LACIN：线性排列的完整互连网络

Ram´on Beivide*†、Crist´obal Camarero*、Carmen Mart´ınez*、Enrique Vallejo*、Mateo Valero†
*坎塔布里亚大学，西班牙
†巴塞罗那超级计算中心，西班牙{ramon.beivide, cristobal.camarero,
carmen.martinez,enrique.vallejo}@unican.es, mateo.valero@bsc.es

这是发表于2025年12月IEEE计算机体系结构快报的一篇论文的预印本版本。R. Beivide、C. Camarero、C. Martinez、E. Vallejo 和 M. Valero，“LACIN：线性排列的完整互连网络”，载于 IEEE计算机体系结构快报，第 卷，第 01 期，第 1–4 页，PrePrints 5555，doi：10.1109/LCA.2025.3649284。可在此处找到：
https://doi.org/10.1109/LCA.2025.3649284

arXiv:2601.05668v1 [cs.AR] 9 Jan 2026

摘要

有几种互连网络采用完全图拓扑。规模适中的网络可以基于单个完全图构建。然而，Dragonfly 和 HyperX 等大规模网络分别采用完全图的层次化组合或多维组合。

这些网络中的链路数量庞大，并且会随网络规模迅速增长。本文介绍 LACIN，这是一组采用相同索引端口连接交换机的完全图实现方案。这种网络实现方式降低了布线和路由的复杂度。LACIN 有助于部署不同规模的并行计算机网络，从超大规模集成电路系统到最大型超级计算机均可适用。

关键词。 互连网络、完全图、布线部署与管理、分组路由。

1 引言

本文将每对交换机之间都通过链路连接的互连网络称为完整互连网络（CIN）。以往研究也将其称为全连接网络、全网状网络和全对全网络。无论连接的交换机数量多少，CIN 都具有可能达到的最小直径和平均距离，二者均为 1，因此相较于任何其他网络都具有优势。

规模为 N 的 CIN 由 N 台交换机组成。每台交换机都有 $N-1$ 网络端口，用于连接所有其他交换机。在均匀随机流量下，每个端点都能以全速率注入数据的适当维度 CIN，需要每台交换机配备 N 边缘端口以连接计算机，从而构成一个拥有 N^2个端点的系统。因此，交换机基数至少必须为$2N-1$，网络链路总数为 $N(N-1)/2$。从成本来看，每个端点需要 $1/N$台交换机和$(N-1)/2N$ 条链路。也就是说，每个端点的链路成本从下方趋近于 $1/2$ 。图 1

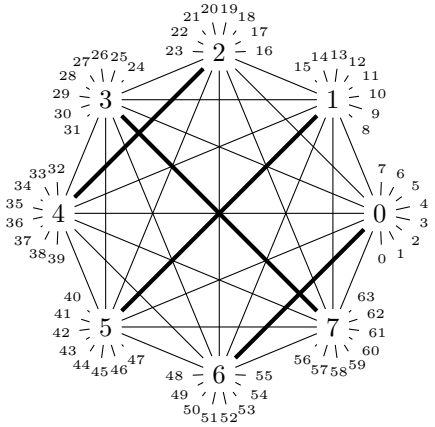

图 1：由 8 个交换机和 64 个端点组成的 CIN，其中 28 条链路中的 4 条以粗体显示。

展示了 $N=8$时 CIN 的一种表示形式，可连接 64 台端点计算机。它要求交换机基数至少为 15 个端口（8 个边缘端口加 7 个网络端口），并需要 $7\times 8/2=28$ 根线缆。

交换机基数无法任意增大，这限制了基于单个 CIN 构建的并行计算机的规模。有些系统规模较小，只需使用 CIN 即可互连。例如，片上网络 [1], ，以及用于连接服务器主板上的 CPU 和加速器芯片的多插槽系统 [2]。

针对更大规模的网络，人们提出了其他拓扑结构。HyperX [3] 是一种多维网络，源自完全图的笛卡尔积。其一维版本是单个 CIN，而二维和三维变体则用于 [4–6]。Dragonfly 应用于大型超级计算机，是一种基于完全图构建的两级分层网络 [7]。基于 CIN 的网络结构包含大量链路，这使其部署和维护更加复杂。

