第 4 章 向量处理机

向量处理机： 具有向量数据表示的处理机

- 向量流水处理机——时间重叠
- 阵列处理机 ——资源重复

4.1 向量的流水处理与向量流水处理机

4.1.1 向量的流水处理

向量的流水处理：选择使向量运算最能充分发挥出流水线性能的处理方式。

？思考：要计算$D=A*(B+C)$，其中，A、B、C、D都是具有N个元素的向量，应该采用什么样的处理方式才能最充分发挥流水线的效能呢?

向量的几种处理方式：以D=A*（B+C）为例

① 横向处理方式

逐个求D向量元素 ：bi+ci-〉k，k*ai-〉di

不是流水处理方式，存在相关

② 纵向处理方式

先求所有的K=（B+C），再求所有的A*K

是流水处理，需要每拍取得成对元素

③ 分组纵横处理方式

把该向量分割成若干个组，使每个组都能装入向量寄存器中，每组按纵向处理，组间采用软件方法编制循环程序的方式依次循环处理。

解决主存与流水线速度不匹配问题

4.1.2 向量流水处理机的结构

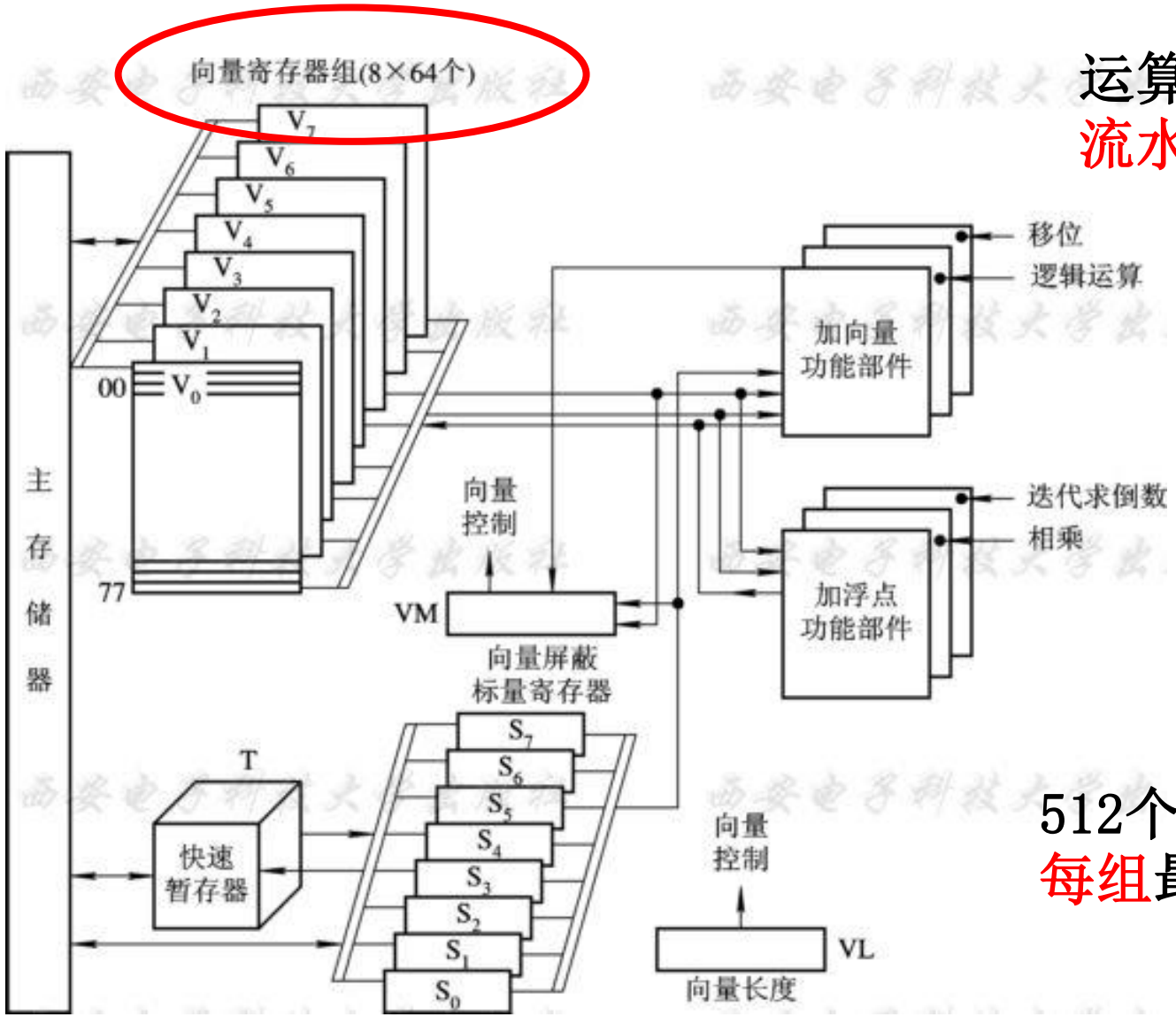
向量流水处理机（Vector Processor）是指将向量数据表示与流水线技术相结合，能进行向量流水处理的处理机。

1. 向量处理机的指令系统

一般应包含有向量型和标量型两类指令。向量型运算类指令一般又可以有如下几种：

- 向量V1运算得向量V2， 如V2=SIN(V1)；
- 向量V运算得标量S， 如 $S=\sum_{i=1}^{n}V_i$ ；
- 向量V1与向量V2运算得向量V3，如V3=V1∧V2；
- 向量V1与标量S运算得向量V2， 如V2=S*V1。

向量流水处理机的结构举例——CRAY-1的向量流水处理部分简图



运算部分共有12条可并行的单功能流水线，仅与向量运算有关的6条：

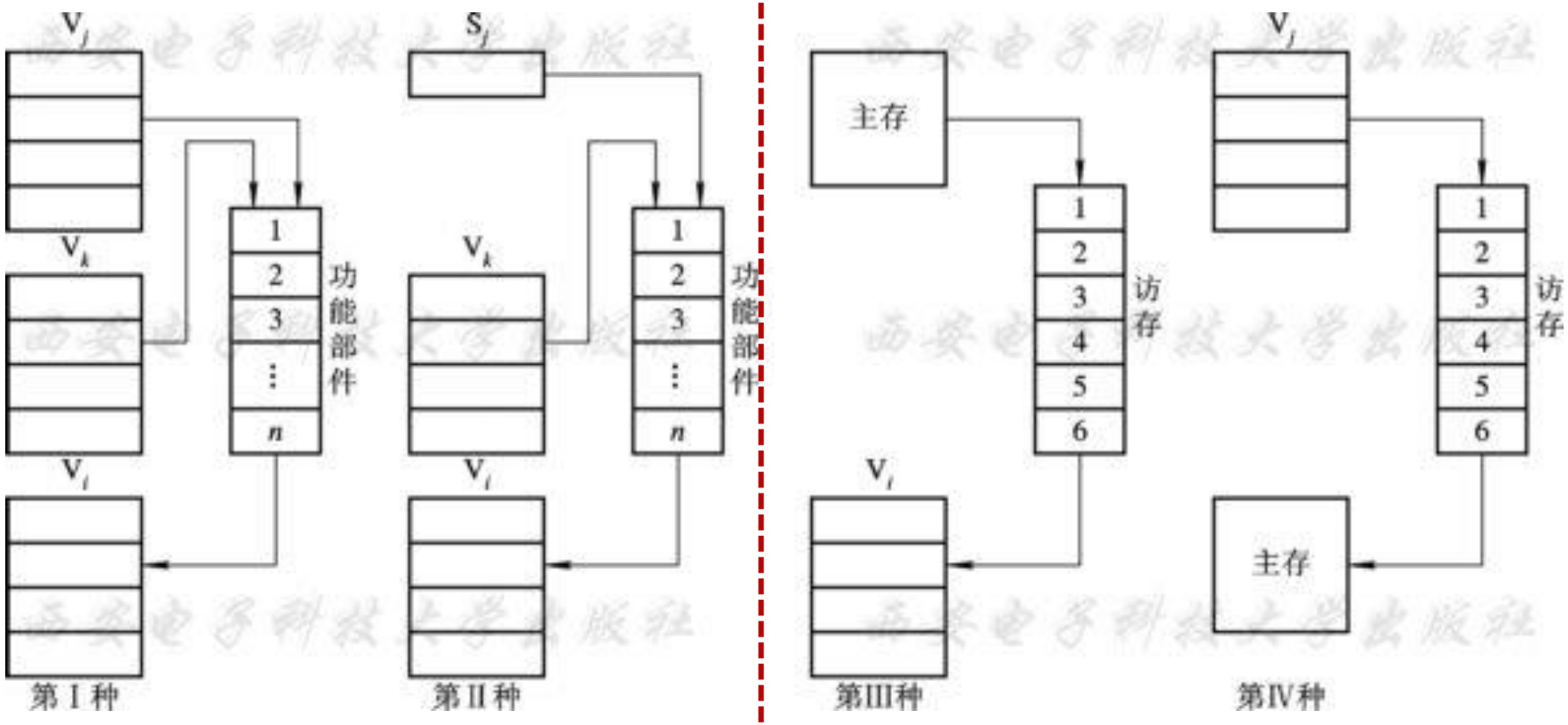
整数运算：加（ 3拍）、移位（4拍）、逻辑运算（2拍）

浮点运算：加（ 6拍）、相乘（7拍）、迭代求倒数（14拍）

512个寄存器分为8个向量寄存器组，每组最多可容纳规模为64的分量

CRAY-1中典型的4种向量指令

访存流水线，其建立需要6拍的时间



CRAY-1共有指令128条

此外，启动访存流水线、把元素送往功能部件、结果存入寄存器组都需要1拍传送延时。

4.1.3. 提高向量流水处理性能的技术

多流水线功能部件并行，流水线链接，加快条件语句和稀疏矩阵处理，加快向量的归约操作等。

□ 流水线链接技术

当两条指令出现“写后读”相关时，若它们不存在功能部件冲突和向量寄存器(源或目的)冲突，就有可能把它们所用的功能部件头尾相接，形成一个链接流水线，进行流水处理。

链接特性实质上是把流水线“设置专用部件”的思想引入到向量执行过程的结果。

向量运算中的相关和冲突

V0 ← V1 + V2

V3 ← V4 × V5

(a)不相关的指令

V0 ← V1 + V2

V3 ← V0 × V4

(b)写后读数据相关

V0 ← V1 + V2

V3 ← V4 + V5

(c)功能部件冲突

一个功能部件被要求并行工作的多条指令使用

V0 ← V1 + V2

V3 ← V1 × V4

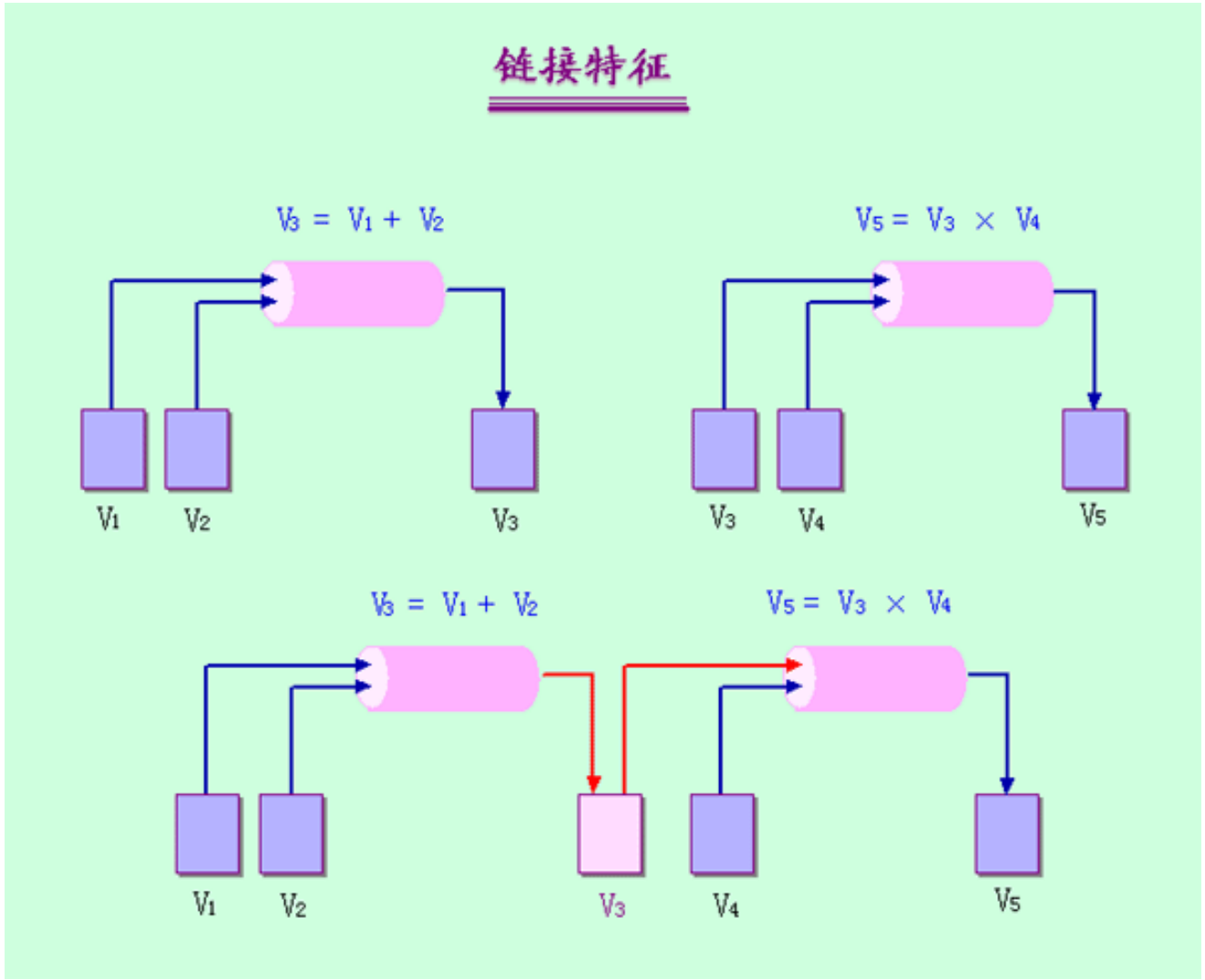
(d)源向量寄存器冲突

两条指令的源向量都来自同一向量寄存器组Vi

链接技术(chaining)

条件：无功能部件冲突，无源向量寄存器冲突

当前一条指令的结果寄存器可以作为后继指令的操作数寄存器时，多条有数据相关的向量指令并行执行，这种技术称为流水线的链接技术。



向量链接的一些主要要求(除了要保证无向量寄存器使用冲突和无向量功能部件使用冲突的条件之外)

