第3章 SIMD计算机

CHAPTER

知识要点

3.1 互连网络的基本概念

互连网络是由多个交换开关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内部多个处理机或多个功能部件之间的相互连接。

互连网络的特性可以从4个方面来表征：

(1) 拓扑结构：网络物理互连结构的几何图。大多数并行处理机都使用具有高度规整拓扑的互连网络。

(2) 交换策略：确定消息中的数据如何经过它的路径，可分为电路交换和包交换两大类，包交换又可分为存储转发方式、虚拟直通方式和虫蚀方式等类型。

(3) 流控制机制：确定消息何时沿它的路径传送，尤其是当两个或多个消息试图同时使用同一网络资源时，流量控制就特别重要，它应解决冲突和避免死锁。

(4) 寻径算法：确定消息可通过网络图的哪些路径，它将一组可能的路径限制到更小一组的合理路径。

互连网络的连接特性可以用互连函数来描述，它代表互连网络的输入端和输出端之间的一一对应关系。如果将互连网络的 N 个输入端和 N 个输出端分别用 $0,1,2,\cdots,N-1$ 来编号，则互连函数 $f(x)$ 表示输入端 x 与相应的输出端 $f(x)$ 实现了互连。一般来说，x 可采用端口编号的二进制值来表示，则相应的互连函数就可以写成 $f(x_{n-1}\ x_{n-2}\cdots\ x_1x_0)$，其中，$x_i$ 为二进制值。此外，互连函数还可以采用输入输出对应表示法，也可以理解为输入到输出的置换。

3.2 单级互连网络

单级互连网络是具有 N 个输入多路分配器和 N 个输出多路选择器的

开关网络，按照互连函数的控制，在处理单元间建立起所需要的连接通路。下面是几种常用的单级互连网络。

1. *n*-立方体互连网络

在 n 维立方体中，每个结点与其他 $n-1$ 个结点直接相连，即结点度为 n。它的结点数 $N=2^n$ 个。结点地址使用 n 位二进制数表示，其值从 0 到 $N-1$。如果一个结点的地址为 d，则与其相邻的结点的地址只有一位二进制位与 d 不同。如果把每一个结点看做一个处理单元(处理器)，则它只能直接连到其二进制地址的某一位取反的结点上，这样就构成了一个 n-立方体网络。n-立方体网络的互连函数为：

$$\mathrm{Cube}_k(x_{n-1}x_{n-2}\cdots x_{k+1}x_kx_{k-1}\cdots x_1x_0)=x_{n-1}x_{n-2}\cdots x_{k+1}\overline{x_k}x_{k-1}\cdots x_1x_0$$

由于 k 可以是其中的任意一位，所以立方体变换可以有 $n=\log_2 N$ 种。对于含 8 个结点的 3-立方体互连网络来说，它的 3 个互连函数为：

$$\mathrm{Cube}_0(x_2x_1x_0)=x_2x_1\overline{x_0},\quad \mathrm{Cube}_1(x_2x_1x_0)=x_2\overline{x_1}x_0,\quad \mathrm{Cube}_2(x_2x_1x_0)=\overline{x_2}x_1x_0$$

Cube_0 实现的是二进制最低位上的编码互反的结点的连接，Cube_1 和 Cube_2 则分别为中间位或最高位上编码互反的结点的连接。

2. PM2I 互连网络

PM2I 互连网络是“加减 2^i”单级网络的简称，它能实现 j 号处理单元与 $j\pm2^i$ 号处理单元的直接连接。其互连函数为：

$$\mathrm{PM2}_{+i}(j)=(j+2^i)\bmod N$$

$$\mathrm{PM2}_{-i}(j)=(j-2^i)\bmod N$$

式中，$0\leqslant i\leqslant n-1$，$0\leqslant j\leqslant N-1$，$n=\log_2 N$，$N$ 为结点数。