本文表明，在 CIN 中选择用于连接交换机的端口是一项重要的设计决策。本文分析了不同方案，并重点介绍了称为同端口的方案，该方案仅使用索引相同的端口连接交换机。同端口方案简化了布线和路由，并有利于线性实现，

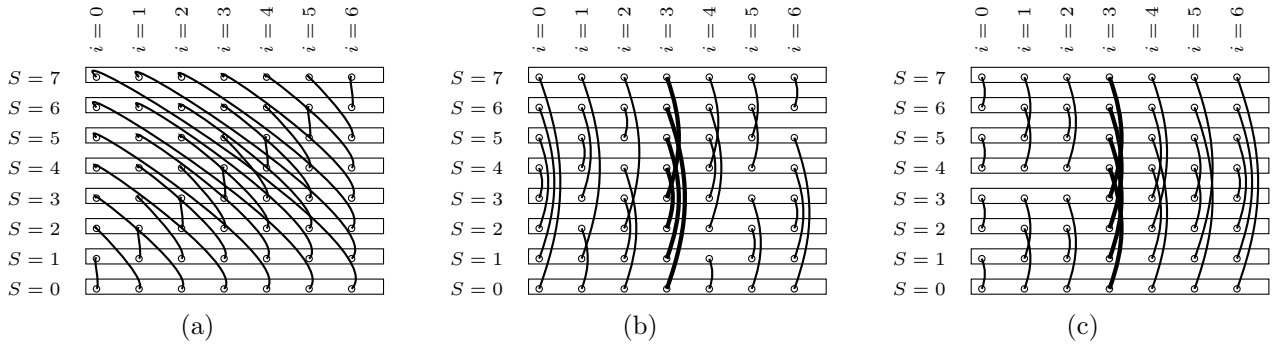

图2：端口Pa　　由 $N=8$ 构成的环绕 P 矩阵：(a) 异端口 Swap、(b) 各向同性端口圆和 (c) 各向同性端口X或 CIN实例。粗体显示的因子 $i=3$ 。

本文将其称为 LACIN。它们至少可应用于片上网络、多模块芯片网络、多插槽网络、印刷电路板网络和系统网络。据作者所知，此前尚未有人探讨过用于互连交换机的端口选择问题。一方面，将网络建模为图时，不会考虑各个端口。此外，这属于低层次的技术细节，公司不会公开披露。

其余内容安排如下：第2节介绍所选的CIN实例。第3节讨论它们的最小路由算法。第4节探讨LACIN的替代方案。第5节将LACIN应用于大规模网络，第6节总结全文。

算法 1 *Circle* LACIN（ N 为偶数）。确定通过端口 $P_{S,\,i}$ 确定每个交换机的邻居。

输入： 偶数N
  对于 i，从 0 到 N − 2 **执行**
    对于 S，从 0 到 N − 1 **执行**
      如果 S = N − 1 **则**
        % 将最后一个节点连接到其余节点。
        将端口 $P_{S,\,i}$ 连接到端口 $P_{i,\,i}$。
      否则，如果 S =i **则**
        % 与最后一个节点相连的任意节点。
        将端口 $P_{S,\,i}$ 连接到端口 $P_{N-1,\,i}$。
      else
        % 并行连接。
        将端口 $P_{S,\,i}$ 连接到端口 $P_{(2i-S) \bmod (N-1),\,i}$。

2 端口和CIN实例的选择

无论采用何种物理实现方式，我们都通过一个端口配对矩阵 P来对 CIN 进行建模，该矩阵有 N 行和 $N-1$ 列。元素 $P_{S,\,i}$ 表示交换机 S中索引为 i 的端口，其中 $0 \le S \le N-1$ 和$0 \le i \le N-2$。矩阵的 N（$N-1$）个端口根据完全图拓扑，通过 N（$N-1$）条/2 链路进行配对。不同的 P矩阵数量极其庞大，它们都对应一种 CIN 拓扑，但其端口的配对方式各不相同。我们将每一种配对方式称为一个 CIN 实例。我们考虑两类：一类要求每条链路都将具有相同索引的端口配对，称为等端口；另一类允许通过具有不同索引的端口将交换机配对，称为异端口。

