



(12) 发明专利

(10) 授权公告号 CN 1901439 B

(45) 授权公告日 2011. 04. 20

(21) 申请号 200610093430. 8

G06F 13/38 (2006. 01)

(22) 申请日 2002. 02. 25

(56) 对比文件

(30) 优先权数据

60/271124 2001. 02. 24 US

US 5459766 A, 1995. 10. 17, 全文.

US 5844908 A, 1998. 12. 01, 全文.

CN 1199967 A, 1998. 11. 25, 全文.

US 5917364 A, 1999. 06. 29, 全文.

(62) 分案原申请数据

02805440. 7 2002. 02. 25

审查员 赵晶晶

(73) 专利权人 国际商业机器公司

地址 美国纽约州

(72) 发明人 韦恩 M·巴雷特 陈东

保罗 W·科特尤斯 艾伦 G·加拉

罗里 D·杰克逊

杰勒德 V·科普克塞 本 J·内桑森

托德 E·塔肯

帕夫罗斯 M·弗拉纳斯

(74) 专利代理机构 北京市中咨律师事务所

11247

代理人 于静 杨晓光

(51) Int. Cl.

H04L 7/00 (2006. 01)

H04L 7/02 (2006. 01)

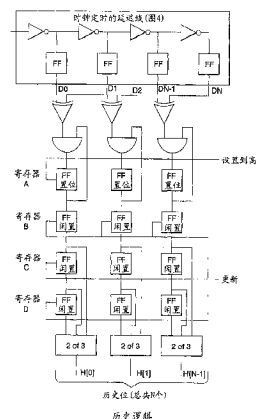
权利要求书 1 页 说明书 14 页 附图 10 页

(54) 发明名称

用于高速信令的数据捕获技术

(57) 摘要

一种用于高速信令的数据捕获技术, 以便于对异步数据流的最佳采样。该技术允许极高的数据速率, 不要求像在源同步系统中的那样要将一个时钟随数据一起发送。本发明也提供一种硬件机构, 用于为最佳双位并发双向 (SiBiDi) 信令自动地调节传输延迟。



1. 一种用于为两个节点之间的最佳双位并发双向信令自动地调节传输延迟的硬件机构，其在每一个节点中包括：

发送器，其在一个设定序列期间，在通信线路上以单向通信方式向另一个节点发送参数设置数据；

捕获器，其在所述设定序列期间，从通信线路以单向通信方式接收另一个节点发送的参数设置数据。

2. 权利要求 1 的硬件机构，其特征在于，在设定序列期间，所述发送器以比在并发双向信令期间更低的数据速率发送数据，其中使用 1 : n 的比率，保持“1”或“0”持续 n 位时间。

3. 权利要求 1 的硬件机构，其特征在于，每个节点在 n 个步骤中可能有范围从最小或零延迟到最大延迟的 n 个步级的不同延迟，因此延迟的可能的组合数是 $n*n$ ，以至选择一个最佳延迟组合要测试 $n*n$ 种可能的组合，该机构每次一个地循环经过所有 $n*n$ 个组合。

4. 权利要求 1 的硬件机构，其特征在于，一个差分数据线连接所述两个节点，所述发送器是 1 位发送器 CPU，所述捕获器是 1 位捕获 CPU。

5. 权利要求 1 的硬件机构，其特征在于，两个差分数据线连接所述两个相同节点，所述发送器是 2 位发送器 CPU，所述捕获器是 2 位捕获 CPU。

用于高速信令的数据捕获技术

[0001] 本申请是申请号为 02805440.7 的专利申请的分案申请。

[0002] 交叉参考

[0003] 本发明要求下述具有共同拥有人的、共同未决的美国临时专利申请的权益：申请序列号 60/271124，申请日 2001 年 2 月 24 日，名称“Massively Parallel Supercomputer(大规模并联超级计算机)”，特此将其全部内容如同全文叙述一样地合并在本文中作为参考。本专利申请还与下列具有共同拥有人的、共同未决的、具有相同申请日的美国专利申请有关，特此将这些美国专利申请的每个的全部内容如同全文叙述一样地合并在本文中作为参考：美国专利申请序列号(YOR920020027US1、YOR920020044US1(15270))，发明名称“ClassNetworking Routing(分级联网传送)”；美国专利申请序列号(YOR920020028US1(15271))，发明名称“A Global Tree Network for computing Structures(用于计算结构的全局树形网络)”；美国专利申请序列号(YOR920020029US1(15272))，发明名称“GlobalInterrupt and Barrier Networks(全局中断和闭锁网络)”；美国专利申请序列号(YOR920020030US1(15273))，发明名称“OptimizedScalable Network Switch(优化的可伸缩的网络开关)”；美国专利申请序列号(YOR920020032US1、YOR920020033US1(15258))，发明名称“Arithmetic Functions in Torus and Tree Networks(环形和树形网络中的算术功能)”；美国专利申请序列号(YOR920020033US1、YOR920020034US1(15259))，发明名称“DataCapture Technique for High Speed Signaling(用于高速信令的数据捕获技术)”；美国专利申请序列号(YOR920020035US1、(15260))，发明名称“Managing Coherence Via Put/Get Windows(通过放置/达到窗管理相干性)”；美国专利申请序列号(YOR920020036US1、YOR920020037US1(15261))，发明名称“LowLatency Memory Access and Synchronization(低等待时间存储器存取和同步)”；美国专利申请序列号(YOR920020038US1(15276))，发明名称“Twin-Tailed Fail-Over for Fileservers MaintainingFull Performance in the Presence of Failure(双拖尾故障-出现故障时的文件服务器维护全性能)”；美国专利申请序列号(YOR920020039US1(15277))，发明名称“Fault isolation ThroughNo-Overhead Link Level Checksums(通过无开销链接级校验和的故障隔离)”；美国专利申请序列号(YOR920020040US1(15278))，发明名称“Ethernet Addressing Via Physical Location for MassivelyParallel Systems(通过大规模并联系统的物理位置的以太网编址)”；美国专利申请序列号(YOR920020041US1(15274))，发明名称“Fault Tolerance in a Supercomputer Through DynamicRepartitioning(通过动态重分配的超级计算机故障容限)”；美国专利申请序列号(YOR920020042US1(15279))，发明名称“Checkpointing Filesystem(校验指示文件系统)”；美国专利申请序列号(YOR920020043US1(15262))，发明名称“EfficientImplementation of Multidimensional Fast Fourier Transformationon a Distributed-Memory Parallel Multi-Node Computer(在分布存储器并联多节点计算机上有效地实现多维快速傅利叶变换)”；美国专利申请序列号(YOR920010211US1(15275))，发明名称“A NovelMassively Parallel Supercomputer(新颖

的大规模并联计算机)”;美国专利申请序列号(YOR920020045US1(15263)),发明名称“SmartFan Modules and Systems(智能扇形组件和系统)”。

[0004] 发明背景

[0005] 1. 发明领域

[0006] 本发明总体涉及用于高速信令(signaling)的数据捕获技术,更具体来说,涉及一种用于异步数据流的最佳采样的技术。该技术允许极高的数据速率,并且不要求像在源同步系统中那样要将一个时钟随同数据一起发送。

[0007] 本发明也提供一种用于为最佳双位并发双向(SiBiDi--two-bitsimultaneous Bi-Directional)信令自动地调节传输延迟的硬件机构。

2. 背景技术

[0008] 大规模并行计算机系统能进行类别宽广的重要计算。这种系统由许多相同的计算机节点组成,每个计算机节点一般有一个或多个CPU、存储器和一个或多个用于将该节点与其它节点连结的网络接口。

[0009] 在相关的美国临时专利申请(申请序列号60/271124,申请日2001年2月24日,名称“Massively Parallel Supercomputer(大规模并联超级计算机)”)中描述的计算机,利用芯片上系统(SOC)技术,建立一个具有高通过量(throughput)的、可伸缩的(scalable)、成本-效用有益的计算系统。SOC技术已经使利用包括具有集成的第一级高速缓存的CPU的内置部件库在单一芯片上建立整个多处理器节点成为可行的。这种包装大大地减少了节点的部件数,便于建造可靠的大规模机器。

[0010] 本发明涉及用于诸如生命科学领域中的应用等各种应用的大规模并行计算机系统的领域。更具体来说,本发明涉及高速信令的领域,涉及单向信令或并发双向(SiBiDi)信令。

[0011] 有这样的情形:要求进行大量的数据传输,但是能用的连线数量有限。并发双向信令允许用相同的连线同时传送和接收信号。这就将连线的数量降低一半。需要进行大量数据传输但电缆数量受到严重限制的一个例子是具有数千个通过连线通信的处理器的大规模并行超级计算机。

[0012] SiBiDi信令的工作方式是在接收数据的相同连线上发送数据。因此,在接收期间,不仅接收从导线另一端发送的需要的数据,也接收它刚刚发出的数据。这当然破坏所需要的信号。然而,由于刚刚发出的数据是已知的,因此可以“将其减去”。这是由标准的SiBiDi电路进行的。

发明内容

[0013] 因此,本发明的主要目的是提供一种用于高速信令的数据捕获技术,尤其是允许在不将时钟随数据一起发送的情况下对异步数据流最佳采样和捕获。捕获数据的方法是,沿着具有一系列延迟抽头(taps)的时钟定时的(clocked)延迟线发送数据流的串行位,并利用一个时钟采样所有的延迟抽头。将每个延迟抽头输出与相邻的延迟抽头输出进行比较,确定是否相同,该比较结果被用来形成一个时钟定时的串,以生成一个数据历史记录,该数据历史记录被加以检查,以确定最佳数据捕获眼(capture eyes),所述确

定最佳数据捕获眼的方法是查找数据在相邻的延迟抽头之间不变换 (transition) 的地方的数据捕获眼, 把它们作为最佳数据捕获眼。

[0014] 本发明的另一个目的是提供一种硬件机构, 它用于为最佳双位 SiBiDi 信令自动地调节传输延迟, 以便改善双位 SiBiDi 信令的信号质量。实行一个特殊的硬件算法, 并将该两个位中的每一位用于单向信道 (channels), 以允许两个节点的硬件算法在设定序列期间安全地交换设置参数。一个相同频率的单向信道有 SiBiDi 信道的带宽的一半, 但是有相当好的信号质量。

附图说明

[0015] 本领域的熟练人员参照以下结合附图对若干实施例的详细说明, 能更好地理解用于高速信令的数据捕获技术的本发明的上述目的和优点。附图中相同的部件用相同的标注号指明。

[0016] 图 1 表示一个数据接收宏, 它能捕获 2G 位速率的串行数据并将其带入本地时钟域。

[0017] 图 2 表示一个数据发送宏块, 其接收 500Mhz 1 字节宽的数据输入, 并生成每个是 2Gb/s 串行数据的两个数据流的数据输出, 该数据输出被在 SiBiDi (并发双向) 差分 (differential) 数据链路上传送, 然后数据被输入到图 1 的数据捕获宏。

[0018] 图 3 表示一个级内 (instage) 2 位宏块。

[0019] 图 4 表示一个时钟定时的延迟线的实现, 其中串行数据经过组合的串行反相器组, 每个反相器增加一个延迟增量。

[0020] 图 5 表示一个历史逻辑块, 它被表示为时钟延迟线的延伸, 并显示一个时钟相位 (phase)。

[0021] 图 6A 和 6B 表示数据和历史样本 MUXs, 它们每个对应于图 3 底部所示的 MUX, 图 6A 和 6B 分别用于先导边沿时钟相位和下降边沿时钟相位, 图 6C 的 MUX 接收图 5 的底部的输出信号。

