[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 15/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 98809446.0

[45] 授权公告日 2007 年 7 月 11 日

[11] 授权公告号 CN 1326061C

[22] 申请日 1998.10.9 [21] 申请号 98809446.0

[30] 优先权

[32] 1997.10.10 [33] US [31] 08/949,122

[86] 国际申请 PCT/US1998/021478 1998.10.9

[87] 国际公布 WO1999/019807 英 1999.4.22

[85] 进入国家阶段日期 2000.3.23

[73] 专利权人 博普斯公司

地址 美国北卡罗莱纳州

[72] 发明人 杰拉尔德·G·帕查尼克

尼克斯·P·皮特西尼斯

埃德温·F·巴瑞

托马斯·L·德拉本斯托塔

[56] 参考文献

US 5566342A 1996.10.15

US 5474856A 1995.12.12

审查员 朱世菡

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所

代理人 于 静

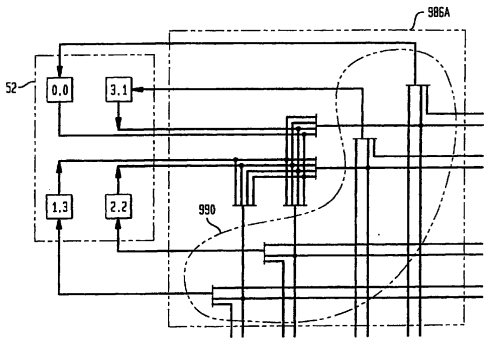
权利要求书 7 页 说明书 37 页 附图 45 页

[54] 发明名称

流形阵列处理方法和装置

[57] 摘要

流行阵列拓扑结构包括排列在一些簇(52)内的一系列处理元、节点、或存储器之类。 这些簇由簇开关机构(986A)连接，这个机构有利地可使组织方式加以改变而不需要重新在物理上排列各处理元。这种拓扑结构还显著地减少了现有阵列典型的互连数。 采用这种拓扑结构可以获得快速、高效、低耗的有效处理和通信，而且还具有配置规模改变方便的优点。



1. 一种在一个 N×M 阵列内连接的多个处理元(PE)的互连系统，其中 N 和 M 都大于 1，每个 PE 都具有一个发送和接收数据和命令的通信口，这些 PE 组合成一些簇，所述互连系统包括:

一些 PE 间的连接通路；以及

一个与所述 PE 连接的簇开关，用来合并簇间互斥的 PE 间连接通路，从而大大减少提供与传统环接 PE 阵列所提供的等效的 PE 间连通性所需的通信通路数，所述簇开关还包括一些在转置 PE 对之间和超立方互补 PE 对之间提供直接通信的连接。

2. 权利要求 1 的互连系统，其中所述数据和命令可以在所述通信口以下列 8 种选定模式之一发送和接收:

a) 东向发送/西向接收模式: 通过通信口向一个东方 PE 发送数据而通过通信口接收一个西方 PE 发来的数据;

b) 北向发送/南向接收模式: 通过通信口向一个北方 PE 发送数据而通过通信口接收一个南方 PE 发来的数据;

c) 南向发送/北向接收模式: 通过通信口向一个南方 PE 发送数据而通过通信口接收一个北方 PE 发来的数据;

d) 西向发送/东向接收模式: 通过通信口向一个西方 PE 发送数据而通过通信口接收一个东方 PE 发来的数据;

e) 转置发送/接收模式: 在转置 PE 对之间发送和接收数据，

超立方体发送/接收模式: 在距离为 1 的超立方 PE 对之间发送和接收数据;

f) 超立方体发送/接收模式: 在选择距离为 2 的超立方 PE 对之间发送和接收数据；以及

g) 超立方互补发送/接收模式: 在距离为 d 的 d 维超立方互补 PE 对之间发送和接收数据。

3. 权利要求 2 的互连系统，其中所述模式允许在所述 PE 之间建立直接连接通路。

4. 权利要求 3 的互连系统，所述互连系统还包括一个同时向每个 PE 控制口发送控制信息和向每个数据口发送需装入每个 PE 的寄存器的数据的控制器和存储器系统。

5. 权利要求 3 的互连系统，其中所述通信口每个都包括 B 比特宽的发送和接收通信，其中 B 是一个大于或等于 1 的整数。

6. 权利要求 3 的互连系统，其中所述 PE 每个都连接成根据通过一个控制口接收到的经配置在各自 PE 内的控制逻辑解码后的通信指令有选择地通过一个通信口发送命令或数据而通过另一个通信口接收命令或数据。

7. 权利要求 6 的互连系统，其中所述通信指令由控制逻辑通过所述控制口从一个控制器接收。

8. 权利要求 6 的互连系统，其中所述簇开关支持所述 PE 同时各都发送命令或数据又接收命令或数据的这种操作。

9. 权利要求 8 的互连系统，其中所述同时操作是有选择地转接成使所述 PE 同时各都通过所述通信口的发送部分发送命令或数据而通过所述通信口的接收部分接收命令或数据。

10. 一种并行处理器，所述并行处理器包括:

多个处理元(PE)，每个 PE 都有单个 PE 间通信口；以及

一些 PE 间通信通路，连接成提供与传统的环接阵列所提供的等效的 PE 间连通性，还提供转置 PE 对间的直接通信、超立方距离为 1 的 PE 对间的直接通信、选择距离为 2 的 PE 对间的通信和超立方互补 PE 对间的通信。

11. 一种并行处理器，所述并行处理器包括:

N 个各有 M 个处理元的簇，每个处理元都有一个通信口，所述处理元通过各自的通信口在总共有 B 根的线上发送和接收数据;

一些少于或等于 M×B 线宽的通信通路，连接在所述簇的各对之间，在簇对内的每个簇成员含有是在这簇对的另一个簇内的处理元的相邻元的处理元，每条通路允许所述簇对之间在两个互斥的环方向，即南和东或南和西或北和东或北和西，进行通信；以及

一些多路复用器，连接成将所述簇对间的 2M×B 线宽通信合并所述少于或等于 M×B 线宽的通路。

12. 权利要求 11 的并行处理器，其中所述每个簇的处理元在北和西环方向上与一个簇通信，而在南和西环方向上与另一个簇通信。

13. 权利要求 11 的并行处理器，其中所述每个簇的处理元在北和东环方向上与一个簇通信，而在南和西环方向上与另一个簇通信。

14. 权利要求 11 的并行处理器，其中所述每个簇的处理元在两个超立方体方向上与一个簇通信，而在另两个超立方体方向上与另一个簇通信。

15. 权利要求 11 的并行处理器，其中至少有一个簇包括一个超立方互补对。

16. 权利要求 11 的并行处理器，其中有一个簇开关包括了所述多路复用器，所述簇开关连接成将从两个互斥环方向接收到的通信多路传输给一个簇内的处理元。

17. 权利要求 16 的并行处理器，其中所述簇开关连接成将来自一个簇内的处理元的通信多路传输给另一个簇。

18. 权利要求 17 的并行处理器，其中所述簇开关连接成多路传输一个簇内的处理元之间的通信。

19. 权利要求 11 的并行处理器，其中所述 N 大于或等于 M。

20. 权利要求 11 的并行处理器，其中所述 N 小于 M。

21. 一种并行处理器，所述并行处理器包括:

N 个各有 M 个处理元的簇，每个处理元都有一个通信口，所述处理元通过各自的通信口在总共有 B 根的线上发送和接收数据，在一个簇内的每个处理元形成与簇内其他处理元比与簇外的处理元更为物理邻近的关系;

一些少于或等于 M×B 线宽的通信通路，连接在所述簇的各对之间，在簇对内的每个簇成员含有是在这簇对的另一个簇内的处理元的超立方相邻元的处理元，每条通路允许所述簇对之间在两个互斥方向，即南和东或南和西或北和东或北和西，或者两个超立方体方向之间进行通信，以及

一些多路复用器，连接成将所述簇对间的 2M×B 线宽通信合并成所述少于或等于 M×B 线宽的通路。

22. 权利要求 21 的并行处理器，其中所述每个簇的处理元在北和西环方向上与一个簇通信，而在南和东环方向上与另一个簇通信

23. 权利要求 21 的并行处理器，其中所述为每个簇的处理元在北和东方向上与又一个簇通信。

24. 权利要求 21 的并行处理器，其中至少有一个簇包括一个超立方互补对。

25. 权利要求 21 的并行处理器，其中有一个簇开关包括了所述多路复用器，所述簇开关连接成将从两个超立方方向上接收的通信多路传输给一个簇内的处理元。

26. 权利要求 25 的并行处理器，其中所述簇的开关连接成将来自一个簇内的处理元的通信多路传输给另一个簇。

27. 权利要求 26 的并行处理器，其中所述簇开关连接成多路传输一个簇内的超立方互补处理元之间的通信。

28. 权利要求 21 的并行处理器，其中所述 N 小于或等于 M。

29. 权利要求 21 的并行处理器，其中所述 N 大于 M。

30. 权利要求 21 的并行处理器，其中所述处理元间的通信是比特串行通信，而每个处理元簇通过所述通信通路与两个其他簇通信。

31. 权利要求 21 的阵列处理器，其中所述处理元间的通信通路包括一个数据总线。

32. 权利要求 21 的并行处理器，其中所述通信通路是双向通路。

33. 权利要求 21 的并行处理器，其中所述通信通路包括一些单向信号线。

34. 权利要求 21 的并行处理器，其中 P 和 Q 分别是一个环接阵列的行数和列数，所述环接阵列具有与所述阵列相同的 PE 数，而 P 和 Q 分别等于 N 和 M。

35. 权利要求 21 的并行处理器，其中 P 和 Q 分别是一个环接阵列的行数和列数，所述环接阵列具有与所述阵列相同的 PE 数，而 P 和 Q 分别等于

M 和 N。

36. 一种并行处理器，所述并行处理器包括：

一些由

$$reshape(\prod_{q=1}^{d-1}(I_{4^{q-1}} \otimes G \otimes I_{4^{d-q-1}})vec(T), \underbrace{4,4,\cdots,4}_{d})$$

定义的一个每维长为 4 的 d 维正则环的处理元 PE 的簇；以及

一些簇开关，连接成多路复用所述簇间的 PE 间通信通路，从而提供了与一个环接阵列所提供的等效的 PE 间连通性。

37. 权利要求 36 的并行阵列处理器，其中所述簇开关还连接成提供一个簇内转置 PE 对之间的直接通信。

38. 权利要求 37 的并行处理器，其中所述簇可通过组合而保持多路复用或所述簇开关形成不同规模的簇。

39. 权利要求 38 的并行处理器，其中所述簇开关还连接成提供一个簇内的超立方互补 PE 对之间的直接通信。

40. 一种形成并行处理器的方法，所述方法包括下列步骤：

将处理元排列成 N 个各有 M 个处理元的簇，满足每个簇由

$$\text{reshape}(G_N \text{ vec}(T),\ N,\ N)$$

给定，使得这些簇各只在互斥方向上与至少一个其他簇的处理元通信；以及

多路传输所述互斥方向的通信。

41. 一种在一个 N × M 阵列内连接的多个处理元(PE)的互连系统，其中 N 和 M 都大于 1，每个 PE 都具有一个发送和接收数据和命令的通信口，这些 PE 组合成一些簇，所述互连系统包括：

一些 PE 间的连接通路；以及

一个与所述 PE 连接的簇开关，用来合并簇间互斥的 PE 间连接通路，从而大大减少提供与传统环接 PE 阵列所提供的等效的 PE 间连通

性所需的通信通路数，所述簇开关还包括一些完全连接每个簇内的 PE 的连接，以提供转置 PE 对之间、超立方互补 PE 对之间和每个簇内任何 PE 对之间的直接通信。

42. 一种并行处理器，所述并行处理器包括：

多个排列成簇的处理元(PE)；

一些连接所述 PE 的 PE 间通信通路，使得在簇内和在一个簇的一个第一 PE 与两个与含有所述第一 PE 的簇相邻的簇之一的一个第二 PE 之间可通过可用通信通路实现单步的 PE 对通信。

43. 权利要求 42 的并行处理器，其中同一簇内的所有 PE 完全连接。

44. 权利要求 43 的并行处理器，其中所述 PE 都具有发送和接收口，在一个第一簇内的任何 PE 与在一个相邻的第二簇内的任何 PE 之间的通信对于在所述第一和第二簇内的所有 PE 都能同时进行。

45. 权利要求 42 的并行处理器，其中在一个第一簇内的任何 PE 能向在一个相邻的第二簇内的任何 PE 发送，而所述在第二簇内的 PE 能向在一个相邻的第三簇内的任何 PE 发送。

46. 一种将多个节点连接成一个 N×M 阵列的互连系统，其中 N 和 M 都大于 1，每个节点都有一个发送和接收数据和命令的通信口，而这些节点组合成一些簇，所述互连系统包括：

一些节点间连接通路；以及

一个与所述节点连接的簇开关，用来合并簇间互斥的 PE 间连接通路，大大减少提供在各簇之间的节点间连接通路所需的通信通路数，从而大大减少提供与传统环接节点阵列所提供的等效的节点间连通性所需的通信通路数，所述簇开关还包括一些在转置节点对之间和超立方互补节点对之间提供直接通信的连接。

47. 权利要求 46 的互连系统，其中所述节点都具有格雷编码地址，排列在一些簇内，使得每个相邻的节点的地址只有一个比特是有差别的。

48. 一种将多个存储元连接成一个 N×M 阵列以形成一个平铺式

存储系统的互连系统，其中N和M都大于1，每个存储器都有一个发送和接收数据和命令的通信口，而这存储器组合成一些簇，所述互连系统包括：

一些存储器间连接通路；以及

一个与所述存储器连接的簇开关，用来合并簇间互斥的存储器间连接通路，大大减少提供与传统环接存储器阵列所提供的等效的存储器间连通性所需的通信通路数，所述簇开关还包括一些在转置存储器对之间和超立方互补存储器对之间提供直接通信的连接。

流形阵列处理方法和装置

本发明属数据处理技术领域，具体地说与改善并行数据处理体系结构有关。

业已开发了许多计算任务对数据进行并行操作。并行处理器的效率取决于并行处理器的体系结构、编码算法和数据在各并行元内的配置。例如，图像处理、模式识别和计算机图形学都是对实质上排列成二维或三维栅状点阵的数据进行操作的应用。数据可以表示各种各样的信号，例如音频、视频、声纳或雷达信号。由于诸如离散余弦变换(DCT)、离散余弦逆变换(IDCT)、卷积之类的通常是对这种数据执行的操作可能同时要对不同的栅段进行，因此开发了多处理器阵列系统，允许多个处理器同时为这个任务工作，从而可以显著地加快这些操作。并行处理是大量专利的研究对象，其中包括美国专利 No. 5,065,339、5,146,543、5,146,420、5,148,515、5,577,262、5,546,336 和 5,542,026，这些专利列作本发明的参考。

并行处理体系结构的一种传统的解决途径是邻元网接计算机(nearest neighbor mesh connected computer)，可参见R. Cypher和J.L.C. Sanz的“图像处理和计算机视觉的SIMD体系结构和算法”(“SIMD Architectures and Algorithms for Image Processing and Computor Vision,” IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No.12, pp.2158-2174; December 1989)、K.E. Batcher的“大规模并行处理器的设计”(“Design of a Massively Parallel Processor,” IEEE Transaction on Computer, Vol. C-29 No.9, pp.836-840, September 1980)和 L. Uhr的“人工智能的多计算机体系结构”(“Multi-Computer Architectures for Artificial Intelligence,” New York, N.Y., John Wiley & Sons, Ch.8, p.97, 1987)，所有这些都列作本发明的参考。

在图 1A 所示的邻元环接计算机(nearst neighbor torus connected

computer)中，多个处理元(PE)通过环接通路 MP 与各自的北、南、东西的相邻 PE 连接，所有的 PE 都以同步的单指令多数据(SIMD)方式进行操作。由于环接计算机可以通过在一个网接计算机上再增添一些环回连接来形成，因此网接计算机可以想像为环接计算机的一个子集。如图 2B 所示，每个通路 MP 可以包括 T 根发送线和 R 根接收线，或者如图 1C 所示，每个通路 MP 可以包括 B 根双向线。虽然本发明考虑了单向和双向通信，但下面在一个通路内除控制信号线外总线的线的总数统标为 K，如果是双向总线结构，K=B；而如果是单向总线结构，K=T+R。假设一个 PE 能向它相邻的任何 PE 发送数据，但一次只能向一个相邻的 PE 发送。例如，每个 PE 可以在一个通信周期内都向它的东邻元发送数据。还假设存在这样的一种广播机制，使得数据和指令能在一个广播分发周期内从一个控制器同时分发给所有的 PE。

虽然通常采用比特串行的 PE 间通信来尽量减小布线的复杂性，但是环接阵列的复杂布线在实现时始终还是有些问题。图 1A 所示的传统环接阵列包括 16 个处理元，连接成一个 4×4 的 PE 阵列 10。每个处理元 PE 用各自所在的行和列的序号 i 和 j 作为下标，标成 $PE_{i,j}$。每个 PE 通过点对点连接与各自相邻的北(N)、南(S)、东(E)、西(W)邻元通信。例如，图 1A 中所示的 $PE_{0,0}$ 与 $PE_{3,0}$ 之间的连接是 $PE_{0,0}$ 的 N 接口与 $PE_{3,0}$ 的 S 接口之间的环回连接，表示将这个阵列配置为环接阵列的环回连接之一。在这种配置中，每行含有一组 N 个相互连接，N 行共有 N^2 个水平连接。同样，N 个各有 N 个垂直相互连接的列共有 N^2 个垂直相互连接。对于图 1A 所示的例子，N=4。包括环回连接在内，连接线(例如 N×N 环接计算机集成电路内的金属化线)的总数因此为 $2kN^2$，其中 k 为每个相互连接中的导线数。导线数 k 可以为 1，例如在比特串行互连中。如果 k=1，对于图 1A 所示的 4×4 阵列 10 有 $2kN^2=32$。

对于 N 比较小的一些应用来说，最好将整个 PE 阵列集成在单个集成电路内。然而，本发明并不排除例如每个 PE 可以是一个独立的

微处理器芯片的实现方式。由于一个环接计算机内的连线总数可能是非常多的，因此这些内部互连将耗费大量可贵的集成电路“不动产”或芯片占用面积。此外，这些PE通路频频相互交叉，使IC布置非常复杂，而且还可能由于串音而将噪声引入通信线路。还有，连接处在阵列南、北两端的PE或处在东、西两端的PE的环回连线的长度随阵列规模的增大而增大。长度增大就使连线的电容增大，从而使连线传输的最大比特率下降，还会将附加噪声引入连线。

这样的环形阵列的另一个缺点与转置操作环境有关。由于一个处理元与它的转置元在通信通路内至少相隔一个中介处理元，因此在要用到转置的操作中会引入等待。例如，在$PE_{2,1}$需要从它的转置元$PE_{1,2}$获取数据时，数据必需经过中介元$PE_{1,1}$或PE2,2。实际上，这本身就会使操作延迟，即使是$PE_{1,1}$和$PE_{2,2}$没有被其他占用。然而，在一般情况下，这些PE都是作为微处理器元，因此很可能$PE_{1,1}$和$PE_{2,2}$当时将在执行其他操作。为了将数据或命令从$PE_{1,2}$传送给$PE_{2,1}$，它们必需以通常的方式暂停执行这些操作和命令。因此，即使开始将数据从$PE_{1,2}$传送给$PE_{1,1}$可能需要采取几个操作，而迫使$PE_{1,1}$暂停来传送转置元数据的操作也将引起延迟。这样的延迟每经一个中介元PE都要累加上去，因此对于相距最远的转置元对来说，引起的等待时间是很显著的。例如，图1A的转置元对$PE_{3,1}$与$PE_{1,3}$之间至少有三个中介元PE，需要的等待时间是四个通信步骤，再加上通常情况下在这些PE内为了在$PE_{3,1}$与$PE_{1,3}$之间传送数据必需暂停所有任务而引起的等待时间。

认识到环接阵列的这些缺点，已经提出了一些新的构成阵列的途径，可参见G.G. Pechanek等人的“大规模并行对角折叠阵列处理器”(“A Massively Parallel Diagonal Fold Array Processor”，1993 International Conference on Application Specific Array Processors, pp.140-143, October 25-27, 1993, Venice, Italy)和G.G. Peckanek等人的“多重折叠成簇处理器环接阵列”(“Multiple Fold Clustered Processor Torus Array”, Proceedings Fifth NASA Symposium on VLSI Design,

pp.8.4.1-11, November 4-5, 1993, University of New Mexico, Albuguergue, New Mexico)，这两篇论文的全部内容都列作本发明的参考。这些环接阵列组织的构成技术是用处在传统的相邻元环面对角线上的 PE 作为对折边折叠 PE 阵列。如图 2A 的阵列 20 所示，这些技术可以用来大大减少 PE 间的连线，减少环回连接线的数目和减小环回连接线的长度，而且将各个 PE 配置在它们的转置元的紧邻处。这种处理器阵列的体系结构例如可参见美国专利 No.5,577,262 和 5,612,908 以及 EP 0,726,532 和 EP 0,726,529，这些专利全部列作本发明的参考。虽然这些阵列与传统的环接体系统构相比有了很大的改进，然而由于 PE 组合的不规则，例如在单折的对角折叠网中，有些 PE 两两成簇，而另一些却仍是单个的。在一个三折的对角折叠网中，有四个 PE 的簇，也有八个 PE 的簇。由于阵列总体呈三角形，对角折叠型阵列在以集成电路实现上无论效率和成本都有相当困难。此外，在对角折叠网和其他传统的网状体系结构中，互连拓扑是 PE 定义的固有部分。这种解决途径固定了 PE 在拓扑结构内的位置，从而将 PE 的拓扑和它们的连通性限制在所实现的固定配置。

许多并行数据处理系统采用超立方体互连拓扑结构。一个超立方体计算机包括 $P=2^d$ 个 PE，它们以提供高度连通性的方式互连。连接可以在几何上或算法上模型化。在几何模型中，各个 PE 相应于一个 d 维超立方体的各个顶点，而各链路相应于这个超立方体的各个边缘。一个具有 $P=2^d$ 个 PE 的超立方体可以想像成两个各具有 2^{d-1} 个 PE 的超立方体，这两个较低维的超立方体相应顶点之间由链路连接。

在算法模型中，为各 PE 分别指定从 0 至 d-1 中的一个互不相同的二进制附标。任何两个 PE 只有它们的附标的二进制表示只有一位不同才是互连的。几何模型与算法模型可以通过将 d 维与 d 个比特位置一一对应联系起来。于是，两个附标只有一个比特不同的性质相当于占据两个(d-1)维超立方体的相应的角。例如，可以为一个 PE 指定一个附标，指示它在这个拓扑结构内的位置。这个附标$\{D_0, D_1, \cdots, D_{r-1}\}$是一个二进制表示，每个比特指出了 r 维超立方体上通信可用的一条 r

维连接通路。在超立方体中的每个节点与它直接连接的那些节点之间附标最多只有一个比特有差别。例如，在超立方体中最长的通路是一个 PE $\{D_0, D_1, \cdots, D_{r-1}\}$与它的补元$\{\neg D_0, \neg D_1, \cdots, \neg D_{r-1}\}$之间的距离，例如 PE101101 与 PE010010 之间的距离。超立方拓扑结构可参见 Robert Cyher 和 L.C. Sanz 的“并行运算的 SIMD 模型”(“The SIMD Model of Parall Computation” 1994 Springer-Verlag, New York, pp.61-68)和 F. Thomas Leighton 的“并行算法和体系结构导论：阵列、树、超立方体”(“Introduction To Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”，1992 Morgan Kaufman Publishers, Inc., San Mateo, CA, pp 389-404)，这两篇著作都列作本发明的参考。超立方拓扑结构的一个缺点是每个处理器的连接数随网络的规模对数增大。此外，一个超立方体内的 PE 间通信可能由于相当长的等待，特别是在 PE 是一对相互互补的元的情况下，成为负担。

多维超立方体可以映射成一个环(torus)、一个对角折叠环或其他 PE 结构。下面简要地说明一下这种映射。除非另有说明，与这说明有关的图以及本申请内所有其他的图都将每个 PE 连接示为一条单线，这条线表示一个互连链路，可以是一个双向三态链路，也可以是两个单向链路。双向三态链路支持在一个链路的多个点上有信号源，在一种控制程式下防止在链路上发生数据冲突。单向链路对于任何对接信号都采用点对点的单个信号源、单个接收机对。此外，可以预料，用比特串行和多比特并行实现也是可以的。

一个超立方体可以映射成一个环，其中二维环由一系列处理元(PE)构成，如图 1A 和 1D 所示，每个 PE 对应一个环节点(行和列)如上 PE 附标所示，和一个由每个 PE 内的下附标所示的超立方 PE 号码。超立方 PE 号码或节点地址作为一个 r 组(rD)超立方体的 r 比特表示给定，每个比特表示一个连通性的维。一个超立方体内的每个 PE 都只是与节点地址号码与它的只有一个比特是不同的那些 PE 邻接。这种互连的方式使得一个 4D 超立方体可以映射成一个如图 1A 和 1D 所示的 4 × 4 环。图 1A 用格雷码将 $PE_{i,j}$ 节点编码成 $PE_{G(i),G(j)}$，这是相继的号码

之间只有一个比特有不同的序列。例如，十进制序列 0，1，2，3 可以写成二进制序列 00，01，10，11，而格雷码序列就是 00，01，11，10。图 1D 示出了映射成一个最近邻环的另一个超立方体。

超立方机的最早实现之一是一个 6D 超立方体的宇宙立方体(Cosmic Cube)，可参见 Caltech, C. Seitz“宇宙立方体”(“The Cosmic Cube”, Communications of the ACM, Vol. 28, No.1, pp.22-33, 1985)。这个宇宙立方体是用一些以多指令多数据(MIMD)模式运行的 Intel 8086 处理器实现的，利用消息进行超立方连接的处理器之间的通信。另一个超立方体实现例是 NCUBE，在一个大规模的配置中包括了一个 10D 超立方体，采用定制的处理器芯片形成这个超立方体的节点。NCUBE 是一个 MIMD 型机器，但也支持单程序多数据(SPMD)工作模式，每个节点处理器都有一个相同程序的拷贝，因此能独立处理不同的条件代码流。由思维机公司(Thinking Machines Corporation 建造的连接机(CM)是另一个超立方体实现例。初型机 CM-1 是一个 12D 超立方体，每个节点包括一个由比特串行处理元构成的 4×4 网格。

诸如上述这些传统的超立方体实现的一个缺点是每个处理元必需为每个超立方体至少各备一个双向数据口。

如下面要详细说明的那样，本发明的一个特点就是我们的 PE 是与网络的拓扑结构无关的，只需要一个输入口和一个输出口。

此外，由于超立方体每增加一维，每个 PE 所需的口数就要加多，因此每个 PE 的设计马上就成为沉重的负担，每个 PE 花在数据口上的比例越来越大。而且，两个互补 PE 之间的通信由于随着一个超立方体的“直径”(即互补PE对之间的通信步数)的增大等待时间越来越大而成为累赘。也就是说，提供一个节点与它的补节点之间的连接，也就是超立方体 PE 节点之间最长通路的连接，会是困难的，耗费也大，而且不一定会是可行的。

因此，极希望能在并行处理器阵列内各处理元之间提供高度的连通性，同时大大减少互连处理元所需的布线和大大减少 PE 间通信所受到的通信等待时间。对于多处理器阵列体系结构和处理器互连方面

也有必要进一步加以改善。本发明就是针对这些和其他一些这样的需要提出的，详细说明如下。

本发明研究的是一种处理元阵列，这种处理元阵列能改善处理元之间的连通性，同时与传统的环或超立方体处理元阵列相比又大大降低了阵列互连布线的要求。在一个优选实施例中，按照本发明设计的阵列能大大减少转置操作的等待和一个 PE 节点与它的超立方补节点之间通信的等待。此外，这种独创性的阵列使环回布线的长度与阵列的总体规模无关，从而减小了最长互连线的长度。而且，对于不会正在通信的 PE 之间引起冲突的阵列通信模式来说，每个 PE 只需要一个发送口和一个接收口，而与具体拓扑结构要求它的 PE 节点需具备的邻元连接数无关。这种阵列的一个优选集成电路实现方式包括将相似的处理元簇合并成呈现具有矩形或正方形的外形。这种处理元的相似性、处理元簇的相似性和阵列外形的规则性使得这种阵列特别适合低成本集成电路生产。

为了按照本发明形成一个阵列，首先可以将处理元合并成合乎单指令多数据(SIMD)操作要求的簇。然后，将簇内的处理元完全连接起来。于是，可以将各处理元组合成使得一个簇的元可以在簇内通信和与仅两个其他簇的成员通信。此外，每个簇的组成处理元只是在两个互斥方向上与这两个其他簇的每个簇的处理元通信。按定义，在一个具有单向能力的 SIMD 环内，北/南方向是与东/西方向互斥的。正如名称所含的意义那样，处理元簇是最好实际相互紧邻的一些处理元形成的组。在集成电路实现中，例如，一个簇的处理元最好布置成尽量相互靠近，最好比与阵列中任何其他处理元靠得更近一些。例如，一个与传统的 4×4 处理元的环阵列相应的阵列可以包括四个各有四个元的簇，每个簇只在北和东的方向上与一个其他簇通信而在南和西的方向上与另一个簇通信，或者在南和东的方向上与一个其他簇通信而在北和西的方向上与另一个簇通信。通过以这种方式将 PE 组合成簇，PE 簇间的通信通路可以通过多路复用共享，因此大大减少了阵列所需的互连布线。

在一个优选实施例中，组成一个簇的这些处理元选择成使得处理元与它们的转置元和超立方补元处在同一个簇内，通过簇内通信通路相互通信，从而消除了与在传统的环阵列上执行转置操作有关的等待和与在传统的超立方体阵列上执行超立方互补时间通信有关的等待。此外，由于将传统的环回通路与任何 PE 至 PE 的通路同样处理，就可以将最长的通信通路缩短成簇间的间隔，而与阵列的总规模无关。

每个 PE 含有一个虚拟 PE 地址存储单元和一个配置控制器。虚拟 PE 号码与配置控制信息的组合确定了簇开关的设置，从而确定了对 PE 阵列的拓扑结构的重新配置。这种重新配置例如可以根据来自控制器的一个分发指令进行。一个阵列内的 PE 组合成簇，使得一个 PE 与它的转置元组合在一个簇内，一个 PE 与它的超立方补元包含在同一簇内。此外，动态重新配置在保证每个簇内的完全 PE 间连通性的簇开关的配合下提供了可将阵列重新配置成许多拓扑结构的能力。

在另一方面，一个簇内的各个 PE 有益地可以具有相同的与簇开关连接的接口，这个簇开关完全连接这个簇内的 PE，允许这个簇内的每个虚拟 PE 同样可以访问两个外部的正交簇。这样，按照本发明，真正有两个网络与一个簇开关配合。一个完全使簇内 PE 相互连接，一个使 PE 与其他簇的 PE 连接，从而提供了环和超立方体的连通性所需的连接通路。对于簇开关是内部的连接通路提供转置元对和超立方互补元对的连通性。采用不同的虚拟 PE 排列，可以横越这些簇实施转置。对于这样一个 4PE 簇开关和它与其他 4PE 簇的互连来说，可以只有四个为任何簇产生的四个输出总线。在任何簇内，这四个总线的每一个都具有两个正交簇连接点。在按照本发明的流形阵列处理中，可以提供连通性增强的超立方体，其中每个有四个节点的簇只有四个输出总线，每个扇出为三，一个内部接回本簇，另两个分别接至两个正交簇。从接收来看，每个虚拟节点有三个信号要接收，一个来自本簇内部，另两个分别来自两个正交簇。

本发明的这些和其他一些特性、情况和优点对于熟悉本技术领域的人员来说从下面结合附图所作的详细说明中可以清楚地看到。

在这些附图中:

图 1A 为现有技术的 4×4 邻接环处理元(PE)阵列的方框图;

图 1B 例示了图 1A 所示现有技术环连接通路可以包括 T 根发送线和 R 根接收线的情况;

图 1C 例示了图 1A 所示现有技术环连接通路可以包括 B 根双向线的情况;

图 1D 为第二个现有技术的 4×4 邻接环 PE 阵列的方框图;

图 2A 和 2B 为现有技术的折迭 PE 阵列的方框图;

图 3A 至 3C 为适合在本发明的 PE 阵列内使用的处理元的方框图;

图 3D 例示了本发明的簇开关控制的详细情况;

图 4 为例示在一个流形阵列内 PE 成簇情况和 PE 的簇间通信情况的方框图;

图 5 为图 4 中的一个 PE 簇的方框图,详细示出了簇开关的情况;

图 6 为本发明中的改善的 PE 簇和簇开关的方框图;

图 7 为详细示出 PE 簇间互连通路的方框图;

图 8A 为示出一个不包括缓冲器的簇开关实现形式的方框图;

图 8B 和 8C 为详细示出利用不包括缓冲器的簇开关的 PE 簇间互连通路的方框图;

图 9A 为一个 2×2 簇的方框图,示出了与组成一个大阵列的其他正交 2×2 簇的互连通路;

图 9B 为一个 2×2 流形阵列的方框图;

图 10 为示出一个 4×4 环的东向通信通路的方框图;

图 11 为示出一个 4×4 环的西向通信通路的方框图;

图 12 为示出一个 4×4 环的北向通信通路的方框图;

图 13 为示出一个 4×4 环的南向通信通路的方框图;

图 14 为示出一个 4×4 流形阵列的转置通信通路的方框图;

图 15 为示出一个 4×4 流形阵列上的四个独立的 1×4 线形环的方框图;

图 16 为示出在一个阵列配置内 z 轴发送操作的通信通路的方框图;

图 17 为示出一个 4×4 流形阵列的超立方体节点标记情况的方框图;

图 18 为示出一个 4×4 流形阵列的超立方互补元通信的方框图;

图 19 为示出一个 5D 超立方体的方框图;

图 20 为示出映射成一个流形阵列的 5D 超立方体的方框图;

图 21A 为示出嵌入一个环的 4D 超立方体的节点元的表;