异端口 CIN 实例可以通过依次将每个交换机与其他所有交换机连接起来自然获得，在每个交换机上使用第一个可用端口。本文将这种方法称为*Swap*：当 $S \le i$时，端口$P_{S,\,i}$ 与端口$P_{i+1,\,S}$ 配对；当 $S > i$时，则与端口 $P_{i,\,S-1}$ 配对。图2（a）展示了一个包含 8 个交换机的 CIN 的 Swap P矩阵。

文献中尚未研究过已知的等端口 CIN 实例。本文介绍了两种基于完全图分解的方案。偶数阶图的 1-因子是完美匹配，也就是说，将其顶点通过边两两配对。对于偶数 N，1-因子分解包含 $N-1$ 1 个因子，这与每台交换机的网络端口数量相同。等端口实例使用索引为 i 的 N 个端口来构建 1-因子i，其中 $0 \le i \le N-2$。接下来的两种分解长期用于分步全到全算法以减少拥塞 [8], [9], ，本文对其进行了定制，用以定义等端口 CIN 实例。

第一个提出的方案称为 *Circle*，它改编了循环赛中用于配对参赛者的一种算法。该 CIN 实例由算法 1 得出，其 P 矩阵（ $N=8$ ）见图2 (b)。虽然奇数规模图无法进行 1-因子分解，但可以通过从偶数情形（$N+1$）的 Circle P矩阵中删除最后一行及其关联链路，得到奇数情形的矩阵。每个交换机有一个端口保持空闲，但仍满足 isoport 属性。

第二个方案称为 异或，可用于 $N=2^n$。它基于逐位异或函数 $\oplus$，并作用于由 n 位表示的交换机标签。若 A 和 B 是两个交换机的标签，则连接这两个交换机的端口索引为 $i = A \oplus B - 1$。由于异或运算是自反的，若 $i = A \oplus B - 1$，则 $B = A \oplus$（$i+1$），这意味着交换机 B 是交换机 A 的邻居，可通过索引为 i的端口到达。因此，一般而言，端口 $P_{S,\,i}$ 连接到端口 $P_{S\oplus(i+1),\,i}$。图2 (c) 展示了一个含 8 个交换机的 CIN 的异或 P 矩阵。

无论交换机的物理排列方式如何，同端口CIN实例都具有多项优点。

它们可以减少网络部署和维护过程中的错误。首先，链路和端口使用 $N-1$ 颜色进行分类，从而便于电缆组织。其次，在某台交换机上连接到特定端口的链路，在其他交换机上也必须连接到具有相同索引的端口，这减少了不确定性。此外，正如下一节所示，它们还支持简单的路由算法。

3 最简路由

在 CIN 中，计算机 C 的全局地址由两位数字组成，即 $C = (C_1, C_0)$，其中 C_1 标识交换机，C_0则标识该交换机中的端点计算机。在图 1 中，C_1 对应于 C 二进制标签的高三位，而 C_0对应于低三位。假设源计算机 $A = (A_1, A_0)$ 通过最短路径向目的计算机 $B = (B_1, B_0)$ 发送数据。无论采用何种路由方式，如果两个端点连接到同一交换机 $A_1 = B_1$，或者数据包经过一次网络跳后已到达目标交换机，就会使用索引为 B_0 的端口将数据包送出。

当 $A_1 \neq B_1$时，在交换机 A_1 处必须使用哪个端口向 B_1 发送数据，由 CIN 实例决定。Swap 使用的端口索引在 $A_1 \leq B_1$ 时为 $i = B_1$，否则为 $i = B_1 + 1$ 。在异或路由中，使用索引为 $i = A_1 \oplus B_1 - 1$的端口。Circle 选择的端口由算法 2 描述，该算法由算法 1 推导而来。它需要将往返于 $N-1$ 交换机的路由视为特殊情况，因为这些路由涉及多个不同的比较或加法分支。

最简单的路由对应于异或，它仅依赖逻辑门和一个递减器。在 Swap 中，路由的成本稍高一些，因为除了递增器之外还需要一个比较器。在 Circle 中，路由比其他方式更复杂，但仍足够简单，可以直接用硬件实现，无需使用路由表。这些路由方式可以扩展到更高维度的网络。

在采用最短路由的 CIN 中，所有路径的长度均为一，因此不会发生死锁。先前的算法可以作为自适应机制的基础，根据流量特征采用最短路径或非最短路径。在这种情况下，必须通过限制非最短路由或使用两个虚拟通道来避免死锁。