[0022] 图 7 表示要被用作能在每个时钟周期捕获 4 个数据位的配对链路 (paired link) 的串行位组合和字节对准逻辑, 它含有两个 2 位宏 (如图 3 中所示的) 并在 2 个输入数据流之间寻找适当的字节对准。

[0023] 图 8 表示一个眼检测流程图中的眼检测过程。

[0024] 图 9 表示两个 2 位宏状态图的第 1 图, 该图表示在眼位置样本点正被确定期间的相位的状态流。

[0025] 图 10 表示两个 2 位宏状态图的第 2 图, 该图表示在正常数据捕获操作期间的重复状态流。

[0026] 图 11 表示一个第一实施例, 其中两个差分数据线连接一对相同节点 1、2, 并且其中每个节点都有一个唯一的 ID, 每个节点与一个 2 位发送器 CPU 和一个 2 位接收 CPU 一起操作。

[0027] 图 12 表示一个在两个节点 1、2 之间的 SiBiDi 电通信的第二实施例, 其中一个差分通信线路连接这些节点。

[0028] 图 13A 和 13B 的状态机表示在同步 Si-Bi-Di 连接的训练中由节点计算芯片采取

的步骤。

[0029] 发明详细说明

[0030] 概述

[0031] 本发明旨在用于在大规模并行超级计算机中实现互连，它解决计算机产业中两个长期存在的问题；(1) 在处理器和存储器之间以时钟周期计量的日益增加的距离，和 (2) 由主流单处理器或对称多处理器构造的并行计算机的高能力密度。

[0032] 本发明总体涉及用于高速信令的数据捕获技术，更具体来说，涉及一种用于异步数据流的最佳采样的技术。该技术允许极高的数据速率，不要求像在源同步系统中的那样要将一个时钟随数据一起发送。

[0033] 串行链路勘测

[0034] 连接大规模并行超级计算机的串行链路的目标带宽是 1.4Gb/s (每个方向)。这个带宽必须是双向的。这个双向要求可以以许多方式处理。所有情况共有的限制是，它们是低功率和低成本的。选择的实现将被集成到处理节点内的 ASIC 中。与这种方法相关联的一个特别问题是低功率限制。加上缺少用于链路传输的相对相位 (phase) 信息，这消除了标准的 PLL 时钟和数据恢复设计。在这种情况下，必须在不用 PLL 的条件下高度可靠地从数据本身抽取相位。

[0035] 数字数据捕获

[0036] 概述

[0037] 本说明书详细地描述一种数字数据捕获技术。图 1 表示一个能捕获 2G 位速率的串行数据并将其带入本地时钟域的数据接收宏。目标是用少量的信元 (cells) 低功率地、可靠地实现这一点。图 2 表示一个比数据接收宏显著简单的数据发送宏块，它将在本说明书的第二部分被说明。这描述一种 DDR (双数据速率) 类型的数据恢复，它允许一个是位时间的频率的一半的内部时钟。这能被用于 SDR (单一数据速率) 模式或者在需要时被扩展到四数据速率方案。

[0038] 参看图 1，至数据接收宏的数据输入是两个 2Gb/s 输入串行数据的数据流，这代表着总共 4Gb/s 的数据流；数据输出是一个 500Mhz 的一字节宽的 (8 位宽的) 数据流。其它输入信号是：一个 1Ghz 的时钟，一个把数据捕获宏复位到已知状态的复位信号，一个用于指导宏寻找最佳眼 (沿多抽头延迟线 (见图 3、4、5) 上的位置或阶段—数据在该处不经历变换、因此是最不可能有数据错误的最准确的数据捕获位置) 以恢复数据的训练信号，一个空置字节信号，它是在数据未被接收时被宏接收的预定空置数据模式 (idle data pattern) 的，一个指令宏在双数据速率方式下操作的 DDR 方式，以及一个是寻找最佳数据或空置眼的约束参数的最小距离。其它输出信号包括，指示空置模式的有效接收的有效空置信号，一个指示已经检测到最佳眼位置和参数的眼 _ 找到信号，一个指示最佳眼位置被锁定的锁定信号，以及一个指示最佳眼位置由于离多抽头数字延迟线的一端太近而处于被丢失的危险中的警告信号。

[0039] 参考图 3、4、5、6 和 7 进一步解释图 1 的宏。

[0040] 接收宏中的延迟在 7 和 12 个位时间 (bit times) 之间，视数据的字节相位 (byte phase) 而定。通过跳过字节输出，可以将延迟削减到 5 至 6 个位时间。对于数据内容可以被忽略的信号再驱动 (redriving) 来说，这是一个合理的方法。

[0041] 图 2 是一个数据发送宏块，其接收 500Mhz 1 字节宽的数据输入，并生成每个是 2Gb/s 串行数据的两个数据流的数据输出，该数据输出被在单向或 SiBiDi（并发双向）差分数据链路上传送，这在下面说明，然后数据被输入到图 1 的数据捕获宏。 另外的一个输入是字节有效信号，它指示一个有效的数据信号正在被接收并要被发送，这与空置信号不同；另外的输入是一个 1GHz 时钟信号和一个把数据发送宏复位到某个已知状态的复位信号。

[0042] 图 3、4 和 5 表示数据接收和捕获。 捕获数据的方法是沿快速抽头延迟线（见图 3、4 和 5）发送数据位，并用本地时钟采样所有的抽头。 将每个抽头与其邻居（见图 3、5）比较，看看是否相同。 这些比较综合起来形成一个时钟定时的串，该串与以前的时钟定时的串组合起来生成一个能被用来确定最佳采样点的历史。 可以通过从历史串中（见图 5，寄存器 A、B、C、D）查找那里的数据在延迟抽头之间不曾改变的区域而找出最佳采样点，它们在本文中被成为“眼”。 历史被定期地更新，例如每隔一个本地时钟就更新。 定期更新弥补了变化的参数，诸如不同部件中的温度或电压的变化。 还将有三个额外的不常被更新的“眼”流水线化的寄存器（见图 5，寄存器 B、C、D）。 这允许一种具有可编程的持续期间并且不受被隔离位错误的影响的捕获方案。 持续时间可以被设置得任意长，但是必须不短于为可靠地采样数据边沿所需的最大时间。 为了容纳快于本地时钟周期的位采样时间，用时钟的两个边沿来捕获 DDR 数据（见图 3、4）。 时钟的每个边沿有其自己相关联的捕获寄存器和独立的逻辑来寻找最佳眼。 这个技术因此大大地免于本地的和发送端时钟占空度（duty cycle）的不对称的影响。

[0043] 由于历史寄存器将改变，最佳采样点也将移动。 这个更新应当在比持续时间短的时间范围进行。 这个功能是在直方图和采样点确定单元中进行的。

[0044] 数据捕获的方法牵涉到一个两阶段初始化，它或者在系统复位后或者在单独的“训练”信号被确定后进行。

[0045] 阶段 1：在复位或“训练”信号（见图 1、2）后，历史寄存器被刷新，在所有 3 个“眼”寄存器中获取一个新的历史模式（pattern）。 在获取一个有效的“眼”寄存器集合后，通过状态机序列（见图 9、10）确定最佳采样点。 这是按时钟的每个相位独立地进行的。 这些采样点然后被使用，并在每一个系统时钟将两个位传送到下一个阶段。

[0046] 阶段 2：两个位被接收并被插入到一个移位寄存器中，该移位寄存器与一个筒式位移器一起被用于允许适当的半字节（1/2 字节或 4 位）边界（见图 7）。 这些边界是通过在初始化序列期间使用唯一性的半字节模式而找到的。

[0047] 时钟定时的延迟线块

[0048] 图 4 和 5 表示一个时钟定时的延迟线的实现，其中串行数据进入左边的反相器 I，经过组合的反相器串，其中每个反相器增加一个延迟增量。 在每个反相器输出端有两个寄存器锁存器 FF，见图 3-5，一个被时钟的正边沿定时，另一个被负边沿定时。 这允许该逻辑以两倍的时钟速度捕获数据。 一组锁存器捕获正时钟相位的数据眼，而另一组锁存器捕获负时钟相位的数据眼。 两种眼被单独地检测和采样，这样，每个时钟相位需要一个单独的、如图 5 中所示的电路。 独立的正、负时钟定时的逻辑电路导致对时钟信号的周期工作循环、尤其是本地的和发送端时钟工作循环的不对称的很小依赖性。

[0049] 该模块有在输入接收器之后的高速信号作为其输入。 该模块的唯一的其它输入

是向所有触发器扇出相等时间的本地时钟。该模块的输出只有 $N+1$ 个时钟定时的延迟抽头 $D[0:N]$ 。每个抽头约 50ps，在上升边沿和下降边沿之间具有较好的匹配。下降延迟与上升延迟之间所需的匹配大约是 20-30%。我们要求时钟对所有相邻锁存器是约 10ps 内的相等时间。这可以用抽头的时钟线而不是时钟树更好地实现。这些数据捕获电路中的许多都是可以实现的，因此功率是关键。

[0050] 该模块是布局关键的，因此要求额外的布局考虑。

[0051] 为了测试芯片的目的，将部件的数量固定在 32。这给出大约 1.6n 秒的额定总延迟，这足以捕获频率下降到大约 1Gb/s 的 DDR 数据。

[0052] 参看图 3 和 5，特别是图 5，每个寄存器 FF(触发器)被导向到一个异或门 XOR，该异或门也接收来自延迟线中下一个寄存器 FF 的输入。参看延迟线的第一和第二级，由于数据位在进入第二寄存器之前被第二反相器倒置，如果数据位不经过在两个连续级之间的转换，则第一和第二寄存器将保持相反的值，使得第一级 XOR 将产生一个 1，指示没有级与级之间的转换。相反，如果数据位在两个连续级之间经过转换，则第一和第二寄存器将保持相同的值，使得第一级 XOR 将产生一个 0，指示有级与级之间的转换。

[0053] 图 5 的系统正在沿着时钟定时的延迟线搜寻一个稳定位置或眼，以检测数据不经历转换处的数据，这是由来自一系列的连续 XORs 的一系列的 1 输出指示的，这样就应当将数据检测眼对准到一系列 1 的中间。每个 XOR 的输出被输入到一个 AND 门，AND 门的输出被输入到寄存器 A，寄存器 A 是一系列 FF(触发器)历史寄存器 A、B、C 和 D 中的第一个。第一寄存器 A 被以满 1Ghz 时钟速率采样，并以较慢的时钟速率(例如 $> 1\text{ms}$ (微秒))定期地被一个“设置到高”的信号复位到高，而寄存器 B、C 和 D 则以与“设置到高”相同的时钟逻辑速率被采样和更新。

[0054] 寄存器 A 被时钟以 $a > 1\text{ms}$ 的时钟速率设置或复位到高 1 输出，在复位后，如果 XOR 的输出是 1，则 AND 门的输出是 1，且寄存器 A 的输出是 1，它随后(被寄存器 B、C 和 D 的负载(1d)输入端的更新信号)顺序地被时钟定时地(clocked)通过寄存器 B、C 和 D。相反，如果 XOR 的输出是 0，且寄存器 A 的输出被设置或复位至 1，则 AND 门的输出是 0，且寄存器 A 输出一个 0，它随后被顺序地时钟定时通过寄存器 B、C 和 D。该安排使得一旦寄存器 A 的输出是 0，则它保持为 0，一直到寄存器被“设置到高”信号复位，使得寄存器 B、C 和 D 的每个的输出被顺序地时钟定时至 0，并保持为 0，一直到寄存器 A 被“设置到高”信号复位到 1。