◆ 在进行链接的时候，只有在前一条向量指令的第一个结果元素送入结果向量寄存器的那一个时钟周期才可以进行链接，若错过该时就不能进行链接。

◆ 只有当前一条向量指令全部执行完毕，释放相应的向量寄存器资源后才能执行后面的向量指令。

当一条向量指令的两个源操作数分别是两条先行向量指令的结果寄存器时，要求先行的两条向量指令产生运算结果的时间必须相等即要求有关向量功能部件的延迟时间相等。

◆ 只有所有可以链接执行的向量指令的向量长度相等时，它们之间才能链接执行，否则它们之间也不能链接执行。

思考：

CRAY - 1 向量处理机中，启动存储器、流水部件及打入寄存器各需要1拍。假设，现有若干向量，长度均为N，试着分析下列指令串最短的执行时间

（1）

V1 <-- 存储器　（从存储器中取数：6拍）

V2 <-- V0+ V1　（向量加：6拍）

分析：两条指令不能并行　，采用链接技术提升效率

启动存储器->存储器中取数->写寄存器V1->打入加流水->向量加->写寄存器V2

（2）

V1 <-- 存储器　（从存储器中取数：6拍）

V3 <-- V0+ V2　（向量加：6拍）

分析：两条指令可以并行

4.2 阵列处理机的原理

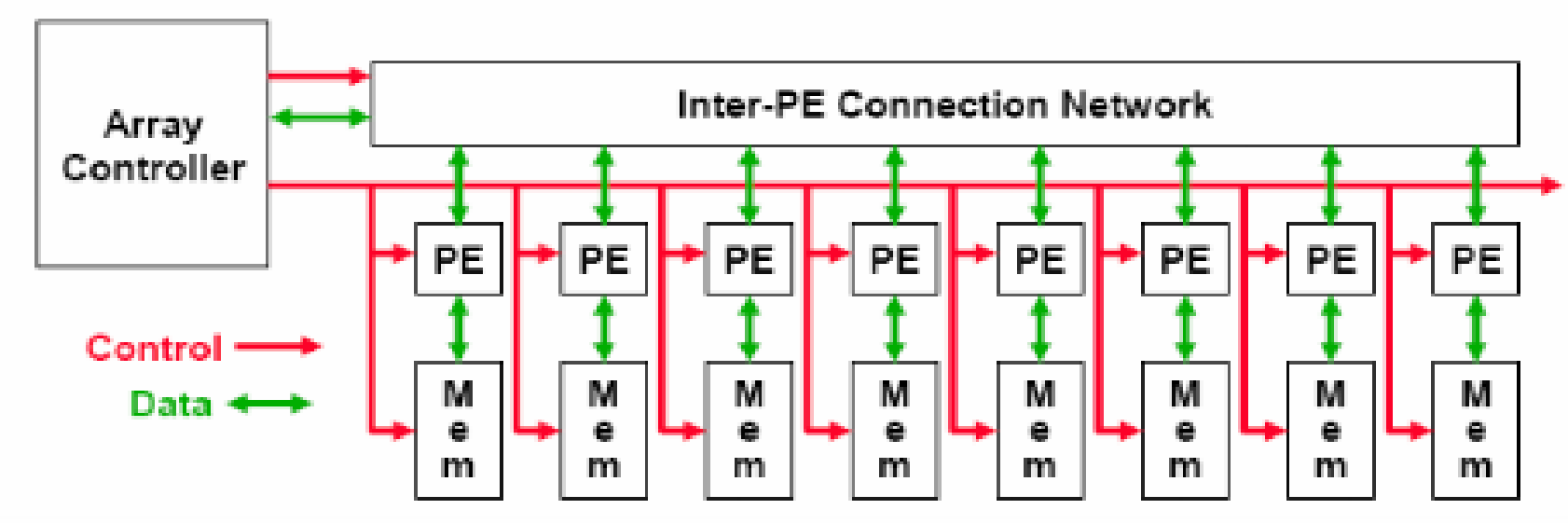
主要内容：

- 阵列处理机的基本结构和特点
- 阵列处理机的特点
- SIMD计算机的互连网络
- 共享贮存构形阵列处理机中并行存储器的无冲突访问

什么是阵列处理机？

阵列处理机是将大量重复设置的处理单元（PE），按一定的方式互连成阵列，在单一控制部件（CU）的控制下，对各自所分配的不同的数据并行执行相同操作

指令级并行的SIMD计算机



4.2.1 阵列处理机的构型和特点

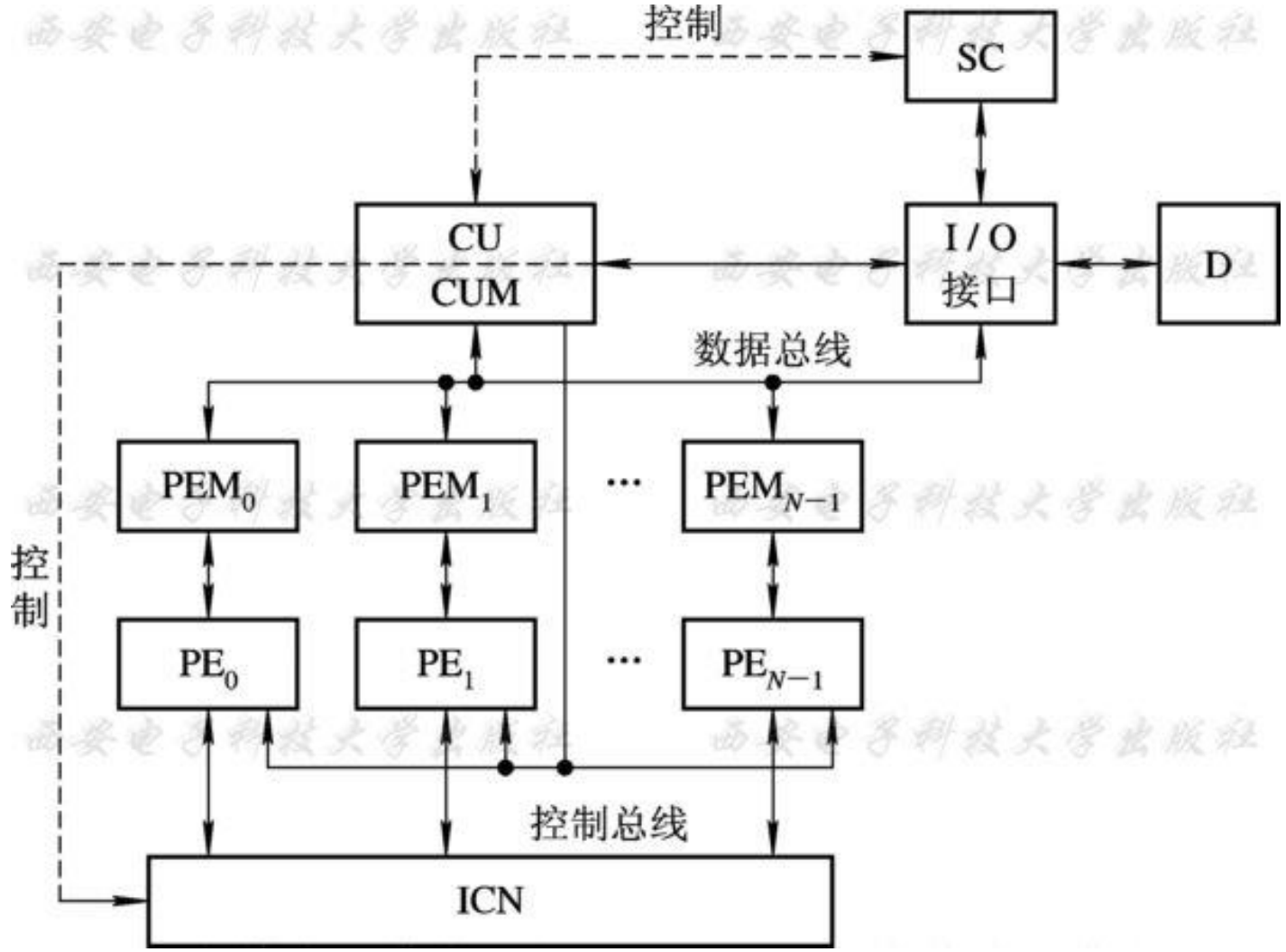
阵列处理机通常由1个控制器(CU)，n个处理器(PE)，m个存储模块(M)及1个互连网络(ICN)组成。

阵列处理机结构的差异主要体现在存储器的组成方式和互连网络的作用不同。

由存储器的组成方式不同互连网络的作用不同进行分类

- 分布存储器的并行处理结构
- 共享存储器的并行处理结构

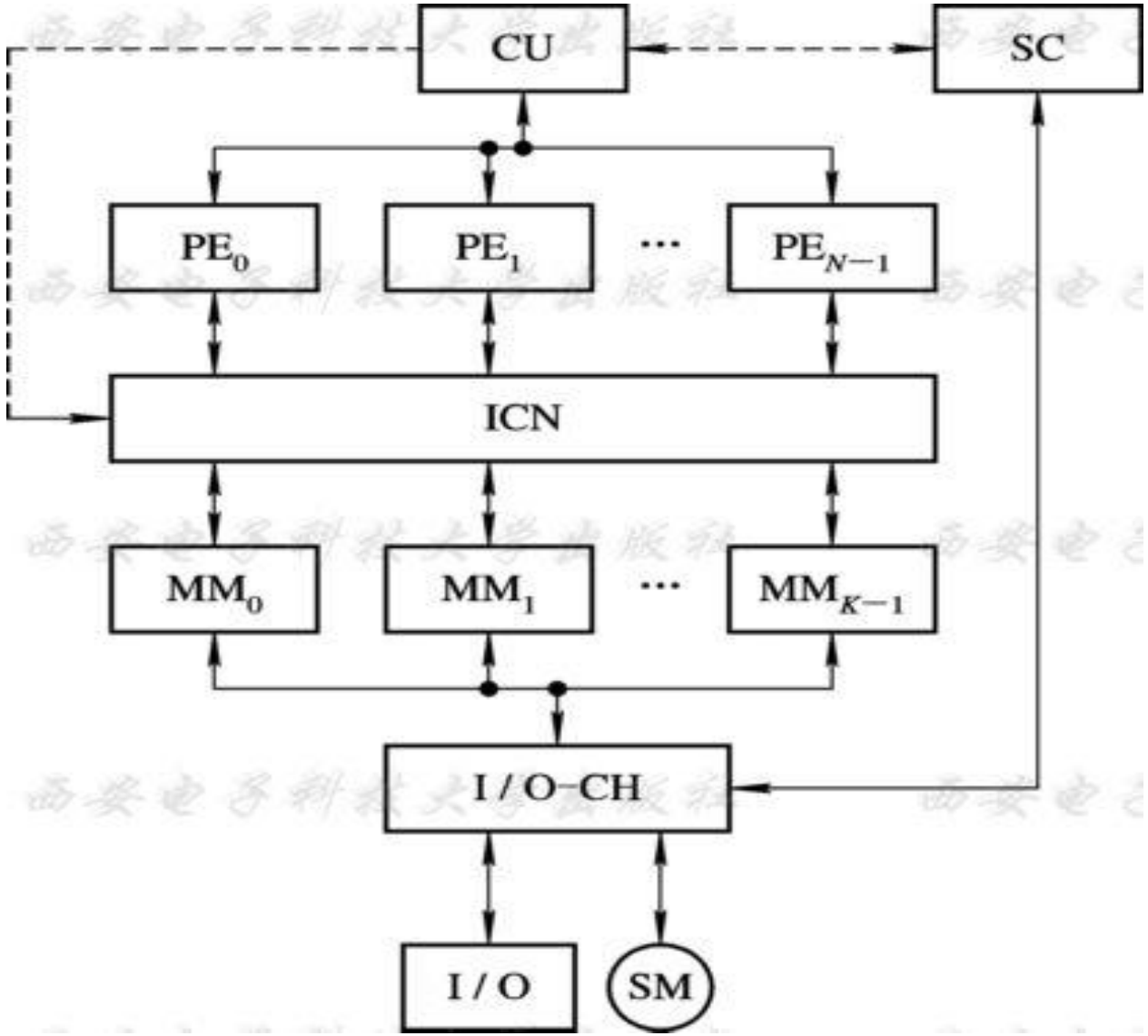
1. 具有分布存储器的阵列机结构



特点：

- 存储模块由每个PE自带。
- 管理处理机SC负责系统资源的管理功能。
- 数据应预先分配到各个处理单元的局部存储器PEM中
- ICN实现PE到PE的通信，从而实现数据的传送、交换。

2. 具有集中式共享存储器的阵列机结构



特点：

- k个存储分体集中组成系统存储器
- 数据应合理分配到各个存储分体中
- n个PE共享k个存储分体。
- ICN实现了PE到PEM的通信，使尽可能多的PE能无冲突地访问共享的主存模块。又称对准网络。

k 总是大于或等于 n

比较：

分布式每个PE有局部存储器，集中式共享存储器。

ICN的作用不同：分布式PE ←→ PE，集中式PE←→M。

4.2.2 阵列处理机的特点

1. 利用资源重复方法，开发并行性中的同时性

所有PE操作相同，数据不同；

与流水线的方法不同点；（时间重叠）

侧重向量处理方面；

2. 连接模式

ICN的不同连接模式确定阵列处理机的不同结构，影响并行算法的实现方法；

ICN的设计成为并行处理的重点问题之一。

3、专用性

根据一定的互连网络连接模式与一定的算法相联系 ，用于完成某一专用的功能。这样的并行处理机属于专用计算机。

4、复合性

一个完整的系统是由三个部分复合而构成的。

① 多个处理单元组成阵列，并行地处理向量

② 功能极强的控制部件实际是一台标量处理机

③ 系统的管理功能由高性能单处理机担负

4.3 SIMD计算机的互连网络

4.3.1 互连网络与互连函数

1. 基本功能

互连网络主要完成结点与结点间的连接，连接和控制方式不同，连接效果不同。

2. 互连网络的设计思路

根据应用需要（互连网络属性），选择合理的特征方式，考虑互连网络的性能因素，综合加以合理组合。

目标：低成本、高灵活性、高连接度、低延时、适合VLSI。

3. 互连网络的表示

四要素：
定时协议、控制策略、开关方法、拓扑结构

（是互连网络 的三个主要操作特征）

- 静态拓扑结构
- 动态拓扑结构

网络入、出端的连接模式 —求出→ 相邻节点（处理单元）间的通路

可用一组互连函数定义

描述入、出节点地址的一一对应关系

由二进制编码表示，即N个节点的地址为$n=\log_2 N$位

互连网络的连接特征一般用连接函数表示。连接函数可以用节点间的连线图表示，也可以用简单的函数式表示。

入端的编码：$x=(b_{n-1}\ldots b_0)$　　$n=\log_2 N$

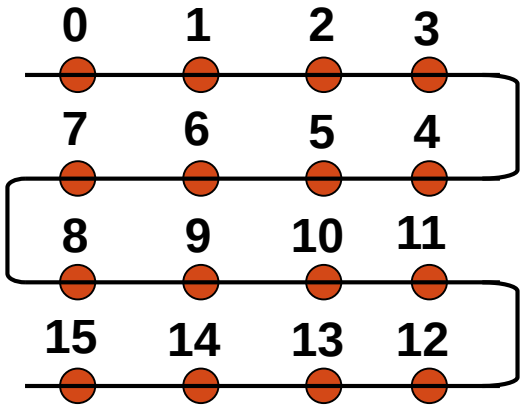
出端的编码：$f(x)=(b_{n-1}\ldots \overline{b_0})$或其他形式。

互连函数为基于$b_{n-1}\ldots b_0$的排列、组合、移位、取反等操作的结果。

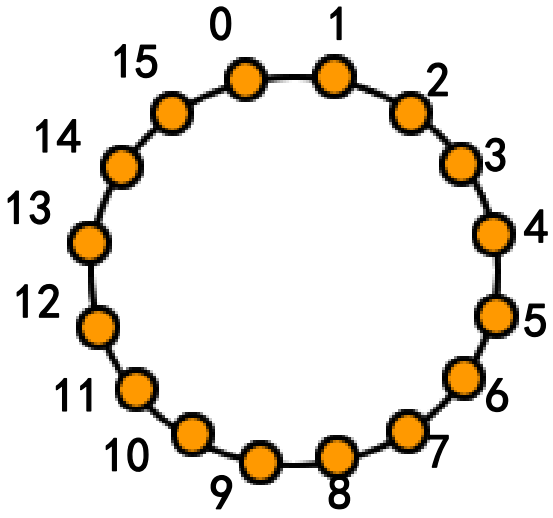
互连网络的拓扑结构根据出、入端可以连接的模式，分为静态和动态两种。

- 静态拓扑结构：两个节点间的链路是固定的
- 动态拓扑结构：两个节点间的链路通过置定网络的开关单元状态可以重新配置

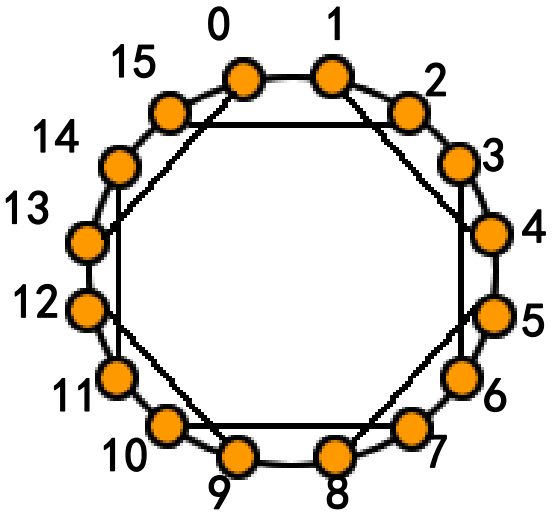
静态拓扑结构



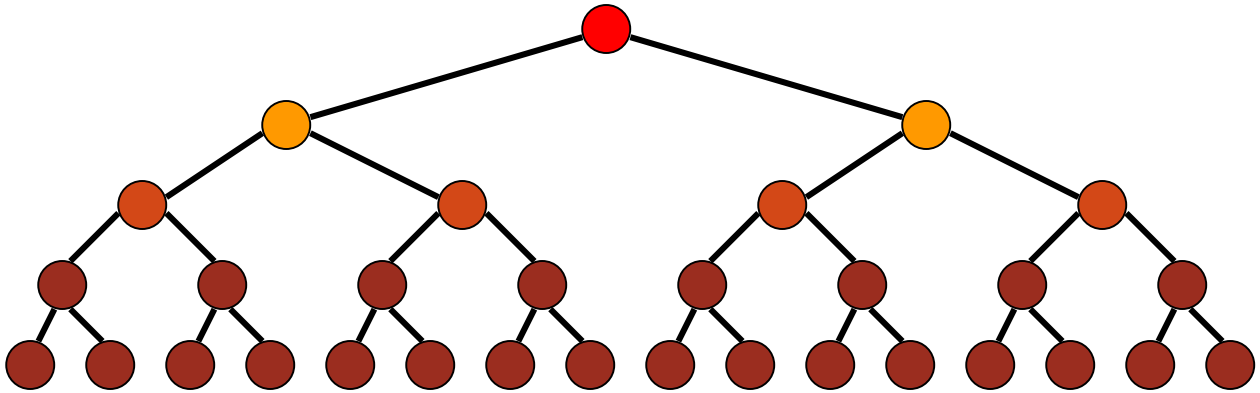
一维线性阵列



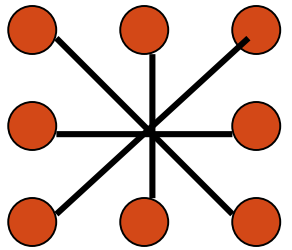
二维环



度为3的带弦环



二叉树



星型连接

动态拓扑结构

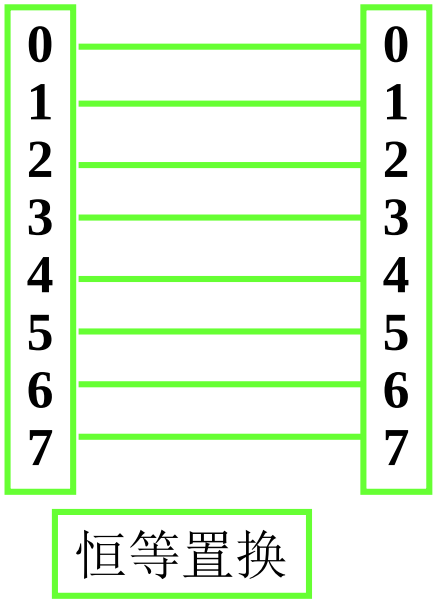
动态单级网络只有有限的几种连接，必须经循环多次通过，才能实现任意两个处理单元之间的信息传送，故称此动态单级网络为循环网络。

动态多级网络是由多个单级网络串联组成的，以实现任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的互连，称循环多级网络或多级循环网络。

4.3.2 基本互连函数举例

1.恒等置换：相同编号的输入端与输出端一一对应的互连

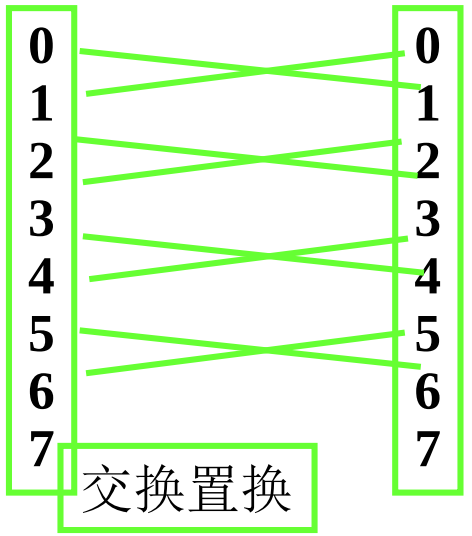
函数式：$I(p_{n-1}p_{n-2}\cdots p_1p_0)=p_{n-1}p_{n-2}\cdots p_1p_0$



