

[19] 中华人民共和国国家知识产权局

[51] Int. Cl⁷

G06F 15/16

G06F 13/00



[12] 发 明 专 利 说 明 书

[21] ZL 专利号 00811557.5

[45] 授权公告日 2005 年 6 月 1 日

[11] 授权公告号 CN 1204509C

[22] 申请日 2000.8.24 [21] 申请号 00811557.5

〔30〕 優先権

[32] 1999. 8.27 [33] US [31] 09/384,744

[86] 国际申请 PCT/US2000/020794 2000.8.24

[87] 国际公布 WO2001/016779 英 2001.3.8

[85] 进入国家阶段日期 2002.2.9

[71] 专利权人 国际商业机器公司

地址 美国纽约

[72] 发明人 布赖恩·M·巴斯

琼·L·卡尔维格纳克

马科·C·赫德斯

皮郁施·C·帕特尔

朱安·G·莱维拉

迈克尔·S·西戈尔

法布里斯·J·沃普兰肯

审查员 于 平

[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

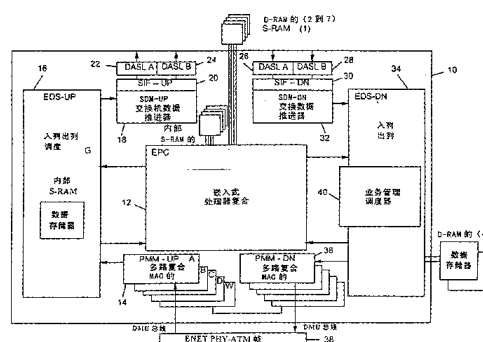
代理人 付建军

权利要求书 4 页 说明书 46 页 附图 17 页

[54] 发明名称 通信网络装置

[57] 摘要

一种网络交换装置(10)、用于该装置的各个部件、以及操作该装置的方法，该方法通过多个存储器单元和装在半导体基片(10)上的多个接口处理器协同工作来提对数据流的处理并增强灵活性。上述存储单元和接口处理器共同构成网络处理器(10)，该网络处理器可与其它单元协同执行有关指令，以控制网络中数据的流动。每个接口处理器都可访问多个存储器单元，每个存储器单元都可以对多个接口处理器的请求作出响应。



LSN 1008-4274

1. 一种装置, 包括:

一个半导体基片;

多个装在所述基片上的接口处理器, 所述处理器的数目至少为 5 个;
装在所述基片上, 并存储可由所述接口处理器访问的指令的内部指令存储器;

装在所述基片上, 并存储可由所述接口处理器访问的经过所述装置的数据的内部数据存储器;

所述内部数据存储器带有多个存储器单元, 所述存储器单元的数目与所述接口处理器的数目相关联;

插在所述接口处理器中的每一个和存储器单元之间的请求控制器, 可控制从所述接口处理器到所述存储器单元的数据请求的流程;

插在所述存储器单元中的每一个与接口处理器之间的存储器判优器, 可控制接口处理器对所述存储器单元的访问;

每一个所述接口处理器都可访问多个所述存储器单元, 每一个所述存储器单元都可对来自多个所述接口处理器的请求作出响应; 以及

所述基片上有多个输入/输出端口;

所述输入/输出端口中至少有一个将所述内部数据存储器与外部数据存储器相连接; 且

至少有另外两个输入/输出端口在所述接口处理器的引导下与外部网络交换经过装置的数据。

2. 一个装置, 包括:

一个半导体基片;

至少 5 个装在所述基片上的接口处理器;

装在所述基片上, 并存储可由所述接口处理器访问的指令的内部指令存储器;

装在所述基片上, 并存储可由所述接口处理器访问的经过所述装置的数据的内部数据存储器; 所述内部数据存储器至少有 5 个存储器单元;

插在所述接口处理器中的每一个与所述存储器单元之间的请求控制器，可控制从所述接口处理器的相应一个到所述存储器单元的数据请求的流程；

插在所述存储器单元中的每一个与所述接口处理器之间的存储器判优器，可控制所述接口处理器对相应的所述存储器单元的访问；

每个接口处理器都可访问多个所述存储器单元，每个所述存储器单元都可以对多个所述接口处理器的请求作出响应；

所述请求控制器与所述存储器判优器协同工作来执行一套访问规则，从而控制所述接口处理器对所述存储器单元的访问；以及

装在所述基片上的多个输入/输出端口；

所述输入/输出端口中至少有一个是用以将所述内部数据存储器与外部数据存储器相连接；

至少有另外两个输入/输出端口在所述接口处理器的引导下与外部网络交换经过装置的数据。

3. 一种装置，包括：

多个存储器，所述多个存储器具有不同的类型；

多个存储器判优器，其中，各个存储器判优器连接到多个存储器之一，并提供对所述多个存储器之一的访问；

多个请求控制器，每个都能生成请求信号以访问至少一个存储器；

多条传输总线，每条总线都将一个请求控制器与所选的若干或全部存储器判优器相连；以及

多个处理器，每个都连接到多个请求控制器之一。

4. 根据权利要求3的装置，其特征在于，不同类型的存储器包括静态随机存取存储器和动态随机存取存储器。

5. 根据权利要求3的装置，其特征在于，不同类型的存储器存储不同类型的数据结构。

6. 根据权利要求4的装置，其特征在于，动态随机存取存储器被分成四个区，每个区都能独立于其它区进行编址。

7. 根据权利要求6的装置，其特征在于，动态随机存取存储器中的

位置可以根据存储器 ID+区编号+偏移量值来确定。

8. 一种装置，包括：

一个存储器系统，包括静态随机存取存储器模块和动态随机存取存储器模块；所述动态随机存取存储器模块对配置信号作出响应，配置信号将动态随机存取存储器模块置于第一种操作模式，该模式在单个访问周期内在所述动态随机存取存储器中的各个区上执行多个连续的读操作或写操作，或者置于第二种操作模式，该模式在一个访问周期内在所述动态随机存取存储器中选定的区上执行读操作或写操作；

多个处理器，设定为生成配置信号、存储器请求信号；以及

一个网络，将多个处理器连接到存储器系统，以便多个处理器中的每个处理器都能访问所述存储器系统中静态随机存取存储器模块和动态随机存取存储器模块。

9. 根据权利要求 8 的装置，其特征在于，网络为判优类型。

10. 根据权利要求 8 的装置，其特征在于，网络包括多个判优器，多个判优器中所选择出来的那些判优器与静态随机存取存储器模块中选择出来的那些静态存取存储器模块相连，多个判优器中选择出来的那些判优器与多个动态随机存取存储器中选择出来的那些动态随机存取存储器模块相连；

多个请求控制器，其中，多个请求控制器中选择出来的那些请求控制器与多个处理器中选择出来的那些处理器相连；以及

多条总线，其中，多条总线中的每条总线将多个请求控制器中的每个与多个判优器相连。

11. 一种装置，包括：

一个半导体基片；

装在所述基片上的多个接口处理器；

存储指令的存储器，这些指令可由装在所述半导体基片上的所述接口处理器访问；

多个装在所述半导体基片上的基片上静态随机存取存储器；

多个装在半导体基片外的基片外静态随机存取存储器；

多个装在半导体基片外的动态随机存取存储器;

多个请求控制器, 其中, 各个请求控制器与多个接口处理器之一相连;

多个存储器判优器, 将其分成三组, 其中, 第一组中的每一个与基片上静态随机存取存储器相连, 第二组中的每一个与基片外静态随机存取存储器相连, 第三组中的每一个与动态随机存取存储器相连; 以及

多条传输总线, 每条总线将多个请求控制器中的每一个与多个判优器相连, 以便每个处理器都能访问静态随机存取存储器和动态随机存取存储器。

12. 根据权利要求 11 的装置, 进一步包括多个装在所述基片上的输入/输出端口, 其中, 所述输入/输出端口中的至少一个将基片上静态随机存取存储器与基片外静态随机存取存储器和动态随机存取存储器互连, 而其它所述输入/输出端口与外部网络交换数据。

13. 根据权利要求 11 的装置, 进一步包括存储经过所述装置的数据的基片上存储器。

14. 根据权利要求 11 的装置, 其特征在于, 存储器判优器根据来自处理器的命令, 将动态随机存取存储器配置为第一种操作模式, 该模式在单个访问周期内在所述动态随机存取存储器中的各个区上执行多个连续的读操作或写操作, 或配置为第二种操作模式, 该模式在一个访问周期内在所述动态随机存取存储器中选定的区上执行读操作或写操作。

15. 根据权利要求 11 的装置, 其特征在于, 存储器判优器响应来自处理器的单个命令, 从所述判优器与之相连的静态随机存取存储器或动态随机存取存储器读取一个地址, 在该地址的内容上添加一个值, 并将结果写入该地址。

通信网络装置

技术领域

本发明涉及一种通信网络装置，可用于将各种类型和功能的信息处理系统或计算机连接到该装置。本发明尤其涉及用以组装上述装置的可伸缩的交换装置和部件。本发明涉及一种经过改进的多功能接口设备以及这种设备与其它元件的组合，从而构成一台中速网络交换机。本发明还涉及操作这种装置的方法，从而改善网络交换机处理数据流的能力。

背景技术

在下文的说明中，我们假定读者对用于通信网络的网络数据通信、交换机和路由器已有一定的认识。我们尤其假定读者对网络体系结构的ISO模式较为熟悉，该模式是把网络操作分成若干层。一个典型的基于ISO模式的体系结构是从第1层（有时也用“L1”来表示）开始向外延伸，第1层是指物理通道或介质，信号就是从这里上传至第2层、第3层、第4层直至第7层，第7层为应用程序层，应用程序运行在与网络相连接的计算机系统中。在本文中，我们将以L1、L2等来表示网络体系结构中相应的各层。本公开还假定读者对网络通信中由位串组成的“包”和“帧”有基本的了解。

在当今的网络世界，带宽是一种极为重要的资源。由于因特网的发展和其它新的应用程序的不断出现，网络通信量与日俱增，从而为网络基础结构的容量带来巨大压力。为跟上要求，许多公司都在寻求更好的技术和方法，从而为通信量的增长和语音与数据的会聚提供支持和管理。