[0055] 历史寄存器 B、C 和 D 的每个的输出是一个 3 之 2(2 of 3) 逻辑元件的输入，3 之 2 逻辑元件在其 3 个输入的任何 2 个是 1 时产生一个 1 或高(H) 输出。3 之 2 逻辑的目的是补偿通过数字延迟线的数据流中的可能错误地导致 3 个输入的其中之一为 0 的假信号(glitches)，使得尽管有数据假信号也能产生准确的输出。此外，可以计算 3 之 2 逻辑检测的出现次数并加以报告，作为正在被接收的数据的完整性的一个指示。H 输出(0 至 $N-1$) 是图 6C 中所指示的 MUX 的输入。H 输出中的一个串 1 一般表示是数据采样眼的一个好候选，应当被置于该串 1 之中的中心。

[0056] 图 3 表示一个级内 2 位宏块，其大部分已经结合图 5 作了描述。直方图(histogram)和采样点确定单元包括图 5 的 AND 门、历史寄存器和 3 之 2 逻辑以及图 6C

的 MUX 以及图 9 和 10 的状态图。宏块也包括一个 MUX，它接收数字延迟线的 FF 寄存器的所有输出作为输入，并在图 5 的底部的信号 H[0]、H[1]、H[N-1] 的控制下选择性地将这些输入作为数据输出而传送出去。

[0057] 历史块

[0058] 图 5 表示一个历史逻辑块，它被表示为时钟延迟线块的延伸，并显示一个时钟相位。另一个时钟相位需要一个相同的电路。图 5 中所示的反相器串对两个时钟相位是共同的。

[0059] 采样点块

[0060] 采样点块最容易用一个状态图来描述，状态图确定两个最佳采样点，每个时钟相位对应一个。由于采样点不是被频繁更新的（更新之间至少有 50 个时钟），我们可用一个时钟过程来寻找最佳采样点。

[0061] 图 6A 和 6B 表示数据和历史样本 MUXs，采样 MUXs 分别具有 D_even[N:0] 和 D_odd[N:0] 的输入，每个输入对应于图 3 底部所示的 MUX，每个分别对应先导边沿时钟相位和下降边沿时钟相位，而图 6C 的 MUX 则如前所述地接收图 5 的底部的信号。

[0062] 组合两个 2 位宏

[0063] 图 7 表示要被用作能在每个时钟周期捕获 4 个数据位的配对链路的串行位组合和字节对准逻辑，它含有两个 2 位宏（如图 3 中所示）并在向两个 2 位宏 70（如图 3 中所示）输入的频率各为 2Ghz 的 2 个输入数据流之间寻找适当的字节对准（alignment）。这两个输入信号可被视为图 1 的被与适当的字节对准组合的输入信号。图 3 的两个 2 位宏的输出是两个数据流，每个速率 1Ghz，它们被输入到寄存器 71，后者延迟并标准化这 2 个数据流，这 2 个数据流被输入到一个 12 位的位移寄存器 72，位移寄存器 72 被以半速 Clk/2 的时钟定时，将这 2 个数据流转换成 500Mhz 的 12 位宽数据流。这 12 位宽数据流被输入到逻辑 73 和筒形位移器 74，筒形位移器 74 具有两个 12 位宽数据流的 24 位的输入，实质上在逻辑 73 的控制下选择 24 位中被适当对准的 8 位，以确定筒形位移器的正确的位位移。逻辑 73 用一个已知的训练模式（train pattern）来生成两个 4 位宽的、控制筒形位移器的输出。逻辑 73 实质上在软件的控制下通过时钟定时的延迟线不断再发送相同的已知数据训练模式，直到它知道筒形位移器的正确的位位移。筒形位移器在该逻辑的控制下选择性地挑选 24 位输入中的最适当对准的 8 位，作为字节输出传送。

[0064] 状态图：

[0065] 眼检测流

[0066] 图 8 表示一个眼检测流程图中的眼检测过程的总流程，从复位 80 开始，把系统初始化到已知值，紧接着在框 81 等待足够数量的数据变换，以保证眼的干净确定。框 82 检查偶数眼历史寄存器中的 1 和 0，以确定一个对应于上升沿的时钟定时数据的偶数眼，并选择一个具有最小延迟的第一眼通过时钟定时的延迟线。框 83 对奇数眼做相同的事，检查奇数眼历史寄存器中的 1 和 0，以确定一个对应于下降沿的时钟定时数据的奇数眼，并选择一个具有最小延迟的奇数眼通过时钟定时的延迟线。

[0067] 可能有几个不同的对应于延迟线上不同位置的偶数相位和奇数相位眼，并在检测到最小延迟眼后，流程图从框 84 重新循环到框 82，寻找具有下一个最大延迟的下一对眼；而逻辑控制再继续再循环到框 82，一直到整个长度的延迟线被作了对应偶数和奇数相

位的眼的检查。框 84 比较每个下一个检测到的眼对与最佳的以前检测的眼对，保留最佳眼对，这样，它就选择所有候选眼对的最佳眼对，这个功能由图 7 的逻辑 73 执行。

[0068] 在这一点，逻辑 73 在框 85 等待一个“对准”包，这是一个已知的训练模式，诸如一个已知的位序列，用于确定此时尚未知的字节边界。在字节边界由图 7 中的逻辑 73 用对准训练模式确定之后，然后框 86（它也对应于逻辑 73）设置对准输入到筒形位移器，用以对准筒形位移器 74，选择和传送正确对准的 8 位作为字节输出，并且逻辑 73 也生成锁定的输出信号。筒形位移器于是准备好在正常的数据接收期间传送 8 个正确对准的位作为字节输出。

[0069] 框 87 指示数据采样眼是不断被更新的。一个优选的再对准在现有的偶数和奇数数据采样点处开始，然后检查现有眼的左右，以确定左眼边和右眼边，然后在偶数和奇数相位眼的左边和右边之间再对准偶数和奇数相位眼的中心，如结合图 9 所解释的那样。

[0070] 训练眼检测的状态图

[0071] 图 9 表示两个 2 位宏状态图的第 1 图，该图表示在眼位置样本点正被确定期间的相位的状态流，总体对应于图 8 的框 82 和 83。

[0072] 在图 9 中，对应于图 5 底部输出的图 6C 的对 MUX 的 $H[N-1:0]$ 输入被顺序采样并逐一检查，作为图 6C 的 MUX 的被传送的输出 H_{samp} 。第一位 $H[0]$ 要么是 0，要么是 1，0 指示在眼的外部数据变换 (transition)，1 指示眼内不可能有数据变换；第一位 $H[0]$ 被图 9 的步骤序列检查，其中 0 被显示为 $\text{Samp}(\text{样本}) = 0$ ，1 被显示为 $\text{Samp} = 1$ 。在第一位 $H[0]$ 被图 9 的所有步骤 $S0$ –总结检查后，第二位 $H[1]$ 通过相同的步骤序列被检查，如此等等，直到最后一位 $H[N-1]$ 被检查。

[0073] 两个两位宏状态图之一在图 9 中的状态如下所述：

[0074] • $S0$ 是复位状态。在复位活动期间，控制依然在这里。

[0075] • $S1$ 是初始化状态。在复位被释放之后，控制在这里等待，直到一个更新计数器到时，然后，如果 $\text{samp} = 0$ （表示图 5 的被检查的抽头产生一个 0，因此不在眼中），前进到 $S2$ ，如果 $\text{samp} = 1$ （被检查的抽头输出是 1 并且可能是某个眼的一部分），前进到 $S3$ 。

[0076] 状态 $S2$ – $S3.5$ 通过递增搜寻整个偶数延迟线历史而查找一个偶数眼，它相应于上升时钟沿定时的数据。

[0077] • $S2$ 通过从左至右递增通过偶数延迟线历史查找一个偶数眼 ($\text{samp} = 1$)。它寻找偶数眼的左端。如果找到或者如果它命中延迟线的右端，则前进到状态 $S3$ ，否则停留在 $S2$ 。

[0078] • $S3$ 搜寻偶数眼的右端，仍然搜寻，直到它发现 $\text{samp} = 0$ 。 $\text{Samp} = 1$ 期间控制留在 $S3$ 中，因为它在眼内。当发现 $\text{smap} = 0$ 时，控制进到 $S3.5$ ，这是控制在某些情况下正确工作所必需的延迟状态。

[0079] • $S3.5$ 立即转换到 $S4$ 。

[0080] 状态 $S4$ – $S13$ 检索对应于下降时钟边沿时钟定时的数据的奇数眼，对奇数眼的搜寻在被检测的偶数眼的中心开始，比对偶数眼的搜寻更加复杂。

[0081] • $S4$ 开始搜寻奇数眼。对于正常的方式，如果奇数眼 $\text{samp} = 0$ ，它前进到 $S7$ ，

否则 $\text{samp} = 1$ ，它前进到 S5。 $\text{samp} = 0$ 意味着奇数眼在奇数延迟线历史中不与偶数眼对准，正常的策略是向左和向右搜索，挑选最接近的奇数眼。 有两个可替代方式。 控制进到 S13 的，则向右搜索；控制进到 S12 的，则向左搜索。

[0082] • S5 意味着奇数眼与偶数眼是对准的，且初始样本点已经在奇数眼中。 因此这个状态通过递减一个样本计数器（用于选择数据点的）从初始样本点向左搜索，一直到它找到奇数眼的左端或者延迟线的左端。 然后，控制进到 S6。

[0083] • S6 向右搜索，寻找奇数眼的右端。 当它找到奇数眼的右端或者延迟线的右端时，控制进到总结状态，该状态是步骤 S13 之后搜索的结束的状态，数据样本计数器在这里被设置得对应于偶数和奇数眼的正常处理。

[0084] • S7 意味着奇数眼与偶数眼是不对准的，且初始样本点在噪声区中任何眼之外。 这个状态通过递减样本计数器向左搜索一个匹配的奇数眼。 当 $\text{samp} = 1$ 时，它已经找到一个奇数眼的右边，它把该右边记在 MAX（右边寄存器）中，然后，控制进到 S8。 或者，它到达延迟线的左端都没有找到左奇数眼，在这种情况下，控制进到 S11。

[0085] • S8 旨在搜索一个不对准的右奇数眼。 它在 $\text{samp} = 0$ 期间不断搜索，一直到 $\text{samp} = 1$ ，这表示一个右奇数眼的左端被找到并被记在 MIN（左边寄存器）中，然后，控制进到 S9。 如果到达延迟线的右端都没有发现 $\text{samp} = 1$ ，则没有右眼，因此控制直接进到 S10 中的左眼处理。

[0086] • S9 中比较 MIN 和 MAX 与右眼的距离。 如果 MIN 较近，控制进到 S6。 如果 MAX 较近，控制进到 S10。

[0087] • S10 搜索奇数眼的左端。 在 $\text{samp} = 1$ 期间，控制保持在 S10。 当 $\text{samp} = 0$ 或者到达延迟线的左端时，控制传送到总结。

[0088] • S11 在没有左奇数眼时搜索一个不对准的右奇数眼。 在 $\text{samp} = 1$ 期间，控制保持在 S11。 $\text{samp} = 1$ 时，控制进到 S6。 如果在发现 $\text{samp} = 1$ 之前到达延迟线的右端，则没有奇数眼。 这是一个错误条件，由警告指示器检测并指示。