2.交换置换：实现二进制地址编号中第0位位置不同的输入端和输出端之间的连接。

函数式：

$EX(p_{n-1}p_{n-2}\cdots p_1p_0)=p_{n-1}p_{n-2}\cdots p_1\overline{p_0}$



3.立方体置换：实现二进制地址编号中的第i位位置不同的输入端和输出端之间的连接

函数式: $Cube_i(p_{n-1}p_{n-2}\cdots p_i\cdots p_1p_0)=\overline{p_{n-1}}p_{n-2}\cdots p_i\cdots p_1p_0$

$(0\le i<n)$

4.全混洗置换：将输入端的二进制地址循环左移一位

函数式：$Sh\ (p_{n-1}p_{n-2}\cdots p_i\cdots p_1p_0)=p_{n-2}\cdots p_i\cdots p_1p_0p_{n-1}$

5.加减2^i置换：将输入端数组循环移动2^i的位置向输出端传送

函数式：$PM2_{+i}(x)=(x+2^i) \bmod N$

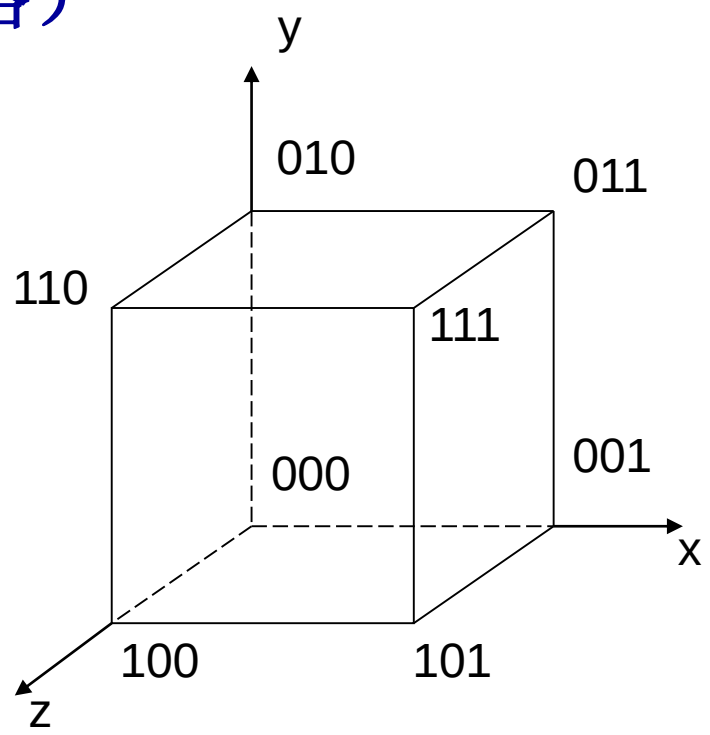
$PM2_{-i}(x)=(x-2^i) \bmod N \quad (0\le x\le N-1,$

$0\le i\le n-1, n=\log 2N)$

4.3.3基本的单级互连网络

1. 立方体单级网络（交换互连网络）

立方体的每一个顶点，代表一个处理单元，共8个处理单元，用 Z Y X 三位二进制代码来表示。立方体单级网络有 3 种互联函数



互连函数：

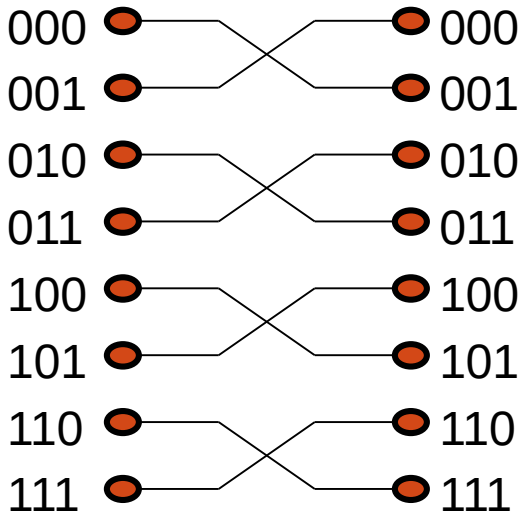
X： $Cube_0=(b_2b_1\overline{b_0})$；

Y： $Cube_1=(b_2\overline{b_1}b_0)$；

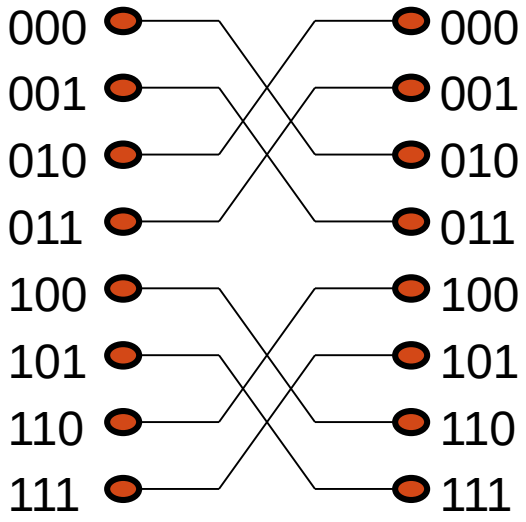
Z： $Cube_2=(\overline{b_2}b_1b_0)$。

连接线的两端上的编码，对应于相应方向的位置（1位）取反，其余不变。

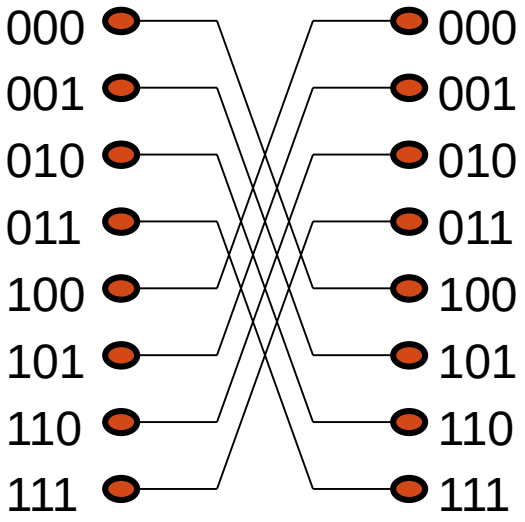
连接图：



Cube0

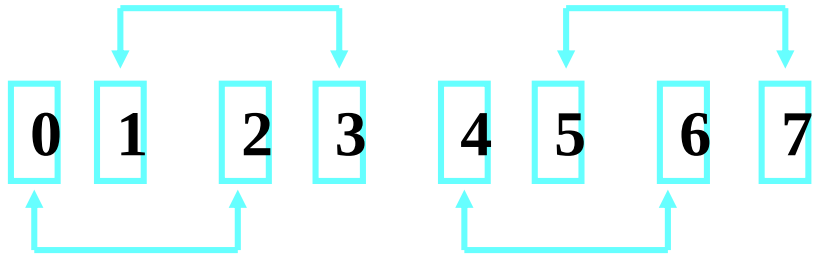


Cube1

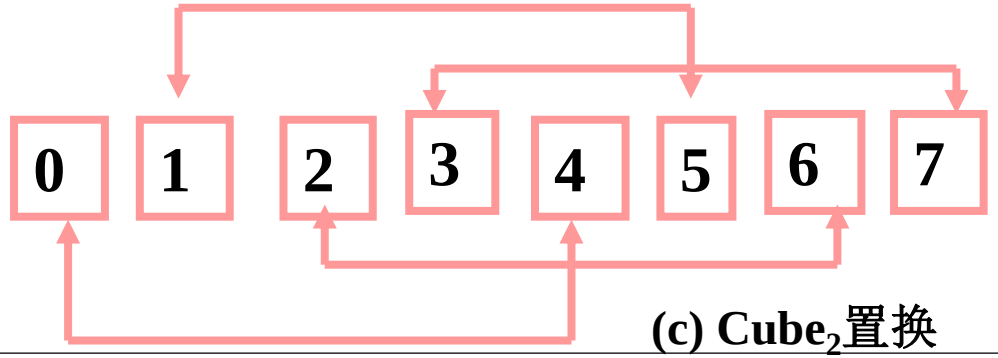


Cube2

(a)Cube$_0$置换



(b) Cube$_1$置换



(c) Cube$_2$置换

多组二元交换

互连特性：

交换功能--互连函数可逆；

互连函数个数=$\log_2 8=3$；最大连接度=$\log_2 8=3$；

结点最大间距=$\log_2 8=3$。

扩展成有N个节点的超立方体：

有$n=\log_2 N$个互连函数；

$Cube_i=(b_{n-1}\ldots\overline{b}_i\ldots b_0)$；

最大连接度=$\log_2 N$； 结点最大间距=$\log_2 N$。

应用：几种互连函数反复调用，任意结点间可连接。

2. PM2I单级网络（循环移数网络）

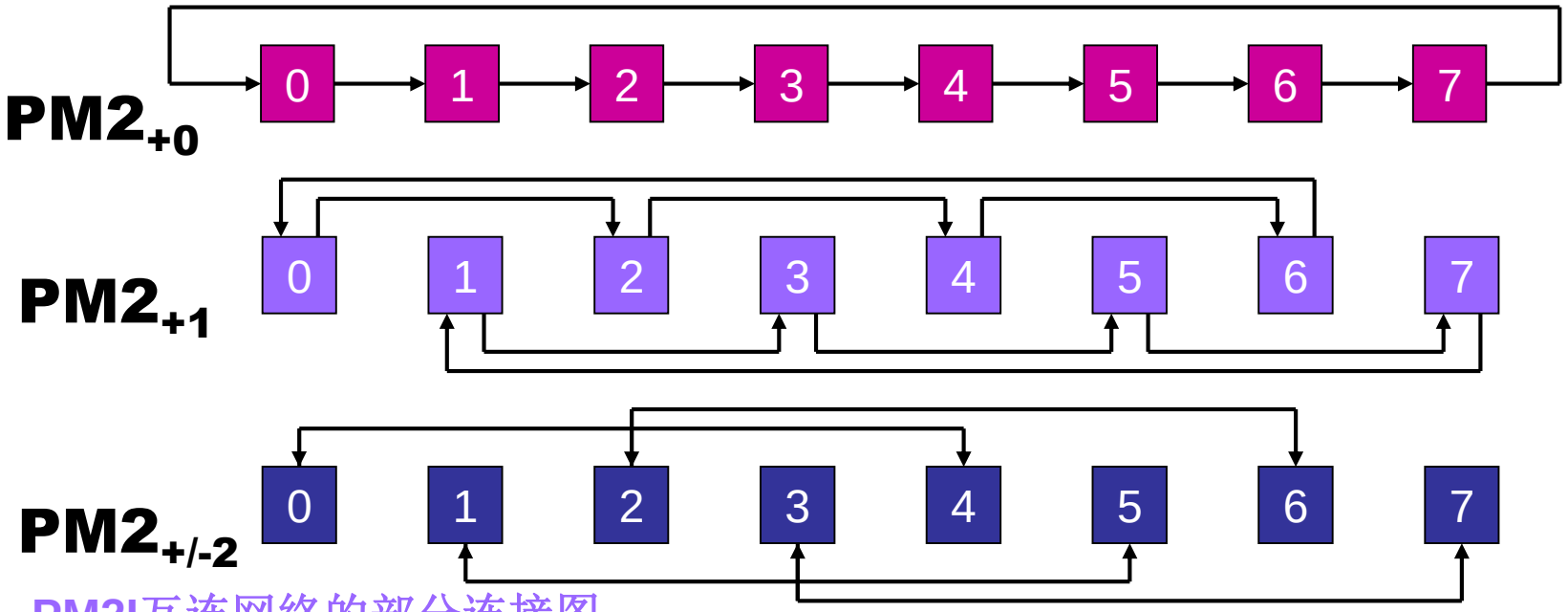
出端编码与连接的入端结点编码相差2^i。

互连函数：$PM2_{+i}(j)=(j+2^i) \bmod N$；

$n=\log_2 N, 0 \leqslant i \leqslant n-1,$

$PM2_{-i}(j)=(j-2^i) \bmod N$；　$0 \leqslant j \leqslant N-1$

共有2n个互连函数，其中2n-1种不同。



PM2I互连网络的部分连接图

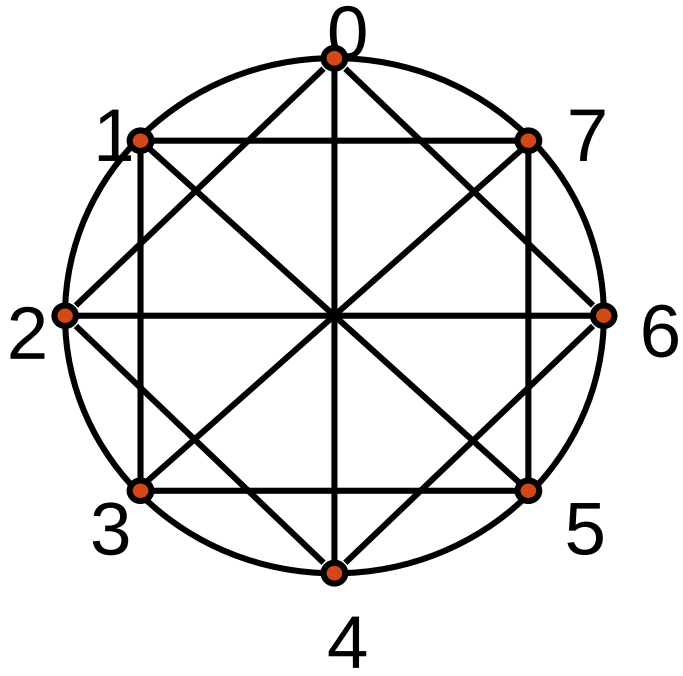
连接图：

±0：顺环圆周连接；

±1：顺环内接N/2边形连接；

±2：顺环内接N/4边形连接；

±(n-1)：顺环内直径连接。



互连特性：

2n个互连函数只有一种函数可逆，其余均不可逆；

最大连接度2n-1；

互连函数个数2n。

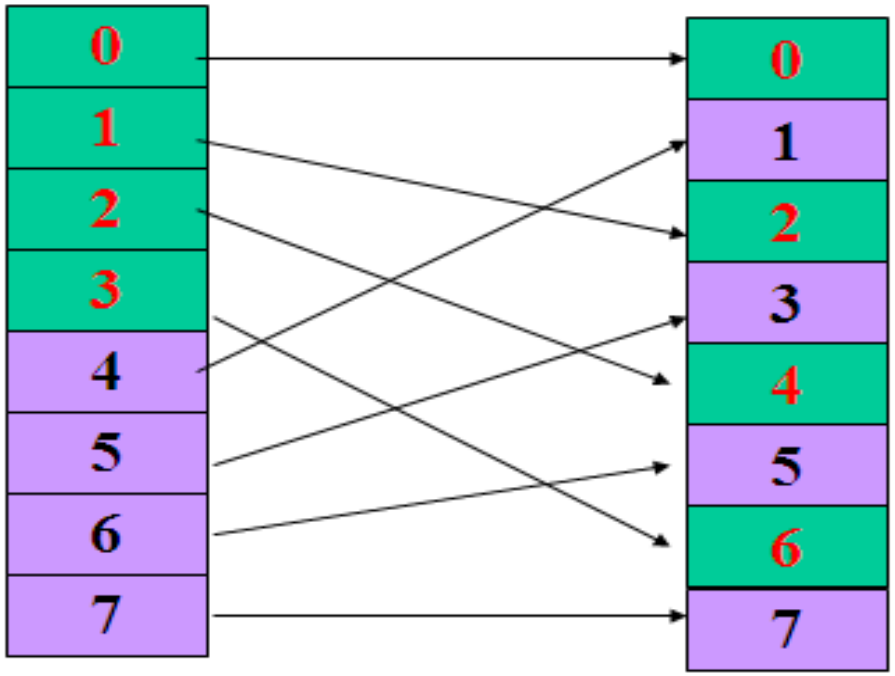
结点最大间距 $\lceil n/2 \rceil = \lceil \log_2 N/2 \rceil \leqslant \log_2 N/2$；

3. 混洗交换单级网络

全混洗互连函数（shuffle）：

$$Shuffle(b_{n-1}b_{n-2}\ldots b_1b_0)=(b_{n-2}\ldots b_1b_0b_{n-1})$$

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



全“0”或全“1”结点无法与其他结点连接，必须辅以交换互连函数，方可实现任意结点间连接。

最简单的交换互连函数为$Cube_0$，因此混洗交换网络由全混洗和交换网络组合而成。

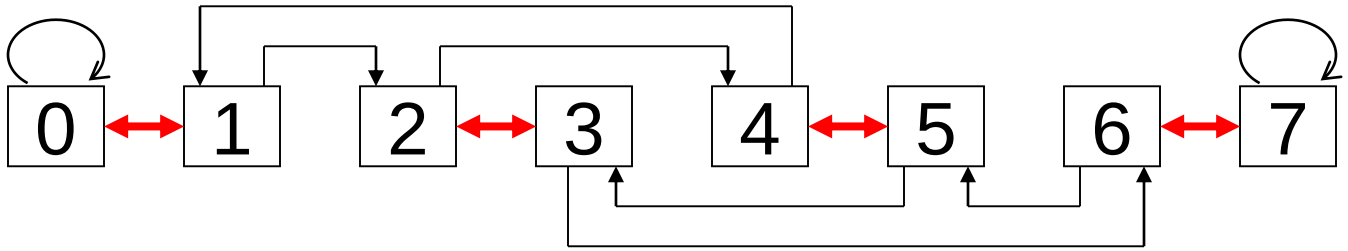
交换互连函数：

$$Exchange(b_{n-1}b_{n-2}\ldots b_1b_0)=(b_{n-1}b_{n-2}\ldots b_1\overline{b_0})$$

混洗交换互连函数：

$$Exchange[Shuffle(b_{n-1}b_{n-2}\ldots b_1b_0)]$$
$$=(b_{n-2}\ldots b_1b_0\overline{b_{n-1}})$$

