[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 12/56
H04L 12/24 H04Q 3/545
H04Q 3/00

[12] 发明专利申请公开说明书

[21] 申请号 03106082.X

[43] 公开日 2003 年 10 月 1 日

[11] 公开号 CN 1445966A

[22] 申请日 2003.2.21 [21] 申请号 03106082.X

[30] 优先权

[32] 2002.2.23 [33] US [31] 10/082,450

[71] 申请人 泰拉鲍尔股份有限公司

地址 香港新界沙田新城市中央广场第二座 19 字楼 1908 室

[72] 发明人 李硕彦 朱 键

[74] 专利代理机构 隆天国际知识产权代理有限公司

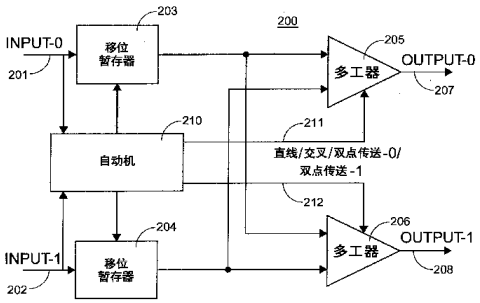
代理人 陈 红 楼仙英

权利要求书 2 页 说明书 20 页 附图 9 页

[54] 发明名称 交换网络的包路由方法与系统

[57] 摘要

本发明是新编码方式，用于使交换单元的总区域延迟时间尽可能缩短，所需的缓存单位尽可能减少。 在此新编码方式中，该交换单元可最优化，而使得无须等到所有区域路由位到齐就能开始产生输出包的第一个输出位，并因此缩短区域缓存时间延迟，及减少和缓存相关的硬件实体。 在实际交换应用中，支持多播的多阶交换网络中的每一个交换单元均为一双点传送交换单元而伴随的新编码方式甚至可在每一个双点传送交换单元中达成最小缓存的目的。 因此，整个交换系统的总延迟时间可被缩短，及总共所需的硬件实体可被减少。



ISSN1008-4274

1.一种交换网络的包路由方法，其中该交换网络包括多阶的交换单元，每一交换单元于其输入端口接收包作为其区域输入包，并于其输出端口产生包作为其区域输出包，每一包均具有至少一个带内控制信号，每一个带内控制信号是有一相对应的交换单元将该带内控制信号视为该交换单元的区域带内控制信号，及为该交换单元做交换决策之用，该方法包括：

根据一预定的编码法则将每一包的每一个带内控制信号编成至少一个位；以及

参考该编码方式，每一交换单元根据该交换单元的每个区域输入包中与该交换单元相对应的区域带内控制信号中的部份或全部位做出交换决策，并根据该交换决策，相应地于其输出端口产生其区域输出包的输出位，以完成对该交换单元的区域输入包的路由。

2.如权利要求 1 的方法，其特征在于每一交换单元是一双点传送交换单元，而输入至每一交换单元的区域输入包均包括闲置、0-向、1-向、以及双点传送等包型式，其中每一种包型式均对应至不同的带内控制信号，该编码包括以至少两个位来对每一带内控制信号进行编码，且该编码法则包括一种位编码方式，是使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

3.如权利要求 1 的方法，其特征在于每一交换单元是一单点传送交换单元，而输入至每一交换单元的区域输入包均包括闲置、0-向、以及 1-向等包型式，其中每一种包型式均对应至不同的带内控制信号，该编码包括以至少两个位来对每一带内控制信号进行编码，且该编码法则包括一种位编码方式，是使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

4.一种包路由系统，包括：

多阶的交换单元，每一交换单元于其输入端口接收包作为其区域输入包，并于其输出端口产生包作为其区域输出包，每一包均具有至少一个带内控制信号，每一个带内控制信号是有一相对应的交换单元将该带内控制信号视为该交换单元的区域带内控制信号，及为该交换单元做交换决策的用；

一编码器，用于根据一预定的编码法则将每一包的每一个带内控制信号编成至少一个位；以及

一产生器，参考该编码方式，于每一交换单元产生其区域输出包的输出位，以完成对该交换单元的区域输入包的路由，此处输出位的产生是相应于该交换单元根据其每个区域输入包中与该交换单元相对应的区域带内控制信号中的部份或全部位所做出的交换决策。

5.如权利要求 4 的系统，其特征在于每一交换单元是一双点传送交换单元，而输入至每一交换单元的区域输入包均包括闲置、0-向、1-向、以及双点传送等包型式，其中每一种包型式均对应至不同的带内控制信号，该编码器包括以至少两个位来对每一带内控制信号进行编码的装置，且该编码法则包括一种位编码方式，使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

6.如权利要求 4 的系统，其特征在于每一交换单元是一单点传送交换单元，而输入至每一交换单元的区域输入包均包括闲置、0-向、以及 1-向等包型式，其中每一种包型式均对应至不同的带内控制信号，该编码器包括以至少两个位来对每一带内控制信号进行编码的装置，且该编码法则包括一种位编码方式，使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

交换网络的包路由方法与系统

技术领域

本发明涉及一种高速交换，特别指一种在巨大宽频交换网络上的次微秒交换及控制设计。

背景技术

交换系统的关键组件为交换引擎（switching fabric），而交换引擎通常由交换核心（core fabric）和交换控制（switching control）二大部分组成。交换核心事实上又是由众多交换单元（switching element）构成的交换网络，是资料交换传送的实际信道；而交换控制则根据资料交换的请求负责完成调度仲裁（scheduling），并在交换核心中建立适当的路径来连接相关的输入/输出端口。交换控制一般都涉及复杂的逻辑，所以其控制速度的快慢往往直接影响交换系统的性能如吞吐量（throughput）、延迟时间（latency）等。随着交换系统的日益庞大复杂，交换控制的设计已成为交换架构（switching architecture）设计的关键所在。

总体来讲，交换控制可分为集中式控制（centralized control）和分布式控制（distributed control）两种。集中式控制往往因为要从一个全局的角度来同时协调或调度（scheduling）所有的资料交换请求并设置交换核心来完成其资料交换，所以其复杂度与交换系统的规模有着直接的关是。随着输入/输出端口数目的增加，集中式控制的运算速度不可避免地成为系统规模增长的一个瓶颈。因此，集中式控制只适合小规模的交换系统。相反的，分布式控制则与交换系统规模无关。

分布式交换控制出现在以交换单元多阶互连网络 (multistage interconnection network) 构成的交换引擎中，其交换任务分配给每一个独立的交换单元。交换单元的间藉由恰当的互连，使得每个交换单元只须根据抵达自己的包（packet）所包含的控制信号做出自己的交换决策（switching decision），而无须理会其它包的情形。每一个交换单元独自做出自己的交换

决策，而所形成的区域信道（local connection path）一起构成了交换核心的整体路径（global connection path），从而完成整个交换。这种控制方式遍布于整个由交换单元所构成的交换网络中，其表现形式就像每个独立的包自我引导穿越整个交换网络。每一个的交换单元内的交换决策是为区域性的，且相互独立。如此一来，不管交换系统规模有多大，理论上讲，整个交换决策都可以非常快。采用分布式控制的交换系统可享有较短延迟时间（latency）和较高扩展性（scalability）等好处，因此，与集中式控制相比的下，大规模交换系统的控制设计倾向于选择分布式交换控制。

在采用分布式控制时，每个包嵌入一定数量的控制信号位（control signal bit）作为其前置码（prefix）。所以，分布式控制也称为“带内控制（in-band control）”，其嵌入的控制信号通常称作“带内控制信号（in-band control signal）”。

由于带内控制交换引擎所需的交换单元数以千计，甚至万计，如果每个交换单元的成本能因某种最优化设计而节省很小的一部份，整个交换引擎的成本即可大幅度降低，所以，交换单元的设计最优化是非常重要的。这里，影响交换单元成本的主要包括以下两个因素：（1）交换单元内部控制逻辑所需的缓存器数目，和（2）包穿越交换单元所要经历的延迟时间长短。这些因素的变化与带内控制信号的编码方式有着十分紧密的关是，而编码方式是交换单元设计中的关键部分，不同的带内控制信号编码方式导致控制逻辑有不同的延迟时间和不同的复杂程度。所以，如何将编码方式精巧地与交换单元最优化联是在一起是至关重要的。

发明内容

本发明以现代技术为基础，将数学原理落实为大型交换引擎实体，以避免现有技术的缺点与其它限制及不足。

本发明的广义形式是关于一种交换网络的包路由方法，其中该 交换网络包括多阶的交换单元，每一交换单元于其输入端口接收包作为其区域输入包，并于其输出端口产生包作为其区域输出包，每一包均具有至少一个带内控制信号，每一个带内控制信号是有一相对应的交换单元将该带内控制信号视为该交换单元的区域带内控制信号，及为该交换单元做交换决策的用，而

该方法包括：(a)根据一预定的编码法则将每一包内的每一个带内控制信号编成至少一个位；以及(b)参考该编码方式，每一交换单元根据该交换单元的每个区域输入包中与该交换单元相对应的区域带内控制信号中的部份或全部位做出交换决策，并根据该交换决策，相应地于其输出端口产生其区域输出包的输出位，以完成对该交换单元的区域输入包的路由。