[0089] • S12 搜索延迟线中偶数眼左边的不对准的奇数眼。 在 $\text{samp} = 0$ 期间，控制保持在 S12。 $\text{samp} = 1$ 时，控制进到 S10。 如果在发现 $\text{samp} = 1$ 之前到达延迟线的左端，则没有左眼，然后，除非控制是从 S13 进入到 S12 的一此时没有奇数眼，否则控制选择在 S13 中向右查找。 这是一个错误条件，由警告指示器检测并指示，然后控制进到总结。

[0090] • S13 搜索延迟线中偶数眼右边的不对准的奇数眼。 在 $\text{samp} = 0$ 期间，控制保持在 S13。 $\text{samp} = 1$ 时，控制进到 S6。 如果在发现 $\text{samp} = 1$ 之前到达延迟线的右端，则没有右眼存在，然后，除非控制是从 S13 进入到 S12 的一此时没有奇数眼，否则控制选择在 S12 中向左查找。 这是一个错误条件，由警告指示器检测并指示，然后控制进到总结。

[0091] • 总结是眼数据采样点被确定用于正常操作时的状态。

[0092] 且有眼样本点再对准的正常操作的状态图

[0093] 图 10 表示两个 2 位宏状态图的第 2 图，该图表示在正常数据捕获操作期间的重复的状态流。 在这个阶段（phase），逻辑要捕获串行数据并将其转换到字节并行格式。

[0094] 图 10 表示正常数据捕获运行状态。 控制一般驻留在 RUN0。 完成更新（done

update) 信号由一个时钟计数器控制定期地使控制能进入 RUN1。状态 RUN1 和 RUN2 递增和递减到偶数眼的两极, 它们是可以改变的。记住新的极限。类似地, 状态 RUN3 和 RUN4 递增和递减到奇数眼的两极, 且极限被记住。状态 RUN5 使用 RUN1 至 RUN4 的发现来计算新的数据采样点, 新的数据采样点被锁存起来供使用。控制然后返回到 RUN0, 进入下一个更新周期。

[0095] SiBiDi

[0096] 本发明也提供一种硬件机构, 用于为最佳双位并发双向信令自动地调节传输延迟。

[0097] 在希望接收的信号不改变时如果需要被减除 (subtracted) 的信号改变, 则所传输的信号的 SiBiDi (并发双向) IO 单元 “减除” (subtraction) 是更成功的。这可以通过按适当的量 (时钟周期的某个分数) 延迟传输而实现。但是在一端延迟传输意味着数据将延迟到达另一端。于是另一端的电路将只好重新调整它的传输延迟, 以便其自己的 “减除” 最优。因此需要寻找一对延迟设置, 每个对应于连线两端的每端的电路, 以便 “减除” 对两端都给出同样好的质量。

[0098] 此外, 为了实现这一点, 这两端需要交换关于对应于另一端上的传输延迟的每个选择的本地减除 (subtraction) 的质量的信息。但是这个信息不能用正在被优化的相同信令传输技术来交换。如果某延迟设置不好, 它会破坏所发送的描述它如何不好的数据。本发明描述一种为最佳的双位 SiBiDi 信令自动地调节传输延迟的硬件机构。

[0099] 图 11 表示的第一实施例中, 两个差分数据线连接一对相同节点 1、2, 并且其中每个节点都有一个唯一的 ID, 每个节点与一个 2 位发送器 CPU 和一个 2 位接收 CPU 一起操作。

[0100] 该方法利用 “安全通信” 设置阶段来传送每个传输延迟集合的结果。通过使用为 IO 单元的单向设置 (其中仅仅为了使噪声最小而在一个方向上传输) 并且只使用该单元的一个 1 位部分, 将 2 位发送器 / 捕获单元用于安全通信。

[0101] 图 11 表示一对结点之间的电连接, 并显示了两个差分数据线 (每个由两个连线组成能实现差分信令)。箭头表示在安全通信设置阶段的单向信号的方向。否则, 在正常通信期间的电连接是双向的。

[0102] Sel A、B 框是 MUX, 其中 Sel = 0 选择从 2 位发送器单元到 2 位捕获器单元的上路径, Sel = 1 选择下路径。

[0103] 1) 将 READY (准备好) 寄存器 (未予示出) 设定为 0。

[0104] 1) IO 单元 A = 单向发送器方式

[0105] IO 单元 B = 单向接收器方式

[0106] Sel A = 1 (选择上路径)

[0107] Sel B = 1 (选择下路径)

[0108] 2) 设定发送器延迟方式为零延迟。

[0109] 3) 开始训练, 上文关于寻找和检测好数据捕获眼时已经作了描述。

[0110] 4) 在好的数据捕获眼中保存安全单向设置阶段通信的眼参数

[0111] 5) 发送 ID 的第一位。在这个实施例中, 节点 1 和 2 的每一个的独有 Ids 确定哪一个节点是主节点, 哪一个节点是从节点, 较高的 ID 自动是主节点。

- [0112] 等到接收到另一个节点的第一 ID 位。
- [0113] 比较第一 ID 位。
- [0114] 如果相等则重复。
- [0115] 如果本地的小于邻居的则设定 PRIORITY(优先级) = 0
- [0116] 如果本地的大于邻居的则设定 PRIORITY(优先级) = 1
- [0117] 6) IO 单元 A = SiBiDi 方式
- [0118] IO 单元 B = SiBiDi 方式
- [0119] Sel A = 0
- [0120] Sel B = 0
- [0121] 7) 发送器延迟方式 = 0
- [0122] 开始训练
- [0123] 寻找眼
- [0124] 保存本地眼参数
- [0125] 8) 至如步骤 1 的安全方式
- [0126] 9) 发送你的本地眼参数至另一个节点。
- [0127] 接收另一个节点的眼参数。
- [0128] 10) 比较参数并保存最小的参数
- [0129] 11) 比较当前最小的眼参数与以前的最小眼参数并将两者中的最大参数与该最大参数的本地发送器延迟方式一起保存。 这个方式被视为最佳方式并被记为 OPT_SENDER_MODE(最佳发送器方式)。
- [0130] 在这个第一实施例中, 假设一个 8 抽头的延迟线, 这样每个节点在 8 个步骤中就可能有范围从零延迟到最大延迟的 8 个不同延迟, 因此可能的组合数是 $8 \times 8 = 64$ 。换言之, 对应一个节点处的 8 个延迟的每一个, 在第二个节点上有 8 个可能的延迟。因此必须测试 64 种可能的组合来选择最佳组合。步骤 12 一次一个地简单循环通过所有 64 个组合。
- [0131] 返回到步骤 6), 用以下的发送器延迟方式序列重复总共 64 次:
- [0132] 如果 PRIORITY = 0 则邻居首先改变方式。本地序列是:
- [0133] 0, 8 步
- [0134] 1, 8 步
- [0135] 2, 8 步
- [0136] 3, 8 步
- [0137] 4, 8 步
- [0138] 5, 8 步
- [0139] 6, 8 步
- [0140] 7, 8 步
- [0141] 如果 PRIORITY = 1, 则本地序列是:
- [0142] 0, 1 步
- [0143] 1, 1 步
- [0144] 2, 1 步
- [0145] 3, 1 步

[0146] 4, 1 步

[0147] 5, 1 步

[0148] 6, 1 步

[0149] 7, 1 步

[0150] 重复 8 次。

[0151] 12) 如 6) 中那样转至 SiBiDi, 其中发送器延迟方式 = OPT_SENDER_MODE(最佳发送器方式)。

[0152] 13) 设定 READY 寄存器至 1 以表示系统被优化并准备好正常的 SiBiDi 通信。

[0153] 如果在任何步骤有故障, 以至达不到步骤 14, 则该节点已经发生故障。可以通过 READY 寄存器的内容识别故障节点。

[0154] 图 12 表示一个在两个节点 1、2 之间的 SiBiDi 电通信的第二实施例, 其与第一实施例的一个区别在于其中一个差分数据线连接这些节点。第二个区别是该实施例对 1 位数据操作, 而不是像图 11 的第一实施例中那样对 2 位数据操作。另一个区别是, 在本实施例中, 主节点和从节点是由控制软件指定的, 而不是根据所分配的 Ids 指定的。本实施例也假设一个 16 抽头的延迟线而不是如图 11 中那样的 8 抽头延迟线。

[0155] 概述:

[0156] 初始对准过程 (IAP) 是一个步骤序列, 每个端口的每个同步信号由其确定最佳传输延迟线设置 (为其级外 (outstage))。美国临时专利申请序列号 60/271124 中描述了一种具有按三位晶格连接的 $32*32*64$ 个节点的、每个节点连接到 6 个相邻节点的大规模并行超级计算机。每个节点有 6 个端口, 每个端口 20 个同步信号, 使得超级计算机的一个节点处的一个节点计算机芯片上的所有 120 个同步信号 (6 个端口 * 20 个同步信号) 都能独立地执行个别训练。所有训练可以并行地发生, 或者只是一次一个。(都在软件的控制下)。训练是同时在 SiBiDi 链路的两个方向上进行的, 这允许必要的 ISI (符号干涉) 和近端噪声 (与环境噪声)。参看图 12, IAP 序列的高级流程是:

[0157] 1. 软件动作: 通过写每个节点计算机芯片的 IAP 控制寄存器而把同步链路的一端标识为“主”, 另一端标识为“从”。

[0158] 2. 硬件动作: 主端 (图 12 中的 A 端) 与从端 (图 12 中的 B 端) 通信, 以开始训练。这是有些复杂的, 因为必须在一个链路在被完全训练之前跨链路传送信息。(见下文的“跨未训练链路的通信”)

[0159] 3. 硬件动作: 一个链路的每端都有一个遍历 (runs through) 所有可能的延迟线设置并比较结果以发现最佳延迟线设置的状态机 (如图 13A 和 13B 中所示)。改变一端上的延迟线设置影响两端上的眼, 因此系统需要遍历所有 $16*16$ 个组合。(注: 级外 (数据发送) 延迟线有 16 个设置, 如前文所解释的那样)。对于每个延迟线训练循环, 级内 (数据捕获) 宏从另一端接收一个伪随机数据流并寻求找到眼和表示眼大小信息供分析。

[0160] 4. 软件动作: 读 IAP 状态寄存器以确定训练的成功 / 失败。可以通过软件可访问的其它链路特定的状态寄存器读取精确的延迟线设置和眼大小范围。

[0161] 链路训练序列:

[0162] 图 13A 和 13B 的状态机表示在同步 Si-Bi-Di 连接的训练中由节点计算芯片采取的步骤。链路的每端使用下列寄存器:

[0163] **DTR**-延迟抽头寄存器-控制级外中的延迟线。(有效设置是 0-15)。在 IAP 序列期间使用两个额外的“工作”拷贝：**Mst-DTR** 和 **Slv-DTR**。

[0164] **LBDTR**-本地最佳延迟抽头寄存器-在训练期间保存对应于最佳要看见的眼大小的 DTR 值。在训练结束时，**LBDTR** 的内容被永久地装载到 **DTR** 中。

[0165] **MBESR**-互相最佳眼大小寄存器-在训练期间保存最佳要看见的眼大小(根据训练中给定步骤 A 端和 B 端眼大小的最小者)。

[0166] 在链路训练序列期间，噪声生成器宏能被启动作为人工添加更多噪声的一种方式以模拟一个非常嘈杂的环境以及保证与眼的大小有关的更多的纵向电压范围。软件通过在 IAP 控制寄存器中写“开始”位开始 IAP 序列，并确定芯片为主(端 A)或从(端 B)。