4 LACIN：线性排列的 CIN

LACIN 是一种等端口 CIN 实例实现，其交换机呈线性排列。它的设计理念源于这样一种布局：将交换机沿直线放置，并使链路平行延伸，从而最大限度地减少交叉。这种布局能够满足多种需求，包括实现介质、大规模网络应用、易于部署、冷却、维修和维护。示例包括 [1] 中的片上CIN，以及采用 HyperX拓扑的 Intel PIUMA [5],。

算法 2偶数情况下 *Circle* 的分组路由 N 。

输入： 交换机数 N 为偶数，源交换机 A，目-
的交换机 B，$0 \leq$ A，B $< N-1$，A $\neq$ B。
输出： i，$0 \leq$ i $\leq N-2$，使得 A 的端口 i 连接到 B。

```
T ← A + B
如果 T = N − 1 则
    i ← 0
否则如果 B = N − 1 则
    i ← A
否则，如果 A = N − 1 则
    i ← B
否则，如果 T 为偶数则
    i ← T/2
否则，如果 T < N − 1 则
    i ← (T + N − 1)/2
else
    i ← (T − N + 1)/2
```

作为一种等端口 CIN 实例，LACIN 的所有端口都在列内相连。这种布局使交换机之间的链路笔直，与其他任何一维实现相比，可缩短链路总长度。LACIN 需要使用不同长度的导线，最长可达每台交换机的网络端口数。它需要w 条长度为 $N-w$ 的导线，其中 $1 \leq w \leq N-1$，因此导线总长度为 $(N^3-N)/6$。

相反，线性异端口实现需要使用斜向电缆，这意味着电缆长度更长。为简化起见，我们假设交换机与端口之间的距离相近。具体而言，采用 Swap 时，垂直长度为 k 的链路，其水平长度为$k-1$。因此，总长度约为 LACIN 总长度的 $\sqrt{2}$ =1。41 倍。

根据部署介质的不同，例如在超大规模集成电路或板载系统中，人们会力求避免导线交叉。在 LACIN 中，连接不同索引端口的导线属于不同的 1-因子，因此不会交叉。如下所示，在 Circle 中，同一 1-因子内的 $N/2$ 链路也不会交叉。

1-因子 i（其中 $0 \leq i \leq N-2$）由节点 i 与节点 $N-1$之间的连接所确定。任何 1-因子至少由 $(N/2)-1$ 条平行链路组成，而这些链路可能与连接节点 i 和节点 $N-1$的单条链路交叉。这条单链路可能与所有、部分或任何一条平行链路交叉。在图1和图2 (b) 中，属于 1-因子 $i=3$ 的链路以高亮显示。该 1-因子中有三条平行链路与连接交换机 3 和 7 的链路交叉。在 1-因子 $i=0$ 和 $i=6$ 中，所有链路都相互平行，因为在图1中，节点 0 和 6 都是节点 7 的邻居。一般而言，对于 1-因子 i，当 $0 \leq i \leq (N/2) - 1$ 时，可能发生交叉的链路会穿过 i 条平行链路；当 $N/2 \leq i \leq N-2$时，则会穿过 $N-2-i$ 条平行链路。如果任意 1-因子 i中的所有平行导线都从端口 i的右侧经过，而可能与它们交叉的导线从其左侧经过，那么最终布局中就完全不会出现导线交叉。

异或 LACIN 的类似布局更加复杂。本文采用的简单绘图会产生导线交叉，其数量会随着网络规模的增大而增加。例如，观察图2 (c) 中以 $i=3$ 高亮显示的 1-因子

表 1：不同一维 CIN 布局的属性汇总。

CIN 布局	同端口	交换机数量	布线长度*	布线成本†
Swap	No	Any	$\sqrt{2}$	1
Circle	Yes	Any	1	5
XOR	Yes	$N = 2^n$	1	—