在前述方法的一较佳实施例中，(i) 每一交换单元是一双点传送交换单元(bicast cell)，(ii)输入至每一交换单元的区域输入包均包括闲置(idle)、0-向(0-bound)、1-向(1-bound)、以及双点传送(bicast)等包型式，其中每一种包型式均对应至不同的带内控制信号，(iii)该编码包括以至少两个位来对每一带内控制信号进行编码，以及(iv)该编码法则包括一种位编码方式，是使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

在另一较佳实施例中，(i) 每一交换单元是一单点传送交换单元(unicast cell)，(ii)输入至每一交换单元的区域输入包均包括闲置(idle)、0-向(0-bound)、以及 1-向(1-bound)等包型式，其中每一种包型式均对应至不同的带内控制信号，(iii)该编码包括以至少两个位来对每一带内控制信号进行编码，以及(iv)该编码法则包括一种位编码方式，是使得对应于 0-向包型式的带内控制信号码的第一位有别于对应于 1-向包型式的带内控制信号码的第一位。

本发明的系统形式与方法形式相对应。

附图说明

本案通过下列图式及详细说明，使之能得到更深入的了解：

图 1(A)描述一交换单元的通貌；

图 1(B)至图 1(E)描述双点传送(bicast)交换单元的“直线(bar)”，“交叉(cross)”，“双点传送-0 (bicast-0)”和“双点传送-1 (bicast-1)”四种连接状态；

图 2 为进行带内控制的一交换单元的方块图；

图 3(A)描述当双点传送交换单元在“input-0”和“input-1”两个输入端口上的输入包分别为“双点传送包”和“闲置包”时的连接状态；

图 3(B)描述当双点传送交换单元在“input-0”和“input-1”两个输入端口上的输入包分别为“闲置包”和“双点传送包”时的连接状态；

入端口上的输入包分别为“闲置包”和“双点传送包”时的连接状态；

图 4 到图 7 描述双点传送交换单元所有可能的输入包情形。

为了方便理解，图中所有相同组件或模块的参考编号是一样的。

具体实施方式

为了完整了解本发明交换电路的重要性以及本发明运作原理的优越性，首先扼要讨论与本发明密切相关的一些基本原理，同时介绍相关术语，以便更详细地说明本发明的实施例。

1. 概观

交换过程中，收到的包其相邻两个位的时间间隔称为“位时间（bit time）”，用 Δt 表示。从包第一个输入位进入交换单元开始，到其输出第一个输出位为止，所经历的位时间就是交换单元的延迟时间。当我们说交换单元的延迟时间为 nΔt 时，即代表包第一个输入位进入交换单元开始，至其输出第一个输出位为止，共经历了 n 个位时间 Δt。

每个包都携带有若干个包含着对应于该包的路由信息（routing information）的路由位（routing bit），对某一个交换单元来说，如果它需要当中 R 个特定的路由位来做出其交换决策，那么这 R 个路由位对该交换单元而言就称为“区域路由字节（local routing bits）”，有时也称为“区域路由控制信号（local routing control signal）”。交换单元依据这 R 个位的区域路由控制信号做出交换决策。明显地，区域路由控制信号应该置于包的最前面，否则交换单元必须缓存(buffer)更多的资料才能产生其第一个输出。

现有的交换单元设计方法通常是在它产生第一个输出位前先将所有R个区域路由位缓存起来。R 个位的缓存会导致交换单元的“区域缓存时间延迟（local buffering delay）”。交换单元内需要缓存的位越多，那么，控制逻辑所需的缓存器也越多，其复杂程度也更高。依据本发明，提出一种整合方式，可于带内控制信号编码方式和交换单元设计的应用上，使得交换单元无须等到所有 R 个位到齐就能产生连续输出位。换句话说，在交换单元确定输入端口和输出端口的间的最终路由连接的前，交换单元仅仅收到 R 个路由位中的前面几个位时，就已经可以确定它的第一个输出位，且随着后续到达的区域路由位，交换单元可以不断地确定相应的后续输出位。

假设交换单元需要知道最近 R_j 个位的路由控制信号才能产生其第 j 个输出位，那么包至少需要经历 B 个位的缓冲，这里 $B=\max_j\{R_j-j\}$。如果交换引擎总共有 K 阶（stage）交换单元，那么包需经历总共 KB 位的延迟时间。另一方面，由于交换单元需要对每个输入包进行 $B=\max_j\{R_j-j\}$位的缓存，其硬件实现时所需的缓冲缓存器数量与 B 成比例，所以，交换单元的最优化事实上就是要使 $B=\max_j\{R_j-j\}$的数值尽可能地小，即 min{B}。

如果一个交换单元有 m 个输入口和 n 个输出口，其规模可以 m×n 表示。理论上，带内控制的交换单元的规模可以是任意的。但是在实际应用中通常倾向于小的交换单元。否则，由于要应对较长的区域路由控制信号和要用较复杂的硬件逻辑，交换决策会很难即刻做出，因此会变成交换引擎的瓶颈，显然这样就失去了采用分布式控制的意义。在大多数情况下，交换单元采用最小规模，即最典型的 2×2 交换单元。在以下的叙述中，除非特别说明，文中所述的交换单元统指 2×2 交换单元。本发明引入一种崭新的带内控制信号的智能编码方式，说明相对最优化的并可实际应用的 2×2 交换单元。

如图 1(A)所示，交换单元（100）有“input-0”（110）和“input-1”（111）两个包输入端口以及“output-0”（120）和“output-1”（121）两个包输出端口。在大多数文献中提到的交换单元是点对点（point-to-point）的，这种交换单元在本发明中称的为“单点传送(unicast)交换单元”或简称的为“单点传送单元”。 如图 1(B)和图 2(C)所示，单点传送单元只有“直线(bar)”和“交叉(cross)”两种连接状态。当处于“bar”状态（151）时，input-0 和 input-1 分别连接到 output-0 和 output-1；而当处于“cross”状态（152）时，input-0 和 input-1 则分别连接到 output-1 和 output-0。本发明提出了另一种有 4 种连接状态的交换单元用于支持具有多播（multicast）功能的交换引擎。对照于单点传送，这种交换单元在本文中称的为“双点传送交换单元”或简称的为“双点传送单元”。 双点传送单元除了有单点传送单元具有的“cross”和“bar”连接状态外，它另外还有如图 1(D)和图 2(E)所示的“bicast-0”和“bicast-1”两种连接状态。当处于 bicast-0 状态（161）时，input-0 同时被连接到两个输出；而当处于 bicast-1 状态（162）时，输入 input-1 则同时连接到两个输出。这也就是为什么称其为双点传送单元的原因。

根据本发明，包的带内控制信号被精巧地编码，使得对于所有的 j，其

$B=\max_j\{R_j-j\}$可降至 0，从而达到最优化交换单元的目的。换句话说，交换单元每收到分别来自两个输入包的各一个位，即可确定其两个输出端口的各自一个输出位。所以，本发明所描述的最优化的交换单元在实际应用中都可获得最小的延迟时间（delay）和最小的缓存（buffering）单位。

图 2 为用于简要说明带内控制交换单元的原理的方块图（200）。两个包比特流（bit stream）分别从输入端口（201，202）进入到两个移位缓存器（shift register）（203，204）。两个包的区域路由控制信号分别被取出一起给状态自动机（automata）（210）去确定交换单元每个位周期的连接状态。连接状态的实现完全依靠两个 2×1 的多任务器（multiplexer）（205，206），每个多任务器的两个输入都分别连接到两个移位缓存器的输出；而它们的输出则分别直接连接至交换单元的两个输出（207，208）。另外，自动机的输出或者控制两个多任务器，以选择适当的移位缓存器输出作为交换单元的输出，或者强制输出适当的位以满足多播特征，细节留待下文再述。简单来说，要实现“bar 状态”，上面的多任务器 205 和下面的多任务器 206 将被分别控制，使其分别选择上面的移位缓存器输出和下面的移位缓存器输出。对应地，要实现“cross 状态”，自动机将分别控制上下两个多任务器选择下面的移位缓存器输出和上面的移位缓存器输出。要实现“bicast-0”状态，自动机将控制两个多任务器都选择上面的移位缓存器输出，同样地，要实现“bicast-1”状态，自动机则将控制两个多任务器都选择下面的移位缓存器输出。须要注意的是，双点传送的情况会略复杂，下文会有详述。

当交换单元的某个输入端口没有包输入时，该端口则称为“闲置端口（idle port）”。在实际应用中，闲置端口会收到一个表示空闲的“闲置包（idle packet）”，这种闲置包通常用“0”比特流表示。因此，输入包不是真正的包，就是闲置包。对于单点传送交换单元来讲，真正的包可能欲交换到 output-0 输出端口或 output-1 输出端口，相应的包分别称为“0-向(0-bound)包”和“1-向(1-bound)包”，因此，对于任一单点传送交换单元来说，来自输入包的区域路由控制信号要表示出 0-bound、1-bound 和 idle（闲置）三种类型。同样，对于双点传送交换单元来讲，真正的包除了 0-bound 包和 1-bound 包外，另外还有一种称为“双点传送(bicast)包”，即欲同时交换到 output-0 输出端口和 output-1 输出端口，因此，对于任一双点传送交换单元来说，来自输入包的