[0167] 跨未训练链路的通信：

[0168] 在链路被完全训练之前有必要在链路的两端之间进行通信。为确保最可靠的数据传输，采用下列过程：

[0169] 1) 一次只在一个方向发送数据。

[0170] 2) 以较低速率发送数据。可采用 1 : 8 的比率(即为 8 位时间保存“1”或“0”)。

[0171] 在训练之前，链路的两端没有可预测的相位(phase)关系。因此，如果一端发送“110011”，并且如果样本点与交换数据对齐，则数据可能被接收为“111011”或“100011”等等。传输速率要低的足以检测到连续样本上的稳定数据，并且不与位流内的在 0->1 或 1->0 变换期间发生的错误采样相混淆。

[0172] 在“主”和“从”之间发送的“命令”有一长串的前导“1”和后继的 8 个 0。命令看起来是：……许多 1，8 个 0，命令的第一位的 8 位时间，命令的第二位的 8 位时间，……命令的最后一位的 8 位时间。接收端检测到 1->0 转变并估计 8 位时间窗的中间。(实际上，这可以是 8 位时间窗的第 3、第 4、或第 5 位；它们全部都应当稳定且有效)。此后，对每一个第 8 位采样，以解译命令/信息。

[0173] 参看图 13A 和 13B，所有命令都以大写字母指示，图 13B 中的框 0、5、6、9、12-14 中的“相同”二字表示它们分别与图 13A 中对应的框的意思相同。

[0174] 参看图 13A，这是主端，在阶段(0)等待一个开始软件命令，然后复位寄存器 **DTRs**、**LBDTR**、**MBDTR**。在阶段(1)，主端等待接收来自从端的 **BEGIN**(开始)命令，如果不，则向从端发送一个 **BEGIN** 命令，并等待来自从端的 **BEGIN** 答复。如果不，则主端等待并向从端再发送 **BEGIN**。如果主端接收到来自另一个主端的 **BEGIN**，则中断。

[0175] 在阶段(4)，如果是，主端发送一个 **TRAIN**(训练)命令，指示主端准备开始同步，然后暂停。

[0176] 在阶段(5)，主端传送一个随机数据位流，以便能捕获眼。

[0177] 在阶段(6)，主端等待眼的捕获并评估关于每个眼的信息，诸如眼大小。

[0178] 在阶段(7)，主端等待接收关于眼大小的数据，如果不，等待(例如 1 微秒)并发送关于眼大小的数据，再次等待接收关于眼大小的数据。

[0179] 接收到时，主端更新 **MBDTR** 和 **LBDTR** 寄存器并递增 **Mst-DTR**，如果重叠(计数器溢出)则递增 **Slv-DTR** 寄存器，对所有 256 个组合重复阶段(4)-(9)。

[0180] 如果是，在阶段 (10)，主端发送一个 END (结束) 命令，以结束眼训练。

[0181] 在阶段 (11)，主端等待接收来自从端的 END 命令。

[0182] 如果是，在阶段 (12)，主端用 LBDTR 装载 DTR，然后复位级内，这是一个设定 (set-up) 节点。

[0183] 在阶段 (13)，最优眼参数被用来传送随机数据。

[0184] 阶段 (12) 和 (13) 用最优眼参数来传送随机数据，然后最优眼参数被重新评估，如果成功则被锁定就位。

[0185] 在阶段 (14)，主端断言一个“复位假信号” (Reset Glitch) 信号，以复位并重新评估数据捕获，对照最小眼大小数据检查眼大小，以及更新 IAP 状态寄存器。

[0186] 图 13B 的从端的操作从上述对主端的描述应当是明显的。

[0187] 尽管本文详细描述了本发明的用于高速信令的数据捕获技术的若干实施例及变体，显然，对于本领域的那些熟练人员来说本发明的内容和教导将揭示许多可替代的设计。

