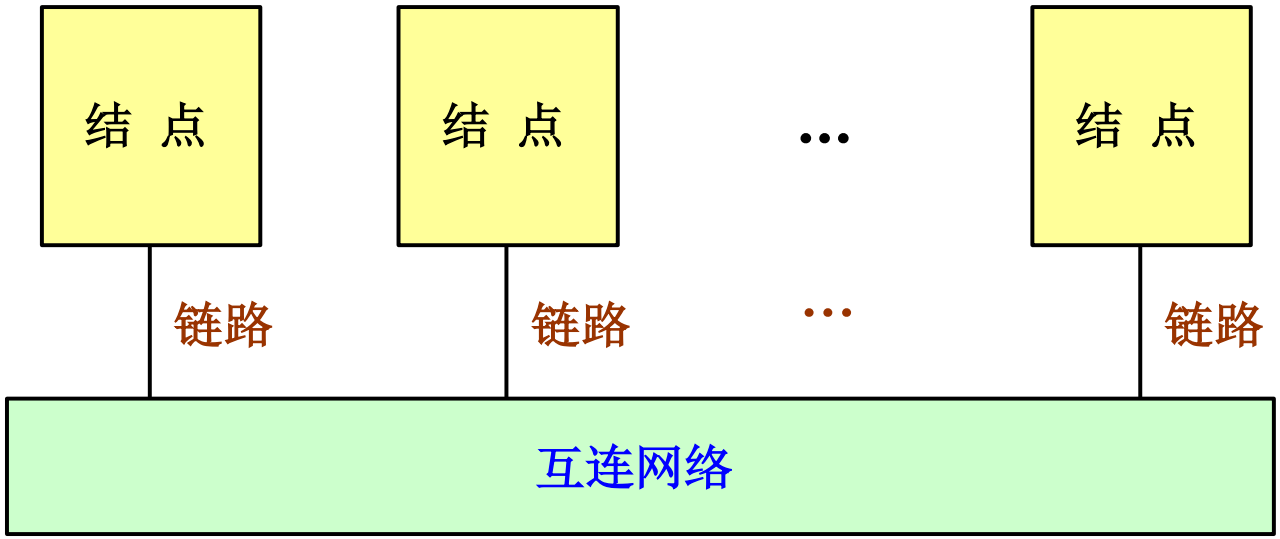
第7章 互连网络

7.1 互连函数
7.2 互连网络的结构参数与性能指标
7.3 静态互连网络
7.4 动态互连网络
7.5 消息传递机制

互连网络是一种由开关元件按照一定的拓扑结构和控制方式构成的网络，用来实现计算机系统中节点之间的相互连接。

- 节点：处理器、存储模块或其他设备。
- 在拓扑上，互连网络为输入节点到输出节点之间的一组互连或映射。
- SIMD计算机和MIMD计算机的关键组成部分。
- 3大要素：互连结构，开关元件，控制方式。



互连网络在系统中的位置

可以从4个不同的方面来描述互连网络

- **定时方式：有同步和异步两种。**
  - 同步系统：使用一个统一的时钟。

    SIMD阵列处理机就属于这一种类型。
  - 异步系统：没有统一的时钟，系统中的各个处理机都是独立地工作。
- **交换方法：有线路交换和分组交换两种。**
  - 线路交换：源结点和目的结点之间的物理通路在整个数据传送期间一直保持连接。
  - 分组交换：把信息分割成许多组（又称为包），将它们分别送入互连网络。

- 这些数据包可以通过不同的路径传送，到达目的结点后再拼合成原来的数据。
- 结点之间不存在固定连接的物理通路。

➢ **控制策略：有集中式和分散式两种**

- 集中控制方式：有一个全局的控制器接收所有的通信请求，并由它设置互连网络的开关连接。
- 分散控制方式：不存在全局的控制器，通信请求的处理和开关的设置由互连网络分散地进行。

- **拓扑结构：有静态和动态两种。**
  - 静态拓扑结构：在各结点之间有专用的连接通路，且在运行过程中不能改变。
  - 动态拓扑结构：可根据需要设置互连网络中的开关，从而对结点之间的连接通路进行重新组合，实现所要求的通信模式。

7.1 互连函数

7.1.1 互连函数

变量x：输入（设x=0，1，…，N－1）

函数f(x)：输出

通过数学表达式建立输入端号与输出端号的连接关系。即在互连函数f的作用下，输入端x连接到输出端f(x)。

- 互连函数反映了网络输入数组和输出数组之间对应的置换关系或排列关系。

（有时也称为置换函数或排列函数）

- 互连函数f(x)有时可以采用循环表示

  即：（x_0 x_1 x_2 … x_{j-1}）

  表示：$f(x_0)=x_1$，$f(x_1)=x_2$，…，$f(x_{j-1})=x_0$

  j称为该**循环的长度**。

- 设$n=\log_2 N$，则可以用n位二进制来表示N个输入端和输出端的二进制地址，互连函数表示为：

  $f(x_{n-1}x_{n-2}\cdots x_1x_0)$

7.1.2 几种基本的互连函数

介绍几种常用的基本互连函数及其主要特征。

1. 恒等函数

- **恒等函数：**实现同号输入端和输出端之间的连接。

$$I(x_{n-1}x_{n-2}\cdots x_1x_0)=x_{n-1}x_{n-2}\cdots x_1x_0$$

2. 交换函数

- **交换函数：**实现二进制地址编码中第k位互反的输入端与输出端之间的连接。

$$E(x_{n-1}x_{n-2}\cdots x_{k+1}x_kx_{k-1}\cdots x_1x_0)=x_{n-1}x_{n-2}\cdots x_{k+1}\bar{x}_kx_{k-1}\cdots x_1x_0$$

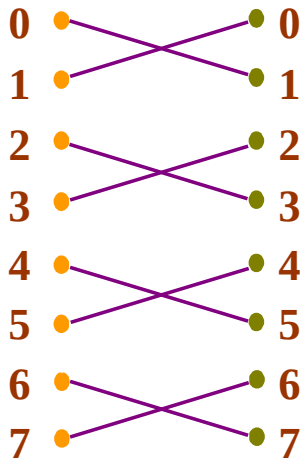
- 主要用于构造立方体互连网络和各种超立方体互连网络。
- 它共有$n=\log_2 N$种互连函数。

（N为节点个数）

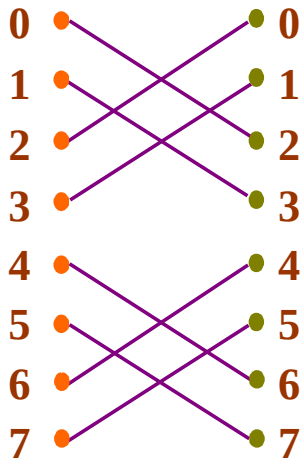
- 当N=8时，n=3，可得到常用的立方体互连函数：

$$Cube_0(x_2x_1x_0)=x_2x_1\bar{x}_0$$
$$Cube_1(x_2x_1x_0)=x_2\bar{x}_1x_0$$
$$Cube_2(x_2x_1x_0)=\bar{x}_2x_1x_0$$

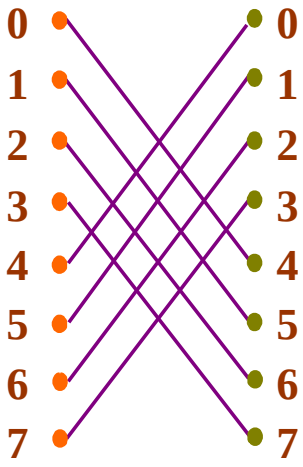
1. 变换图形



(a) $Cube_0$ 交换函数

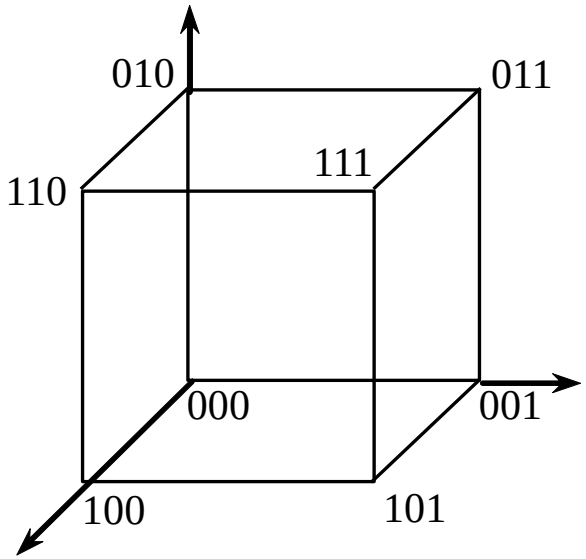


(b) $Cube_1$ 交换函数



(c) $Cube_2$ 交换函数

N=8 的立方体交换函数



立方体网络

3. 均匀洗牌函数

- **均匀洗牌函数：**将输入端分成数目相等的两半，前一半和后一半按类似均匀混洗扑克牌的方式交叉地连接到输出端（输出端相当于混洗的结果）。
  - 也称为混洗函数（置换）
  - 函数关系

$$\sigma\left(x_{n-1}x_{n-2}\cdots x_1x_0\right)=x_{n-2}x_{n-3}\cdots x_1x_0x_{n-1}$$

即把输入端的二进制编号循环左移一位。

- 互连函数（设为s）的**第k个子函数**：把s作用于输入端的二进制编号的低k位。
- 互连函数（设为s）的**第k个超函数**：把s作用于输入端的二进制编号的高k位。

例如：对于均匀洗牌函数

第k个子函数：

$$\sigma_{(k)}(x_{n-1}\cdots x_k \mid x_{k-1}x_{k-2}\cdots x_0) = x_{n-1}\cdots x_k \mid x_{k-2}\cdots x_0x_{k-1}$$

即把输入端的二进制编号中的低k位循环左移一位。

第k个超函数：

$$\sigma^{(k)}(x_{n-1}x_{n-2}\cdots x_{n-k} \mid x_{n-k-1}\cdots x_1x_0) = x_{n-2}\cdots x_{n-k}x_{n-1} \mid x_{n-k-1}\cdots x_1x_0$$

即把输入端的二进制编号中的高k位循环左移一位。

下列等式成立：

$$\sigma^{(n)}(X)=\sigma_{(n)}(X)=\sigma(X)$$

$$\sigma^{(1)}(X)=\sigma_{(1)}(X)=X$$

- 对于任意一种函数f(x)，如果存在g(x)，使得

$$f(x)\times g(x)=I(x)$$

则称g(x)是**f(x)的逆函数**，记为$f^{-1}(x)$。

$$f^{-1}(x)=g(x)$$

- **逆均匀洗牌函数**：将输入端的二进制编号循环右移一位而得到所连接的输出端编号。

- 互连函数

$$\sigma^{-1}\left(x_{n-1}x_{n-2}\cdots x_1x_0\right)=x_0x_{n-1}x_{n-2}\cdots x_1$$

  - 逆均匀洗牌是均匀洗牌的逆函数

➢ **当N=8时，有：**

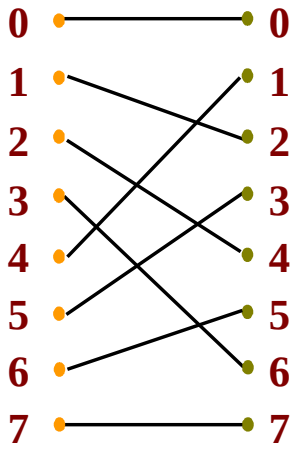
$$\sigma(x_2x_1x_0)=x_1x_0x_2$$

$$\sigma_{(2)}(x_2x_1x_0)=x_2x_0x_1$$

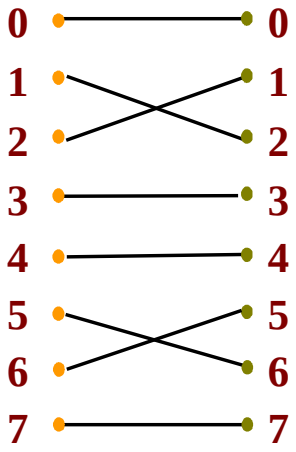
$$\sigma^{(2)}(x_2x_1x_0)=x_1x_2x_0$$

$$\sigma^{-1}(x_2x_1x_0)=x_0x_2x_1$$

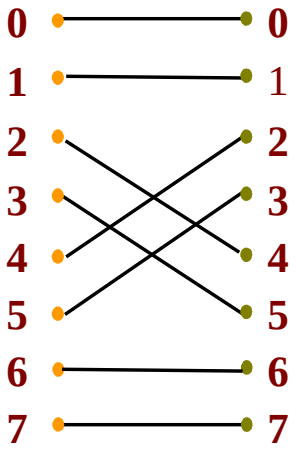
- N=8 的均匀洗牌和逆均匀洗牌函数



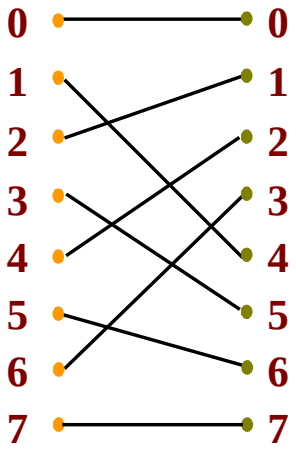
(a) 均匀洗牌函数 σ



(b) 子洗牌函数 $\sigma_{(2)}$



(c) 超洗牌函数 $\sigma^{(2)}$



(d) 逆均匀洗牌函数 σ^{-1}

N=8 的均匀洗牌函数

4. 蝶式函数

- **蝶式互连函数：把输入端的二进制编号的最高位与最低位互换位置，便得到了输出端的编号。**

$$\beta\left(x_{n-1}x_{n-2}\cdots x_1x_0\right)=x_0x_{n-2}\cdots x_1x_{n-1}$$

- **第k个子函数**

$$\beta_{(k)}(x_{n-1}\cdots x_kx_{k-1}x_{k-2}\cdots x_1x_0)=x_{n-1}\cdots x_kx_0x_{k-2}\cdots x_1x_{k-1}$$

把输入端的二进制编号的低k位中的最高位与最低位互换。

- **第k个超函数**

$$\beta^{(k)}(x_{n-1}x_{n-2}\cdots x_{n-k+1}x_{n-k}x_{n-k-1}\cdots x_1x_0)=x_{n-k}x_{n-2}\cdots x_{n-k+1}x_{n-1}x_{n-k-1}\cdots x_1x_0$$

把输入端的二进制编号的高k位中的最高位与最低位互换。

- 下列等式成立

$$\beta^{(n)}(X)=\beta_{(n)}(X)=\beta(X)$$

$$\beta^{(1)}(X)=\beta_{(1)}(X)=X$$

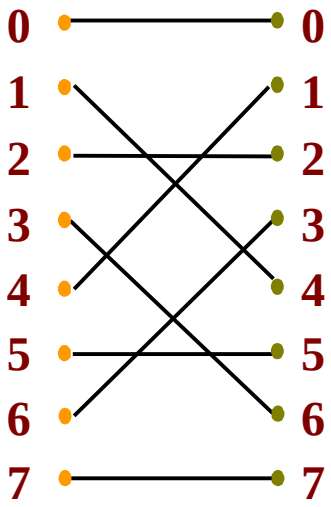
- 当N=8时，有：

$$\beta(x_2x_1x_0)=x_0x_1x_2$$

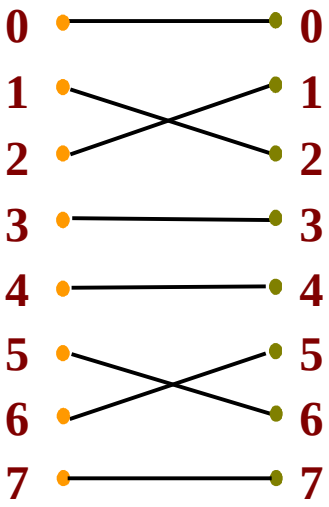
$$\beta_{(2)}(x_2x_1x_0)=x_2x_0x_1$$

$$\beta^{(2)}(x_2x_1x_0)=x_1x_2x_0$$

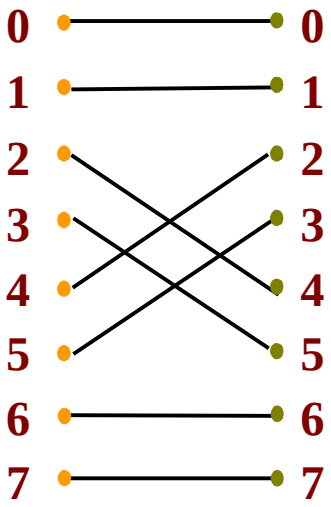
- 蝶式变换与交换变换的多级组合可作为构成方体多级网络的基础。



(a) $\beta = \rho$



(b) $\beta_{(2)} = \rho_{(2)}$



(c) $\beta^{(2)} = \rho^{(2)}$

N=8 的蝶式函数和反位序函数

5. 反位序函数

- **反位序函数**：将输入端二进制编号的位序颠倒过来求得相应输出端的编号。
  - 互连函数

$$\rho\left(x_{n-1}x_{n-2}\cdots x_1x_0\right)=x_0x_1\cdots x_{n-2}x_{n-1}$$

- **第k个子函数**

$$\rho_{(k)}(x_{n-1}\cdots x_kx_{k-1}x_{k-2}\cdots x_1x_0)=x_{n-1}\cdots x_kx_0x_1\cdots x_{k-2}x_{k-1}$$