一般情况下，总有 $\mathrm{PM2}_{+(n-1)}=\mathrm{PM2}_{-(n-1)}$，因此，PM2I 互连网络共有 $2n-1$ 种不同的互连函数。ILLIAC-Ⅳ中的处理单元间的互连，就是只采用了 PM2I 互连网络中的 $\mathrm{PM2}_{\pm0}$ 和 $\mathrm{PM2}_{\pm3}$ 4 个互连函数，它是 PM2I 网络的特例。PM2I 网络的最大距离为 $\left\lceil\frac{n}{2}\right\rceil$。

3. 混洗交换互连网络

混洗交换互连网络由全混洗和交换两种互连函数组成。

全混洗是将处理单元分成数目相等的两部分，进行像均匀洗牌一样的操作，即一个隔一个地相连接。全混洗互连网络的互连函数为：

$$\mathrm{Shuffle}(x_{n-1}x_{n-2}\cdots x_1x_0)=x_{n-2}\cdots x_0x_{n-1}$$

这相当于将处理单元的二进制地址位中的最左位移到最右位的循环移位。此时每个处理单元只有一条直接连接通路。这种互连网络的缺点是：当处理单元地址位为全“0”或全“1”时，它们将无法与网络中其他的处理单元相连。

混洗交换网络中，最远的两个处理单元(全“0”和全“1”)间的连接需要 n 次交换和 $n-1$ 次混洗，因此它的最大距离为 $2n-1$。

4. 蝶式互连网络

蝶式互连网络的互连函数定义为：

$$\mathrm{Butterfly}(x_{n-1}x_{n-2}\cdots x_1x_0)=x_0x_{n-2}\cdots x_1x_{n-1}$$

也就是将二进制的最高位和最低位相互交换位置。类似地，还可以定义子蝶式互连

(Butterfly$_{(k)}$)和超蝶式互连(Butterfly$^{(k)}$),它们的互连函数如下:

$$\text{Butterfly}_{(k)}(x_{n-1}x_{n-2}\cdots x_{k+1}x_kx_{k-1}\cdots x_1x_0) = x_{n-1}x_{n-2}\cdots x_{k+1}x_0x_{k-1}\cdots x_1x_k$$

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

3.3　多级互连网络

为了构建多级互连网络,可使用交换开关将单级网络级联起来,其中每级都使用多个交换开关模块,而相邻级间则使用固定的单级网络作为级间连接。不同类型的多级互连网络,其交换开关模块和使用的级间连接方式也有所不同。为了在输入和输出之间建立需要的连接模式,可动态改变交换开关的状态。因此,决定多级互连网络特性的主要因素有交换开关、拓扑结构和控制方式。

1. 多级立方体网络

多级立方体网络是将具有 Cube_0、Cube_1 和 Cube_2 三种互连函数的三个单级立方体网通过交换开关级联起来构成的。

多级立方体网络可以同时支持多个配置,但有些配置则由于开关设置上的冲突而无法实现,因此多级立方体网络属于阻塞型网络。

STARAN 网络是已获得成功应用的一种多级立方体网络,它的开关可以按级控制,也可以按部分级控制。

按级控制方式可以实现输入输出端的交换置换,这时的网络称做交换网络。

当第 i 级的控制信号为"1"时,该级所有的交换开关处于交换状态;控制信号为"0"时,该级所有的交换开关处于直通状态。设 $F=f_2\ f_1\ f_0$ 表示第 2、1、0 级的控制信号值。当第 i 级的控制信号为"1",而其余各级的控制信号为"0"时,实现的互连函数恰好是 Cube_i。而如果 $F=(011)$,就有 Cube_0 置换,再加上 Cube_1 置换,记为 $\text{Cube}_0+\text{Cube}_1$。