区域路由控制信号要表示出 0-bound、1-bound、bicast 和 idle 四种类型。

当两个输入包欲交换到同一个输出口时，输出冲突（output contention）便会出现。解决输出冲突有多种不同的方法，在下文的例子中会提到其中一种方法。一个理想的单点传送单元是在任何时候只要没有输出冲突，它总是把 0-bound 包交换到 output-0 输出端口以及把 1-bound 包交换到输出端口 output-1。因此，单点传送单元收到的两个输入包的所有可能组合以及相应的连接状态列表如下：

表 1

单点传送单元的连接状态		input-1 的包		
		“idle”	“0-bound”	“1-bound”
input-0 的包	“idle”	任意	Cross	bar
	“0-bound”	bar	取决于怎样解决 output-0 的输出冲突	bar
	“1-bound”	cross	cross	取决于怎样解决 output-0 的输出冲突

双点传送交换单元的情形要比单点传送交换单元复杂得多。当两个输入包中没有双点传送包时，双点传送交换单元的所有行为完全同单点传送交换单元一样。然而，当其中有一个双点传送包时，其情形将是下列几个情形的一：

(a)另外一个输入包是“闲置包”：

双点传送包依据其所在的输入端口为 input-0 或 input-1 成功地通过相对应的 bicast-0 或 bicast-1 连接状态同时被复制（bicasted）到 output-0 输出端口和 output-1 输出端口。在此要特别注明的是，这里所说的“复制”还包含了以下两个特别处理过程：（1）将输送到 ouput-0 输出端口的包中的控制信号由“bicast”修改成“0-bound”和（2）将输送到 ouput-1 输出端口的包中的控制信号由“bicast”修改成“1-bound”。不然的话，这两个被复制的双点传送包有可能会被较后阶的双点传送交换单元再次被复制。所以，除非另外说明，否则本文所提到的和双点传送交换单元相关的“复制（bicasted）”

都是指如上所述的过程。

(b)另外一个包是单点传送包（0-bound 包或 1-bound 包）:合理的方法是将单点传送包交换到它欲去的输出口，而将双点传送包交换到另外一个端口。

(c) 另外一个输入包也是“bicast 包”：将它们分别交换到两个输出显然比复制其中一个到两个输出，另一个被删除掉要公平得多。因此，这种情况的连接状态可以是“bar”或“cross”,而不应是“bicast-1”或“bicast-1”。

总之，双点传送交换单元只有当输入包中一个是双点传送包而另一个是闲置包时才会激活复制功能。否则，双点传送包将只被交换到其中一个适当的输出端口。

双点传送单元收到的包的所有组合以及相应的连接状态如表 2 所示。

表 2

双点传送单元的连接状态		input-1 的包			
		“idle”	“0-bound”	“1-bound”	“bicast”
input-0 的包	“idle”	任意	cross	bar	bicast-1
	“0-bound”	bar	取决于怎样解决 output-0 的输出冲突	bar	bar
	“1-bound”	cross	cross	取决于怎样解决 output-1 的输出冲突	cross
	“bicast”	bicast-0	cross	bar	bar 或 cross

在本文中或相应的图中有时会用符号 “0”、“1”、“I”和“B”来分别表示“0-bound”、“1-bound”、“idle”和 “bicast” 包或对应的 0-bound、1-bound、idle 和 bicast 的路由控制信号。

图 3(A)表示出双点传送交换单元（300）两个输入端口 input-0（310）和

input-1（311）的输入包分别为双点传送（330）包和闲置包（331）时的情形。这种情况下，其连接状态被设置为 bicast-0(360)，双点传送包通过 bicast-0 连接状态被复制到两个输出端口，其中 output-0 输出端口（320）的输出包的 bicast 路由控制信号被设定为 0-bound（332），而 outpt-1 输出端口（321）的输出包的双点传送路由控制信号被设定为 1-bound（333）。稍微具体一点来说，当两个输入包的比特流顺序进入该双点传送单元时，前置码代表区域路由控制信号的路由字节首先到达，当两个路由字节分别被认定为代表双点传送和闲置时，双点传送单元就立即可以采取相应的行动：锁定连接状态为 bicast-0，而双点传送包的那个双点传送路由字节在被送往 output-0 和 output-1 两个输出端口时，分别被强制设定为代表 0-bound 和 1-bound 控制信号的路由字节（此举等同于该双点传送路由字节分别被取代为代表 0-bound 控制信号的路由字节而在 output-0 输出端口输出和代表 1-bound 控制信号的路由字节而在 output-1 输出端口输出）最后该双点传送包的其余资料位将直接穿过由被锁定的 bicast-0 连接状态提供的数据信道被同时分送到两个输出端口，从而正确地完成预期的交换。须要注意的是，此处对整个机制的描述为了便于理解而略有简化，实际的运作情况在下文中会有更详细的描述。

同样地，图 3(B)表示出双点传送交换单元两个输入端口 input-0 和 input-1 的输入包分别为闲置包（370）和双点传送包（371）的情形。这种情况下，其连接状态被设置为 bicast-1（361），双点传送包通过 bicast-1 连接状态被复制到两个输出端口，其中 output-0 输出端口的输出包的双点传送路由控制信号被设定为 0-bound（372），而 outpt-1 输出端口的输出包的双点传送路由控制信号被设定为 1-bound（373）。

本发明所描述的最优化交换单元涉及到如何用简洁的交换控制方法完成自身的局部交换。事实上这种双点传送交换单元可用有限状态自动机（finite-state automata）来描述。这里共有五种状态，其中四种直接对应交换单元的 bar、cross、bicast-0 和 bicast-1 四种连接状态，而另外一种是与复制有关的“置 0/1（set 0/1）”状态。针对每一次的包输入，交换决策过程事实上就是初始连接状态到最终连接状态的转移过程，过程是从包抵达开始至获得最终的交换状态止。当获得最终的交换状态时，自动机被锁定（latched）直至新的包抵达。这种自动机事实上已包括单点传送交换单元功能，因为单

点传送交换单元只是双点传送交换单元的一种简化版本。有限状态自动机状态迁移的驱动事件主要是两个输入包的路由控制信号的位序列。每一次新的交换决策都针对两个输入包的路由控制信号，而这里每个包的路由控制信号有两个位，这样共有两个输入序列，每个序列有四种组合，合计共有十六种序列组合，即共有十六个可能的状态迁移驱动事件。具体情况要取决于采用何种编码，下文的实例中会有详述。表 3 则枚举出两个输入包的所有组合以及分别和每个组合相对应的正确结果，即每个相对应的自动机的最终状态和两个输出端口的输出包。所有交换决策过程最后都会相应地锁定在 bar、cross、bicast-0 和 bicast-1 四种连接状态之一，所以后续的资料位将直接贯穿由锁定的连接状态构成的数据信道抵达它们各自的目的端口。当两个输入都是“0-bound”包或都是“1-bound”包时便会出现输出冲突，其结果是可用不同的方法来处理，例如将其中一个包故意地交换到一个非其想去的输出端口或直接将其删除掉。对输出冲突的处理方法和本发明并没有直接关是，只是当考虑到为了尽可能减少交换的错误损失，我们会倾向于用前者，因为那个在此暂时走错路的包也许可以借着较后阶的交换单元最终还是有机会到达其最后的目的地。当两个输入都是“闲置包”时，自动机状态可以任意改变或保持其连接状态。当两个输入都是 bicast 包时，自动机状态可任意设置并锁定为 cross 或 bar 连接状态。

表 3

输入包		最终自动机状态	输出包	
input-0	input-1		output-0	output-1
“idle”	“idle”	任意	“idle”	“idle”
“idle”	“0-bound”	cross	“0-bound”	“idle”
“idle”	“1-bound”	bar	“idle”	“1-bound”
“idle”	“bicast”	bicast-1	“0-bound”	“1-bound”
“0-bound”	“idle”	bar	“0-bound”	“idle”
“0-bound”	“0-bound”	取决于怎样解决输出冲突		
“0-bound”	“1-bound”	bar	“0-bound”	“1-bound”
“0-bound”	“bicast”	bar	“0-bound”	“bicast”

“1-bound”	“idle”	cross	“idle”	“1-bound”
“1-bound”	“0-bound”	cross	“0-bound”	“1-bound”
“1-bound”	“1-bound”	取决于怎样解决输出冲突		
“1-bound”	“bicast”	cross	“bicast”	“1-bound”
“bicast”	“idle”	bicast-0	“0-bound”	“1-bound”
“bicast”	“0-bound”	cross	“0-bound”	“bicast”
“bicast”	“1-bound”	bar	“bicast”	“1-bound”
“bicast”	“bicast”	bar 或 cross	“bicast”	“bicast”

2. 新发明的编码方式

2.1 一个随意编码方式的例子

通常用 2 个位就足以双点传送交换单元的四种可能的带内控制信号或单点传送交换单元的三种可能带内信号进行编码。表 4 表示一种随意的编码方式。

表 4

	带内控制信号	位码
双点传送单元	‘Idle’	“00”
	‘0-bound’	“10”
	‘1-bound’	“11”
	‘Bicast’	“01”
单点传送单元	‘Idle’	“00”
	‘0-bound’	“10”
	‘1-bound’	“11”