即把输入端的二进制编号的低k位中各位的次序颠倒过来。

- 第k个超函数

$$\rho^{(k)}(x_{n-1}x_{n-2}\cdots x_{n-k+1}x_{n-k}x_{n-k-1}\cdots x_1x_0)=x_{n-k}x_{n-k+1}\cdots x_{n-2}x_{n-1}x_{n-k-1}\cdots x_1x_0$$

即把输入端的二进制编号的高k位中各位的次序颠倒过来。

- 下列等式成立

$$\rho^{(n)}(X)=\rho_{(n)}(X)=\rho(X)$$

$$\rho^{(1)}(X)=\rho_{(1)}(X)=X$$

- 当N=8时，有：

$$\rho(x_2x_1x_0)=x_0x_1x_2$$

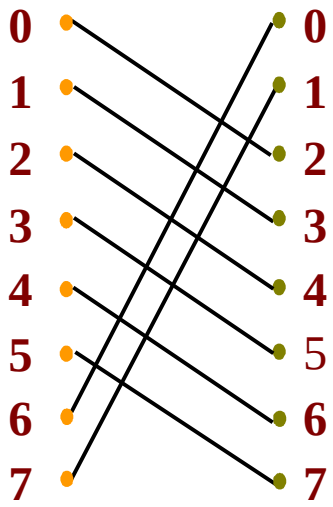
$$\rho_{(2)}(x_2x_1x_0)=x_2x_0x_1$$

$$\rho^{(2)}(x_2x_1x_0)=x_1x_2x_0$$

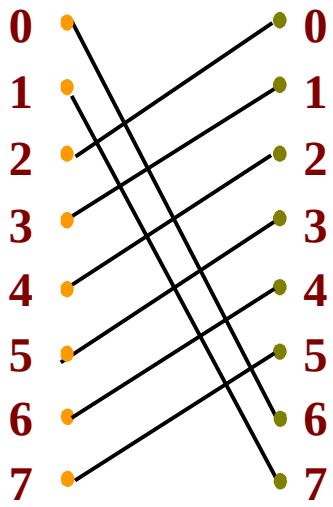
6. 移数函数

- **移数函数**：将各输入端都错开一定的位置（模N）后连到输出端。
  - 函数式

$$\alpha(x)=（x\pm k）\bmod N \quad 1\le x\le N-1，1\le k\le N-1$$



(a) 左移移数函数 k＝2



(b) 右移移数函数 k＝2

7. PM2I函数

- P和M分别表示加和减，2I表示2^i。
  - 该函数又称为“加减2^i”函数。
- PM2I函数：一种移数函数，将各输入端都错开一定的位置（模N）后连到输出端。
- 互连函数

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

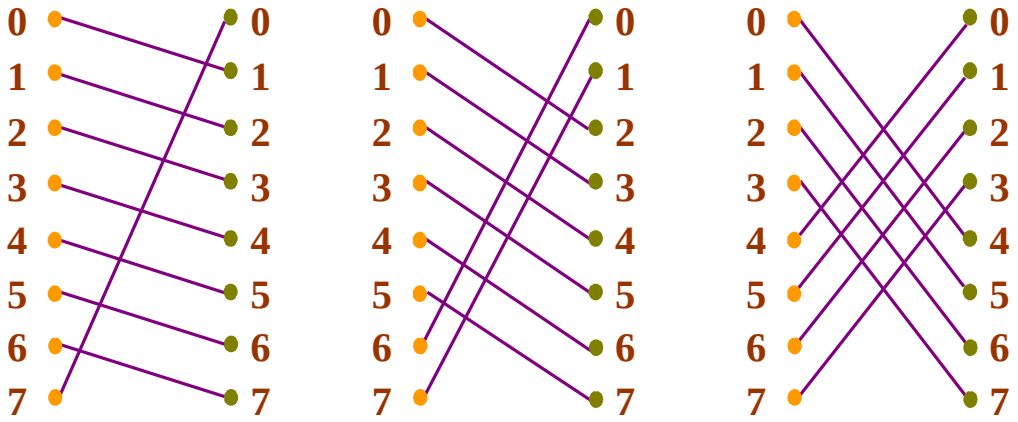
$$PM2_{-i}(x) = x - 2^i \bmod N$$

其中：

$0 \leqslant x \leqslant N-1$，$0 \leqslant i \leqslant n-1$，$n = \log_2 N$，N为节点数。

- PM2I互连网络共有2n个互连函数。

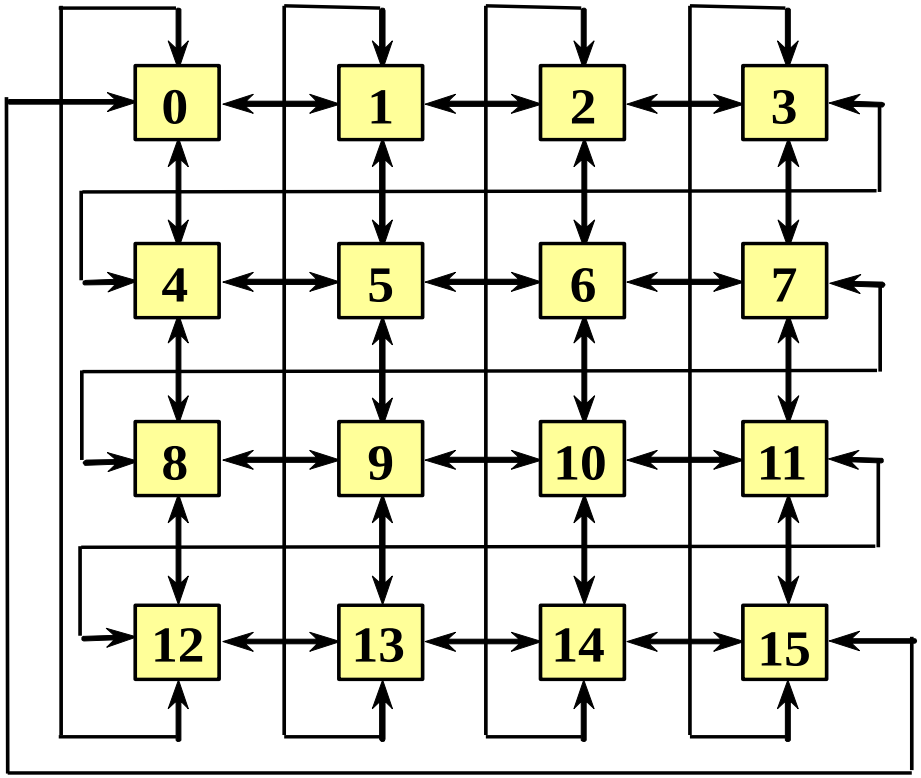
- 当N＝8时，有6个PM2I函数：
  - $PM2_{+0}$ ： (0 1 2 3 4 5 6 7)
  - $PM2_{-0}$ ： (7 6 5 4 3 2 1 0)
  - $PM2_{+1}$ ： (0 2 4 6) (1 3 5 7)
  - $PM2_{-1}$ ： (6 4 2 0) (7 5 3 1)
  - $PM2_{+2}$ ： (0 4) (1 5) (2 6) (3 7)
  - $PM2_{-2}$ ： (4 0) (5 1) (6 2) (7 3)



N=8 的PM2I函数

➢ 阵列计算机ILLIAC Ⅳ

采用$PM2_{\pm 0}$和$PM2_{\pm n/2}$构成其互连网络，实现各处理单元之间的上下左右互连 。



用移数函数构成ILLIAC Ⅳ 阵列机的互连网络

(新版)例7.1 现有16个处理器，编号分别为0，1，…，15，用一个N=16的互连网络互连。处理器i的输出通道连接互连网络的输入端i，处理器i的输入通道连接互连网络的输出端i。当该互连网络实现的互连函数分别为：

（1）$Cube_3$

（2）$PM2_{+3}$

（3）$PM2_{-0}$

（4）σ

（5）σ（σ）

时，分别给出与第13号处理器所连接的处理器号。

解：（1）由 $Cube_3(x_3x_2x_1x_0)=\bar{x}_3x_2x_1x_0$ ，

得$Cube_3$（1101）= 0101，即处理器13连接到处理器5。

令$Cube_3$（ $x_3x_2x_1x_0$ ） = 1101，得$x_3x_2x_1x_0$= 0101，故与处理器13相连的是处理器5。

所以处理器13与处理器5双向互连。

（2）由$PM2_{+3} = j+2^3 \bmod 16$，得$PM2_{+3}$（13）= $13+2^3 = 5$， 即处理器13连接到处理器5。

令$PM2_{+3}$（j）= $j+2^3 \bmod 16 = 13$，得j= 5，故与处理器13相连的是处理器5。

所以处理器13与处理器5双向互连。

（3）由$PM2_{-0}$（j）= $j-2^0 \bmod 16$，得$PM2_{-0}$（13）= $13-2^0 = 12$，即处理器13连接到处理器12。

令$PM2_{-0}$（j）= $j-2^0 \bmod 16 = 13$，得j= 14，故与处理器13相连的是处理器14。

所以处理器13连至处理器12，而处理器14连至处理器13。

（4）由$\sigma(x_3x_2x_1x_0)= x_2x_1x_0x_3$，得$\sigma(1101)= 1011$，即处理器13连接到处理器11。

令$\sigma(x_3x_2x_1x_0)= 1101$，得$x_3x_2x_1x_0= 1110$，故与处理器13相连的是处理器14。

所以处理器13连至处理器11，而处理器14连至处理器13。

（5）由$\sigma(\sigma(x_3x_2x_1x_0))= x_1x_0x_3x_2$，得$\sigma(\sigma(1101))= 0111$，即处理器13连接到处理器7。

令$\sigma(\sigma(x_3x_2x_1x_0))= 1101$，得$x_3x_2x_1x_0= 0111$，故与处理器13相连的是处理器7。

所以处理器13与处理器7双向互连。

7.2 互连网络的结构参数与性能指标

7.2.1 互连网络的结构参数

1. 网络通常是用有向边或无向边连接有限个节点的图来表示。
2. 互连网络的主要特性参数有：
   - **网络规模N**：网络中节点的个数。

     表示该网络所能连接的部件的数量。
   - **节点度d**：与节点相连接的边数（通道数），包括入度和出度。

- 进入节点的边数叫入度。
- 从节点出来的边数叫出度。

- **节点距离：对于网络中的任意两个节点，从一个节点出发到另一个节点终止所需要跨越的边数的最小值。**
- **网络直径D：网络中任意两个节点之间距离的最大值。**

  网络直径应当尽可能地小。
- **等分宽度b：把由N个节点构成的网络切成节点数相同（N/2）的两半，在各种切法中，沿切口边数的最小值。**

- 线等分宽度：B＝b×w
  - 其中：w为通道宽度（用位表示）
  - 该参数主要反映了网络最大流量。

➢ **对称性：从任何节点看到的拓扑结构都是相同的网络称为对称网络。**

对称网络比较容易实现，编程也比较容易。

7.2.2 互连网络的性能指标

评估互连网络性能的**两个基本指标**：时延和带宽

1. 通信时延

指从源节点到目的节点传送一条消息所需的总时间，它由以下4部分构成：

- **软件开销**：在源节点和目的节点用于收发消息的软件所需的执行时间。
  - 主要取决于两端端节点处理消息的软件内核。
- **通道时延**：通过通道传送消息所花的时间。
  - 通路时延 = 消息长度/通道带宽
  - 通常由瓶颈链路的通道带宽决定。

- **选路时延**：消息在传送路径上所需的一系列选路决策所需的时间开销。
  - 与传送路径上的节点数成正比。
- **竞争时延**：多个消息同时在网络中传送时，会发生争用网络资源的冲突。为避免或解决争用冲突所需的时间就是竞争时延。
  - 很难预测，它取决于网络的传输状态。

2. 网络时延

通道时延与选路时延的和。

- 它是由网络硬件特征决定的，与程序行为和网络传输状态无关。

3. 端口带宽

- 对于互连网络中的任意一个端口来说，其端口带宽是指单位时间内从该端口传送到其他端口的最大信息量。
  - 在对称网络中，端口带宽与端口位置无关。网络的端口带宽与各端口的端口带宽相同。
  - 非对称网络的端口带宽则是指所有端口带宽的最小值。