连接图：



互连特性：

互连函数不可逆；

对于shuffle函数，n次全混洗之后，还原；

最大间距：n次交换，n-1次混洗，共2n-1次；

全混洗最先改变最高位(左移)，交换取反最低位。

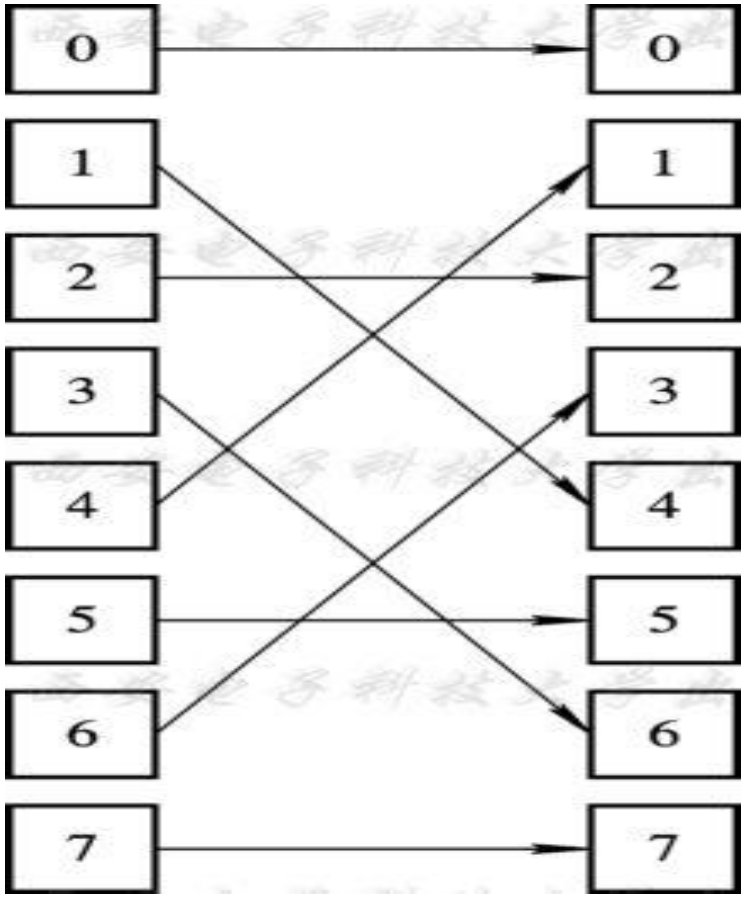
应用：

多次调用混洗交换互连函数，可实现任意结点间的连接。

4. 蝶形单级网络

蝶形互连函数（Butterfly）：

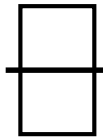
$$\text{Butterfly}\ (P_{n-1}P_{n-2}\cdots P_1P_0)=P_0P_{n-2}\cdots P_1P_{n-1}$$



即将二进制地址的最高位和最低位相互交换位置。

单级互连网络总结

(1)单级互连网络特性

任一单级互连网络均可表示成$N_{入}$  $N_{出}$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。

(2)单级互连网络应用

利用单级互连网络的特性作为实际ICN的拓扑结构；

通过交换开关作为ICN的可变因素；

通过交换开关多次控制实现ICN的结点间任意互连。

4.3.4基本的多级互连网络

目标：完成某结点与其它任一结点的连接；
同时完成多对结点的连接。

方法：从时间性和空间性方面开发。

□ 循环互连网络（时间性）

组成：DTR_{in}、DTR_{out}、MUX、IN。

结构：

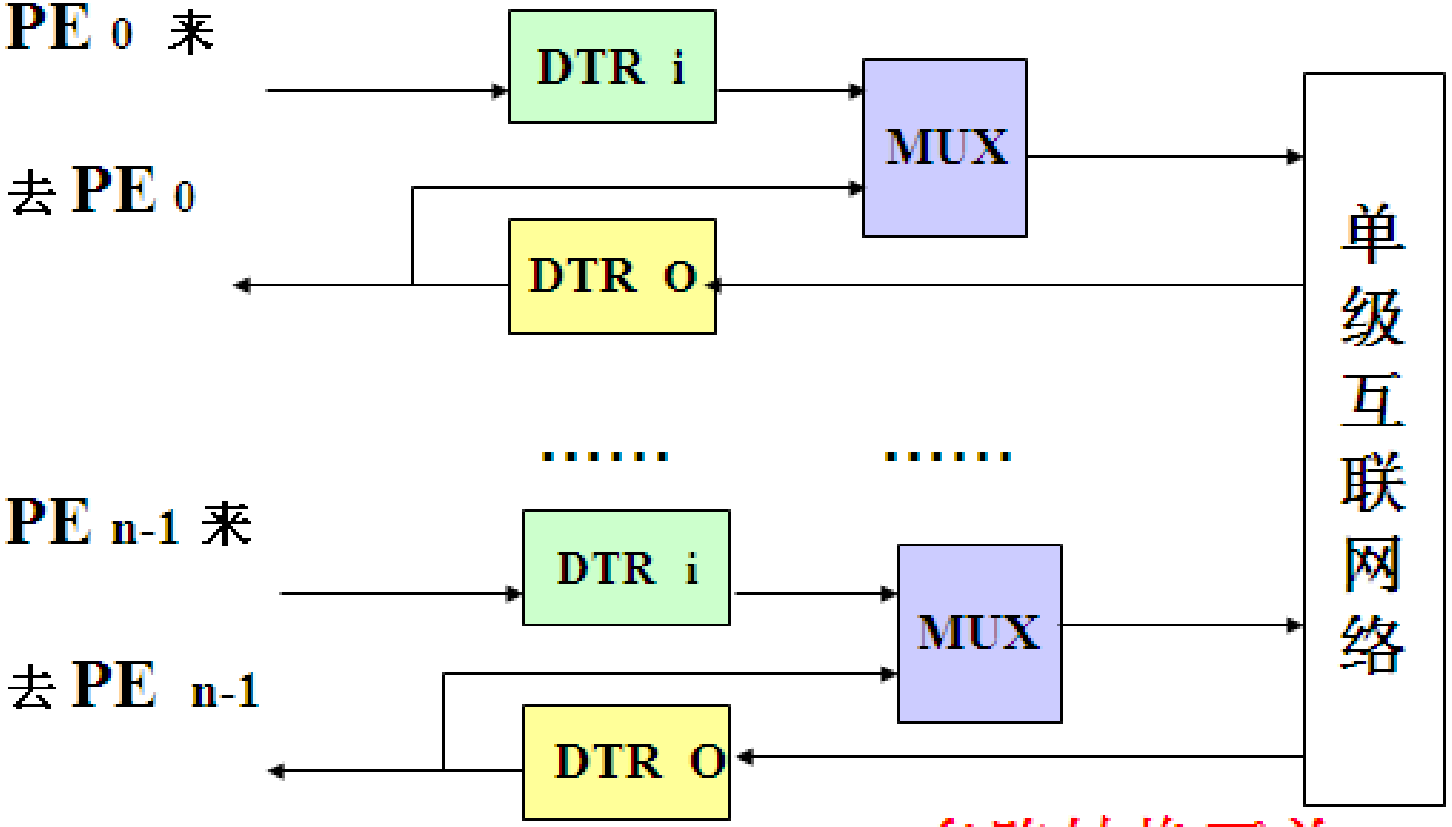
一个单级ICN+MUX。

特点：

节省了设备，增加了时间，每个MUX可单独控制。

DTR i 输入端传送寄存器

DTR o 输出端传送寄存器

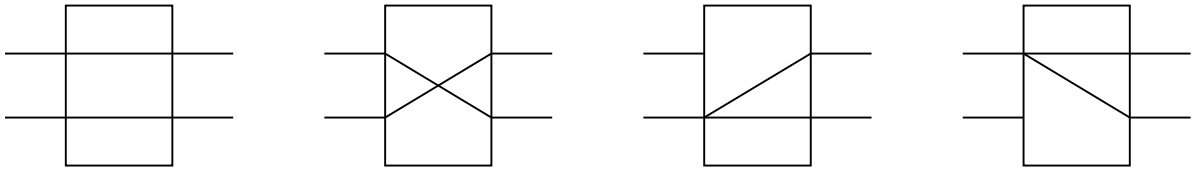


MUX 多路转换开关

多级互连网络（空间性）

不同的多级互连网络反映在所采用的交换开关，拓扑结构和控制方式上有所不同。

交换开关：是具有两个入端和两个出端的交换单元。用作各种多级互连网络的基本构件。



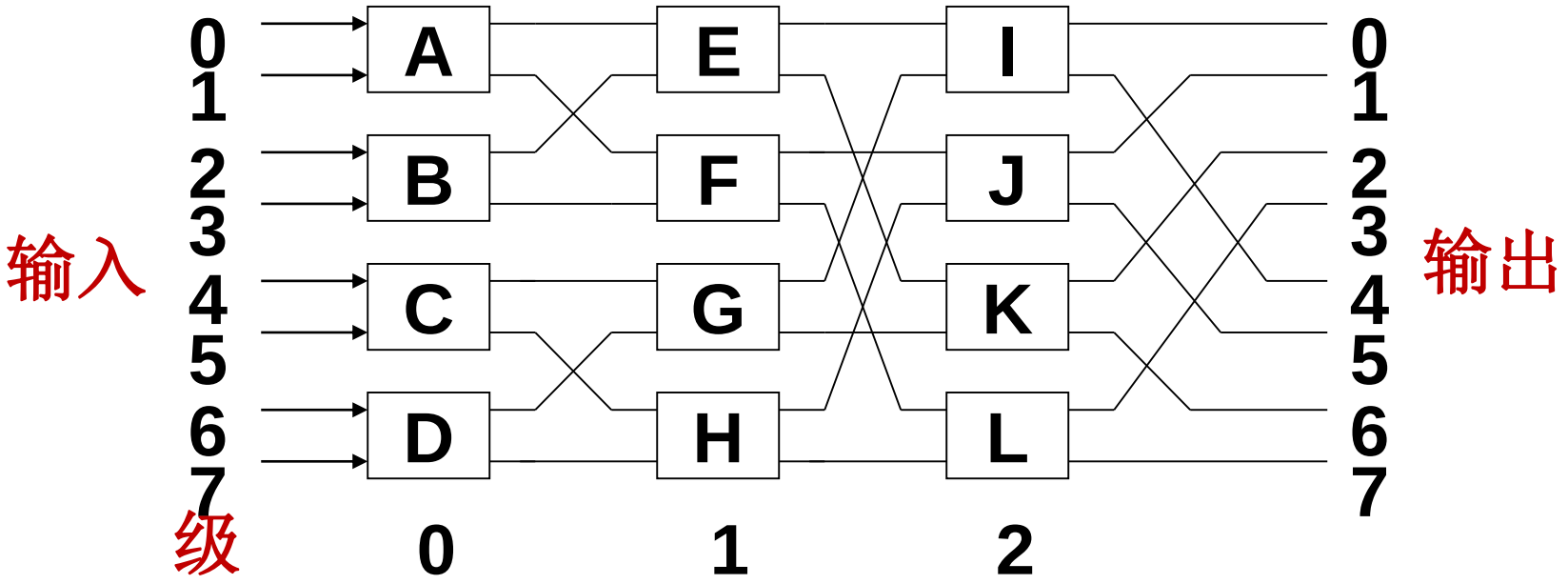
拓扑结构：不同级开关间连接方式(单级CIN的连接功能)，即各级之间出端和入端相互连接的模式。

控制方式：级控制、部分级控制、单元控制。

分类：根据拓扑结构进行分类

多级立方体网络
多级混洗交换网络
多级PM2I网络

1、多级立方体网络



交换开关：二功能（直通和交换）

拓扑结构：当第i级交换开关处于交换功能时，该多级网络实现的是$Cube_i$互连函数；

◆ 对于第i级和第i+1级之间，把2^{i+1}个开关分为一组，组内采用蝶式变换。

8个处理单元的网络为什么只有三级？

任意两节点最大间距为3，3级可实现任意两个节点互连

根据控制方式不同，多级立方体网络有STARAN网络和间接二进制n立方体网络两种。

- STARAN网络：级控制和部分级控制
- 间接二进制n立方体网络：单元控制

STARAN网络根据控制方式不同可以实现不同的功能：交换功能和移数功能。

(1)交换功能

- 控制方式：级控制

  控制信号为1：开关实现交换功能

  为0：开关实现直通功能

- 交换功能：组间次序不变，组内元素镜像。

所谓交换(Flip)函数，是将一组元素首尾对称地进行交换。

4组2元交换	$Cube_0$
2组4元交换	$Cube_1+Cube_0$
1组8元交换	$Cube_2+Cube_1+Cube_0$

- 应用：对集中式处理机同时数据传输作用很大。

STARAN交换网络在不同控制信号下执行的交换函数功能

控制信号	000	001	010	011	100	101	110	111
0	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
1	001 1	000 0	011 3	010 2	101 5	100 4	111 7	110 6
2	010 2	011 3	000 0	001 1	110 6	111 7	100 4	101 5
3	011 3	010 2	001 1	000 0	111 7	110 6	101 5	100 4
4	100 4	101 5	110 6	111 7	000 0	001 1	010 2	011 3
5	101 5	100 4	111 7	110 6	001 1	000 0	011 3	010 2
6	110 6	111 7	100 4	101 5	010 2	011 3	000 0	001 1
7	111 7	110 6	101 5	100 4	011 3	010 2	001 1	000 0
控制函数		Cube 0	Cube1	Cube0+ Cube 1	Cube 2	Cube0+ Cube2	Cube1+ Cube 2	Cube 0 +Cube1 +Cube 2

(2)移位功能

控制：部分级控制（第i级有i+1种控制信号）

功能：控制信号不同，功能不同。

2级	K,L	0	0	1	0	0	0	0
	J	0	1	1	0	0	0	0
	I	1	1	1	0	0	0	0
1级	F,H	0	1	0	0	1	0	0
	E,G	1	1	0	1	1	0	0
0级	A,B,C,D	1	0	0	1	0	1	0
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