目前的网络通信量之所以迅猛增长，主要归因于因特网的普及、对远程信息访问的日益频繁、以及新应用程序的不断出现。单说因特网一项，

由于电子商务的飞速发展，已经使网络主干线路不堪重负。数据信息量不断增加并首次超过电话业务量，这其中，因特网起到了极为重要的作用。与此同时，对远程访问应用程序（包括电子邮件、数据库访问和文件传输）日益旺盛的需求，也在增大对网络的压力。

话音与数据的会聚将成为未来网络环境中的主角。目前，通过因特网协议(IP)网络的数据传输是不收费的。这是因为音频通信自然会寻求最低成本方式，话音必将与数据会聚在一起。在这种变化的市场中，诸如通过 IP 的话音传输(VoIP)、通过 ATM 的话音传输(VoATM)以及通过帧中继的话音传输(VoFR)等技术的性能价格比都使其能占有一席之地。不过，要使这些技术得以应用，业界还必须保证话音服务质量(QoS)，并确定如何对数据线上的话音传输进行收费。1996 年的《电信自由化法案》使这一环境进一步复杂化。该法规旨在实现精选出来的语音协议 ATM 与精选出来的数据协议 IP 之间的协作关系。

随着新产品的不断问世和新功能的不断出现，能否将其集成到原有系统中成了众多公司十分关心的问题。为保留对原有设备和软件的投资，很多公司急需这样的解决方案，即能够将新技术移入，同时又不妨碍原有设备的使用。

消除网络瓶颈仍是众多服务供应商的当务之急。这些瓶颈的根源往往是路由器。然而，通常的网络拥挤往往被错判为带宽问题，于是把寻求较高的宽带当作解决方案。目前，生产商正逐渐意识到这个问题。他们开始转向网络处理器技术，以期以有线速度更有效地管理带宽资源，并提供先进的数据服务、这些服务通常在路由器和网络应用服务器中提供。这些服务包括负载平衡、QoS、网关、防火墙、安全以及 Web 高速缓存。

对于用于远程访问的应用程序来说，性能、符合需要的带宽、安全性和身份验证是头等重要的方面。将 QoS 与 CoS 相结合的需求、集成话音处理、更为复杂的安全性解决方案，这些也将影响未来远程访问网络交换机的设计思路。另外，远程访问还必须能适应物理介质数目的不断增加，如 ISDN、T1、E1、OC-3 到 OC-48、电缆以及 xDSL 调制解调器等。

业界顾问为网络处理器（这里也称作“NP”）下的定义是：一种可编

程的通信集成电路，能执行以下一项或多项功能：

包分类——根据已知特征（如地址或协议）来标识包

包修改——对包进行修改，从而使其符合 IP、ATM 或其它协议（例如，更新 IP 首标中的实时激活字段）

队列/政策管理——反映包队列的设计策略

对于特定应用进行包出列和调度

包转发——通过交换结构发送和接收数据，并将包转发或路由到适当的地址。

尽管该定义对早期 NP 的基本特征给出了准确的说明，但 NP 的全面能力和效益的开发尚待实现。由于原来由软件处理的任务现在都可由硬件来执行，故网络处理器可以将带宽提高，并解决大量应用程序中所存在的问题。此外，NP 还对体系结构（如并行分布式处理与流水式处理设计）中的速度予以改善。上述功能可提高搜索引擎的效率、增加处理量，并对复杂任务进行快速处理。

有人预计，就象 CPU 之于 PC 机一样，网络处理器将成为网络中的基本网络结构单元。NP 能够提供的基本功能包括：实时处理、安全性、储存和转发、交换结构、IP 包处理和学习功能。NP 目标是 ISO 第 2 层至第 5 层，主要负责优化与网络相关的各项任务。

处理器模式的 NP 将多个通用处理器与专门的逻辑电路结合在一起。当前，供货商们都看好这种设计，它可以提供可伸缩的、灵活的解决方案，从而及时、经济地应对各种变化。处理器模式的 NP 可以在集成的较低层次上进行分布式处理，且具有较高的处理量、较好的灵活性和控制性。可编程性还可实现向新协议和新技术的方便迁移，而无需重新设计专用集成电路。通过处理器模式的 NP，NEV 可以从两个方面受益：一是降低了不可返还的工程成本，二是缩短了上市时间。

发明内容

本发明的目的之一是提供一个用于数据通信网络的可伸缩的交换机体系结构，它在改善传输数据处理速度的同时，还可以为各种潜在需求提供强大的支持。为实现上述目的，本发明提供了若干部件以及各个部

件的组合，这些部件使处理单元的数据处理量与以前相比大大降低。

本发明的另一个目的是提供一种接口设备或网络处理器（这两个术语可互换使用），其中包含集成在一个基片上的若干子部件，这些子部件共同动作，对第2层、第3层、第4层和第5层中的帧进行中等速度的交换。该接口设备可作为单一解决方案为工作组交换机提供基本功能，也可作为互联解决方案为工作组交换机提供高级功能，或是通过与交换结构装置的合作来提供高级功能。

本发明提供了一种装置，包括：一个半导体基片；多个装在所述基片上的接口处理器，所述处理器的数目至少为5个；装在所述基片上，存储可由所述接口处理器访问的指令的内部指令存储器；装在所述基片上，并存储可由所述接口处理器访问的经过所述装置的数据的内部数据存储器；所述内部数据存储器带有多个存储器单元，所述存储器单元的数目与所述接口处理器的数目相关联；插在所述接口处理器中的每一个和存储器单元之间的请求控制器，可控制从所述接口处理器到所述存储器单元的数据请求的流程；插在所述存储器单元中的每一个与接口处理器之间的存储器判优器，可控制接口处理器对所述存储器单元的访问；每一个所述接口处理器都可访问多个所述存储器单元，每一个所述存储器单元都可对来自多个所述接口处理器的请求作出响应；以及所述基片上有多个输入/输出端口；所述输入/输出端口中至少有一个将所述内部数据存储器与外部数据存储器相连接；且至少有另外两个输入/输出端口在所述接口处理器的引导下与外部网络交换经过装置的数据。

附图说明

前面已对本发明的一些目的作了表述，下面将结合附图对本发明的其它目的进行说明。在这些附图中：

图1是表示本发明接口设备的一个方框图。

图1A是说明MAC的一个方框图。

图2A到2D为在不同系统配置中该接口设备与其它部件相互连接的情况。

图3表示压缩引导帧的流程和处理步骤。

图 4 表示内部引导帧的流程和处理步骤。

图 5 为引导单元的概括形式。

图 6 所示为帧控制信息的格式。

图 7 所示为关联子的形式。

图 8 为命令控制信息格式。

图 9 为地址信息格式。

图 10 为结构编址的一般格式。

图 11 为编址和岛编码图解。

图 12A 为嵌入式处理器相复合的方框图。

图 12B 为嵌入式处理器的示意图。

图 12C 所示为 GxH 处理器的结构。

图 13 为存储器复合的方框图。

图 14 为固定匹配 (FM) 搜寻算法流程图。

图 15 为不使用直接表和使用直接表两种情况下的数据结构流程图。

图 16 是以 Prizma 为例的交换系统的方框图。

图 17 为 CP 的方框图。

图 18 为单片网络处理器的方框图，主要突出其 EDS-UP、EDS-DOWN 和 EPC 功能。

本发明的实施例

在下文中，我们将通过本发明的优选实施例并结合附图对本发明进行更加充分的说明，不过读者应事先明白，本领域的技术人员可以对本发明进行某些修改，但仍能达到本发明的理想效果。因此，应当说，下面的说明只是为本领域的技术人员提供了一种广义的、启发性的揭示，并不等于对本发明的限制。

本公开中的装置是可伸缩的，具有互联桌面或工作组交换机功能，可将其接入网络主干线路，并可提供主干交换机服务。该装置可支持第 2

层、第 3 层和第 4 层以上的硬件。该装置的一些形式是为桌面或工作组交换机集合体设计的，另外一些形式则是为核心骨干网交换机设计的。

用于该装置的体系结构，其基础是接口设备或网络处理器硬件子系统以及在控制点运行的一个软件库，这些在本文后面的内容中将进行充分说明。该接口设备或网络处理器子系统是一种高性能帧转发引擎，它以对 L2、L3 和 L4+ 的协议首标进行解析和翻译。这样就可以通过使用硬件以更快的速度来交换协议。该接口设备或网络处理器子系统提供一个通过箱的快速通道，软件库和控制点处理器提供维护此快速通道所需的管理及路由显示功能。在其上运行的控制点处理器和软件库共同限定系统的控制点（CP）。现实网桥和路由协议（如透明网桥和 OSPF）都在 CP 上运行。CP 也可作为系统中的低速通道。

本公开中的该装置除了可支持硬件设备中的多层转发外，还可单独作为 L2 上的交换机来运行，而且这也是它在本专利的简式公开中的默认运行方式。每个域中都有一个端口，从而使任何设备都能与其它设备通信。该装置可在 L2 配置，从而使系统管理员能够对某些功能进行配置，比如将通信口分为单独的域或主干区、配置虚拟数据网络（VLAN）段、或是配置过滤器以控制广播和多播业务。

这一可伸缩的装置有多种优点。首先，它使系统管理员能够配置 IP 和 IPX 业务的 L3 转发和路由，所使用的硬件与 L2 相同，速度也相同。其次，它在提高性能的同时不需要使用外部路由器来连接园区建筑。第三，它简化或合并了对 L2/L3 服务的管理，将其集中到一个控制点上。最后，它还为 L4+ 功能提供了一些增值特征，从而使系统管理员能够对不同类别的通信量进行分配，从而支持任务关键的应用，它还可使网络分配器达到不同服务器间的动态平衡。

该装置可用作一个模块单元，其基本组成包括：一个接口设备或网络

处理器、一个控制点（CP）和一个交换结构装置（可选）。该接口设备优选情况下提供 L2/L3/L4+ 的快速路径转发服务，而 CP 可提供维护该快速通道所需的管理及路由显示功能。上述可选的交换结构装置用于两个以上的接口设备子系统相互连接的情况。有关该可选交换结构装置，详见美国专利第 5,008,878 号，公开日为 1991 年 4 月 16 日，标题为“High Speed Modular Switching Apparatus for Circuit and Packet Switched Traffic”。