图 21B 为示出嵌入的超立方体节点的布置的改善流形阵列表;

图 21C 为一个带有上 PE 附标(PE-x,y)的 8×8 2D 环、一个带有中 PE 附标(x,y,z)的 3D 立方体和一个带有下 PE 附标 $G_xG_yG_z=d_5d_4d_3d_2d_1d_0$ 的 6D 超立方体的 4×4×4 表示;

图 22 为列 1D 下转后的 4×4×4 表示;

图 23 为图 22 中的节点的 4×4×4 z 平面表示;

图 24 为列 1D 下转后的 4×4×4 z 平面表示;

图 25 为布置连通性的对 z 平面表示进行重排的情况;

图 26 示出了将各 z 平面分离为一些 2×2 z 簇的情况;

图 27A 示出了将一些 2×2 z 簇互连成四个 4×4 流形阵列的情况;

图 27B 示出了一个 4×4×4 PE 节点输入(接收)连通性的例子;

图 28A 为一个 4×4 流行阵列方框图,示出了有一个外部接口和每个簇配置一个控制器的情况;

图 28B 示出了图 28A 的 4×4 多控制器流形阵列接收送入它的外部接口的 32 个数据项的情况;

图 29 示出了图 28A 的 4×4 多控制器流形阵列将例示的 32 个数据项装入四个存储器控制器的情况;

图 30 示出了图 28A 的 4×4 多控制器流形阵列将 32 个数据项分配给各簇内的各个 PE 的情况;

图 31 列出了结合图 28A 至 30 所示例每执行一个完全混洗操作后所得出的 32 个例示数据项的分布情况;

图 32 示出了一个 4×4 流形阵列在执行一个北向交换操作时在各 PE 之间数据所取的通路，还示出了在这个完全混洗例的通信操作完成时各 PE 寄存器内的结果；以及

图 33 示出了一个 4×4 流形阵列在执行一个南向交换操作时在各 PE 之间数据所取的通路，还示出了在这个完全混洗例的通信操作完成时各 PE 寄存器内的结果。

在一个实施例中，按本发明构成的流形（manifold）阵列处理器将 PE 组合成簇，使得一个簇的元只能与两个其他簇的成员直接通信，而且每个簇的各构成处理元只能在两个互斥方向上与可与之通信的每个其他簇的处理元通信。以这种方式使 PE 成簇，使得 PE 簇之间的通信通路可以共享，从而大大减少了构成一个阵列所需的互连布线。此外，每个 PE 可以只有一个发送口和一个接收口，或者在采用双向、顺序或时分通信的情况下可以只有一个收/发口。这样，各个 PE 就与阵列的体系结构解耦。也就是说，不像传统的 N 维超立方连接阵列那样每个 PE 要有 N 个通信口。在采用单发送口和单接收口的实施方式中，阵列内的所有 PE 可以同时发送和接收。在一个传统的 6 维超立方体内，这要求每个 PE 有 6 个发送口和 6 个接收口，总共 12 个数据口。采用本发明，无论超立方体是几维的，都只需要一个发送口和一个接收口，总共两个数据口。如上所述，发送和接收数据口可以合并成一个收/发口，如果采用双向、顺序或时分数据通信的话。每个 PE 有一个虚拟 PE 存储单元和一个配置控制单元。虚拟 PE 号码与配置控制信息组合起来可以确定簇开关的设置，控制通信的方向和重新配置 PE 阵列的拓扑结构。这种重新配置可以例如根据控制器分发的指令进行。阵列内的 PE 组合成簇，使得一个 PE 与它的转置元合并在一个簇内，而一个 PE 与它的超立方补元包含在同一个簇内。

在以下这个优选实施例中，组成一个簇的 PE 选择成使处理元和它们的转置元处在同一个簇内，通过簇内通信通路相互通信。为了说明方便起见，处理元标成像在传统的环接阵列内呈现的那样，例如，处理元 $PE_{0,0}$ 是会在一个传统的环接阵列的“西北”角出现的处理元，

也就是说在行 0、列 0 处的处理元。因此，虽然新的簇阵列的布置与传统的阵列处理器的大不相同，但相同的数据会送至传统的环接阵列和新的簇阵列的相应处理元。例如，新的簇阵列的 0，0 元会接收与一个传统的环接阵列的 0，0 元要加以操作的相同的数据。此外，在本说明中所指的方向将参照一个环接阵列的方向。例如，在说到处理元之间的通信从北向南发生时，那些方向是指在一个传统的环接阵列内的方向。

PE 可以是单指令流单数据流(SISD)型的单处理器芯片。虽然并不局限于以下说明，但将对一个基本的 PE 进行说明，以例示所涉及的这些概念。图 3A 示出了一个 PE40 的基本结构，用来说明本发明的新 PE 阵列的各 PE 可采用的一个适当实施例。为了简明起见，接口逻辑和缓存器都未示出。指令总线 31 接成接收从一个 SIMD 控制器 29 分发的指令，而数据总线 32 接成接收从存储器 33 或 PE40 外的另一个数据源发来的数据。寄存器文件存储媒体 34 为执行单元 36 提供源操作数的数据。指令解码器/控制器 38 接成接收通过指令总线 31 送来的指令和通过总线 21 向寄存器文件 34 内的寄存器提供控制信号。文件 34 的这些寄存器通过通路 22 向执行单元 36 提供它们的内容作为相应的操作数。执行单元 36 从指令解码器/控制器 38 接收控制信号 23 而将结果通过通路 24 提供给寄存器文件 34。指令解码器/控制器 38 还将标为 Switch Enable 的簇开关允许信号加到输出线 29 上。簇开关的功能将在下面结合图 5 和 6 详细说明。PE 间通信的数据或命令在标为 Receive 的接收输入端 37 上接收和从标为 Send 的发送输出端 35 上发送。

虚拟 PE 存储单元 42 通过存储线 43 和检索线 45 与指令解码器/控制器 38 连接。虚拟 PE 号码可由控制器 29 通过发给解码器/控制器 38 的指令程控，解码器/控制器 38 将新的虚拟 PE 号码发给存储单元 42。虚拟 PE 号码可以由控制器 29 用来在连接网络所给定的范围内动态地控制每个 PE 在一个拓扑结构内的位置。

配置控制器 44 通过存储线 47 和检索线 49 与指令解码器/控制器

38 连接。配置控制器 44 提供诸如当前配置之类的配置信息和提供对簇开关的控制信息。这些开关控制阵列内的 PE 与其他 PE 的连接。解码器/控制器 38 组合来自配置控制器 44 的当前配置、来自虚拟 PE 存储单元 42 的虚拟 PE 地址和由来自控制器 29 的指令传达的诸如“在转置 PE 之间通信”之类的通信操作信息，并将这信息送至簇开关。解码器/控制器 38 包括利用这信息确定对簇开关的适当设置(这将结合图 6 详细说明)和通过开关激活接口 39 发送这信息的开关控制逻辑。开关控制逻辑、簇开关指令解码器/控制器和配置控制器可以合并在 PE 界限外的簇开关内。由于新的 PE 节点定义为与拓扑连接无关，因此能将这些功能分离出来。在本优选实施例中，总体逻辑和综合功能通过不将这些控制功能分离出来而得到改善，尽管这些控制功能是独立的。

例如，在图 3D 中示出了一个适当的簇开关 60，用来详细说明簇开关的控制情况。簇开关 60 示为划分成四个组 A、B、C、D，每个组包括一个 4 输入多路复用器和一个 3 输入多路复用器。这些组各与 PE 簇内的一个相应的 PE 关联，这种对应的关联关系以虚线箭头示出。例如，$PE_{0,0}$ 与“A”组多路复用器 a1 和 a2 关联。这些组内的多路复用器受它们关联的 PE 控制。通过如图所示那样控制这些多路复用器，保留了通常的 SIMD 工作模式。

在 SIMD 工作模式，所有的 PE 都接收控制器分发的指令，根据这些指令同步工作。所有的指令，包括那些唯一地规定一个取决于 PE 的 ID 的操作的指令，分发给所有的 PE。这些指令由所有的 PE 接收，由所有的 PE 或 PE 的一些子集解码和执行，这取决于指令内的操作码(可能还带有一个操作码扩展段)和在控制器 29 分发的指令的程序控制下可选的 PE 允许/禁止标志。操作码和它的扩展段规定了要执行接收到的指令的 PE 集合，包括只有一个 PE 的集合。PE 允许/禁止标志确定一个 PE 能响应的活动的级别。例如，可适当采用以下标志:

级别 0: 完全禁止 PE;

所有接收到的指令全都处理为 NOP。

级别 1: 部分允许 PE，PE 接收控制信息:

允许装载诸如虚拟 PE ID、饱和/非饱和模式之类的控制信息，

允许存储诸如读状态寄存器之类的控制信息，

所有的运算和通信指令都处理为 NOP。

级别 2: 部分允许 PE; PE 接收控制信息:

允许装载诸如虚拟 PE ID、饱和/非饱和模式之类的控制信息，

允许存储诸如读状态寄存器之类的控制信息，

所有的运算指令都处理为 NOP，

执行所有的通信指令。

级别 3: 完全允许 PE;

执行所有接收到的指令。

对于一个给定规模的流形阵列，可以预先确定允许的配置，例如选择 4 维、5 维或 6 维超立方体。在这样的一个实施例中，可能的节点标识号码可以是“硬连线”的，也就是说实际上以非易失方式固定为集成电路实现的一部分。于是，由控制器 29 发送给阵列内所有的 PE40 的一个单指令隐含地指出对于一个给定配置的各个虚拟 PE 号码。这个指令最好由每个 PE 内的解码器/控制器 38 变换成为本 PE 指定的适当虚拟 PE 号码。每个解码器/控制器 38 可以有效地执行表查找操作，由于每个 PE 存储区 42 内相应位置存有本 PE 在这种配置下的虚拟 PE 号码。

图 3B 的 PE40′ 除了与图 3A 的 PE40 中相同的器件(用相同的标号标示)外还包括一个接口控制单元 50，与指令解码器/控制器 38 和寄存器文件 34 连接。控制单元 50 根据通过信号线 25 从解码器/控制器 28 获得的控制信号提供诸如并行-串行变换、数据加密和数据格式变换之类的数据格式化操作。在图 3C 所示的另一个实施例 PE40″ 中，发送通路 35 由一个或多个执行单元 36 产生，而接收通路 37 直接或通过接口控制单元 50 间接接至寄存器文件 34。接口控制单元 50 根据通过

信号线或线25从指令解码器/控制器38接收的控制信号对数据进行格式化。由接口控制单元执行的数据格式化可以包括例如并行-串行变换、串行-并行变换、数据加密和数据格式变换。

这种可选用的PE40″还包括为每个PE40″增添的本地存储块48和52。在图3C中较详细地示出了来自图3A和3B的数据总线32，它包括一个装载通路总线26和一个存储总线26′。这两个总线都可以用三态技术或作为多单向总线实现，而且具有可变的总线宽度，例如16比特、32比特、64比特。可以适当地应用各种控制信号，诸如地址和总线协议信号。此外，总线26和26′每个都能作为两个各在控制器29和DMA单元直接控制下的总线实现。控制器装载总线例如可用来将内部控制器寄存器内容装入PE，而读总线可用来读诸如状态比特那样的内部PE寄存器内容。控制器29可以通过将控制器29接至存储器的接口线路61接入这些总线。DMA装载总线可用来将存储器的数据块从存储器33装入PE本地存储器48，而DMA读总线可用来将数据块从本地存储器48存入存储器33。DMA功能最好是控制器29的一部分。存储器开关46用来将PE本地存储器48通过总线51和53以及总线55和57接至PE寄存器文件34。存储器开关46还将存储器33通过总线26和26′以及总线55和57接至PE寄存器文件34。存储器开关控制信号由装载和存储单元28通过控制接口59提供。装载和存储单元28通过接口27从指令解码器/控制器38接收装载和存储指令信息。根据接收到的指令信息，装载和存储单元28产生对存储器开关46的开关控制信号。从控制器29分发给PE阵列的所有指令在阵列的每个PE内都以同样的方式解释。这些分发的PE指令并不标有发给个别PE或个别PE组的标记。

图4中示出了一个4×4的PE阵列。在图4的阵列中组合了四个各有四个PE的簇52、54、56和58。簇开关86和通信通路88以在1997年6月30日递交的共同未决专利申请08/885,310中详细说明的方式连接这些簇，该申请的内容全部列作本发明的参考。虽然，在图中这个优选实施例内的每个处理元示为具有两个输入口和两个输出口，但是

在簇开关内多路复用的另一个层使每个 PE 的通信口的数目减少为一个输入口和一个输出口。在一个每个 PE 有四个邻元发送连接和单向通信(即每个 PE 只允许一个发送方向)的标准环中，在每个 PE 内需要有四个多路复用的发送通路。这是由于互连拓扑结构定义为 PE 的一部分。总的结果是在这个标准环中有 $4N^2$ 个多路发送通路。在具有等效的连通性和非限制的通信的流形阵列中，只需要 $2N^2$ 个多路复用发送通路。这样减少发送通路就显著地节约了集成电路的“不动产”，因为多路复用器和 $2N^2$ 个发送通路所占用的面积明显小于 $4N^2$ 个发送通路所占用的面积。各通信通路分别标为 N、S、E、W，与环接阵列内的各通信方向相应。

一个完整的簇开关 86 的实施例示于图 5 的方框图。北(N)、南(S)、东(E)、西(W)输出如前面所示。簇开 86 已增添了多路复用的另一层 112。这个多路复用层在标为 A 的东/南接收和标为 B 的北/西接收之间进行选择，从而将对每个 PE 的通信口要求降低为一个接收口和一个发送口。此外，转置元 $PE_{1,3}$ 和 $PE_{3,1}$ 之间的多路复用连接通过标为 T 的簇内转置连接实现。在对于一个特定的多路复用器的 T 多路复用器允许信号有效时，来自一个转置 PE 的通信就在与这个多路复用器关联的 PE 被接收。

在这个优选实施例中，所有的簇都包括诸如在一个 PE 与它的转置 PE 之间的通路那样的转置通路。这些图例示了总体连接方案，但并不意图例示一个多层集成电路实现方式怎样可以完成通常会是常规的设计选择问题的常规阵列互连的实体。如同任何集成电路布局那样，IC 设计者可以在设计本发明的阵列的实际 IC 实现方式的过程中分析各种折衷方案。例如，可以将簇开关分布在 PE 簇内，以减小众多接口的布线长度。

为了将 4PE 簇内的连通性扩展到支持多维阵列和简化实现方式在多路复用上所要求的改变示于图 6 的簇开关 686。为了简化说明，假设都是单向连接，除非另有说明。图 6 中的这些多路复用器对于每个数据通路输入都有一个与之关联的允许信号。这些允许信号可以由来

自 SIMD 控制器的独立信号线产生，或者间接地在各个 PE 内将所接收的分发指令解码后产生，或者在簇开关内产生。可以在簇开关内配置独立的控制机制，接收 SIMD 控制器分发的指令加以解码，通过独立的解码器/控制器机制产生多路复用的允许信号。在本优选实施例中，簇开关多路复用允许信号在 PE 内产生。在这个优选实施例内应用了分别标为 4/1 和 3/1 的四个 4-1 发送多路复用器和四个 3-1 接收多路复用器。

图 6 示出了图 5 所示簇开关 586 的扩展，包括用四个 4 输入多路复用器 4/1 代替八个 2 输入发送多路复用器{X1, X3, X5, X7, X2, X4, X6, X8}。图 5 中带有标为{A, B}和{A, B, T}允许信号的四个 2 输入和三个 3 输入接收多路复用器用四个 3 输入接收多路复用器 3/1 代替，还用了门/缓存器 B1 至 B8 对发送线进行选通/缓冲。四个 4 输入发送多路复用器提供簇 52 内的四个 PE 之间的完备连通性。图 6 的簇开关 686 表示由图 5 的簇开关 586 提供的连通性的一个超集。有许多方式来布置簇开关的内部布线，图 6 示出了这些连接点，但并没有示出一个多层硅实现怎样完成这些连接的情况。

图 7 以方框图形式示出了在一个 4×4 流形阵列内 PE 簇 52、54、56 和 58 的组织情况。对于这些 PE 簇和它们的簇开关不在同一块硅片上、可能相隔一段距离需要用电路板线路或电缆连接的一般情况，需要对发送线进行选通/缓冲。此外，就功率和布局上的原因，为了减小噪声和功率，对发送信号进行选通/缓冲也是很重要的。然而，在这个优选实施例中，流形阵列组织在单个芯片或集成电路上，四个 PE 簇的簇开关紧挨在一起，因此可以省去选通/缓冲电路。在图 9A 的簇 986A 中示出了图 6A 的簇组织在省去了缓冲后的情况。

类似，图 7 的 4×4 流形阵列的缓冲对于优选的单芯片实现来说也可省去，结果为如图 8A 所示的流形阵列 800A。在 4×4 流形阵列 800A 连接成如图 8A 所示时，一个簇内的每个 PE 能与两个相互正交的其他簇连接。图 8B 示出了 4×4 流形阵列 800B 内的一个 PE ($PE_{1,3}$)的输出发送连通性。4×4 流形阵列 800B 内的每个 PE 都能接回本身和本簇内

的其他 PE，而且能接至两个其他簇。图 8C 示出了一个阵列 800C 的一个 4×4PE 节点的输入(接收)连接性。在本发明的 4×4 流形阵列的这个优选实施例中，任何两个节点之间最大通信距离为 2。

在图 9B 这个方框图中示出了一个 2×2 流形阵列 900B，其中包括一个连接簇 952 内各 PE 的簇开关 986B。这个图中的任何 PE 都能与簇 952 内任何其他 PE 通信。例如，PE_{00} 可以将数据发送给它本身、PE_{01}、PE_{10} 或 PE_{11}，PE_{01} 可以与 P_{00}、它本身、PE_{10} 或 PE_{11} 通信，对于簇内其他两个 PE 来说情况相同。对于转置操作，PE_{00} 和 PE_{11} 不需要做什么，最多是与本身通信，而 PE_{01} 与 PE_{10} 通信。对于超立方情况，PE_{00} 与 PE_{11} 通信，而 PE_{01} 与 PE_{10} 通信。从图 9B 的 2×2 流形阵列 900B 到 4×4 流型阵列需要附加一组多路复用器来连接四个 2×2 流型阵列，如图 9A 中对簇 52 所示。图 9A 的簇开关 986A 包括了这组附加的多路复用器 990。因此，可以看到本发明为处理器、节点之类的互连提供了高度灵活和可变规模的途径。

图 10 至 13 这些方框图分别示出了一个流形阵列 1000 的各个方向的邻元通信通路，即东、西、北、南向通信通路。每条通路用箭头指示。在给定时刻每个多路复用器内只有一条输入通路是允许的。为了使数据能在选定通路上传送，控制器 29 向所有的 PE 分发一个通信指令。这些 PE 接收到分发的 PE 指令后，对它进行解码，从各自的寄存器文件 34 中检索出所选数据发送给各自的簇开关 86。根据从接收到的指令结合得出所选通信信息结合已编程的虚拟 PE 号码和配置控制器 44 的输出产生在图 3A 中所示的开关允许信号。

在图 14 这个方框图中，示出了对同一个 4×4 流形阵列 1000 进行转置操作的通信通路。同样，有效数据通路以沿通路的有向箭头指示，一个时刻每个多路复用路只有一个输入是有效的。这些开关允许信号以与对图 10 至 13 所述相同的方式形成。图 15 示出在一个 4×4 流形阵列 1500 上的四个独立的 1×4 线性环的通信通路。这些 1×4 线性环都由 2×2 阵列以行为主序构成的。也就是说行为主序的线性环通路是从 PE_{00} (A，B，C 或 D)经 PE_{01}(A，B，C 或 D)、PE_{10}(A，B，C 或 D)、

PE_{11}(A，B，C 或 D)再回到 PE_{00}(A，B，C 或 D)。每个由 PE 00、01、10、11 组成的组(A、B、C 或 D)构成了一个 1×4 的 PE 线性环。

图 16 这个方框图示出了本发明的流形阵列的灵活性的又一情况。图 16 示出了一个配置成两个阵列的 4×4 流形阵列 1600 的通信通路。上面的 2×2×2 阵列由一些“A” PE 构成。下面的那些“B” PE 构成了第二个 2×2×2 阵列。

图 16 采用了(行)×(列)×(面)的标记方式，面之间沿 z 轴通信。在 PE 之间进行这样的通信根据用的是双向口还是单向口通常会需要三个或六个轴的通信口。相反，实现这个优选的流形只需要每个 PE 一个输入口和一个输出口。应注意的是，阵列 1600 可以利用 PE 物理布置与图 11 至 14 所示的阵列 1000 的邻元通信所用的相同的 PE 簇 1652、1654、1656，只要对簇开关的开关设置进行设置，形成图 16 的互连格式。本发明的这种流行阵列组织的许多优点之一是这种配置和连接方法可以产生一些新的强大的能力。

作为这种强大能力的又一个例子，图 17 示出了一个以四维超立方体实现的由簇开关如图所示互连簇 1752、1754、1756、1758 而成的流形阵列 1700。在图 17 中，上 PE 号码，例如 0，0 或 3，1，表示 PE 在环内的位置，如图 1 和 2 中所示，而下 PE 号码，例如 0000 或 1001，表示它在一个超立方体内的位置或地址。在标准超立方体的 PE 之间的通信中，应该超立方的号码中只有一个比特不同的两个 PE 之间才能直接通信。例如，在图 1 中 PE0111 (PE12)与 PE0011、0110、1111 和 0101 通信，每条独立的通路都表示超立方 PE 号码有一个比特不同。然而在图 17 中，不仅 PE 与它们的转置元处在同一个簇内，而且有利的是 PE 与它们的超立方补元也处在同一簇内。可是图 7 的阵列在超立方的两个互补元之间没有直接通路。

图 18 示出了本发明的在每个各有四个 PE 的簇 1852、1854、1856 或 1858 内的 PE 完全连接的流形阵列 1800。对于一个有 N 个各由 N 个 PE 组成的簇的 N×N 流形的情况，簇可以通过选择一个 i 和一个 j，再利用公式：$PE_{(i+a)modN,(j+N-a)modN}$，对于任何 i,j 和对于所有的“a” +

∈{0, 1,…, N-1}，来形成。对于一个诸如流形阵列 1800 那样的四维超立方体，对簇节点号码可以适当地进行格雷编码，成为 $PE_{G((i+a)modN),G((j+N-a)modN)}$，其中 G(X)为 X 的格雷码。下面将简短地以流形阵列的数学表示说明推广到一般情况的过程。

对于 N=4 来说，一个带有超立方节点的适当流形阵列 1800 示于图 18。这个 4×4 流形阵列 1800 包括连接超立方体互补元的连接通路。也就是说，一个超立方 PE 与它的超立方补元之间允许有直接的通信通路。例如，PE0111 ($PE_{1,2}$)可以与 PE 1000 ($PE_{3,0}$)以及它的簇内的其他成员通信。在考虑一个超立方 PE 与它的超立方补元之间的通信关系中，通信通路对于最长在这种情况下为 4 步的被减少到只有 1 步。这样减少通路长度对于根据按本发明的流形阵列组织处理元的情况产生非常有效的超立方型算法是具有相当深远的意义的。此外，4×4 流形阵列的 4PE 簇与现有技术的实现情况相比为 PE 与它的超立方补元之间的通信连接提供了一种低成本的解决方案，因为在现有技术的折叠阵列的情况下，要获得同样的四维超立方连通性需要有一些 8PE 簇。

图 19 示出了一个 5D(五维)超立方体 1900，共有 4×4×2 个 PE，在每个 PE 中以行、列、面作为上 PE 号码，而以 5D 超立方号码作为下 PE 号码。对于一个传统的 5D 超立方体，每个 PE 要有 5 个双向连接口和 10 个单向连接口。对于图 20 所示的流形阵列 2000 这样的将一个 5D 超立方体映射成一个 4×4×2 流形阵列的实现来说，每个 PE 只需要有一个双向口或两个单向口。此外，标准的超立方体需要有 2^5 即 32 个 PE，总共 $5N^2$(N=4)个双总线或 $10N^2$(N=4)个单向总线。然而，图 20 的 5D 超立方流形阵列 2000 只需要在所有的 PE 簇之间有总共 $2N^2$(N=4)个双向总线或 $4N^2$(N=4)个单向总线。图 20 示出了每个簇之间有 8 条发送通路和 8 条接收通路的单向总线情况。注意，如图 20 所示，超立方 PE 及其补元可以有利地配置在同一个 PE 簇内。此外，每个 PE 面的每个 PE 及其相邻转置元也配置在同一个 PE 簇内。而且，各面的相应簇元也分别组合在一起。这种组合方式对于未示出的每簇 16

个 PE 的 6D 超立方体同样适用。

要指出的是，簇开关在 5D 时与在 4D 时是不同的，然而构成的方式是类似的，以提供同样的互通性。注意，这种流形阵列形成技术可以形成具有不同规模的簇。簇的规模根据应用和产品要求选定。

一个具有北、南、东、西和 Z 轴输入/输出(I/O)口的 3D 环拓扑结构要求每个 PE 有 6 个双向三态型链路或 12 个单向链路。这意味着 N × N × N 的 3D 环需要有 $3(N^3)$个双向三态型链路和 $6(N^3)$单向型链路。对于一个 4 × 4 × 4 的 3D 环的流形阵列拓扑结构与对于一个 6D 超立方体的流形阵列拓扑结构是没有区别的。这些 PE 可以按环或超立方体要求标记。此外，8 × 8 的环可以认为是 4 × 4 × 4 的 3D 环在连通性要求上有所降低的一个子图。在以流形阵列实现一个 3D 立方体或 6D 超立方体的拓扑结构中，各 PE 都只需要一个发送口和一个接收口，而与拓扑结构所需的口的数目无关。将一个 PE 布置在 3D 立方体拓扑结构内它的所要求位置上可以由这个 PE 外的开关机构适当处理。对于流形阵列的 3D 环来说，簇间布线的复杂程度得到减小，开关机构只需要 $3(N^3)$的三分之一个链路或(N^3)个双向三态型链路和对于单向型链路来说只需要 $2(N^3)$个，而不是现在通常所需的 $6(N^3)$。这表示实现成本可以大大降低。

下面将给出本发明的流形阵列的各种情况的数学说明。一个超立方体是一个每维规模为 2 的正则环。例如，一个 4 维超立方体可以看作一个 2 × 2 × 2 × 2 的环。然而，以下说明将针对体现为具有较少维数的环为情况。2d 维的超立方体等效于一个边长为 4 的 d 维正则环，而 2d+1 维超立方体等效于一个边长除最后一维为 2 而其他各维均为 4 的 d+1 维正则环。首先，2d 维超立方体等效于一个每维长为 4 的 d 维正则环，其中 d 为自然数。

2d 维超立方体 H 包括 2^{2d} 个节点，等于每边长为 4 的 d 维正规环 T 的节点数，$4^d=(2^2)^d$。按照定义，H 的每个节点与 2d 个其他节点邻接，每维一个。T 的每个节点与每维两个其他节点邻接，也就是说，在一个 d 维正则环中，每个节点也与 2d 个其他节点邻接。因此，H 和 T

具有相等的节点数和相等的边沿数。

为了在它们之间定义一一对应关系，设：$(i_1, i_2, \cdots, i_d)$是 T 的一个节点，其中 i_j 表示节点的对于维 j 的座标。由于 T 是一个每边长为 4 的 d 维正则环，因此对于从 1 至 d 的所有的 j，i_j 取 0 至 3 的值。沿维 k，这个节点与节点$(i_1, i_2, \cdots, i_k-1, \cdots, i_d)$和$(i_1, i_2, \cdots, i_k+1, \cdots, i_d)$邻接。我们假设运算 i_k-1 和 i_k+1 是模 4 运算，即 3+1=0 和 0-1=3，以包括环的环回边沿。

考虑 T 的节点$(i_1, i_2, \cdots, i_d)$与 H 的节点 $G(i_1), G(i_2), \cdots, G(i_d))$之间的一对一映射。这里，G(0)=00、G(1)=01、G(2)=11 和 G(3)=10 都是 2 位格雷码。虽然环的节点用一个元组标示，但超立方体的节点用一个二进制串标示，因此当我们写出$(G(i_1), G(i_2), \cdots, G(i_d))$时，我们实际上是指一个连在一起的相应二进制串。为了弄清楚这一点和我们所提出的一对一映射，考虑一个 3 维正则环的节点(3，1，0)，相应的 6 维超立方体附标就为 100100，这是连接 G(3)、G(1)、G(0)得出的。

由于相继的格雷码只有一个比特是不同的，因此环的邻接节点也是超立方体的邻接节点，反之亦然。所以，在 H 的节点和边沿与 T 的节点和边沿之间存在着一个一对一的映射，也就是说这两个图是对等的。因此，一个维数为 2d 的超立方体可以体现为一个每维长为 4 的 d 维正则环。

对于格雷码和利用格雷码标示超立方节点的详细情况例如可参见 F. Thomson Leighton 的“并行算法和体系结构导论：阵列，树，超立方体”(“Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”，Morgan Kaufmann, 1992, ch.3)，该书列作本发明的参考。

2d+1 维超立方体等效于一个除最后一维长为 2 所有各维长均为 4 的 d+1 维环。从 Leighton 的著作中可见，2d+1 维超立方体可以看成为相应节点互连的两个 2d 维超立方体(见该书第 393 页)。但是，由于 2d 维超立方体等效于每边长为 4 的 d 维正则环，因此两个 d 维的环通过将它们相应的节点连接起来而形成的并就是一个最后一维长为 2 的

d+1 维环。

由上可见，2d 维的超立方体等效于一个边长为 4 的 d 维正则环，而 2d+1 维超立方体等效于一个除最后一维长为 2 而其他各维长均为 4 的 d+1 维环。

流形阵列的组合，或簇，最好将完全相对的节点列在一起，情况如下。一个 d 维超立方体的邻接节点相互只有一个比特是不同的，因此节点地址中所有 d 个比特全不同的节点相互离得最远，也就是说完全相对。离得最远的两个节点的地址互为二进制的补码。因此，我们也将一个给定节点的完全相对节点称为它的补节点。

例如，考虑一个 2 维的 4×4 环和相应的嵌入 4 维超立方体，这个超立方体的节点附标列成图 21A 所示的 4×4 表。沿着这个表的行和列，相邻元之间的距离为 1。如果列 2、3 和 4 向上轮转一个位置，那么第一列与第二列之间相应元的距离就成为 2。对于列 3 和 4 再对列 4 重复同样的操作，那么一个列的元与相邻列的相应元之间的距离就都为 2。这样得到的 4D 流形阵列表如图 21B 所示。

重要的是应注意到这个表的每一行都含有一种对 4 个节点的组合方式，也就是说有两对完全相对的节点，因为 4 是在 4 维超立方体上最大的距离。表的这行确定了将完全相对的节点列入同一组的组合情况。

在较高维的环因此也就是较高维的超立方体的情况下，通过沿最后一维外每个新维同样旋转的方式来达到组合完全相对的节点。

为了在数学上描述这种组形成置换情况，定义两个算子：将一个张量多维阵列拆开用它的元构成一个向量，以及这个算子的逆算子。vec()算子用单个自变量，张量 T，得出一个沿张量的第一维各列排列 T 的元形成的向量。例如，如果 T 是 2 维张量，则 v=vec(T)=[11 21 31 12 22 32 13 23 33]^{T}。另一方面，张量 T 可以利用源结构和各维的列表作为自变量的算子 reshape()根据向量 V 重构。按上个例子，T 用 reshape (v,3,3)重构。

两个矩阵 A 和 B 的 Kronecker 积为一个由 B 的拷贝来以 A 的相应

元构成的块矩阵。即:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$$

为了展现所需组合对多维阵列 T 的操作可以定义为一个矩阵-向量积，其中矩阵是一个置换矩阵，每行和每列都只有一个非零元的正交矩阵，而向量是 vec(T)。首先需要矩阵 S，确定向上轮转的大小为 4 的置换矩阵。

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

还要确定实际执行组合置换的块对角矩阵 G

$$G = \begin{bmatrix} S^0 & 0 & 0 & 0 \\ 0 & S^1 & 0 & 0 \\ 0 & 0 & S^2 & 0 \\ 0 & 0 & 0 & S^3 \end{bmatrix}$$

G 的各对角块分别是 S 的各次方。如果 T 是 4×4 环，reshape(G vec(T),4,4)就是所得到的具有上述提出的组合性质的环。类似，如果 T 代表 4×4×4 的环，那么给定提出的组合的运算为

reshape((G ⊗ I_4)(I_4 ⊗ G) vec(T),4,4,4),

其中 I_4 是 4×4 的单位矩阵。

推广到每维长为 4 的 d 维正则环 T 的一般形式，展现组的置换为

$$reshape(\prod_{q=1}^{d-1}(I_{4^{q-1}} \otimes G \otimes I_{4^{d-q-1}})vec(T), \underbrace{4,4,\cdots,4}_{d}).$$

对 4×4×4 环进行组合置换的一个例子:

第一次用($I_4 \otimes G$)进行矩阵相乘后，四个面分别成为:

000	110	220	330
100	210	320	030
200	310	020	130
300	010	120	230

001	111	221	331
101	211	321	031
201	311	021	131
301	011	121	231

002	112	222	332
102	212	322	032
202	312	022	132
302	012	122	232

003	113	223	333
103	213	323	033
203	313	023	133
303	013	123	233

第二次用$(G \otimes I_4)$进行矩阵相乘后，我们得到：

000	110	220	330
100	210	320	030
200	310	020	130
300	010	120	230

111	221	331	001
211	321	031	101
311	021	131	201
011	121	231	301

222	332	002	112
322	032	102	212
022	132	202	312
122	232	302	012

333	003	113	223
033	103	213	323
133	203	313	023
233	303	013	123

如果不向上轮转而是向下轮转，那么也可保证同样的使完全相对的节点列在一起的组合性质。此外，组也含有对称的节点对，节点(i,j)与它的转置节点(j,i)分在同一组内。在数学上，向下轮转置换是向上轮转置换的转置。在置换矩阵是正交矩阵时，向下轮转置换也是向上轮转置换的逆。具体地说，向下轮转的大小为 4 的置换矩阵为：

$$S_4^T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

在这里，由于我们并没有限制长度为 4，因此我们需要指定矩阵的大小，从而能对任何长度的 2 维环或高维环的 2 维子图进行轮转。类似，为了对所有的列进行相应的轮转，我们确定矩阵 G 的转置，它的各对角块是 S^T 的相应次方。