应用：

移数功能很适合于累加求和算法实现；

不同的Mod，可用作不同的分组操作。

例1：并行处理机有16个PE，实现相当于4组4元交换，然后2组8元交换，再1组16元交换功能。写出互连函数一般式、各级交换开关状态。

答：因需实现交换功能，故选择STARAN的交换网络(级控制方式）。

4组4元交换　$Cube_0+Cube_1$

2组8元交换　$Cube_0+Cube_1+Cube_2$

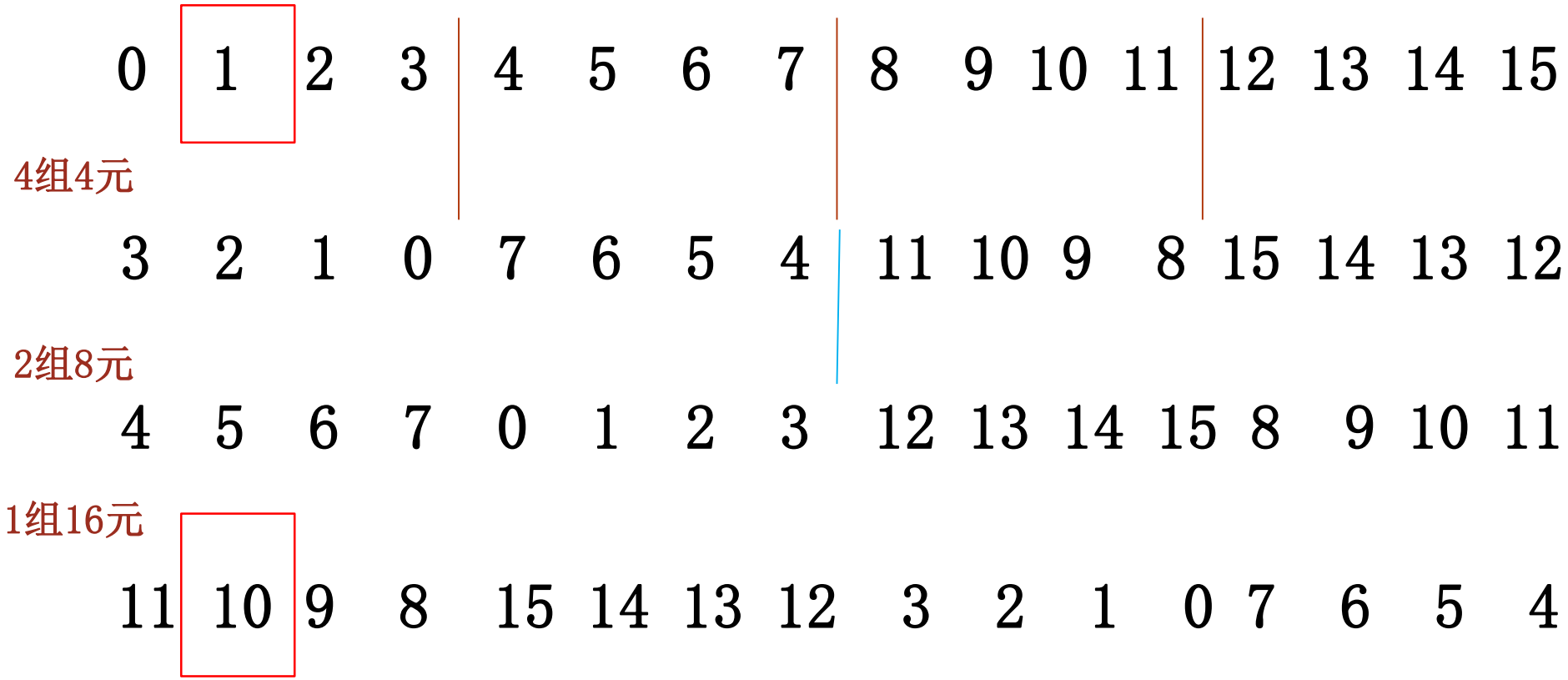
1组16元交换　$Cube_0+Cube_1+Cube_2+Cube_3$

合并　$Cube_0+Cube_1$　$+Cube_3$

互连函数：$f(b_3b_2b_1b_0)=(\overline{b_3}b_2\overline{b_1}\overline{b_0})$

各级开关状态：$k_3k_2k_1k_0=(1011)$

方法二：



$f(b_3b_2b_1b_0) = b_3\#\ b_2\ b_1\#b_0\#$

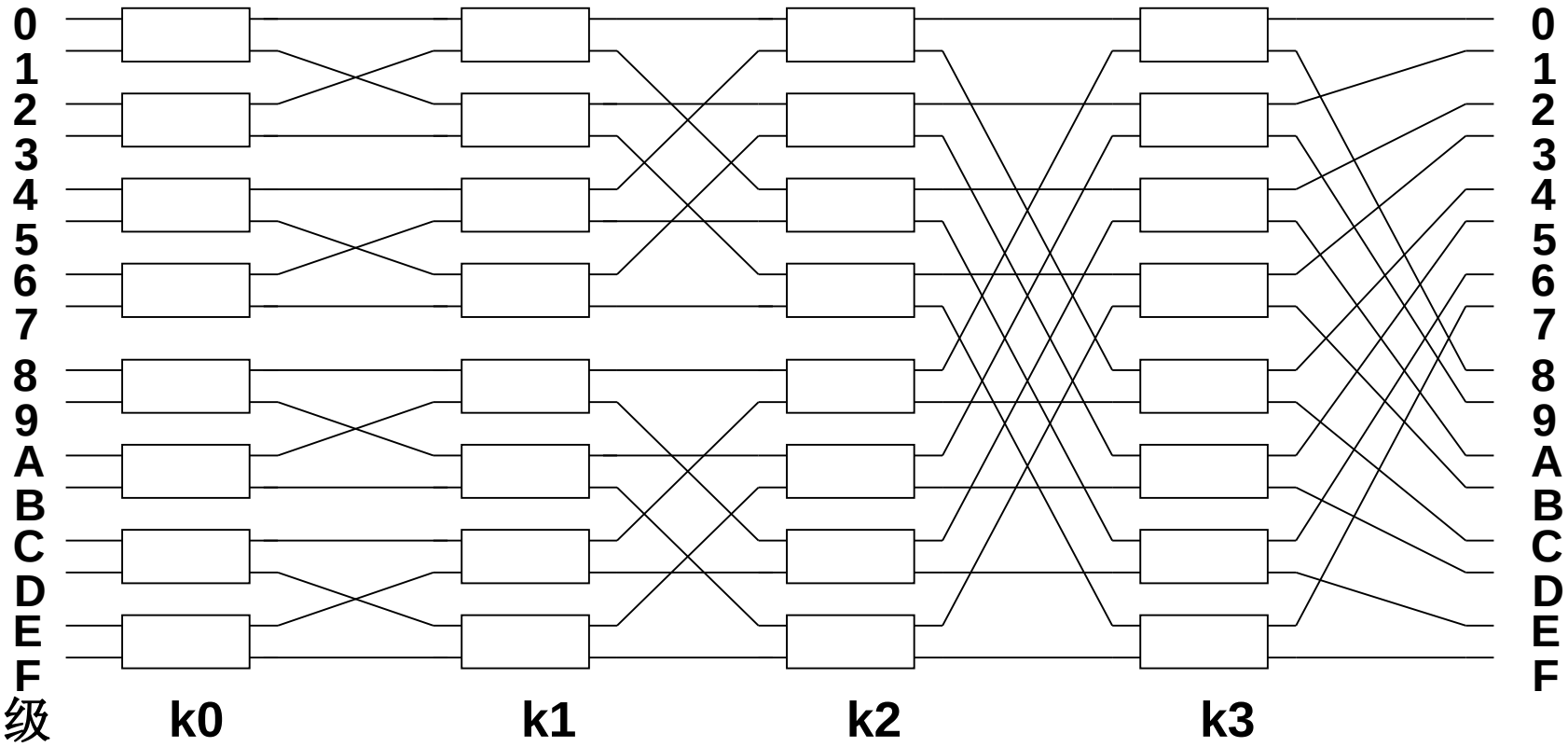
第0/1/3级的开关均为交换状态，第2级开关均为直通

例2：编号0~F的PE间，要实现下列通信配对：

(7,D),(6,C),(5,F),(4,E),(3,9),(2,8),(1,B),(0,A)

画出互连网络结构图，写出控制方式、各开关状态。试写出实现了哪些交换函数功能。

答：因共有16个结点，编码需4位，则开关共4级。



因≤7的结点需与＞7的结点配对，故需1组16元交换；

因0~3的结点与8~B的结点配对，故需2组8元交换；

结果：0~3 ←→8~B，4~7 ←→C~F

因0~1的结点与A~B的结点配对，故需4组4元交换；

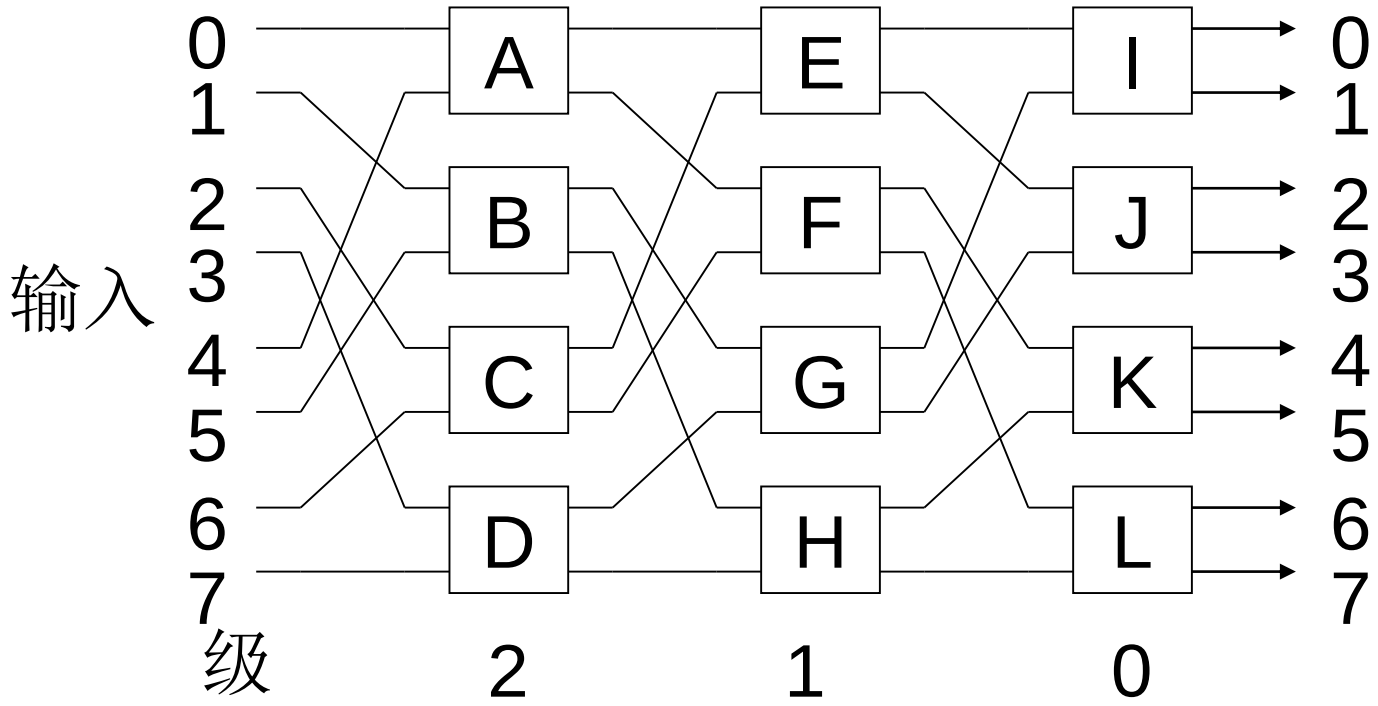
结果：0~1 ←→B~A，2~3 ←→9~8

因0结点与A结点配对，故需8组2元交换。

1组16元交换	$Cube_0+Cube_1+Cube_2+Cube_3$
2组8元交换	$Cube_0+Cube_1+Cube_2$
4组4元交换	$Cube_0+Cube_1$
8组2元交换	$Cube_0$
相加	$Cube_1+$ $Cube_3$

各级开关状态：$k_3k_2k_1k_0=(1010)$

2. 多级混洗交换网络（ω网络）



交换开关：四功能（允许实现一对多的连接）

拓扑结构：不同级相同，均为全混洗结构；

控制方式：级控制、部分级控制、单元控制；

连接图：第n-1级靠近入端；

功能：

若为级控制且开关为二功能：

是STARAN交换网络的逆网络；

（F、G交换位置）

部分级控制且开关为二功能：

是STARAN移数网络的逆网络；

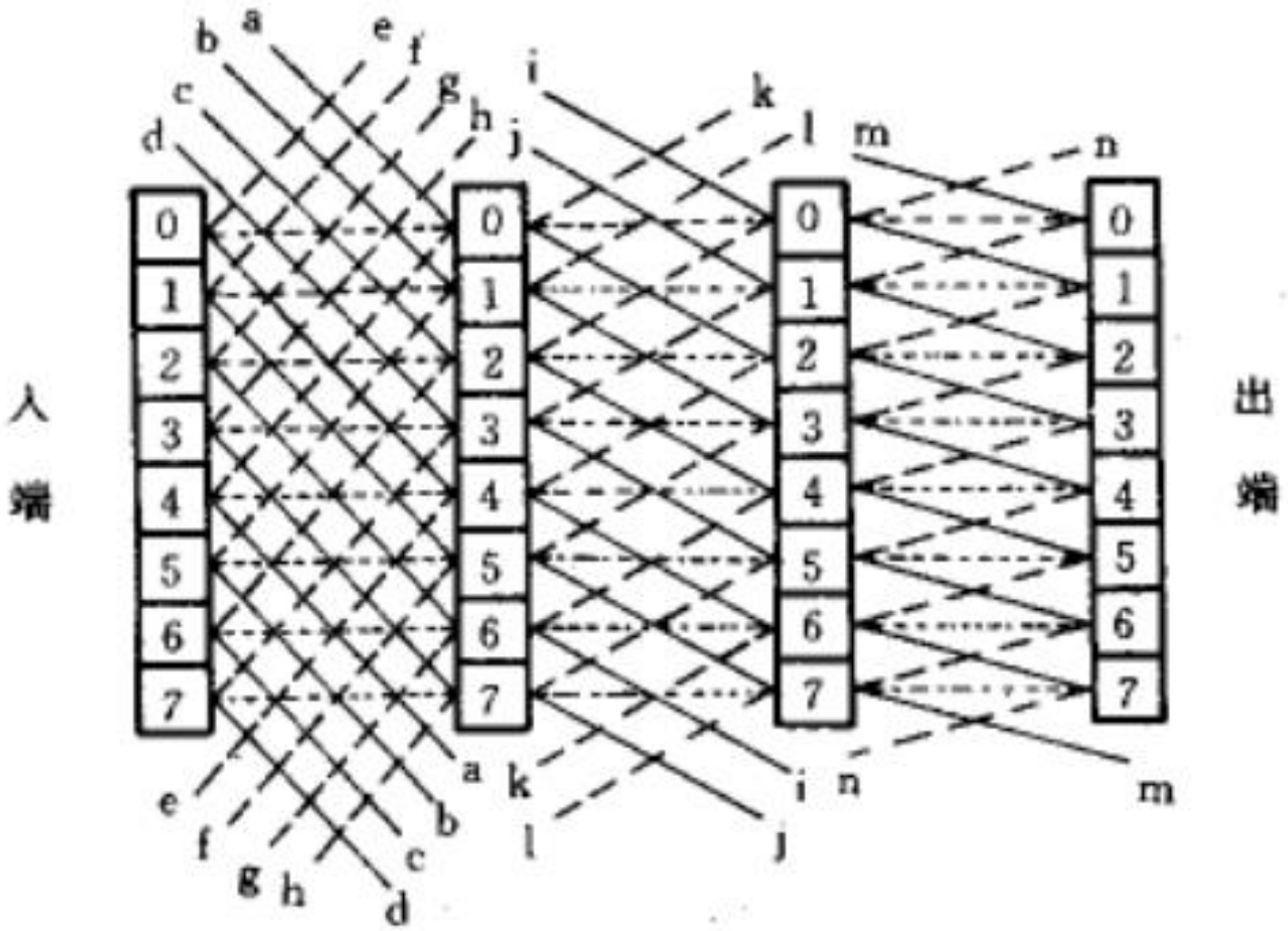
单元控制：可实现更强大的功能。

利用交换开关的播送功能实现一对多的连接。

典型应用：恒等置换、移数置换等各种函数的变形置换；可完成数组按行、列、对角线、子块等无冲突访问

3. 多级PM2I网络

拓扑结构：不同级相同，**均为PM2I**；

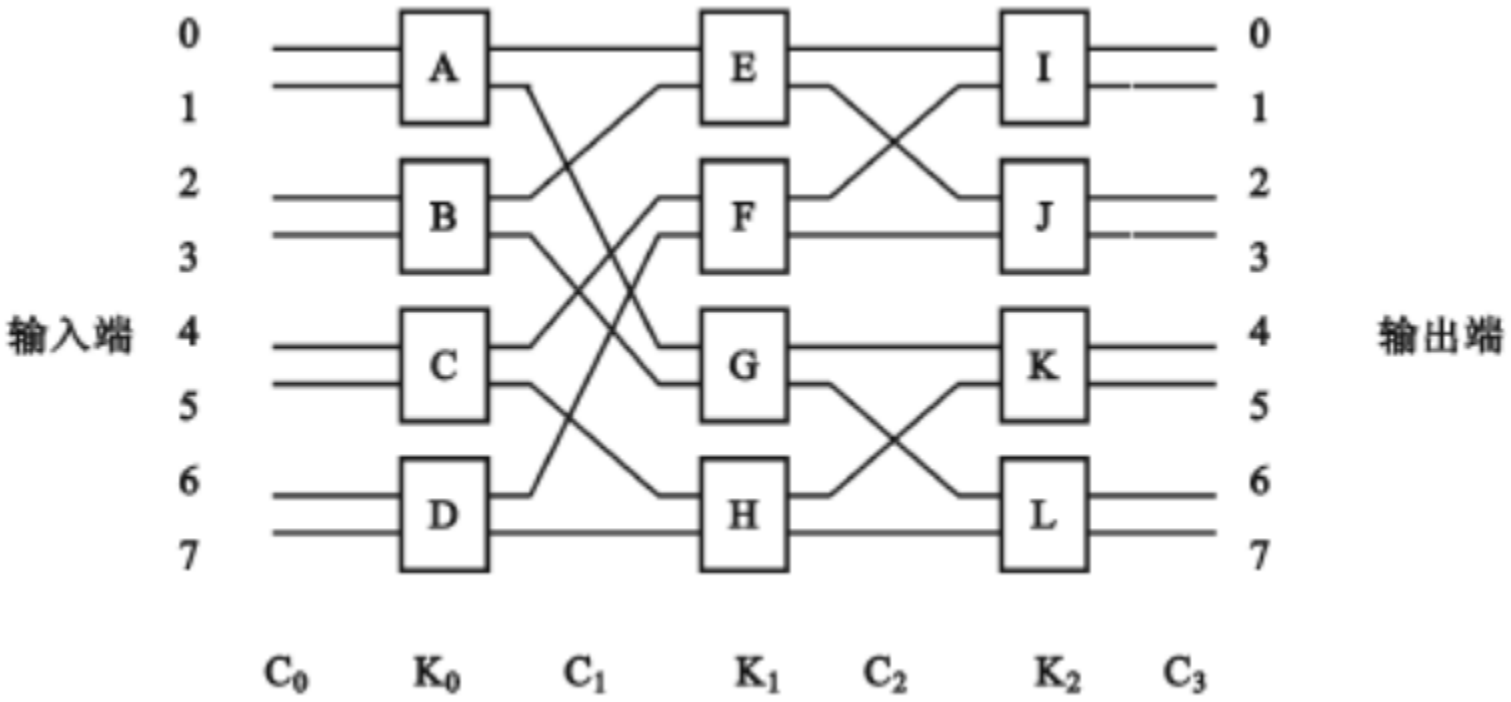


交换开关：四功能

控制方式：部分级控制（DM）、单元控制（ADM）；

连接图：第n-1级靠近入端；

4. 基准网络



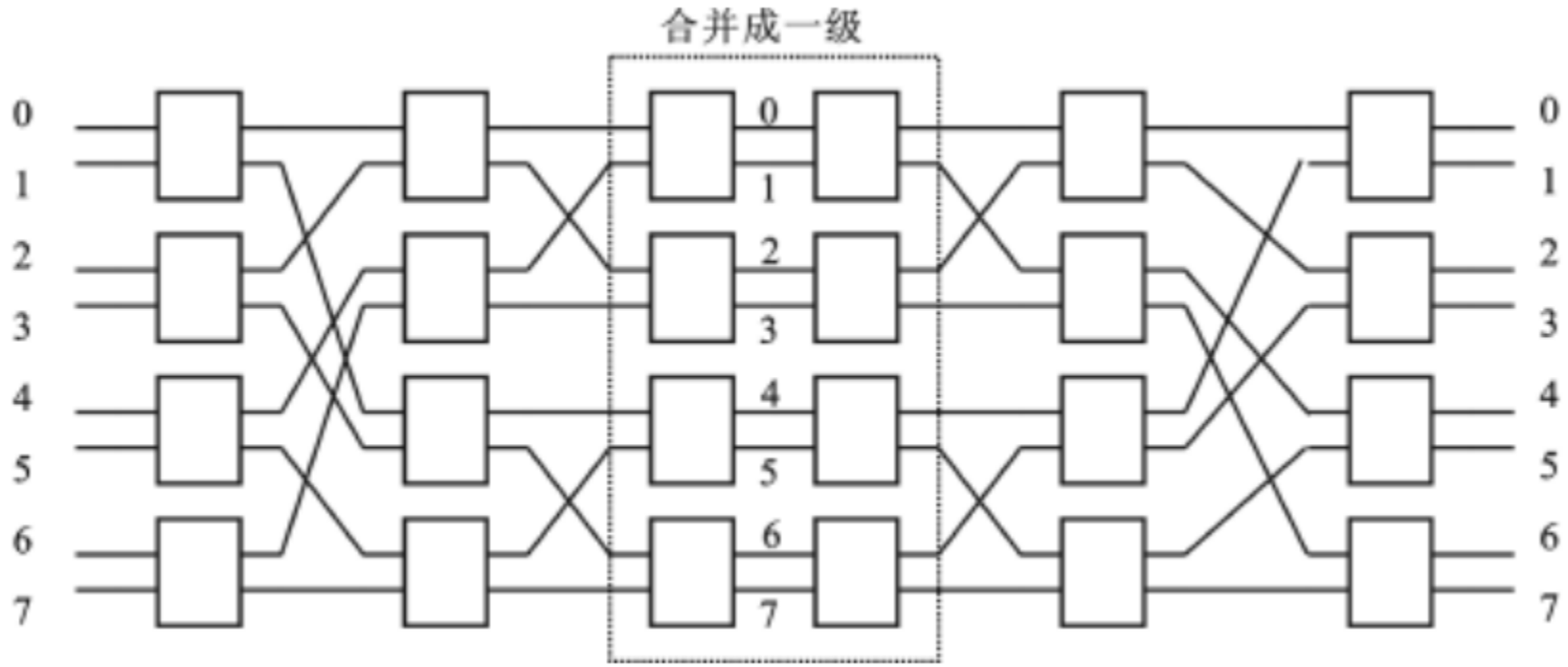
交换开关：二功能

控制方式：单元控制

拓扑结构：C0和Cn是恒等置换，C1是全混洗的逆，后续是子逆混

典型应用：**一次通过基准网络**可以实现**位序颠倒置换**，对实现 FFT 很有利；**二次通过**基准网络可以实现**任意置换**。

5. 全排列网络



灵活性好、连线数多、控制复杂、成本高

各种基本多级网络都能实现任意一个输入端与任意一个输出端间的连接，但要**同时实现两对或多对输入、输出端间的连接**时，都有可能发生争用数据传送路径的**冲突**。我们称有这类性质的互连网络为**阻塞式网络 Blocking Network**，称无这类性质的互连网络为**非阻塞式网络或全排列网络**。