4. 聚集带宽

网络从一半节点到另一半节点，单位时间内能够传送的最大信息量。

例如，HPS是一种对称网络

网络规模N的上限：512

端口带宽：40MB/s

HPS的聚集带宽：（40MB/s×512）/2＝10.24GB/s

5. 等分带宽

与等分宽度对应的切平面中，所有边合起来单位时间所能传送的最大信息量。

7.3 静态互连网络

互连网络通常可以分为两大类：

- **静态互连网络**

  各节点之间有固定的连接通路、且在运行中不能改变的网络。

- **动态互连网络**

  由交换开关构成、可按运行程序的要求动态地改变连接状态的网络。

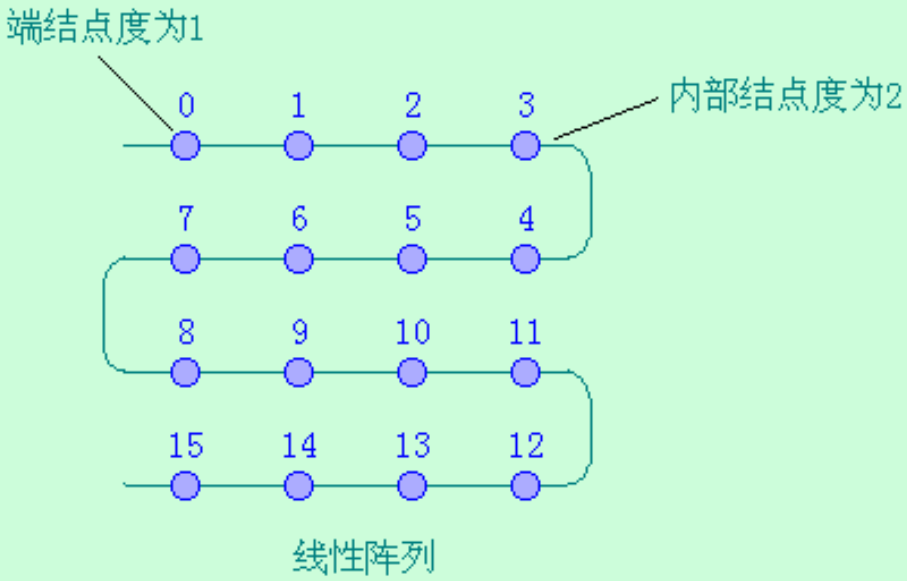
下面介绍几种静态互连网络。

（其中：N表示节点的个数）

1. 线性阵列 一种一维的线性网络，其中N个节点用N-1个链路连成一行。

- 端节点的度：1
- 其余节点的度：2
- 直径：N－1
- 等分宽度b=1

线 性 阵 列



线性阵列与总线的区别：

总线是通过切换与其连接的许多结点来实现时分特性的

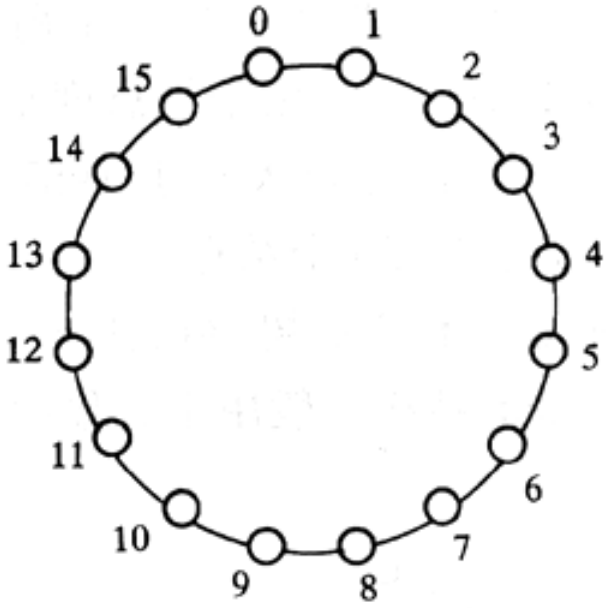
线性阵列允许不同的源结点和目的结点对并行地使用其不同的部分

2. 环和带弦环

➢ 环

用一条附加链路将线性阵列的两个端点连接起来而构成。可以单向工作，也可以双向工作。

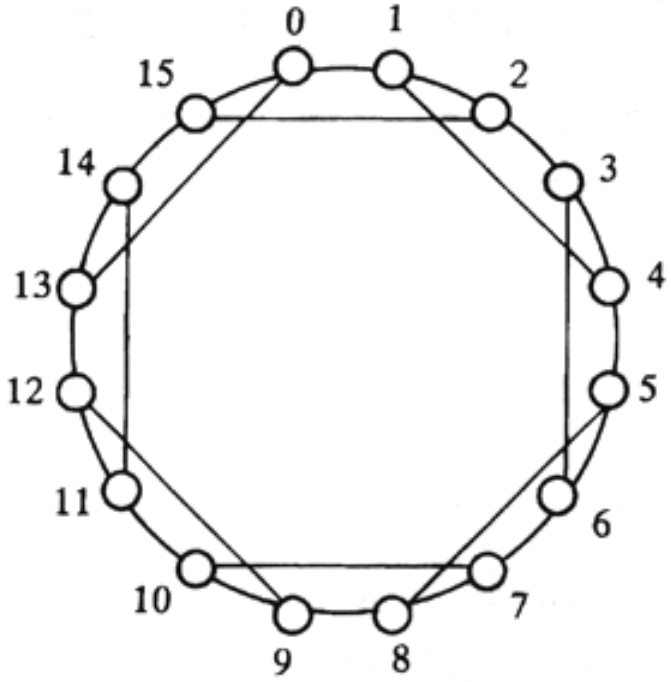
- 对称
- 节点的度：2
- 双向环的直径：N/2
- 单向环的直径：N-1
- 环的等分宽度b=2



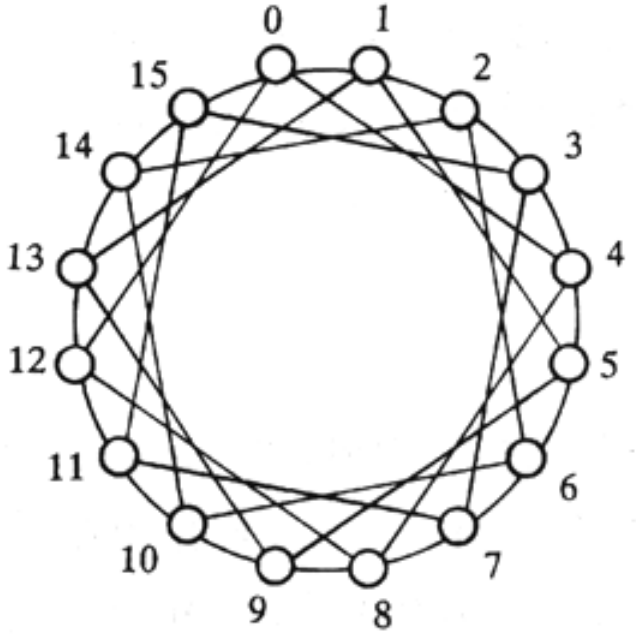
(b)环

- **带弦环**

增加的链路愈多，节点度愈高，网络直径就愈小。

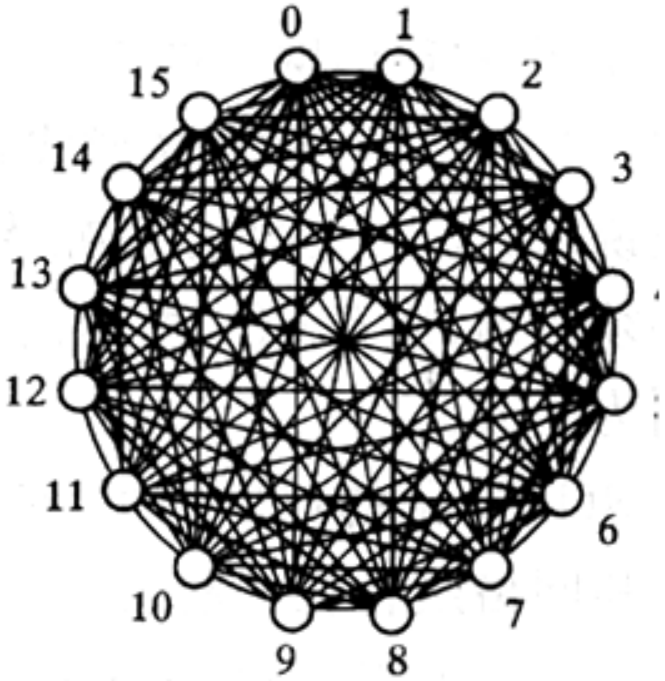


(c)度为 3 的带弦环



(d)度为 4 的带弦环(与 Illiac 网相同)

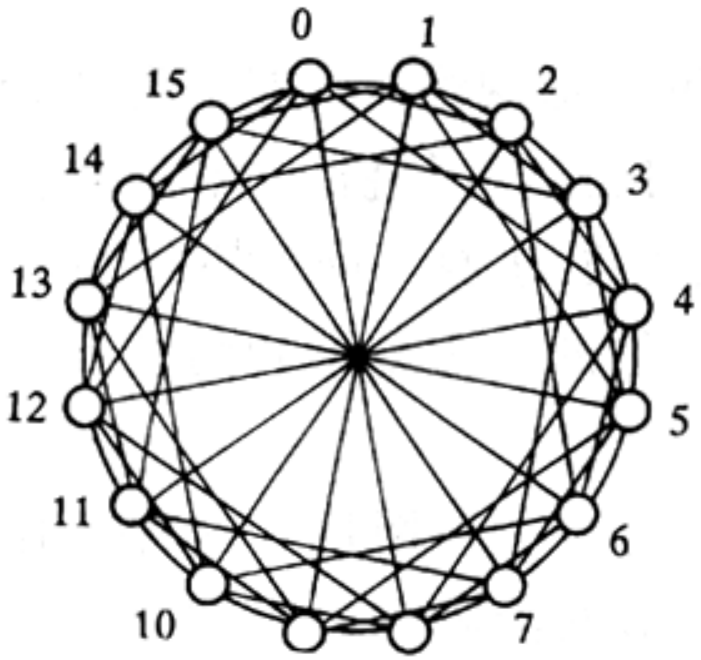
- **全连接网络**
  - 节点度：15
  - 直径最短，为1。



(f)全连接

3. 循环移数网络

- 通过在环上每个节点到所有与其距离为2的整数幂的节点之间都增加一条附加链而构成。



(e)循环移数网络

N=16

- 节点度：7
- 直径：2

- 一般地，如果 $|j-i|=2^r$（$r=0,1,2,\ldots,n-1$，$n=\log_2 N$），则节点i与节点j连接。
  - 节点度：$2n-1$
  - 直径：$n/2$
  - 网络规模$N=2^n$

4. 树形和星形

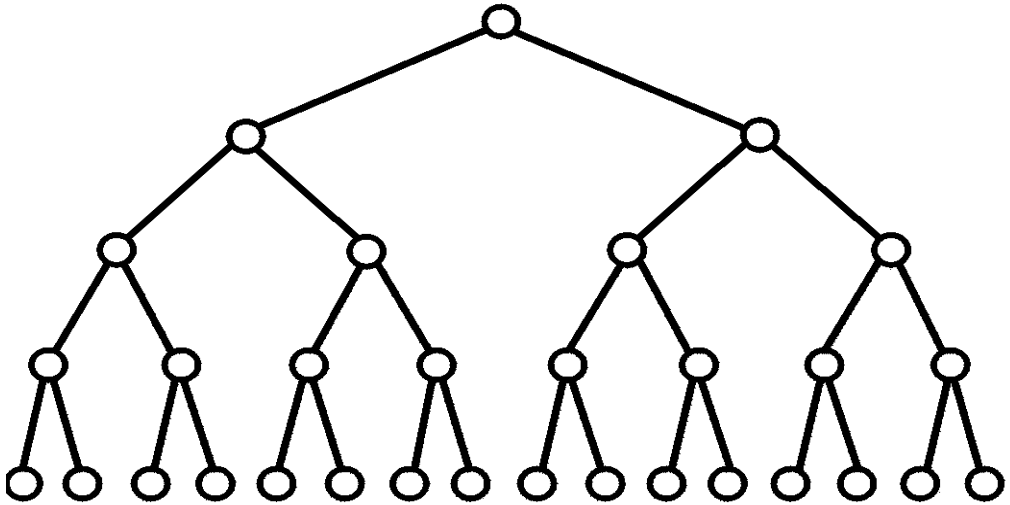
- 一棵5层31个节点的二叉树

  一般说来，一棵k层完全平衡的二叉树有$N=2^k-1$个节点。

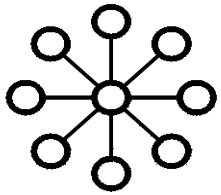
  - 最大节点度：3
  - 直径：$2(k-1)$
  - 等分宽度b=1

- 星形

  - 节点度较高，为$N-1$。
  - 直径较小，是一常数2。等分宽度$b=\lfloor N/2 \rfloor$
  - 可靠性比较差，只要中心节点出故障，整个系统就会瘫痪。

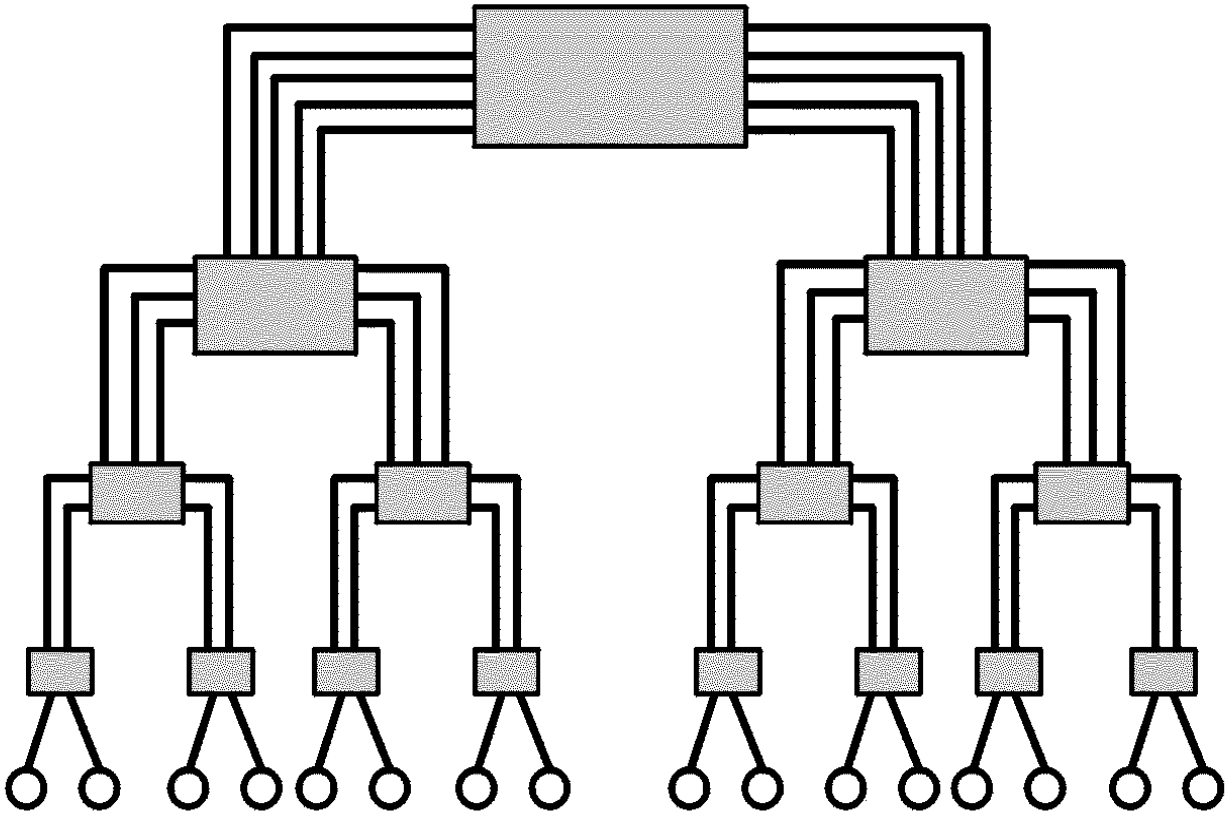


(a) 二叉树



(b) 星形

5. 胖树形



(c) 二叉胖树

6. 网格形和环网形

➢ 网格形

- 一个3×3的网格形网络
- 一个规模为$N=n\times n$的2维网格形网络
  - 内部节点的度d=4
  - 边节点的度d=3
  - 角节点的度d=2
  - 网络直径D=2（n-1）
  - 等分宽度b=n
- 一个由$N=n^k$ 个节点构成的k维网格形网络（每维n个节点）的内部节点度d=2k，网络直径D=k(n-1) 。

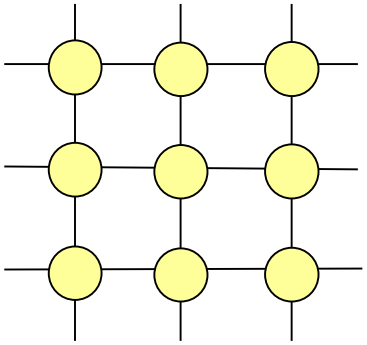
➢ Illiac网络

- 名称来源于采用了这种网络的Illiac Ⅳ计算机
- 把2维网格形网络的每一列的两个端节点连接起来，再把每一行的尾节点与下一行的头节点连接起来，并把最后一行的尾节点与第一行的头节点连接起来。
- 一个规模为n×n的Illiac网络
  - 所有节点的度d=4
  - 网络直径D=n-1

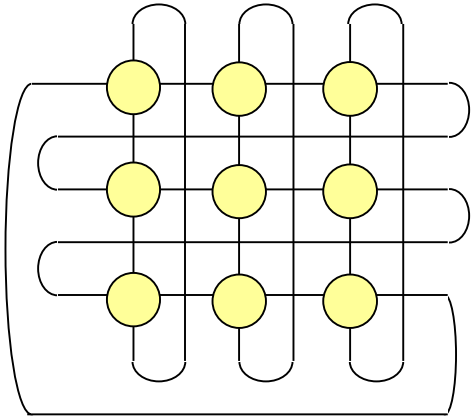
    Illiac网络的直径只有纯网格形网络直径的一半。
  - 等分宽度：2n

环网形

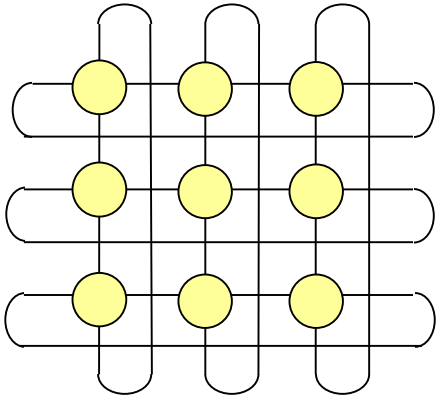
- 可看作是直径更短的另一种网格。
- 把2维网格形网络的每一行的两个端节点连接起来，把每一列的两个端节点也连接起来。
- 将环形和网格形组合在一起，并能向高维扩展。
- 一个n×n的环网形网
  - 节点度：4
  - 网络直径：$2\times\lfloor n/2\rfloor$
  - 等分宽度**b=2n**