再回到图 21C 所示的每边长为 4 的 3 维正则环的前面这个例子，第一次用($I_4 \otimes G^T$)进行矩阵相乘后，其中 G^T 为

$$G_4^T = \begin{bmatrix} S^0 & 0 & 0 & 0 \\ 0 & S^1 & 0 & 0 \\ 0 & 0 & S^2 & 0 \\ 0 & 0 & 0 & S^3 \end{bmatrix}^T$$

4 个面成为:

000	310	220	130
100	010	320	230
200	110	020	330
300	210	120	030

001	311	221	131
101	011	321	231
201	111	021	331
301	211	121	031

002	312	222	132
102	012	322	232
202	112	022	332
302	212	122	032

003	313	223	133
103	013	323	233
203	113	023	333
303	213	123	033

以上组合以从 z 轴透视的方式示于图 22，也示于图 23。在第二次用($G^T \otimes I_4$)进行矩阵相乘后，我们得到：

000	310	220	130
100	010	320	230
200	110	020	330
300	210	120	030

131	001	311	221
231	101	011	321
331	201	111	021
031	301	211	121

222	132	002	312
322	232	102	012
022	332	202	112
122	032	302	212

313	223	133	003
013	323	233	103
113	023	333	203
213	123	033	303

这样，不仅是完全相对的节点沿第三维各不同面的相同位置组合

在一起，如图 24 所示，而且相对于前两维的对称节点也沿第二维或面上的行也组合在一起。在图 24 中，特别示出了一个组合 89。组合 89 用作下一组图的参考点。图 25 对 A、B、C 和 D 面根据它们在簇间的连通性进行了重排。图 26 根据子簇之间的本地连通性将每个面分成 2 × 2 的子簇。组 89 示为面 A 内的一个子簇。这种流形阵列的核心体系结构现在可以用来直接代替每个标识的 2 × 子簇，如图 27A 中所示。在图 27B 中，未出了一个 4 × 4 × 4 PE 节点 $PE_{2,2,2}$ 的输入(接收)连通性。注意，其中增添了一组附加的多路复用器，标为 xx1、xx2 和 xx3。对于 4 × 4 × 4 的每个 4 × 4 流形阵列子集，都增添了一组 xx#型的 16 个多路复用器。这些多路复用器都以与图 27B 中为加粗的 PE 节点 2，2，2 示出的相同的方式连接。

应当指出的是，可以通过不同的置换序列来达到同样的对节点的组合。设 P 为在上面所示的这些步骤所需的置换乘在一起得出的置换矩阵。对这个矩阵的任何因子分解都相当于达到同样结果的不同操作步骤序列。例如，假设 $P = A_1A_2A_3$。考虑随机的置换矩阵 Q、R。我们能得出一个达到与置换 P 同样的对节点的组合的不同置换序列。例如，由于 $P = A_1Q^TQA_2R^TRA_3$，因此通过令 $B_1=A_1Q^T$、$B_2=QA_2R^T$ 和 $B_3=RA_3$，就可得到 $P=B_1B_2B_3$。此外，可以置换组的元和/或组的相对次序，达到一种看来不同实质相同的对节点的组合。

按照本发明根据流形阵列构成的网络的特性还可以有益地用来连接节点，以形成一个具有许多优点的网络。下面对此进一步加以说明。

网络直径定义为所有节点之间的距离中的上界。网络直径反映了最差情况下节点对节点通信所需的步骤数。直径越小，最远节点之间通信所需的步骤越少。对于一个 d 维的超立方体来说，H′ 是对 H 增添了连接互补节点的边而形成的新图。设 s 和 t 为 H 的两个互补节点，而 v 是 H 的另一个任何节点。可以证明，从任何一对超立方体互补节点到任何一个超立方体节点 v 的距离之和等于这个超立方体的维数。也就是说，给定一对互补节点 s 和 t 以及另一个节点 v，在 s 与 t 之间有一条通过 v 的最短通路。

从 s 到 v 的距离等于 s 与 v 的二进制表示中数字相互不同的比特数，设为 k。由于 t 是 s 的补码，因此 t 与 s 的二进制表示中数字不同的比特数等于 d-k。所以，从 s 到 v 的距离为 k，而从 t 到 v 的距离为 d-k。也就是说，这两个距离之和为 d。此外，从 s 经 v 到 t 的这条通路具有长度 d，这是一条最短的通路。

此外，通过增添连接一个 d 维超立方体的互补节点的边可以将图的直径在 d 为偶数时减小为 d/2，而在 d 为奇数时减小为(d+1)/2。设 v 是 H 的一个节点，而 k 和 d-k 分别为从 H 的两个互补节点 s 和 t 到 v 的距离。不失一般性，可假设 k < d-k。于是在 H′ 内从 s 到 v 的距离为 k，因为我们可以使用与在 H 内相同的最短通路。在 H′ 内从 t 到 v 的距离为 k+1，因为可以使用通过连接这两个互补节点的新边经 s 的通路。这也就是说，对于 H′ 的任何一对节点 v 和 s，它们的距离不可能在 d 为偶数时超过 d/2，在 d 为奇数时超过(d+1)/2，于是，在 H′ 内经 s 的补节点 t 的最短通路的长度小于 d/2。

一个 d 维超立方体的网络直径，在增添了互补节点连接后成为[d/2]，如上所述。这个结果总结在下表内。注意，在中间的一列只是考虑了连接互补节点的边。标为流形阵列的第三列指出了按照本发明的流形阵列设计的结构内所含的边的数目，以及为 2 的恒定网络直径。

	超立方体 (流形阵列的子图)	带互补边的 超立方体 (流形阵列的子图)	流形阵列 (本发明)
节点数	2^d	2^d	2^d
边数	$d2^{d-1}$	$(d+1)2^{d-1}$	$2^{2k-1}((4*3^{k-1})-1)$;for d=2k $2^{2k}((8*3^{k-1})-1)$;for d=2k+1
边数 网络距离	d	$\left[\frac{d}{2}\right]$	2

上表表明这个子图含有比超立方体多 2^{d-1} 条边，用来连接一个超立方体网络的互补节点，使性能得到惊人的改善。网络直径减小为原超立方体情况下的 1/2。在采用了上表第三列中给出的全部流形阵列边后，按照本发明，网络直径将减小为 2，而与 d 无关。超立方体和带互补边的超立方体是流形阵列的真子图。

虚拟节点的仿真模拟可以如下实现。假设我们具有一个高维网络，需要用一个较小的网络加以模拟。这个要求意味着必需用每个物理节点对多个虚拟节点仿真。下面介绍了几种方式将虚拟网映射到物理网络，使得仿真保持超立方体网络上的超立方邻接关系和超立方互补性以及 2 维环网络上的矩阵转置关系。超立方体仿真可以比较简单。假设有一个 d 维超立方体需要在一个较小的 q 维超立方体上仿真。于是，每个物理节点必需对 2^{d-q} 个节点仿真。一种非常简单的示出本发明的途径的方式是考虑节点的二进制地址。d 维超立方节点需要 d 比特的二进制地址。从这 d 个比特中，q 个比特定义了进行仿真的物理节点的地址，而 d-q 个比特定义了在物理节点内的本地虚拟节点 ID 的地址。确实，对于一个具有 d 个比特的地址 $(i_0 i_1 \cdots i_{q-1}\ i_q \cdots i_{d-1})$（virtual；physical；local）的虚拟节点 v 来说，地址的前 q 个比特标明模拟一组 2^{d-q} 个以虚拟地址的本地 ID 相区别的虚拟节点的物理节点的 ID。v 的任何相邻节点 w 在地址上只有一个比特是不同的。这个比特可能是虚拟 ID 的前 q 个比特中的一个比特，因此 w 属于这个物理节点的相邻节点，或者可能是后 d-q 个比特中的一个比特，这表示 w 是由同一个物理节点模拟的。此外，虚拟节点 v 的补节点可以用模拟 v 的物理节点的补节点来模拟，因为虚拟地址的补码就等于物理和本地地址的补码的串接。由于互补的物理节点在流形阵列内属于同一簇，因此互补的虚拟节点也属于同一簇。

通常，虚拟节点 ID 可以按需要分为两个不一定要接连的部分，即一个物理节点 ID 和一个本地节点 ID。虚拟节点的相邻节点将具有一个节点 ID，它或者在 ID 的物理部分或者本地部分有一个比特是不同的。因此，它分别由物理节点的相邻节点或同一个物理节点模拟。此

外，由于虚拟 ID 的补码，因此一个虚拟节点的补节点始终由物理节点的补节点模拟，它在流形阵列上也是一个相邻节点。

相反，如果一个较小的超立方体用一个较大的超立方体仿真模拟，那么一切都像所预料的那样在流形阵列的一个子集上进行，因为流型阵列是递推定义的。也就是说，有一个流形阵列的子图，其规模等于需加以模拟的超立方体的规模，而且上述所讨论的成立。

环的仿真模拟也很容易加以研究，因为同样的概念对模拟一个环也是成立的。虚拟节点 ID 的(每维)相连接的段相应于一个物理 ID 和一个本地 ID。在物理节点 ID 含有虚拟节点 ID 的一些最高有效位时，我们具有虚拟节点的块分布。否则，在物理节点 ID 含有虚拟节点 ID 的一些最低有效位时，我们具有虚拟节点的循环分布。在块分布中，一组具有相继 ID 的虚拟节点由同一个物理节点模拟。16 个虚拟节点在 4 个物理节点上的块分布为物理节点 0 模拟虚拟节点 0、1、2 和 3，物理节点(模拟虚拟节点 4、5、6 和 7，如此等等。在循环分布中，一组具有相继 ID 的虚拟节点由不同的物理节点模拟。此时，16 个虚拟节点在 4 个物理节点上的循环分布为物理节点 0 模拟虚拟节点 0、4、8 和 12，物理节点 1 模拟虚拟节点 1、5、9 和 13，如此等等。

通过将一个虚拟地址加 1 或减 1，我们找到这个节点的一个沿所规定的轴的相邻节点。这样加 1/减 1 相当于对虚拟地址的本地部分或物理部分或者两者加 1/减 1，因此保证相邻的虚拟节点由同一物理节点或一个相邻物理节点模拟。为了保证转置虚拟节点由同一个物理节点或相邻物理节点模拟，虚拟地址的物理段和本地段的指定必需对于所有各维是相同的。也就是说，块-块或循环-循环的虚拟节点分布可以维持转置节点的相邻关系。

回到并行机内的超立方流形阵列的例子，对于高性能的算法运算来说数据安排可能是头等重要的。在一个阵列处理器中，为了使在处理元之间移动数据引起的等待最小，数据必需一开始就安排在适当的 PE 内，而后在运算期间在直接连接的 PE 之间移动。因此，随着算法进行到各个运算阶段，数据移动必需加以优化，以便使整个算法的总

通信等待最小。为了演示流形阵列的性能，将在图 28A 至 30 所示的 4 × 4 流形阵列 2800 上对一种完全混洗算法和一种在一个 PE 与它的超立方补元之间的通信的算法进行考察。利用张量积代数将完全混洗算法映射成流形阵列处理器。

张量积代数，也称为 Kronecker 积，表示一种将数学方程式映射成适合针对目标机体系结构编码的算法的矩阵形式的方法。对于张量积引论和在将问题映射成目标体系结构中的应用可参见例如 J. Granata, M. Conner, R. Tolimieri 的“张量积: FFT 和其他快速 DSF 运算的数学编程语言”(“The Tensor Product: A Mathematical Programming Language for FFTs and other Fast DSP operations”, IEEE SP magazine, pp. 40-48, January 1992)和 J.R. Johnson, R.W. Johnson, D. Roodriguez 和 R. Tolimieri 的“根据不同体系结构设计、修改和实现付立叶变换算法的方法”(“A Methodology for Designing, Modifying and Implementing Fourier Transform Algorithms on Various Architectures”, Circuits Systems Signal Process Vol. 9, No.4, pp.449-500, 1990)。这两篇论文在此列作参考。

在对一个流形阵列完全混洗例的张量表示法中，完全混洗定义为一个用 $P^{2n}_{2^{n-1}}$ 表示的置换矩阵(见 J.R. Johnson 等人的论文 p472)。这个置换矩阵通常解释为定义了一种寻址机制，以访问需装入一个特定机器单元的数据或需存储来自一个特定机器单元的数据。通常，置换矩阵表示将数据放入下次要运算操作的适当位置所需的数据移动。因此，重要的是优化数据移动，置换矩阵，到目标体系结构的映射。对于作为一个单指令多数据流(SIMD)超立方体机运行的流形阵列机构来说，可以方便地实现完全混洗，如果数据在阵列内布置适当的话。一个 $P^{2n}_{2^{n-1}}$ 在 n=5(P^{32}_{16})时的完全混洗结合图 28 至 33 加以说明。图 28A 示出了用来说明完全混洗算法的总线结构和各操作单元。在图 28A 内包括有多个控制器、存储单元 0 至 3 和一个专用 FIFO 缓存器。这样组织的这个优选实施例将这些存储单元、控制器和 FIFO 缓存器与 PE

阵列配置在同一个芯片上。然而，应当看到，本发明并不局限于单芯片实现。流形阵列的概念很方便地可以用于配有电缆总线、外部存储器和外部控制器的微处理器芯片 PE 阵列。就本讨论而言，说明的是一个单片高性能的阵列处理器，使一种单体系结构和机器组织得以限定，规模对于整个机器系列是可变的。

所以，为了降低成本，这些控制器、存储单元、数据缓存器(例如 FIFO)和 PE 全都配置在单个芯片上。这些控制器通过控制信号，例如存储器地址和装载/存储信号，或者通过在指令总线上发送给例如各 PE 的分发指令对各阵列簇、存储器和 I/O 功能进行控制。这些控制器都是 SIMD 机内的典型功能单元，因此只在支持下面的算法中加以说明。数据缓存器，在图 28 至 30 中示为一个专用的 FIFO 缓存器，也是在存储器/直接存储器访问(DMA)接口单元中通常使用的，因此在此也只作一般性的说明。

图 28A 例示了多个控制怎样可以用来支持一个可重新配置的成簇 PE 拓扑结构。在图 28A 中，有一个控制器与每个有 4 个 PE 的簇都关联。控制器 0 认为是一个主控制器，虽然也可以采用其他方案。指令总线 IO 接至它身的簇和那些分别与其他控制器配合的指令开关(ISW)。图 28A 中的各个 I_{SW} 可以使指令总线 IO 接至输出端 C 或将各自的控制器输入指令总线 I1、I2 和 I3 接至各自的输出端 C。这些 I_{SW} 由主控制器直接或间接提供的控制信号控制。主控制器也可以接收系统组织内指定提供这信息的另一个处理器(诸如一个主处理器)直接或间接发来的信息。对于这个完全混洗的例子来说，这些 I_{SW} 设置成将 IO 接至所有的指令总线通路。因此，控制器 O 起着控制图 28A 中所示的所有四个 PE 簇 2852、2854、2856 和 2858 的单控制器的作用。在图 28A 中示出了一个外部接口通路，它接至一个数据源，例如存储器子系统。

首先看图 28B 的底部，那里示出了一个线性增长的数据项序列，作为一个例子，8 个数据项一组按 FIFO 地址成组输入片上 FIFO。地址示于 FIFO 内相应各列的上方。如图 28B 所示，第一组数据项{0 至

7}存储在 FIFO 0 内，下一组{8 至 15}存储在 FIFO 1 内，如此类推。在本例中，FIFO 提供 8 条输出通路，每个列数据项一条，给四个由控制器(在本例中为控制器 0)或本地缓存控制功能控制的多路复用器，以将数据以下面结合图 29 要说明的那种方式进行装载。在图 28B 中，所示各总线 D0、D1、D2 和 D3 都可以是一个三态双向总线，或者是独立的装载和独立的存储总线。这些总线的宽度可以是任何与所期望的应用匹配的，通常是 8 比特、16 比特、32 比特或 64 比特，当然也可以是其他宽度。

为了简明起见，没有示出 PE 簇间的互连情况。图 3A 至 3C 的基本存储器块在图 28 的 4×4 流形阵列 2800 内已扩展为 N=4 个存储器块，以支持按超立方体模式布置数据和增大对簇的数据接口带宽。这些存储器块分别标以 0 至 3，而总线通路也是依次每簇一条。例如，存储器 0 通过数据总线 D0 接至簇 A 2852 的各 PE{(0,0), (3,1), (1,3), (2,2)}，而存储器 1 通过数据总线 D1 接至簇 B 2854 的各 PE{(3,2), (0,1), (2,3), (1,0)}，如此类推。注意，其他总线结构也是可以的，并非必需采用所例示的方式。

数据装入 FIFO 缓存器后，过程的下个步骤就是将数据装入内部存储器 0 至 3 (M0，M1，M2 和 M3)，如图 29 所示。对于本说明来说，假设数据项是 32 比特，数据总线也是 32 比特。用 8 个 FIFO 至存储器装载周期按下面次序每次并行装载 4 个数据项，所用表示法为：存储器单元-数据项或 Mx-a。用来产生装入图 29 中各存储单元的所示数据模式的次序是：第一周期(M0-0，M1-1，M2-3 和 M3-2)，第二周期(M0-6，M1-4，M2-5 和 M3-7)，如此继续到第八周期(M0-31，M1-30，M2-28 和 M3-29)。例如，存储器 2(M2)用来自 FIFO 线的数据(如图 29 中用虚线 57 圈出的项)装载。现在必需将存储器的数据装入 PE 阵列 2800。这些存储器块由控制器 0 寻址和控制。为了将数据装入 PE，控制器 0 向各 PE 分发一个指令，通知它们需要从它们的数据总线装入数据和需要将数据装入 PE 内部的什么地方。然后，控制器 0 同步地向存储器块提供一个地址，在这种情况下，对于四个存储器 0 至 3 的

每个存储器都是相同的地址。在与地址同步中，控制器 0 于是产生存储器单元从受寻址的位置读出数据送至存储器单元各自数据总线所必需的信号。

在同步情况下，适当的 PE 按控制器 0 分发的指令所明确的从它们的数据总线上取得数据予以装入。控制器 0 标明选择适当 PE 的情况。这种选择可以用多种方式实现，例如在控制器发送给 PE 的分发指令内加以标识，或用分别配置在各 PE 内的可编程允许/禁止比特，等等。控制器在指令总线上向这些 PE 分发一个 PE 装载指令序列来达成这个结果。用总共 8 个存储器至 PE 装载周期装载 32 个数据项，按下面次序每个装载周期并行将 4 个数据项装入相应 PE，所用的表示法为：存储单元-数据项-PE#。用来产生装入图 30 中各 PE 的所示数据模式的次序是：第一周期(M0-0-$PE_{0,0}$, M1-1-$PE_{0,1}$, M2-3-$PE_{0,2}$, M3-2-$PE_{0,3}$)，第二周期(M0-6-$PE_{1,3}$, M1-4-$PE_{1,0}$, M2-5$PE_{1,1}$, M3-7-$PE_{1,2}$)，第三周期(M0-9-$PE_{3,1}$, M1-11-$PE_{3,2}$, M2-10-$PE_{3,3}$, M3-8-$PE_{3,0}$)，第四周期(M0-15-$PE_{2,2}$, M1-14-$PE_{2,3}$, M2-12-$PE_{2,0}$, M3-13-$PE_{2,1}$)，一直继续到这 32 个数据项全部装入 PE 阵列，如图 30 中所示。这种装载模式在以正确次序读出数据项时就实现了一次完全混洗。依次对这 32 数据项的表执行一个完全混洗操作的情况示于图 31。

我们知道 $X=(P^{32}_{16})(P^{32}_{16})(P^{32}_{16})(P^{32}_{16})(P^{32}_{16})X$。这个等式可以用来在流形阵列上演示一个通信例子。32 元的向量 X 的首次置换是由如图 30 中所示的装载操作完成的。下次置换是由 PE 相邻对之间的北向交换操作实现的。下表定义了四个方向的邻元交换操作。

- 东向交换 {0,0 & 0,1}, {0,2 & 0,3}, {1,0 & 1,1}, {1,2 & 1,3},
  {2,0 & 2,1}, {2,2 & 2,3}, {3,0 & 3,1}, {3,2 & 3,3}
- 南向交换 {0,0 & 1,0}, {2,0 & 3,0}, {0,1 & 1,1}, {2,1 & 3,1},
  {0,2 & 1,2}, {2,2 & 3,2}, {0,3 & 1,3}, {2,3 & 3,3}
- 西向交换 {0,0 & 0,3}, {0,1 & 0,2}, {1,0 & 1,3}, {1,1 & 1,2},
  {2,0 & 2,3}, {2,1 & 2,2}, {3,0 & 3,3}, {3,1 & 3,2}

· 北向交换 {0,0 & 3,0}, {1,0 & 2,0}, {0,1 & 3,1}, {1,1 & 2,1},
{0,2 & 3,2}, {1,2 & 2,2}, {0,3 & 3,3}, {1,3 & 2,3}

交换操作使得在规定的 PE 之间进行一次寄存器数据值交换。需交换的寄存器值的选择在每个 PE 接收到的分发指令内给定。对于本例来说，一个 PE 内的寄存器 R1 与另一个 PE 内的寄存器 R2 交换。图 31 示出了这个完全混洗序列，以列的形式列出各 PE 的超立方号码和所含有的寄存器 R1 和 R2。由一个交换(方向)指令隔开的每一列指出了这个交换操作的结果。如图 31 所示，在每个只需要单个在成对 PE 之间移动相邻数据的周期的交换操作中实现完全混洗。

图 32 和 33 用来进一步扩展这个说明。图 32 示出了在完成给各 PE 的北向交换分发指令时的寄存器结果。图 33 示出了在完成给各 PE 的南向交换分发指令时的寄存器结果。西向交换和东向交换指令以同样方式处理。这个所说明的例子的重要性在于对于许多需要完全混洗数据的算法来说，如果数据能以所示的超立方模式装载，那么在流形阵列 2000 上就能得到极高速度的处理。

最后，说明一个流形阵列超互补的例子。在这个例子中，在超立方 PE 与其超立方补元之间的寄存器值交换如以上这个完全混洗例中所示那样利用交换命令执行。超互补提供了在每个超立方节点一个 PE 情况下将超立方机内最长的通路缩减为单个周期的最佳方案。图 18 示出了超互补连接所用的通路，使得所连接 PE 之间可以在单个周期内直接交换数据。

虽然本发明以具体的优选实施例和典型应用为背景作了说明，但可以看到，本发明能用于许多应用，这由所附权利要求限定。举例来说，虽然这些优选实施例针对的是由处理元构成的簇，但也可以用于由节点构成的簇。这样的节点可以是一些存储元，用以形成一个平铺式存储系统，使得同时可以存取多个存储数据块。此外，节点也可以是连接口、输入/输出装置之类。再举例来说，这些节点可以用来连接通信网内的多个通信信道。

图 1A
（现有技术）

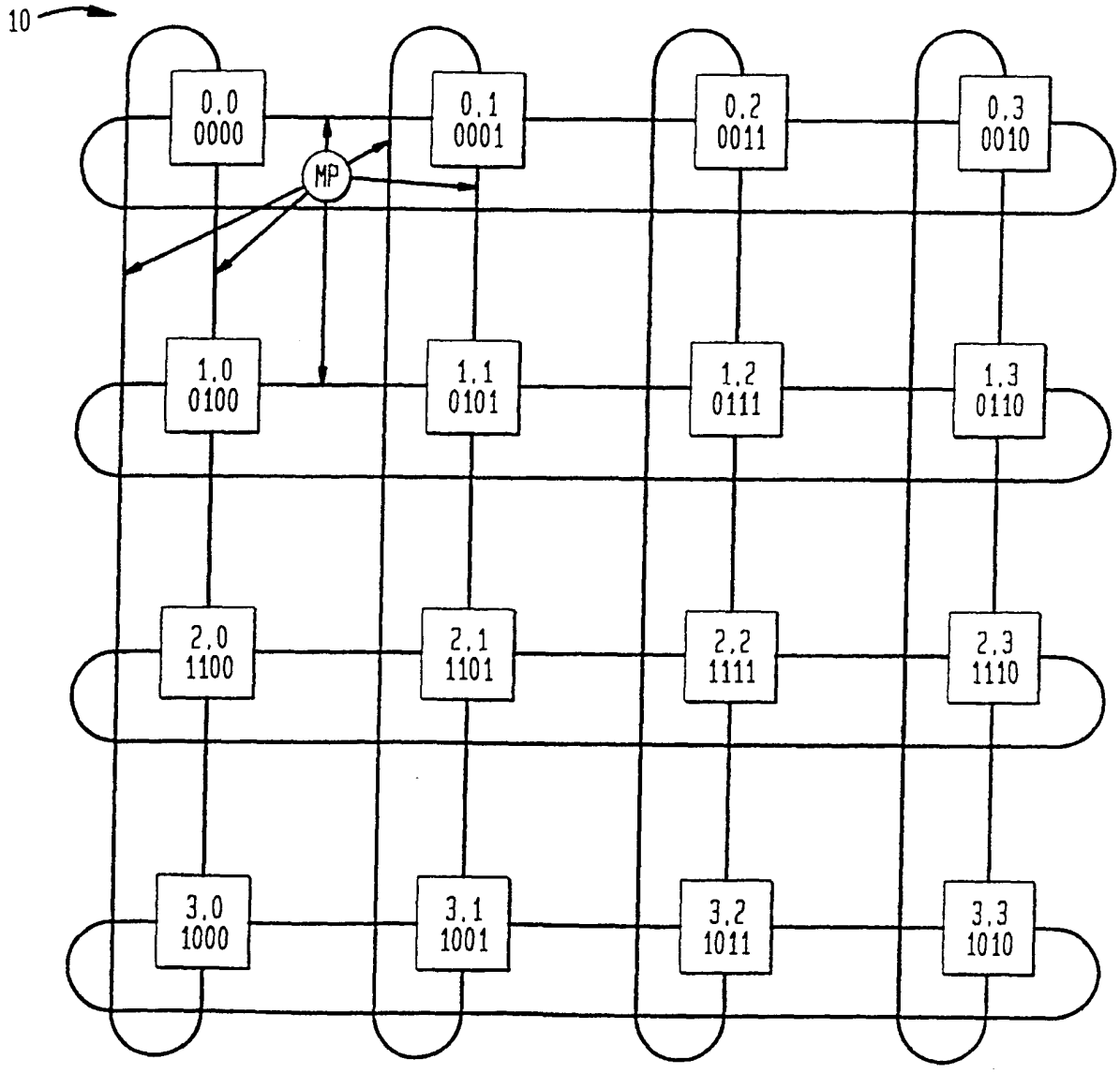

10
0,0
0000
0,1
0001
0,2
0011
0,3
0010
MP
1,0
0100
1,1
0101
1,2
0111
1,3
0110
2,0
1100
2,1
1101
2,2
1111
2,3
1110
3,0
1000
3,1
1001
3,2
1011
3,3
1010

图 1B
（现有技术）

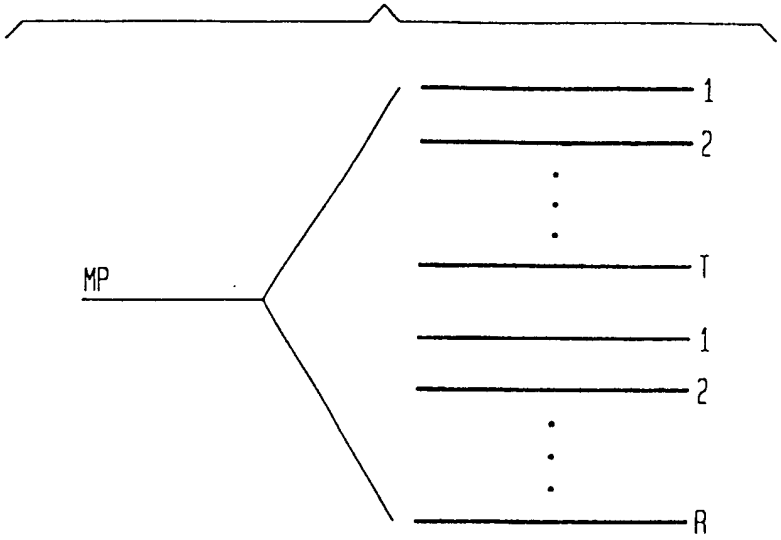


图 1C
（现有技术）

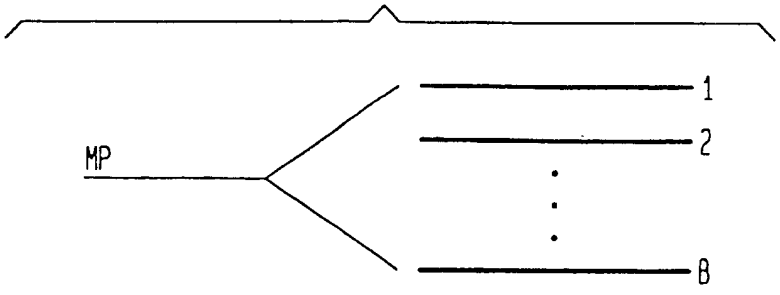


图1D
（现有技术）

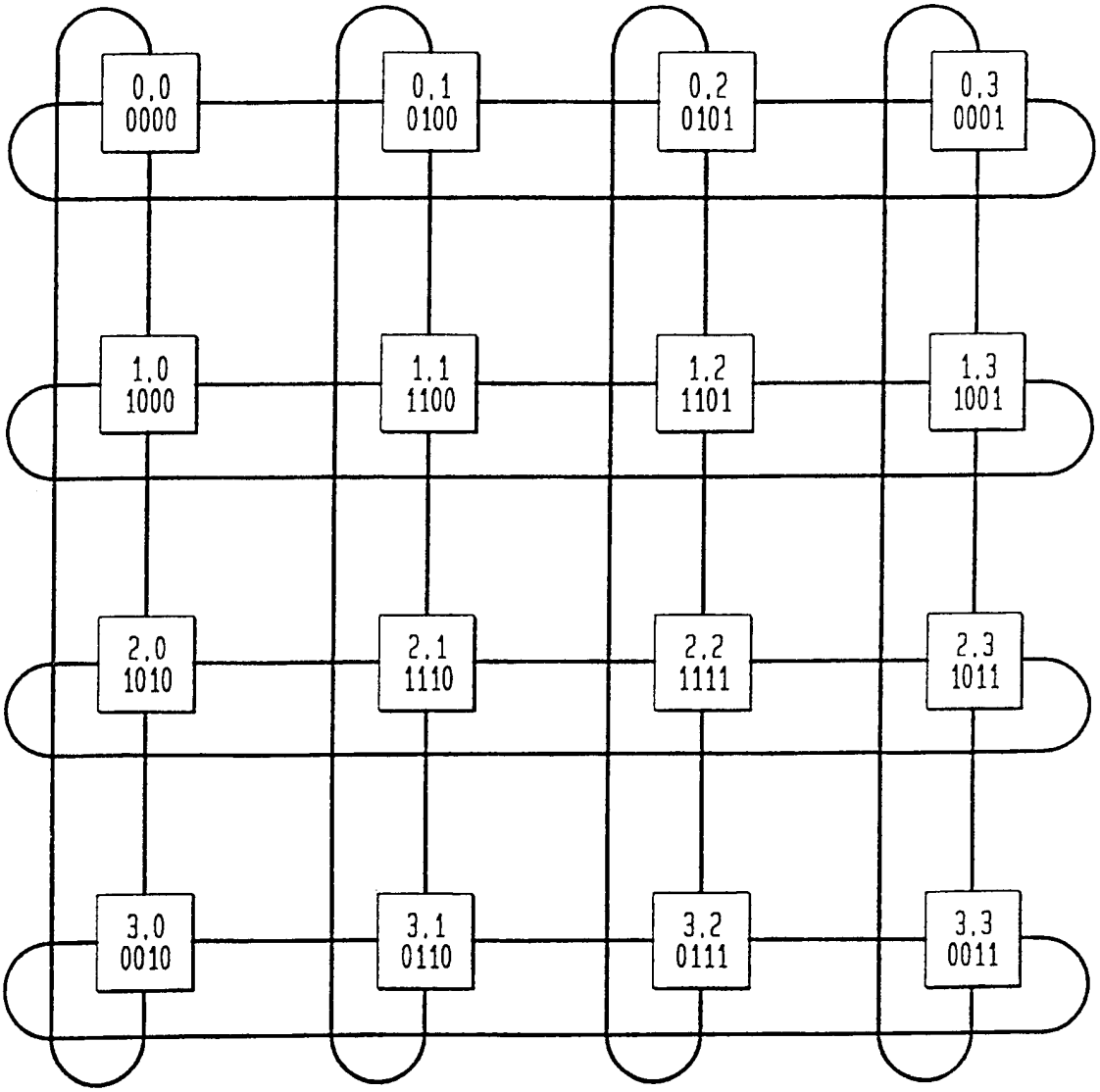

0,0
0000
0,1
0100
0,2
0101
0,3
0001
1,0
1000
1,1
1100
1,2
1101
1,3
1001
2,0
1010
2,1
1110
2,2
1111
2,3
1011
3,0
0010
3,1
0110
3,2
0111
3,3
0011