4.4 并行存储器无冲突访问

一、访问需求

并行存取向量中各分量信息；

可按行、列、对角线等方法存取(步长不一致)。

二、存在问题

存储器带宽限制—存储器带宽达不到向量带宽；

访存方式(步长)不同，产生访存冲突。

三、解决方法

1. 采用多体交叉存储器--增加MEM带宽

2. 对向量分组操作--解决MEM带宽小于向量带宽问题

3. 选择适当的存储体数m--达到无冲突访问

一维向量：顺序存放，防止步长与m成比例；

m取质数，且与步长互质。

多维向量：错位存放，满足行、列、对角线等方式；

当m大于每次访问向量元素个数时，

$m=2^{2P}+1$，$\sigma_1=2^P$，同一列不同行错开距离

$\sigma_2=1$，同一行不同列错开距离

对A_{ab}，体号：$j=(a\sigma_1+b\sigma_2+C) \bmod m$

体内序号：$i=a$

当向量元素不固定，或非n×n时，

将多维变换成一维数组S，再对S进行处理。

对S(a)，体号：　　$j=a \bmod m$

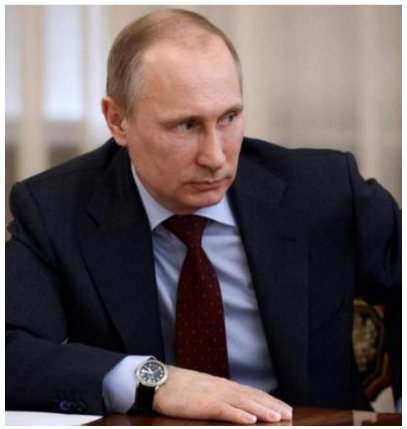
体内序号：$i=\lfloor a/n \rfloor$

通过浪费少量存储带宽和空间来避免冲突。

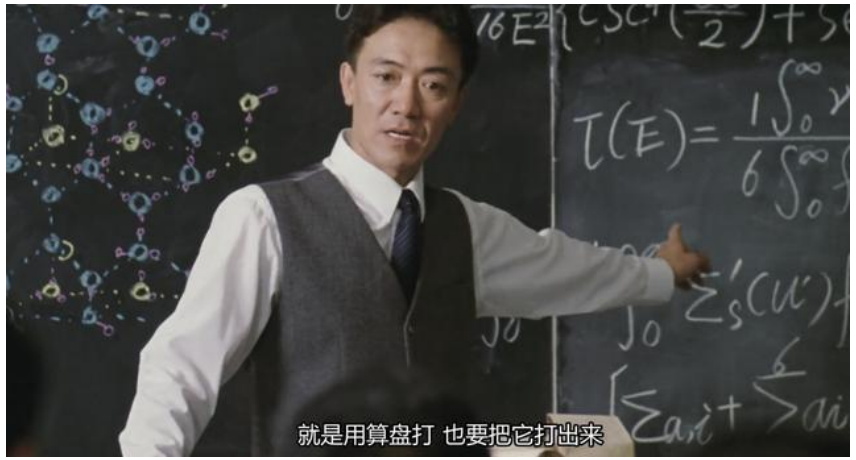
浪费比例：(m-n)/m，一般n=m-1。

常用方法：存储体数为质数，将向量变换成一维数组S，再对S进行处理。

扩展：从 ILLIAC 阵列处理机 到 GPU



俄乌冲突



影视作品中原子弹数据用算盘计算

主要内容：

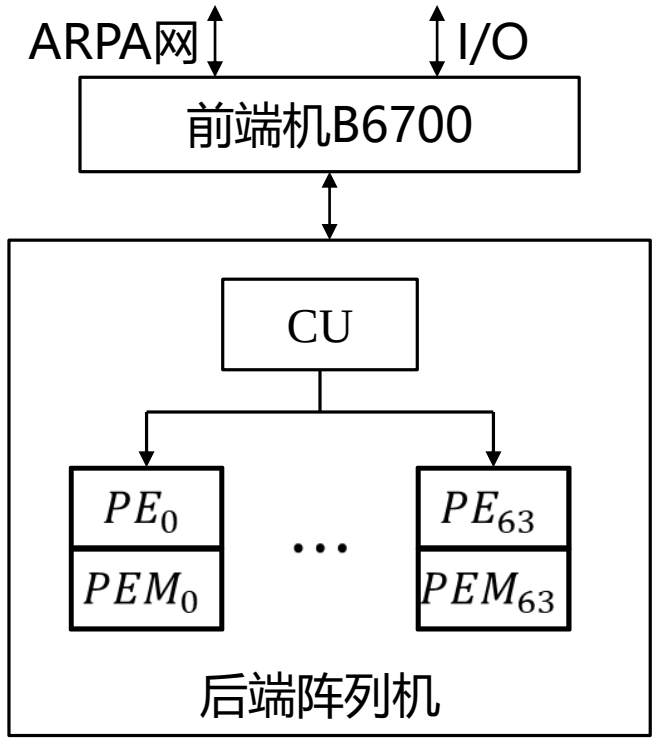
- **ILLIAC-IV阵列计算机**
- **SIMD 并行计算机算法**
- **GPU(Graphics Processing Unit)**

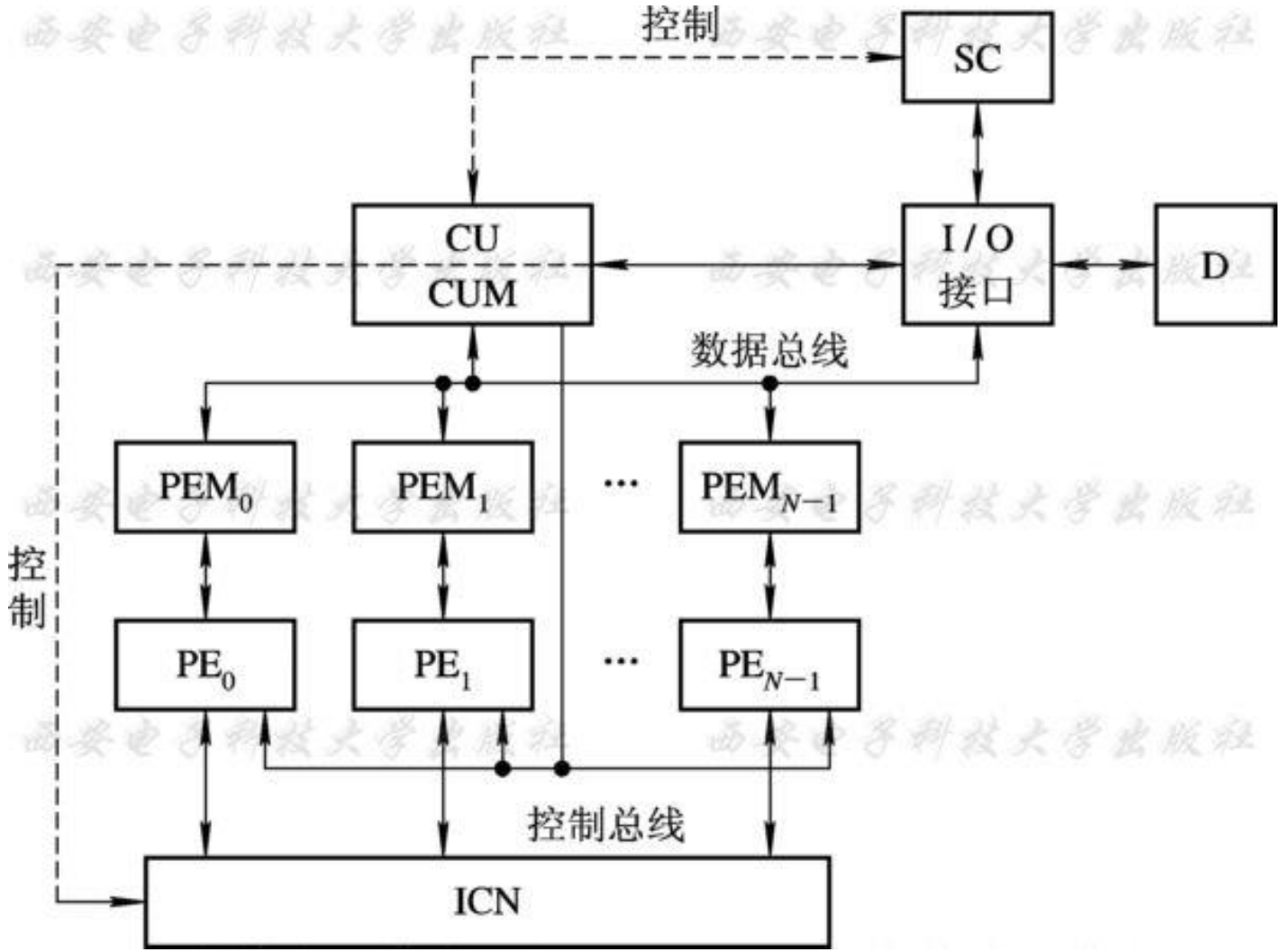
1、ILLIAC-IV阵列计算机（阵列机）

- 第一台全面使用大规模集成电路作为逻辑元件和存储器的计算机，标志着计算机进入第四代；
- 首个大规模并行（Massively parallel）计算机，同时也是最早实现单指令多数据（SIMD）的计算机。
- 原计划256个处理单元，算力1GFLOPS，每秒10亿次浮点运算，实际算力达到了15MFLOPS，每秒一亿五千万次的运算。