（a）网格形



（b）Illiac网



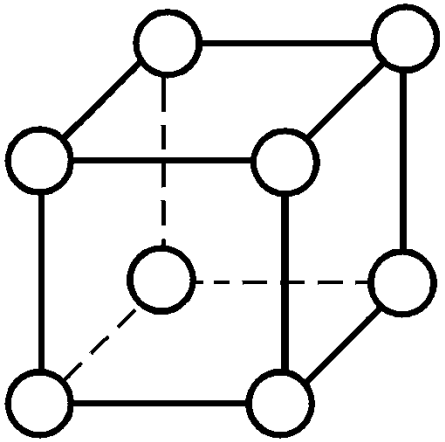
（c）环网形

7. 超立方体

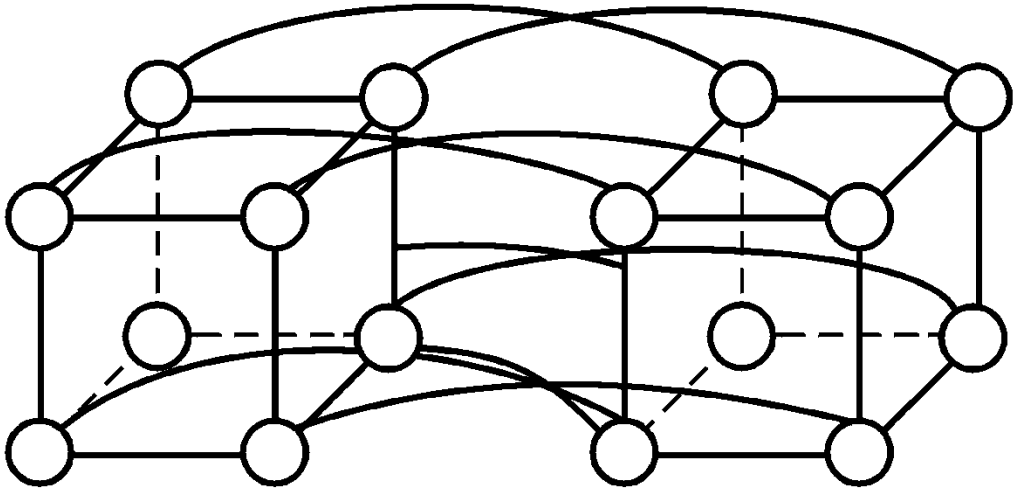
- 一种二元n-立方体结构
- 一般来说，一个二元n-立方体由$N=2^n$ 个节点组成，它们分布在n维上，每维有两个节点。

  例　8个节点的3维立方体

  4维立方体
- 为实现一个n-立方体，只要把两个（n－1）立方体中相对应的节点用链路连接起来即可。共需要2^{n-1}条链路。
- n-立方体中节点的度都是n，直径也是n，等分宽度为$b=N/2$ 。



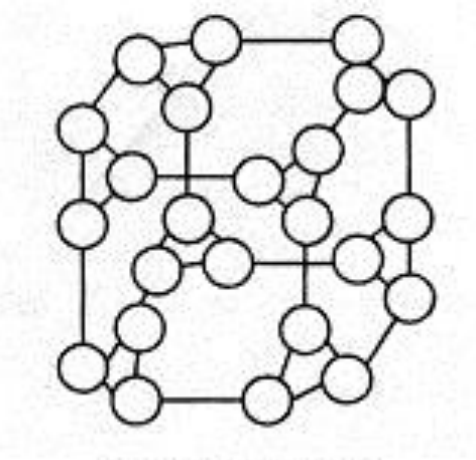
(a) 3–立方体



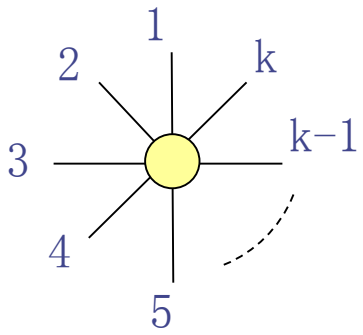
(b) 由 2 个 3– 立方体组成的 4– 立方体

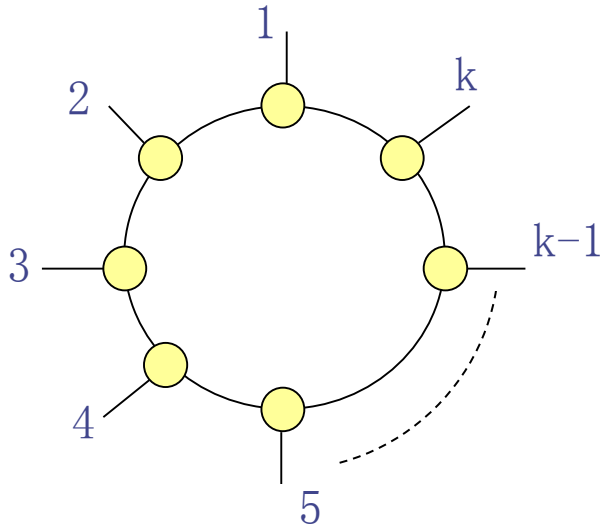
8. 带环立方体 （简称3-CCC）

- 把3-立方体的每个节点换成一个由3个节点构成的环而形成的。
- 带环k-立方体（简称k-CCC）
  - k-立方体的变形，它是通过用k个节点构成的环取代k-立方体中的每个节点而形成的。
  - 网络规模为$N=k\times 2^k$
  - 网络直径为$D=2k-1+\lfloor k/2 \rfloor$
    - 比k-立方体的直径大一倍
  - 等分宽度为$b=N/（2k）$



(a) 带环3-立方体



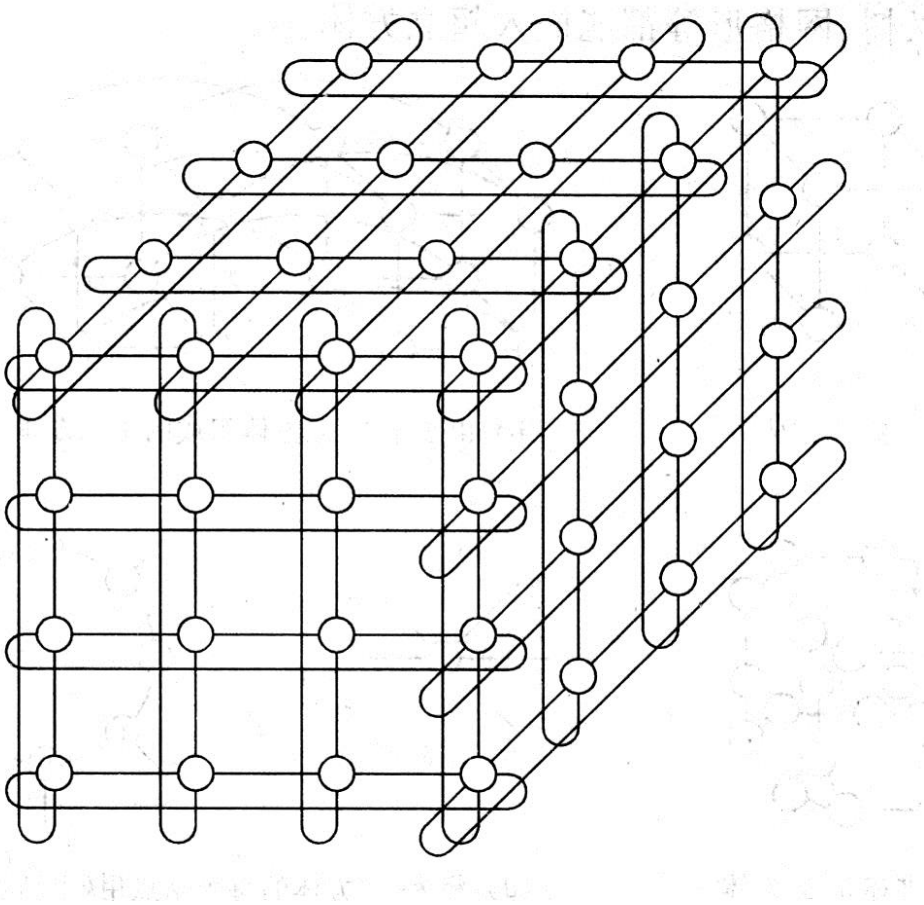


(b) 将k-立方体的每个节点用由k个节点的环来代替，组成带环k-立方体组成

9. k元n-立方体网络

- 环形、网格、环网形、二元n-立方体（超立方体）和Omega网络都是k元n-立方体网络系列的拓扑同构体。
- 在k元n-立方体网络中，参数n是立方体的维数，k是基数，即每一维上的节点个数。

  $N=k^n$，（ $k=\sqrt[n]{N}$ ，$n=\log_k N$）
- k元n-立方体的节点可以用基数为k的n位地址$A=a_1a_2\ldots a_n$来表示。
  - 其中a_i表示该节点在第i维上的位置
- 通常把低维k元n-立方体称为环网，而把高维k元n-立方体称为超立方体。



4元3-立方体网络

静态互连网络特征一览表

网络类型	节点度d	网络直径D	链路数l	等分宽度B	对称性	网络规格说明
线线阵列	2	N-1	N-1	1	非	N个节点
环形	2	[N/2]	N	2	是	N个节点
全连接	N-1	1	N(N-1)/2	$(N/2)^2$	是	N个节点
二叉树	3	2(h-1)	N-1	1	非	树高h=[$\log_2 N$]
星形	N-1	2	N-1	[N/2]	非	N个节点
2D网格	4	2(r-1)	2N-2r	r	非	r×r网格，$r=\sqrt{N}$
Illiac网	4	r-1	2N	2r	非	与 $r=\sqrt{N}$ 的带弦环等效
2D环网	4	2[r/2]	2N	2r	是	r×r环网，$r=\sqrt{N}$
超立方体	n	n	nN/2	N/2	是	N个节点，n=[$\log_2 N$]（维数）
CCC	3	2k-1+[k/2]	3N/2	N/(2k)	是	N=k×2^k节点 环长k≥3
k元n-立方体	2n	n[k/2]	nN	$2k^{n-1}$	是	N=k^n个节点

例7.2 已知有16台个处理器用Illiac网络互连，写出Illiac网络的互连函数，给出表示任何一个处理器PU_i（0≤i≤15）与其他处理器直接互连的一般表达式。

解：Illiac网络连接的节点数N=16，组成4×4的阵列。每一列的4个处理器互连为一个双向环，第1列～第4列的双向环可分别用循环互连函数表示为：

（0 4 8 12）　　（12 8 4 0）

（1 5 9 13）　　（13 9 5 1）

（2 6 10 14）　　（14 10 6 2）

（3 7 11 15）　　（15 11 7 3）

其中，传送方向为顺时针的4个单向环的循环互连函数可表示为：

$$PM2_{+2}（X）=（X+2^2）\bmod N=（X+4）\bmod 16$$

传送方向为逆时针的4个单向环的循环互连函数可表示为：

$PM2_{-2}(X) = (X - 2^2) \bmod N = (X - 4) \bmod 16$

16个处理器由Illiac网络的水平螺线互连为一个双向环，用循环互连函数表示为：

（0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15）

（15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0）

其中，传送方向为顺时针的单向环的循环互连函数可表示为：

$PM2_{+0}(X) = (X + 2^0) \bmod N = (X + 1) \bmod 16$

传送方向为逆时针的单向环的循环互连函数可表示为：

$PM2_{-0}(X) = (X - 2^0) \bmod N = (X - 1) \bmod 16$

所以，N=16的Illiac网络的互连函数有4个：

$PM2_{\pm 0}(X)$和$PM2_{\pm 2}(X)$

由互连函数可得任何一个处理器i直接与下述4个处理器双向互连：

$i \pm 1 \bmod 16$

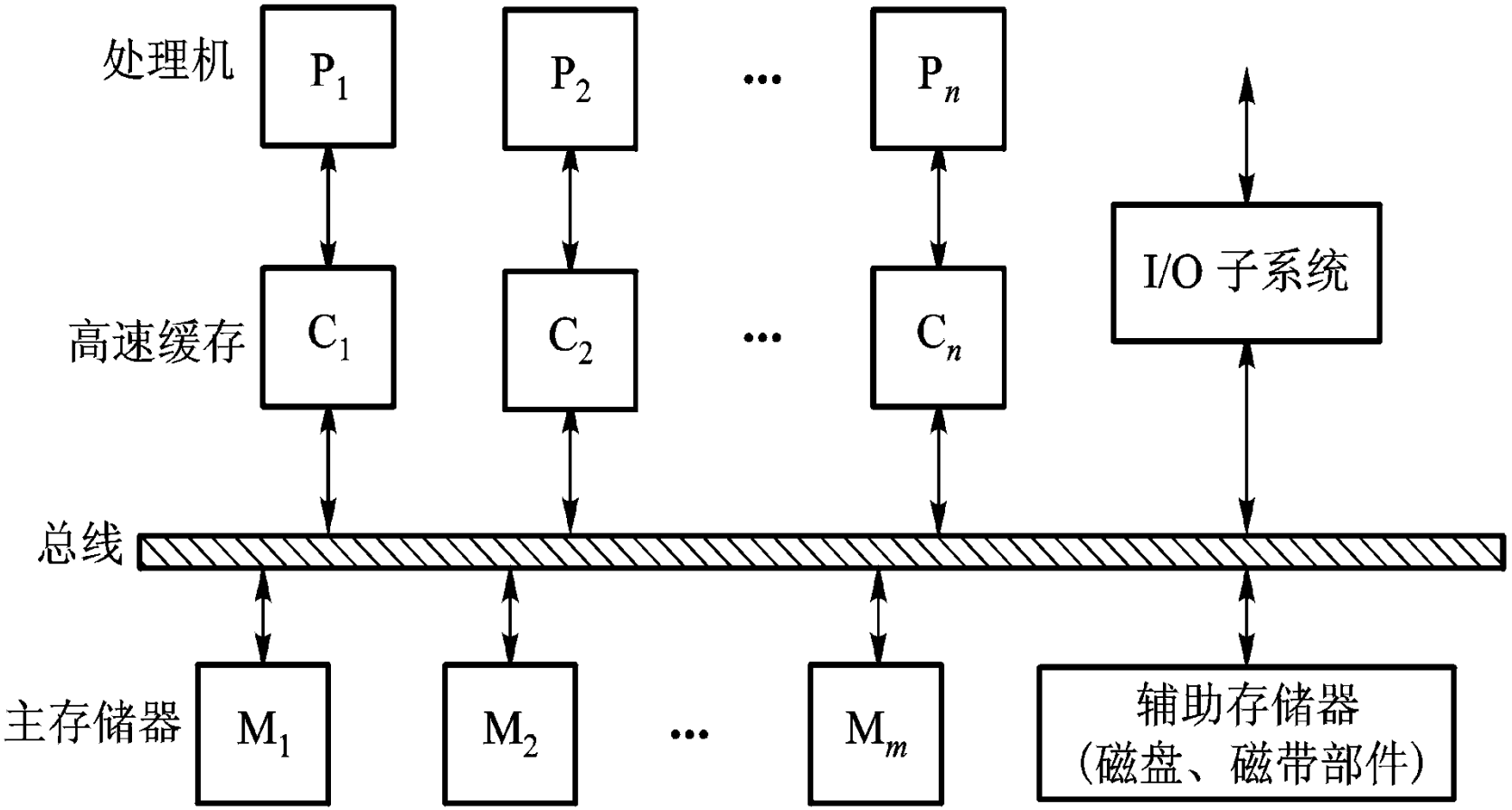
$i \pm 4 \bmod 16$

7.4 动态互连网络

7.4.1 总线网络

1. 由一组导线和插座构成，经常被用来实现计算机系统中处理机模块、存储模块和外围设备等之间的互连。
   - 每一次总线只能用于一个源（主部件）到一个或多个目的（从部件）之间的数据传送。
   - 多个功能模块之间的争用总线或时分总线
   - 特点
     - 结构简单、实现成本低、带宽较窄