图 2A
（现有技术）

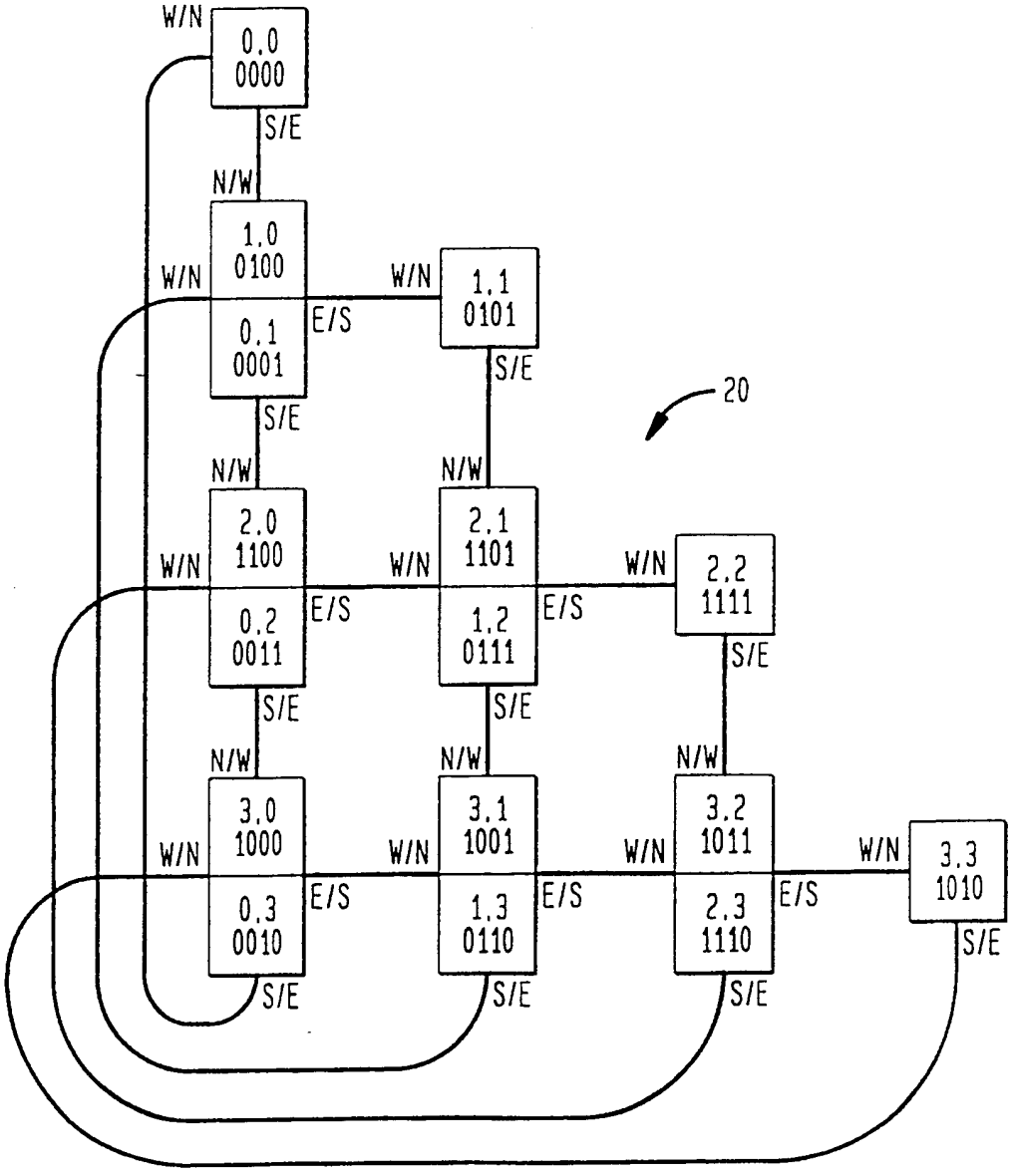

W/N
0,0
0000
S/E
N/W
1,0
0100
W/N
W/N
1,1
0101
E/S
0,1
0001
S/E
S/E
20
N/W
N/W
2,0
1100
2,1
1101
W/N
W/N
W/N
2,2
1111
E/S
E/S
0,2
0011
1,2
0111
S/E
S/E
S/E
N/W
N/W
N/W
3,0
1000
3,1
1001
3,2
1011
W/N
W/N
W/N
W/N
3,3
1010
E/S
E/S
E/S
0,3
0010
1,3
0110
2,3
1110
S/E
S/E
S/E
S/E

图 2B
（现有技术）

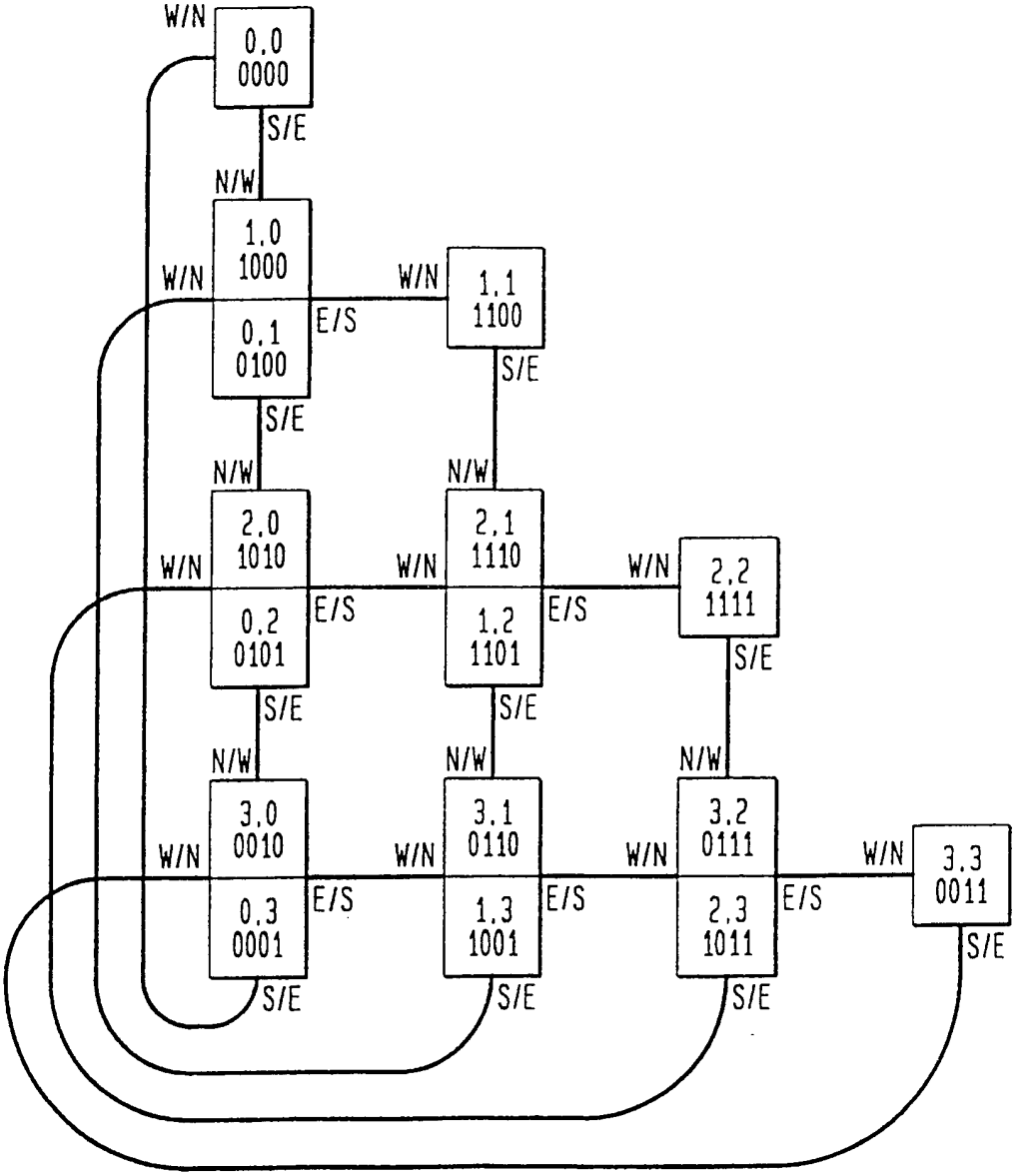

W/N
0,0
0000
S/E
N/W
1,0
1000
W/N
W/N
1,1
1100
E/S
0,1
0100
S/E
S/E
N/W
N/W
2,0
1010
W/N
W/N
2,1
1110
W/N
2,2
1111
E/S
0,2
0101
E/S
1,2
1101
E/S
S/E
S/E
S/E
N/W
N/W
N/W
3,0
0010
W/N
W/N
3,1
0110
W/N
3,2
0111
W/N
3,3
0011
E/S
0,3
0001
E/S
1,3
1001
E/S
2,3
1011
E/S
S/E
S/E
S/E
S/E

图 3A

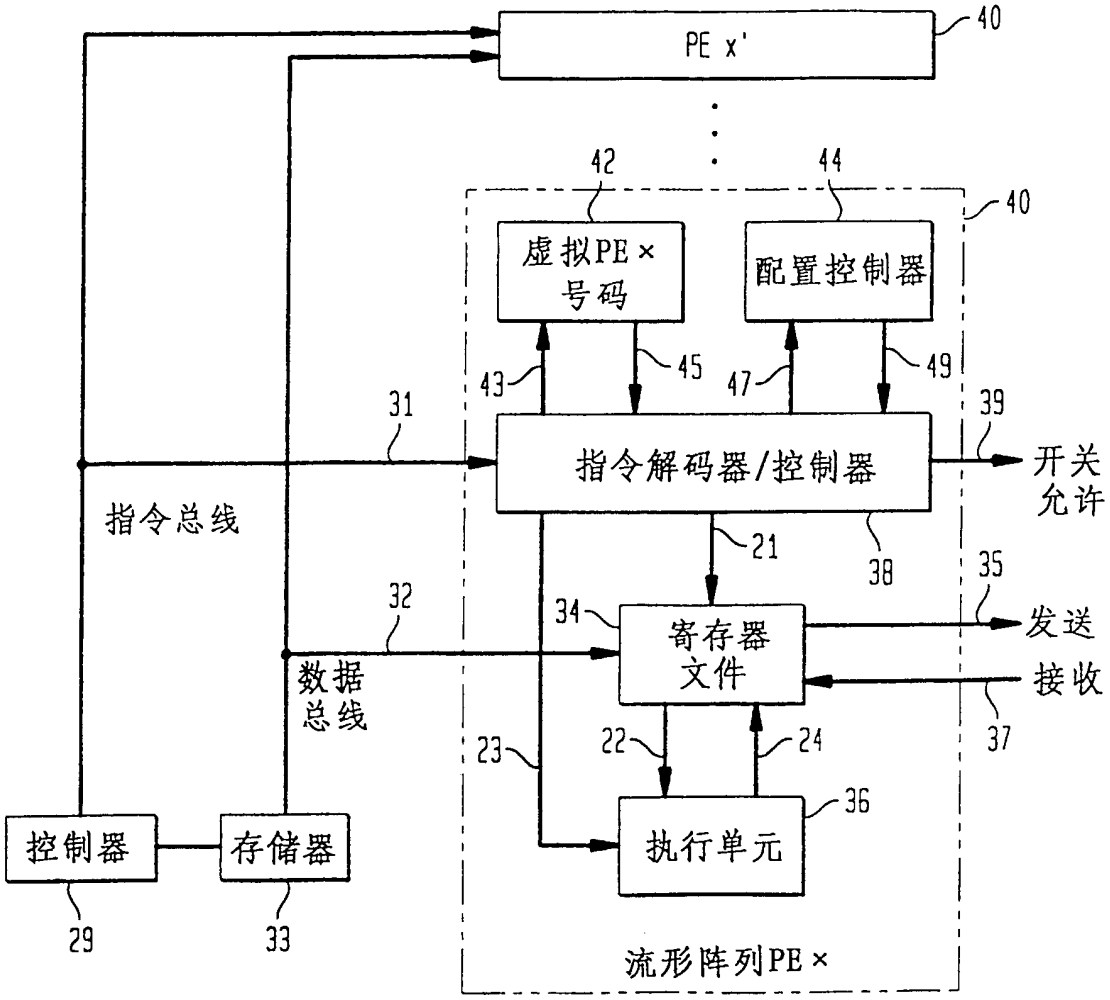

PE x'
40
42
44
40
虚拟PE×
号码
配置控制器
43
45
47
49
31
39
指令解码器/控制器
开关
允许
指令总线
21
38
35
32
34
寄存器
文件
发送
接收
数据
总线
37
23
22
24
36
执行单元
控制器
存储器
29
33
流形阵列PE×

图 3B

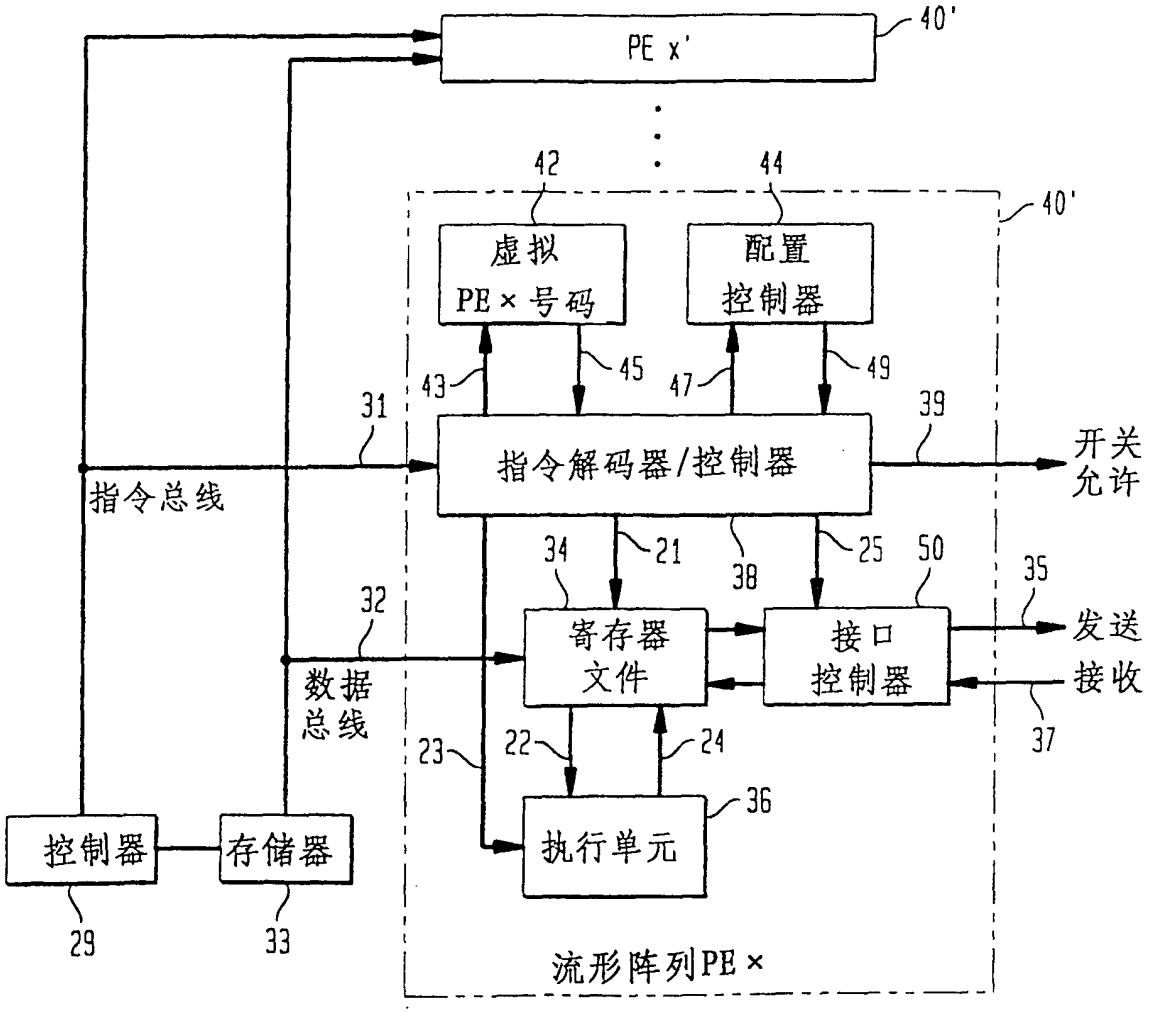
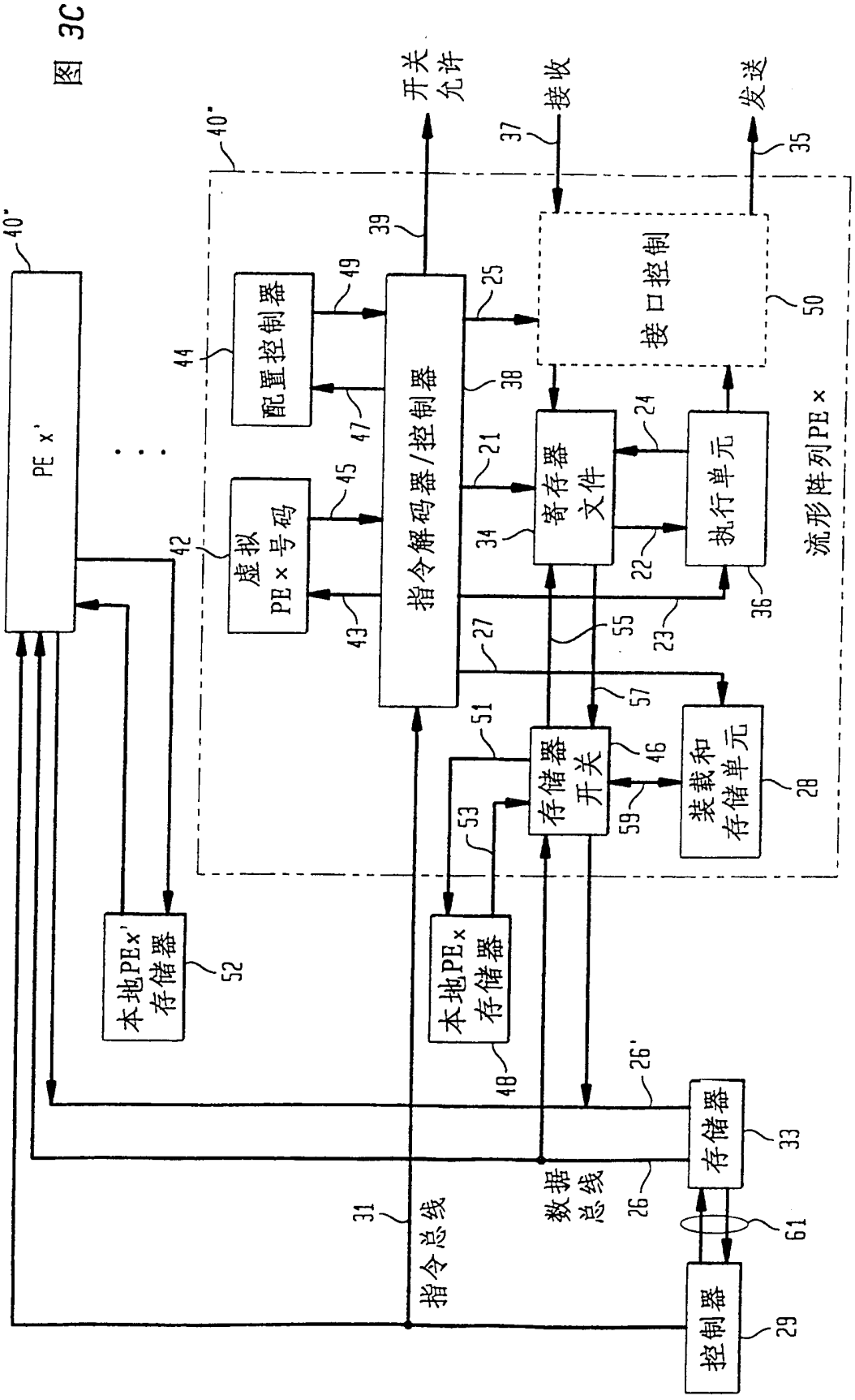

PE x'
40'
42
44
虚拟
PE × 号码
配置
控制器
40'
43
45
47
49
31
39
指令解码器/控制器
开关
允许
指令总线
34
21
38
25
50
35
32
寄存器
文件
接口
控制器
发送
接收
数据
总线
37
23
22
24
36
执行单元
控制器
存储器
29
33
流形阵列PE ×

图 3C


PE x'
40"
40'
44
配置控制器
49
47
虚拟
PE x 号码
45
43
42
指令解码器/控制器
39
开关
允许
25
38
21
34
27
寄存器
文件
24
22
执行单元
36
23
55
57
51
53
存储器
开关
46
59
装载和
存储单元
28
接口控制
50
37
接收
35
发送
流形阵列PE x
本地PEx'
存储器
52
本地PEx
存储器
48
26'
存储器
33
61
数据
总线
26
31
指令总线
控制器
29

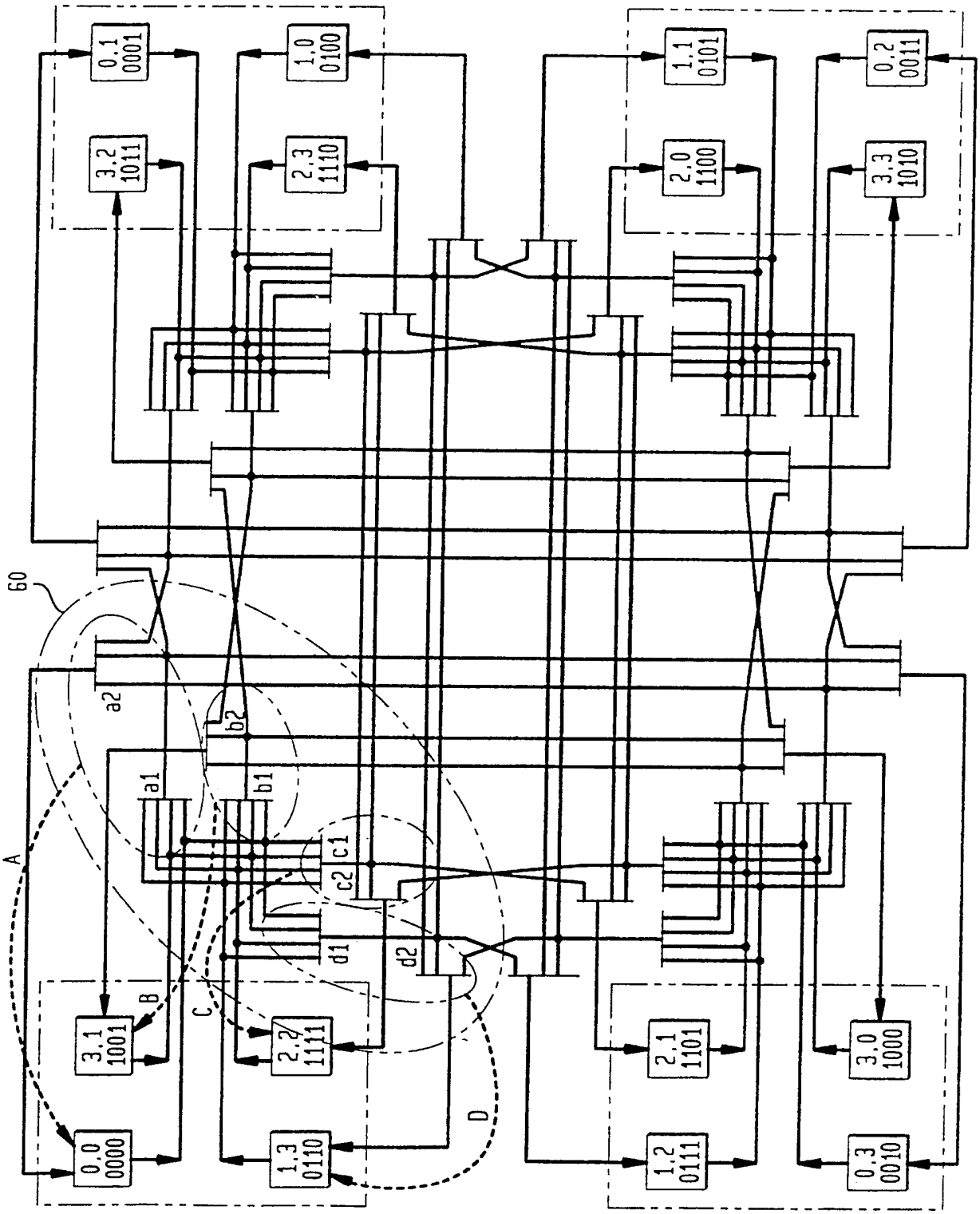
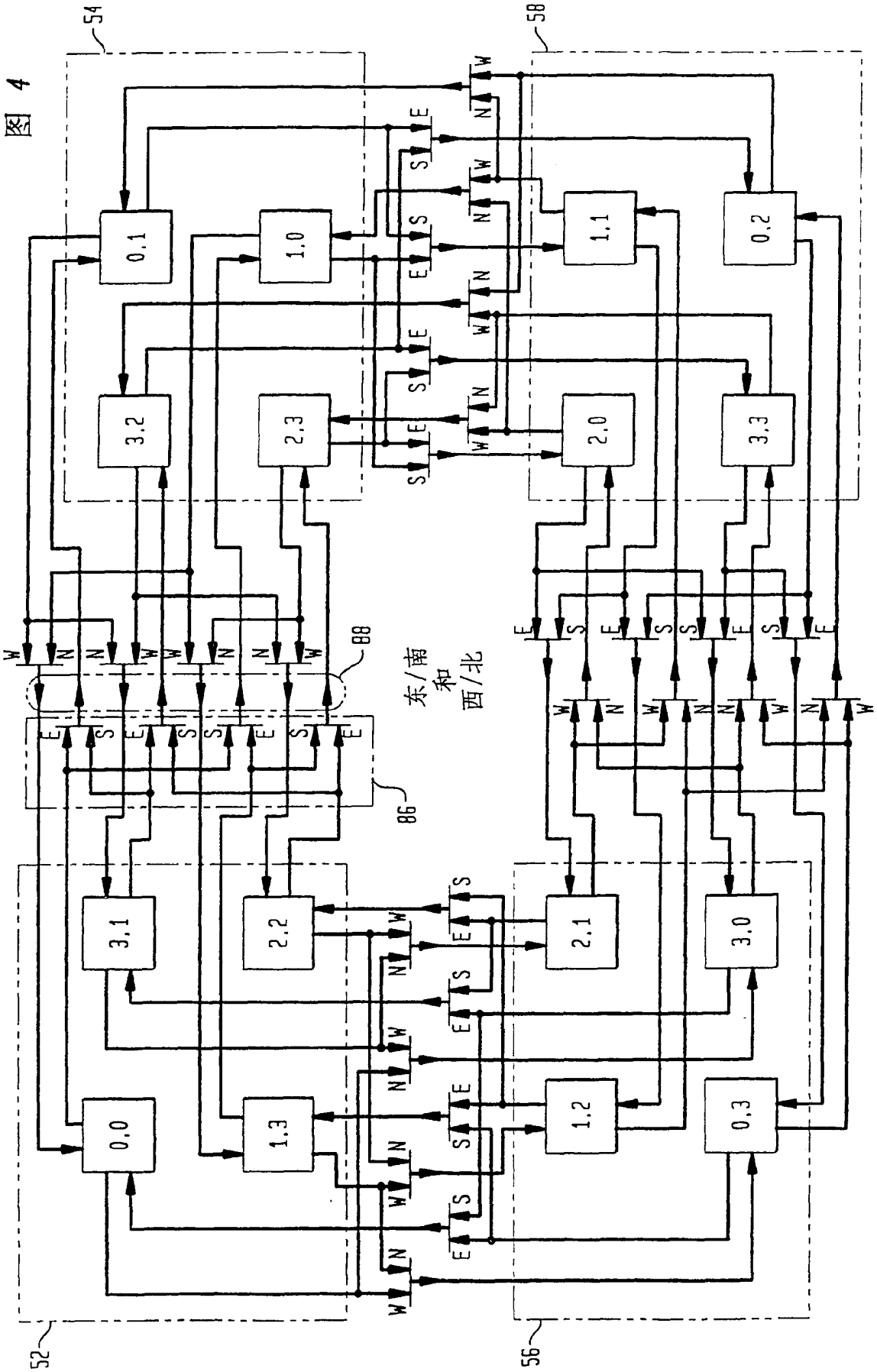

0.1
0001
1.0
0100
1.1
0101
0.2
0011
3.2
1011
2.3
1110
2.0
1100
3.3
1010
60
a2
b2
a1
b1
c1
c2
d1
d2
A
B
C
D
3.1
1001
2.2
1111
2.1
1101
3.0
1000
0.0
0000
1.3
0110
1.2
0111
0.3
0010

图 3D

图 4


54
58
52
56
88
86
东/南
和
西/北
0,0
0,1
0,2
0,3
1,0
1,1
1,2
1,3
2,0
2,1
2,2
2,3
3,0
3,1
3,2
3,3
W
N
E
S

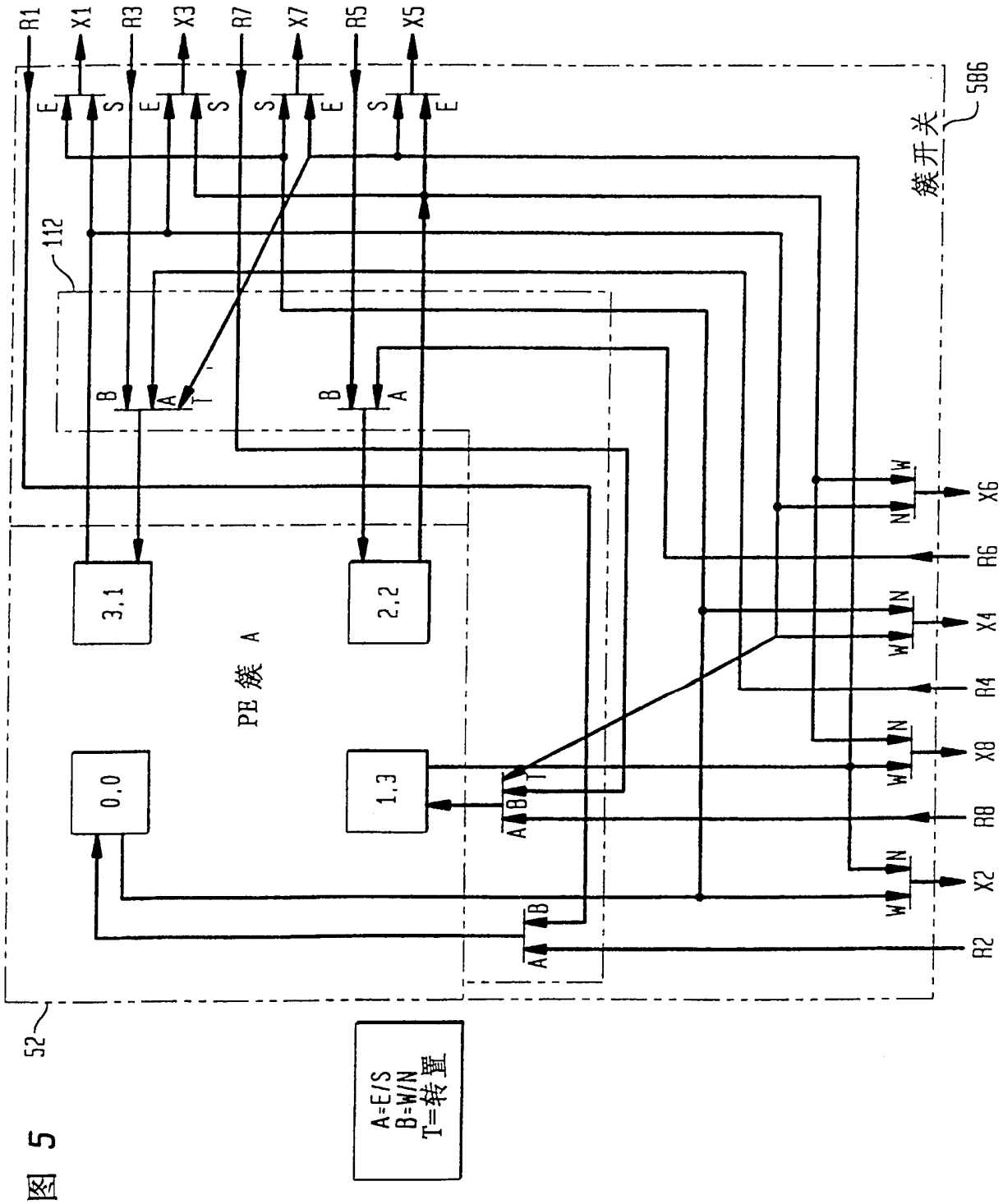

R1
X1
R3
X3
R7
X7
R5
X5
E
S
E
S
S
E
S
E
586
簇开关
112
B
A
T
B
A
3.1
2.2
PE 簇 A
0.0
1.3
A
B
T
A
B
W
N
W
N
W
N
W
N
X6
R6
X4
R4
X8
R8
X2
R2
52
A=E/S
B=W/N
T=转置