- **阵列机总体构型：具有分布式存储器的阵列处理机**

采用64个处理单元在统一控制下并行处理，分为两部分，即ILLIAC-IV阵列和ILLIAC-IV输入输出系统。





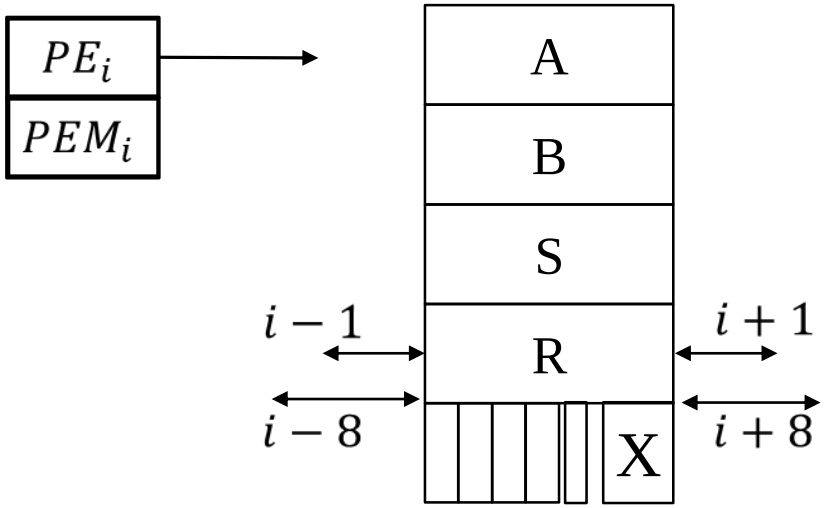
- **ILLIAC-IV的处理单元结构**

64个PU，每个PU_i包含：

64位的PEi

局部存储器 PEMi

存储逻辑部件MLUi



ILLIAC-IV PE结构

PEi内有

累加寄存器RGAi：存放第一操作数和操作结果

操作数寄存器RGB：存放加、减、乘、除等二元操作的第二操作数；

互连寄存器RGRi：与其他处理单元数据传送

模式寄存器RGM：8位，用于标记处理单元的活动状态

RGS、RGX……

- **ILLIAC-IV的阵列**

N=64 个处理单元构成8 × 8阵列

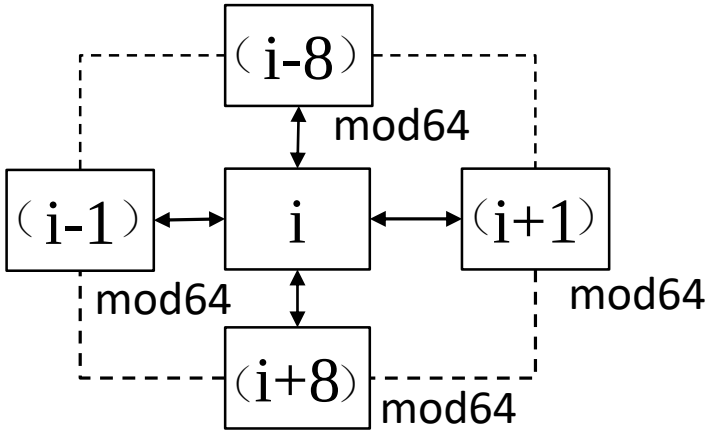
任意第i个处理单元PU_i与四个方向的单元连接

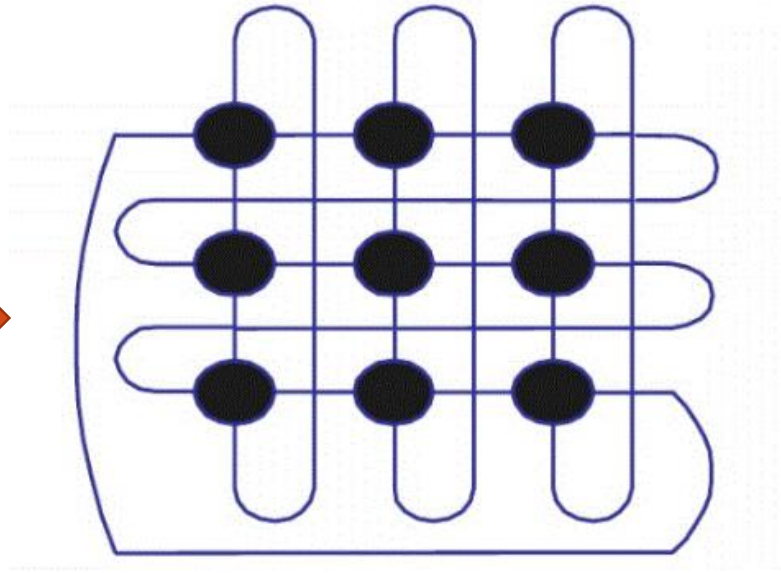
上：$(i-8) mod\ 64$

下：$(i+8) mod\ 64$

左：$(i-1) mod\ 64$

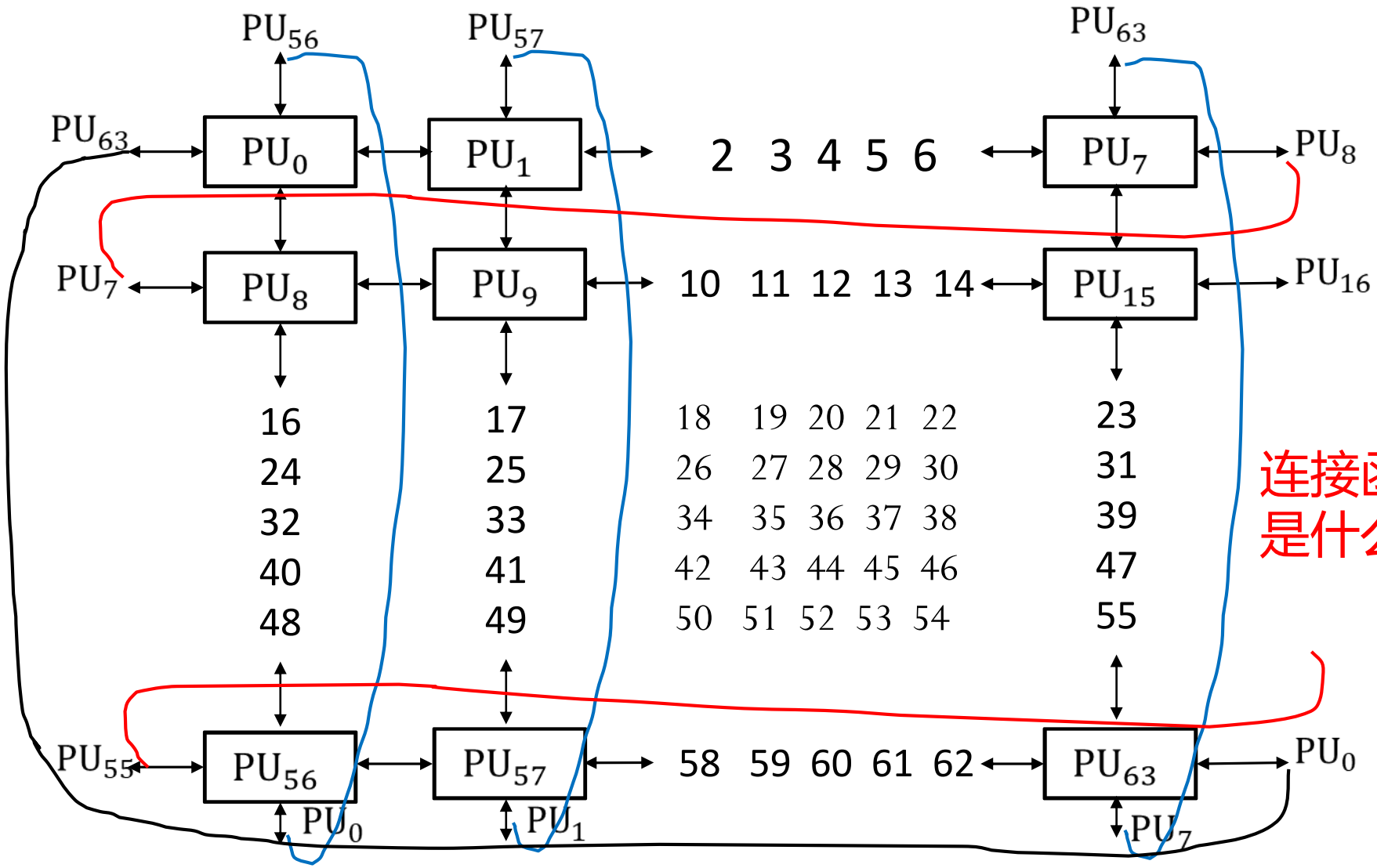
右：$(i+1) mod\ 64$



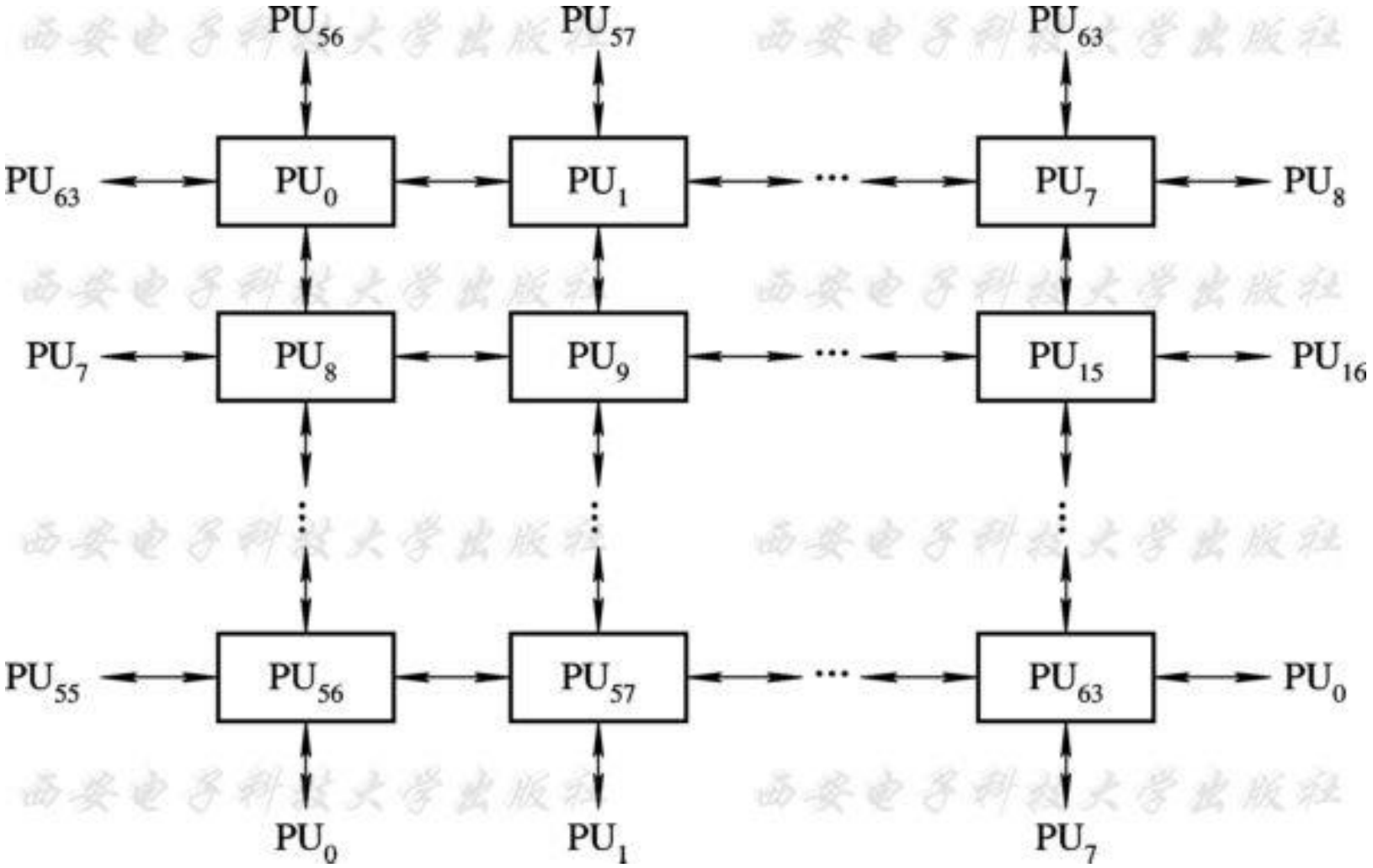


闭合螺线阵列结构

- **ILLIAC-IV的互连网络**　　以闭合螺线结构连接成阵列结构。



连接函数：$PM2_{+0}$、$PM2_{-0}$、$PM2_{+3}$、$PM2_{-3}$



如何定义、描述系统中各节点间的连接方式？

SIMD计算机的互连网络

Q：将数据从一个PU送达给另一个PU，最多几步可以实现？

从一个PE将数据达到另一个PE时，中间要经过若干个PE转送，
传送步数$S \leq \sqrt{N} - 1$，这里N为PE总数64，可得$S \leq 7$

例如：从PE_0到PE_{36}的距离，采用普通网格必须8步：　**采用普通网格至少8步**

$PE_0 \rightarrow PE_1 \rightarrow PE_2 \rightarrow PE_3 \rightarrow PE_4 \rightarrow PE_{12} \rightarrow PE_{20} \rightarrow PE_{28} \rightarrow PE_{36}$

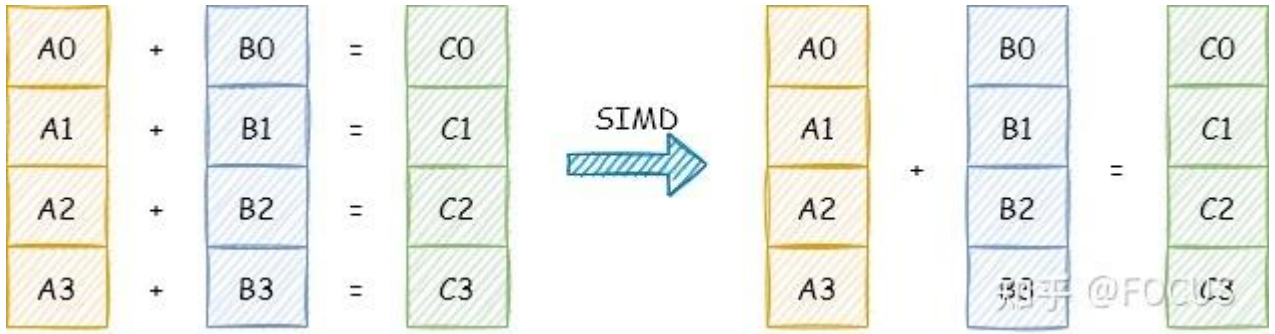
$PE_0 \rightarrow PE_8 \rightarrow PE_{16} \rightarrow PE_{24} \rightarrow PE_{32} \rightarrow PE_{33} \rightarrow PE_{34} \rightarrow PE_{35} \rightarrow PE_{36}$

如果采用闭合螺旋线，只需7步：

$PE_0 \rightarrow PE_{63} \rightarrow PE_{62} \rightarrow PE_{61} \rightarrow PE_{60} \rightarrow PE_{52} \rightarrow PE_{44} \rightarrow PE_{36}$

从PE_9到PE_{45}的最短路径：　$PE_9 \rightarrow PE_1 \rightarrow PE_{57} \rightarrow PE_{56} \rightarrow PE_{48} \rightarrow PE_{47} \rightarrow PE_{46} \rightarrow PE_{45}$

2、ILLIAC-IV的经典并行算法



与串行操作相比，速度提升了多少？

（1）矩阵加

- 局部存储器分配 ——使PEi尽量只访问自身的局部存储器

  让对应分量取相同存储地址

- 每条向量指令控制多个PU同时执行

```
LDA  K;    ADRN  K+1;    STO   K+2
```

（2）矩阵乘

$$C_{ij} = \sum_{k=0}^{7} a_{ik} \times b_{kj} \ (0 \le i \le 7, 0 \le j \le 7)$$

串行操作I, J, K三重循环

```
1.for i in range(0,7):
2.   for j in range(0,7):
3.       c(i,j)=0
4.     for k in range(0,7):
5.         c(i,j)=c(i,j)+a(i,k)*b(k,j);
```

与串行操作相比，速度提升了多少？

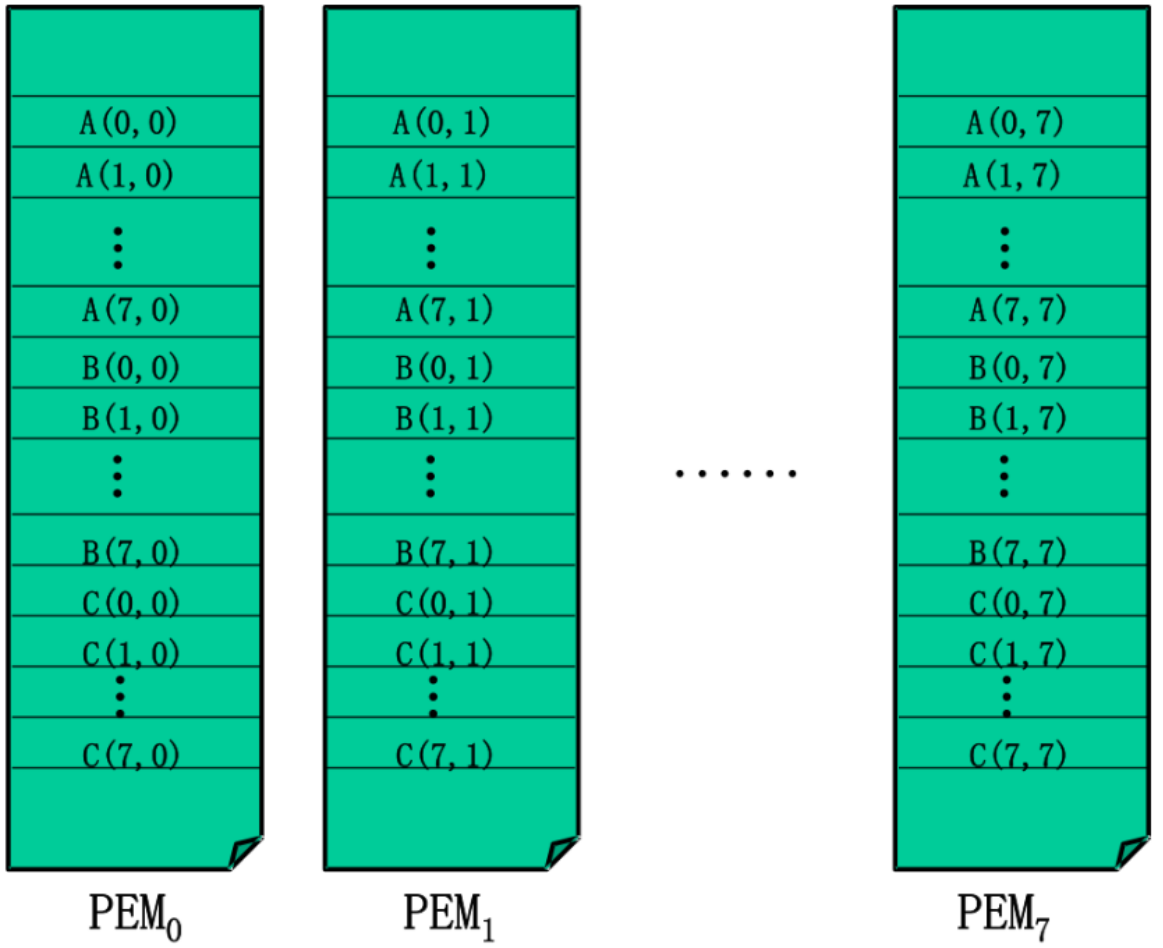
将分量分配到8个PE去做乘法，I, K两个循环

```
1.for i in range(0,7):
2.     c(i,j)=0
3.     for k in range(0,7):
4.         c(i,j)=c(i,j)+a(i,k)*b(k,j)
```

ILLIAC-IV阵列机完成矩阵乘的存储器数据分配

如果当64个处理器全部用来并行运算，需要在阵列存储器中重新恰当地分配数据，执行流程如下：

```
1.c(i,j)=0
2.i in range(0,7):
3.    c(i,j)=c(i,j)+a(i,k)*b(k,j)
```



由于要保证 8 个中间积 $A(I,K) \times B(K,J)$ 能够并行相加（累加），时间复杂度减少到 $\log_2 8$，速度提高接近$8/\log_2 8 = 2.7$倍

（3）累加和——成对递归相加算法

For k=0 to $\log_2 N-1$

C = C + SHFTR（C，2k）

- 运算前：把N个数分散到多个处理单元上，放入RGAi
- SHFTR（C，2k）：
  - 将全部PEi中的RGAi内容经过RGRi，向右传递2^k步
  - 将部分PEj设为不活跃 （j= 0到2^k-1）
  - 活跃的PEi，执行累加 RGAi += RGRi
- 运算后：RGAi内容存入PEMi

算法举例——图像平滑

输入： I为512×512像素大小的图像；每个点是一个8位无符号整数，用来表示256个灰度级，0位白色，255位黑色；

输出： 一个平滑后的图像S，要求$S(i,j)$是$I(i,j)$和它8个最近邻像素灰度级的平均值，边线上的灰度级置为0。

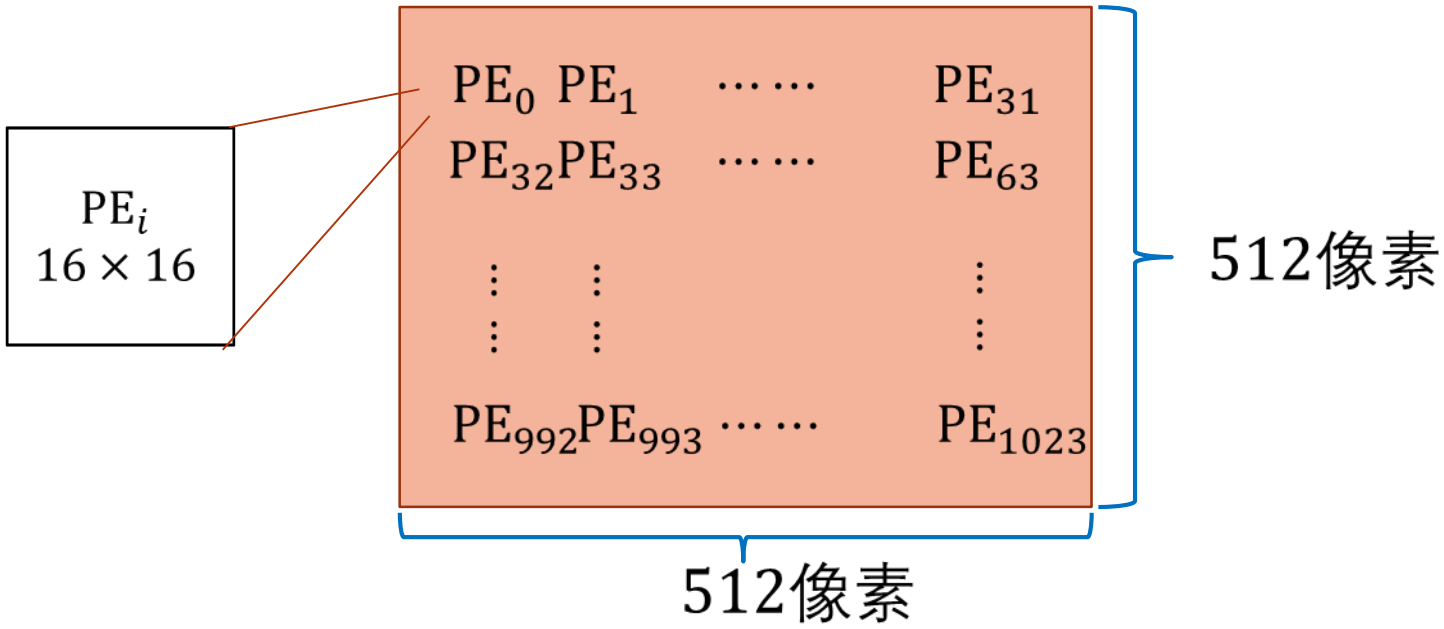
图像像素的8领域表示

$X_3(i-1,j-1)$	$X_2(i-1,j)$	$X_1(i-1,j+1)$
$X_4(i,j-1)$	$X(i,j)$	$X_0(i,j+1)$
$X_5(i+1,j-1)$	$X_6(i+1,j)$	$X_7(i+1,j+1)$

图像平滑处理公式

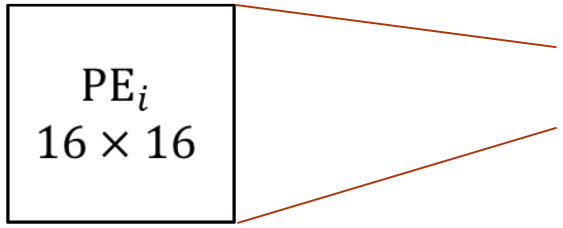
$$S(i,j) = \left[X + \sum_{I=0}^{7} x_i \right] / 9$$