这种编码方式的不足的处是交换单元，无论是单点传送交换单元还是单点传送交换单元，都必须将所有两个位读入后才能确定两个输出包的第一个输出位。例如，一个输入包的第一位在交换单元的输入端为“0”，而另一个输入包的第一个位在交换单元的另一输入端为“1”，由于以“1”开头的包既可能是“0-bound”包 又可能是“1-bound”包。 对于“0-bound”的情况，根据此编码方式，在 output-0 输出端口的第一个输出位应为“1”而在 output-1 输

出端口的第一个输出位应为“0”。相对应地，对于 “0-bound”的情况，根据此编码方式在 output-0 输出端口的第一个输出位应为“0”，而在 output-1 输出端口的第一个输出位应为“1”。也就是说要正确确定交换状态，交换单元必须等到每个包的第二个路由位到达后才能确定两个输出端口的第一个输出位。所以，$R_1 = 2$。因此，此编码方式的区域缓冲延迟时间 $B=max_j\{R_j-j\} \geq R_1-1 = 1$。

2.2 依据本发明的新编码方式

依据本发明设计的新编码方式去除了类似上例中的不足，使得区域缓冲延迟时间为零。具体地讲，一旦收到输入包的第一个位，交换单元便可立即确定第一个输出位，依此类推。

依据本发明最广义层面上讲，在新的编码方式中，不管是相对于双点传送交换单元的 bar、cross、bicast 和 idle 等四种可能带内控制信号或相对于单点传送交换单元的 bar、cross 和 idle 等三种可能带内控制信号，对“O-bound”和“1-bound”的编码，其第一个位必须是不同的。

实施例

对应于双点传送交换单元可考虑如下新的编码方式：

表 5

带内控制信号	位码
‘Idle’	“00”
‘0-bound’	“01”
‘1-bound’	“11”
‘bicast’	“10”

为了更深入了解本发明，本发明的详细描述将采用实例（如表 5 所示的新编码方法）的方式来说明。然而，本发明所涉及的方法并不局限于所示的例证。归纳起来，采用本发明的各种编码方式中，其表示“0-bound”和“1-bound”包的相应路由位的第一个位必须是不同的。

图 4 至图 7 表示出所有双点传送交换单元可能遇到的各种情形。在这些图中，所有的双点传送交换单元以左边两个输入端口（上面是 input-0 输入端口,下面是 input-1 输入端口）和右边两个输出端口（上面是 output-0 输出端口，下面是 output-1 输出端口）的方式描述。各种情形的描述采用了序列符

号 I_t 和 O_t，它们表达的意义如下：

序列表达式 I_t=AB 表示交换单元的 input-0 输入端口和 input-1 输入端口的第 t 个序列输入位分别为 A 和 B。

序列表达式 O_t=AB 表示交换单元的 output-0 输出端口和 output-1 输出端口的第 t 个序列输出位分别为 A 和 B。

例如：序列表达式 I_1=01 表示交换单元的第一个输入序列在 input-0 输入端口的输入位为 0，而在 input-1 输入端口的输入位为 1。

最优化的双点传送交换单元获得 $B=max_j\{R_j-j\}$ 为 0 的方法描述如下：

情形 1：I_1=01

第四图说明情形 I_1=01。第一个输入序列 I_1=01 意味着在 input-0 输入端口的包第一个位（4011）为“0”，而在 input-1 输入端口的包第一个位（4012）为“1”。那么连接状态暂时设置为“bar”连接状态（4001）。这样，在 output-0 输出端口的第一个输出位（4021）为“0”,同样地，在 output-1 输出端口的第一个输出位（4022）为“1”，即 O_1=01。当输入第一个输入序列 I_1=01 后，第二个输入序列 I_2 的可能组合有四种。接下来的分析将表明不管 I_2 是这四种组合中的哪一种，O_1=01 都是正确的。换句话说 R_1=1。

情形 1.1：I_2=00

第二个输入序列 I_2=00(4100)的情形，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是闲置包和双点传送包。这种情况下，其中的双点传送包可复制到两个输出端口，即将其中 output-0 输出端口的第二个输出位（4121）将强制“置 1”并设置和锁定（图中用字母“L”表示）连接状态为 bicast-1（4101）。这样的话，在 output-1 输出端口的第二个输出位（4122）为 1，即 O_2=11。结果 output-0 输出端口的输出包的带内控制信号为“01”（0-bound)而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound)。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2=2。

情形 1.2：I_2=01

第二个输入序列 I_2=01 的情形(4200)，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是闲置包和 1-bound 包。这种情况下，其中的 1-bound 包可以交换到它的目的输出端口 output-1 输出端口。所以，连接状

态被设置成 bar（4201）并锁定。这样的话，在 output-0 输出端口的第二个输出位（4221）为“0”，而在 output-1 输出端口的第二个输出位（4222）为 1，即 O_2=01。结果，output-0 输出端口的输出包的带内控制信号为“00”（闲置）而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2=2。

情形 1.3：I_2=11

第二个输入序列 I_2=11 的情形（4300）类似于情形 I_2=01。即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 0-bound 据包和 1-bound 包。这种情况下，两个输入包都可以交换到它的目的输出端口。所以，连接状态被设置成 bar（4301）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 1，即 O_2=11。结果，output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2=2。

情形 1.4：I_2=10

第二个输入序列 I_2=10 的情形（4400）类似于情形 I_2=01。即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 0-bound 包和 bicast 包。这种情况下，0-bound 包交换到它的目的端口 output-0 输出端口，而 bicast 包则交换到另外一个输出端口 output-1 输出端口。所以，连接状态被设置成 bar（4401）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 0，即 O_2=10。结果，output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而 outpt-1 输出端口的输出包的带内控制信号为“10”（bicast）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2=2。

情形 2：I_1=10

第五图说明第一个输入序列 I_1=10 的各种情形。与情形 1 相对称，这种情形下的 O_1 同样为“01”。当输入第一个输入序列 I_1=10 后，第二个输入

序列 I_2 的可能组合有四种。接下来的分析将表明不管 I_2 是这四种组合中的哪一种，O_1＝01 都是正确的。换句话说 R_1＝1。

情形 2.1：I_2＝00

第二个输入序列 I_2＝00(5100)的情形，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是双点传送包和闲置包。这种情况下，其中的双点传送包可复制到两个输出端口，即将其中 output-0 输出端口的第二个输出位将强制“置 1”并设置和锁定连接状态为 bicast-0（5101）。这样的话，在 output-1 输出端口的第二个输出位为 1，即 O_2＝11。结果 output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2＝2。

情形 2.2：I_2=01

第二个输入序列 I_2＝01(5200)的情形，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 bicast 包和 0-bound 包。这种情况下，0-bound 包交换到它的目的端口 output-0 输出端口，而 bicast 包则交换到另外一个输出端口 output-1 输出端口。所以，连接状态被设置成 cross（5201）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 0，即 O_2＝10。结果，output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而输出端口 outpt-1 的输出包的带内控制信号为“10”（bicast）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，R_2＝2。

情形 2.3：I_2＝11

第二个输入序列 I_2＝11 的情形(5300)，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 1-bound 据包和 0-bound 包。这种情况下，两个输入包都可以交换到各自的目的输出端口。所以，连接状态被设置成 cross（5301）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 1，即 O_2＝11。结果，输出端口 output-0 的输出包的带内控制信号为“01”（0-bound）而输出端口 outpt-1 的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯

穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

情形 2.4：$I_2=10$

第二个输入序列 $I_2=01$ 的情形(5400)，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 1-bound 包和闲置包。这种情况下，其中的 1-bound 包可以交换到它的目的输出端口 output-1 输出端口。所以，连接状态被设置成 cross（5401）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“0”，而在 output-1 输出端口的第二个输出位为 1，即 $O_2=01$。结果，output-0 输出端口的输出包的带内控制信号为“00”（闲置）而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

情形 3：$I_1=00$

第六图说明第一个输入序列 $I_1=00$ 的情形，这种情形下任何一种连接状态都可以达到 $O_1=$“00”。当输入第一个输入序列 $I_1=00$ 后，第二个输入序列 I_2 的可能组合有四种。接下来的分析将表明不管 I_2 是这四种组合中的哪一种，$O_1=00$ 都是正确的。换句话说 $R_1=1$。

情形 3.1：$I_2=00$

第二个输入序列 $I_2=00$(6100)的情形，即在 input-0 输入端口和在 input-1 输入端口上的输入包都是闲置包。这种情况下，连接状态可以设置成是任何一个状态并且与是否锁定无关，这样的话，在 output-0 输出端口和 output-0 输出端口的第二个输出位都是为 0，即 $O_2=00$。结果，输出包都是闲置包。另一方面，从工程实现方面考虑，设定并锁定一个确定的连接状态如“bar”（6101）是合理的，如第六图所示。这种情况下，$R_2=2$。