图 5

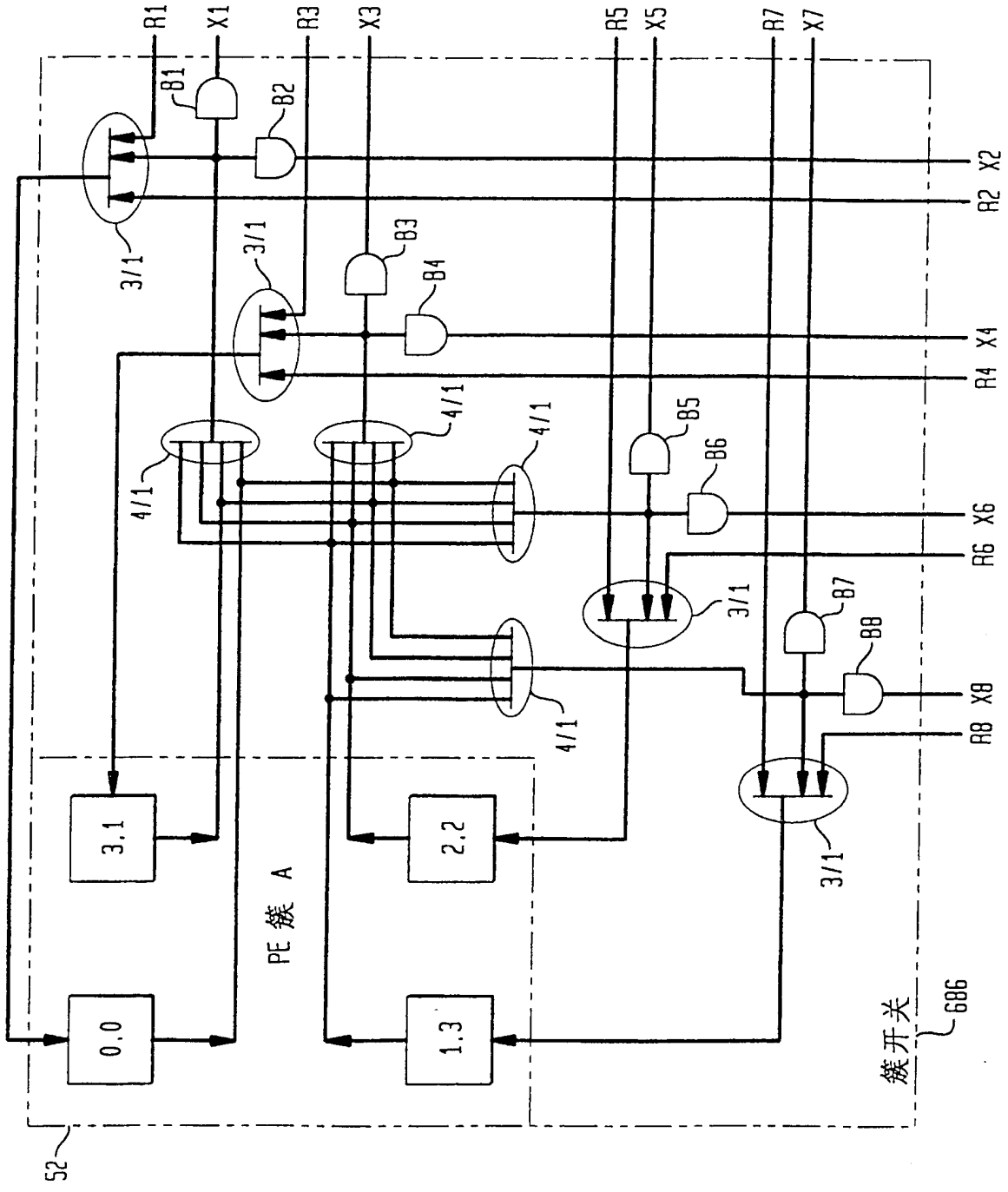

R1
X1
R3
X3
R5
X5
R7
X7
B1
B2
B3
B4
B5
B6
B7
B8
3/1
4/1
X2
R2
X4
R4
X6
R6
X8
R8
3.1
2.2
0.0
1.3
PE 簇 A
簇开关
686
52

图 6

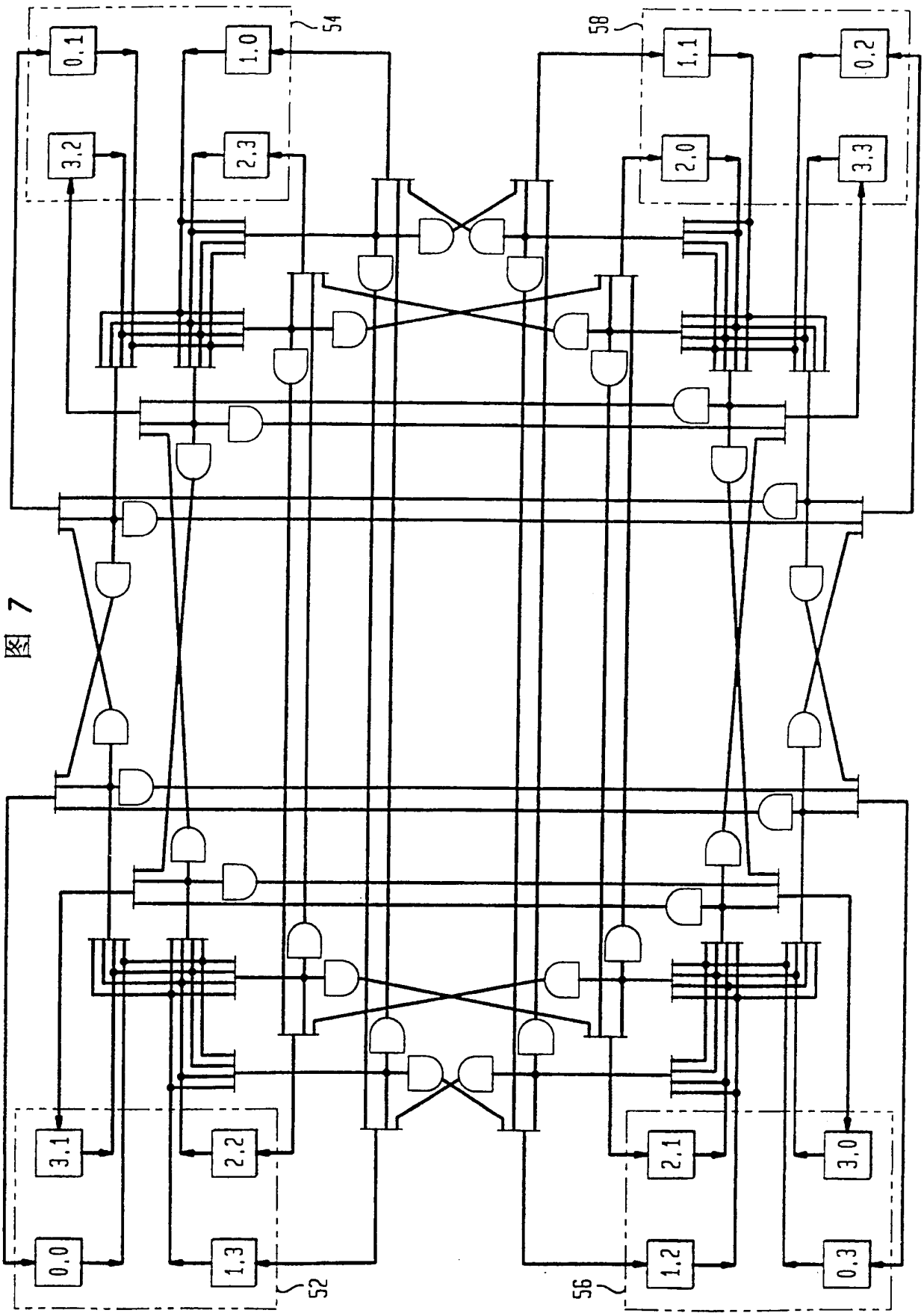


图 7

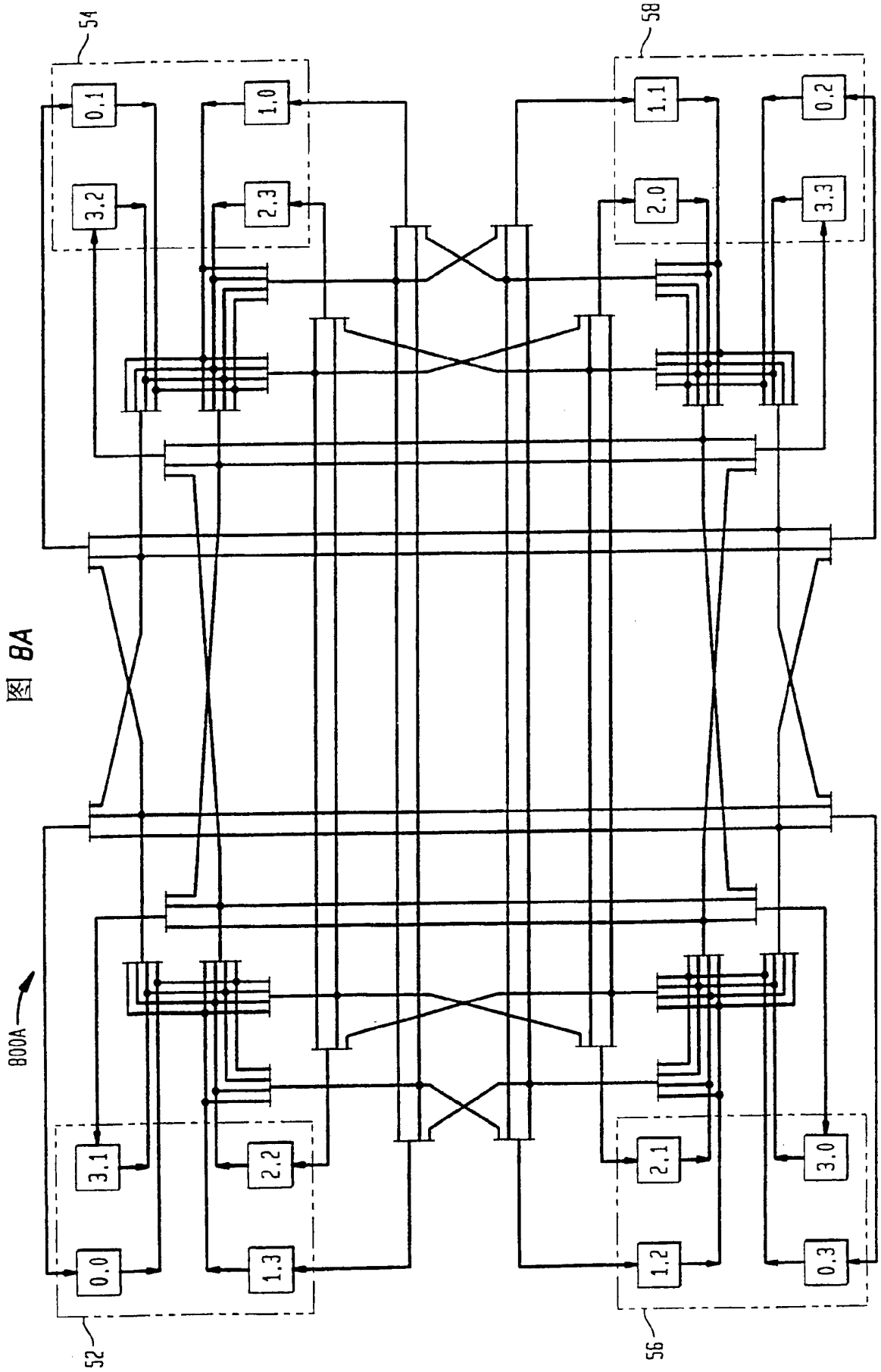


图 8A

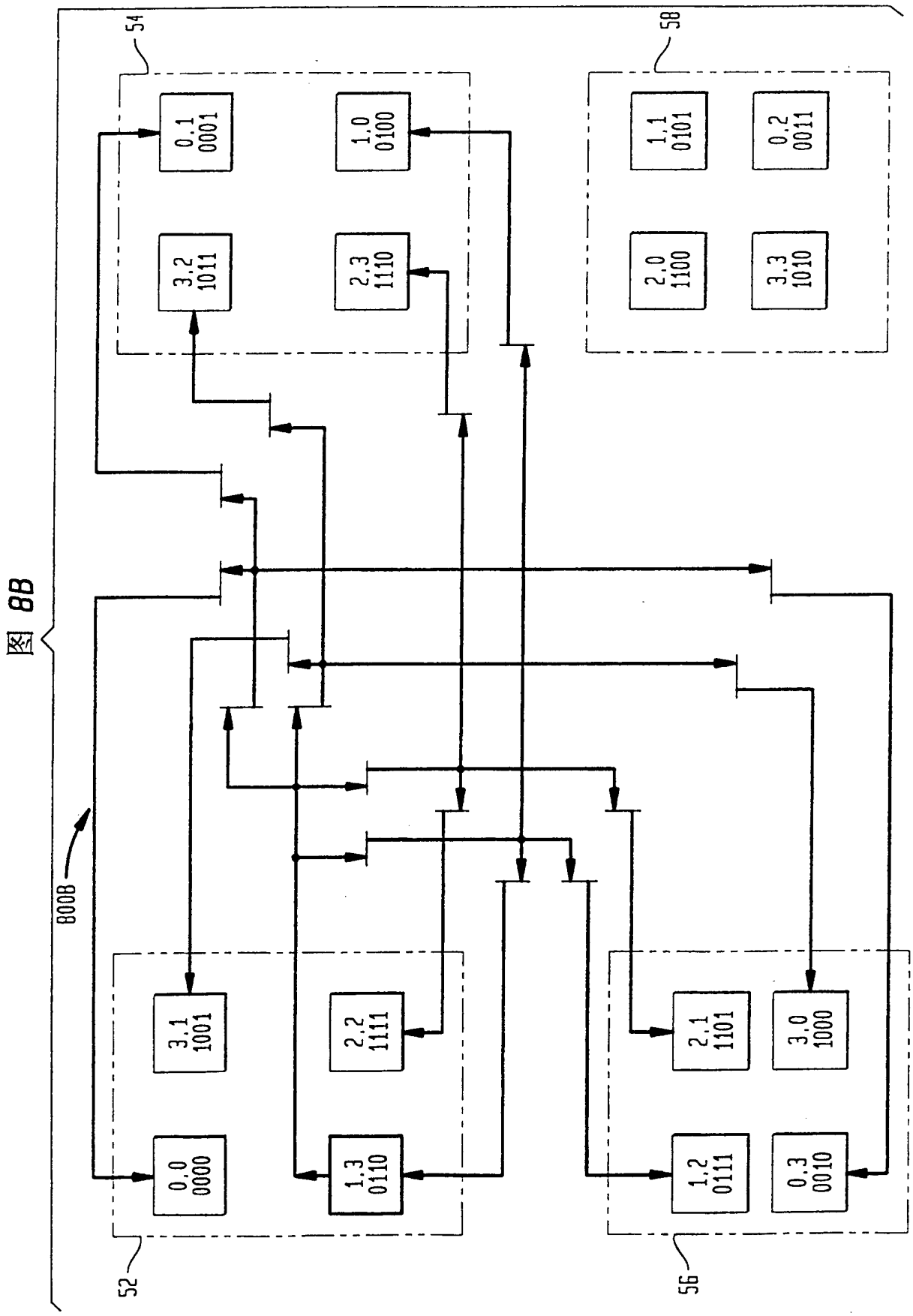

800B
52
54
56
58
0,0 0000
0,1 0001
1,0 0100
1,1 0101
0,2 0011
2,0 1100
3,2 1011
2,3 1110
3,3 1010
3,1 1001
2,2 1111
1,3 0110
2,1 1101
3,0 1000
1,2 0111
0,3 0010

图 8B

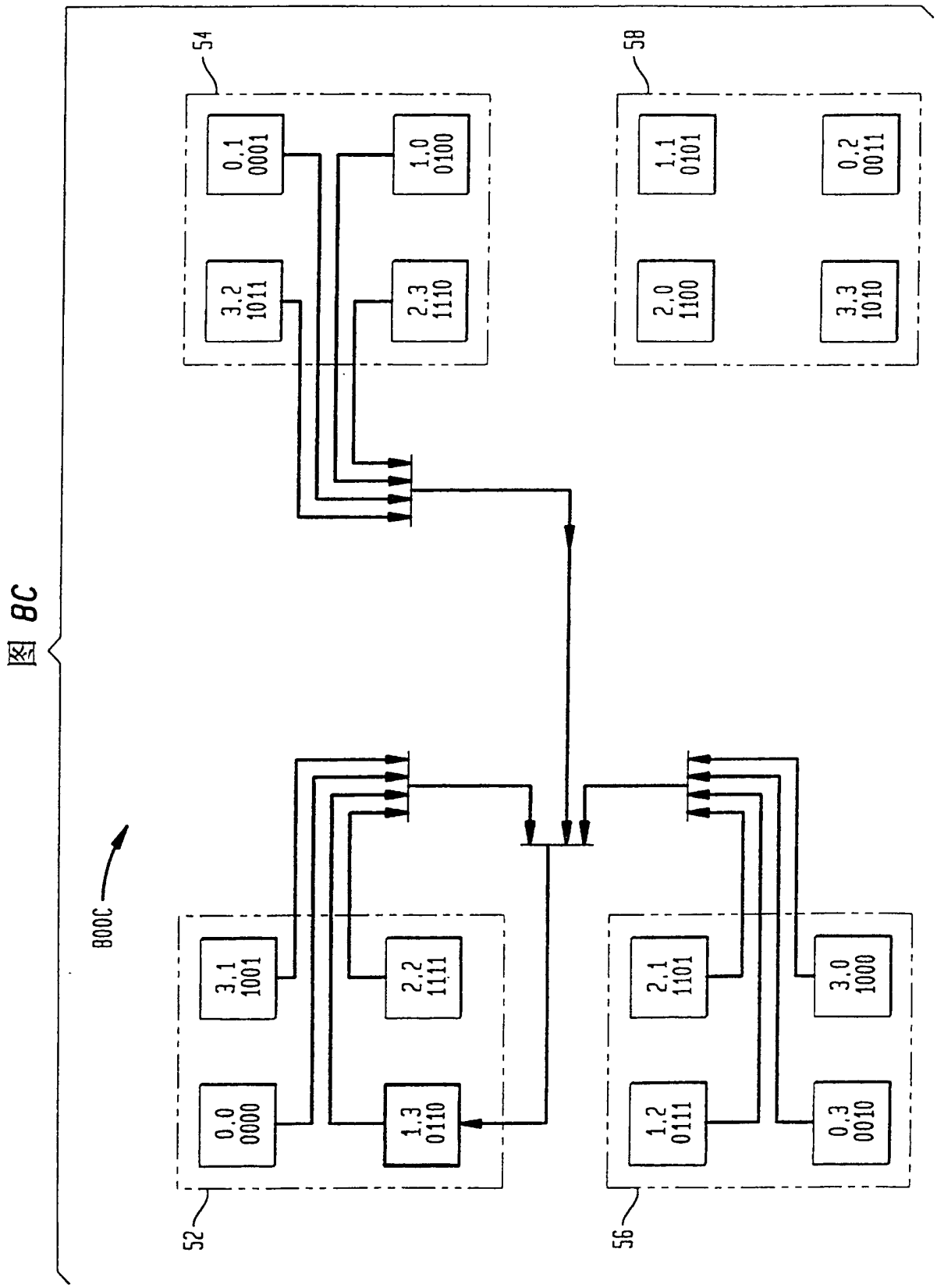

300C
52
54
56
58
0.0 0000
0.1 0001
1.0 0100
1.1 0101
0.2 0011
1.2 0111
1.3 0110
2.0 1100
2.1 1101
2.2 1111
2.3 1110
3.0 1000
3.1 1001
3.2 1011
3.3 1010
0.3 0010

图 8C

图 9A

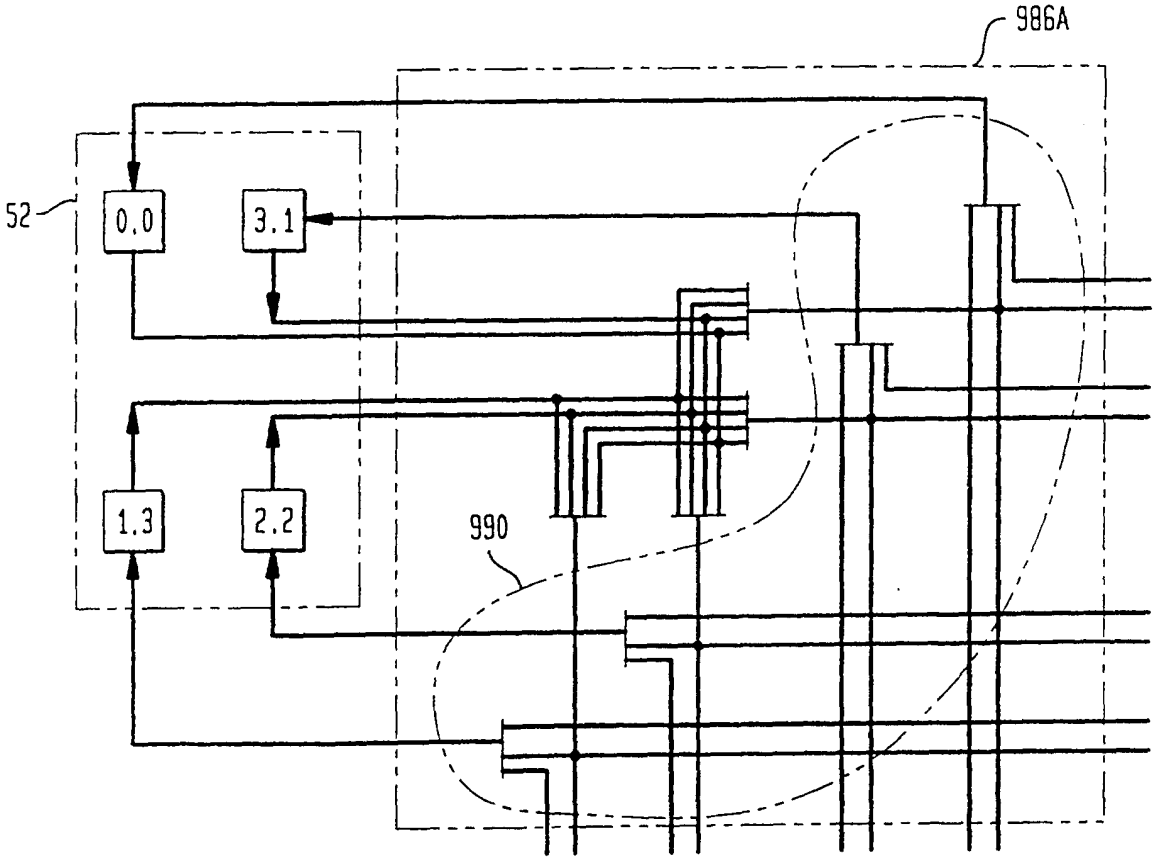
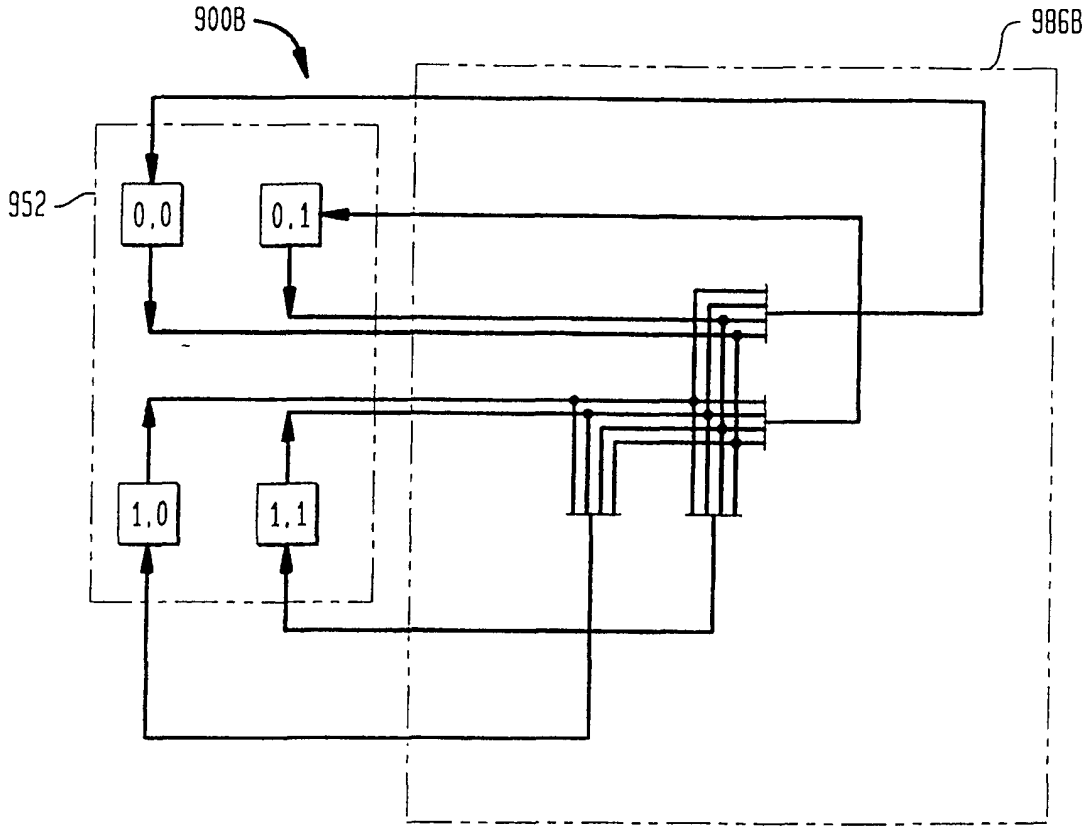


图 9B



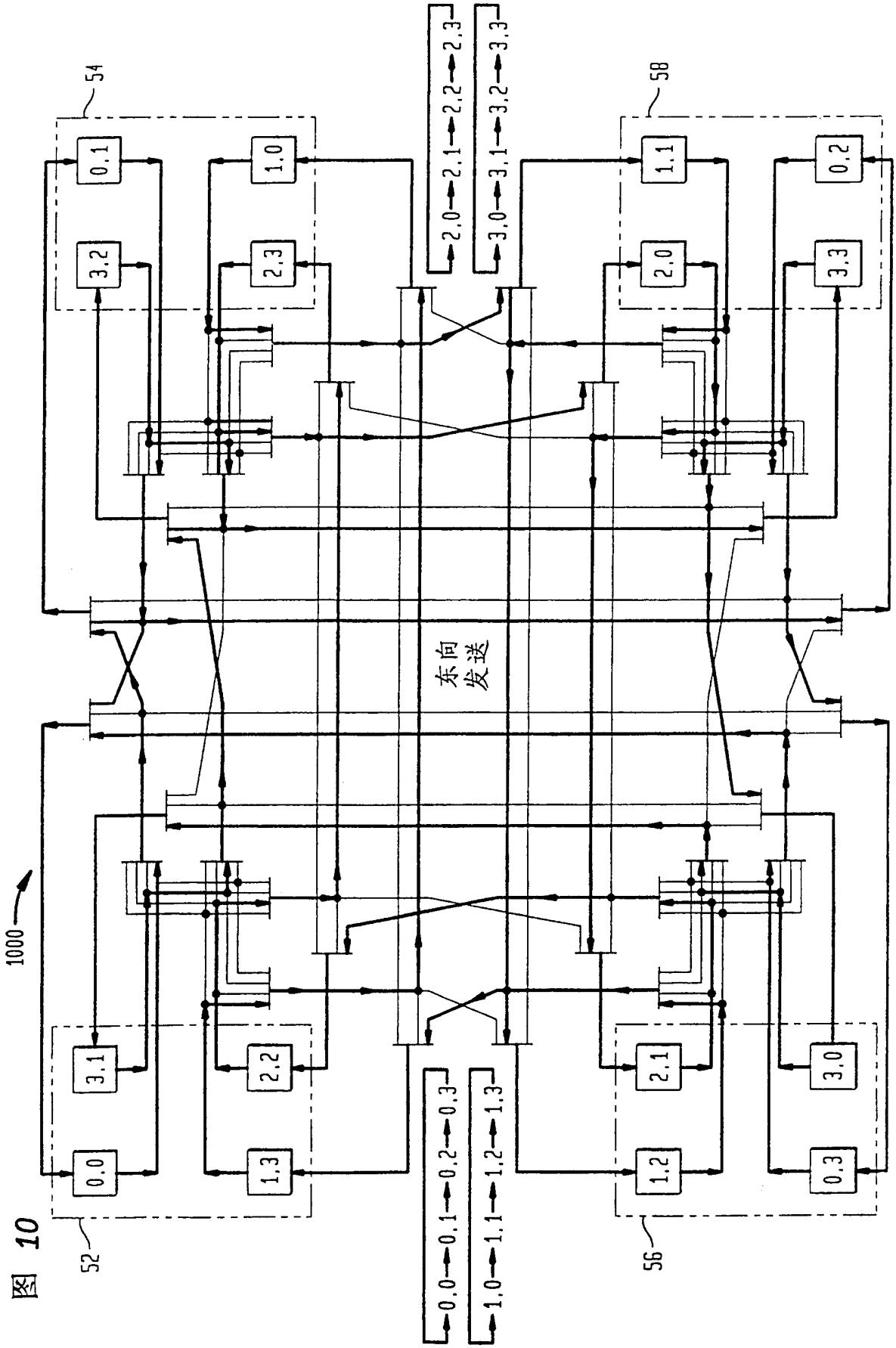


图 10

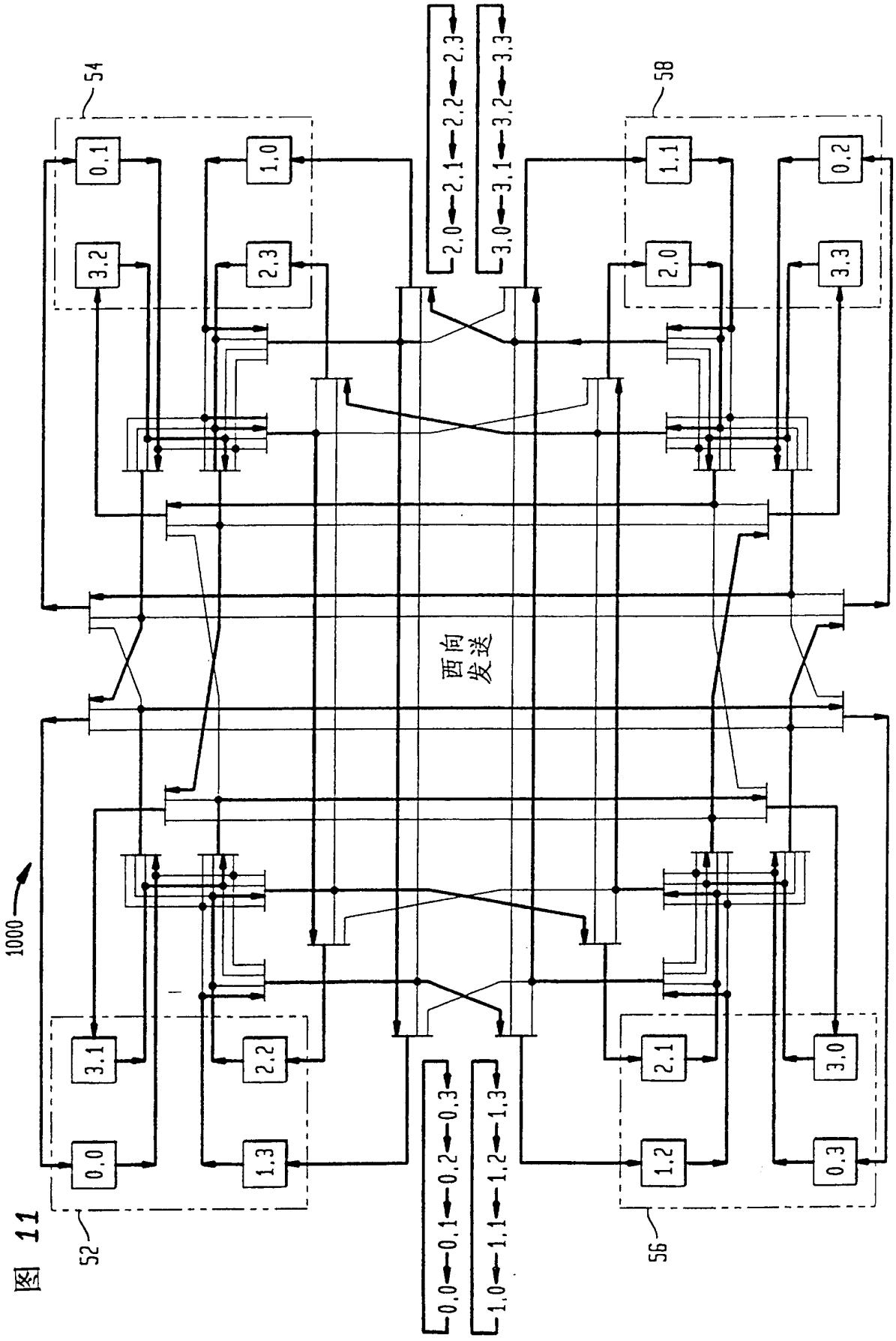

1000
54
58
52
56
西向
发送
0,1
1,0
2,3
3,2
0,0
1,3
2,2
3,1
2,0
2,1
2,2
2,3
3,0
3,1
3,2
3,3
0,0
0,1
0,2
0,3
1,0
1,1
1,2
1,3
0,1
1,0
2,3
3,2
0,0
1,3
2,2
3,1
1,1
0,2
2,0
3,3
2,1
3,0
1,2
0,3