2. 一种由总线连接的多处理机系统



- 系统总线在处理机、I/O子系统、主存储器以及辅助存储设备（磁盘、磁带机等）之间提供了一条公用通路。
- 系统总线通常设置在印刷电路板底板上。处理器板、存储器板和设备接口板都通过插座或电缆插入底板。

3. 解决总线带宽较窄问题：采用多总线或多层次的总线

- **多总线是设置多条总线**

  有两种做法：

  - 为不同的功能设置专门的总线
  - 重复设置相同功能的总线

- **多层次的总线是按层次的架构设置速度不同的总线，使得不同速度的模块有比较适合的总线连接。**

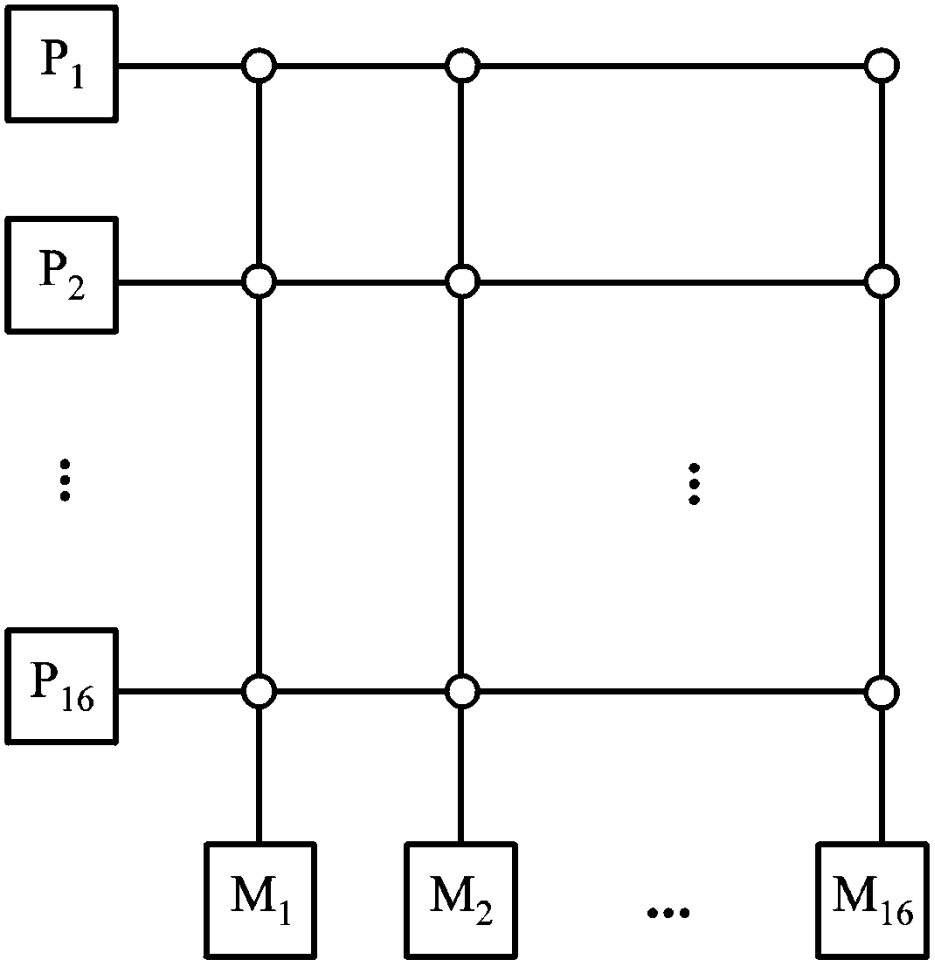
7.4.2 交叉开关网络

1. 单级开关网络

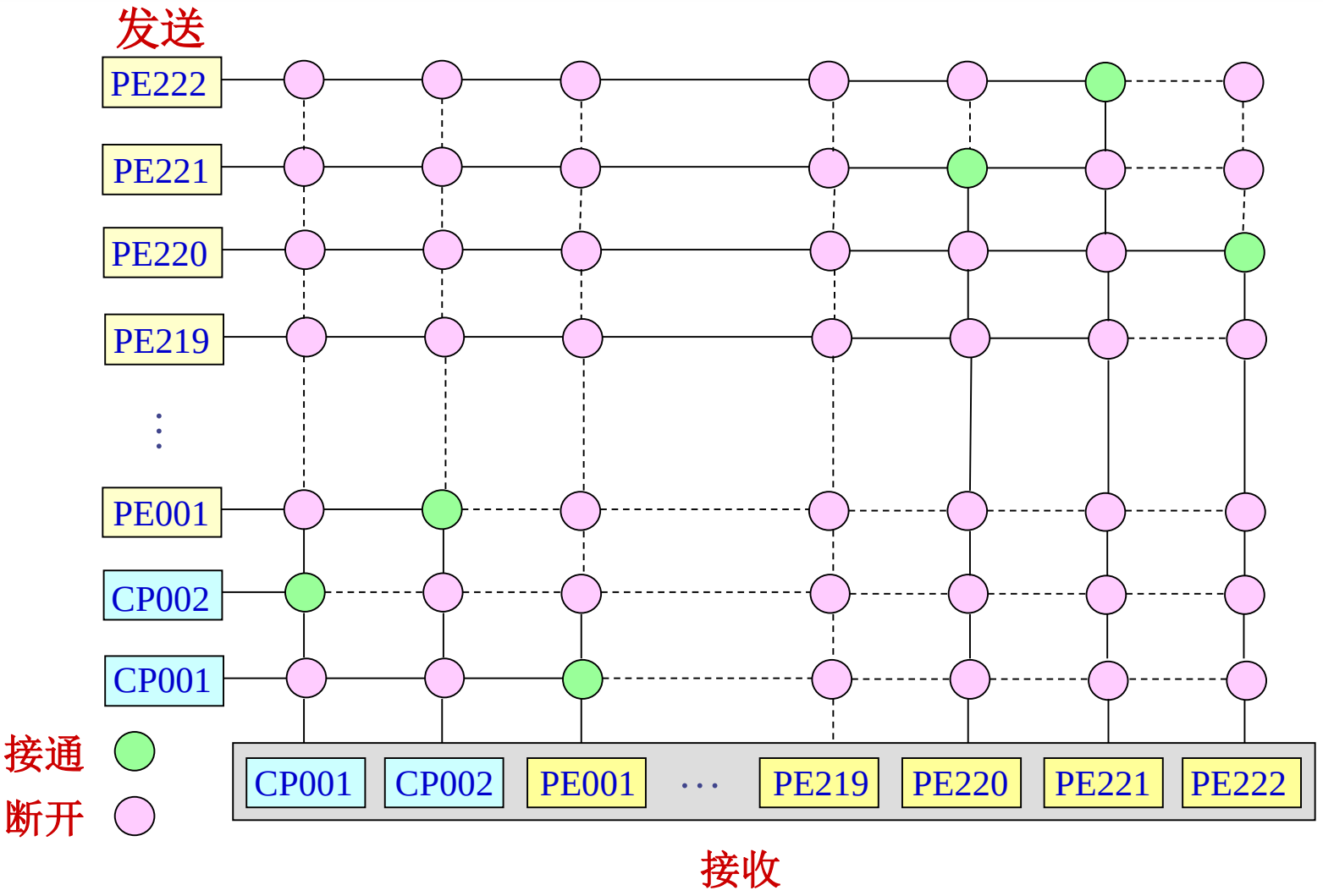
- 交叉点开关能在对偶（源、目的）之间形成动态连接，同时实现多个对偶之间的无阻塞连接。
- 带宽和互连特性最好。
- 一个n×n的交叉开关网络，可以无阻塞地实现n!种置换。
- 对一个n×n的交叉开关网络来说，需要n^2套交叉点开关以及大量的连线。
  - 当n很大时，交叉开关网络所需要的硬件数量非常巨大。

2. C.mmp多处理机的互连结构

- 用16×16的交叉开关网络把16台PDP-11处理机与16个存储模块连在一起
- 最多可同时实现16台处理机对16个不同存储模块的并行访问
  - 每个存储模块一次只能满足一台处理机的请求
  - 当多个请求要同时访问同一存储模块时，交叉开关就必须分解所发生的冲突，每一列只能接通一个交叉点开关。
  - 为了支持并行（或交叉）存储器访问，可以在同一行中接通几个交叉点开关。


P1
P2
P16
M1
M2
...
M16

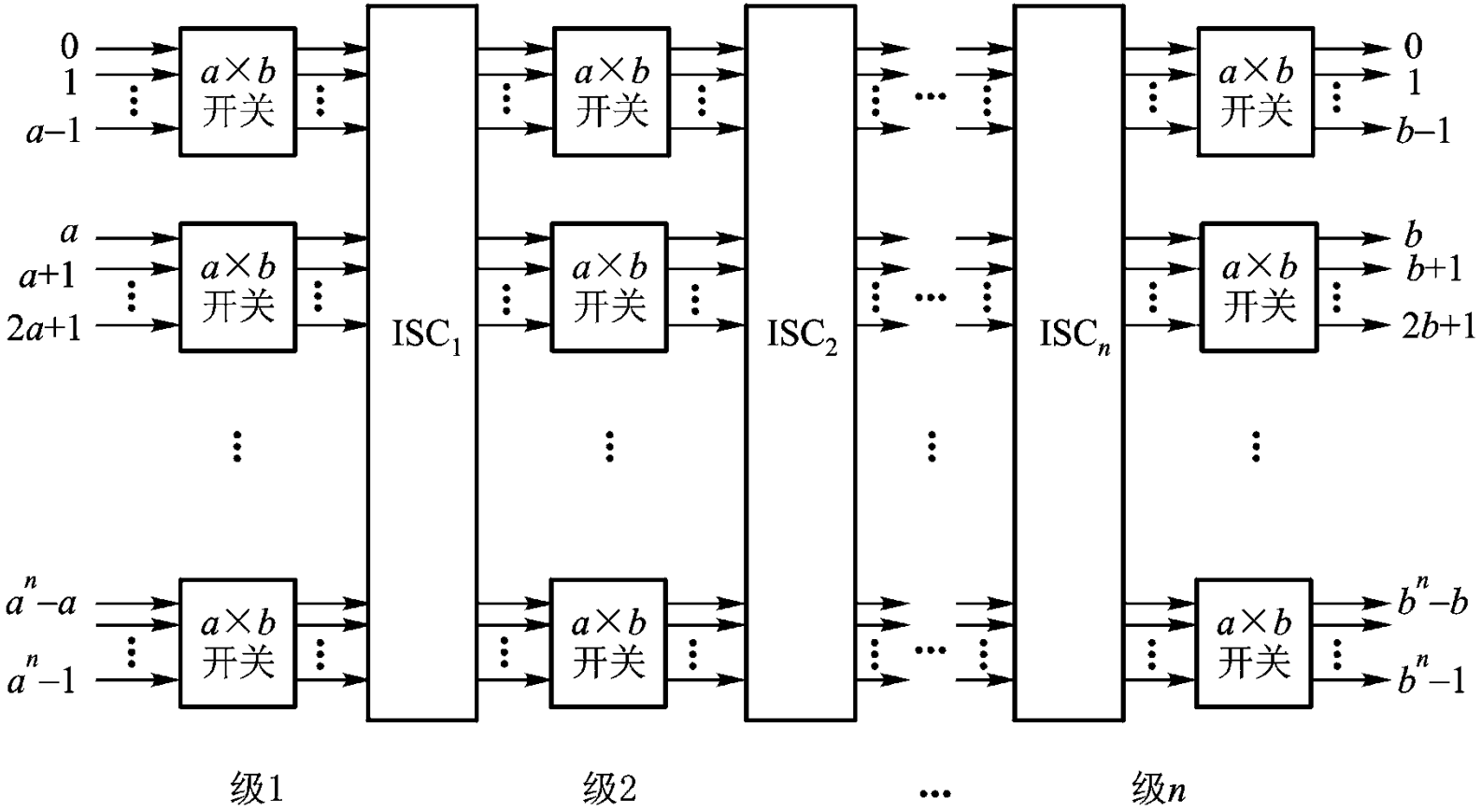
3. Fujitsu公司制造的向量并行处理机VPP500所采用的大型交换开关网络（224×224）
   - PE：带存储器的处理机
   - CP：控制处理机
   - 每一行和每一列只能接通一个交叉点开关


发送
PE222
PE221
PE220
PE219
PE001
CP002
CP001
接通
断开
CP001
CP002
PE001
PE219
PE220
PE221
PE222
接收

7.4.3 多级互连网络

1. 多级互连网络的构成

- MIMD和SIMD计算机都采用多级互连网络MIN（Multistage Interconnection Network）
- 一种通用的多级互连网络
  - 由a×b开关模块和级间连接构成的通用多级互连网络结构
  - 每一级都用了多个a×b开关
    - a个输入和b个输出
    - 在理论上，a和b不一定相等，然而实际上a和b经常选为2的整数幂，即$a=b=2^k$，$k\geqslant 1$。
  - 相邻各级开关之间都有固定的级间连接

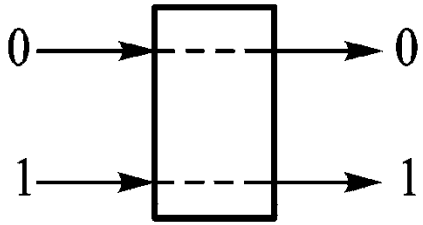

0
1
a–1
a×b 开关
a
a+1
2a+1
a^n–a
a^n–1
ISC1
ISC2
ISCn
0
1
b–1
b
b+1
2b+1
b^n–b
b^n–1
级1
级2
…
级n

➢ 几种常用的开关模块

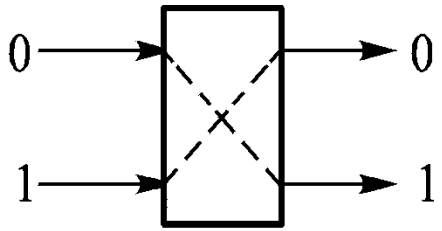
模块大小	合法状态	置换连接
2×2	4	2
4×4	256	24
8×8	16 777 216	40 320
n×n	n^n	n!

