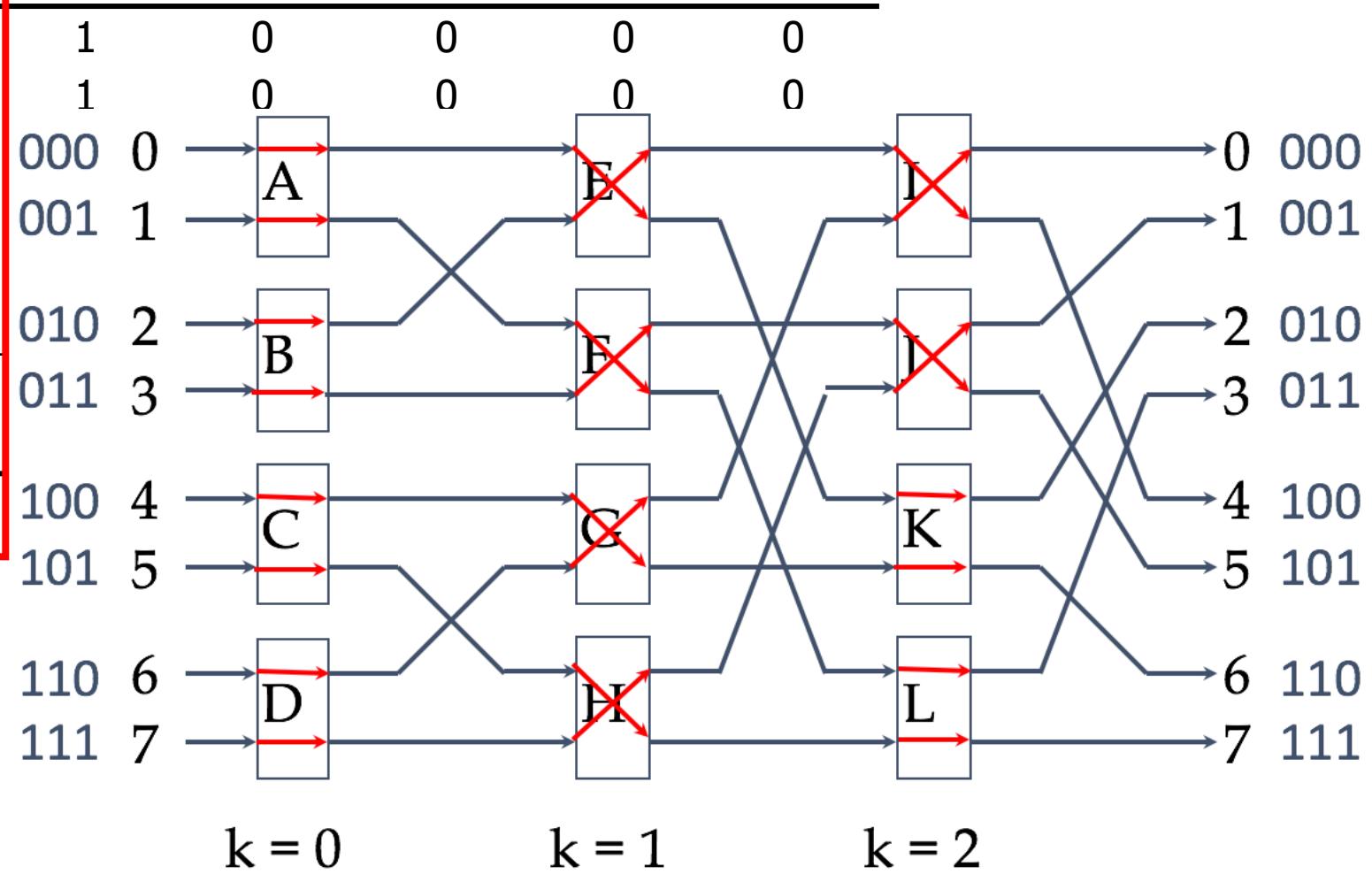
- 控制信号111, 1组8元交换
 - 入端排列: 0 1 2 3 4 5 6 7
 - 出端排列: 7 6 5 4 3 2 1 0
- 控制信号001, 4组2元交换
 - 入端排列: 01 23 45 67
 - 出端排列: 10 32 54 76
- 控制信号010, 4组2元交换+2组4元交换
 - 入端排列: 01 23 45 67
 - 10 32 54 76
 - 出端排列: 2301 6745