按部分级控制方式可以实现移数置换,这时的网络称做移数网络。

在 $N\times N$ 的 STARAN 网络中,把第 i 级的 $N/2$ 个开关分成 $i+1$ 组,每组一个控制信号。对于 $n=\log_2 N$ 级开关来说,从第 0 级到第$(n-1)$级所需的控制信号分别为 1,2,…,n 个,因此控制信号共有 $n(n+1)/2$ 个。当 $N=8$ 时,设 $F=(f_{23}f_{22}f_{21}f_{12}f_{11}f_0)$,共包含 6 个控制信号。

一个 $N\times N$ 的 STARAN 网络,在采用部分级控制后,可以实现$(n^2+n+2)/2$ 种移数置换。当 $N=8$ 时,可实现的移数置换为 7 种。

2. Omega 网络

通过若干级交换开关将多个全混洗网络连接起来组成的多级网络称为 Omega 网络,它又称为多级混洗交换网络。

一个 $N\times N$ 的 Omega 网络,所用的开关级数为 $n=\log_2 N$,每一级由一个全混洗网络和一列 $N/2$ 个开关组成。每个开关都有两个输入、两个输出以及直送、交叉、上播、下播 4 种可能的连接状态。网络使用的总开关数为$(N/2)\log_2 N$。

3. 基准网络

在输入输出之间用交换开关依次将一个逆混洗网络和一个子蝶式网络连接起来,就构成了基准网络。该网络所用的交换开关有"直通"和"交叉"两种状态,并采用终端标记

法对每一个开关进行单元控制。

基准网络常用于对多级互连网络的研究，将基准网络作为中间介质模拟某种网络的拓扑和功能等。一次通过基准网络，可以达到位序颠倒的置换，二次通过基准网络，则可实现任意组合的置换功能。

4. 多级 PM2I 互连网络

多级 PM2I 互连网络又称为数据变换网络。教材中图 3.16 给出了一个 $N=8$ 的 3 级的 PM2I 互连网络连接情况。各级中的处理单元按 PM2I 互连函数连接起来。就第 i 级而言，每个输入端 j 都有三根连接线分别连到出端 j、$(j+2^i) \bmod N$ 和 $(j-2^i) \bmod N$。第 0 级完成的是 $PM2I_{\pm 0}$，第 1 级完成的是 $PM2I_{\pm 1}$，而第 2 级实现的是 $PM2I_{\pm 2}$。由于单级 PM2I 互连网络的最大距离为 $\left\lceil \frac{n}{2} \right\rceil$，但组成多级 PM2I 网时仍用了 n 级，因此在这种网络中提供了冗余通路。

在有 $\log_2 N$ 级的多级开关网络中，从任一个输入端到任一个输出端间都只有一条特定的通路可以接通，但有时两条通路却因为交换开关设置上的冲突，并不都能够同时接通。所以在使用一次网络的情况下，只能接通一条通路，而阻塞另一条通路。

实际上，N 个输入端与 N 个输出端之间连接的可能排列有 $N!$ 种，但输入与输出间的 $\log_2 N$ 级交换开关共有 $(N/2) \times \log_2 N$ 个，它们可能的排列只有 $2^{\left(\frac{N}{2} \times \log_2 N\right)} = (\sqrt{N})^N$ 种，比 $N!$ 少，因此，必然有一些连接情况会被阻塞。而后面将要介绍的两种多级开关网络则是非阻塞网络。

5. Benes 网络

将两个基准网络进行背对背的互连，就构成了一个基本的 Benes 网络。该网络至少有两个以上的通道能满足同一对结点间的互连要求，因此可以避免可能发生的阻塞情况。Benes 网络不但可以满足输入输出间所有可能的 $N!$ 种置换，还会有超额的通路冗余量，因此，Benes 网络是一个可重排的非阻塞网络，网络中如果发生阻塞，可以通过重新设置开关状态加以避免。Benes 网络需要的开关级为 $2 \times \log_2 N - 1$，每级由 $N/2$ 个具有“直通”和“交换”功能的 2×2 开关组成。