图 11

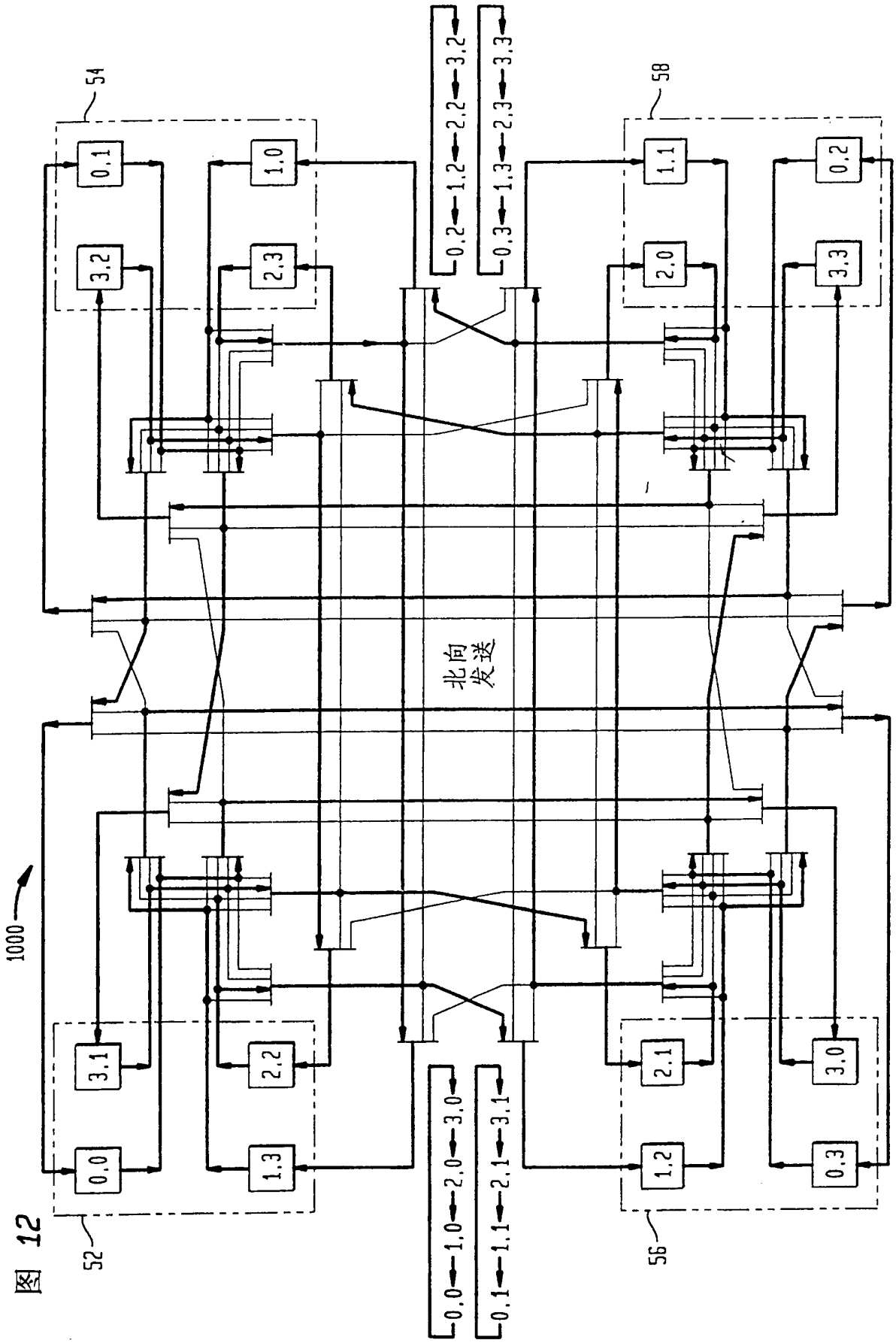

1000
54
58
52
56
北向
发送
0.1
1.0
3.2
2.3
1.1
0.2
2.0
3.3
3.1
2.2
0.0
1.3
2.1
3.0
1.2
0.3
0.2→1.2→2.2→3.2
0.3→1.3→2.3→3.3
0.0→1.0→2.0→3.0
0.1→1.1→2.1→3.1

图 12

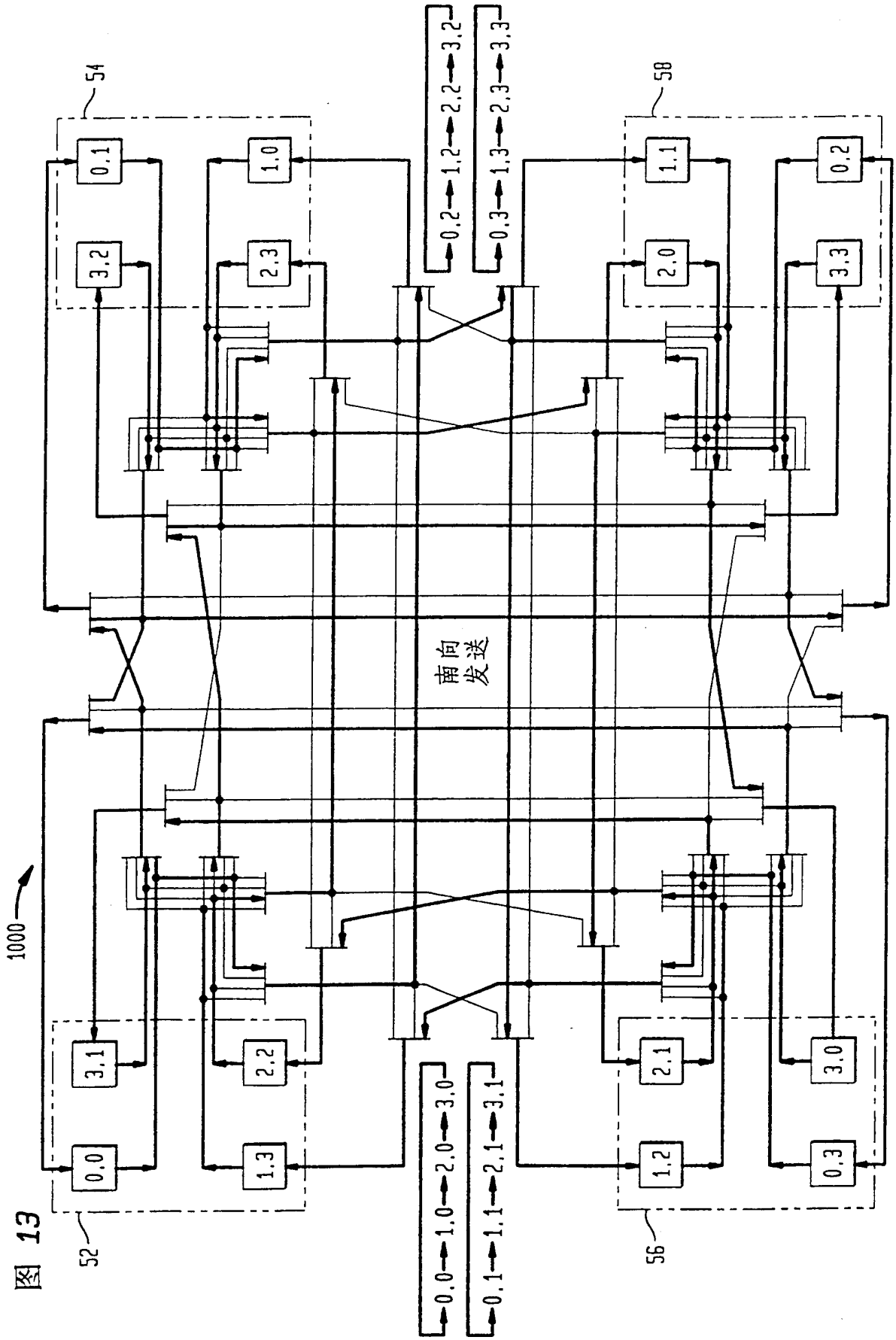

1000
54
58
52
56
南向
发送
0.1
1.0
3.2
2.3
1.1
0.2
2.0
3.3
3.1
2.2
2.1
3.0
0.0
1.3
1.2
0.3
0,2→1,2→2,2→3,2
0,3→1,3→2,3→3,3
0,0→1,0→2,0→3,0
0,1→1,1→2,1→3,1

图 13

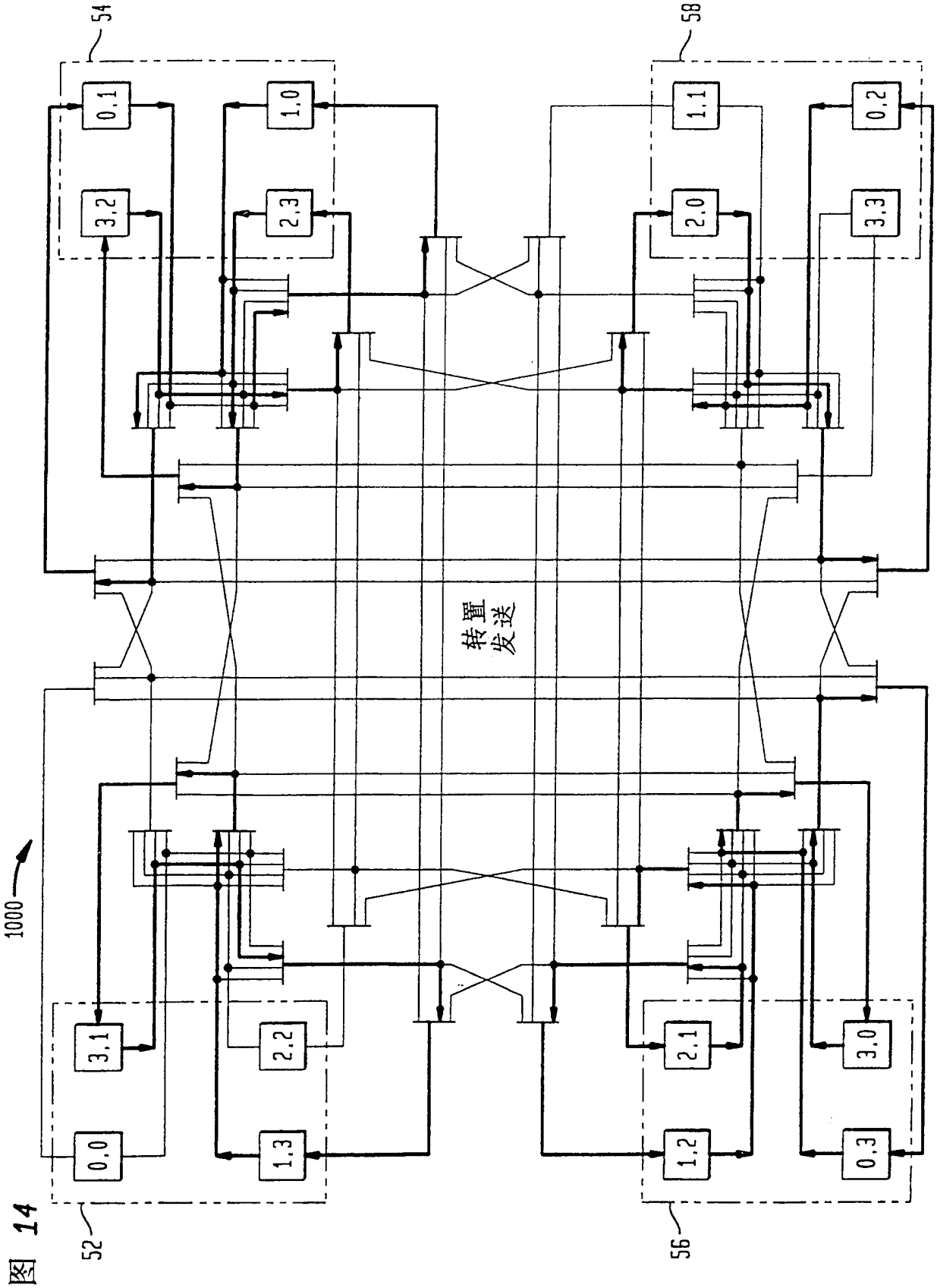


图 14

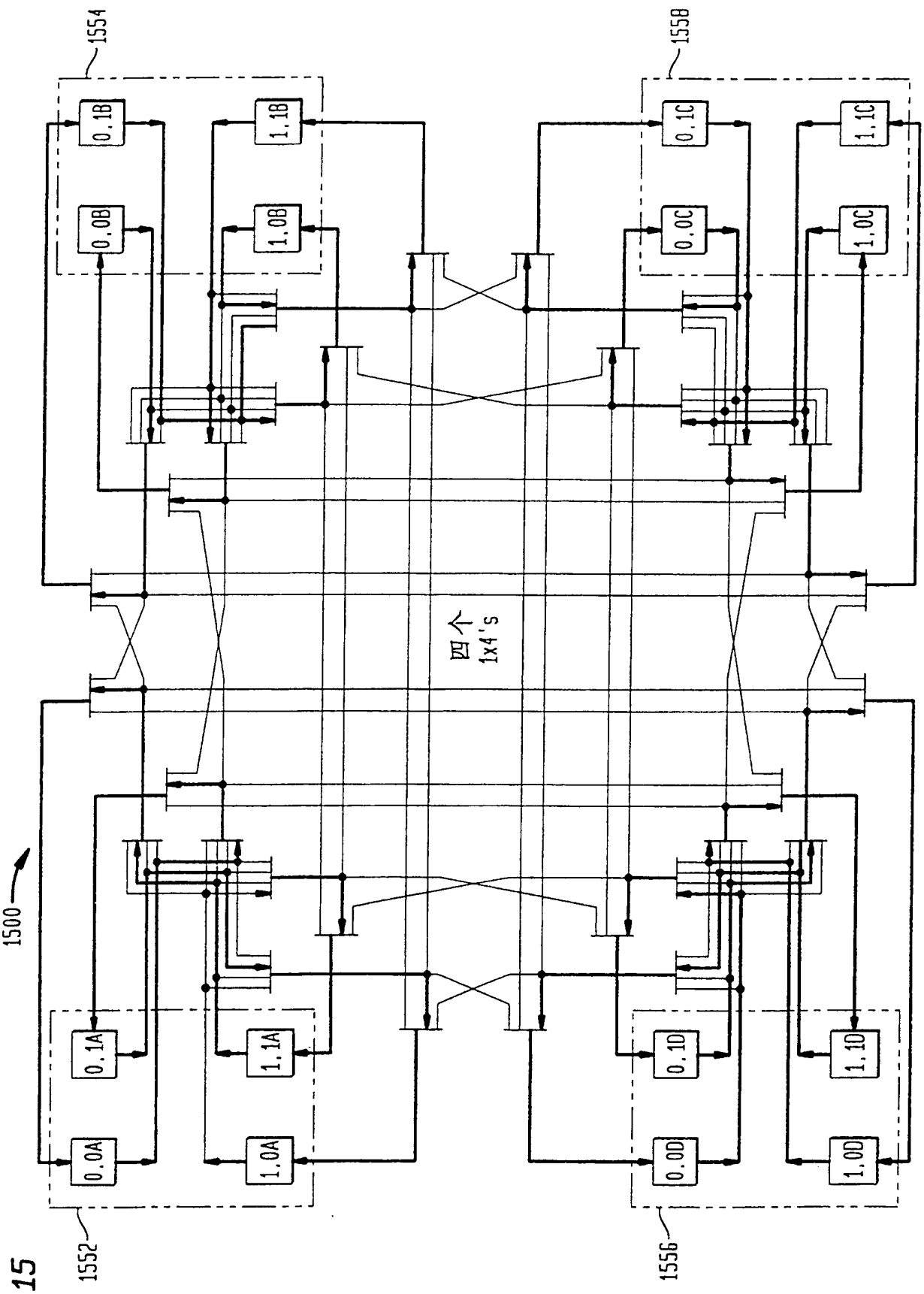

1554
1558
0.1B
1.1B
0.0B
1.0B
0.1C
1.1C
0.0C
1.0C
四个
1x4's
1500
0.1A
1.1A
0.0A
1.0A
0.1D
1.1D
0.0D
1.0D
1552
1556

图 15

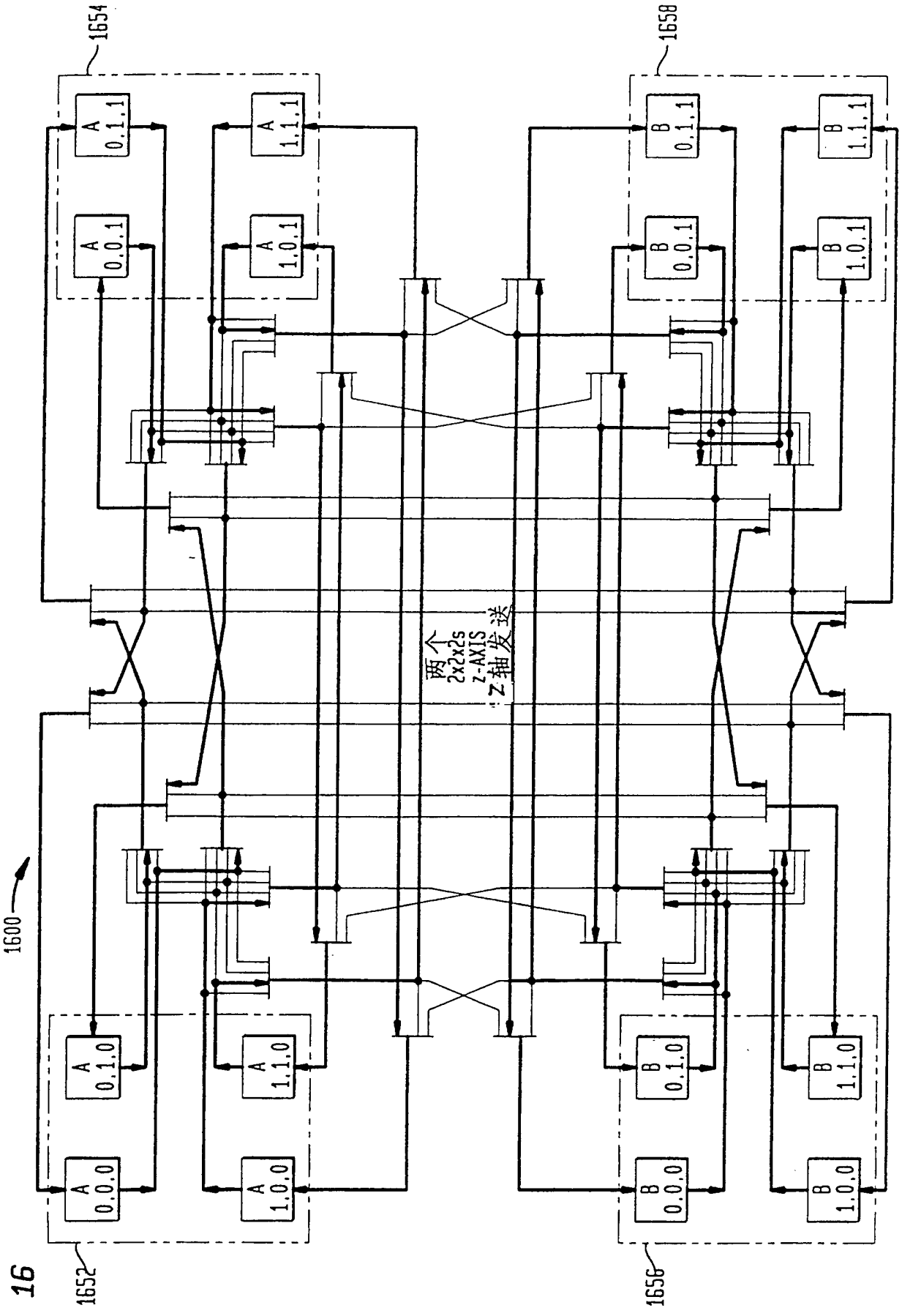
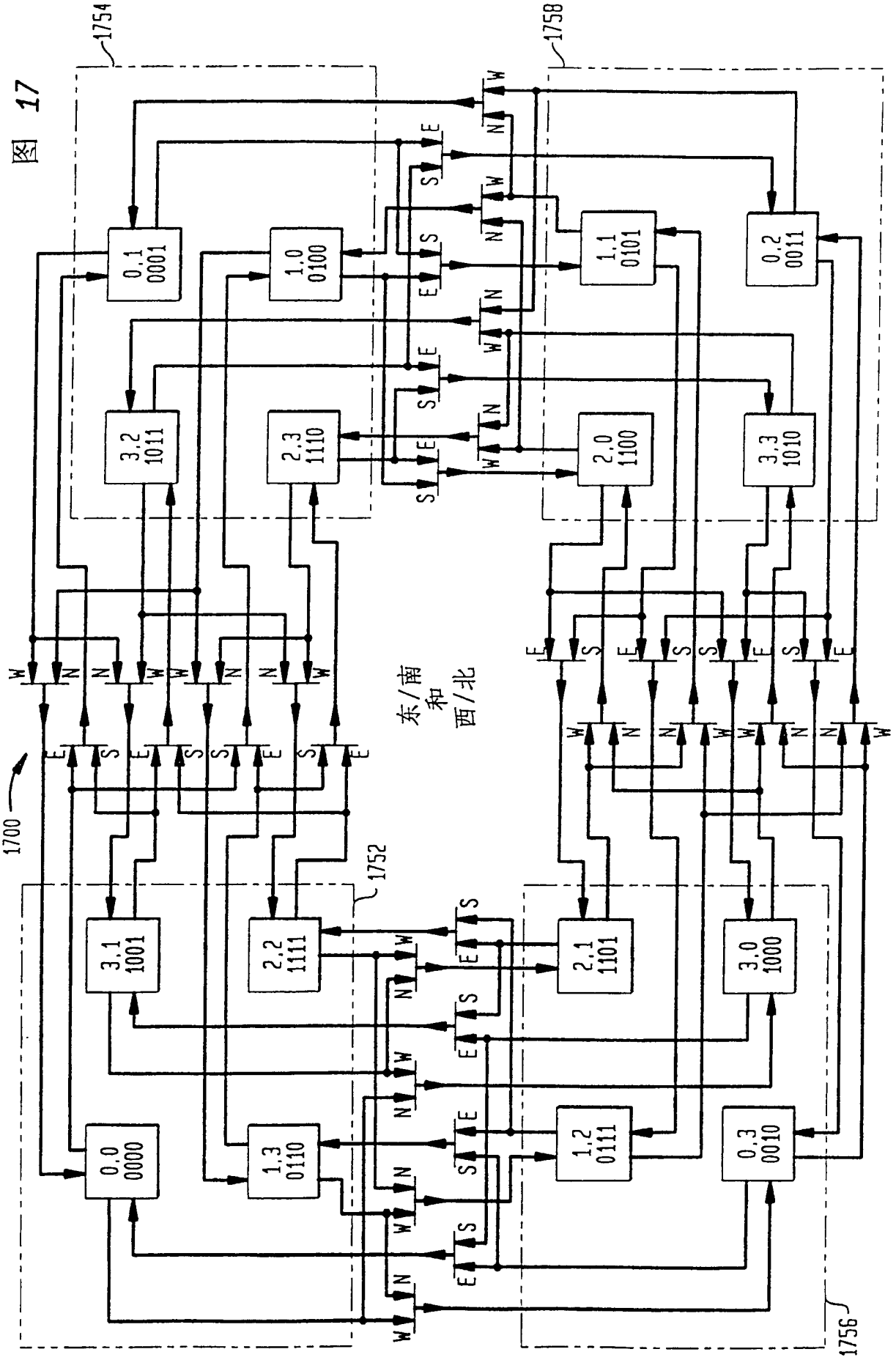

1654
1658
1600
1652
1656
A 0,1,1
A 1,1,1
A 0,0,1
A 1,0,1
B 0,1,1
B 1,1,1
B 0,0,1
B 1,0,1
A 0,1,0
A 1,1,0
A 0,0,0
A 1,0,0
B 0,1,0
B 1,1,0
B 0,0,0
B 1,0,0
两个
2x2x2s
Z-AXIS
Z轴发送

图 16

图 17


1754
1758
1700
1752
1756
0,1 0001
1,0 0100
1,1 0101
0,2 0011
3,2 1011
2,3 1110
2,0 1100
3,3 1010
3,1 1001
2,2 1111
2,1 1101
3,0 1000
0,0 0000
1,3 0110
1,2 0111
0,3 0010
东/南
和
西/北

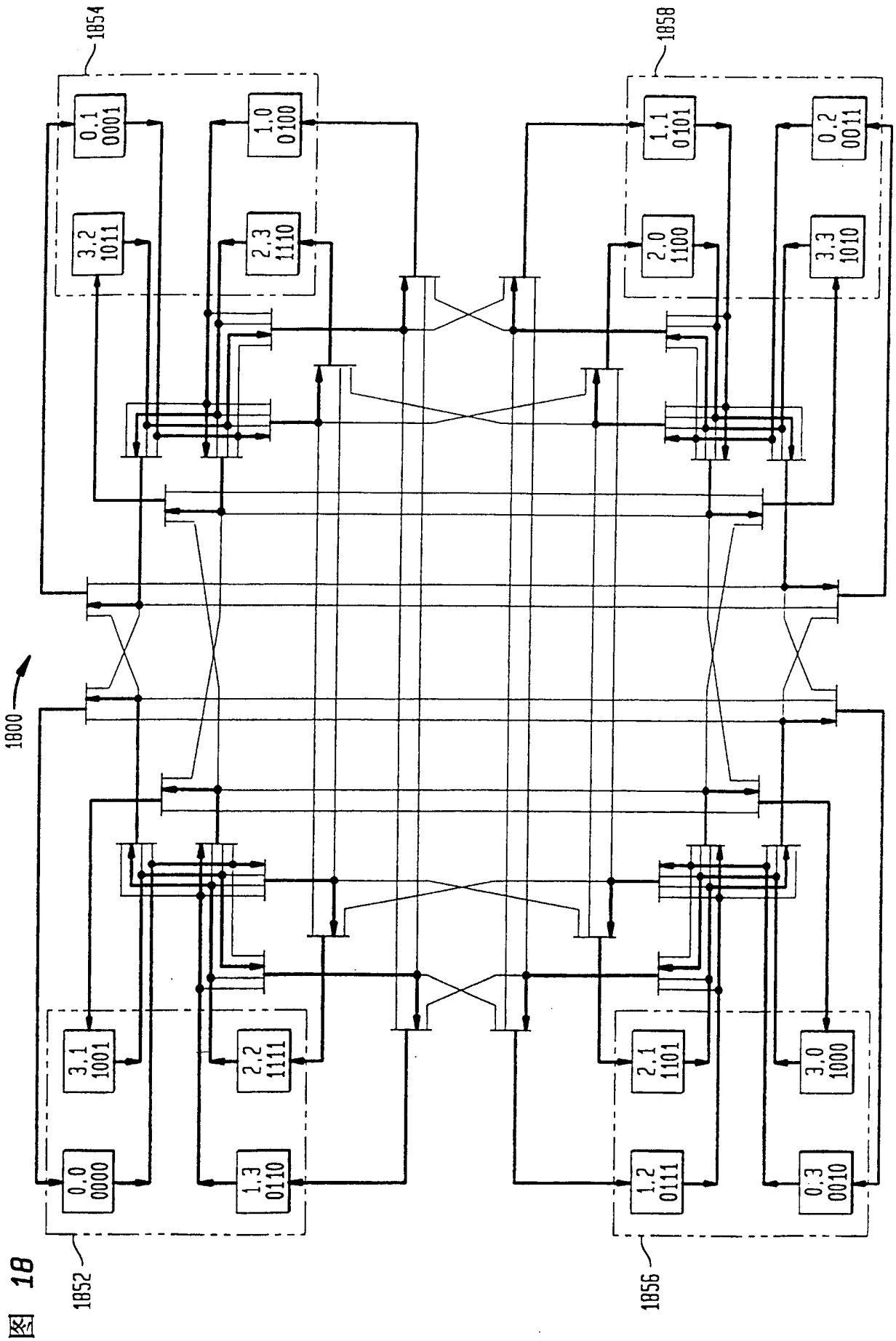

1800
1852
1854
1856
1858
0.0 0000
0.1 0001
1.0 0100
1.1 0101
0.2 0011
3.1 1001
3.2 1011
2.3 1110
2.0 1100
3.3 1010
2.2 1111
2.1 1101
3.0 1000
1.3 0110
1.2 0111
0.3 0010

图 18

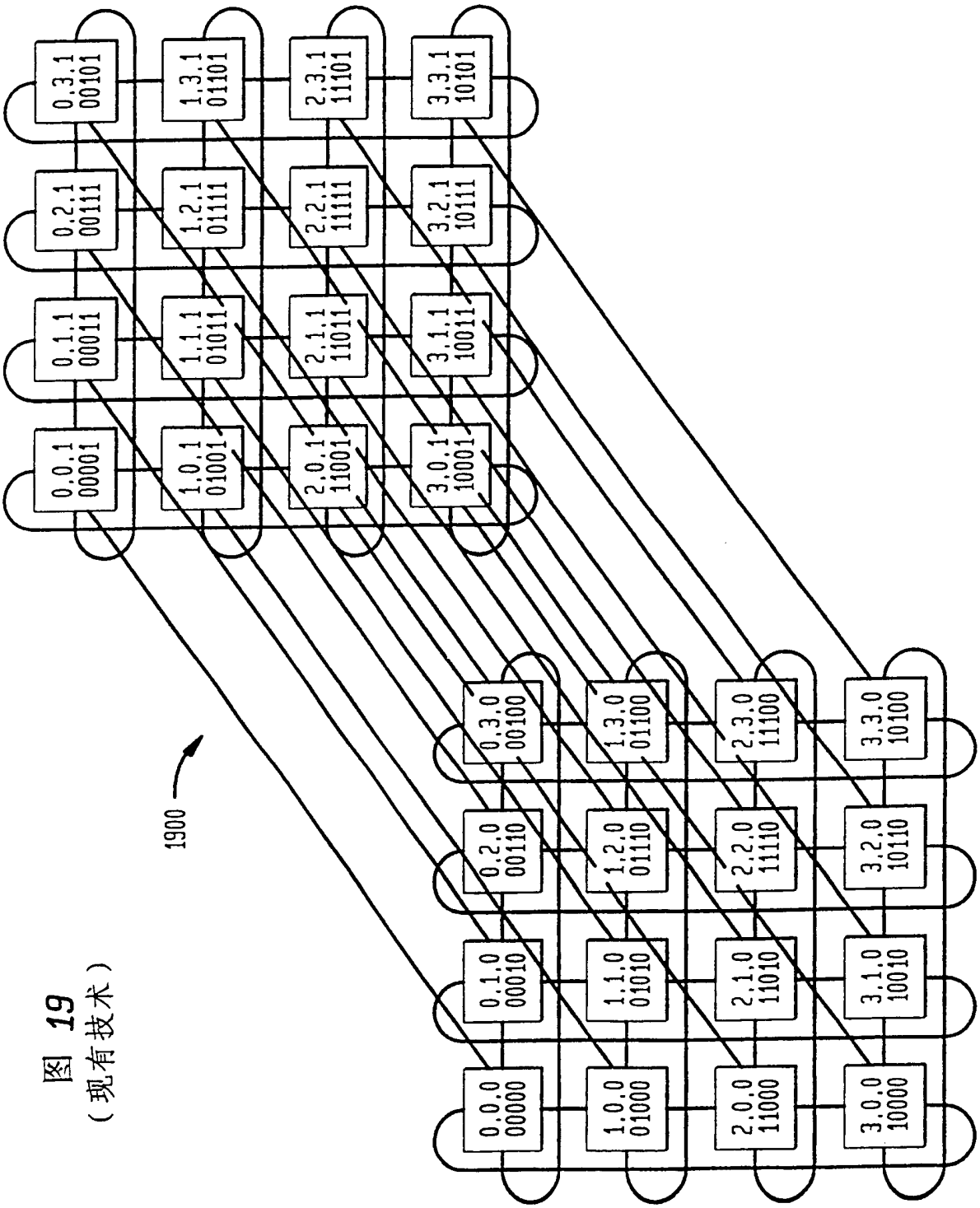

1900
0,0,0 00000
1,0,0 01000
2,0,0 11000
3,0,0 10000
0,1,0 00010
1,1,0 01010
2,1,0 11010
3,1,0 10010
0,2,0 00110
1,2,0 01110
2,2,0 11110
3,2,0 10110
0,3,0 00100
1,3,0 01100
2,3,0 11100
3,3,0 10100
0,0,1 00001
1,0,1 01001
2,0,1 11001
3,0,1 10001
0,1,1 00011
1,1,1 01011
2,1,1 11011
3,1,1 10011
0,2,1 00111
1,2,1 01111
2,2,1 11111
3,2,1 10111
0,3,1 00101
1,3,1 01101
2,3,1 11101
3,3,1 10101

图 19
（现有技术）

图 20

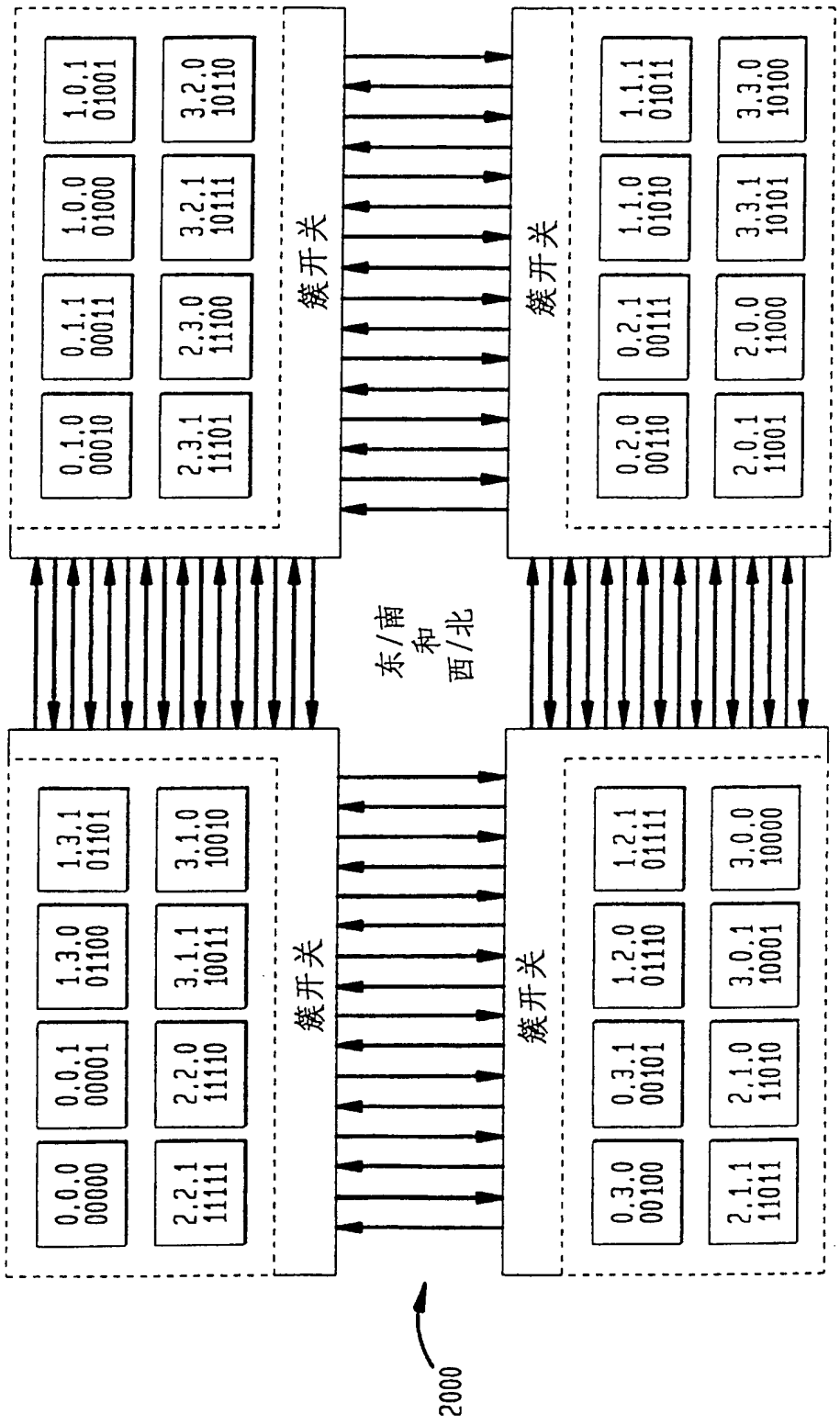

簇开关
簇开关
簇开关
簇开关
东/南
和
西/北
2000
0,0,0 00000
0,0,1 00001
1,3,0 01100
1,3,1 01101
2,2,1 11111
2,2,0 11110
3,1,1 10011
3,1,0 10010
0,3,0 00100
0,3,1 00101
1,2,0 01110
1,2,1 01111
2,1,1 11011
2,1,0 11010
3,0,1 10001
3,0,0 10000
0,1,0 00010
0,1,1 00011
1,0,0 01000
1,0,1 01001
2,3,1 11101
2,3,0 11100
3,2,1 10111
3,2,0 10110
0,2,0 00110
0,2,1 00111
1,1,0 01010
1,1,1 01011
2,0,1 11001
2,0,0 11000
3,3,1 10101
3,3,0 10100