移位功能

2级	K,L J I	0	0	1	0	0	0
		0	1	1	0	0	0
		1	1	1	0	0	0
1级	F,H	0	1	0	0	1	0
	E,G	1	1	0	1	1	0
0级	A,B,C,D	1	0	0	1	0	1
功 能		移1 Mod 8	移2 Mod 8	移4 Mod 8	移1 Mod 4	移2 Mod 4	移1 Mod 2
							不移 衡等

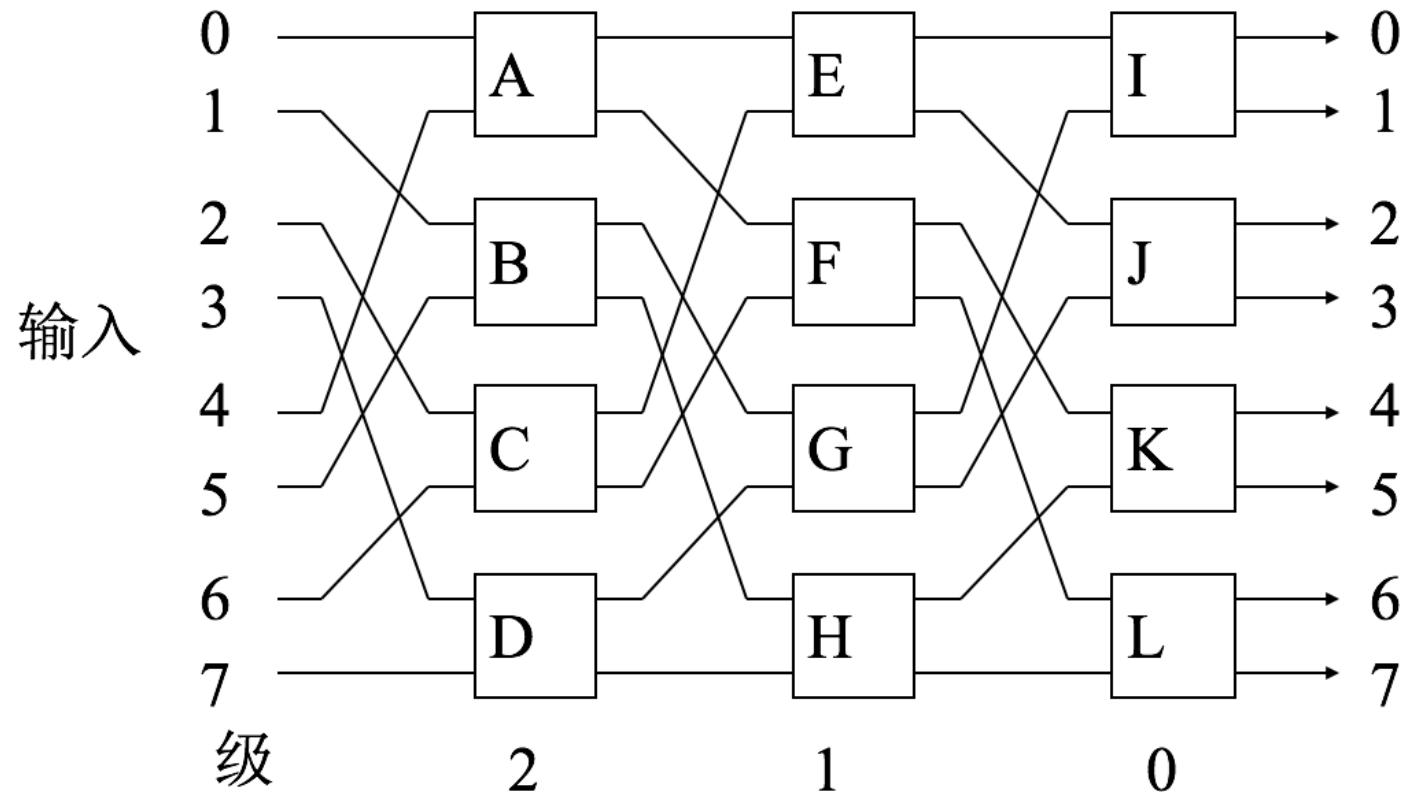
移位功能

2级	K,L J I	0 0 1 1 1	0 1 1 1 0
1级	F,H E,G	0 1	0 1
0级	A,B,C,D	1	0
功 能	移1 Mod 8	移2 Mod 8	