- 最简单的开关模块：2×2开关

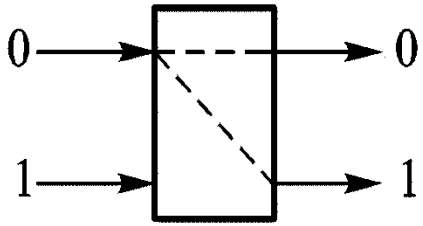
  2×2开关的4种连接方式



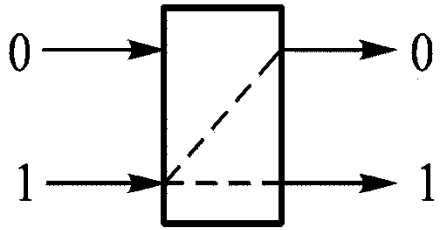
(a) 直送



(b)交叉



(c)上播



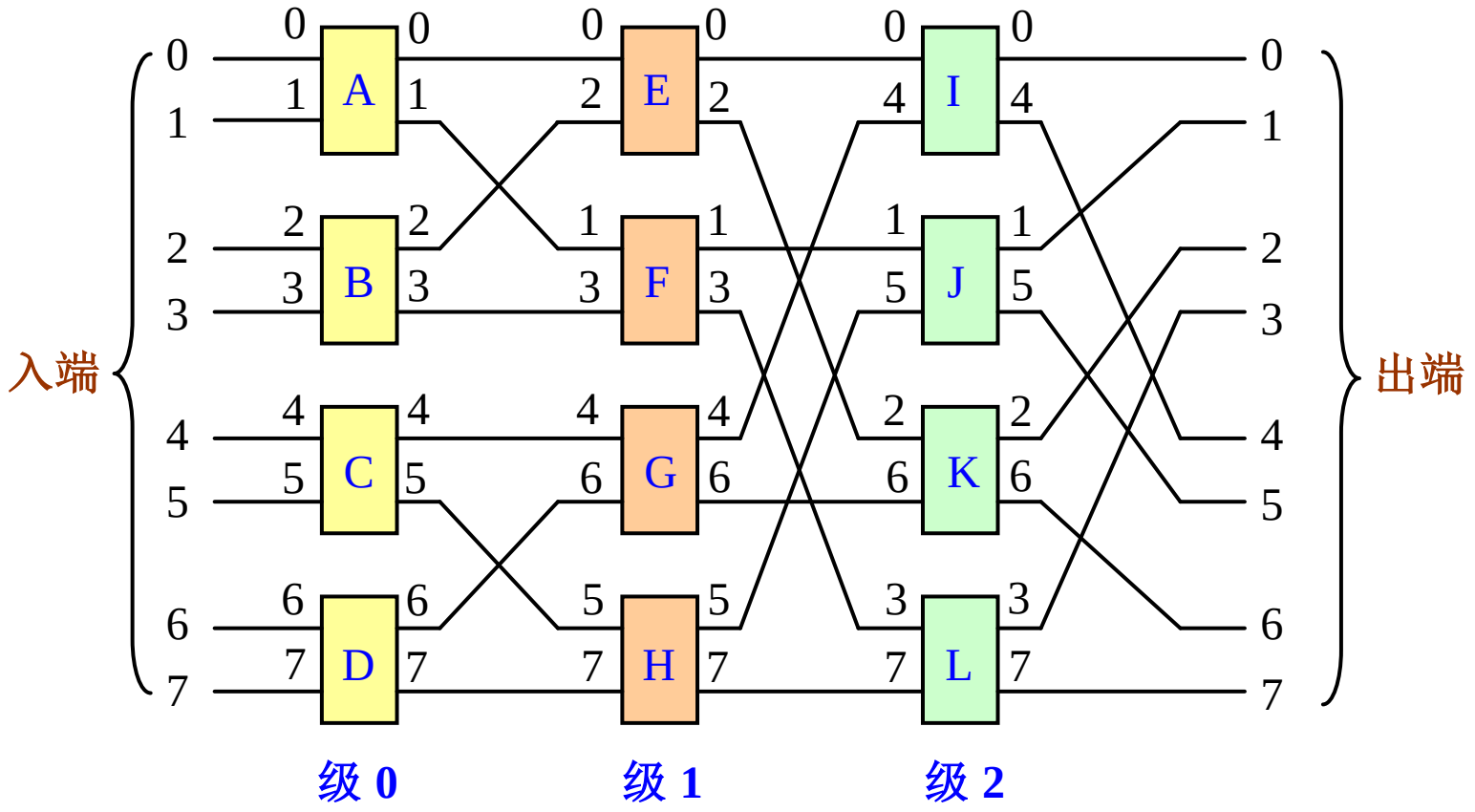
(d)下播

- 各种多级互连网络的区别在于所用开关模块、控制方式和级间互连模式的不同。
  - 控制方式：对各个开关模块进行控制的方式。
    - 级控制：每一级的所有开关只用一个控制信号控制，只能同时处于同一种状态。
    - 单元控制：每一个开关都有一个独立的控制信号，可各自处于不同的状态。
    - 部分级控制：第i级的所有开关分别用i＋1个信号控制，0≤i≤n－1，n为级数。
  - 常用的级间互连模式：

均匀洗牌、蝶式、多路洗牌、纵横交叉、立方体连接等

2. 多级立方体网络

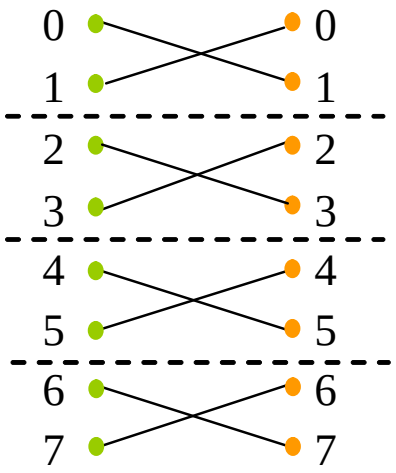
- 多级立方体网络包括STARAN网络和间接二进制n方体网络等。
  - 两者仅在控制方式上不同，在其他方面都是一样的。
  - 都采用二功能（直送和交换）的2×2开关。
  - 当第i级（0≤i≤n-1）交换开关处于交换状态时，实现的是$Cube_i$互连函数。
- 一个N输入的多级立方体网络有$\log_2 N$级，每级用N/2个2×2开关模块，共需要$\log_2 N \times N/2$个开关。
- 一个8个入端的多级立方体网络



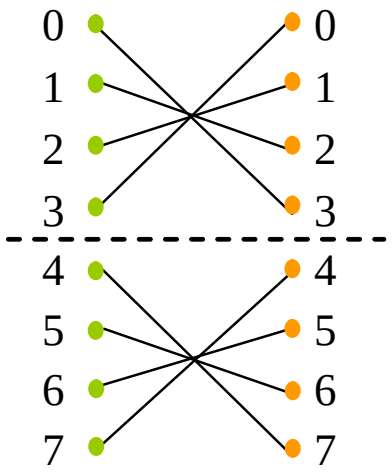
多级立方体网络

- **STARAN网络采用级控制和部分级控制。**
  - 采用级控制时，所实现的是交换功能；
  - 采用部分级控制时，则能实现移数功能。
- **间接二进制n方体网络则采用单元控制。**
  - 具有更大的灵活性。
- **交换**
  - 将有序的一组元素头尾对称地进行交换。

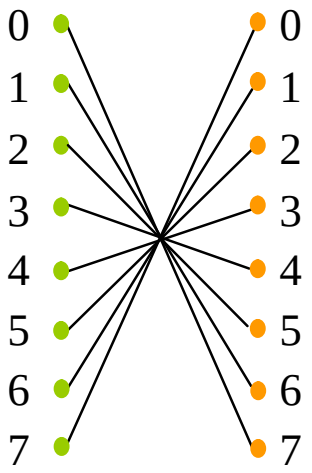
例如：对于由8个元素构成的组，各种基本交换的图形：



(a) 4 组 2 元交换



(b) 2 组 4 元交换



(c) 1 组 8 元交换

8个元素的基本交换图形

- 3级STARAN网络在各种级控制信号的情况下所实现的入出端连接以及所实现的交换函数和功能。

其中：

- $K_2k_1k_0$：控制信号，k_i（i=0，1，2）为第i级的级控制信号。
- 从表中可以看出

下面的4行中每一行所实现的功能可以从级控制信号为其反码的一行中所实现的功能加上1组8元变换来获得。

例如：级控制信号为110所实现的功能是其反码001所实现的4组2元交换再加上1组8元交换来获得。

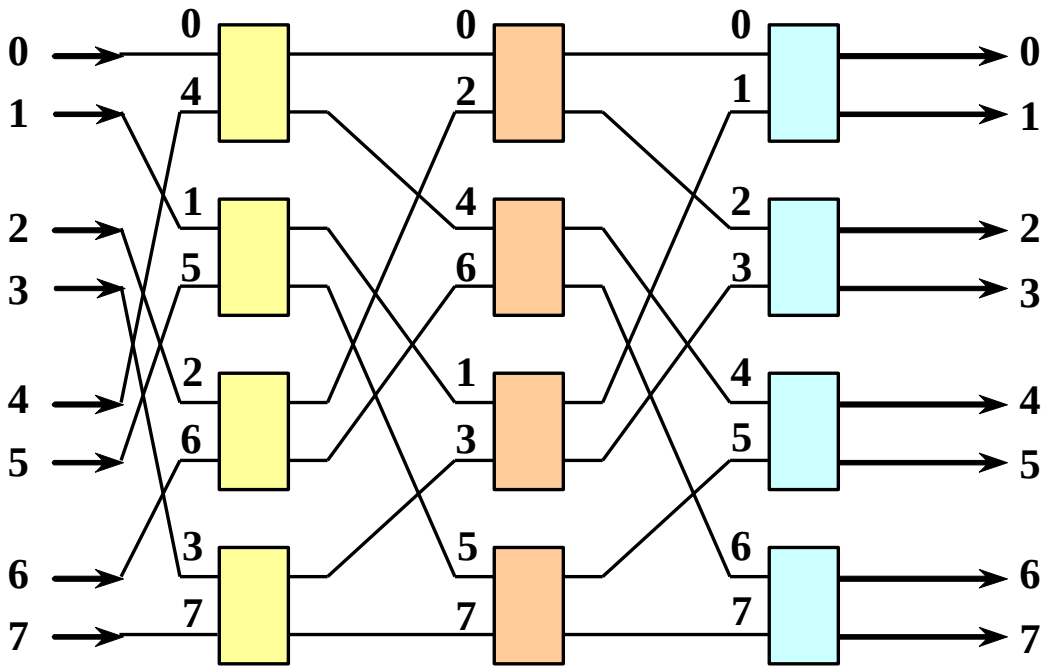
级控制信号 $k_2k_1k_0$	连接的输出端号序列（入端号序列：01234567）	实现的分组交换	实现的互连函数
000	0 1 2 3 4 5 6 7	恒等	I
001	1 0 3 2 5 4 7 6	4组2元交换	$Cube_0$
010	2 3 0 1 6 7 4 5	4组2元交换＋ 2组4元交换	$Cube_1$
011	3 2 1 0 7 6 5 4	2组4元交换	$Cube_0$＋$Cube_1$
100	4 5 6 7 0 1 2 3	2组4元交换＋ 1组8元交换	$Cube_2$
101	5 4 7 6 1 0 3 2	4组2元交换＋ 2组4元交换＋ 1组8元交换	$Cube_0$＋$Cube_2$
110	6 7 4 5 2 3 0 1	4组2元交换＋ 1组8元交换	$Cube_1$＋$Cube_2$
111	7 6 5 4 3 2 1 0	1组8元交换	$Cube_0$＋$Cube_1$＋ $Cube_2$

- 当STARAN网络用作移数网络时，采用部分级控制，控制信号的分组和控制结果。

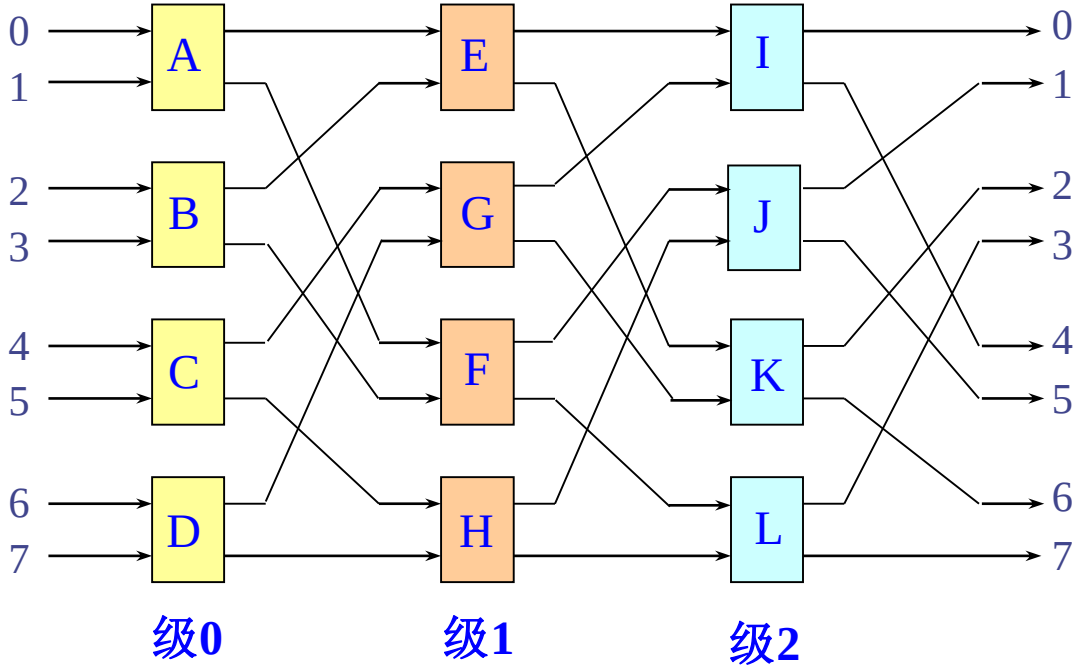
部分级控制信号						连接的输出端号序列（入端号序列：01234567）	所实现的移数功能
第0级	第1级		第2级				
A B C D	E G	F H	I	J	K L		
1	1	0	1	0	0	1 2 3 4 5 6 7 0	移1 mod 8
0	1	1	1	1	0	2 3 4 5 6 7 0 1	移2 mod 8
0	0	0	1	1	1	4 5 6 7 0 1 2 3	移4 mod 8
1	1	0	0	0	0	1 2 3 0 5 6 7 4	移1 mod 4
0	1	1	0	0	0	2 3 0 1 6 7 4 5	移2 mod 4
1	0	0	0	0	0	1 0 3 2 5 4 7 6	移1 mod 2
0	0	0	0	0	0	0 1 2 3 4 5 6 7	不移 全等

3. Omega网络

- 一个8×8的Omega网络
  - 每级由4个4功能的2×2开关构成
  - 级间互连采用均匀洗牌连接方式



- **一个N输入的Omega网络**
  - 有$\log_2 N$级，每级用$N/2$个2×2开关模块，共需要$N\log_2 N/2$个开关。
  - 每个开关模块均采用单元控制方式。
  - 不同的开关状态组合可实现各种置换、广播或从输入到输出的其他连接。
- **N=8的多级立方体互连网络的另一种画法**


0
1
2
3
4
5
6
7
A
B
C
D
E
G
F
H
I
J
K
L
0
1
2
3
4
5
6
7
级0
级1
级2

7.4.4 动态互连网络的比较

网络特性	总线系统	多级网络	交叉开关
单位数据传送的最小时延	恒定	$O(\log_k n)$	恒定
每台处理机的带宽	O(w/n)至O(w)	O(w)至O(nw)	O(w)至O(nw)
连线复杂性	O(w)	$O(nw\log_k n)$	$O(n^2w)$
开关复杂性	O(n)	$O(n\log_k n)$	$O(n^2)$
连接特性和寻径性能	一次只能一对一	只要网络不阻塞，可实现某些置换和广播	全置换，一次一个
典型计算机	Symmetry S1，Encore Multimax	BBNTC-2000 IBM RP3	Cray Y-MP/816 Fujitsu VPP 500
说明	总线上假定有n台处理机；总线宽度为w位	n×n MIN采用k×k开关，其线宽为w位	假定n×n交叉开关的线宽为w位

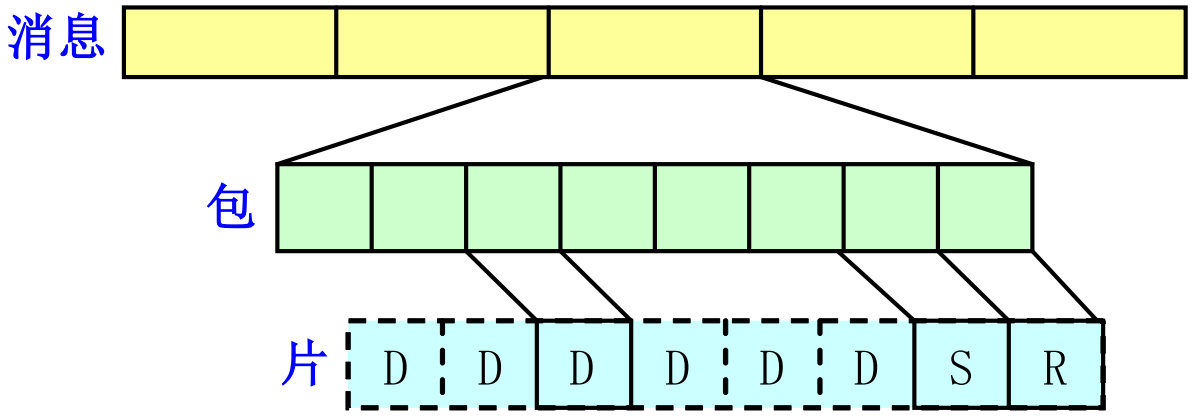
7.5 消息传递机制

当源节点和目的节点之间没有直接的连接时，消息需要经过中间的节点进行传递。寻径就是用来实现这种传递的通信方法和算法。有的称之为路由。

7.5.1 消息寻径方案

1. 消息的格式

- 消息：节点之间进行通信的逻辑单位
  - 由若干个“包”组成
  - 包的长度是固定的，一条消息中所包含的包的个数是可变的，消息的长度是不定长的。



R：寻径信息　　S：顺序号　　D：数据片

消息、包和片的格式

- **包：包含寻径所需目的地址的基本单位。**
  - 一条消息中的各个包都依次被分配一个序号
    以便这些包到达目的节点后能重新组装出消息。
  - 包可以进一步分成一些更小的固定长度的单位，称为“片”。
  - 寻径信息和包序列号形成头片，其余的是数据片。
  - 包的长度主要是由寻径方案和网络的具体实现所决定的
    - 典型的长度是64～512位不等
  - 片的长度经常是受网络大小的影响