该装置使用印刷电路板单元（这里也称作“托板”）来组合安装。这些印刷电路板单元上装有电路元件，这些电路元件插在装置外壳的连接器的上。类似的装置又称作“可选插件”。对于该装置，我们假设，在有合适的连接器和底板电源接头的情况下，托板的底盘或外壳均可变换。构成各种托板的基本部件是托架子系统。从托架子系统的角度来讲，可以有三种托板。第一种是只有 CP 的托板，它由一个托架子系统和一个 CP 子系统组成。这种只有 CP 的托板主要用于优先考虑冗余性的产品。第二种是 CP+介质托板，它由一个托架子系统、一个 CP 子系统和一个 1-3 介质子系统组成。这种 CP+介质托板主要用于视端口密度比冗余性更重要的产品。第三种是介质托板，它由一个托架子系统、和一个 1-4 介质子系统组成。这种介质托板可用于任何底盘，而且这种介质子系统还可配置。

托板管理包括故障检测、电源管理、新装置检测、初始化和配置。上述管理是通过使用各种寄存器、I/O 信号以及用于 CP 和托架子系统之间通信的引导单元接口来实现的。不过，与底盘不同的是，所有的托板上都有可编程装置和存储器。可编程性的程度取决于托板的种类。当托板上存在 CP 子系统时，CP 和托架子系统均可编程。介质子系统也是可编程的，不过只能间接地通过托架子系统来实现。

在某些功能较强的产品中，还有一个交换托板，其中包含交换结构装置子系统。该托板的管理包括故障检测、电源管理、新装置检测和初始

化和配置。上述管理是通过使用各种寄存器和 I/O 信号来实现的，这些寄存器和信号可映射到 CP 子系统中。

本发明假定这种交换装置的简式形态中有一个控制点处理器，还有一个连接到该控制点处理器的可操作的接口设备。在本公开中，我们更倾向于将该接口设备(又称网络处理器)假设为一个单一的、超大规模集成电路装置(VLSI，超大规模集成电路)或集成电路块，它包含一个半导体基片，基片上有多个接口处理器，所述基片上有内部指令存储器，并存储可以由接口处理器访问的经由该设备的数据，位于基片上的内部数据存储器 and 通过该装置可到达接口处理器的存储数据；以及多个输入/输出端口。在本文中，该接口处理器有时也称为微微处理器或处理单元。在所提供的端口中，至少有一个端口用于将内部数据存储器与外部数据存储器相连接，另外有两个以上的端口用于在接口处理器的指导下通过该接口设备与外部网络交换数据。控制点与接口设备协同工作，先由控制点将指令输入指令存储器，然后由接口处理器来执行这些指令，从而控制在输入端口与输出端口之间的数据交换以及通过数据存储器的数据流量。

本公开中的网络处理器被当成是除组成交换机的各个部件之外的发明。此处公开的网络处理器被认为在其单元内具有其它一些未在此处讨论的发明。

图 1 是一个方框图，用以说明该接口设备集成电路块，其中包含基片 10 和集成在该基片上的多个子部件。这些子部件被排列成上行配置和下行配置。在这里，“上行”是指数据流从网络进入到该装置，而“下行”则是指数据从该装置出来，进入到网络服务中去。数据流分别依照不同的配置流动。这样，就有了一个上行数据流和一个下行数据流。上行线中的子部件包括：入列-出列-调度 UP (EDS-UP) 逻辑电路 16、多路复合 MAC-UP (PPM-UP) 14、交换数据推进器-UP (SDM-UP) 18、系

统接口 (SIF) 20、数据调整串行链接连接 A (DASLA) 22、以及数据调整串行链接 B (DASLB) 24。数据调整串行链接在 1999 年 6 月提出的标题为 High Speed High Speed Parallel / Serial Link for Data for Data Communication 的美国专利申请 第 09/330,968 号中有详细的说明。虽然在本发明中该装置优选了 DASL 连接的形式, 但本发明认为, 为达到较高的数据流速率, 也可使用其它形式的连接, 尤其是将数据流限在 VLSI 结构的范围之内。

下行线中的子部件包括: DASL-A 26、DASL-B 28、SIF 30、SDM-DN 32、EDS-DN 34 以及 PPM-DN 36。集成电路块中也包含多个内部 S-RAM、通信管理调度器 40, 以及嵌入式处理器复合 (EPC) 12。接口设备 38 通过相应的 DMU 总线与 PMM 14 和 36 相连接。接口 38 可以是任何适当的 L1 线路, 如以太网实体 (ENET PHY)、ATM 帧调节器等。接口的种类一定程度上决定于集成电路块所连接的网络介质。有多个外部 D-RAM 和 S-RAM 可供该集成电路块使用。

虽然本公开的网络中一般数据是通过导电体 (如布设在楼宇中的电线和电缆) 从相关的转换器和路由装置中流出的, 但本发明假定, 本公开中的网络交换机和部件也可用于无线环境。举例来说, 这里所讲的媒体接入控制 (MAC) 单元就可以由适当的射电频率单元来替代, 如可以使用知名的硅锗技术, 从而将上述单元直接连接到无线网络。在适当应用上述技术时, 射电频率单元可以由本领域的技术人员集成到本公开所讲的 VLSI 结构中。另外, 射电频率抑或是无线响应器 (如红外线响应器) 可以与本公开的其它单元一道安装在托板上, 从而组成用于无线网络系统的交换装置。

箭头所示为数据在接口设备内部的通常流向。从以太网 MAC 接收的帧由 EDS-UP 放在内部数据存储器的缓冲区。这些帧被当作普通的数据帧或系统控制引导帧排入 EPC 的队列中 (图 1)。EPC 中包含 N 个协议

处理器，最多可并行处理 N 个帧 ($N>1$)。在一个有 10 个协议处理器的实施例 (图 12B) 中，有两个协议处理器是专用的：一个用于处理引导帧 (普通中央处理器，或 GCH)，另一个用于在控制存储器中建立查找数据 (普通树形处理器，或 GTH)。帧如图 12A 所示，EPC 中包含有一个分配器，可将新的帧与空闲的处理器相匹配；一个完成单元，用以维护帧序列；一个由全部十个处理器共享的公共指令存储器；一个用以决定帧的类别的分类器硬件加速器和一个用以帮助确定帧的起始指令地址的协处理器；用以控制帧缓存的读写运算的入口和出口数据存储连接；一个能使十个处理器共享控制存储器的控制存储器判优器；能够对内部接口装置的数据结构进行调试的 Web 控制器、判优器和连接；以及其它硬件结构。

GCH 处理器一旦可用，分配器就会把引导帧发送进来。然后由 GCH 处理器来执行引导帧中各种代码化的操作，如寄存器执行写入，计数器执行读取，以太网 MAC 执行配置更改，等等。查找表格的各种更改，如 MAC 或 IP 条目的增加，也将被传送到查找数据处理器，以便进行控制存储器运算，如存储器读出和写入等。有些命令，如 MIB 计数器读取，要求建立一个应答帧并将其转发到适当接口设备的适当端口上去。在有些情况下，引导帧为接口设备的出口端编码。这些帧被转发到正在查询的接口设备的出口端上，然后执行编码运算并建立适当的应答帧。

数据帧被发送到下一个可用的协议处理器上，在那里执行帧查找。帧数据连同来自分类器硬件加速器 (CHA) 引擎的结果一起被送到该协议处理器上。CHA 对 IP 或 IPX 进行解析。其结果将决定树搜索算法和公共指令地址 (CIA) 的起始。所支持的树搜索算法包含固定匹配树型结构 (要求准确吻合的固定尺寸格式，如第 2 层以太网 MAC 表)，最长前缀匹配树型结构 (要求可变长度匹配的可变长度格式，如子网 IP 转发) 以及软件处理树型结构 (定义范围或字节掩模组的两个图案，如过滤器法则中所使用的)。

查找是在树形搜索引擎（TSE）协处理器的帮助下执行的，该处理器属协议处理器的一部分。TSE 协处理器执行控制存储器访问，空出协议处理器来继续执行。控制存储器上储存着全部表、计数器及其它该微码所需的数据。控制存储器运算是由控制存储器判优器来管理的，判优器在十个处理器间对存储访问进行判优。

帧数据是通过数据存储协处理器来访问的。数据存储协处理器中包含一个初级数据缓冲区（最多可容纳 8 个 16 字节的帧数据段），一个便笺式数据缓冲器（最多也可容纳 8 个 16 字节的帧数据段）和若干用于数据存储操作的控制寄存器。一旦发现了某个匹配，入口帧变更中就会包含一个 VLAN 首标插入或覆盖。这种变更不是由接口设备处理器复合来执行，而是首先得到硬件标志，再由其它入口转换接口硬件来执行。其它帧的变更可由微码和数据存储协处理器通过修改帧的入口数据存储中所包含的内容来完成。

其它数据被收集起来并用于在向交换结构装置发送帧之前构建交换机首标和帧标题。帧数据是通过数据存储协处理器来访问的。

在完成之后，入列协处理器就会建立起必要的形式，以便将该帧加入到交换结构队列并将其发送到完成单元。完成单元保证帧从十个协议处理器到交换结构队列的顺序。从交换结构队列出来的帧被分成以 64 个字节为单位的段，并嵌有帧标题字节和交换首标字节，就象将被送到 Prizma-E 交换机时一样。

从结构接收到的帧由出口 EDS（34）放置在出口数据存储（出口 DS）缓冲区并排入 EPC 队列。一部分帧由分配器发送到某个空闲的协议处理器来执行帧查找。帧数据与来自分类器硬件加速器的数据一起被发送到该协议处理器。分类器硬件加速器运用由入口接口设备生成的帧控制数

据来帮助确定密码指令地址（CIA）的起始。

与在入口检索中的情况相同，出口树形搜索也支持同样的算法。查找是在 TSE 协处理器上执行的，这样可以使协议处理器能够继续执行。所有的控制存储器运算是由控制存储器判优器来管理的，判优器在十个处理器间对分配存储器访问进行判优。

出口帧数据是通过数据存储协处理器来访问的。数据存储协处理器中包含一个初级数据缓冲区（最多可容纳 8 个 16 字节的帧数据段），一个便笺式数据缓冲器（最多也可容纳 8 个 16 字节的帧数据段）和若干用于数据存储操作的控制寄存器。成功查找的结果中包含转发信息，有时还包含帧变更信息。帧变更中也可包含 VLAN 首标缺失、实时激活增值（IPX）或递减（IP），IP 首标和校验重算、以太网帧 CRC 覆盖或插入，以及 MAC DA/SA 覆盖或插入。IP 首标和校验是由校验和协处理器来准备的。这种变更不是由接口设备处理器复合来执行，而是首先生成硬件标志，再由 PMM 入口硬件来执行。在完成之后，入列协处理器就会根据要求建立必要的形式，以便将 EDS 出口队列中的帧加入列列并将其发送到完成单元。完成单元保证帧从十个协议处理器到 EDS 出口队列的顺序，以便馈送到出口以太网 MAC 36。