情形 3.2：$I_2=11$

第二个输入序列 $I_2=11$（6200）的情形意味着在 input-0 输入端口和在 input-1 输入端口上的输入包都是 0-bound 包。这种情况下，连接状态可以设定成 bar 或 cross 并且与是否锁定无关。显然，其中一个包会错误地交换到某一输出端口。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位也为 1，即 $O_2=11$。结果，output-0 输

出端口的输出包的带内控制信号为“01”（0-bound）而输出端口 outpt-1 的输出包的带内控制信号也为“01”（0-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

另一方面，从实际应用方面考虑，暂时不锁定状态是合理的。例如，如果包又分成一定数量的优先级（priority）分类，并且用相应的位资料大小表示优先级的大小，这些优先级位资料紧跟在带内控制信号后面。在其后的处理中，即使比较输入的优先级位，当发现两个输入包的优先级位位不同时，立即以优先级较高的包为目标设定连接状态并锁定。在所的这种情形中，由于两个输入包都是 0-bound 包，有较高优先级的包将有特权交换到输出端口 output-0。所以当 $I_3=10$（6500），意味着在输入端口 input-0 上的包比在输入端口 input-0 上的包有较高优先级，因此连接状态设置成“bar”并锁定（6501）；当 $I_3=01$（6600），情形恰恰相反，连接状态设置成“cross”并锁定（6601）。一旦两个输入包的优先级完全一样，连接状态设定成 bar 或 cross 已无任何区别。在一些工程实现时，所有余下的资料位仍可被用作区别位。需要注意的是在这种情况下，$B=max_j\{R_j-j\}$仍然为 0。

情形 3.3：$I_2=10$

第二个输入序列 $I_2=10$ 的情形（6300），即在 input-0 输入端口和 input-1 输入端口上的输入包分别是 0-bound 包和闲置包。这种情况下，0-bound 输入包可以交换到它的目的输出端口 output-0。所以，连接状态被设置成 bar（6301）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 0，即 $O_2=10$。结果，output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而 outpt-1 输出端口的输出包的带内控制信号为“00”（idle）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

情形 3.4：$I_2=01$

第二个输入序列 $I_2=01$ 的情形（6400），即在 input-0 输入端口和 input-1 输入端口上的输入包分别是闲置包和 0-bound 包。这种情况下，0-bound 输入包可以交换到它的目的输出端口 output-0。所以，连接状态被设置成 cross

（6401）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“1”，而在 output-1 输出端口的第二个输出位为 0，即 $O_2=10$。结果，output-0 输出端口的输出包的带内控制信号为“01”（0-bound）而 outpt-1 输出端口的输出包的带内控制信号为“00”（闲置）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

情形 4：$I_1=11$

第七图说明第一个输入序列 $I_1=11$ 的情形。这种情形下任何一种连接状态都可以达到 $O_1=$“11”。当输入第一个输入序列 $I_1=11$ 后，第二个输入序列 I_2 的可能组合有四种。接下来的分析将表明不管 I_2 是这四种组合中的哪一种，$O_1=11$ 都是正确的。换句话说 $R_1=1$。

情形 4.1：$I_2=00$

第二个输入序列 $I_2=00$ 的情形（7100）意味着两个输入包都是 bicast 包，这种情况下连接状态可以设定成 bar 或 cross 并且与是否锁定无关，其输出都是 $O_2=00$。无论它们的优先级是否相同，其连接状态为“bar”或“cross”都一样好。这种情况下，$R_2=2$。

情形 4.2：$I_2=11$

第二个输入序列 $I_2=11$ 的情形（7200）意味着两个输入包都是 1-bound 包，这种情况下连接状态可以设定成 bar 或 cross，其输出都是 $O_2=11$。随后的处理可以采用类似情形情形 3.2 所述的优先级处理方式处理。这种情况下，$R_2=2$。

情形 4.3：$I_2=10$

第二个输入序列 $I_2=10$ 的情形（7300），即在 input-0 输入端口和 input-1 输入端口上的输入包分别是 1-bound 包和 bicast 包。这种情况下，1-bound 包交换到它的目的端口 output-1 输出端口，而 bicast 包则交换到另外一个输出端口 output-0 输出端口。所以，连接状态被设置成 cross（7301）并锁定。这样的话，在 output-0 输出端口的第二个输出位为“0”，而在 output-1 输出端口的第二个输出位为 1，即 $O_2=01$。结果， output-0 输出端口的输出包的带内控制信号为“10”（bicast），而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状

态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

情形 4.4：$I_2=01$

第二个输入序列 $I_2=01$（7400）的情形，即在 input-0 输入端口和在 input-1 输入端口上的输入包分别是 bicast 包和 1-bound 包，这种情况下，1-bound 包交换到它的目的端口 output-1 输出端口，而 bicast 包则交换到另外一个输出端口 output-0 输出端口。所以，连接状态被设置成 bar（7401）并锁定。这样的话，输出端口 output-0 的第二个输出位为“0”，而 output-1 输出端口的第二个输出位为 1，即 $O_2=01$。结果，output-0 输出端口的输出包的带内控制信号为“10”（bicast），而 outpt-1 输出端口的输出包的带内控制信号为“11”（1-bound）。余下的包的资料位将直接贯穿由被锁定的连接状态提供的数据信道，从而正确地完成所期望的交换。这种情况下，$R_2=2$。

综上结果，依本发明而提出的编码方式，$max_j\{R_j-j\}$ 总是为 0，这样，相应的双点传送交换单元被最优化。

以上叙述的方法是以表 5 所代表举例的编码方式进行的。表 5 左边列中的 4 项，即“00”，“01”，“11”和“10”可以映像成其它可能的编码方式，只要这些方式能满足以下要求：

0-bound 包编码的第一个位与 1-bound 包编码的第一个位必须不同。

此外，当单点传送交换单元采用同上述双点传送交换单元编码方法中对“idle”、“0-bound”和“1-bound”相同编码时，总延迟时间完全主要取决于多阶互连网络的阶数与每一阶交换单元区域延迟时间的乘积。因此，当交换单元中所有区域延迟时间减少时，总延迟时间也会按比例减少。或者说交换单元中所有的区域延迟时间减小到某种比率，那么总延迟时间也会相应地缩减大致相同的比率。同样地，当交换单元最优化后，总的缓存要求也会降低很多。比较前面所阐明的两个例子，一个是随机编码方式，另一个是依据本发明提出的新编码方式，它们的不同的处是分别以区域缓存数 $max_j\{R_j-j\}$ 为 1 和 0 的交换单元堆积成整个多阶互连网络形式的交换引擎。

本发明虽已于此详细表示与说明，然先前的说明仅是例示发明的原理与精神，本领域的专业人士可依据本发明而轻易衍生出其它变化例，但这些变化例亦应包含于本案的范围内。此外，此处所述的实例与条件用语是为有助于读者了解本发明的原理与概念，而非用于将本发明限制于此范围。另外，

此处所有关于本案精神、形式与实施例的陈述以及其特定实例均包含与其构造与功能上均等者。同时，此均等涵盖过去已知与未来将发展的所有此等构造与功能。

此外，本领域的专业人士应了解，此处所用的方块图是一种举例说明实施本发明原理电路设计的示意图。

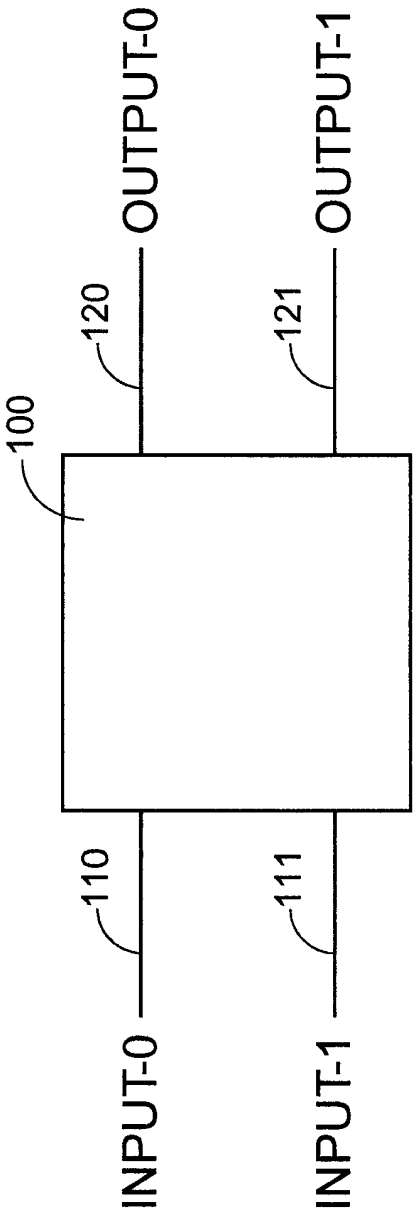


图1（A）

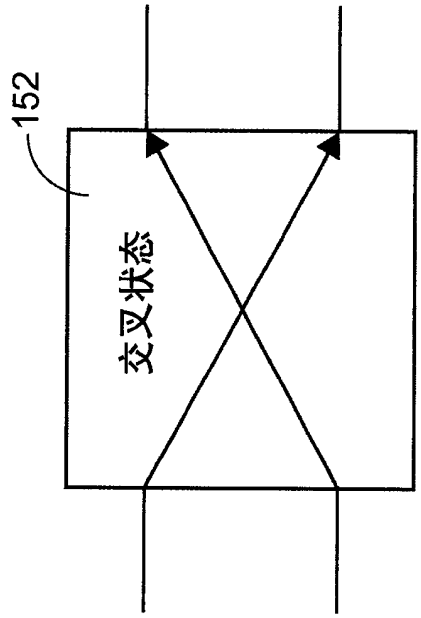


图1(C)