在介绍的阻塞网络中，为实现某种连接关系，二输入开关的两个输入的开关状态设置有可能会发生冲突，因此，Benes 网络规定对开关的两个输入控制只能以其中的一个为准，而另一个则作为被动连接。如果以上端为准，当 $d_i=0$ 时，两个输入都作为直通连接；反之，则进行交换连接。如果规定以下端为准，当 $d_i=1$ 时，两个输入都作为直通连接；反之，则进行交换连接。这种规定虽然使连接的灵活性受到限制，但是有效地防止了开关中冲突的发生。而且，由于 Benes 网络的开关级增加了将近一倍，一个开关中被动连接的那个输入端经过级间互连的置换，通常还会有不少于 $n-1$ 次作为主动控制的机会，使之能够准确地连通到指定的输出端。

6. Clos 网络

Clos 网络的结构如配套教材中图 3.20 所示，该网络为一个 3 级 Clos 网络。这种类型的网络可以用 m、n、r 三个参数来表示。其中，m 是输入级所用开关的输出端数或输出级所用开关的输入端数，n 为每个输入级开关的输入端数或每个输出级开关的输出端数，

r 则是中间级每个开关的输入输出端数。

该网络的每一级都有 12 个开关结点，共 36 个开关结点。如果使用交换开关网络连接，由于输入和输出结点各有 4 个，共需 $4^2=16$ 个交叉开关。在这种情况下，使用交换开关网络更经济一些。但是，当输入结点数量较大时，Clos 网络使用的开关结点就会小于 N^2。

3.4　互连网络的消息传送

并行多处理机系统的各处理机结点通过在互连网络上传送消息实现信息交换。因此，消息传递机制的研究在互连网络的研究中占有非常重要的地位。本节将介绍互连网络消息传递机制中的基本问题，包括寻径方式、死锁的避免、流量控制策略和寻径方法等。

1. 消息格式和寻径方式

消息是结点间通信的逻辑单位，它由任意数目且长度固定的包组成，因此消息的长度是可变的。包是包含寻径目的地址的基本单位，其典型长度为 8～64 字节。由于同一条消息的不同包选择的路径不同，可能不按发送顺序到达目的结点，因此每个包都需要一个序号。包可以被进一步分成一些固定长度的数据片，寻径的目的地址和包序号形成头片，其后是数据片，另外还可能有完成检错或纠错功能的尾片。头片和尾片将包封装起来。包的长度取决于寻径方式和网络的实现方法，典型的包长度为 64～512 位。另外，包和片的大小还与通道频宽、寻径器设计以及网络流量密度等有关。

消息的寻径可分为线路交换、存储转发、虚拟直通和虫蚀寻径等 4 种方式。

1）线路交换

线路交换方式是指在传递一个消息之前，在源结点和目的结点之间先建立一条物理通路，然后再传递消息，参看教材中图 3.22，其中的阴影部分为包的头片，其余为数据片。因此，该寻径方式需要提前预订整个路径，预留所需的开关端口，一旦预订成功，包就可以全速地由源结点流向目的结点。

线路交换方式的优点是在包发送时，能实现无竞争无干扰的全速传送；而缺点是需要提前预留网络资源，使用效率低。另外，在并行多处理机中，一般以频繁的小信息包的方式通信，因此，如果采用线路交换方式，在传递一个消息之前，需频繁地建立从源结点到目的结点的物理通路，这部分开销将会很大。

2）存储转发

在存储转发网络中，包是信息流的基本单位。网络中每个结点都要有一个包缓冲区。在包从源结点经过一系列中间结点到达目的结点的过程中，每个中间结点都要将整个包存入包缓冲区，当所要求的输出链路和下一结点的包缓冲区都可用时，再将包转发到下一结点。

存储转发寻径方式虽然不需要预订资源，但其缺点是传送延迟时间大且与源至目的之间的行程数成正比。此外，为能保存最大的包和避免几条通路向同一结点传送时造成包的丢失，需要较大的包缓冲区。