已完成的帧最终由 PMM 出口硬件发送到以太网 MAC，并从以太网端口输出。

一条内部总线（亦称为 Web）提供对内部寄存器、计数器和存储器的访问。该 Web 中还包含一个外部接口，用以在调试和诊断中控制指令步骤或中断控制。

树形搜索引擎协处理器可提供存储范围检查、非法存储器访问提示，此外还可与协议处理器并行操作执行树形搜索指令（如存储器读出、写

入或读取-增加-写入)。

公共指令存储器由一个 1024x128 的 RAM 和两组复式 512x128 RAM 组成。每套复式 RAM 可提供两份微码，从而使处理器能在相同的地址范围内独立访问指令。每个 128 位字中包含四条 32 位指令，总共可提供 8192 条指令。

分配器控制帧的通过使其到达十个协议处理器，同时管理中断信号和计时器。

完成单元保证帧从处理器复合到交换结构和目标端口队列的顺序。一套内容丰富的指令集中包含条件执行、拼（用于输入散列密钥）、条件转移、有签名的和未签名的操作、先导零字符统计等等。

分类器硬件加速器引擎将每个帧的第 2 层和第 3 层传送到协议首标，并在将帧发送到协议处理器的过程中提供这些信息。

控制存储判优器控制处理器对内部和外部存储器的访问。

外部控制内存选项包含 5 到 7 个 DDR DRAM 子系统，每个子系统支持一对 2Mx16 位 x 4 排或一对 4M x 16 位 x 4 排的 DDR DRAM。DDR DRAM 连接以 133 兆赫的时钟频率运行，一个 266 兆赫的数据频闪仪支持可配置的 CAS 延迟和驱动强度。另外还可根据需要进行选择增加一个 133 兆赫的 ZBT SRAM，其配置可为 128kx36、2x256kx18 或者 2x512kx18。

出口帧可以储存于一个外部数据缓冲区（如 DSO），也可以储存于两个外部数据缓冲区（DSO 和 DS1）。每个缓冲区可以由一对 2Mx16 位 x4 排的 DDR DRAM 组成（最多可储存 256k 的 64 字节帧），或是由一对 4Mx16 位 x4 排 DDR DRAM 组成（最多可储存 512k 的 64 字节帧）。

对于 2.28 Mbps 可选择一个外部数据缓冲区（如 DSO），若要支持 4.57 Mbps 的第 2 层和第 3 层就要增加第二个缓冲区（如 DS1）。增加第二个缓冲区可改善性能，但并不增加帧容量。外部数据缓冲区连接以 133 兆赫的时钟频率运行，它还带有一个 266 兆赫的数据频闪仪，用以支持可配置的 CAS 延迟与驱动强度。

内部控制存储中包含两个 512x128 位的 RAM、两个 1024x36 位的 RAM 以及一个 1024x64 位的 RAM。

内部数据存储器在入口方向（上行）最大可提供 2048 个 64 字节的帧的缓冲。

固定的帧变更包括在入口方向的 VLAN 标记插入和删除、Time To Live 的增量/递减（IP、IPx），以太网 CRC 覆盖/插入，以及在出口（下行）方向的 MAC DA/SA 的覆盖/添加。

端口映射功能可以将一个接收端口和一发送端口复制到指定系统的观察端口，而无须使用协议处理器资源。映射而来的接口设备端口可用以增加帧和交换控制数据。一个单独的数据通道可以对帧进入入口交换接口队列进行控制。

接口设备中集成了四种以太网宏指令。每个宏又可以分别进行配置，从而在 1G 比特或 10/100 的快速以太网模式下运行。每个以太网宏最多可支持十个 10/100 Mbps 的 MAC，或者说四个宏指令中的每一个都可支持一个 1000 Mbps 的 MAC。

图 1A 为一个 MAC 核心部分的方框图。每个宏中包括三个以太网核心设计；即多口 10/100 Mbps MAC 核心（Fenet）、1000 Mbps MAC 核心（Genet）以及 100 Mbps 物理代码底层核心（PCS）。

多端口以太网 10/ 100 MAC 的功能:

支持该物理层上的十个串行介质设备独立连接

能够处理速度为 10 Mbps 或 100 Mbps 的中速或任意速度组合的十个端口

一个 MAC 通过一个时分多路复用连接服务于全部十个端口

在所有端口上以中速支持完全/半双向通信

支持 IEEE 802.3 二进制指数补偿

1000 Mbps 以太网 MAC 核心的功能:

支持到物理 PCS 层上的十亿比特介质独立连接 (GMII) 直接到达物理层

通过 PCS 核心来支持完全的 TBI (8b/10b) 解决方案

以中速支持全双向点到点连接

支持 IBM PCS 核心有效字节信号

1000 Mbps 以太网物理代码底层核心的功能:

执行 8b/10b 代码及解码

支持如 IEEE 802.3z 中所定义的 PMA (10 位) 服务接口, 该接口可与符合 IEEE 802.3z 定义的任何 PMA 相连。

将来自 PMA (二相时钟) 的数据与 MAC (单相) 时钟同步

支持包括下两页在内的自动洽谈

从标准中定义的二相时钟系统向单相时钟转换

向 MAC 提供信号指示出包含新数据的时钟周期

核对 COMMA 收到的代码组 (10 位) 并建立起单词同步

计算并核对 8b/10b 运行的不均匀性

图 2A 到 2D 说明接口设备集成电路块的不同配置。该配置中由于有了 DASL 并连接到交换结构装置而变得更为方便。每个 DASL 中包括

两个信道，即一个发射信道和一个接收器信道。

图 2A 表示一单一接口设备的包配置。在该配置下，发射信道被轮流到接收信道当中。

图 2B 表示两个接口设备集成电路块相连接时的配置。每个接口设备集成电路块中至少装有两个 DASL。在该配置中，一个集成电路块上的 DASL 上的信道与位于另一个集成电路块上的相匹配的 DASL 上的信道相连接而运行。每个集成电路块上的另外一个 DASL 则被卷包起来。

图 2C 表示交换结构中多接口设备相连接时的配置。其中双向箭头表示沿两个方向传输。

图 2D 表示将一台主交换机和一台备用交换机连接到多个接口设备时的配置情况。如果主交换机发生故障，就可以起用备用交换机。

控制点（CP）包括连接到每一种配置的一个系统处理器。CP 上的系统处理器主要提供初始化以及与集成电路块相关的配置服务。CP 可处于以下三个位置之一：在接口设备的集成电路块上，在装有集成电路块的托板上，或是在托板之外。如果是在托板以外，CP 就可能远程的；也就是说，它身处别处，通过接口设备和中央处理器所属的网络来进行通信。CP 上的各个单元如图 17 所示，其中包括若干存储器单元（高速缓存、闪存和 SDRAM）、一个存储器控制器、一个 PCI 总线、以及用于底板和 L1 连网介质的连接器。

图 18 为一个单片网络处理器的方框图，主要突出其 EDS-UP、业务管理（MGT）调度器和 EDS-DOWN（DN）功能。其中的 U-形图标表示队列，而跟踪队列中的内容的控制块（CB）则由长方形图标表示。

下面将对各元件、其各自的功能和交互作用进行说明。

PMM:是网络处理器的一部分，该网络处理器中包含 MAC（FEnet、POS、GEnet）并与外部 PHY 设备相连接。

UP-PMM:这一逻辑电路从 PHY 当中提取字节，将其转换为 FISH（16

字节)的格式,然后送至 UP-EDS。在 PMM 中有 4 个 DMU,每个 DMU 都能够使用 1 GEnet 或 10 FEnet 的装置。

UP-EDS: 该逻辑电路从 UP-PMM 上提取 FISH,将其存入 UP-数据存储器(内部随机存取存储器)。它能够同时处理 40 帧,而且可以在收到适当的脉码调制数字节之后将帧排入 EPC 队列。当 EPC 处理完帧之后,UP-EDS 就将该帧排入适当的目标端口队列并开始将该帧发送到 UP-SDM。UP-EDS 负责全部缓冲区和帧的管理,并负责在向 UP-SDM 的传递完成之后将缓冲区/帧送回到空闲库。

EPC:该逻辑电路中包含一个微微处理器,并可以包含一个嵌入式 PowerPC。该逻辑电路能够通过查看帧标题来决定如何处理该帧(转发、修改、过滤等)。EPC 可以对若干个查找表格进行访问,其硬件可帮助微微处理器跟上网络处理器对高速带宽的要求。

UP-SDM: 该逻辑电路可提取帧并将其转变成 PRIZMA 单元的格式,以便于向交换结构传输。该逻辑电路还能够将 VLAN 首标插入到帧。

UP-SIF:该逻辑电路中包含 UP-DASL 宏指令,与外部开关 I/O 相连。

DN-SIF:该逻辑电路中包含 DN-DASL 宏指令,可以从外部 I/O 接收 PRIZMA 单元。

DN-SDM:该逻辑电路接收 PRIZMA 单元并对其进行预处理,以便帮助重新组帧。

DN-EDS:该逻辑电路提取每个单元并将其组合还原为帧。该单元储存在外部数据存储器,与缓冲区相联来构造帧。在接收整个帧之后,便将该帧排入 EPC 队列。当 EPC 处理完该帧之后,就把该帧排入调度器(如

果存在)的队列中,或是排入目标端口队列。然后 DN-EDS 通过帧发送程序,把帧发送到适当的端口,把所有的变更信息和某些控制信息发送到 DN-PMM。

DN-PMM:从 DN-EDS 中提取信息,将帧格式化为以太网、POS 等,然后将帧发送到外部 PHY。

SPM:该逻辑电路可使网络处理器联接到外部设备(PHY、LED、FLASH 等),但只需要 3 个 I/O。网络处理器使用一个串行接口与 SPM 通信,然后 SPM 启动必要的功能来管理这些外部设备。