每个PE处理一个16×16的子图像块，则需要1024个PE。



那具体每个PE如何操作呢？

数据分配

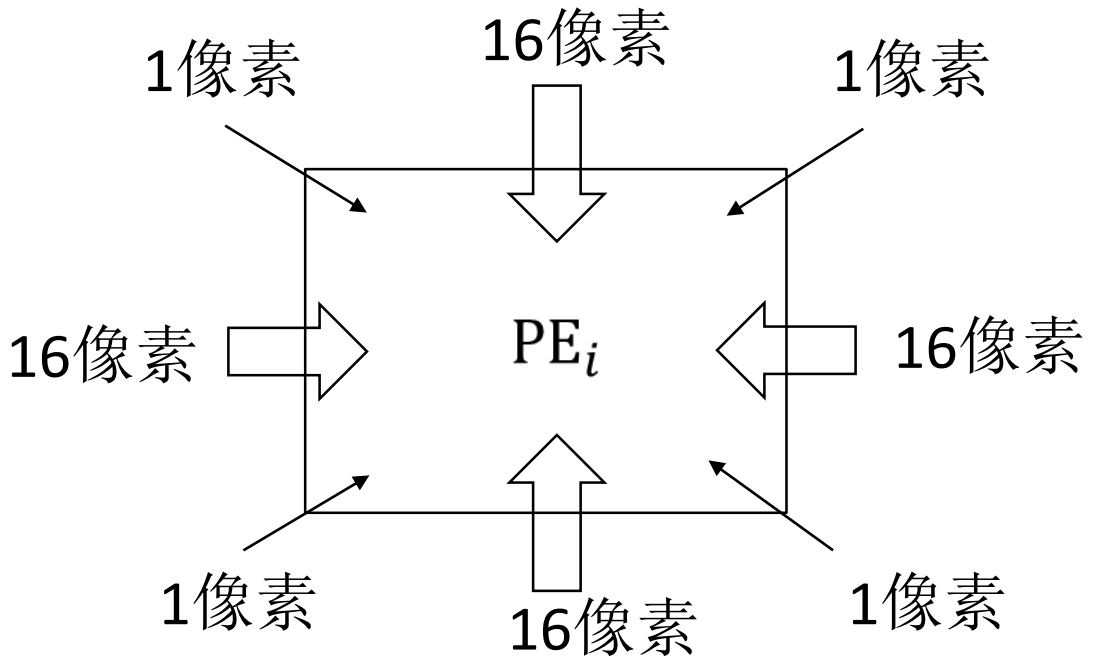


PE_0 存储行0~15和列0~15的子图像块，PE_1 存储行0~15和列16~31的子图像块，依次类推，每个PE平滑自己的子图像，即同时平滑操作

共需要$16 \times 16 = 256$次平滑操作

PE间传送

共需要$4 \times 16 + 4 = 68$次数据传送



性能分析

串行执行：

$$\text{共需} 512 \times 512 = 262144 \text{次平滑操作}$$

不计数据传送时间，并行比串行改进倍数为：

$$512 \times 512 \div 256 = 1024 \text{倍}$$

假定并行数据传送时间相当一次平滑操作时间，改进倍数为：

$$512 \times 512 \div （256 + 68） = 809 \text{倍}$$

对于图像像素的处理经常是独立且重复的操作，所以科学家尝试把这些高频操作硬件固化提升速度，随着我们技术的发展，就有了GPU的出现。

3、GPU

GPU的出现

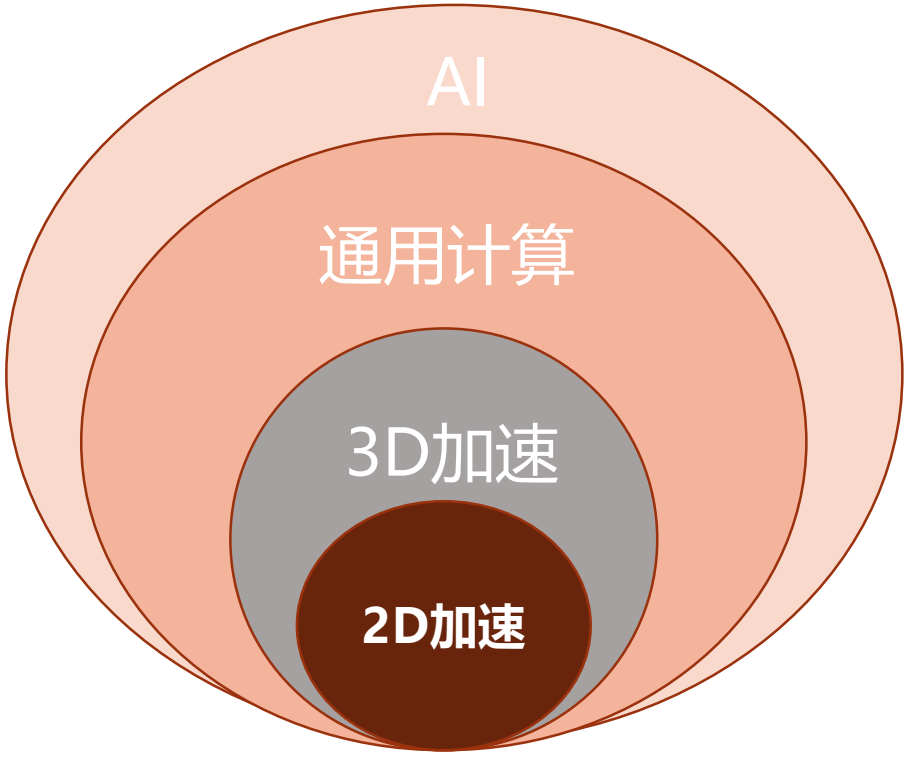
- “渲染”——具体来说就是几何点位置和颜色的计算，这两者的计算在数学上都是用四维向量和变换矩阵的乘法。随着3D的发展，渲染占用了CPU的很大部分时间，**硬件T&L(几何转换和光照处理)**并集成就诞生了GPU。

将立方体8个顶点坐标进行向量旋转，用顶点向量乘以旋转矩阵。8个定点做8次矩阵乘法操作，2000个顶点就是2000次。计算不复杂，但计算量大，且相互独立。

Direct X的立方体旋转

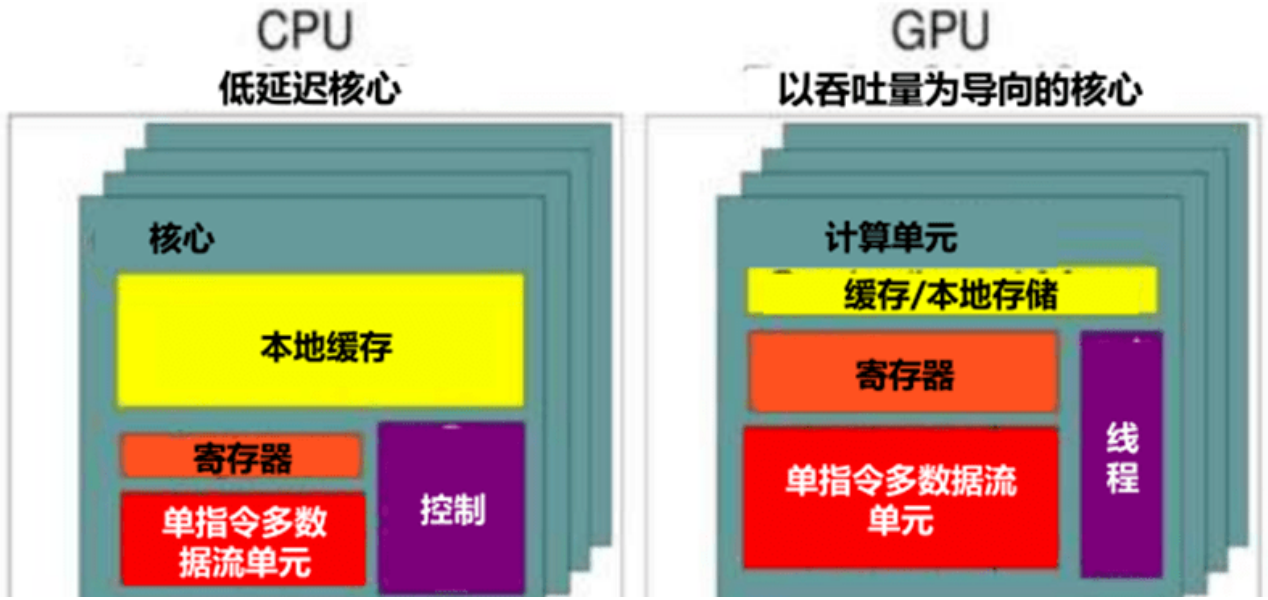
GPU的需求

过去20多年了，GPU的基本需求源于视频加速，2D/3D游戏。随后GPU运用自身在并行计算的优势，拓展到服务器、汽车、挖矿、人工智能和边缘计算领域。



GPU VS. CPU

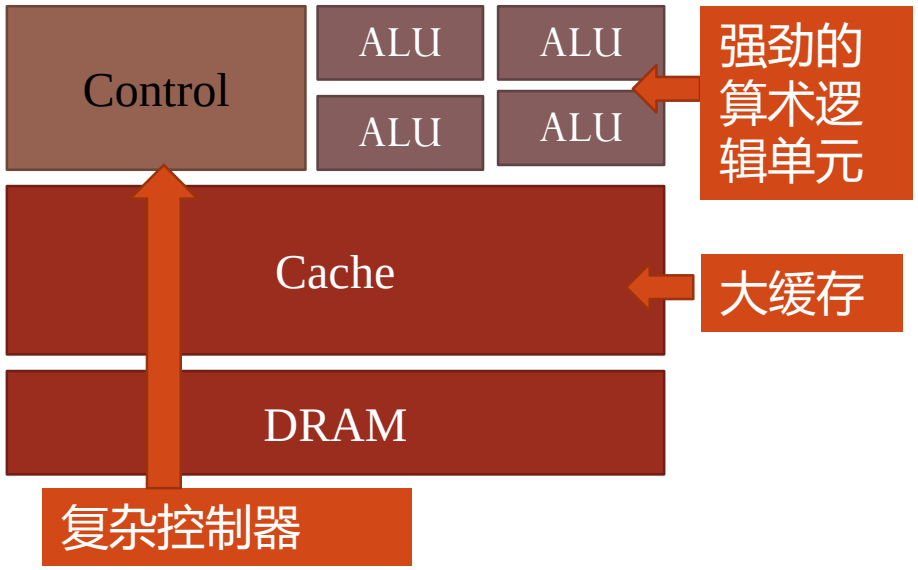
从芯片设计思路看，CPU是以低延迟为导向的计算单元，通常是转为串行处理而优化的几个核心组成；**GPU是以吞吐量为导向的计算单元**，由数以千计更小更高效的核心组成，为并行多任务设计。



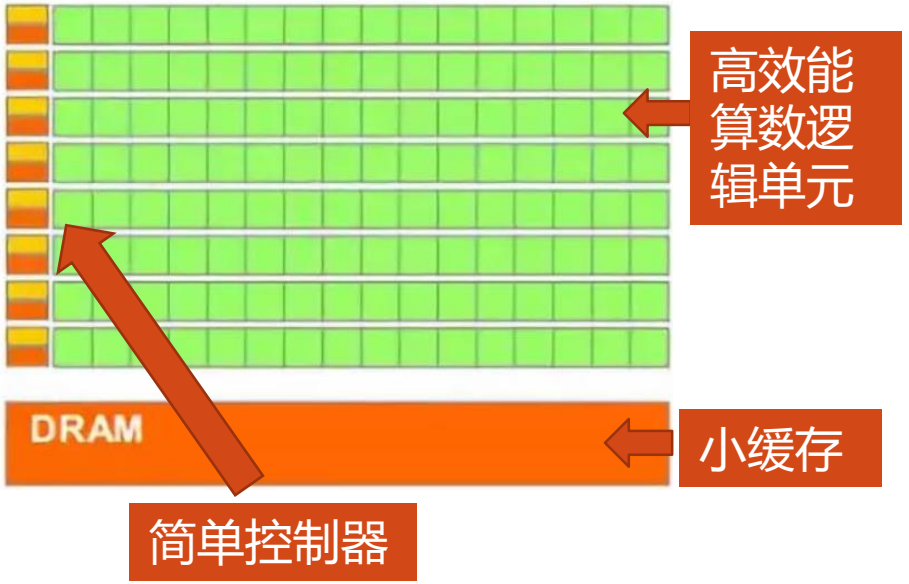
GPU和CPU的核心设计思想对比

微架构的不同导致CPU中大部分晶体管用于构建控制电路和缓存；
GPU的流处理器和显存控制器占据了大部分晶体管，控制器相对简单。

GPU和CPU的核心对比



CPU



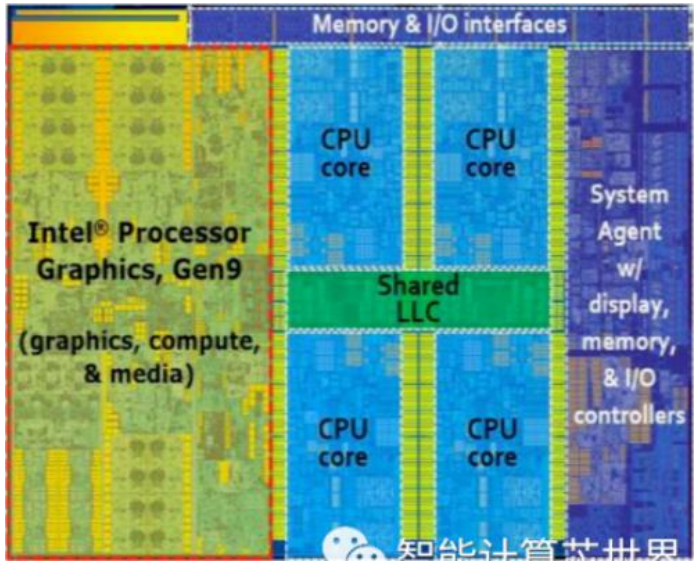
GPU

GPU的分类

- GPU根据接入方式分为独立GPU和集成GPU。
- 根据应用终端分为PC GPU，服务器GPU，移动GPU.

	类别	主要厂商、产品（及客户）
接入方式	独立GPU	AMD（Radeon系列）、NVIDIA（Geforce系列）
	集成GPU	英特尔（HD系列）、AMD（APU系列）
应用端	PC GPU	英特尔、NVIDIA、AMD
	服务器GPU	NVIDIA（Tesla）、AMD（FireStream）
	移动GPU	Imagination（PowerVR系列）；高通骁龙（Adreno系列）；ARM（公版Mali系列）；苹果A系列自研GPU

独立GPU

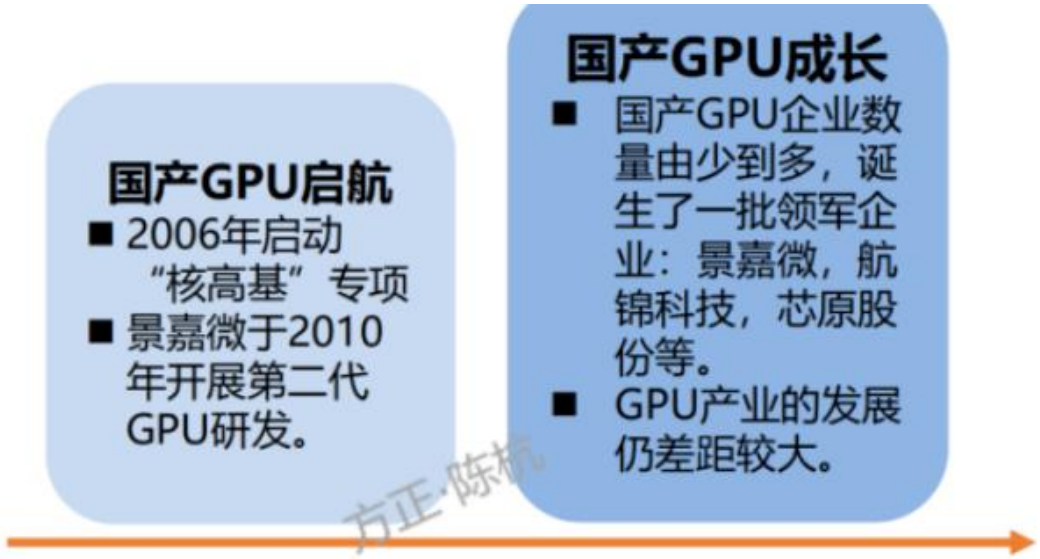


集成GPU核心

国产GPU的发展

GPU的发展史概括来说就是NVIDIA、AMD(ATI)的发展史。如今独立显卡领域主要由NVIDAI和AMD控制，集成显卡由英特尔和AMD控制。

国产GPU的发展落后于国产CPU，首先GPU对CPU有依赖，其次GPU技术难度很高。



景嘉微　航锦科技

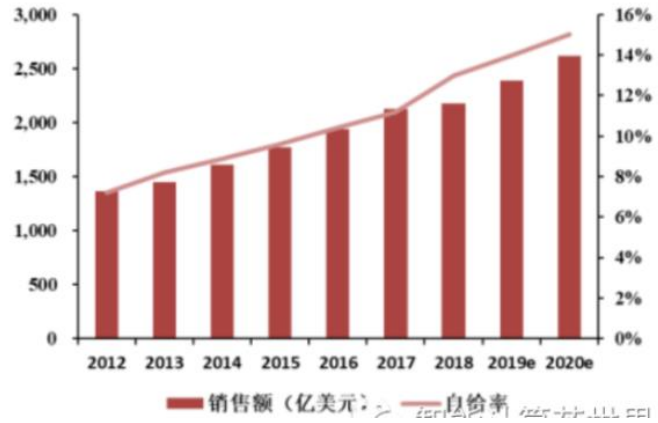
芯原股份　壁刃科技

中船重工　龙芯

兆芯　芯瞳半导体

芯动科技　海思

西邮微电　天数智芯



2012-2020大陆集成电路自给率