3）虚拟直通

在虚拟直通寻径方式中，每个结点仍有包缓冲区，但为了减少时延，只要含有寻径信息的包的头片到达后即可做寻径选择，而不必等到全部包被缓存之后再转发，这大大地减

少了延迟。

当出现结点阻塞时，虚拟直通方式只有将整个消息全部存储在寻径结点中，直到寻径链路不阻塞时才能将消息发出，这就需要每个寻径结点都有足够的缓冲区来存储可能出现的最大的信息包。在这一点上，虚拟直通方式与存储转发的寻径方式是一样的。不仅如此，虚拟直通寻径方式在最坏情况下的通信延迟与存储转发包交换方式也是一样的。

4）虫蚀寻径

虫蚀寻径方式把包进一步分成更小的片，在与结点相连的寻径器中设置片缓冲区，消息从源结点传送到目的结点要经过一系列寻径器，同一个包中所有的片连续不分离地以流水方式顺序传送。只有头片知道包将发往何处，所有的数据片必须跟着头片，用头片直接开辟一条从输入结点到输出结点的路径。每个包相当于一条蠕虫，包中的每个片相当于虫的一个节，各片以流水方式在网络中向前“蠕动”，即以节为单位顺序地向前爬行，因此被形象地称为“虫蚀寻径”。

在虫蚀寻径方式下，不同的包可以交替地传送，但不同包的片不能交叉，否则它们有可能被送到错误的目的地。

虫蚀寻径方式已在许多新型的多处理机系统中被采用。其优点是：各结点不再需要大的包缓冲区，只要很小的片缓冲区就够了；所有的片以流水方式向前传输，利用时间并行性来减少网络传输时延，其网络传输时延正比于消息包的长度，传输距离对它的影响很小，在最好情况下的传送延迟时间 $T=P/b$，等同于虚拟直通寻径方式；链路共享性好、利用率高，对链路的预约和释放是结合在一起的过程，一旦有一段新的链路后，将立即放弃用过的一段旧链路；允许寻径器复制消息包的片并把它们从多个输出链路输出，因此易于实现选播和广播通信方式。

虫蚀寻径方式的缺点是：当消息的一个片被阻塞在某结点时，整个消息的所有片都将被阻塞，占用了结点资源。

2. 死锁的产生和规避

1）产生死锁的原因

在网络信息传送中的死锁是指一个包在向它的目的地前进的过程中等待一个事件的发生，而这个事件又不能发生。缓冲区或链路上的循环等待都会引起死锁。

对于存储转发网络来说，缓冲区死锁请参看教材中的图 3.26。A、B、C、D 四个结点分别向 C、D、A、B 结点发送信息。传输开始时，四个结点的缓冲器都是空的，所以这四个结点都沿逆时钟方向向下一结点发送数据包。B 结点中存放的是 A 发送给 C 的数据包，标注为(A,C)；C 结点中存放的是(B,D)；D 结点中存放的是(C,A)；A 结点中存放的是(D,B)。此时，四个数据包占满了四个结点的四个缓冲区，因此，这四个结点都向前一个结点发出禁止传输的信号，导致四个结点循环等待，这样就形成了缓冲器死锁。此时除非丢弃某个数据包，否则死锁不会解除。

在采用虫蚀寻径的网络中，也会产生死锁现象。请参看教材中图 3.27，四个消息 M1～M4 沿四条链路 L1～L4 正在做虫蚀方式的传输，四个消息的四个片同时占用了四条链路，且它们所要到达的结点恰好都被另一条消息占用着。于是四个链路都没有空，只好将各自的片存放在当前已到达的结点上，形成了链路死锁。如果循环中没有一条链路

被释放，死锁状态将一直持续下去。

2）规避死锁的方法

网络的通信链路实际上是由许多源和目的对共享的，因此可以形成多个虚拟通道。虚拟通道是结点之间建立起来的逻辑通道，它由源结点的片缓冲区、结点间的物理通道以及接收结点的片缓冲区组成。教材中的图 3.28 说明了四条虚拟通道共享一条物理通道的原理。