2. 四种寻径方式

- **线路交换：**在线路交换方式下，在传递一个信息之前，需要先建立一条从源节点到目的节点的物理通路，然后再传递信息。
  - 传输时延T

$$T = \frac{L + L_t \times (D+1)}{B}$$

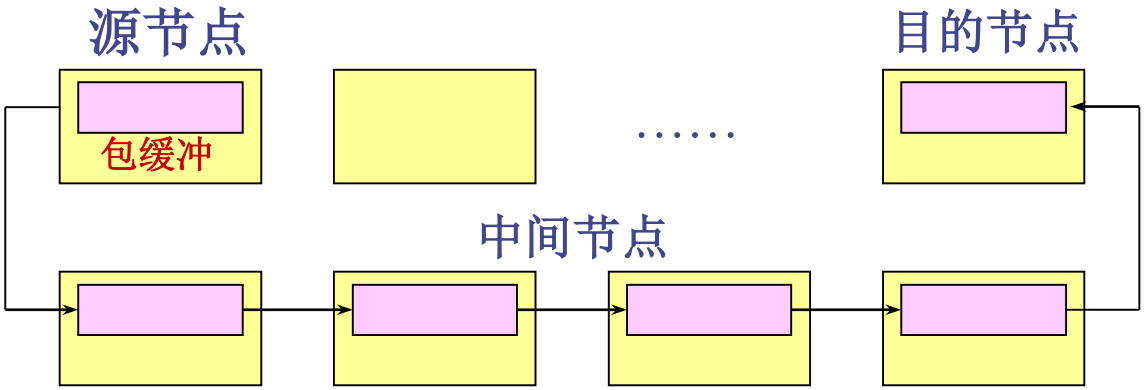
其中：

- L：信息包的长度（位数）
- L_t：建立路径所需的小信息包的长度
- D：经过的中间节点个数
- B：带宽

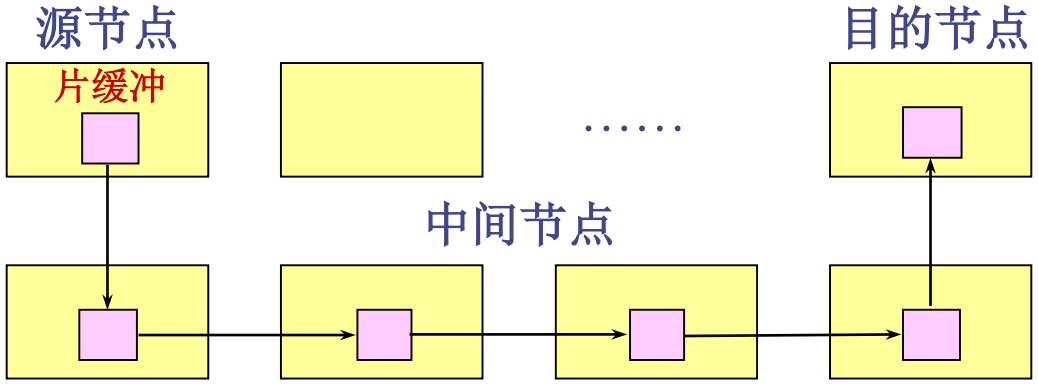
- 优点：传输带宽较大，平均传输时延较小，而且使用的缓冲区小。
  - 适合于具有动态和突发性的大规模并行处理数据的传送。
- 缺点：需要频繁地建立源节点到目的节点的物理通路，时间开销会很大。

➢ **存储转发：最简单的分组交换方式。**

- 包是信息传递的基本单位。包从源节点经过一系列中间节点到达目的节点。
- 要求：所经过的每个中间节点都要设置一个包缓冲器，用于保存所传递的包。当一个包到达某个中间节点时，该节点先把这个包全部存储起来，然后在出口链路可用、而且下一个节点的包缓冲器也可用的情况下，传递给下一个节点。



（a）存储转发



（b）虫蚀方式

- 网络的时延与源和目的地之间的距离（跳数）成正比

$$T_{SF} = \frac{L}{B}(D+1)$$

- 缺点
  - 包缓冲区大，不利于VLSI实现；
  - 网络时延大，与节点距离成正比。

➢ **虚拟直通：对存储转发方式的一种改进，减少了网络时延。**

- 基本思想：没有必要等到信息包全部放入缓冲器后再作路由选择，只要接收到用作寻径的包头，就可作出判断。

- 如果节点的输出链路空闲，信息包可以不必存储在该节点的缓冲器中，而是立即传送到下一个节点。
- 如果整条链路都空闲，包就可以立即直达目的节点。
- 在输出链路不空闲时，要用缓冲器进行存储。

- 通信时延

$$T = \frac{L + L_h \times (D+1)}{B} \approx \frac{L}{B}$$

- L_h：信息包寻径头部的长度
- 一般来说，$L >> L_h \times (D+1)$，所以$T \approx L/B$。

- 当出现寻径阻塞时，虚拟直通方式需要将整个信息包全部存储在寻径节点中，要求每个节点都有足够大的缓冲区。

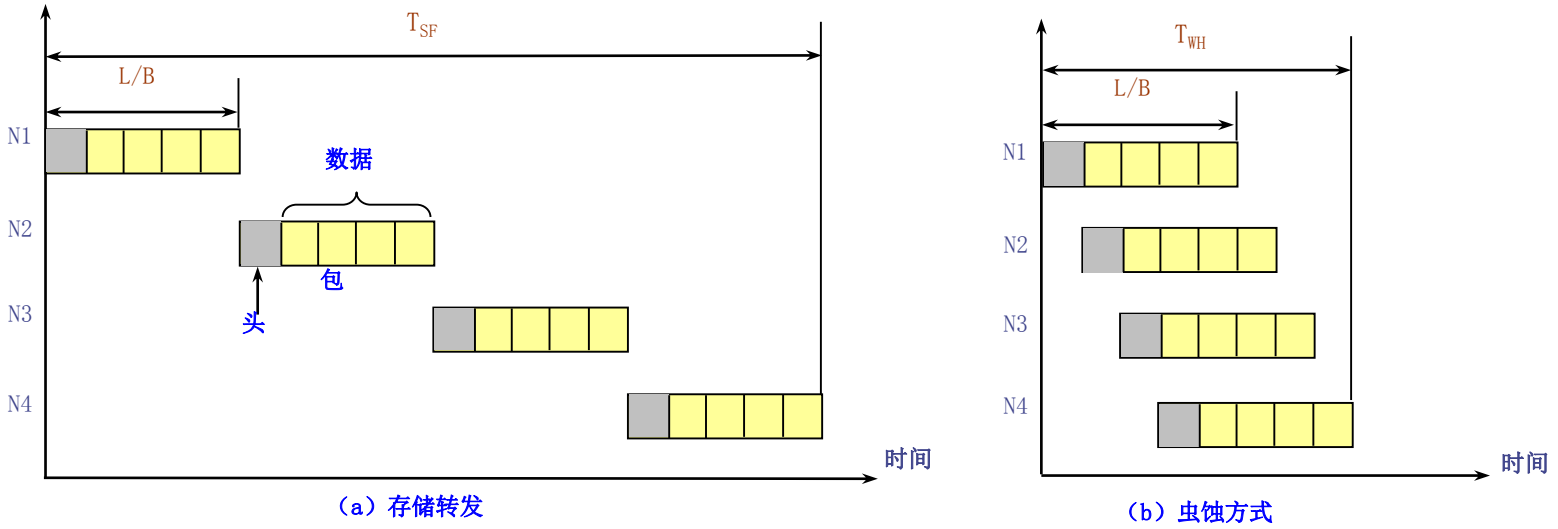
（不利于VLSI的实现 ）

- **虫蚀方式：把信息包“切割”成更小的单位——“片”，而且使信息包中各片的传送按流水方式进行。**
  - 可以减少节点中缓冲器的容量，缩短传送延迟时间。
  - 在新型的多计算机系统中得到了广泛的应用。
  - 处理的最小信息单位是“片”。当一个节点把头片送到下一个节点后，那么接下来就可以把后面的各个片也依次送出。

- 一个节点一旦开始传送一个包中的头片后，这个节点就必须等待这个包的所有片都送出去后，才能传送其他包。不同包的片不能混合在一起传送。
- 与虚拟直通的不同之处
  - 当输出通路忙时，节点是把一个片存储到缓冲器中。
  - 由于片的大小比包小很多，所以能有效地减少缓冲器的容量，使得它易于用VLSI实现。
- 通信时延

$$T_{WH} = T_f \times D + \frac{L}{B} = \frac{L + L_f \times D}{B} \approx \frac{L}{B}$$

  - L_f：“片”的长度
  - T_f：片经过一个节点所需时间，$L >> L_f \times D$ 。

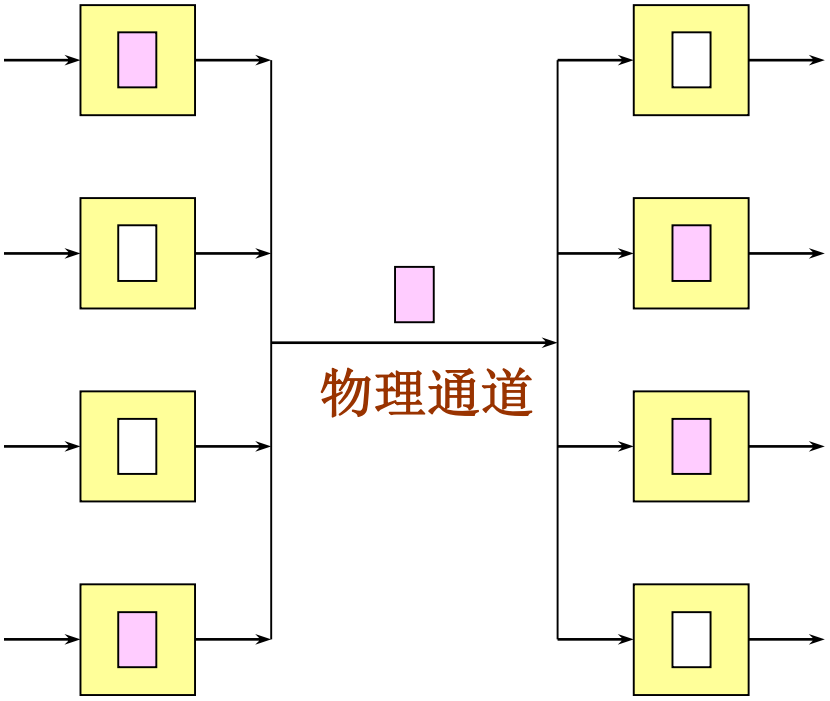


存储转发与虫蚀方式的时间比较

- 优点
  - 每个节点的缓冲器较小，易于VLSI实现；
  - 有较小的网络传输延迟；
  - 通道共享性好，利用率高；
  - 易于实现选播和广播通信模式。
- 缺点
  - 当消息的一片被阻塞时，整个消息的所有片都将被阻塞在所在节点，占用了节点资源。

7.5.2 死锁与虚拟直通

1. **虚拟通道：两个节点间的逻辑链接，它由源节点的片缓冲区、节点间的物理通道以及接收节点的片缓冲区组成。**
   - **4条虚拟通道共享一条物理通道**
     - 源节点和接收节点各有4个片缓冲区。
     - 当物理通道分配给某对缓冲区时，这一对的源缓冲区和接收缓冲区就形成了一条虚拟通道。
     - 物理通道是由所有的虚拟通道分时地共享。
   - **虚拟通道也可以用双向通道实现。把两条单向通道组合在一起可以构成一条双向通道。**
     - 增加了利用率，使通道的带宽加倍。



4条虚拟通道以片传递为基础分时地共享一条物理通道

2. 避免死锁

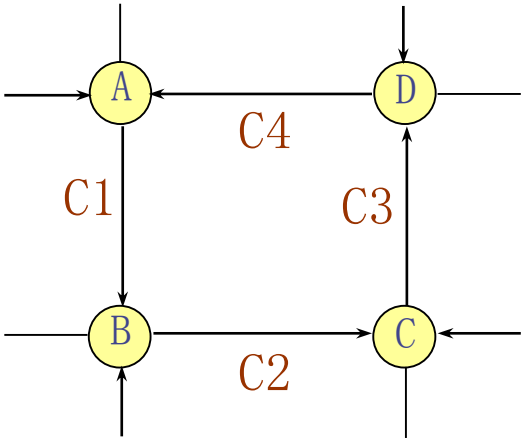
- 缓冲区或通道上的循环等待会引起死锁。

例如：图（a）：出现循环的通道相关而产生死锁

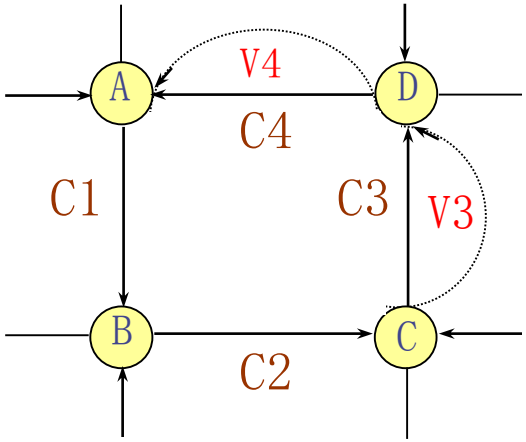
图（b）： 利用虚拟通道方法可以避免这个死锁，可以增加两条虚拟通道V3和V4。

图（c）：避免了死锁

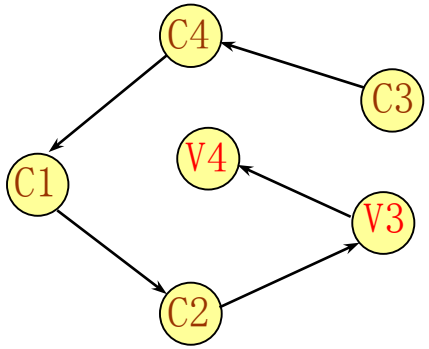
增加虚拟通道可能会使每个请求可用的有效通道带宽降低。为此，当实现数目很大的虚拟通道时需要用高速的多路选择开关。



（a）通道死锁



（b）增加虚拟通道



（c）利用虚拟通道后的通道相关图

利用虚拟通道减少死锁

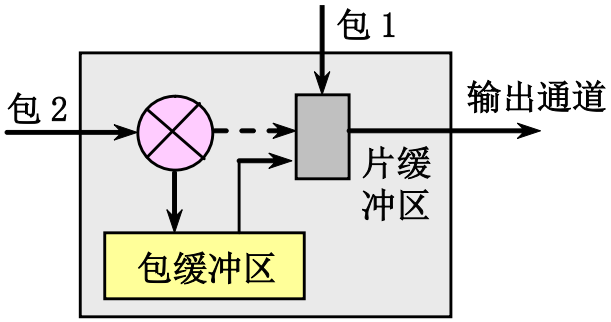
7.5.3 流控制策略

1. 包冲突的解决

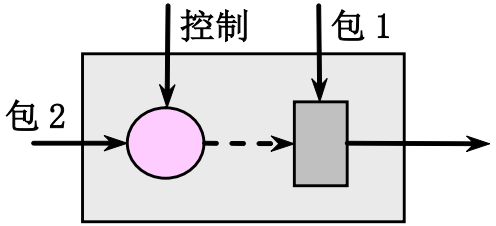
- 为了通过通道在两个相邻节点之间传送一个片，要同时具备3个条件：
  - 源缓冲区已存有该片；
  - 通道已分配好；
  - 接收缓冲区准备接收该片。
- 当两个包到达同一个节点时，它们可能都在请求同一个接收缓冲器或者同一个输出通道，这时必须对两个问题进行仲裁。