*渐近总布线长度相对于最小值的归一化结果。
†相对于异或，额外的加法器和/或比较器数量。

可以通过在某些导线上引入转弯来避免其交叉，但由此产生的布局不会像 Circle 的布局那样简单。

表 1 总结了不同 CIN实例线性部署的主要特性。

或者，CIN的交换机无论是否为同端口，都可以沿两个物理维度排列。例如，对于 $N = 16$，基于 4 个分区、每个分区含 4 台交换机的二维部署如图3所示。其120条链路分为两个层级： $24 = 4{\times}6$ 条链路位于分区内部，另有96条链路连接不同分区，并分组为6束线束，每束包含16根导线。

HPE Dragonflies [7]采用相同方案实现组，不过使用了8个分区。这些分区在机架内分布为2列，每列包含4个分区；要将8个分区完全互连，需要28束、每束16根导线的线束。这种二维布局比LACIN更复杂，因为它在两个层级都需要对角链路，并会产生导线交叉。维护这种布局可能导致成本更高，也可能在隔离和更换故障链路时带来问题。此外，若将端点计算机所占的机架空间也计入部署范围，两种方案的布线总长度相近。

与这两种情况一样 $N = 2^n$，之前的二维实现完全可以基于异或CIN实例，并受益于其线缆组织方式和路由。

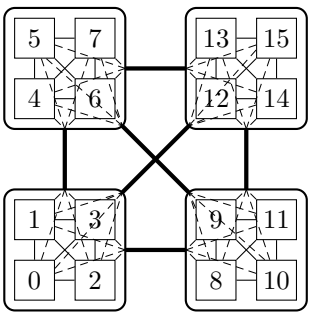

图3：一种规模为16的二维CIN实现，它由规模为4的CIN构成层次结构。外层CIN的链路为线束，每束包含16条内层链路。

5 面向大规模网络的 LACIN

LACIN 可应用于多维 HyperX 网络。以一个异或 LACIN 实现的直径为 3 的网络为例，该网络包含 65,536 个端点和 4,096 个交换机，排列方式为 16×16 HyperX。交换机使用 61 个端口：16 个边缘端口用于连接端点，每个维度使用 15 个网络端口连接其他交换机。端点计算机 C 的地址为 16 位，由 4 个各为 4 位的数字组成。三个高位数字 C_3，C_2，C_1标识交换机的坐标 Z，Y，X，低位数字 C_0则标识连接到该交换机的特定计算机。

假设一个机箱包含一台交换机及其对应的16台端点计算机。若一个机架堆叠16个机箱，则可在机架内部使用LACIN实现一个网络维度，例如 Z。 120 Z 链路沿机架的垂直维度延伸，并排列为15列，每列8根线缆。系统由 $16{\times}16$ 个机架组成一个正方形，通过LACIN连接，以在 XY 平面内实现另外两个网络维度。机架设有用于维度 X 的15个超级端口，以及用于维度 Y的另外15个超级端口，每个超级端口汇集16个独立端口。每行或每列的超级端口之间由120束线缆连接，每束包含16根线缆。超级端口和线束以15种颜色标记，从而可将线束划分为15个类别，每类8束。图4展示了一个小型 $4 \times 4 \times 4$ HyperX的此类部署。

为了说明路由，假设源计算机 A =(A_3，A_2，A_1，A_0) 通过一条最短路径向目标计算机B = (B_3，B_2，B_1，B_0) 发送数据包。它将在源交换机处使用端口 $P_{A_3 \oplus B_3 - 1}$，经由维度 Z进行路由。端口 $P_{A_2 \oplus B_2 - 1}$ 和 $P_{A_1 \oplus B_1 - 1}$分别用于 Y和 X 维度，而目标交换机处的端口 P_{B_0} 则用于传送数据包。如果某个维度的源数字与目的地数字进行异或运算后结果为 0，则跳过该维度。数据包穿越各维度的顺序可以是确定性的（例如维序路由），也可以是自适应的。HyperX 中的维序路由不需要虚拟通道（即每个端口只有一个缓冲区，或只有 1 个物理通道），而自适应机制通常需要虚拟通道来避免死锁。自适应非最短路由算法可以很容易地从这一最短路由方案推导出来。

Dragonfly 网络通过全局 CIN 连接交换机组，每个组则连接为本地 CIN。目前，组规模受电线传输范围限制，因此每组只能包含一台机架，可通过沿机架垂直维度部署 LACIN 来实现。将 LACIN 用于全局网络会形成线性机架布局。共封装硅光子技术消除了这一限制，使规模更大的组能够按机架行排列，并可将 LACIN 水平部署于本地 CIN。每台机架交换机分配部分端口用于全局 LACIN，该网络按列连接各行的全局端口。