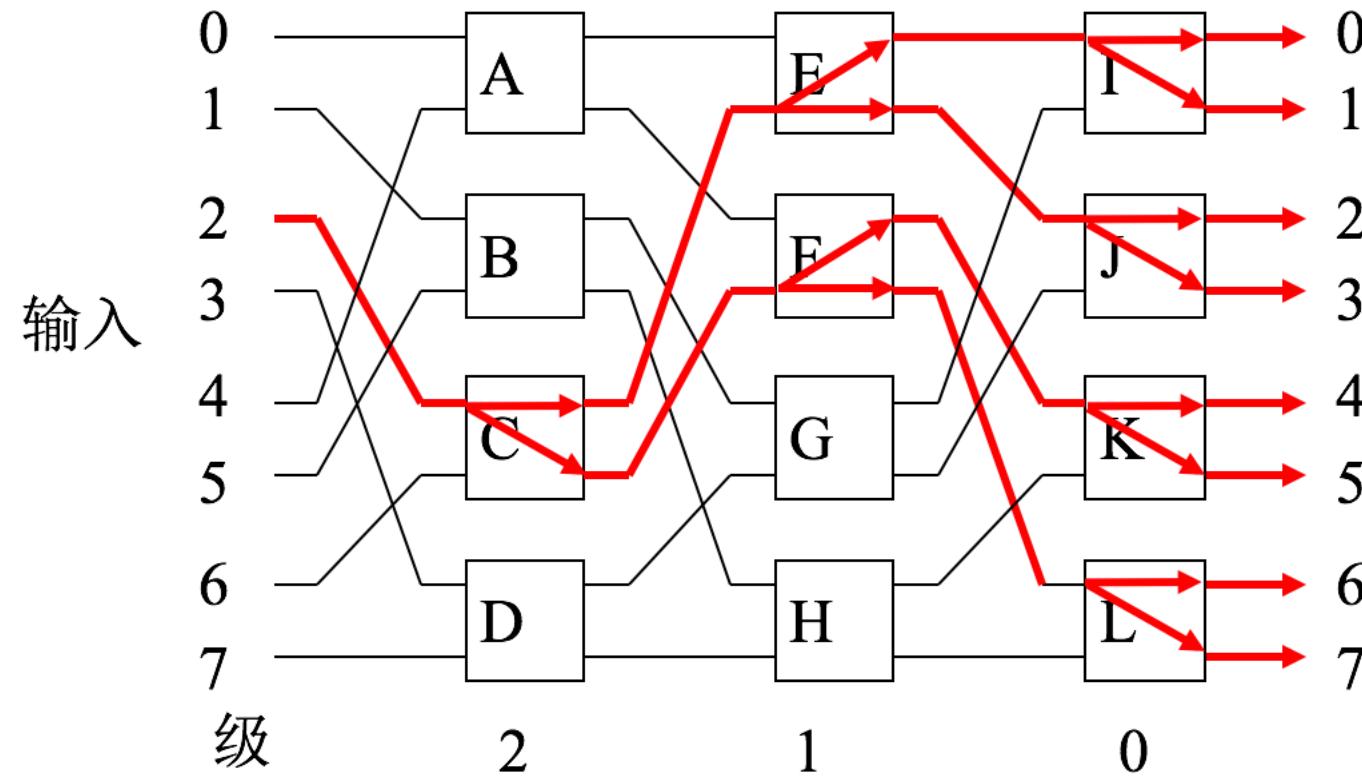


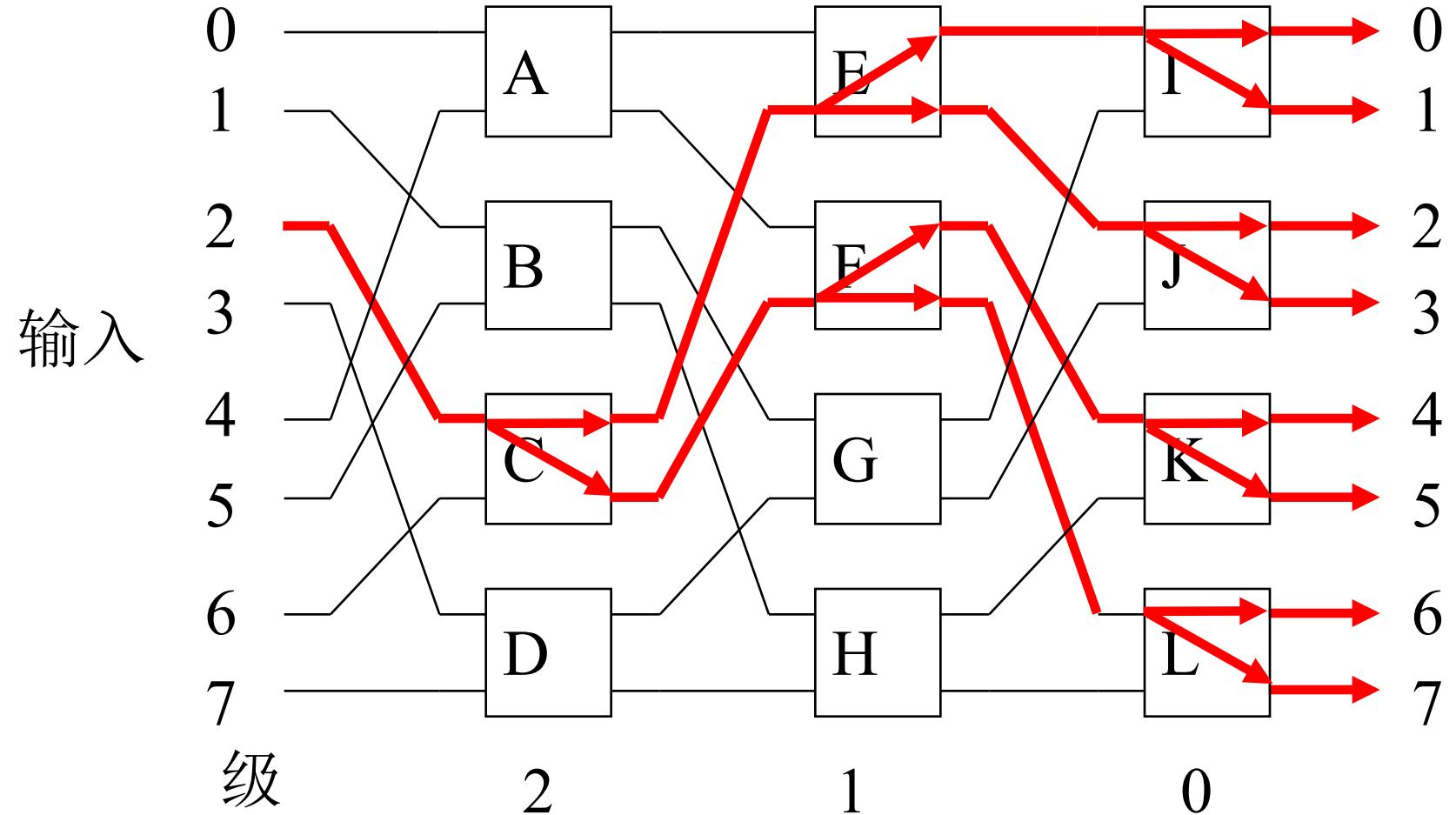
多级混洗交换网络