当物理通道分配给某一对缓冲区时，这一对的源缓冲区和接收缓冲区形成了一条虚拟通道。图 3.28 中有两个片缓冲器已经分配给两个包使用，即在一条物理通路上建立了两个虚拟通道。这条物理通道被两条虚拟通道分时共享。由于源结点和接收结点各有 4 个片缓冲区，因此最多可以在这两个结点之间建立四个虚拟通道。

除了有关的缓冲区和通道之外，必须用通道状态标志来表示不同的虚拟通道。源缓冲区存放等待使用通道的片；而接收缓冲区存放由通道刚刚传送过来的片。

虚拟通道可以用单向通道或双向通道实现。把两条单向通道组合在一起可以构成一条双向通道，这不仅提高了通道的利用率，而且还可以使通道的频宽加倍。当然，双向通道工作时需要专用的仲裁线来控制信息流的方向，这会造成网络延时和成本的增加。

此外，虚拟通道可能会使每个请求可用的有效通道频宽降低，因此在确定虚拟通道数目时，需要对网络吞吐量和通信时延进行折中考虑。实现数目很大的虚拟通道需要使用高速的多路选择开关。

增设虚拟通道是解决死锁问题的有效方法。根据虚拟通道的含义，如果在结点上增加一个缓冲器，就相当于多了一条虚拟通道。

在虫蚀寻径方式下，也可以通过在结点中设立若干个片缓冲器，建立多个虚拟通道来避免死锁。

3. 流量控制策略

当两个或更多的包到达同一结点时，它们可能会同时请求使用同一个接收缓冲区或者同一个输出链路，这样就会产生竞争缓冲区或链路资源的冲突，因此必须研究解决冲突的策略。

1）缓冲策略

这种控制策略是在结点内另设一个包缓冲区，当两个（或多个）包同时到达时，只允许一个包使用片缓冲区和输出链路，而使其他的包暂存在包缓冲区中，当片缓冲区和输出链路可以使用时再传送，如教材中图 3.31 所示。这种办法的好处是不丢失也不拒绝其他的包，但是需要一个容量较大的包缓冲区。若包缓冲区过大，就不可能放在寻径器上，因此要用结点的本地存储器作为包缓冲区，这将会引起较大的存储延迟。

2）阻塞策略

这种控制策略在允许一个包在使用片缓冲区时，同时以不就绪信号通知另一个包的发送结点，令其停止发送。这种信号可一级级地向上传递，直到源结点停止发送包（片）。虫蚀寻径方式的互连网络结点在出现冲突时，采用的就是这种阻塞策略。

3）扬弃策略

这种策略在处理包冲突时，只允许一个包使用片缓冲区和输出链路，而丢弃另一个包，同时通知被丢弃包的发送结点等待一段时间后再重发被丢弃包。这种策略可能会出现严重的资源浪费并导致包的输送率不稳定，现在已很少使用了。

4）绕道策略

这种控制策略在处理包冲突时，不是把另一个包丢弃，而是为它选择另一条链路送出，绕道到达目的结点。这种绕道策略为包寻径提供了更大的灵活性，要求互连网络结点具备自适应选择路径的能力。但是绕道可能要使用更多的链路资源才能到达目的地，而且绕行的包有可能进入一个活锁循环，这将浪费网络资源。所谓活锁，是指包仍在传送但总不能到达它的目的结点。

4. 寻径方法

在互连网络中，消息由源结点传送到目的结点的过程中，通常存在多条路径可供选择。为了充分利用互连网络的可用带宽，尽量减少传送延迟并避免死锁，必须设计有效的寻径方法。此外，网络的拓扑结构在很大程度上决定了各类不同寻径方案的可用性和实现效率。对于不同的拓扑结构来说，所采取的寻径策略是不一样的。