6 结论

完全图在当前互连网络中的重要性值得详细分析。本文提出了连接 CIN 端口的不同方案，这些方案独立于其实现方式，但会影响布线组织和路由算法。本文提出了一种连接交

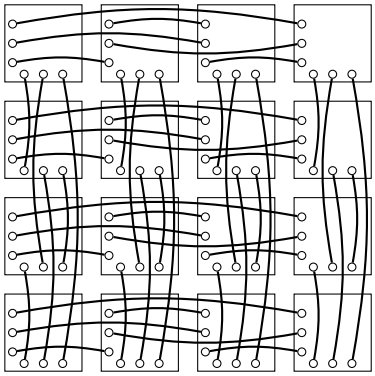

图 4：使用异或 LACIN 的 $4\times4\times4$ HyperX 顶视图。每个机架堆叠 4 个交换机。图中仅显示机架间连接，这些连接被分组为超级端口。

换机的方法，使每条链路在两端都使用相同的端口索引。在此基础上，本文介绍了 LACIN，即一种特定的线性CIN实现。LACIN具有多项优势，包括能够组织链路和端口，从而便于部署基于单个 CIN 的系统以及 HyperX 和 Dragonfly 等大型网络。此外，与任何线性CIN实现相比，LACIN所需的布线长度最短。

LACIN 的实现展现出有趣的权衡。虽然 Circle 可定义用于任意网络规模，但异或只能用于每个维度上的交换机数量为 2 的幂的网络。Circle 可实现避免导线交叉的最简单方案。在路由方面，尽管两者都可以避免使用路由表，但异或中的实现更为简单。

致谢

本工作获得西班牙科学与创新部以下合同的支持：PID2019- 105660RB-C22、TED2021-131176B-I00 和 PID2022- 136454NB-C21。C. Camarero 获得西班牙科学与创新部 Ramón y Cajal 合同 RYC2021-033959-I 的支持。R. Beivide 获得巴塞罗那超级计算中心（BSC）合同 CONSER02023011NG 的支持。

参考文献

[1]N. Akbari 和 M. Modarressi，《一种适用于神经形态架构的高性能片上网络拓扑》，*2017*年 *IEEE* 计算科学与工程国际会议，第 9–16 页，2017年。

[2] A. Cho 和 A. Daglis，《StarNUMA：通过内存池化缓解非统一内存访问挑战》，第*57*届 *IEEE*/美国计算机协会微体系结构国际研讨会（*MICRO*），第997–1012页，2024年。

[3] J. H. Ahn、N. Binkert、A. Davis、M. McLaren 和 R. S. Schreiber，“HyperX：拓扑、路由与高效大规模网络的封装”，载于高性能计算、网络、存储与分析大会论文集，丛书 SC ’09。美国纽约州纽约市：美国计算机协会，2009 年，第 1–11 页。

[4] G. Faanes 等，《Cray Cascade：基于 Dragonfly 网络的可扩展高性能计算系统》，载于高性能计算、网络、存储与分析国际会议论文集，SC ’12 系列。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会出版社，2012年11月，第1–9页。

[5] J. Howard、J. B. Fryman 和 S. Abedin，《首个直接网格到网格光子互连结构》，*IEEE Micro*，第1–7页，2024年。

[6] J. Kim、J. Balfour 和 W. Dally，“片上网络的扁平化蝴蝶拓扑”，计算机体系结构快报 r 体系结构快报，第 6 卷，第 2 期，第 37–40 页，2007 年 2 月。

[7] S. Atchley等，“Frontier：探索首台百亿亿次级超级计算机的系统架构”，载于*SC23*：高性能计算、网络、存储与分析国际会议。IEEE，2023 年，第 1–16 页。

[8] S. H. Bokhari，“iPSC-860 上的完全交换”，*NASA* 承包商报告 *187498*，*ICASE* 报告第 *91-4* 号，1991 年。

[9] S. R. Seidel 和 T. Schmiermund，“Intel iPSC/2 的通信模型”，技术报告 *CS-TR9002*，计算机科学系，密歇根理工大学，1990 年。