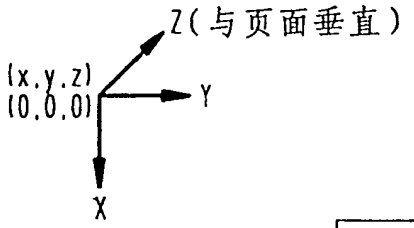
图 21A

1	2	3	4
PE-0,0 0000	PE-0,1 0001	PE-0,2 0011	PE-0,3 0010
PE-1,0 0100	PE-1,1 0101	PE-1,2 0111	PE-1,3 0110
PE-2,0 1100	PE-2,1 1101	PE-2,2 1111	PE-2,3 1110
PE-3,0 1000	PE-3,1 1001	PE-3,2 1011	PE-3,3 1010

图 21B

1	2	3	4
PE-0,0 0000	PE-1,1 0101	PE-2,2 1111	PE-3,3 1010
PE-1,0 0100	PE-2,1 1101	PE-3,2 1011	PE-0,3 0010
PE-2,0 1100	PE-3,1 1001	PE-0,2 0011	PE-1,3 0110
PE-3,0 1000	PE-0,1 0001	PE-1,2 0111	PE-2,3 1110

图 21C



A

PE-0,0 0,0,0 000000	PE-0,1 0,1,0 000100	PE-0,2 0,2,0 001100	PE-0,3 0,3,0 001000
PE-1,0 1,0,0 010000	PE-1,1 1,1,0 010100	PE-1,2 1,2,0 011100	PE-1,3 1,3,0 011000
PE-2,0 2,0,0 110000	PE-2,1 2,1,0 110100	PE-2,2 2,2,0 111100	PE-2,3 2,3,0 111000
PE-3,0 3,0,0 100000	PE-3,1 3,1,0 100100	PE-3,2 3,2,0 101100	PE-3,3 3,3,0 101000

B

PE-0,4 0,0,1 000001	PE-0,5 0,1,1 000101	PE-0,6 0,2,1 001101	PE-0,7 0,3,1 001001
PE-1,4 1,0,1 010001	PE-1,5 1,1,1 010101	PE-1,6 1,2,1 011101	PE-1,7 1,3,1 011001
PE-2,4 2,0,1 110001	PE-2,5 2,1,1 110101	PE-2,6 2,2,1 111101	PE-2,7 2,3,1 111001
PE-3,4 3,0,1 100001	PE-3,5 3,1,1 100101	PE-3,6 3,2,1 101101	PE-3,7 3,3,1 101001

C

PE-4,4 0,0,2 000011	PE-4,5 0,1,2 000111	PE-4,6 0,2,2 001111	PE-4,7 0,3,2 001011
PE-5,4 1,0,2 010011	PE-5,5 1,1,2 010111	PE-5,6 1,2,2 011111	PE-5,7 1,3,2 011011
PE-6,4 2,0,2 110011	PE-6,5 2,1,2 110111	PE-6,6 2,2,2 111111	PE-6,7 2,3,2 111011
PE-7,4 3,0,2 100011	PE-7,5 3,1,2 100111	PE-7,6 3,2,2 101111	PE-7,7 3,3,2 101011

D

PE-4,0 0,0,3 000010	PE-4,1 0,1,3 000110	PE-4,2 0,2,3 001110	PE-4,3 0,3,3 001010
PE-5,0 1,0,3 010010	PE-5,1 1,1,3 010110	PE-5,2 1,2,3 011110	PE-5,3 1,3,3 011010
PE-6,0 2,0,3 110010	PE-6,1 2,1,3 110110	PE-6,2 2,2,3 111110	PE-6,3 2,3,3 111010
PE-7,0 3,0,3 100010	PE-7,1 3,1,3 100110	PE-7,2 3,2,3 101110	PE-7,3 3,3,3 101010

图 22

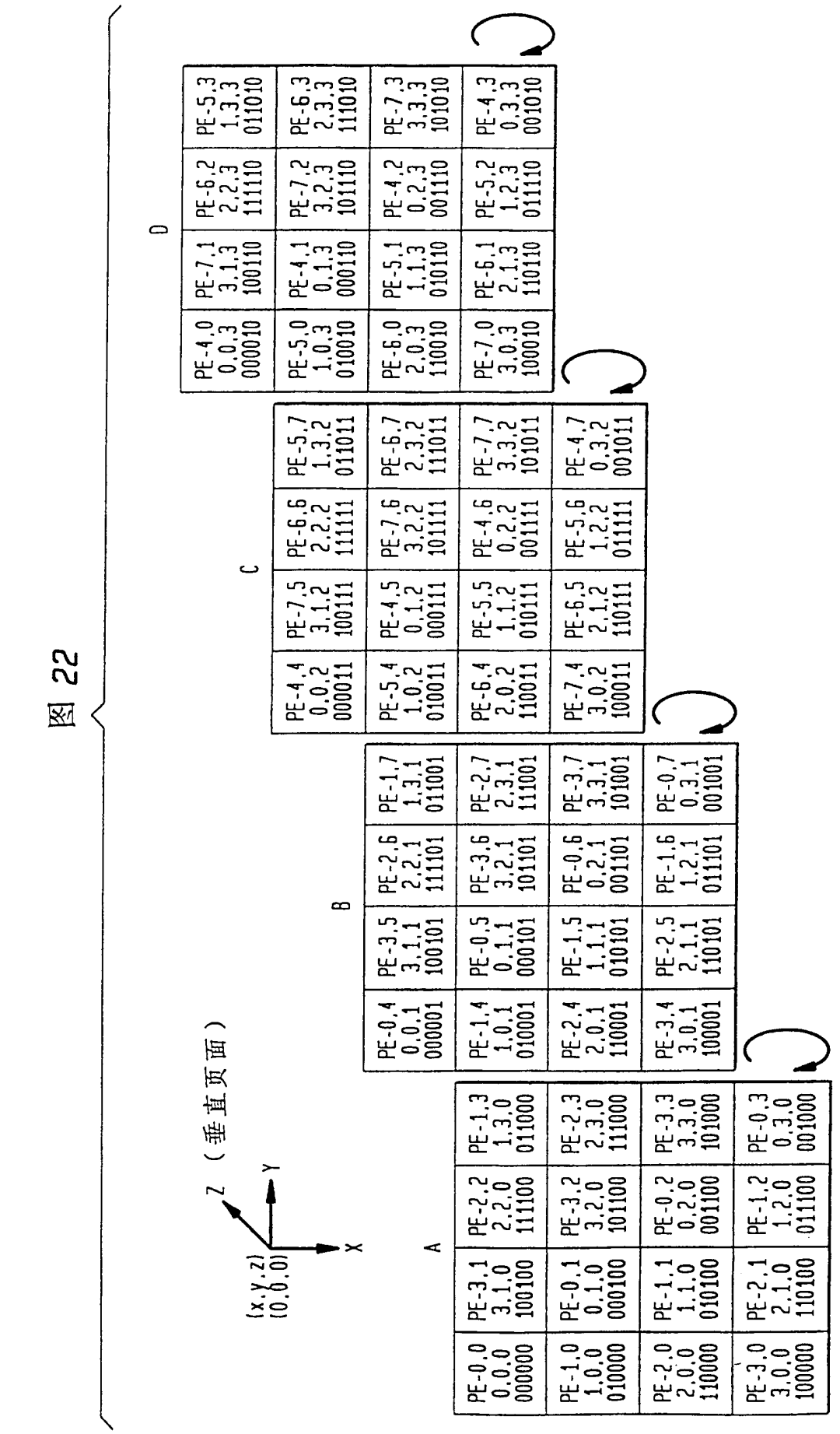
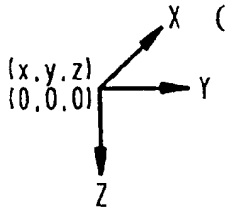


图 23



A

PE-0,0 0,0,0 000000	PE-3,1 3,1,0 100100	PE-2,2 2,2,0 111100	PE-1,3 1,3,0 011000
PE-0,4 0,0,1 000001	PE-3,5 3,1,1 100101	PE-2,6 2,2,1 111101	PE-1,7 1,3,1 011001
PE-4,4 0,0,2 000011	PE-7,5 3,1,2 100111	PE-6,6 2,2,2 111111	PE-5,7 1,3,2 011011
PE-4,0 0,0,3 000010	PE-7,1 3,1,3 100110	PE-6,2 2,2,3 111110	PE-5,3 1,3,3 011010

B

PE-1,0 1,0,0 010000	PE-0,1 0,1,0 000100	PE-3,2 3,2,0 101100	PE-2,3 2,3,0 111000
PE-1,4 1,0,1 010001	PE-0,5 0,1,1 000101	PE-3,6 3,2,1 101101	PE-2,7 2,3,1 111001
PE-5,4 1,0,2 010011	PE-4,5 0,1,2 000111	PE-7,6 3,2,2 101111	PE-6,7 2,3,2 111011
PE-5,0 1,0,3 010010	PE-4,1 0,1,3 000110	PE-7,2 3,2,3 101110	PE-6,3 2,3,3 111010

C

PE-2,0 2,0,0 110000	PE-1,1 1,1,0 010100	PE-0,2 0,2,0 001100	PE-3,3 3,3,0 101000
PE-2,4 2,0,1 110001	PE-1,5 1,1,1 010101	PE-0,6 0,2,1 001101	PE-3,7 3,3,1 101001
PE-6,4 2,0,2 110011	PE-5,5 1,1,2 010111	PE-4,6 0,2,2 001111	PE-7,7 3,3,2 101011
PE-6,0 2,0,3 110010	PE-5,1 1,1,3 010110	PE-4,2 0,2,3 001110	PE-7,3 3,3,3 101010

D

PE-3,0 3,0,0 100000	PE-2,1 2,1,0 110100	PE-1,2 1,2,0 011100	PE-0,3 0,3,0 001000
PE-3,4 3,0,1 100001	PE-2,5 2,1,1 110101	PE-1,6 1,2,1 011101	PE-0,7 0,3,1 001001
PE-7,4 3,0,2 100011	PE-6,5 2,1,2 110111	PE-5,6 1,2,2 011111	PE-4,7 0,3,2 001011
PE-7,0 3,0,3 100010	PE-6,1 2,1,3 110110	PE-5,2 1,2,3 011110	PE-4,3 0,3,3 001010

图 24

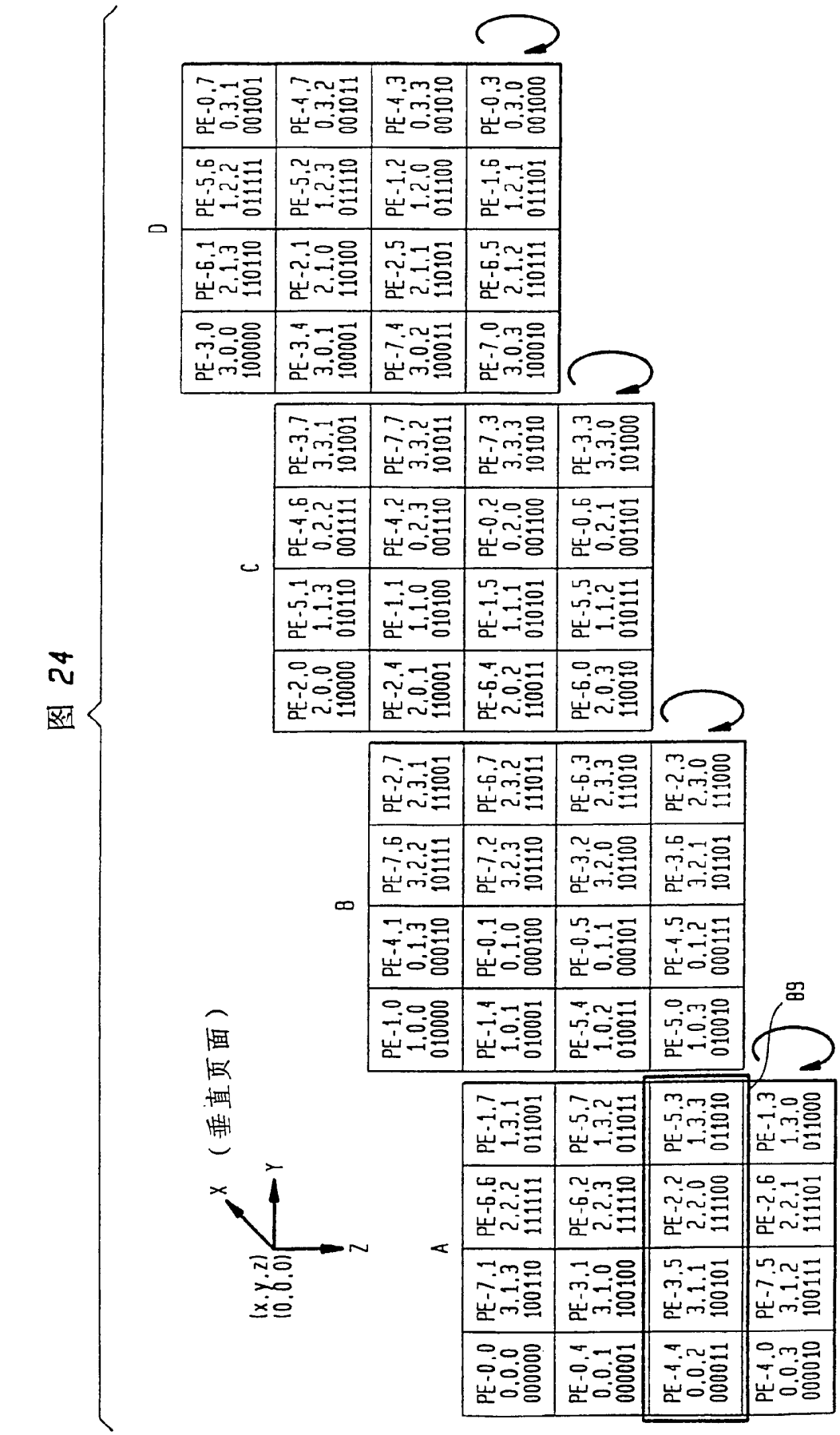


图 25

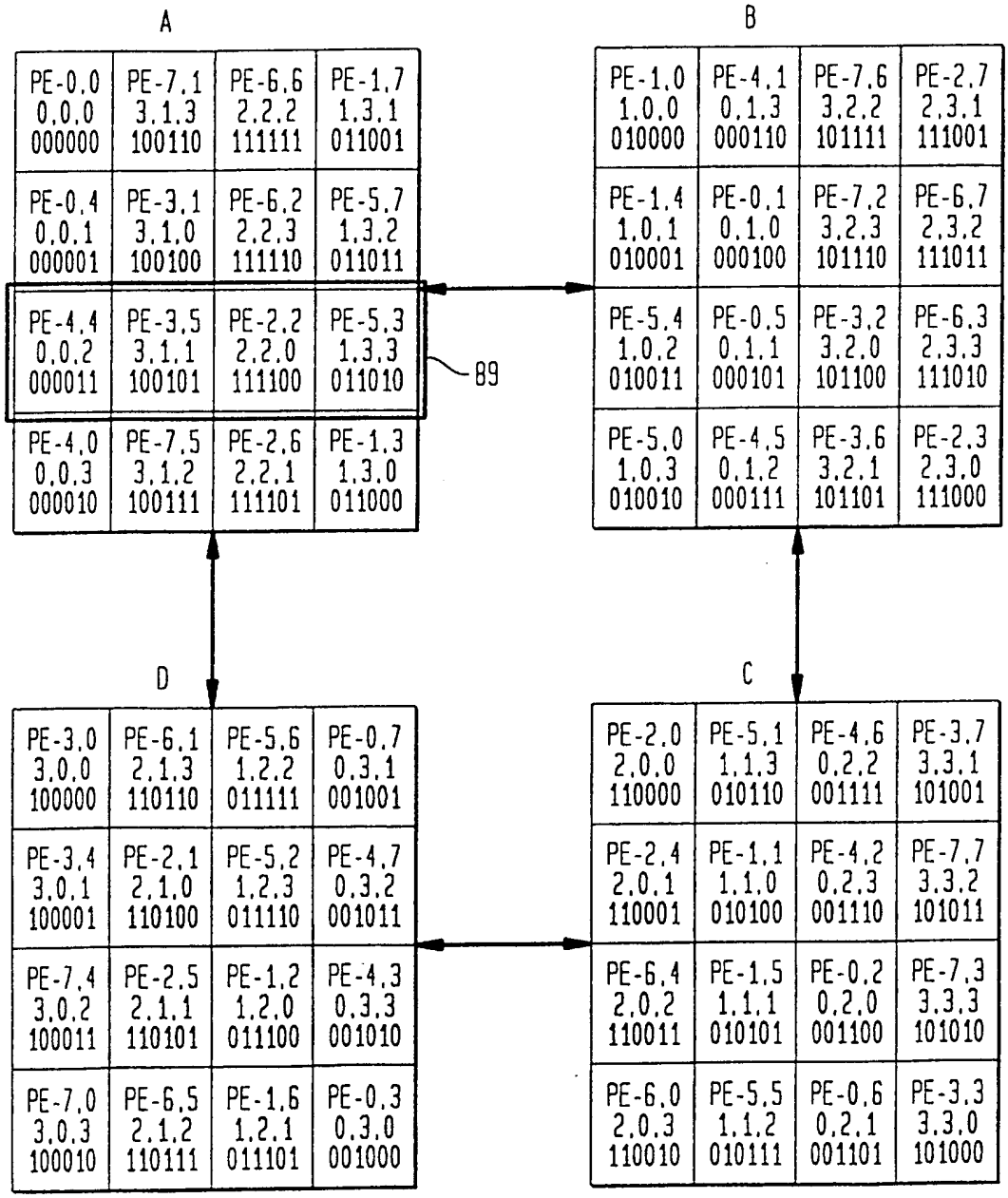
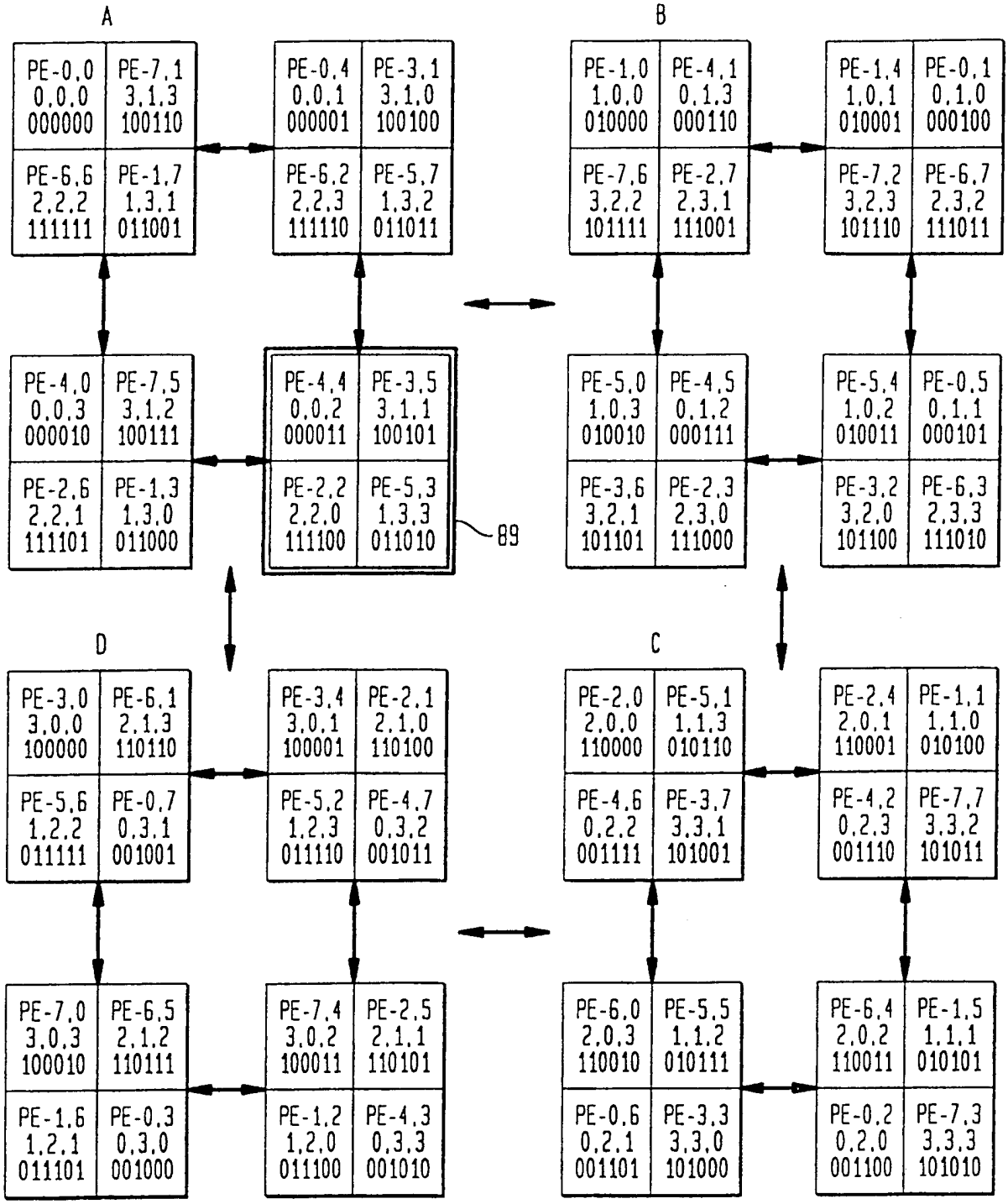

A
B
C
D
89
PE-0,0 0,0,0 000000
PE-7,1 3,1,3 100110
PE-6,6 2,2,2 111111
PE-1,7 1,3,1 011001
PE-0,4 0,0,1 000001
PE-3,1 3,1,0 100100
PE-6,2 2,2,3 111110
PE-5,7 1,3,2 011011
PE-4,4 0,0,2 000011
PE-3,5 3,1,1 100101
PE-2,2 2,2,0 111100
PE-5,3 1,3,3 011010
PE-4,0 0,0,3 000010
PE-7,5 3,1,2 100111
PE-2,6 2,2,1 111101
PE-1,3 1,3,0 011000
PE-1,0 1,0,0 010000
PE-4,1 0,1,3 000110
PE-7,6 3,2,2 101111
PE-2,7 2,3,1 111001
PE-1,4 1,0,1 010001
PE-0,1 0,1,0 000100
PE-7,2 3,2,3 101110
PE-6,7 2,3,2 111011
PE-5,4 1,0,2 010011
PE-0,5 0,1,1 000101
PE-3,2 3,2,0 101100
PE-6,3 2,3,3 111010
PE-5,0 1,0,3 010010
PE-4,5 0,1,2 000111
PE-3,6 3,2,1 101101
PE-2,3 2,3,0 111000
PE-3,0 3,0,0 100000
PE-6,1 2,1,3 110110
PE-5,6 1,2,2 011111
PE-0,7 0,3,1 001001
PE-3,4 3,0,1 100001
PE-2,1 2,1,0 110100
PE-5,2 1,2,3 011110
PE-4,7 0,3,2 001011
PE-7,4 3,0,2 100011
PE-2,5 2,1,1 110101
PE-1,2 1,2,0 011100
PE-4,3 0,3,3 001010
PE-7,0 3,0,3 100010
PE-6,5 2,1,2 110111
PE-1,6 1,2,1 011101
PE-0,3 0,3,0 001000
PE-2,0 2,0,0 110000
PE-5,1 1,1,3 010110
PE-4,6 0,2,2 001111
PE-3,7 3,3,1 101001
PE-2,4 2,0,1 110001
PE-1,1 1,1,0 010100
PE-4,2 0,2,3 001110
PE-7,7 3,3,2 101011
PE-6,4 2,0,2 110011
PE-1,5 1,1,1 010101
PE-0,2 0,2,0 001100
PE-7,3 3,3,3 101010
PE-6,0 2,0,3 110010
PE-5,5 1,1,2 010111
PE-0,6 0,2,1 001101
PE-3,3 3,3,0 101000

图26


A
B
C
D
PE-0,0 0,0,0 000000
PE-7,1 3,1,3 100110
PE-6,6 2,2,2 111111
PE-1,7 1,3,1 011001
PE-0,4 0,0,1 000001
PE-3,1 3,1,0 100100
PE-6,2 2,2,3 111110
PE-5,7 1,3,2 011011
PE-1,0 1,0,0 010000
PE-4,1 0,1,3 000110
PE-7,6 3,2,2 101111
PE-2,7 2,3,1 111001
PE-1,4 1,0,1 010001
PE-0,1 0,1,0 000100
PE-7,2 3,2,3 101110
PE-6,7 2,3,2 111011
PE-4,0 0,0,3 000010
PE-7,5 3,1,2 100111
PE-2,6 2,2,1 111101
PE-1,3 1,3,0 011000
PE-4,4 0,0,2 000011
PE-3,5 3,1,1 100101
PE-2,2 2,2,0 111100
PE-5,3 1,3,3 011010
89
PE-5,0 1,0,3 010010
PE-4,5 0,1,2 000111
PE-3,6 3,2,1 101101
PE-2,3 2,3,0 111000
PE-5,4 1,0,2 010011
PE-0,5 0,1,1 000101
PE-3,2 3,2,0 101100
PE-6,3 2,3,3 111010
PE-3,0 3,0,0 100000
PE-6,1 2,1,3 110110
PE-5,6 1,2,2 011111
PE-0,7 0,3,1 001001
PE-3,4 3,0,1 100001
PE-2,1 2,1,0 110100
PE-5,2 1,2,3 011110
PE-4,7 0,3,2 001011
PE-2,0 2,0,0 110000
PE-5,1 1,1,3 010110
PE-4,6 0,2,2 001111
PE-3,7 3,3,1 101001
PE-2,4 2,0,1 110001
PE-1,1 1,1,0 010100
PE-4,2 0,2,3 001110
PE-7,7 3,3,2 101011
PE-7,0 3,0,3 100010
PE-6,5 2,1,2 110111
PE-1,6 1,2,1 011101
PE-0,3 0,3,0 001000
PE-7,4 3,0,2 100011
PE-2,5 2,1,1 110101
PE-1,2 1,2,0 011100
PE-4,3 0,3,3 001010
PE-6,0 2,0,3 110010
PE-5,5 1,1,2 010111
PE-0,6 0,2,1 001101
PE-3,3 3,3,0 101000
PE-6,4 2,0,2 110011
PE-1,5 1,1,1 010101
PE-0,2 0,2,0 001100
PE-7,3 3,3,3 101010

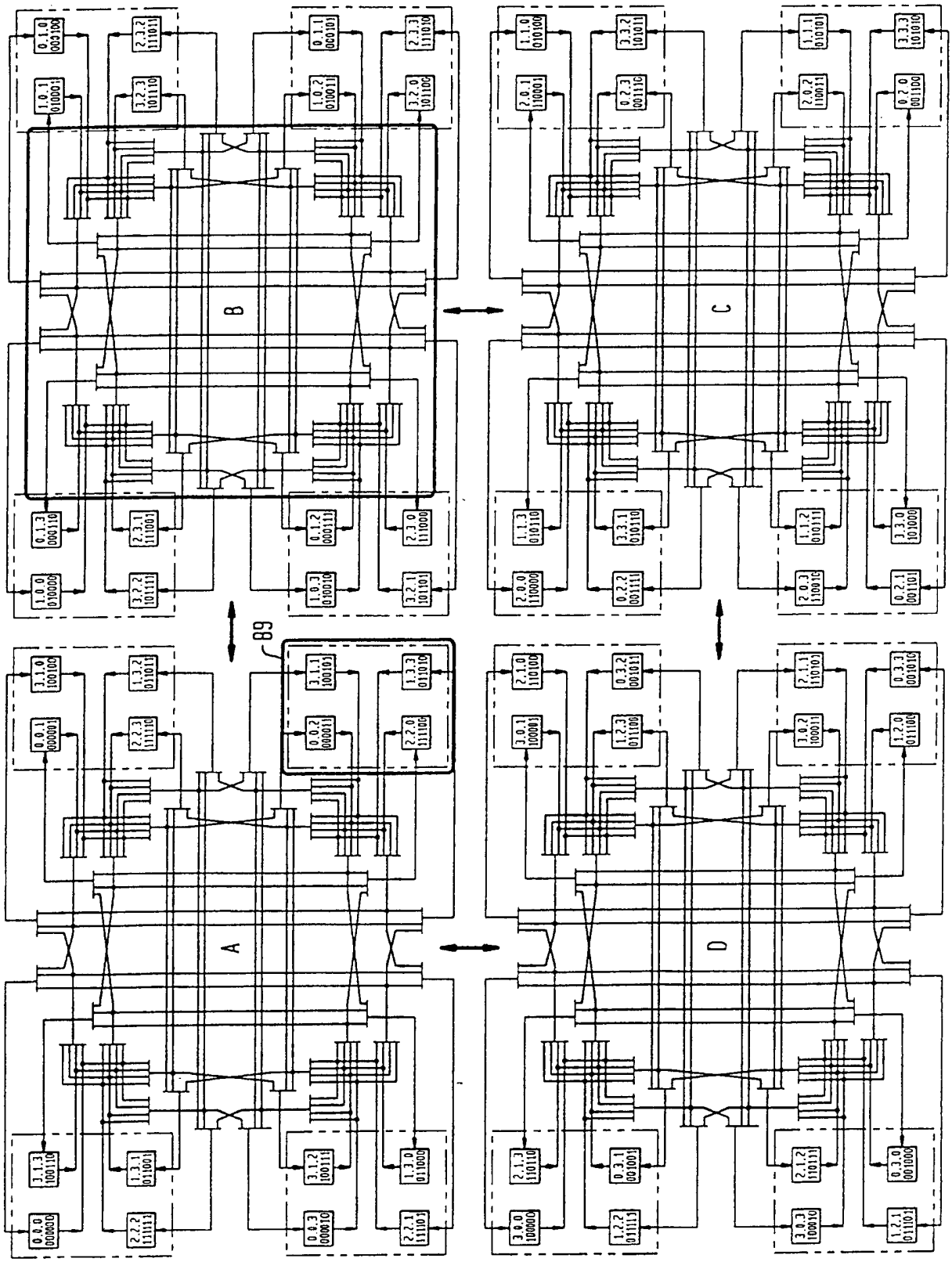


图 27A

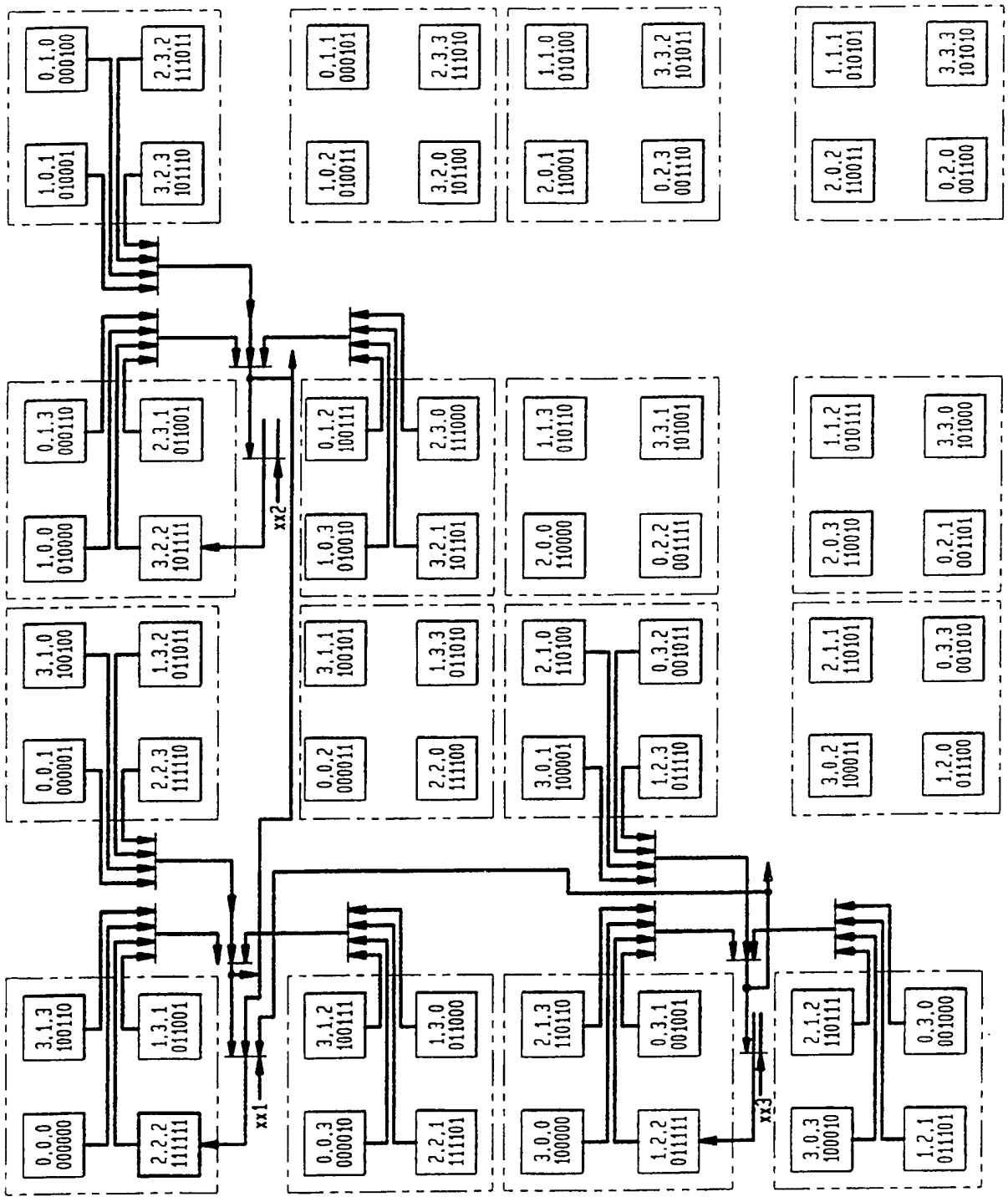

0.0.0 000000
0.0.1 000001
0.0.2 000011
0.0.3 000010
0.1.0 000100
0.1.1 000101
0.1.2 000111
0.1.3 000110
0.2.0 001100
0.2.1 001101
0.2.2 001111
0.2.3 001110
0.3.0 001000
0.3.1 001001
0.3.2 001011
0.3.3 001010
1.0.0 010000
1.0.1 010001
1.0.2 010011
1.0.3 010010
1.1.0 010100
1.1.1 010101
1.1.2 010111
1.1.3 010110
1.2.0 011100
1.2.1 011101
1.2.2 011111
1.2.3 011110
1.3.0 011000
1.3.1 011001
1.3.2 011011
1.3.3 011010
2.0.0 110000
2.0.1 110001
2.0.2 110011
2.0.3 110010
2.1.0 110100
2.1.1 110101
2.1.2 110111
2.1.3 110110
2.2.0 111100
2.2.1 111101
2.2.2 111111
2.2.3 111110
2.3.0 111000
2.3.1 111001
2.3.2 111011
2.3.3 111010
3.0.0 100000
3.0.1 100001
3.0.2 100011
3.0.3 100010
3.1.0 100100
3.1.1 100101
3.1.2 100111
3.1.3 100110
3.2.0 101100
3.2.1 101101
3.2.2 101111
3.2.3 101110
3.3.0 101000
3.3.1 101001
3.3.2 101011
3.3.3 101010
xx1
xx2
xx3