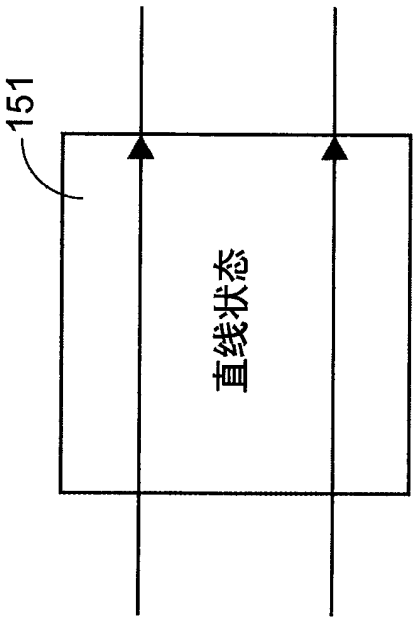


图1(B)

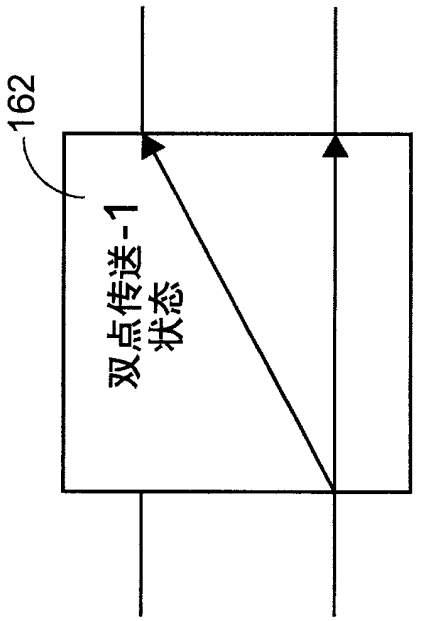


图1(E)

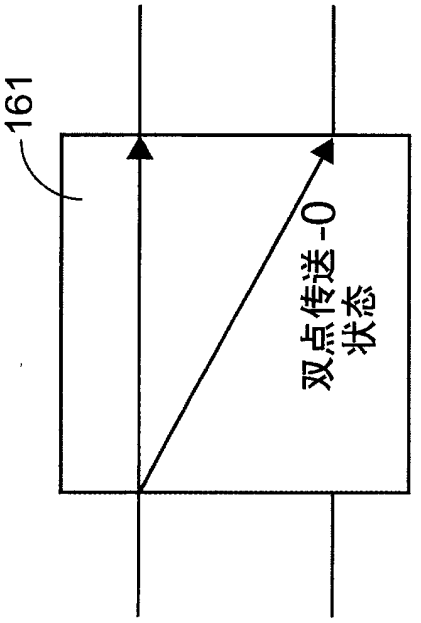


图1(D)

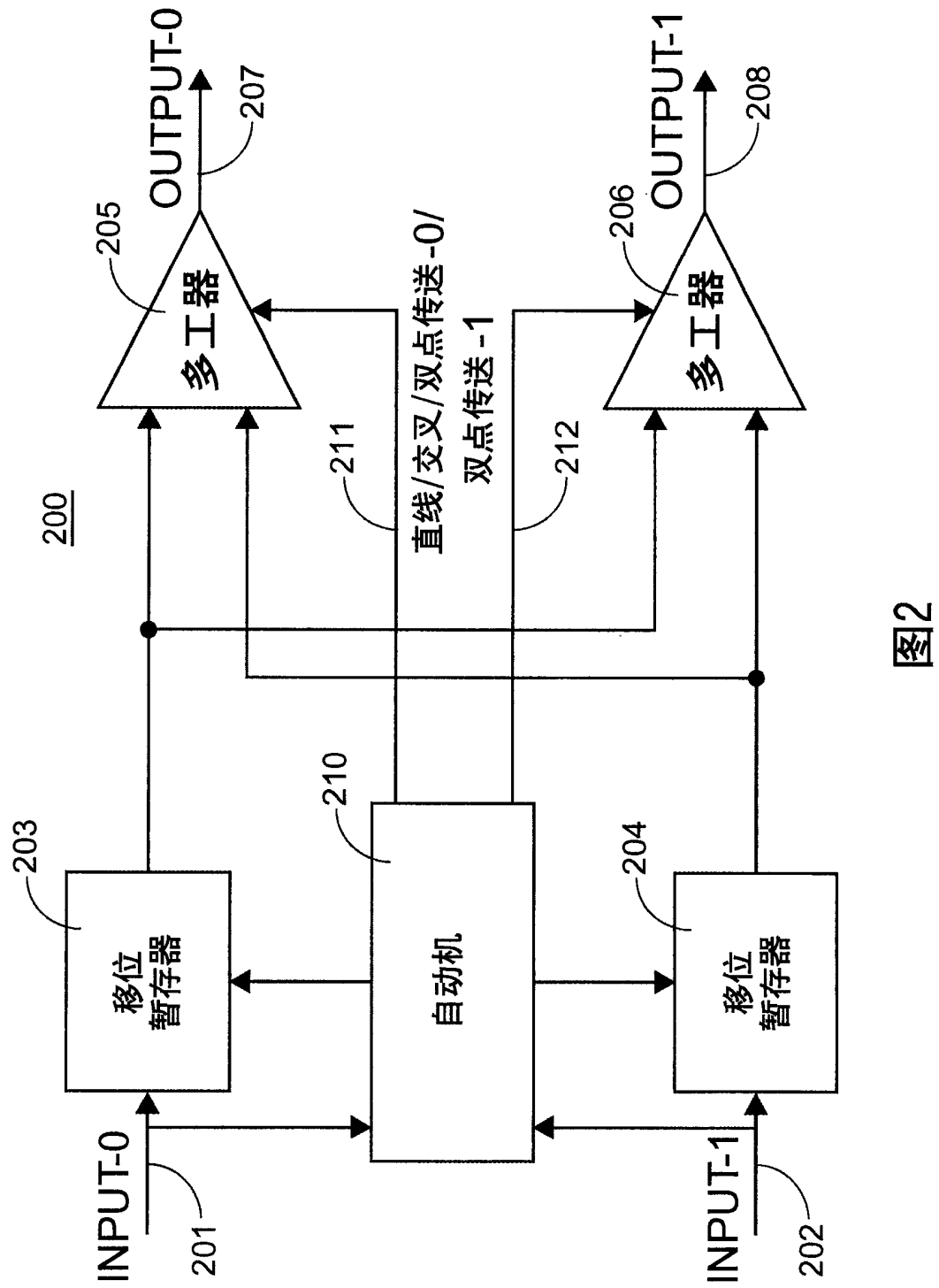

200
INPUT-0
201
INPUT-1
202
移位暂存器
203
移位暂存器
204
自动机
210
直线/交叉/双点传送-0/双点传送-1
211
212
多工器
205
多工器
206
OUTPUT-0
207
OUTPUT-1
208

图2

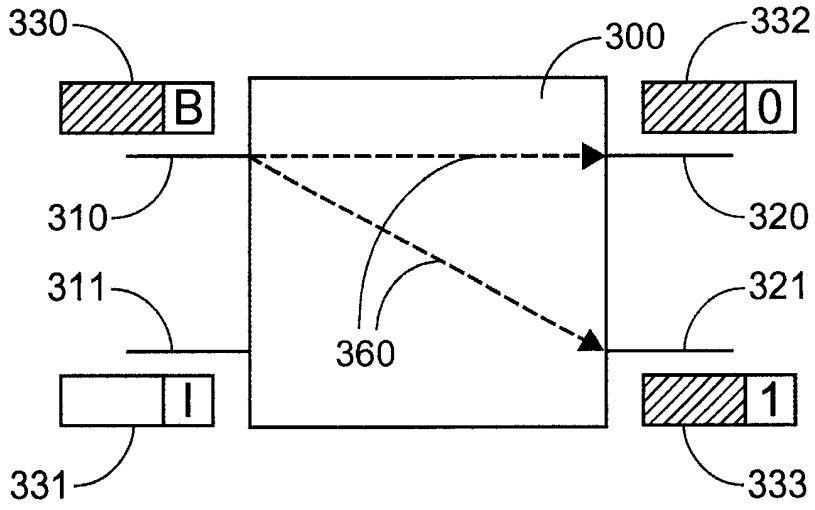

330
300
332
B
0
310
320
311
321
360
I
1
331
333

图3(A)