- omega网络，由N级相同的网络组成，每一级包含一个全混拓扑和一列 2^{n-1} 个四功能交换单元，采用单元控制方式



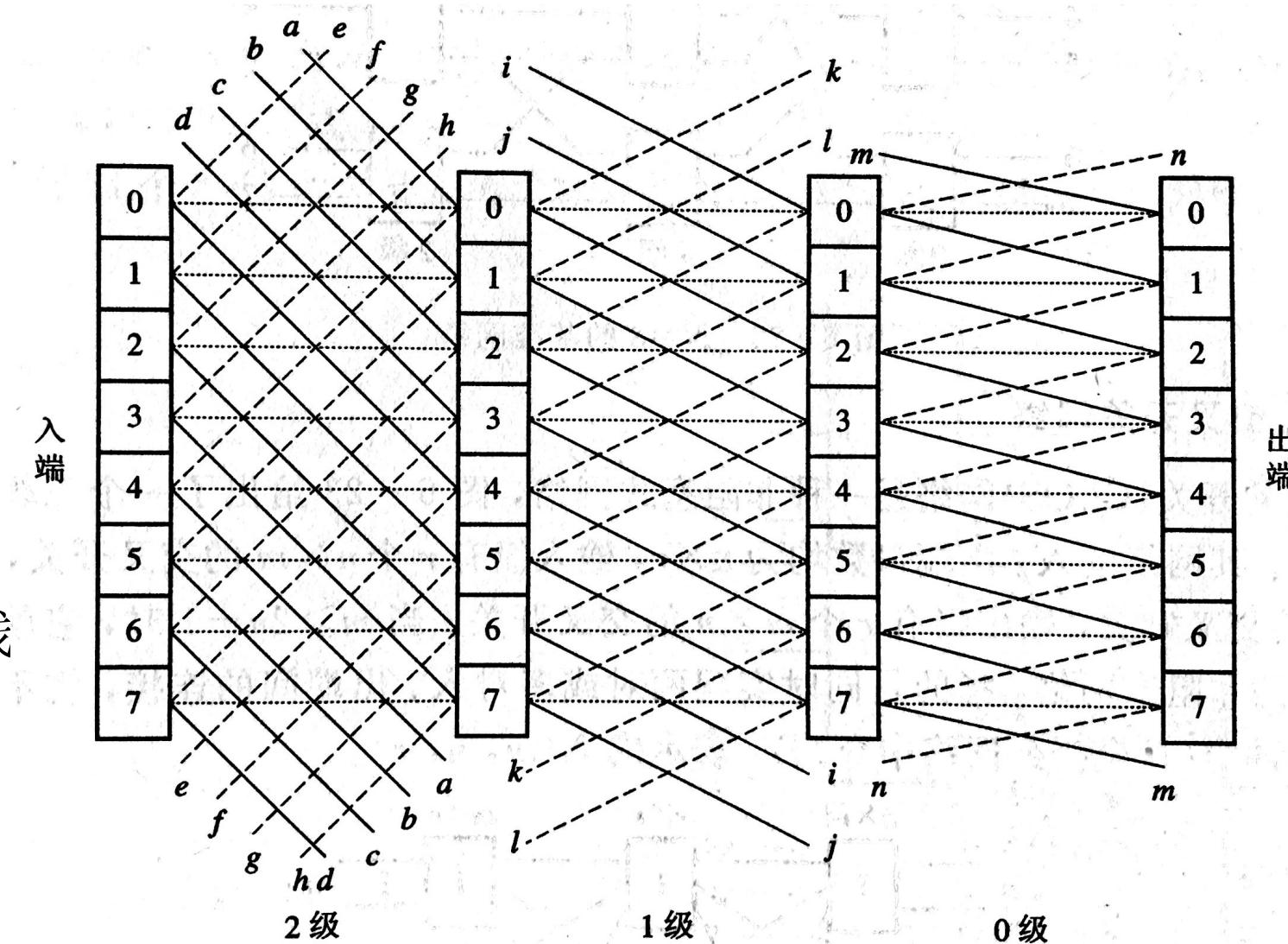
- 若omega网络采用二功能交换单元，可以看成是n立方体网络的逆网络
- 由于omega网络采用四功能交换单元，可以实现同一处理单元与多个处理单元的连接





多级PM2I网络(Plus-minus 2^i)