上行流程

- 1) 帧达到 PHY
- 2) UP-PMM 收到字节
- 3) UP-PMM 将 FISH 发送到 UP-EDS (Fish 是指帧的一个部分)
- 4) UP-EDS 将 FISH 储存在 UP-DS
- 5) UP-EDS 将首标发送到 EPC
- 6) EPC 处理首标后将入列信息送回到 UP-EDS
- 7) UP-EDS 继续从 UP-PMM 上接收帧的其余部分
- 8) 当可以把适当的数据发送到交换机上时,UP-EDS 就将信息发送到 UP-SDM
- 9) UP-SDM 读取帧数据,然后将其格式化为 PRIZMA 单元
- 10) UP-SDM 将各个单元发送到 UP-SIF
- 11) UP-SIF 将各个单元通过 DASL 序列传输到 PRIZMA
- 12) 当全部数据提取完毕之后,UP-EDS 释放缓冲区/帧

下行流程

- 1) DN-SIF 接收 PRIZMA 单元
- 2) DN-SDM 存储这些单元并对其进行预处理,以便对信息进行重新组合

- 3) DN-EDS 接收上述单元数据并重新组合信息，然后将这些单元连接成下行线上的一个新帧
- 4) DN-EDS 将单元储存在 DN-DS
- 5) 在所有数据接收完毕之后，DN-EDS 将帧排入到 EPC 队列
- 6) EPC 处理首标并将入列信息送回 DN-EDS
- 7) DN-EDS 将帧排入调度器队列（如果存在）或是目标端口队列
- 8) DN-EDS 服务于队列并将帧信息发送到 PCB 中
- 9) DN-EDS 使用该 PCB 将帧“散开”，然后读取适当的数据并这些数据发送到 DN-PMM
- 10) DN-PMM 将数据格式化（如有要求不可更改），然后将帧发送到外部 PHY
- 11) 当不再需要缓冲区时，DN-PMM 就通知 DN-EDS，DN-EDS 就会释放这些资源

帧控制流程

- 1) 将首标从 UP-DS 或 DN-DS 送到 EPC
- 2) EPC 在查找表格中查找标头资料并接收帧队列信息
- 3) EPC 将排队信息送回到 EDS，于是该帧被排入适当的队列
- 4) 单元首标和帧标题与帧数据一起被发送出去，从而有助于帧的重新组合和转发

CP 控制流程

- 1) 控制点将所引导的帧格式化并将它发送到网络处理器
- 2) 网络处理器将所引导的帧排入 GCH 微微处理器队列
- 3) GCH 处理对引导帧进行处理并读取或写入所要求的 Rainier 区域
- 4) GCH 将所有的表更新要求传递到 GTH
- 5) GTH 根据来自引导帧的信息更新适当的表
- 6) 将引导帧确认送回到 CP

网络处理器控制流程

- 1) 微微处理器可建立一个引导帧，以便将信息发送到另一个 Rainier 或控制点
- 2) 该引导帧被送到适当的位置进行处理

接口设备提供中速交换，最多可达 40 个快速以太网端口（图 2A）。如果使用 IBM 的数据调整同步联动装置（DASL）将两个接口设备连接起来，就可支持 80 个快速以太网端口。每个 DASL 差分器对可承载 440 Mbps 的数据。两组八对可提供 3.5 Gbps 的双向连接（8X 每个方向的 440 Mbps）。如图 2C 和 2D 所示，通过将多个接口设备互连到一台交换机（如 IBM 的 Prizma-E 交换机）可建立起更大的系统。该接口设备可提供两个 3.5 Gbps 的双向 DASL 连接，一个主要的和一个辅助的，可用来对本地帧通信提供卷包后路径（当两个接口设备直接相连时，图 2B）或连接到一个多余的交换结构（图 2D，备用交换机）。由以上所述可见，这样的网络处理器芯片是有伸缩性的，即同一集成电路块既可提供低端系统（具有相对较低的端口密度，如 40），又可提供高端系统（具有比较高的端口密度，如 80-N 端口）。

系统中的接口设备经由 10/100 Mbps 的快速以太网端口（最多可达十个）或一个 1000 Mbps 以太网端口与系统处理器相连。配置到系统处理器的以太网位于 EEPROM，后者与上述接口设备相连，于初始化过程中载入。系统处理器通过建立专门的引导帧（压缩为以太网帧）可与同系统中的所有接口设备通信。压缩引导帧通过与其它装置相连接的 DASL 来转发，从而使得系统中所有的接口设备都可通过一个点来控制。

引导帧用以在控制点（CP）与嵌入式处理器复合之间以及在同一接口设备内部进行通信。先前有关引导单元的公开将有助于阐明本专利中的讨论，详见 1998 年 3 月提出的标题为 Efficient Hardware / Software Interface Software Interface for Data Switch 的美国专利 5,724,348。

对于那些源自 CP 的引导帧通信，CP 会在其本地存储器中的数据缓冲器中构建引导帧。CP 设备驱动程序将引导帧发送到网络处理器的介质接口。媒体接入控制（MAC 硬件）将引导帧进行恢复，并将其存入内部数据存储器（U_DS）中。引导帧被送往适当的托板，经处理后根据需要再路由回到 CP。在外部 CP 与接口设备之间传递的引导帧是压缩过的，以便适应外部网络协议。因此，如果外部网络中有以太网，则引导帧就被压缩为以太网帧，如此等等。

以太网压缩为引导通信在 CP 与接口设备之间传输提供了一种装置。接口设备的以太网 MAC（Enet MAC）在接收帧时并不解析目的地址（DA）或源地址（SA）。这种解析是由 EPC 微微码来执行的。引导通信假设接口设备尚未配置，DA 和 SA 无法由 EPC 微微码解析。因此，这些帧本身就能够自我路由。然而，Enet MAC 会解析以太网类型字段以便从数据通信当中辨别引导通信。该引导帧的以太网类型值必须与载入 E_Type_C 寄存器中的值相匹配。寄存器是从闪速存储器中由接口设备的根目录微微码载入的。

CP 在其本地存储器的数据缓冲区中构建引导帧。CP 处理器中的 32 位寄存器的内容以大头格式保存在本地存储器中（如图 3 所示）。当构造好帧之后，CP 的设备驱动程序就将以太网帧发送出去，发送的内容包括：一个有关具体引导单元处理器（GCH）的 DA，一个与 CP 全程 MAC 地址相对应、或是与专用接口的 MAC 地址相对应的 SA，一个表明某个引导帧的特定以太网类型字段，以及引导帧数据。所有到达该端口的以太网帧都是由 Enet MAC 来接收和解析的。对于那些以太网类型值与 E_Type_C 寄存器中的内容相匹配的帧，Enet MAC 就会剥离掉 DA、SA 和以太网类型字段，将引导帧数据存入 UDS 存储器。Enet MAC 一次一个地收集字节，并将其做成以 16 字节为单位的块，称作 Fish。这些字节以大头格式保存，引导帧的第一个字节保存在 Fish 中重要的字节位置（字

节 0)。以后的各个字节相继保存在该 Fish 中随后的字节位置（字节 1，字节 2...字节 15）。上述 16 字节的块又被保存在 U_DS 中的缓冲区，其排位从 Fish 0 开始。以后的各个 Fish 相继保存在该缓冲区中随后的 Fish 位置（Fish 1, Fish 2, Fish 3, 等）。根据需要，还可从空闲库中获得额外的缓冲区来储存引导帧的其余部分。

接口设备 10 内部引导通信流程如图 4 所示。接口设备的 Enet MAC 功能会检查该帧的标题信息以确定该帧为引导帧。Enet MAC 从引导帧中将帧标题删除，然后其余内部缓冲到接口设备的内部 U_DS 存储器。Enet MAC 显示出该帧将被排入总控制（GC 队列），以便由 GCH 进行处理。当到达该引导帧的结尾时，入列、出列和调度（EDS）逻辑电路就会将该帧排入 GC 队列。

与本地 CP 相连的托板上的 GCH 微微码就会检查帧控制信息（参见图 6），以确定该引导帧是否用于系统中的其它托板以及是否在接口设备的下行线执行。如果该帧用于本地托板之外的其它托板，则 GCH 微微码就会将帧控制块（FCB）中的 TB 值更新为引导帧的帧控制信息中的 TB 值，并指示 EDS 将该帧排入多路传送目标托板帧起点（TB_SOF）队列。由于性能方面的原因，所有引导通信都被排入到多路传送 TB_SOF 队列当中，与所述的目的地托板数目无关。

如果该帧只用于送给本地托板，GCH 微微码就会检查帧控制信息中的上行/下行区段，以确定是否在接口设备的上行或下行线来执行该引导帧（参见图 6）。如果引导帧在接口设备的下行线执行，则 GCH 微微码就会将帧控制区（FCB）中的 TB 值更新为引导帧的帧控制信息中的 TB 值，并指示 EDS 将该帧排入多路传送目标接口设备帧起点（TB_SOF）队列。如果帧控制信息表明引导帧在上行线执行，GCH 微微码就会对该引导帧进行解析并执行引导命令中所显示的操作。

在处理引导命令之前，微微码会检查帧控制信息中 $ack/\overline{noack}$ 字段的值。如果该值为 ‘0’ b，则在处理后将该引导帧删除，引导读出命令将不再作用于这一类帧。

如果 $ack/\overline{noack}$ 字段中的值是 ‘1’ b，而 “early/late” 字段中的值是 ‘1’ b，则微微码会在处理任何引导帧中的引导命令之前构造一个早确认引导帧，其帧控制的 TB 字段中的值与 Early_Ack 引导帧中的值相等，而帧控制的 TB 字段中的值与 My_TB 寄存器中的值相等。微微码将早确认引导帧路由回到 CP，其方法是将该帧的 FCB 中的 TB 值更新为局域网控制点地址 (LAN_CP_Addr) 寄存器的 TB 字段中所包含的值，然后指示 EDS 将该帧排入多路传送 TB_SOF 队列。然后微微码对该引导帧的引导命令进行处理并删除该引导帧。引导读出命令不属于此类。

相反，如果 $ack/\overline{noack}$ 字段中的值是 ‘1’ b，而 early/late 字段中的值是 ‘0’ b，则微微码将帧控制信息中的 $resp/\overline{req}$ 字段更改为 ‘1’ b，以此来表明引导帧响应，同时用 My_TB 寄存器的内容替换 TB 字段，并处理该引导帧内的每个引导命令。在处理引导命令的过程中，微微码会用当前引导命令的完成状态代码值来更新下一个引导命令的完成代码字段。微微码将响应路由回到源，其方法是将 (FCB) 中的 TB 值更新为与之相应的 CP 源托板 (LAN_CP_Addr) 寄存器的值，然后指示 EDS 将该帧排入多路传送 TB_SOF 队列。