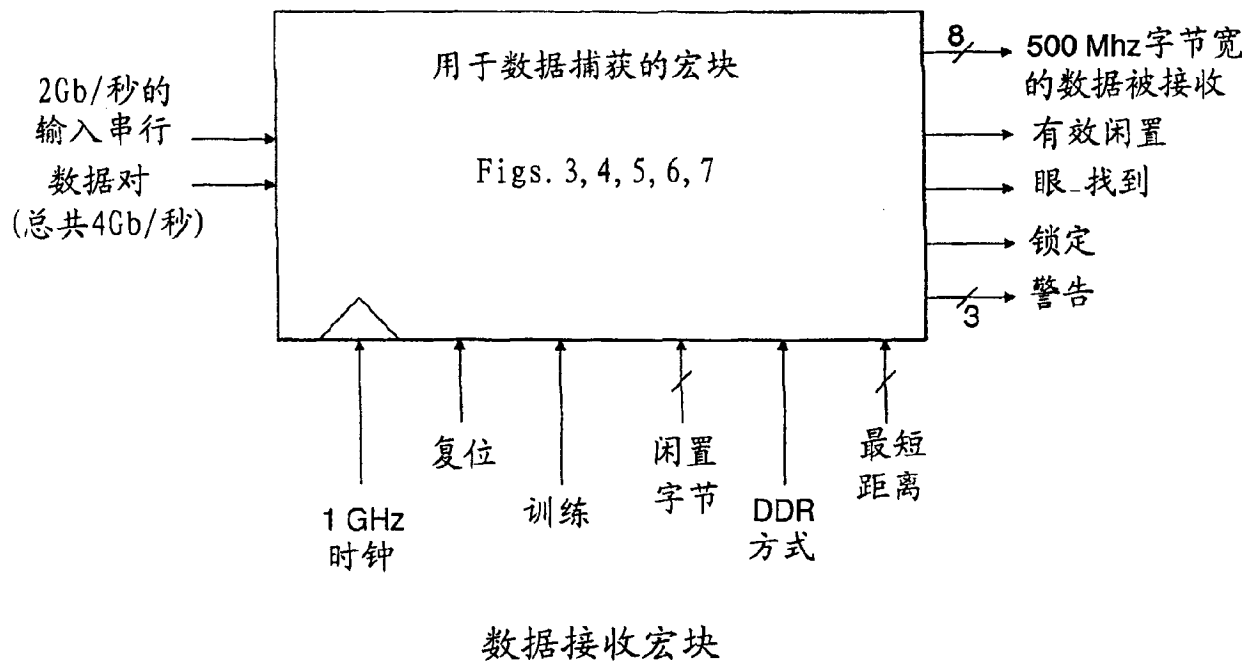


图 1

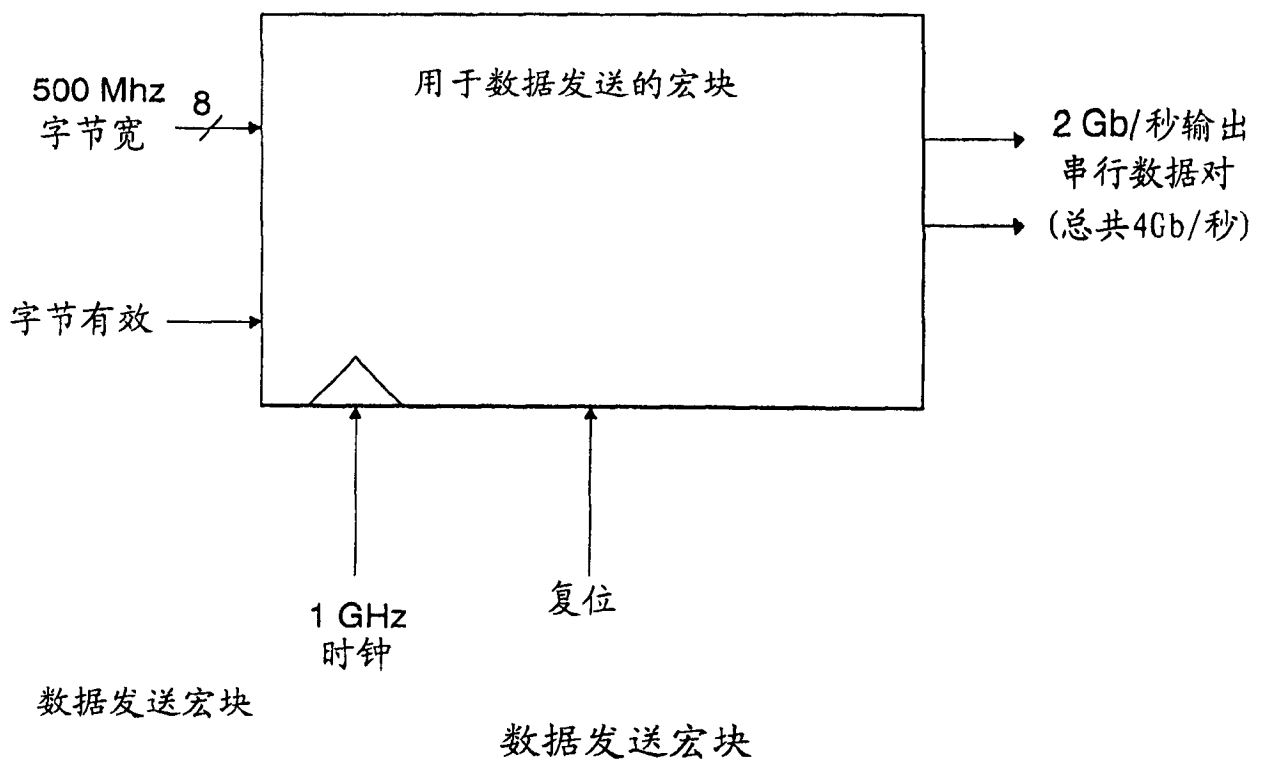


图 2

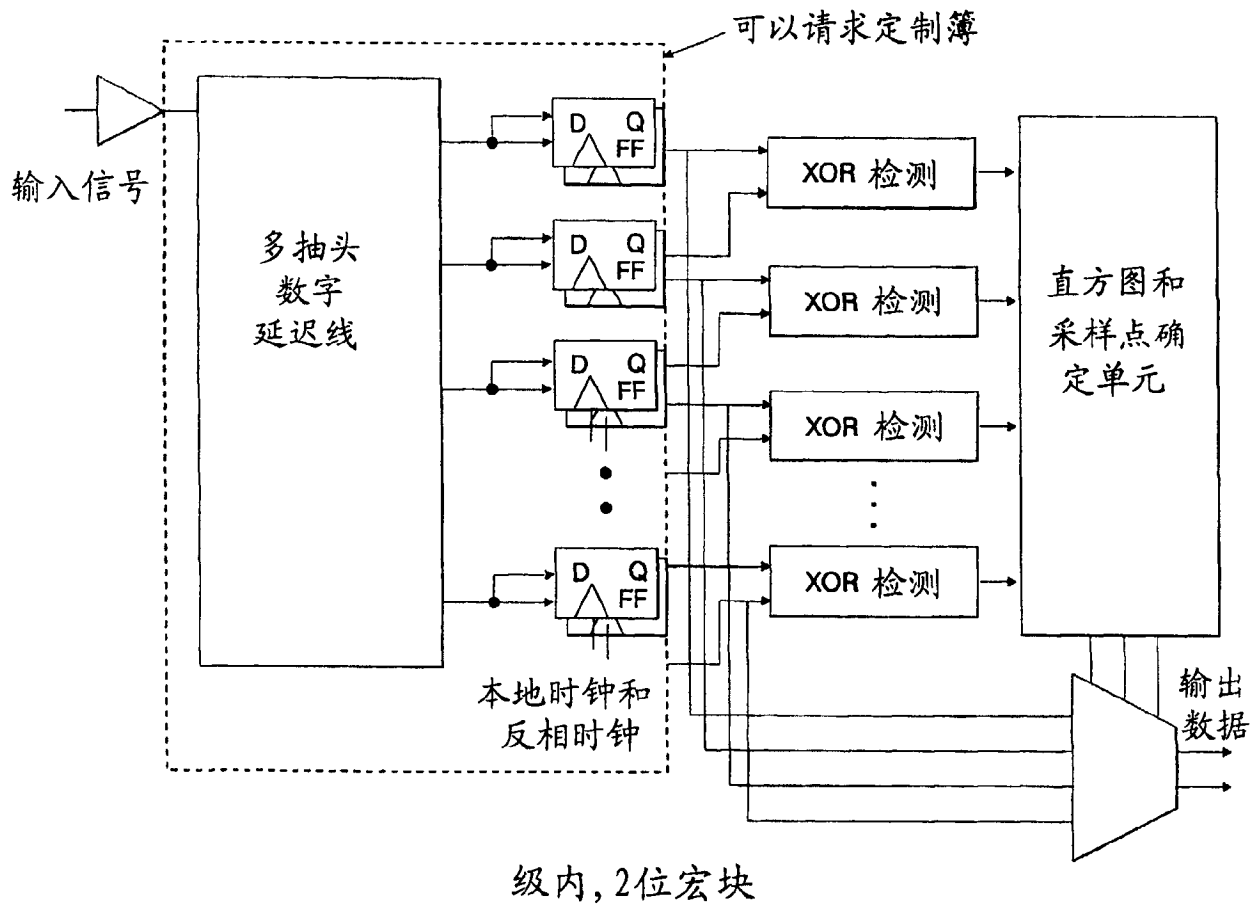


图 3

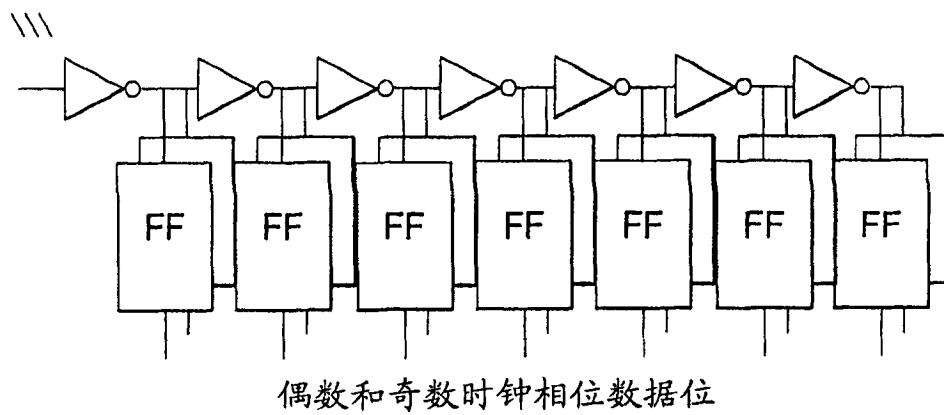
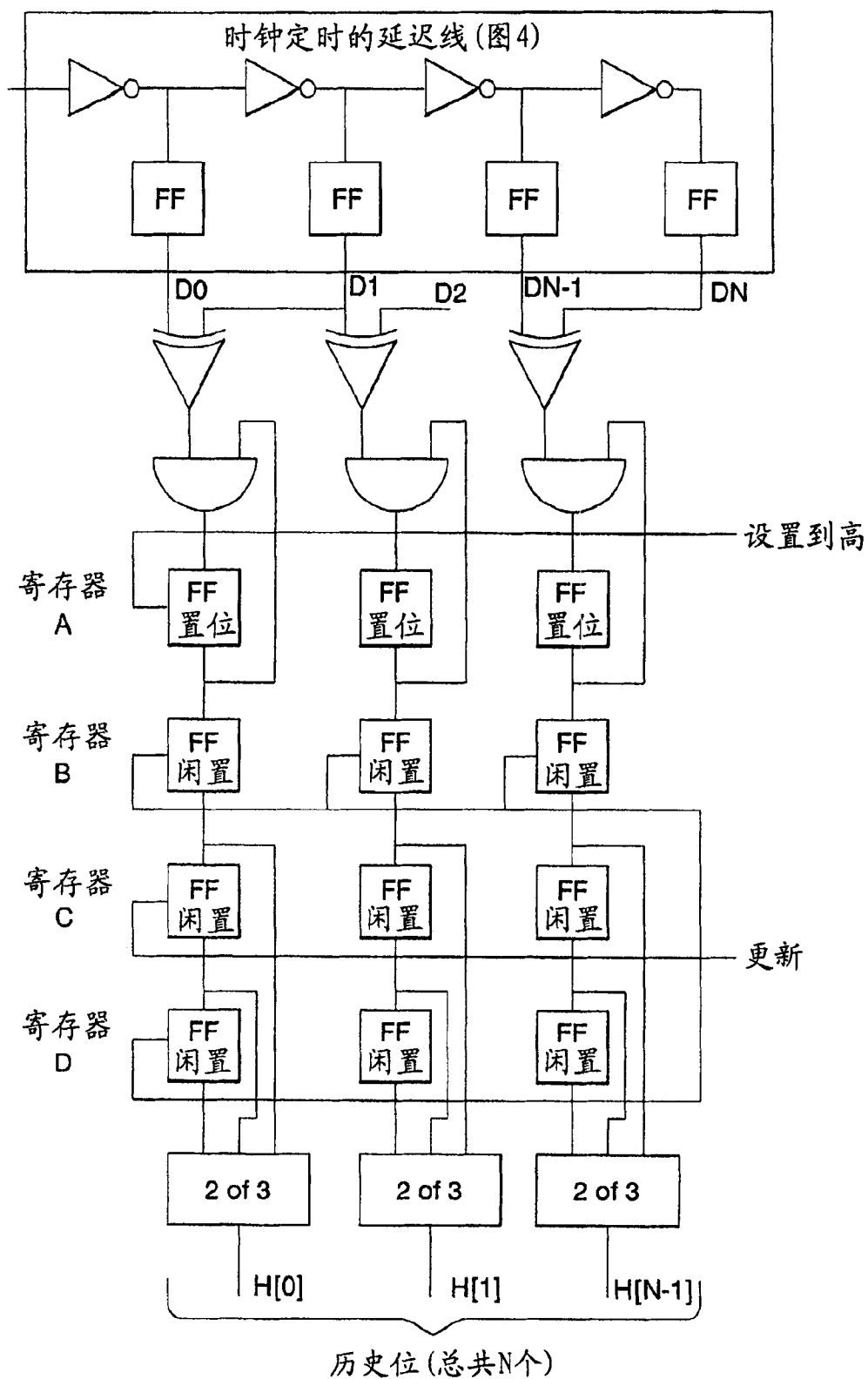
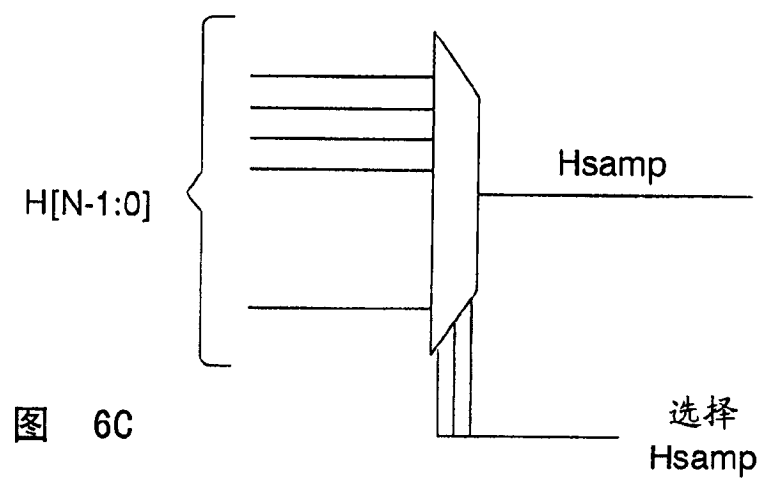
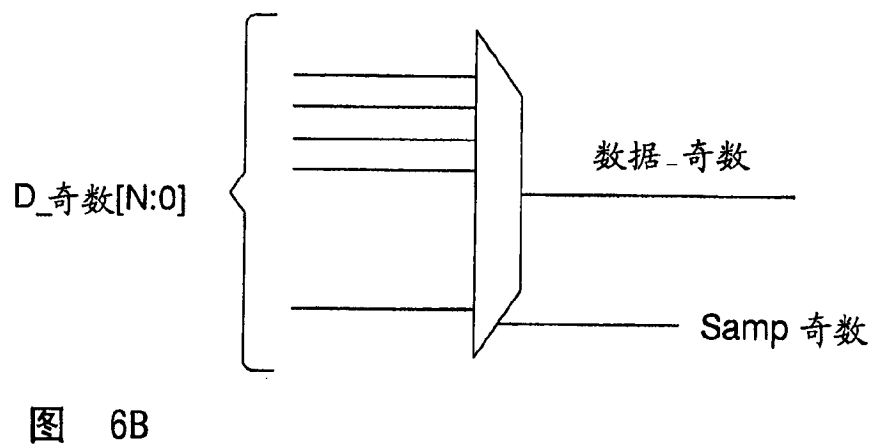
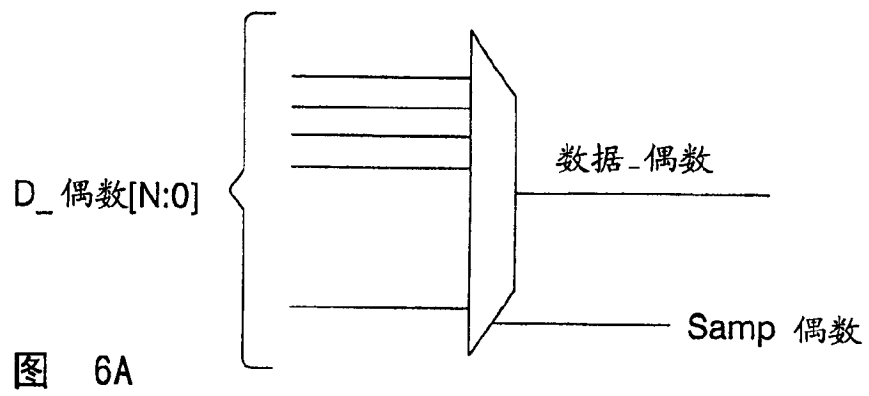