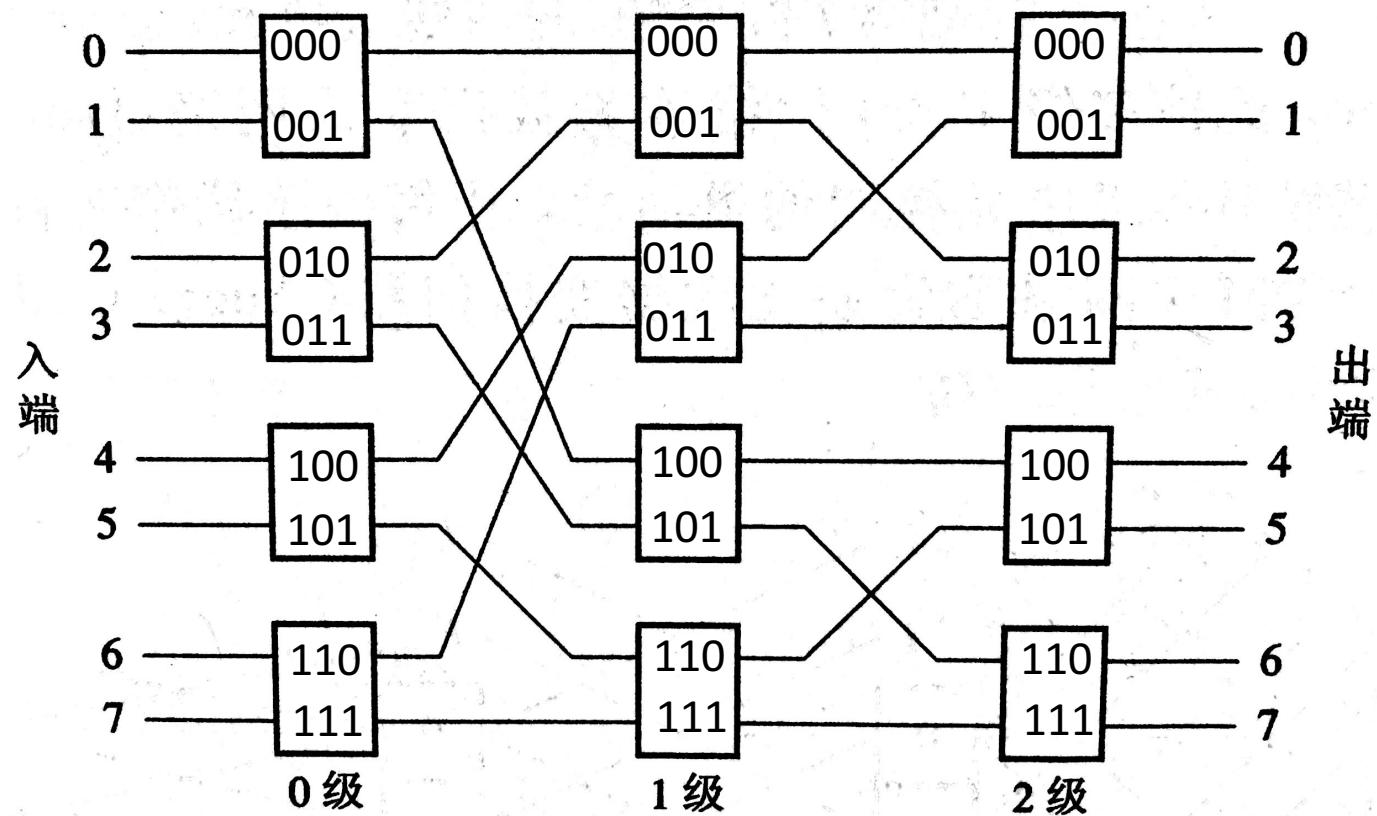
- 包含n级单元间连接，每一级都是把前后两列各 $N=2^n$ 个单元按PM2I拓扑相互连接起来
- 对第*i*级，每一个入单元*j*分别连接到 j （点线）、 $j + 2^i \bmod N$ （实线）和 $j - 2^i \bmod N$ （虚线）



- 冗余路径：两个PE之间可以有多条路径
- 数据变换网络 Data Manipulator, DM
 - 控制三种连接线的信号分别为平控H，下控D和上控U
 - 对于第*i*级，让 H_1^i 、 D_1^i 、 U_1^i 控制第*i*位为“0”的入单元，让 H_2^i 、 D_2^i 、 U_2^i 控制第*i*位为“1”的入单元
- 强化数据变换网络 Augmented Data Manipulator, ADM
 - 采用单元控制增强各级单元控制的灵活性
 - 每一级单元都有独立的控制信号
 - 控制线多，成本高

基准网络

- 与二进制立方体网络的逆网络相似，只是第1级的级间连接不同
- 从输入到输出的级间连接为恒等、逆全混、子逆全混和恒等
- 采用单元控制
- 作为中间介质，模拟一种网络的拓扑和功能



总结

- 单级互连网络特性
 - 任一单级互连网络均可表示成 $N_{\text{入}}, N_{\text{出}}$ 的过程。
 - 任一单级互连网络可实现部分结点(一对或几对)间的连接，不能实现在任意多对结点间的同时连接。
 - 单级互连网络含义：某些连接方法或拓扑结构。
- 单级互连网络应用
 - 利用单级互连网络的特性作为实际IN的拓扑结构；
 - 通过交换开关作为IN的可变因素；
 - 通过交换开关多次控制实现IN的结点间任意互连。