EDS 对驻留在 TB_SOF 队列中的帧安排转发时间。交换机数据推进器(SDM)根据 FCB 中所包含的信息建立交换结构单元首标和接口设备帧标题。上述单元通过交换结构装置达到目标托板，在托板的 D-DS 存储器中由这些单元重新集合为帧。下行线 SDM 经识别确定该帧为引导帧，然后指示 EDS 将其排入 GC 队列。

来自 GC 队列或 GT 队列的压力促使微微码对该引导帧进行访问和

解析。所有到达下行线的引导帧最初都是排在 GC 队列中。GCH 微微码检查这些帧的帧控制信息 $gth/\overline{gch}$ 的值。如果该 $gth/\overline{gch}$ 值是 '0' b, 则将引导帧排入 GT 队列。否则, 则由 GCH 微微码检查帧控制信息中的 $resp/\overline{req}$ 字段, 以便确定该引导帧是否已执行。如果 $resp/\overline{req}$ 的值为 '1' b, 就说明该引导帧已经执行过, 于是将其路由到 CP。与 CP 连接相对应的目标端口值是由 EPC 微微码来维护的。来自目标端口队列的帧从接口设备送回到 CP。

如果 $resp/\overline{req}$ 字段的值是 '0' b, 则可能相对于 CP 为本地托板或远程托板。解决的办法是将 LAN_CP_Addr 寄存器 TB 字段中的值与我的目标托板 (My_TB) 寄存器中的内容相比较。如果二者相匹配, 则该托板为本地托板, 否则该托板就是 CP 的远程托板。不论是哪种情况, 微微码都会检查帧控制信息中的上行/下行值。如果 $up/\overline{down}$ 为 '1' b, 则将该帧排入轮循 TP 队列, 以便转发到 U_DS 并由 GCH 上行线处理。否则, 微微码 (GCH 或 Gth) 就执行该引导帧的引导命令中所包含的操作。在执行引导命令之前, 微微码会检查帧控制信息中 $ack/\overline{noack}$ 字段的值。如果该值为 '0' b, 则在处理后将该引导帧删除。引导读出命令不属于此类。

如果 $ack/\overline{noack}$ 字段中的值是 '1' b, 而 $early/\overline{late}$ 字段中的值也是 '1' b, 则微微码会在处理任何引导帧中的引导命令之前构造一个早确认引导帧, 其帧控制信息中 TB 字段的值与 My_TB 寄存器中的内容相等。如果该托板远离 CP, 则微微码会将早确认引导帧路由到轮循端口。相反, 如果该托板处于 CP 本地, 则该帧被路由到与 CP 相对应的端口队列。微微码处理引导命令, 与此同时, 轮循端口将早确认引导帧从 D_DS 移动到 U_DS 并将该帧排入上行线 GC 队列, 或者将该帧从端口队列送回到 CP。对于轮循后回到 U_DS 的帧, GCH 微微码会再次看到它, 不过 $resp/\overline{req}$ 字段的值将变为 '1' b。GCH 微微码将该帧路由回到 CP, 其方法是将 FCB 的 TB 字段更新为 LAN_CP_Addr 寄存器的 TB 字段中所包含的值, 然后指示 EDS 将该帧排入多路传送 TB_SOF 队列。EDS 对驻留在 TB_SOF

队列中的帧安排转发时间。SDM 根据 FCB 中所包含的信息建立 Prizma 单元首标和接口设备帧标题。来自该帧的各个单元通过 Prizma 并在 CP 的本地托板上重新集合为帧。下行线 SDM 经识别确定该帧为引导帧，然后指示 EDS 将其排入 GC 队列。这次当 GCH 微微码解析帧时， $resp/\overline{req}$ 字段的值就变为 '1' b。这就意味着该托板是 CP 的本地托板，于是该引导帧就被路由到与 CP 相对应的端口队列。来自该队列的帧从接口设备送回到 CP。

相反，如果 $ack/\overline{noack}$ 字段中的值是 '1' b，而 $early/\overline{late}$ 字段中的值是 '0' b，则微微码将 $resp/\overline{req}$ 字段的值更改为 '1' b，以此来表明引导帧响应，同时用 My_TB 寄存器的内容替换 TB 字段，然后处理该引导帧内的每个引导命令。在处理引导命令的过程中，微微码会用当前引导命令的完成状态代码值来更新下一个引导命令的完成代码字段。如果该托板远离 CP，则微微码会将引导帧路由到轮循端口。相反，如果该托板处于 CP 本地，则该帧被路由到与 CP 相对应的端口队列。轮循端口将引导帧从 D_DS 移动到 U_DS 并将该帧排入上行线 GC 队列，或者将该帧从端口队列送回到 CP。对于轮循后回到 U_DS 的帧，GCH 微微码会再次看到它，不过 $resp/\overline{req}$ 字段的值将变为 '1' b。GCH 微微码将该帧路由回到 CP，其方法是将 FCB 的 TB 字段更新为 LAN_CP_Addr 寄存器的 TB 字段中所包含的值，然后指示 EDS 将该帧排入多路传送 TB_SOF 队列。EDS 对驻留在 TB_SOF 队列中的帧安排转发时间。SDM 根据 FCB 中所包含的信息建立 Prizma 单元首标和接口设备帧标题。来自该帧的各个单元通过 Prizma 并在 CP 的本地托板的下行线上重新集合为帧。下行线 SDM 经识别确定该帧为引导帧，然后指示 EDS 将其排入 GC 队列。这次，当 GCH 微微码从 D_DS 中解析帧时， $resp/\overline{req}$ 字段的值已变为 '1' b。这就意味着该托板是 CP 的本地托板，于是该引导帧就被路由到与 CP 相对应的端口队列。来自该队列的帧从接口设备送回到 CP。

如果由于某种原因 GCH 微微码遇到这样的引导帧，其帧控制信息

的 TB 字段等于 '0000' h, 那么 GCH 微微码就会认为该帧只用于此托板, 并相应行事。这一动作在初始化时很必要, 因为此时所有托板的 My_TB 寄存器中的值都是 '0000'h。CP 通过发送写入引导命令将本地托板的 My_TB 寄存器初始化, 上述命令在引导帧中, 其帧控制信息的 TB 值为 '0000'h。

EPC 中的任何微微处理器都可生成引导帧。该帧可以是非请求引导帧, 也可以是任何其它形式的引导帧。内部生成的这类帧是以一种无确认 (即 $\overline{ack/noack} = '0'b$) 的方式建立的。这些帧可送往同一 EPC 中的两个微微处理器 (GCH 或 GTH) 之一, 或是送往其它托板的 GCH 或 GTH。

非请求引导帧还可发送到 CP。送往相同 EPC 的引导帧是用 D_DS 中的数据缓冲区来构造的。这些帧随后被排入 GC 或 GT 的队列中等待处理。将这些帧处理, 然后以通常方式删除。送往本地 CP 的非请求引导帧是用 D_DS 中的数据缓冲区来构造的。在构造这些帧的同时会表明它们已经由 EPC 执行过 (即 $\overline{resp/req} = '1'b$, TB=My_TB)。这些帧被排入相应 CP 的端口队列。来自该队列的帧被送回到 CP。

送往另一个托板的引导帧可使用 D_DS 或 U_DS 中的数据缓冲区来构造。在构造送往 CP 的非请求引导帧时, 会同时表明它们已经由 EPC 执行过 (即 $\overline{resp/req} = '1'b$, TB=My_TB)。使用 D_DS 中的缓冲区构造而成的帧被排入轮循端口队列。这些帧被移动到 U_DS 并排入上行线 GC 队列。那些 $\overline{resp/req}$ 的值为 '1'b 的非请求引导帧将使用 LAN_CP_Addr 寄存器中的 TB 值被路由到 CP。否则, GCH 微微码就会使用引导帧的帧控制信息中的 TB 值将这些帧路由。在接收托板上, 该帧被排入下行 GC 队列。该托板的 GCH 执行并删除帧 ($\overline{resp/req} = '0'b$, $\overline{gth/gch} = '1'$), 或是将该帧排入 GT 队列 ($\overline{resp/req} = '0'b$, $\overline{gth/gch} = '0'$), 或是将该帧排入相应的 CP 端口队列 ($\overline{resp/req} = '1'b$)。使用 U_DS 中的数据缓冲区构造而成的帧被直接排入上行线 GC 队列。从该点开始, 这些帧将遵循相同的路径,

所接受的处理程序与使用 D_DS 数据缓冲区构造而成的帧相同。图 5 为引导帧的概括形式。

图中的格式是一种逻辑表示法，其中重要的字节在左边，最不重要的字节在右边。4 字节单词从顶端的 0 开始，向页下方依次递增。

由于引导帧必须在 CP 配置接口设备之前被路由并处理，故这些帧必须能自我路由。通常经查找和分类所得的结果包含在引导帧的帧控制信息字段中，从而使集成电路块能够以该信息来更新 FCB，而不必再执行查找操作。引导帧中所包含的目标托板信息用于引导帧控制程序，用于准备 FCB 中的叶 Page 字段。CP 提供目标托板信息，与此同时，GCH 微微码填充 FCB 中的其它字段。SDM 使用这些 FCB 信息来准备单元和帧标题。引导帧的帧控制信息字段的格式见图 6。

图 6 中的每个位的位置的缩写解释如下：

$resp/\overline{req}$ 响应和不要求指示器值。该字段用于区分要求（未处理的）和响应引导帧。

0 要求

1 响应

$ack/\overline{noack}$ 确认或不确认控制值。该字段用于控制 GCH 微微码是（ack）否（noack）对引导帧进行确认。不需确认的引导帧中将不包含任何执行读操作的引导命令。

0 不确认

1 确认

$early/\overline{late}$ Early 与 Late 确认控制值。该字段用于控制是需要在引导帧被处理之前（early）进行确认（ $ack/\overline{noack} = '1'b$ ）还是在其之后（late）进行确认。当 $ack/\overline{noack} = '0'b$ 时，即忽略该字段。

0 在引导帧处理之后确认

1 在引导帧处理之前确认

$neg/\overline{all}$ 不确认或确认所有控制值。当 $ack/\overline{noack}$ 字段的值为 '0'b 时，即忽略该字段，引导命令未成功完成的情况除外。