图 4

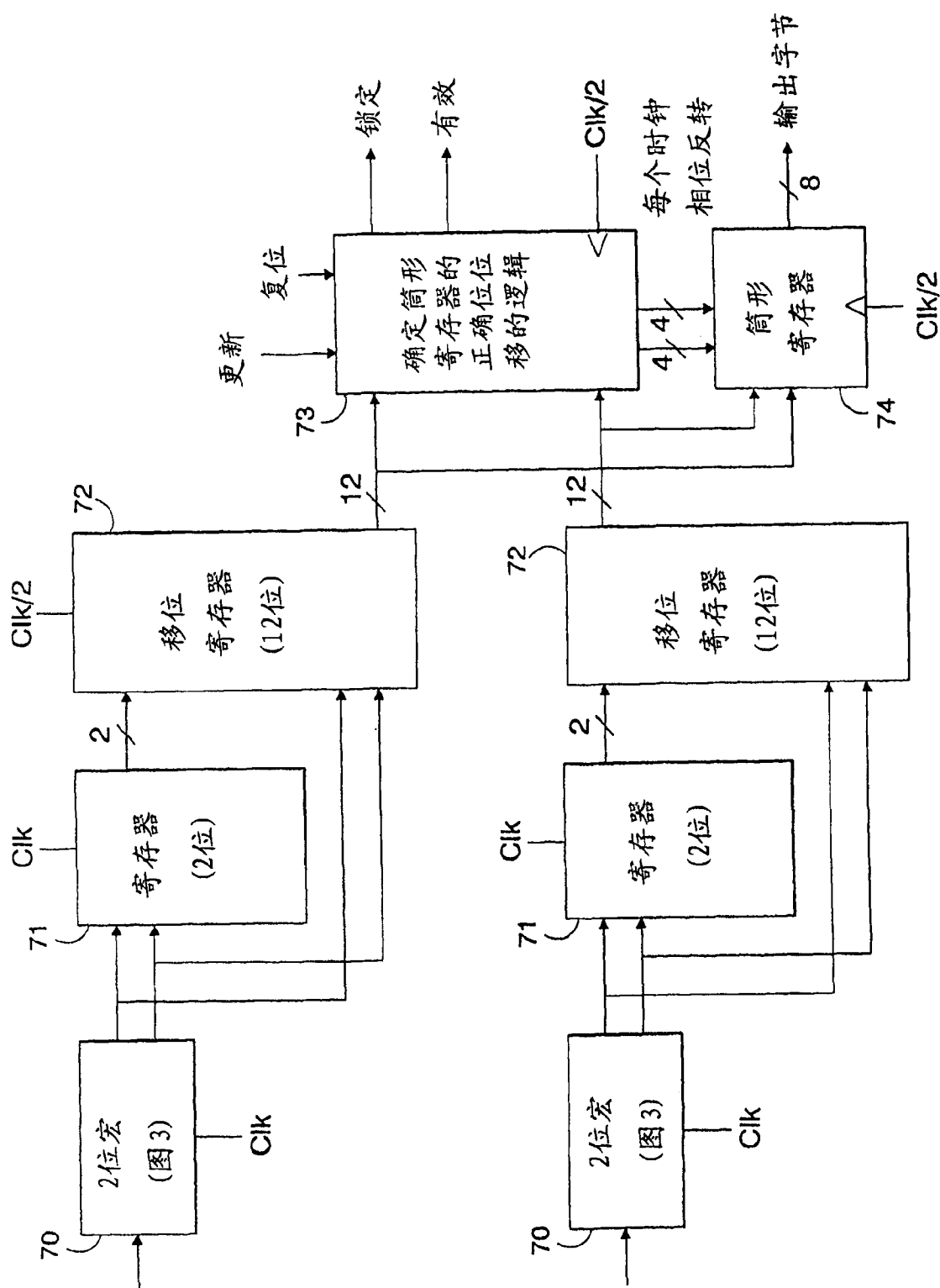


历史逻辑

图 5

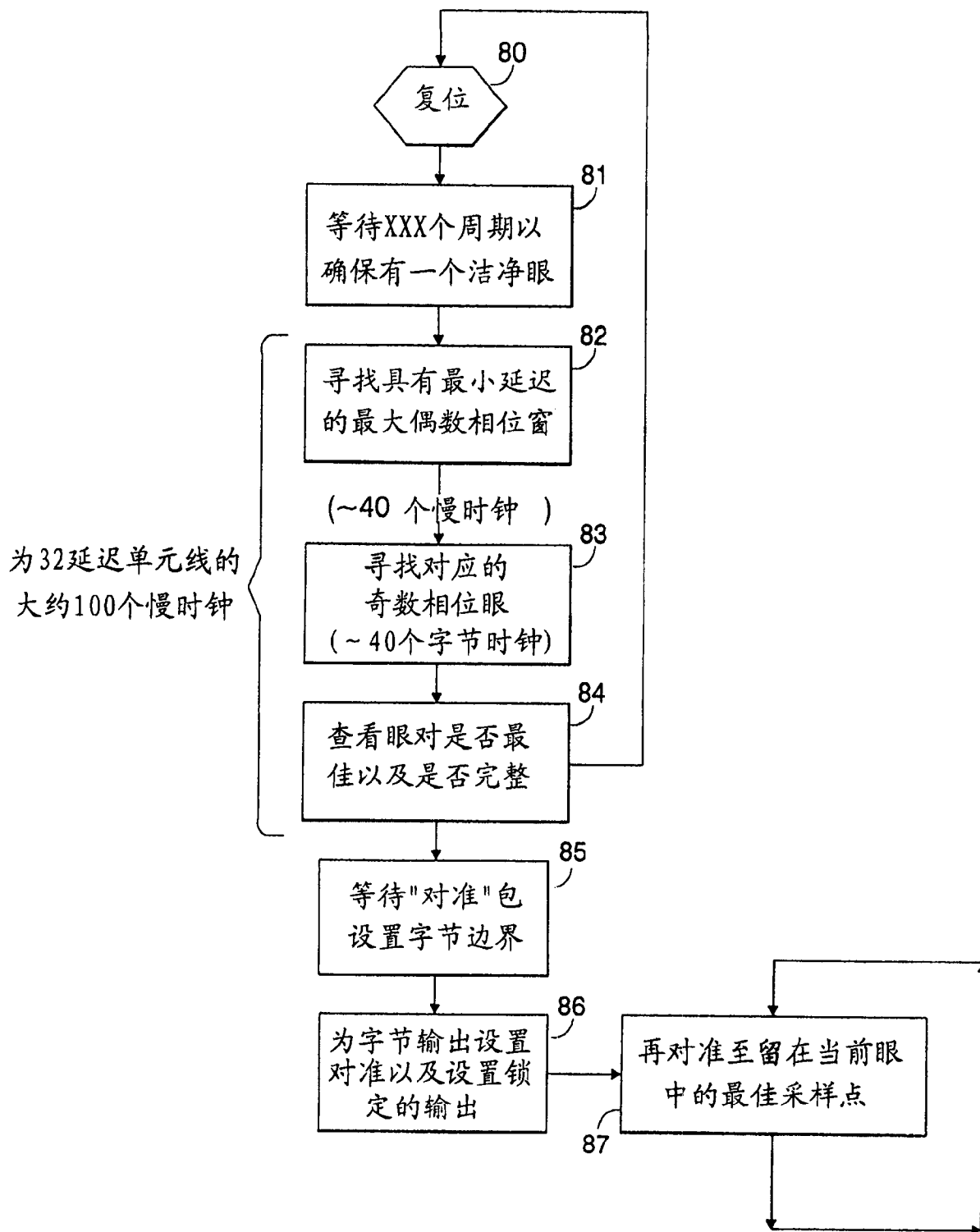


数据和历史样本多路转换器



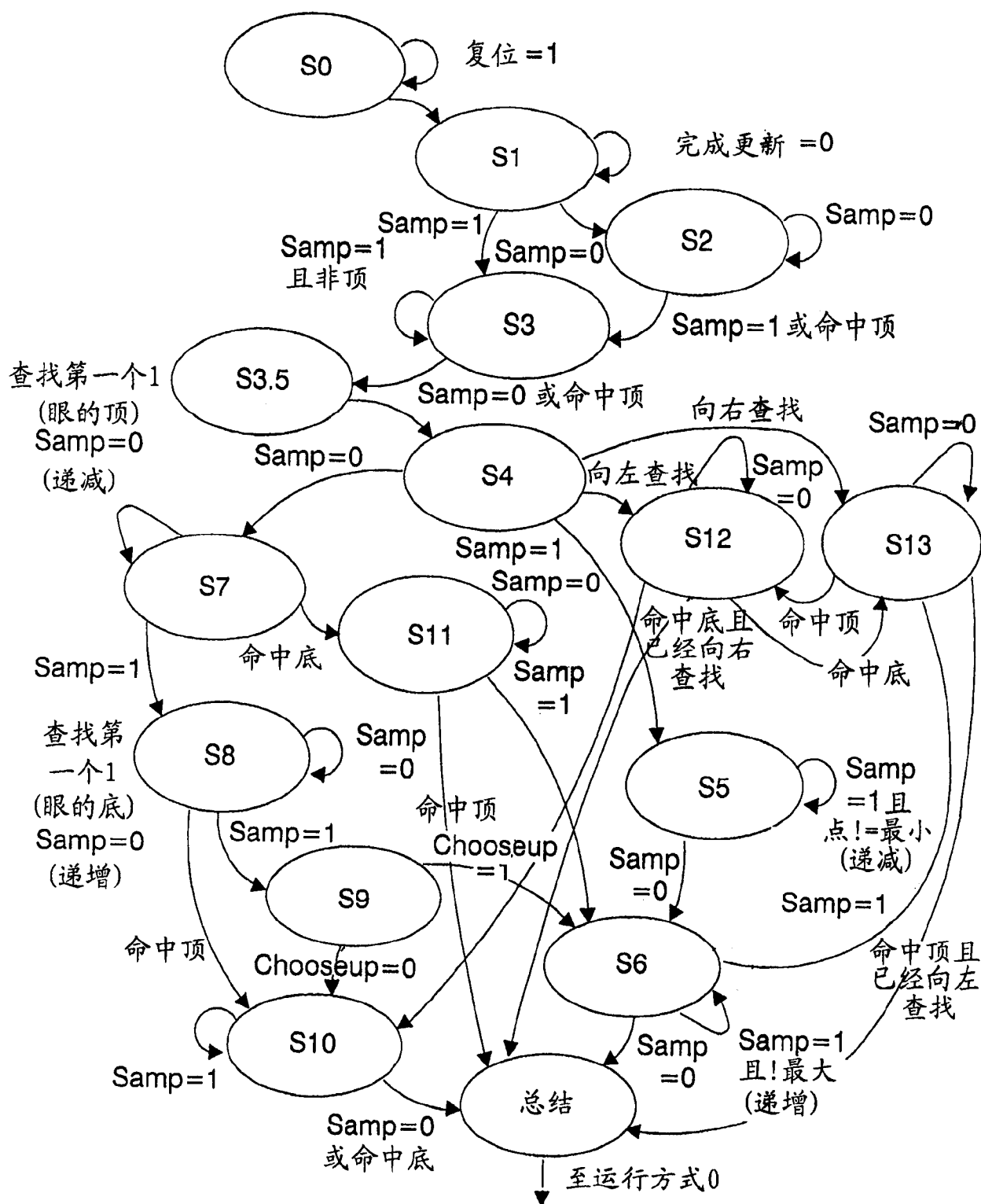
串行位组合和字节对准逻辑

图 7



眼检测流程图

图 8



两位宏状态图二之一

图 9