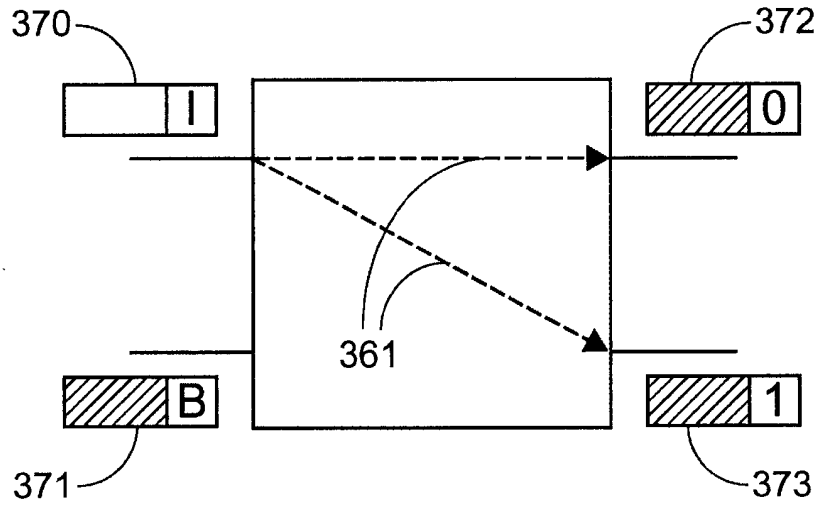

370
372
I
0
361
B
1
371
373

图3(B)

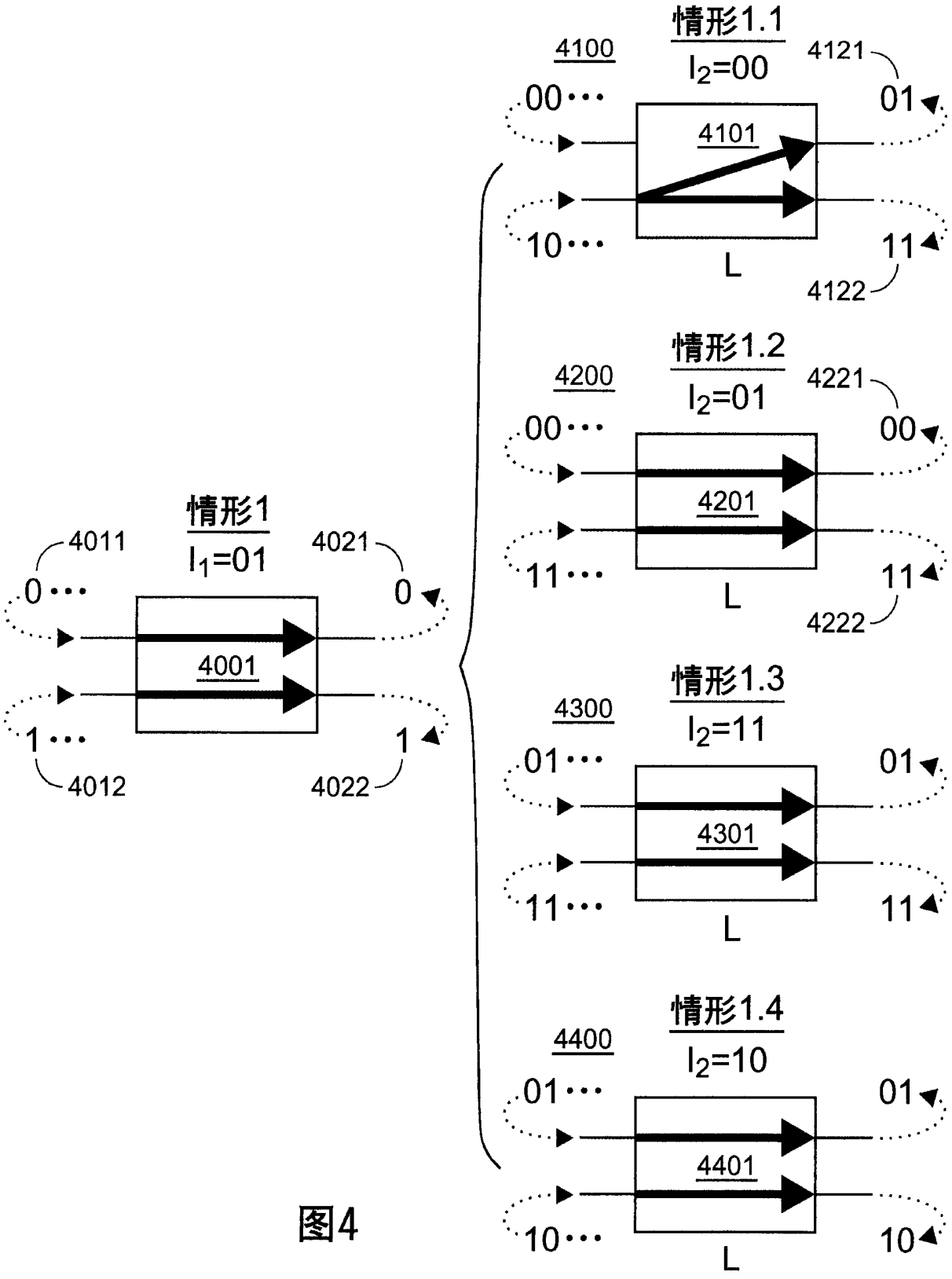

情形1
I_1=01
4011
4021
0
0
4001
1
1
4012
4022
情形1.1
I_2=00
4100
4121
00
01
4101
10
11
L
4122
情形1.2
I_2=01
4200
4221
00
00
4201
11
11
L
4222
情形1.3
I_2=11
4300
01
01
4301
11
11
L
情形1.4
I_2=10
4400
01
01
4401
10
10
L

图4

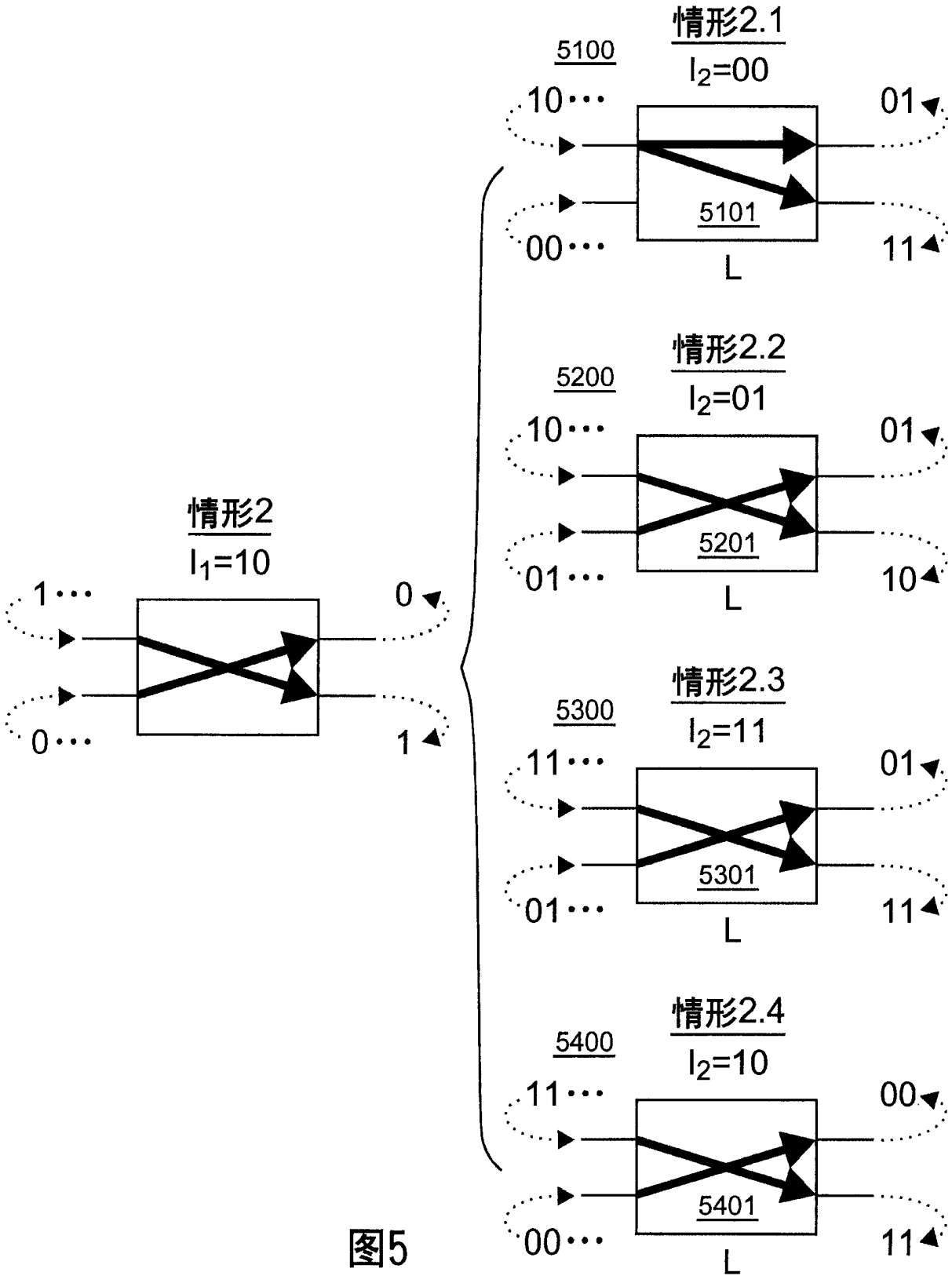

情形2
I_1=10
1•••
0•••
0
1
情形2.1
I_2=00
5100
10•••
00•••
01
11
5101
L
情形2.2
I_2=01
5200
10•••
01•••
01
10
5201
L
情形2.3
I_2=11
5300
11•••
01•••
01
11
5301
L
情形2.4
I_2=10
5400
11•••
00•••
00
11
5401
L