0 如果 $ack/\overline{noack}$ ='1'b 即确认全部引导帧。根据 $early/\overline{late}$ 的值来确定是早确认还是晚确认。

1 仅确认那些未成功完成的帧。该确认与 $ack/\overline{noack}$ 和 $early/\overline{late}$ 的值无关，所以自然是晚确认。

$up/\overline{down}$ 上行或下行控制值。该值用来控制是在上行线处理帧还是在下行线处理帧。当 $resp/\overline{req}$ 为 '1'b 时，即忽略该字段。所有多路传送引导帧的 $up/\overline{down}$ 值都应应为 '0'b。此外，需要使用 GTH 硬件加速器指令的引导命令其 $up/\overline{down}$ 值应为 '0'b。

0 下行线处理

1 上行线处理

$gth/\overline{gch}$ 普通目录树处理器或引导单元处理器控制值。该值用来控制引导帧，使其到达适当的微微处理器。

0 GCH 微微处理器

1 GTH 微微处理器

TB 目标托板值。当 $resp/\overline{req}$ 为 '0' b 时，该字段中包含路由信息，可供 Prizma 使用。每个位的位置与一个目标托板相对应。如果该值为 '0000'h，则认为该引导帧就是对应该托板的，并执行相应的操作。如果 TB 字段中的一个或多个位单元的值为 '1'b，则表明应将该单元路由到相应的目标托板。当 $resp/\overline{req}$ 为 '1'b 时，该字段中就会包含相应托板的 My_TB 值。

引导帧的第 1 个字中包含一个相关子值（图 7）。该值是由 CP 指定

的，其作用是将引导帧的响应与其各自的要求相关联。相关子中包含多个具有特定功能的位。

每个引导命令从一个命令控制信息字段开始。该命令控制中包含有对 GCH 微微码处理引导帧起辅助作用的信息。这些信息的格式见图 8。

长度值：该值表示控制信息中所包含的 32 位字的总数（Cmd Word 0）、地址信息（Cmd Word 1）以及引导帧的操作对象（Cmd Words 2+）等部分。

完成码值：该字段由 CP 初始化并由 GCH 微微码在处理引导命令时修改。在执行命令表中的引导命令时，GCH 微微码使用该字段表示完成状态。由于所有引导命令表都是以末端定界符引导命令来结束的，故上一个命令的完成状态包含在末端定界符的完成码字段中。

引导命令类型值（符号名称）

符号名称	类型值	类型说明
End_Delimiter	0000	标记引导帧序列的结尾
Build_TSE_Free_List	0001	建立一个自由表。
Software_Action	0010	执行软件动作
Unsolicited	0011	由 EPC 微微码初始化的帧
Block_Write	0100	将一块数据写入串行地址
Duplicate_Write	0101	将复制的数据写入寄存器或存储器。
Read	0110	请求和响应读取存储数据
register	0111	保留
Insert_Leaf	1000	在搜索树中插入一个叶。
Update_Leaf	1001	更新搜索树的一个叶
Read_Leaf	1010	请求和响应读取页数据

Leaf	1011	保留
Delete_Leaf	1100	删除搜索树的一个叶
	1101 - 1111	保留

包含在引导帧中的地址信息可根据在网络处理器内部的编址方案识别元件。地址信息字段的一般格式见图 9。

接口设备使用一个 32 位的编址方案。该编址方案可以为接口设备中每个可访问的结构分配一个地址值。这些结构可以是位于处理器内部的，也可以是与该处理器所控制的接口相连接的。有些结构可以由嵌入式处理器复合（EPC）经由一个名为 Web 接口的内部接口来访问。其余结构可以经存储器控制器接口来访问。在所有情况下，上述地址的一般格式见图 10。

网络控制器被再分成主要集成电路块岛。每个岛都有一个独特的岛 ID 值。这些 5 位的岛 ID 值组成了结构地址中 5 个最为重要的位，这些结构由集成电路块岛来控制。编码的岛 ID 值与集成电路块岛名称之间的通信见图 11。Web 地址的第二部分由接下来最重要的 23 位组成。这些地址字段被分成结构地址部分和单元地址部分。岛之间每个分段的位数各不相同。有些岛可能只含有少数几个大的结构，而其它岛则可能包含许多小的结构。由于上述原因，地址分段也没有固定的大小。结构地址部分用来为岛内部的排列编址，而单元地址部分则用来为该排列内部的元件编址。地址的剩余部分用于容纳 Web 接口的 32 位数据总线限制。这个 4 位字的地址用于选择地址元件的 32 位段。这在将宽于 32 位的结构元件在网络控制器的 Web 数据总线中移动是很必要的。字地址值'0'h 是指结构元件中 32 个最重要的位，而顺序字地址值则先后对应于结构元件中次重要的段。对于未经由 Web 接口访问的结构，则不需要地址中的字地址部分。因此，上行数据存储器、控制存储器和下行数据存储器都使用地址中全部 27 个最不重要的位来存取结构元件。这种格式的另一个例外是

用于 SPM 接口的地址。在这种情况下，地址中的全部 27 个位都被使用，没有一个元件的宽度超过 32 位。

嵌入组合体（EPC）提供并控制接口设备集成电路块的可编程性。其中包含以下部件（参见图 12A）：

N 个处理单元，称作 GxH：这些 GxH 可同时执行储存在一公共指令存储器中的微微码。每个 GxH 由一个称为 CLP 的处理单元核心、16 个 GPR 和一个运算器组成。CLP 中包含一个 3 级管路。每个 GxH 还包含几个协处理器，比如树形搜索引擎。

指令存储器：在初始化期间载入，其中包含微微码，用于转发帧并管理系统。

一个分配器：使帧-地址从上行和下行调度队列中出列。在出列之后，分配器从上行或下行数据存储（DS）中预先提取帧的一部分——首标，并将其储存到内部存储器中。GxH 一旦空闲，分配器就会将带有适当控制信息如密码指令地址（CIA）的帧标题传递到 GxH。分配器还负责计时器和中断信号。

树状检索存储器（TSM）判优器：每个 GxH 都可以使用若干个共享的内部和外部存储器单元。由于存储器单元是共享的，故需要一个判优器来控制存储器的访问。TSM 可由微微码来直接访问，TSM 可用来储存 TSM 中的时效表等。TSM 也可由 TSE 在树状检索期间进行访问。

完成单元（CU）：完成单元执行两个功能。首先，它可将 N 个处理单元连接到上行和下行 EDS（入列、出列和调度岛）。EDS 执行入列动作：将一个帧地址连同称作 FCBPage 的适当的参数排入传送队列、删除队列或调度队列。第二，完成单元可保证帧序列。由于可能出现多个 GxH

处理相同流程中的帧的情况，因此必须采取预防措施，使这些帧以正确的顺序排入上行下行传送队列。完成单元在帧发送时使用一个由分类器硬件加速器生成的标签。

分类器硬件加速器:用于上行帧，分类器硬件加速器为众所周知的帧格式提供分类。分类结果按照 CIA 以及一个或多个寄存器的内容在帧发送期间被送到 GxH。对于下行帧，分类器硬件加速器根据帧标题来确定 CIA。对于上行和下行帧发送，分类器硬件加速器都会生成一个标签用于完成单元中，以维护帧序列。

上行和下行数据存储接口和判优器:每个 GxH 都可使用上行和下行数据存储。当读取多个 Fish 时就会提供读取访问，当将 FishPool 中的内容写回到数据存储时就会提供写入访问。由于有个 N 个处理单元，而同一时间只有一个处理单元可以访问上行数据存储，同一时间也只有一个处理单元可以访问下行数据存储，因而每个数据存储都需要有一个判优器。

WEB 判优器和 WEBWatch 连接:WEB 判优器在各个 GxH 访问 WEB 时判优。所有的 GxH 都可以访问 WEB，这样就可以访问接口设备中所有的存储器和寄存器功能。这样，任何 GxH 也都可以修改或读取所有的配置区域。WEB 可视作接口设备的内存映象。WEBWatch 连接，使用 3 个集成电路块 IO 插脚提供从集成电路块外部到整个 WEB 的访问。

调试中断信号和单步控制:WEB 允许 GCH 或 WEBWatch 在必要时控制集成电路块上的每个 GxH。例如，WEB 可被 GCH 或 WEBWatch 用于 GxH 上的单步指令。

一个嵌入式通用处理器，如 PowerPC。

GxH 有 4 种类型 (图 12B):

GDH (一般数据处理器)。有 8 个 DH。每个 GDH 都有一个由 5 个协处理器组成的完全 CLP (在后面的部分另有说明)。GDH 主要用于转发帧。

GCH (引导单元处理器, Guided Cell Handler)。GCH 的硬件与 GDH 的完全相同。不过, 引导帧只能由 GCH 来处理。通过在 WEB 上 (CLP_Ena 寄存器) 编程可以使 GCH 也能够处理数据帧 (此时它起到 GDH 的作用)。与 GDH 相比, GCH 还具有额外的硬件: 即协助执行目录树插入和删除的硬件。GCH 用于执行与引导单元有关的微微码, 还用于执行集成电路块以及与微微码有关的和目录树管理, 诸如时效管理以及与 CP 和/或其它 GCH 的控制信息交换。在不执行上述任务的情况下, GCH 就执行与帧转发有关的微微码, 此时, 它的功能与 GDH 完全一样。

GTH (普通目录树处理器)。GTH 还具有其它一些硬件来协助执行目录树插入、目录树删除和绳管理。当 GPQ 中无帧 (包含目录树管理命令) 时, GTH 将处理数据帧。

GPH (普通 PowerPC 处理器)。与 GDH 和 GTH 相比, GPH 还具有其它一些硬件。GPH 通过一个邮箱接口 (i/f) 与通用处理器相联接。

GxH 的数目 (10) 是一个很好猜的数字。性能评价可确定实际需要多少个 GxH。该体系结构和结构是完全可伸缩的, 它可以升级为更多的 GxH, 而唯一的限制是硅面积的大小 (这里也需要一个较大的判优器和指令存储器)。