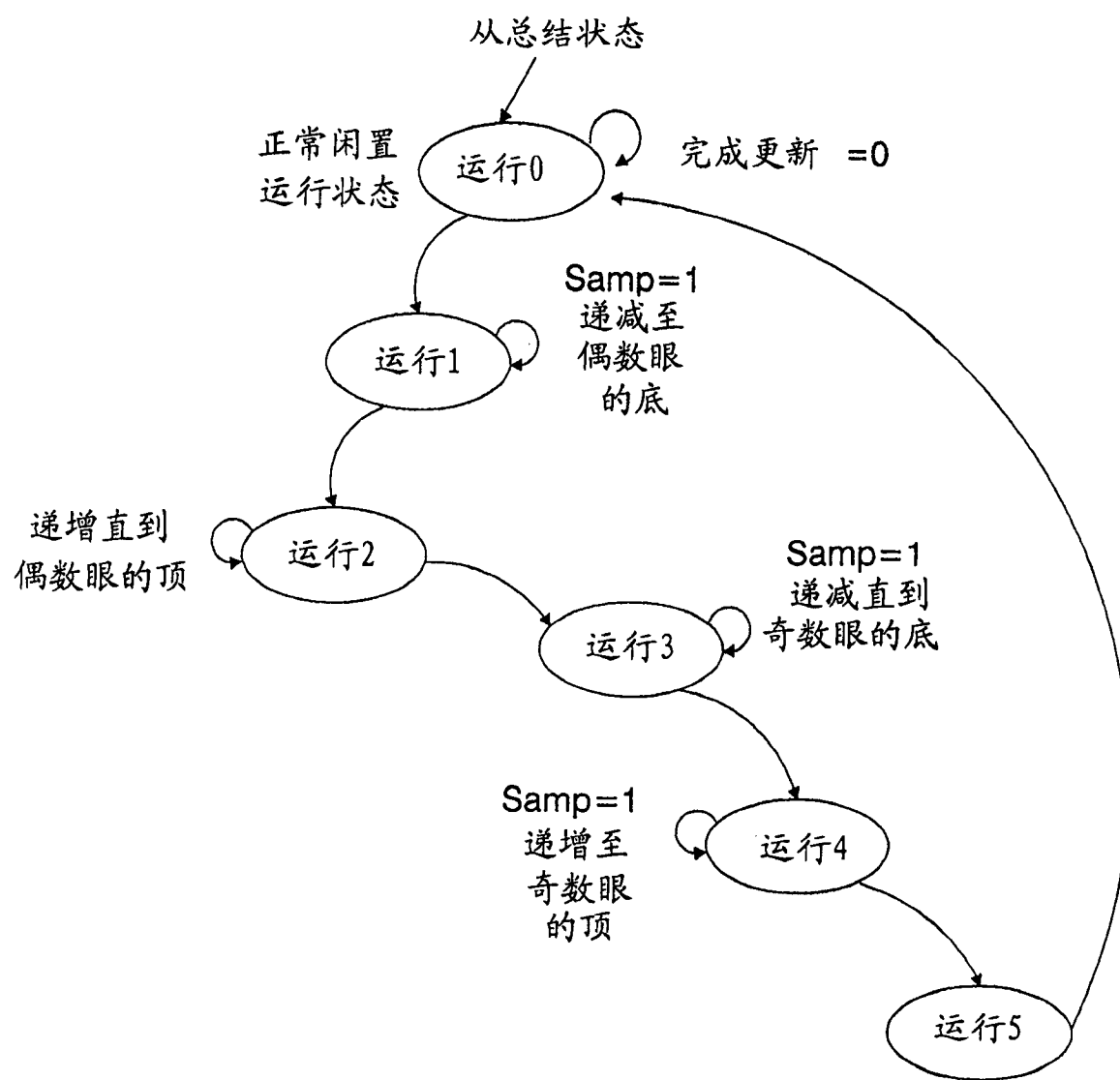


图 10

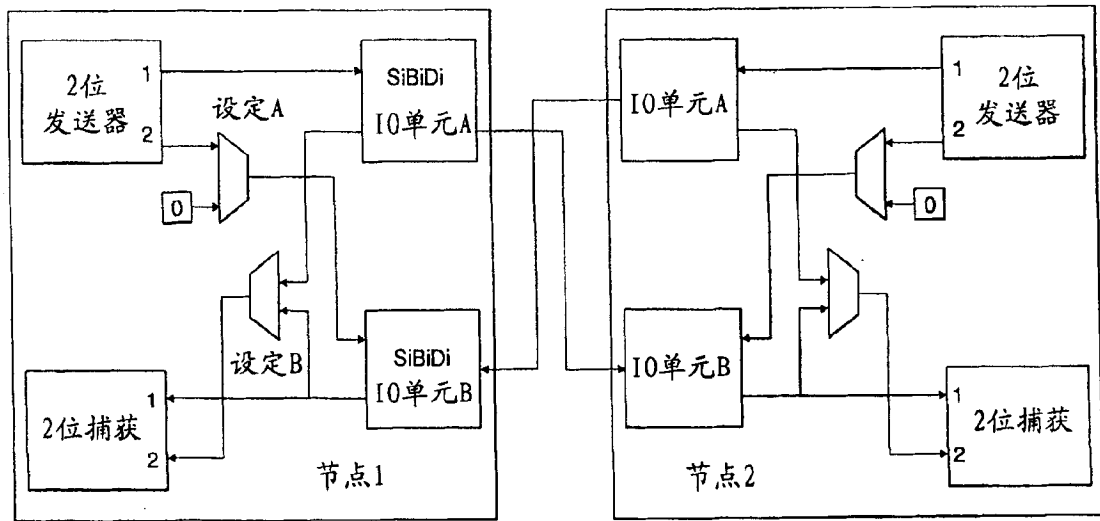


图 11

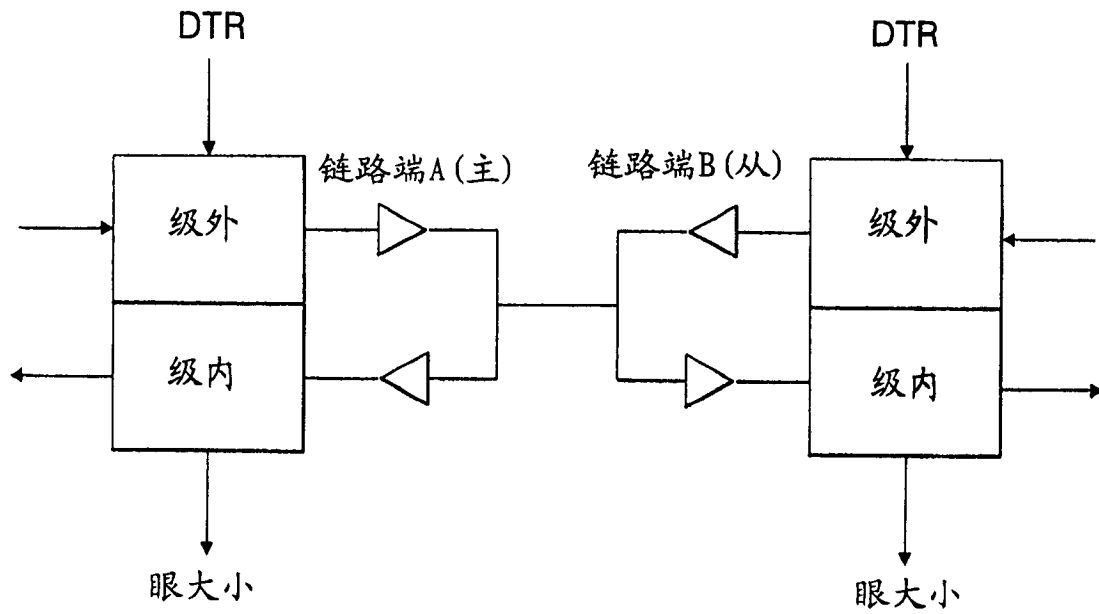


图 12

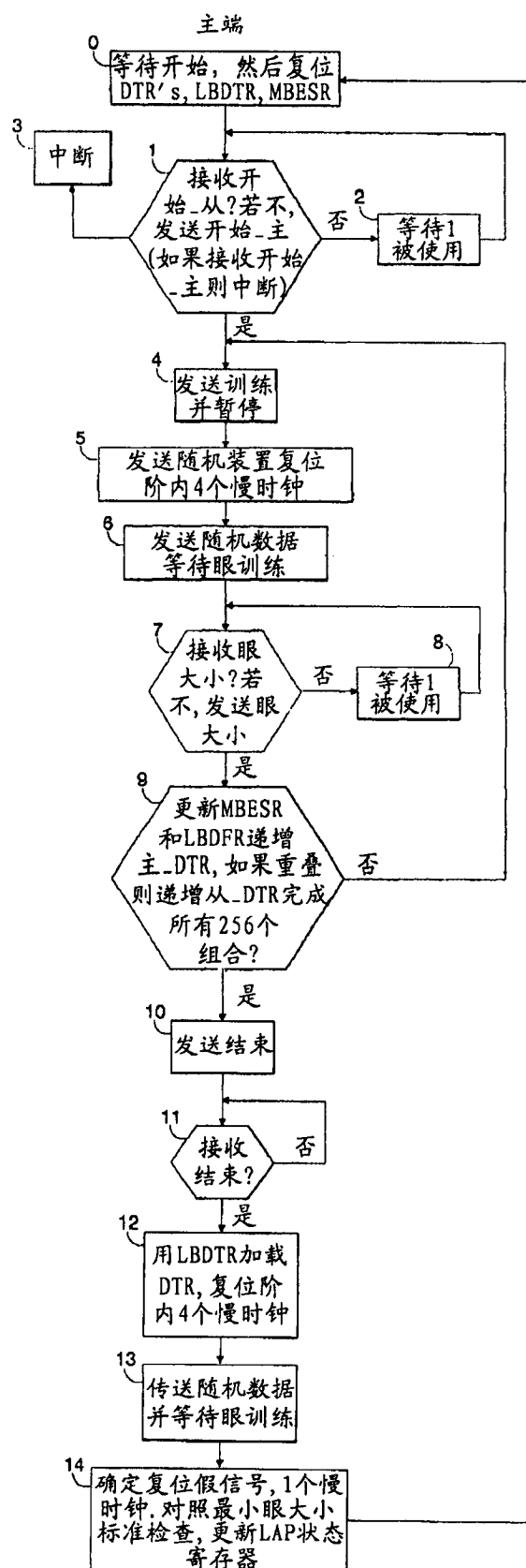


图 13A

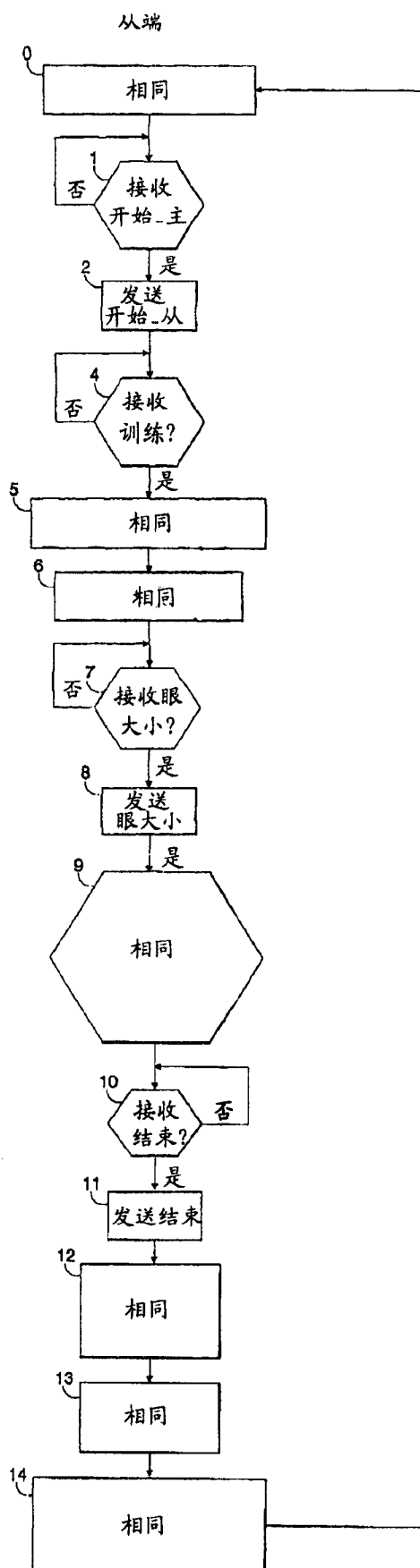


图 13B