- 把通道分配给哪个包？
- 如何处理被通道拒绝的包？

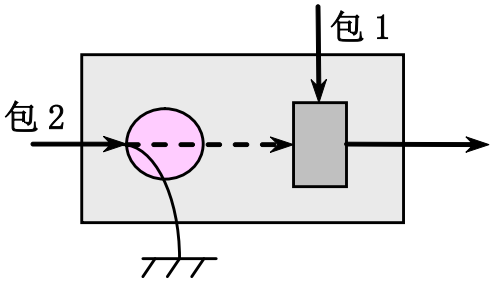
➢ 4种解决方案



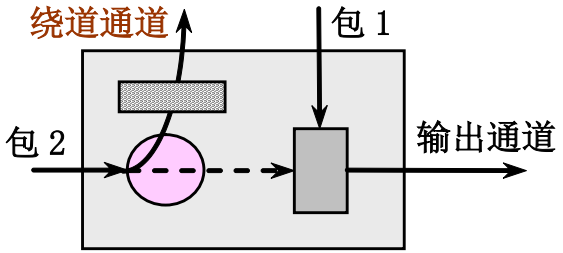
（a）用缓冲实现虚拟通道



（b）阻塞流控制



（c）扬弃并重发



（d）阻塞后绕道

- 把第二个包暂存在缓冲区
  - 优点:不会浪费已经分配了的资源，但它要求节点中有一个足够大的缓冲器来存放整个信息包。
- 阻塞第二个包
- 丢弃第二个包
  - 有可能会造成严重的资源浪费，而且要求重新进行被丢弃包的传输与确认。
- 绕道
  - 在包寻径方面提供了更多的灵活性，但为了到达目的节点，可能要花费超过实际需要的通道资源，造成浪费。

2. 确定性寻径和自适应寻径

- **确定性寻径**：通信路径完全由源节点地址和目的地址来决定，也就是说，寻径路径是预先唯一地确定好了的，而与网络的状况无关。
- **自适应寻径**：通信的通路每一次都要根据资源或者网络的情况来选择。
  - 可以避开拥挤的或者有故障的节点，使网络的利用率得到改进。
- 两种确定性寻径算法
  - 都是建立在维序概念之上的
  - 对于一个多维网来说，维序寻径要求对后继通道的选择是按照各维的顺序来进行的。

- 对于二维的网格网络来说，这种寻径方法被称为X-Y寻径。
  - 先沿X维方向进行寻径，然后再沿Y维方向寻找路径。
- 对于超立方体来说，这种寻径方法被称为E-cube寻径。

➢ 二维网格网络的X-Y寻径

任意一个源节点：s=（x_1，y_1）

任意一个目的节点：d=（x_2，y_2）

- 从s出发，先沿X轴方向前进，直到找到d 所在的列x_2；
- 然后再沿Y轴方向前进，直到找到目标节点（x_2，y_2）。

例7.3 对于图所示的二维网格，确定以下4组“源节点-目的结点”所需要的路经。

（2，1）到（7，6）

（0，7）到（4，5）

（6，4）到（2，0）

（5，3）到（1，5）

解

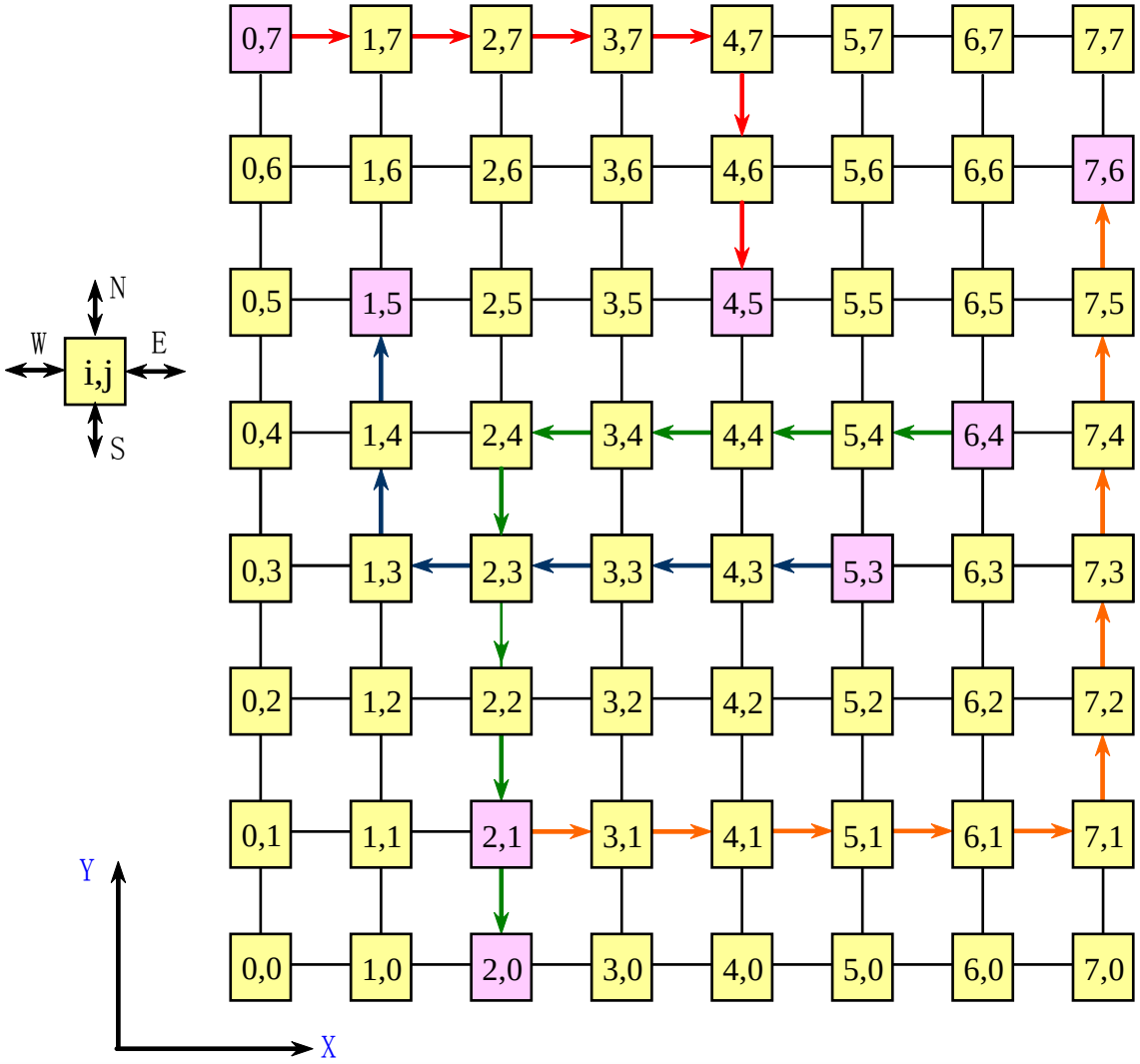
所需要的路径如图所示。其中：

（2，1）到（7，6）需要用到的是一条东-北路径；

（0，7）到（4，5）需要用到的是一条东-南路径；

（6，4）到（2，0）需要用到的是一条西-南路径；

（5，3）到（1，5）需要用到的是一条西-北路径。


0,7
1,7
2,7
3,7
4,7
5,7
6,7
7,7
0,6
1,6
2,6
3,6
4,6
5,6
6,6
7,6
0,5
1,5
2,5
3,5
4,5
5,5
6,5
7,5
0,4
1,4
2,4
3,4
4,4
5,4
6,4
7,4
0,3
1,3
2,3
3,3
4,3
5,3
6,3
7,3
0,2
1,2
2,2
3,2
4,2
5,2
6,2
7,2
0,1
1,1
2,1
3,1
4,1
5,1
6,1
7,1
0,0
1,0
2,0
3,0
4,0
5,0
6,0
7,0
N
W
E
i,j
S
Y
X

考虑一个由$N=2^n$个节点构成的n方体，每个节点的编号是形为$b=b_{n-1}b_{n-2}\cdots b_1b_0$的二进制编码。

设：源节点$s=s_{n-1}s_{n-2}\cdots s_1s_0$

目的节点$d=d_{n-1}d_{n-2}\cdots d_1d_0$

现在要确定一条从s到d的步数最少的路径。

将这个n方体的各维表示成$i=1，2，\ldots，n$，其中第i维对应于节点地址中的第$i-1$位。

设$v=v_{n-1}v_{n-2}\cdots v_1v_0$是路径中的任一节点。路径可以根据以下算法唯一地确定：

① 计算方向位 $r_i = s_{i-1} \oplus d_{i-1}$，其中i＝1，2，…，n。

令v=s，i=1，反复执行以下步骤：

② 如果r_i＝1，则从当前节点v寻径到下一节点；否则，就跳过这一步。

③ i←i+1。如果i≤n，则转第②步，否则退出。

例7.4 假设有一个N=16个节点的4方体，每个节点的二进制编码如图所示。请寻找一条从节点0110到1101的距离最短的路径。

解　s=0110，d=1101

第1步：计算方向位（$r_4r_3r_2r_1$）= $0110 \oplus 1101 = 1011$

令 v=s=0110，i=1

第2步：$r_1=1$

所以从v=0110寻径到 $v \oplus 2^0 = 0110 \oplus 0001 = 0111$

i=i+1=2

第3步：$r_2=1$

所以从v=0111寻径到 $v \oplus 2^{2-1} = 0111 \oplus 0010 = 0101$

i=i+1=3

第4步：$r_3=0$，所以跳过一步

$i=i+1=4$

第5步：$r_4=1$

所以从v=0101寻径到 $v\oplus 2^{4-1}=0101\oplus 1000=1101$，结束。

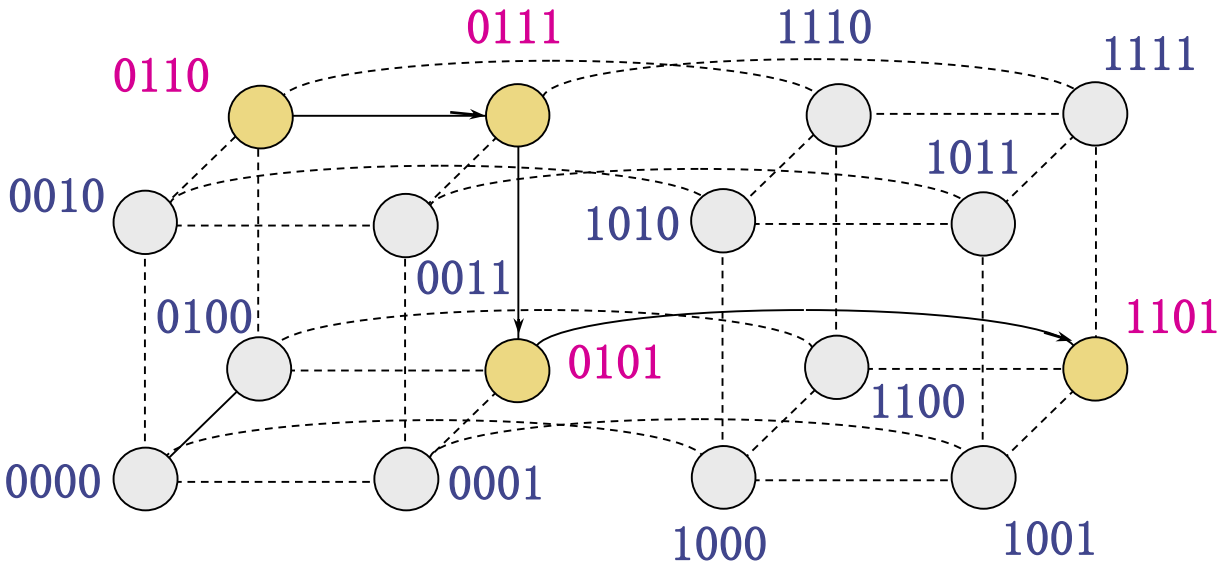
因此路径为：0110→0111→0101→1101

源：s=0110

目的：d=1101

路径： 0110→0111

→0101→1101



7.5.4 选播和广播寻径算法

1. 多计算机网络中会出现以下4种通信模式：

- **单播**：对应于一对一的通信情况，即一个源节点发送消息到一个目的节点。
- **选播**：对应于一到多的通信情况，即一个源节点发送同一消息到多个目的节点。
- **广播**：对应于一到全体的通信情况，即一个源节点发送同一消息到全部节点。
- **会议**：对应于多到多的通信情况。

- 通道流量和通信时延是常用的两个参数。
  - 通道流量可用传输有关消息所使用的通道数来表示。
  - 通信时延则用包的最大传输时间来表示。
- 优化的寻径网络应能以最小流量和最小时延实现相关的通信模式。

2. 以网格网络为例，讨论选播和广播。

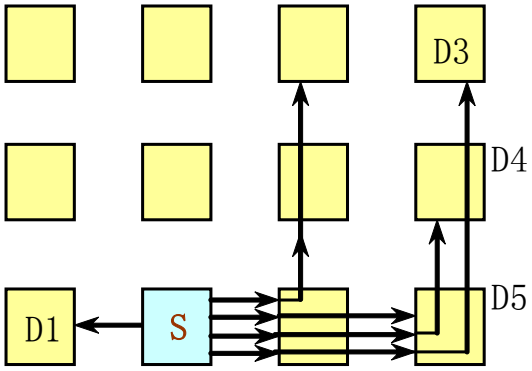
在3×4网格上实现的选播寻径

源节点：S

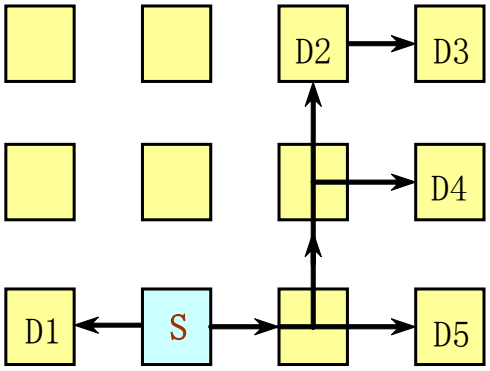
传送一个包到标号为Di 的5个目的节点

（i=1，2，…，5）

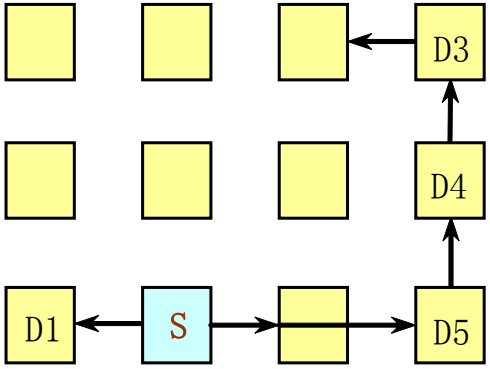
- 图（a）：目的节点为5个的选播可以用5次单播来实现
  - X-Y寻径的流量需要用1＋3＋4＋3＋2＝13条通道。
  - 到D3的路径最长，所以时延是4。
- 图（b）和（c）给出了两种选播寻径模式，流量分别为7和6。
  - 在虫蚀网络中，用图（c）的选播寻径模式比较好。
  - 在存储转发网络中，则用图（b）的寻径模式比较好，时延较短。
- 图（d）：使用一棵4层的生成树可以把一个包从节点S广播到所有的网络节点
  - 产生的时延和流量最小



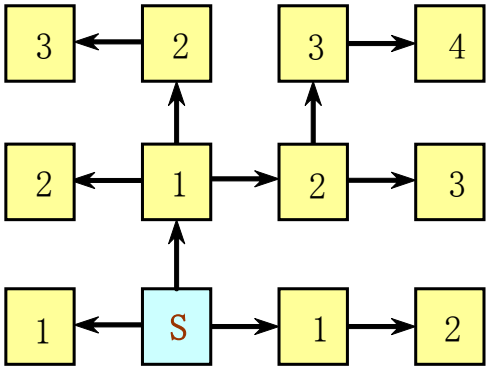
（a）5 次单播，流量为 13，距离为 4



（b）流量为 7，距离为 4 的选播方式



（c）流量为 6，距离 5 的选播方式



（d）通过树结构广播给所有结点