图27B

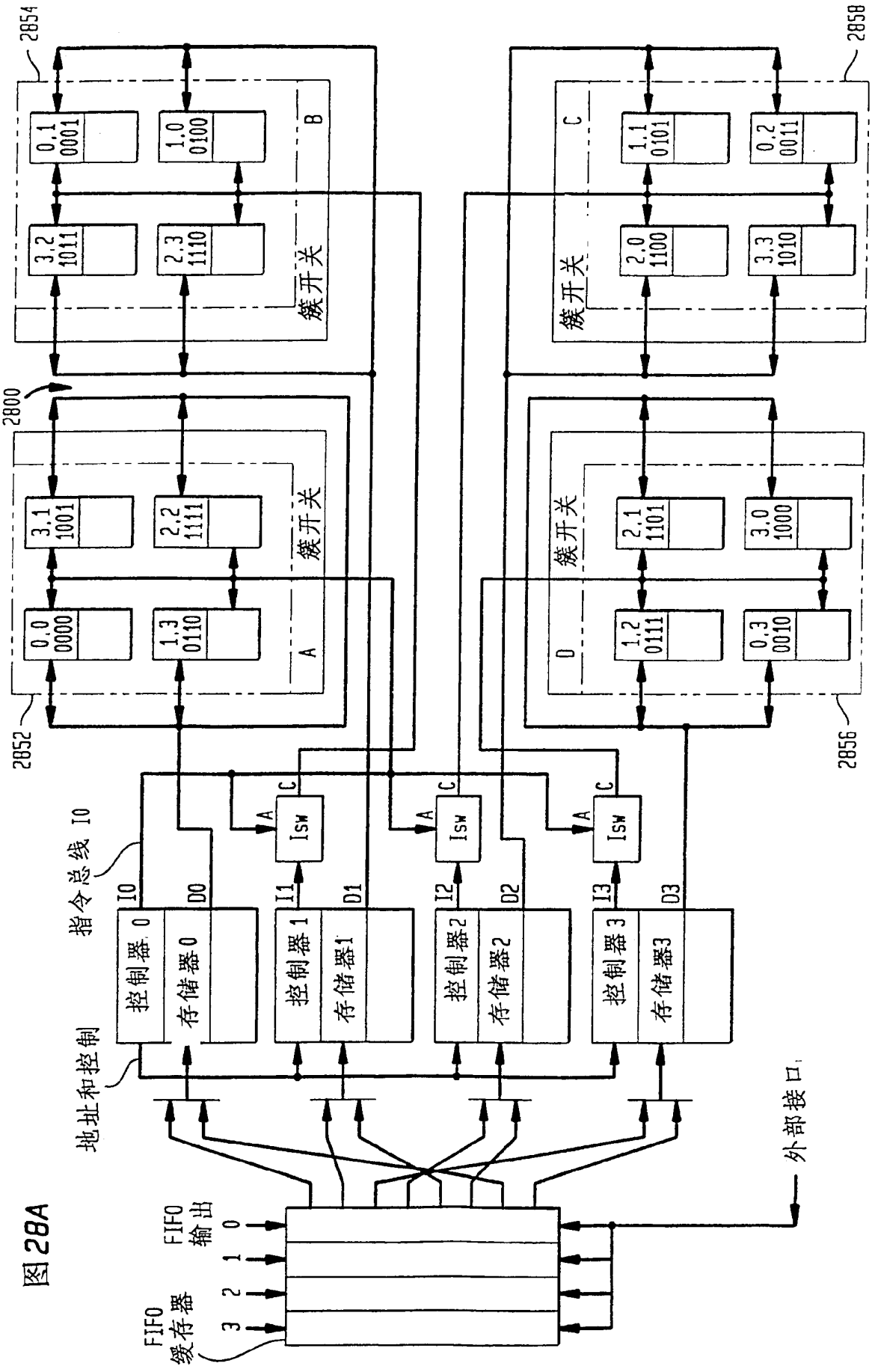

2854
2858
2800
2852
2856
簇开关
A
B
C
D
0,0 0000
1,3 0110
3,1 1001
2,2 1111
0,1 0001
1,0 0100
3,2 1011
2,3 1110
1,2 0111
0,3 0010
2,1 1101
3,0 1000
1,1 0101
0,2 0011
2,0 1100
3,3 1010
指令总线 I0
地址和控制
控制器 0
存储器 0
控制器 1
存储器1
控制器2
存储器2
控制器 3
存储器3
I0
D0
I1
D1
I2
D2
I3
D3
Isw
外部接口
FIFO 输出
FIFO 缓存器
3 2 1 0

图 28A

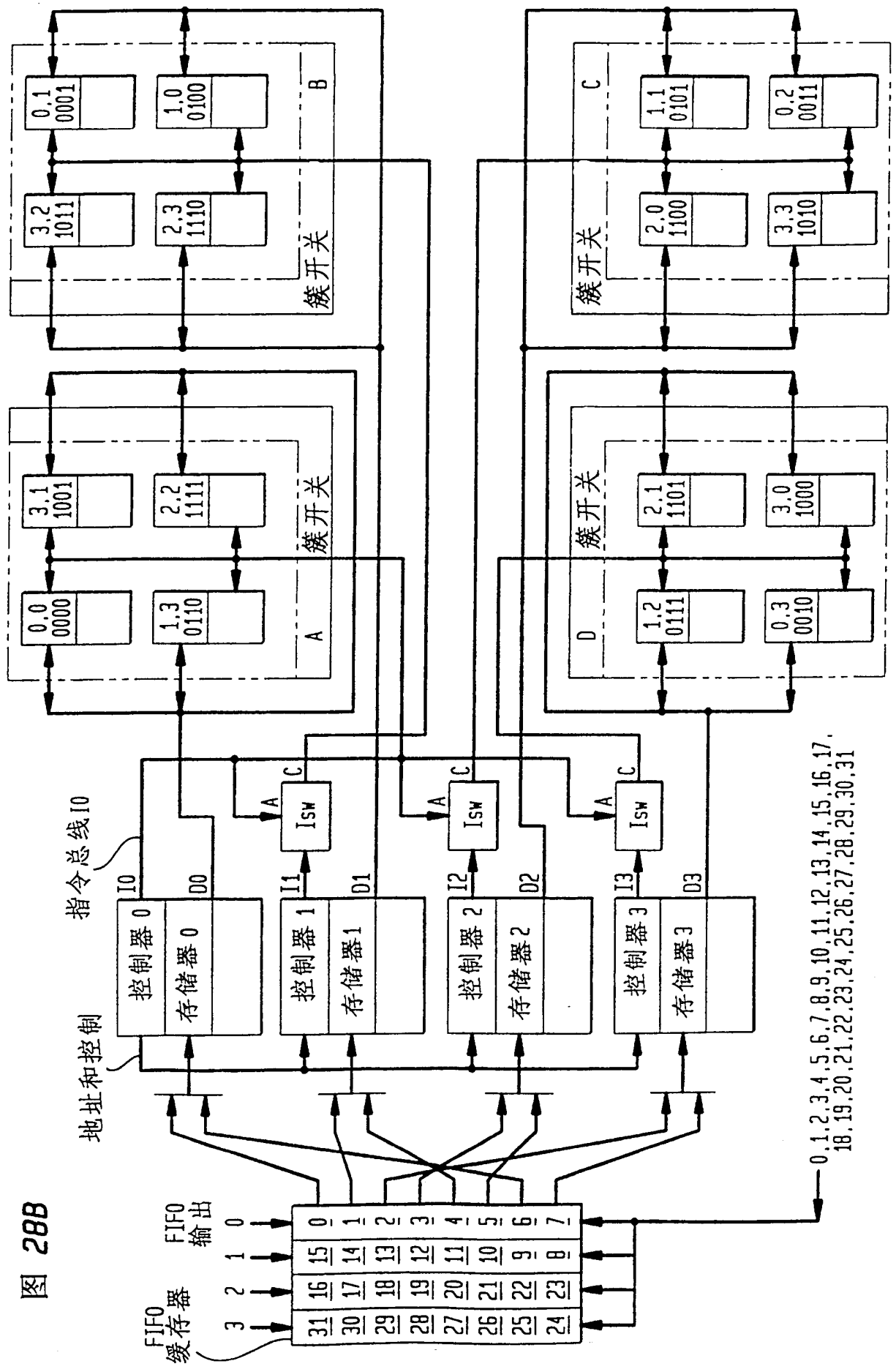

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23,24,25,26,27,28,29,30,31
FIFO
缓存器
FIFO
输出
地址和控制
指令总线I0
控制器 0
存储器 0
控制器 1
存储器1
控制器 2
存储器 2
控制器 3
存储器 3
I0
D0
I1
D1
I2
D2
I3
D3
Isw
A
C
簇开关
A
B
C
D
0,0
0000
1,3
0110
3,1
1001
2,2
1111
0,1
0001
1,0
0100
3,2
1011
2,3
1110
1,2
0111
0,3
0010
2,1
1101
3,0
1000
1,1
0101
0,2
0011
2,0
1100
3,3
1010

图 28B

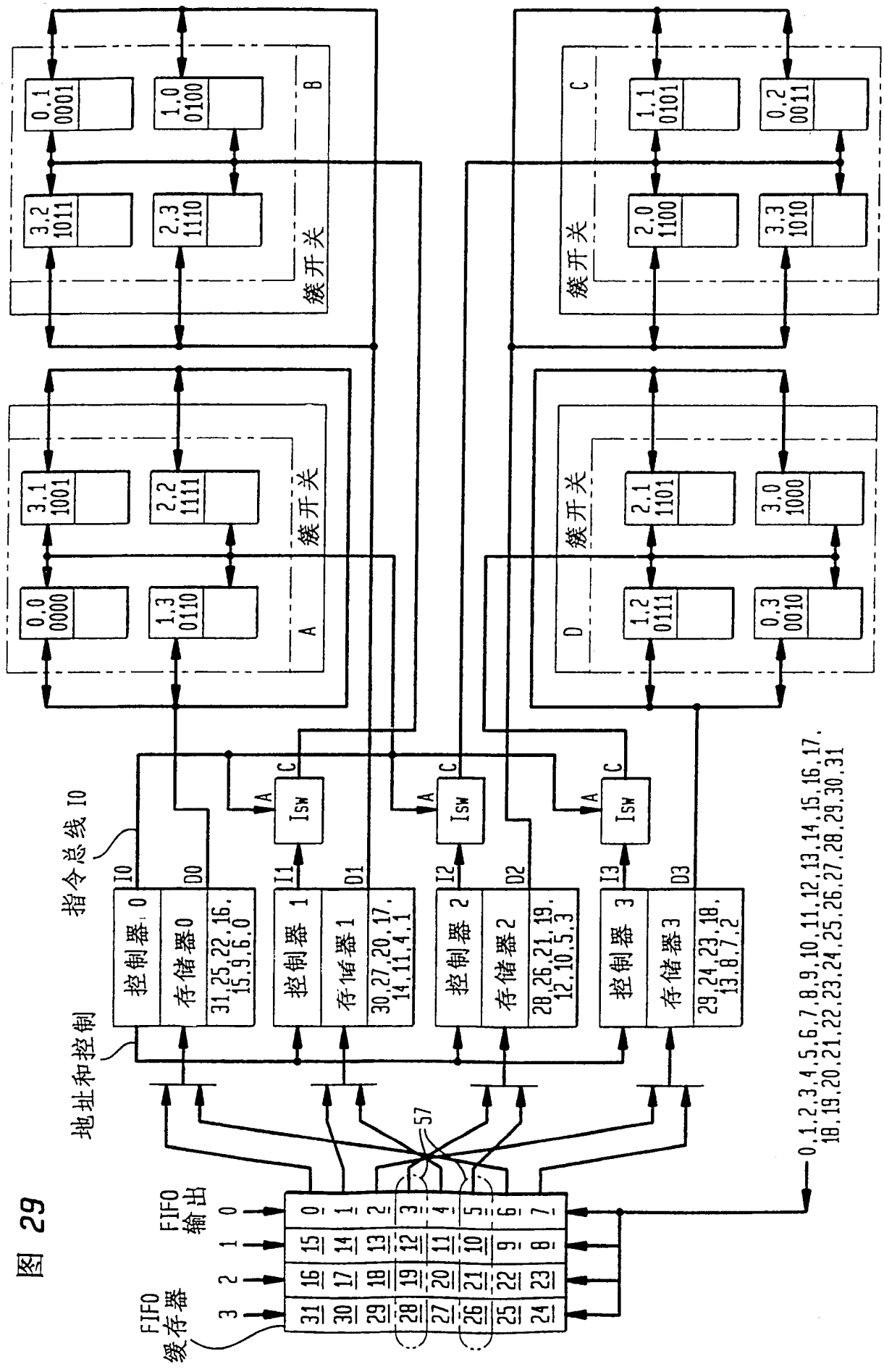


图 29
FIFO 缓存器
FIFO 输出
地址和控制
指令总线 I0
控制器 0
存储器 0
31,25,22,16,15,9,6,0
控制器 1
存储器 1
30,27,20,17,14,11,4,1
控制器 2
存储器 2
28,26,21,19,12,10,5,3
控制器 3
存储器 3
29,24,23,18,13,8,7,2
ISW
簇开关
A
B
C
D
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
57

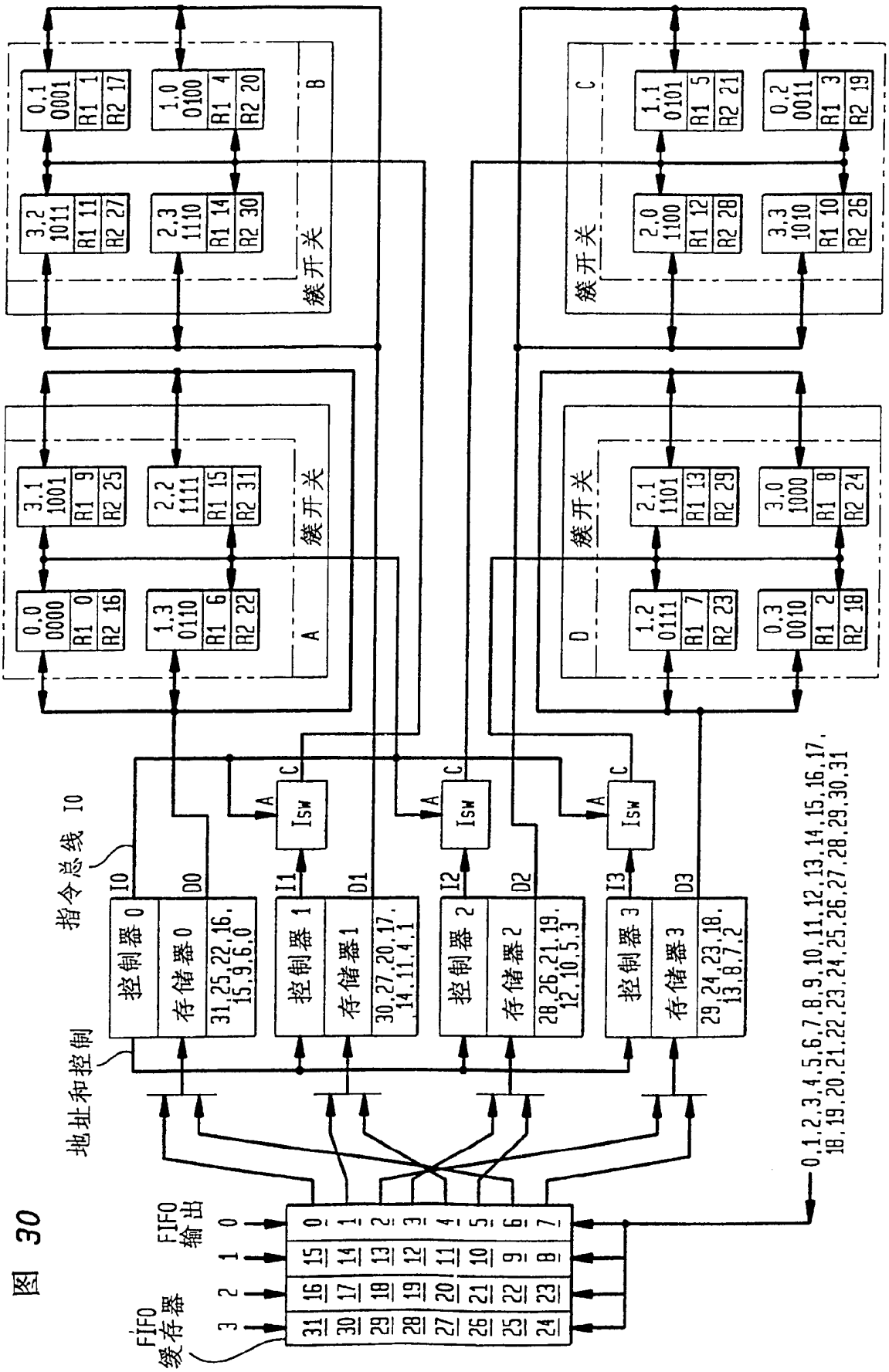
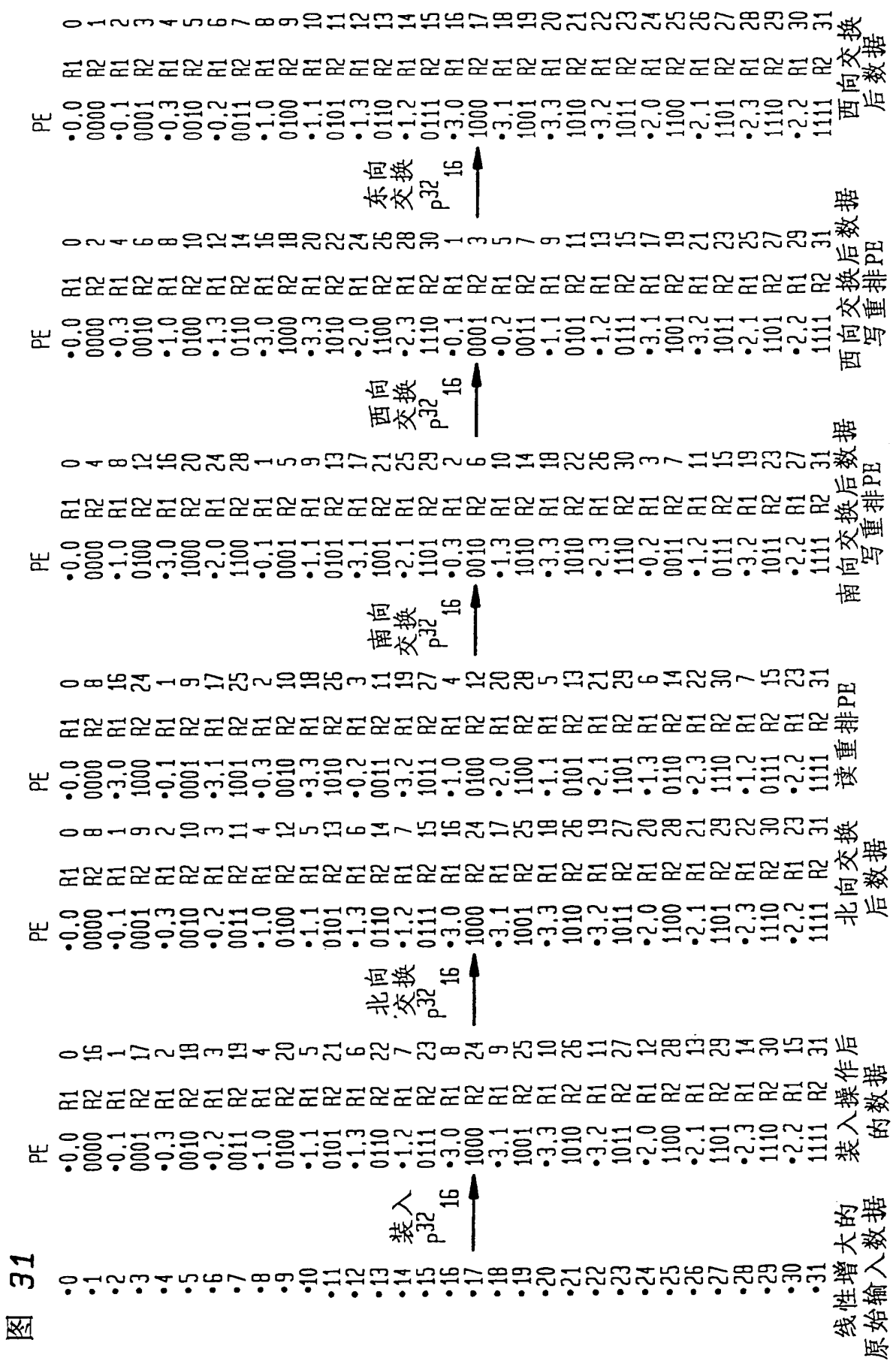


图 30
FIFO 缓存器
FIFO 输出
3 2 1 0
31 30 29 28 27 26 25 24
16 17 18 19 20 21 22 23
15 14 13 12 11 10 9 8
0 1 2 3 4 5 6 7
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23,24,25,26,27,28,29,30,31
地址和控制
指令总线 I0
I0
D0
I1
D1
I2
D2
I3
D3
控制器 0
存储器 0
31,25,22,16,
15,9,6,0
控制器 1
存储器 1
30,27,20,17,
14,11,4,1
控制器 2
存储器 2
28,26,21,19,
12,10,5,3
控制器 3
存储器 3
29,24,23,18,
13,8,7,2
Isw
A
C
簇开关
A
B
C
D
0,0 0000 R1 0 R2 16
1,3 0110 R1 6 R2 22
3,1 1001 R1 9 R2 25
2,2 1111 R1 15 R2 31
0,1 0001 R1 1 R2 17
1,0 0100 R1 4 R2 20
3,2 1011 R1 11 R2 27
2,3 1110 R1 14 R2 30
1,2 0111 R1 7 R2 23
0,3 0010 R1 2 R2 18
2,1 1101 R1 13 R2 29
3,0 1000 R1 8 R2 24
1,1 0101 R1 5 R2 21
0,2 0011 R1 3 R2 19
2,0 1100 R1 12 R2 28
3,3 1010 R1 10 R2 26

图 31


线性增大的原始输入数据
装入 P_{16}^{32}
装入操作后的数据
北向交换 P_{16}^{32}
北向交换后数据
读重排PE
南向交换 P_{16}^{32}
南向交换后数据
写重排PE
西向交换 P_{16}^{32}
西向交换后数据
写重排PE
东向交换 P_{16}^{32}
西向交换后数据

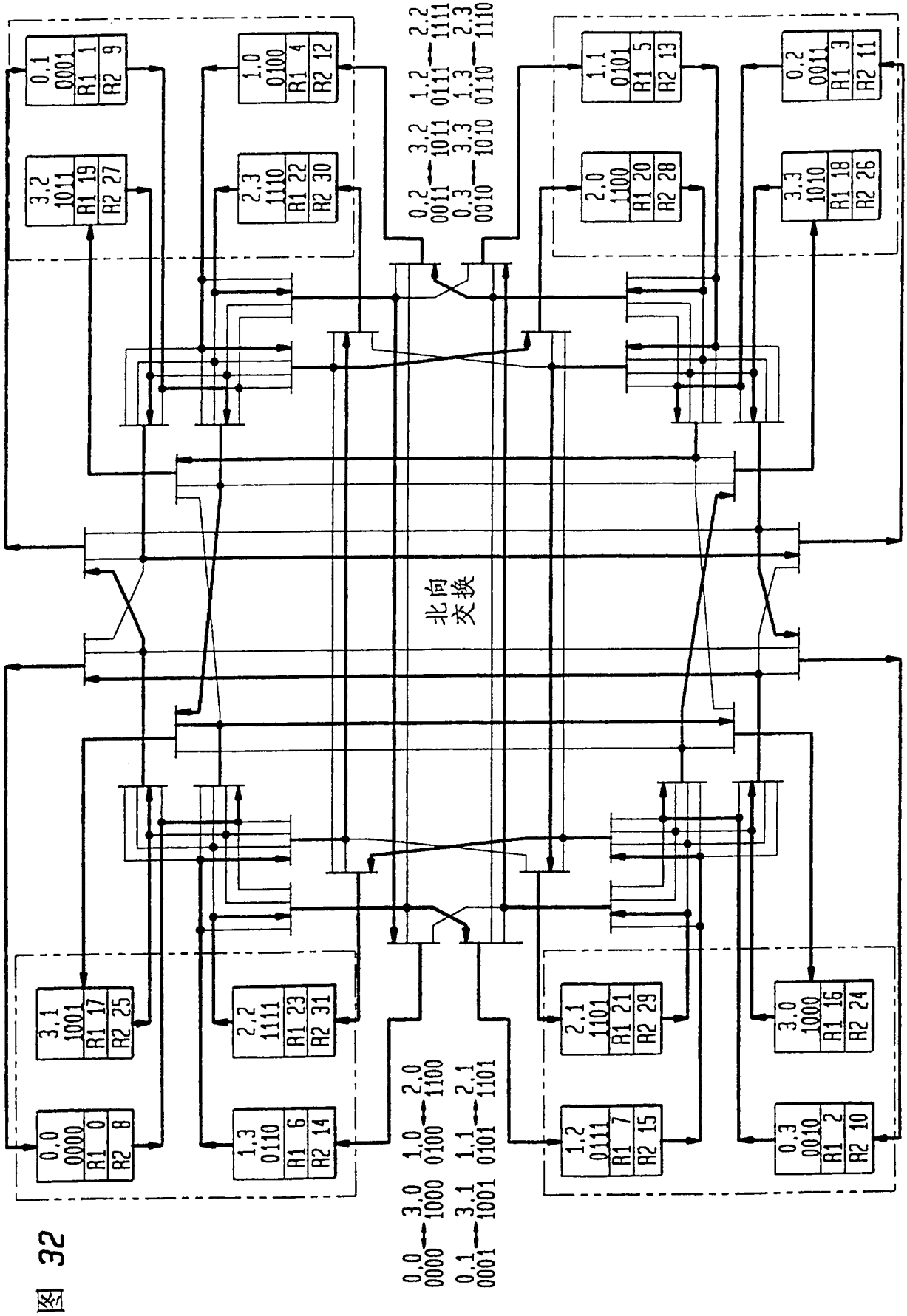

北向
交换
0,0 0000 R1 0 R2 8
0,1 0001 R1 1 R2 9
1,3 0110 R1 6 R2 14
1,0 0100 R1 4 R2 12
2,2 1111 R1 23 R2 31
2,3 1110 R1 22 R2 30
3,1 1001 R1 17 R2 25
3,2 1011 R1 19 R2 27
1,2 0111 R1 7 R2 15
1,1 0101 R1 5 R2 13
2,1 1101 R1 21 R2 29
2,0 1100 R1 20 R2 28
0,3 0010 R1 2 R2 10
0,2 0011 R1 3 R2 11
3,0 1000 R1 16 R2 24
3,3 1010 R1 18 R2 26
0,0 0000
0,1 0001
3,0 1000
3,1 1001
1,0 0100
1,1 0101
2,0 1100
2,1 1101
0,2 0011
0,3 0010
3,2 1011
3,3 1010
1,2 0111
1,3 0110
2,2 1111
2,3 1110

图 32

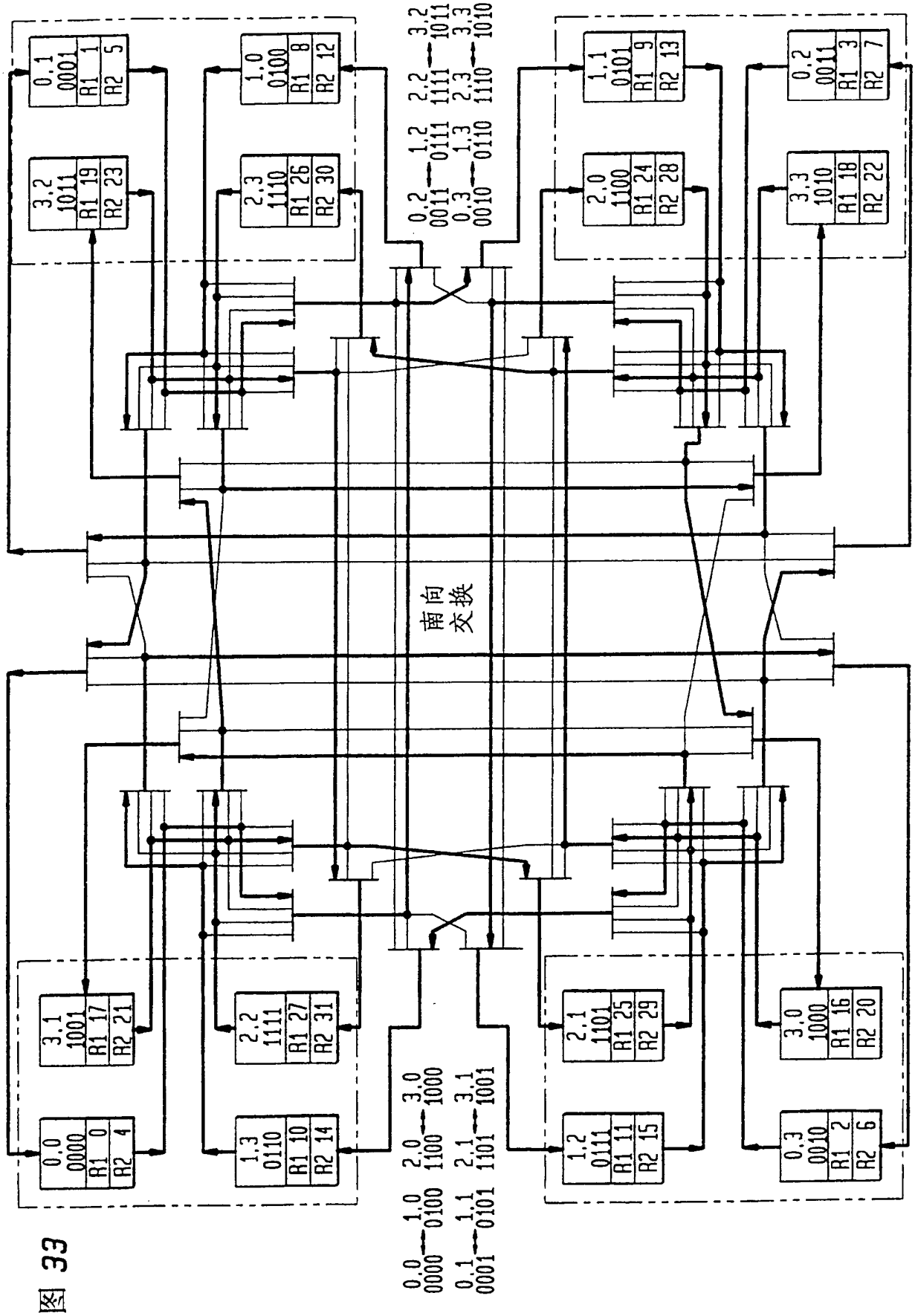

南向
交换
0,0 0000
0,1 0001
1,0 0100
1,1 0101
2,0 1100
2,1 1101
3,0 1000
3,1 1001
0,2 0011
0,3 0010
1,2 0111
1,3 0110
2,2 1111
2,3 1110
3,2 1011
3,3 1010
R1 0
R2 4
R1 1
R2 5
R1 8
R2 12
R1 9
R2 13
R1 3
R2 7
R1 10
R2 14
R1 11
R2 15
R1 2
R2 6
R1 17
R2 21
R1 25
R2 29
R1 16
R2 20
R1 27
R2 31
R1 24
R2 28
R1 18
R2 22
R1 19
R2 23
R1 26
R2 30

图 33