互连网络中的消息寻径方法可分为确定寻径和自适应寻径两类。

1）确定寻径

采用确定寻径方法时，通信路径是预先唯一确定的，它完全由源和目的地址决定，与网络的状况无关。*X*-*Y* 寻径和 *E*-立方体寻径是两种典型的基于维序概念的确定寻径方法。它们的共同特点是按照多维网络维序的特定顺序来选择后继链路。

（1）*X*-*Y* 寻径

该方法是在二维网格网络中的寻径方法。假定消息要从源结点 $S=(x_1,y_1)$ 到目的结点 $D=(x_2,y_2)$，从 S 开始寻径，首先沿 X 方向前进，直到 D 所在的第 x_2 列为止，然后沿 Y 方向前进，直到 D。

（2）*E*-立方体寻径

在 *n*-立方体网络中，可采用 *E*-立方体寻径算法。对于 $N=2^n$ 个结点的 *n*-立方体网络来说，令源结点的二进制编码为 $S=S_{n-1}\cdots S_1S_0$，目的结点的二进制编码为 $D=d_{n-1}\cdots d_1d_0$。n 维中的第 i 维对应于结点地址的第 $i-1$ 位。

除了最短路径之外，还有多条非最短路径。这样一来，*n*-立方体网络的可选路径就会有很多，因此它的可靠性是比较高的，若某个或某些结点出现故障，剩下的结点仍可以完成源-目的结点间的通信。对于其他拓扑结构的网络，有些也可能蕴涵在 *n*-立方体中，因此也可以利用 *E*-立方体寻径算法进行消息寻径。

2）自适应寻径

如果寻径方法仅允许一条确定的路径，那么单条链路失效将导致消息传送失败。自适应寻径允许结点对之间有多条合法路径，因此可以绕开故障传送。此外，自适应寻径还可以在可用链路上更广泛地分布流量，从而改善网络的利用率。前面讲到的简单的确定寻径的方法有可能因为流量的集中而在网络中引起竞争。

自适应寻径可以有多种方式。对基于源结点的寻径来说，源结点可以简单地在多条

合法路径中挑选传送路径，并根据选择建立信息包的头片。对基于表驱动的寻径来说，可以通过为多条路径建立寻径表项来完成，通过查找寻径表可以了解链路的情况，自动选择合适的链路来传送消息。对基于运算的寻径来说，需要给数据包头附加额外的控制信息，并由寻径器解释这些信息，以选择最佳的传送路径。

自适应寻径的链路选择是由结点上的寻径器根据寻径中碰到的流量动态地决定的。如果所希望的输出端口之一被阻塞或失效，寻径器可以选择一个替代链路送出数据包。最小自适应寻径仅沿着到达目的结点的最短路径引导数据包，每一次寻径都必须缩短到达目的结点的距离。允许使用所有最短路径的自适应算法称为完全自适应算法，否则就是部分自适应的。非最小自适应寻径的一种做法是：寻径器从不缓冲数据包，如果一个以上的数据包指向同一个输出链路，寻径器只将其中的一个包送往其目的结点，并将其他的包传送到别的链路，而不管经过这些链路传送后是否会离目的结点更近。

由于自适应寻径的路径选择灵活且具有动态性，因此很容易产生死锁。采用自适应寻径时避免死锁的方法有：

(1) 转弯模型；

(2) 虚拟网络。

5. 选播和广播寻径

互连网络中的通信除了一对一的单播通信模式外，还有一对多的选播、广播模式以及多对多的会议模式。前面介绍的结点间的通信情况主要是单播模式，这里主要介绍选播和广播通信模式的寻径算法，不但要满足基本的通信传输需要，还要达到较高的通信效率。

一般来说，通信时延和通道流量是描述通信效率常用的两个参数。通信时延反映了网络中消息的传输速度，可以用信息包的最长传输时间来描述。通道流量则反映了网络中的通信负载情况，可以用传输消息所使用的链路数来表示。它不但与网络中的物理链路、缓冲器和寻径器等通信资源有关，还与网络的拥塞和死锁问题密切相关。