图5

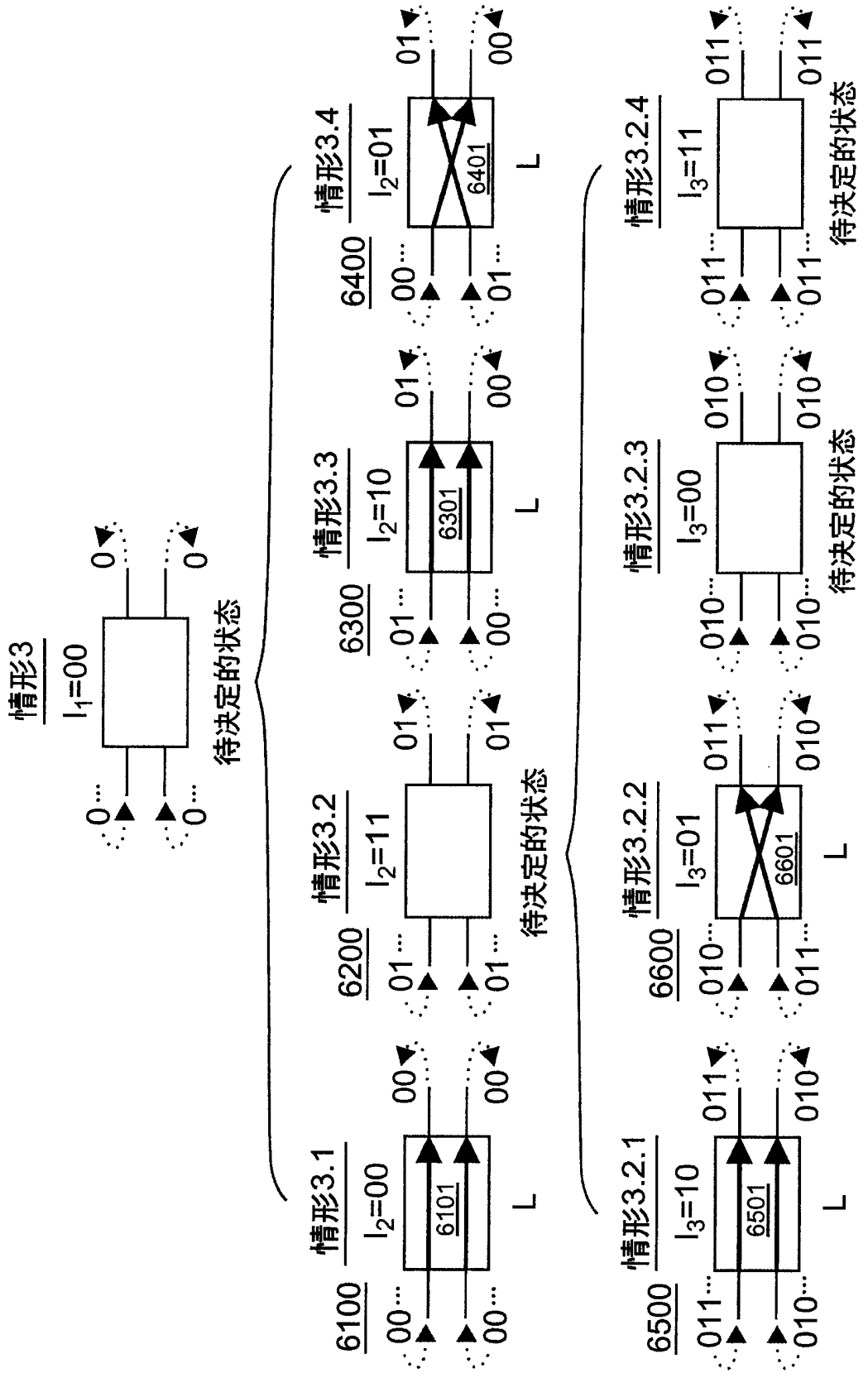

情形3
l_1=00
0
0
0...
0...
待决定的状态
情形3.1
6100
l_2=00
6101
00
00
00...
00...
L
情形3.2
6200
l_2=11
01
01
01...
01...
L
情形3.3
6300
l_2=10
6301
01
00
01...
00...
L
情形3.4
6400
l_2=01
6401
01
00
00...
01...
L
待决定的状态
情形3.2.1
6500
l_3=10
6501
011
010
011...
010...
L
情形3.2.2
6600
l_3=01
6601
011
010
010...
011...
L
情形3.2.3
l_3=00
010
010
010...
010...
待决定的状态
情形3.2.4
l_3=11
011
011
011...
011...
待决定的状态

图6

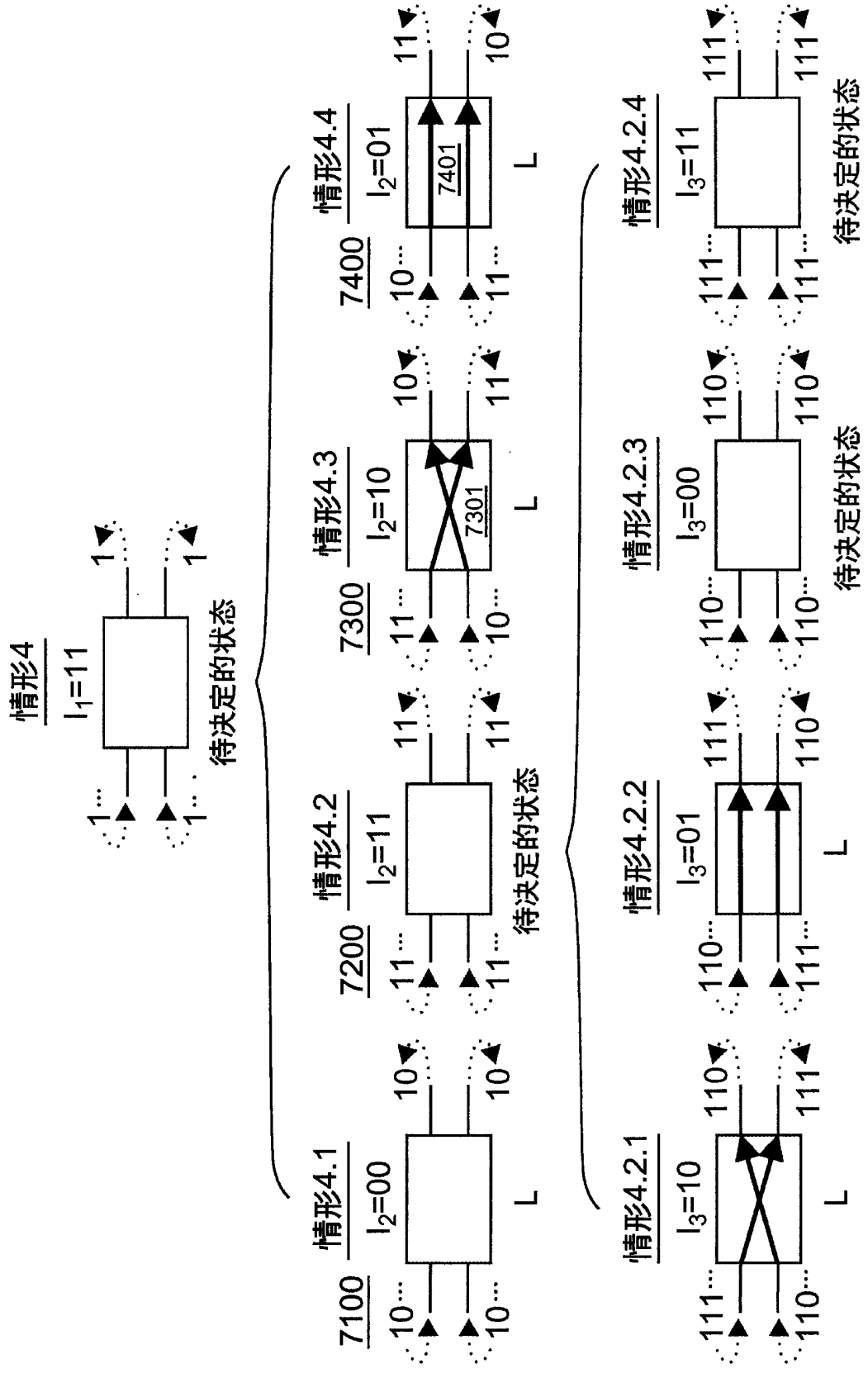

情形4
l_1=11
待决定的状态
情形4.1
l_2=00
7100
情形4.2
l_2=11
7200
情形4.3
l_2=10
7300
7301
情形4.4
l_2=01
7400
7401
情形4.2.1
l_3=10
情形4.2.2
l_3=01
情形4.2.3
l_3=00
情形4.2.4
l_3=11
L

图7