在各种通信模式下，网络寻径算法的设计与优化的目的是要达到流量最大化和时延的最小化。然而与大多数多目标优化算法类似，它只能是在这两个优化目标之间的折中。而且，在使用不同的交换技术时，其优化的侧重点也不尽相同。例如，在存储转发网络中，时延优化是最重要的问题；而在虫蚀寻径网络中，流量的优化对通信效率来说则具有更重要的意义。

3.5　阵列处理机

并行处理机的基本形式是阵列处理机，它是采用资源重复方式进行并行处理的 SIMD 计算机。它将大量重复设置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并行执行同一指令规定的操作，所有处理单元同时进行相同的操作。

阵列处理机特别适合于对规则数据进行运算，如向量、矩阵运算等。它可将向量或矩阵的各元素分布到各个处理单元中，在阵列运算指令控制下同时进行运算。因此，要求在

阵列处理机上运行的程序具有较好的数据并行性。

阵列处理机的结构是与所采用的并行算法紧密联系在一起的。由于阵列处理机中通常采用简单和规则的互连网络来实现处理单元间的连接操作，从而限定了它的应用领域，因此它是以应用算法为背景的专用计算机。

由阵列处理机执行的并行指令包括向量运算指令、数据寻径指令以及屏蔽操作指令。向量运算指令对向量数据进行运算操作，其表示方式类似于流水向量计算机中的指令，只是并行处理的方式不同。数据寻径指令完成运算单元之间的各种数据交换操作。当结点间的数据交换需要经过多个中间结点时，交换步骤需要在统一指令的控制下完成。屏蔽操作指令用于控制处理单元参加阵列运算和数据寻径操作。由它来决定在给定操作中的活跃处理单元和不参加指令操作的处理单元。阵列机中所有的处理单元均以同步锁定方式工作，所有处理单元的并行操作同时启动，同时结束。每个处理单元都具有局部的寄存器，能够分别对阵列中的不同元素进行算术和逻辑运算。

由此可见，阵列处理机是一个同构型并行机，但它的控制器是一个标量处理机，而且为了完成 I/O 操作和对操作系统的管理，还需要一个前端机，因此，实际的阵列处理机系统是由上述三部分构成的一个异构型多处理机系统。

1. 阵列处理机的基本结构

阵列处理机通常由一个控制器(CU)、若干个处理器单元(PE)、若干个存储器模块(M)和一个互连网络(ICN)组成。由 CU 控制将指令广播给系统中的各个 PE，所有活跃的 PE 以同步方式执行相同的指令，从相应的存储器中取得所需数据。ICN 用于各个 PE 之间或 PE 和 M 之间的通信连接。阵列处理机可分为分布式存储器的阵列机和集中共享存储器的阵列机两种基本结构。

2. 阵列处理机的并行算法

在阵列处理机上运行的并行算法是与阵列处理机的结构紧密联系的。对于给定的题目，要研究和设计某种算法，使之能有效地归结为在某种结构的阵列处理机上的向量和数组运算。

1) 累加和

求累加和的并行算法是一个将 N 个数的顺序相加转换为并行相加的问题，一般采用成对递归相加算法。

2) 图像平滑算法

图像平滑算法可用来平滑输入图像的灰度级。I 和 S 分别表示输入和输出图像。假定 I 和 S 均含有 512×512 个像素。I 中的每个点是一个 8 位不带符号整数(0～255)，用来表示 256 个灰度级。平滑后图像中的每个点 $S(i,j)$ 是 $I(i,j)$ 和它的 8 个最邻近的像素的灰度级的平均值。这 8 个邻近像素点为 $I(i-1,j-1)$、$I(i-1,j)$、$I(i-1,j+1)$、$I(i,j-1)$、$I(i,j+1)$、$I(i+1,j-1)$、$I(i+1,j)$、$I(i+1,j+1)$。S 中的上、下、左、右边线上的像素，由于在 I 中的相应像素没有 8 个邻近像素，因而置为 0。