每个 GxH 的构造如图 12C 所示。除带有通用寄存器 (GPR) 和算术及逻辑部件 (ALU) 的 CLP 之外, 每个 GxH 中还包含以下 5 个协处

理器：

(DS) 协处理器接口。与分配器和各个子岛的接口，上述接口可提供对上行和下行数据存储的读取和写入访问。DS 接口中包含所谓的 FishPool。

树形搜索引擎协处理器 (TSE)。TSE 执行在目录树中的搜索，还可与树状检索存储器 (TSM) 相联接。

入列协处理器。与完成单元接口相连，其中包含 FCBPage。该协处理器中包含一个 256 位的寄存器，上面带有额外的硬件加速器，微微码必须使用该硬件加速器才能建立起包含入列参数的 FCBPage。在 FCBPage 建立起来之后，微微处理器就执行入列指令，该指令让协处理器将该 FCBPage 送往完成单元。

WEB 接口协处理器。该协处理器提供一个与 WEB 判优器的连接，允许从接口设备 WEB 读取以及向接口设备 WEB 写入。

校验和协处理器。在帧上生成和校验，这些帧保存在 Fishpool (有关 Fishpool 下文另行说明) 中。

这些处理单元是在入口处理与出口处理之间共享的。可通过编程来设定为入口处理保留多少带宽，为出口处理保留多少带宽。在当前设备中有两种模式：50/50 (即入口和出口取得相同的带宽) 或 66/34 (即入口所获的带宽为出口的两倍)。

处理单元的操作是受事件驱动的。也就是说，是把帧的到达如同计时器的间歇或中断信号一样作为一个事件来处理。交换器以相同的方式处理不同的事件，尽管顺序分先后 (首先是中断信号，然后是计时器事

件，最后是帧到达事件）。当一个事件被送到处理单元时，相应的信息也随之送到该处理单元。帧到达事件中包含帧标题的一部分以及来自硬件分类器的信息。计时器和信号中断事件则包括代码进入点及其它与该事件相关的信息。

当帧到达入口端时，且从该帧接收到的字节数超过了可编程的限度时，该帧控制块的地址就被写入 GQ。

当在出口端完成一个帧的重新组合之后，该帧的地址就被写入 GQ。GQ 共有四种类型（每一种类型都有入口型式和出口型式，见图 12B）：

GCQ: 包含必须由 GCH 处理的帧

GTQ: 包含必须由 GTH 处理的帧

GPQ: 包含必须由 GPH 处理的帧

GDQ: 含有可由任何 GDH（或 GCH / GTH，当它们能够处理数字帧时）处理的帧。GDQ 有多项优先权，这样，排入较高优先级的 GDQ 中的帧就比排入较低优先级队列中的帧得到优先处理。

有些处理单元有专门的用途。在当前设备中，有四种类型的处理单元（GxH），参见图 12B:

GDH（一般数据处理器）。GDH 主要用于转发帧。

GCH（引导单元处理器）。GCH 的硬件与 GDH 的完全相同。不过，引导帧只能由 GCH 来处理。通过在 WEB（CLP_Ena 寄存器）上编程可以使 GCH 也能够处理数据帧（此时它起到 GDH 的作用）。

GTH（普通目录树处理器）。与 GDH / GCH 相比，GTH 具有额外的硬件：即协助执行目录树插入、目录树删除和绳管理的硬件。当 GPQ 中无帧（包含目录树管理命令）时，GTH 就处理数据帧。

GPH（普通 PowerPC 处理器）。与 GDH/GTH 相比，GPH 还具有其它一些硬件。GPH 通过一个邮箱接口与嵌入式 PowerPC 相连接。

在现实设备当中，GCH、GTH 和 GPH 的角色可由一个处理单元来担当。例如，一个设备可以将一个处理单元当作 GCH 和 GPH 来使用。类似的解释也适用于 GCQ、GTQ 和 GPQ。

数据存储协处理器的用途为：

与上行数据存储相连，上行数据存储中包含从该介质接收到的帧；与下行数据存储相连，下行数据存储中包含从 Prizma Atlantic 收到的重新集合后的帧。

数据存储协处理器也可以在发送计时器事件或中断信号时接收配置信息。

数据存储协处理器可以在帧上计算和校验。

数据存储协处理器中包含一个 FishPool（可容纳 8 个 Fish）、一个暂存器（可容纳 8 个 Fish）以及若干个控制寄存器，用以从上行数据存储中读取 FishPool 的内容，或是将 FishPool 的内容写入下行的数据存储当中。我们可以把 FishPool 想象为某种数据存储的工作区域：它不是直接读取/写入到数据存储，而是将大量的帧数据从数据存储中读出后放入 Fishpool，或是将大量的帧数据从 Fishpool 写入到数据存储中。传输单位为 Fish，一个 Fish 为 16 个字节。

我们可以把 Fishpool 想象为一个容纳 8 个 Fish 的存储器，也就是 8 个 128 位的字。在 CLP 处理器体系结构中，Fishpool 是一个 128 字节的寄存器阵列。Fishpool 中的每个字节中含有一个 7 位字节的地址（0...

127)，而入口的基础是一个 16 位或 32 位的基片。与所有的寄存器阵列一样，Fishpool 中有一个循环编址方案。也就是说，为一个字（即 4 个字节）编址是从 Fishpool 中的位置 126 开始，然后回到字节 126、127、0 和 1。而且，从数据存储协处理器的角度来看，Fishpool 中的 Fish 位置有一个 3 位 Fish 地址。

在帧发送时，帧的前 N 个 Fish 会由分配器自动复制到 Fishpool 中。N 的值可以在 PortConfigMemory 中进行编程。一般来说，对于上行帧发送 N 等于 4，对于下行单路传送的帧发送 N 等于 2，对于下行多路传送的帧发送 N 等于 4，对于中断信号和计时器 N 等于 0。

微微码可以从帧中读取更多的帧数据，此时数据存储协处理器会自动将帧数据读入位于下一个 Fish 地址的 Fishpool，并在到达 Fishpool 的边界时自动绕回到 0。微微码还可以在绝对地址上读取或写入上行/下行数据存储。

WEB 协处理器与 EPC WEB 判优器相连接。EPC WEB 判优器在十个 GxH 之间判优，于是 WEB 监视器成为接口设备 WEB 接口上的主导。这就允许全部 GxH 在 WEB 上读写。

接口设备存储复合可以为嵌入式组合体（EPC）提供储存设备，见图 12A。该存储复合中包含目录树搜索存储器（TSM）判优器和多个芯片内存储器和芯片外存储器。这些存储器可储存树形结构、计数器以及微微码要求通过存储访问的其它任何内容。此外，这些存储器还可用来储存用于硬件的数据结构，如自由表、队列控制块等。在默认情况下，任何未分配给目录树或未被硬件使用的存储器单元都可供微微码使用，如计数器和时效列表。

图 13 为存储器组合的更详细的方框图。目录树搜索存储器（TSM）

判优器提供在嵌入式处理器（GxH）与各个存储器之间的通信联络。这些存储器包括 5 个集成电路块上的 SRAM、1 个集成电路块外的 SRAM 以及 7 个集成电路块外的 DRAM。TSM 判优器包括十个请求控制单元（每个控制单元与一个嵌入式处理器 GxH 相连接）和 13 个存储器判优器单元，每个单元分别用于一个存储器。一个总线结构将请求控制单元与判优器单元相连接，使得每个控制单元以及与之相连接的 GxH 都可以访问所有的存储器。

控制单元中包含必要的硬件，可以在嵌入式处理器（GxH）与判优器之间引导数据。

SRAM 判优器单元着重对嵌入式处理器 GxH 与芯片内和芯片外的 SRAM 之间的数据流进行管理。

DRAM 判优器单元则着重对嵌入式处理器（GxH）与芯片外 DRAM 装置之间的数据流进行管理。

每个存储器判优器中包含一个“后门”入口，一般用于该集成电路块的其它部分，这些部分具有最高级的访问优先权。

DRAM 存储器可以以两种模式来运行：

TDM 模式。对于 DDRAM 中 4 个区块的存储访问是在读取窗和写入窗之间交互进行的。在读取窗中，对 4 个区块中任何一个的访问都是只读的；而在写入窗中，对 4 个区块中任何一个的访问都是只写的。对多个 DDRAM 使用 TDM 模式可以在各个 DDRAM 之间共享某些控制信号，从而省去一些集成电路块 IO（集成电路块 IO 是非常稀有的资源）。

非 TDM 模式。对 DDRAM 中 4 个区块的存储访问可以是读写结合的，但这必须遵循一定的法则。例如，我们可以通过同一个访问窗口在区块 A 中读，同时又在区块 C 中写。

TSM 判优器允许 N 个请求者同时访问 M 个存储器。当有多个请求者想访问同一存储器时，就会执行循环判优。

这 M 个存储器可以有不同的属性。在我们当前的设备中，有三个存储类型：内部 SRAM、外部 SRAM 和外部 DDRAM。

这 M 个存储器与 N 个请求者属同类：任一请求者都可以访问任一存储器。

某些存储器在逻辑上分成多个子存储器（如 DDRAM 中的 4 个区块），这些子存储器逻辑上可以同时访问。

这 M 个存储器中的一部分用作控制存储器，其中含有内部使用过的数据结构，这部分存储器比微微处理器具有更高的优先访问级。由于微微处理器可以读取控制存储器的内容，故也能够对集成电路块进行调试。

判优器支持读取访问、写入访问和读取-添加-写入。通过读取-添加-写入，可以通过原子运算将一个 N 字节的整数添加到存储器的内容中。

访问 M 个存储器所使用的是普通编址方案，因而存储器中对象的物理位置是透明的。

树形搜索引擎中的目录树的概念是指存入和检索信息。检索就是指基于密钥的树形搜索以及插入和删除操作，密钥为位式组合格式，比如 MAC 源地址，或是串联的 IP 源地址和 IP 目的地址。信息储存在一个称作叶的控制块中，叶中至少含有上述密钥（所储存的位组合格式实际上为散列密钥，这一点下文还会谈到）。叶中还包含一些其它信息，如时效信息或用户信息，用户信息可以是转发信息（如目标托板和目标端口数）等。

有各种不同的树型（FM、LPM 和 SMT），于是就相应地有不同种类的树型搜索：固定匹配、软件管理树型结构以及最大前缀匹配。在树形搜索中，还有另外一个自选的标准可用于叶的检查，那就是 VectorMask。还可运用围起、老化和锁定来提高搜索性能。

用于 FM 树型结构的检索算法见图 14。该检索算法在包含密钥的输入参数进行运算，首先对密钥执行散列，然后访问直接表（DT）、穿过模式搜索控制箱（PSCB）中的树型结构，最后在叶上结束（如图 14）。有三种不同的树型结构，每种树型结构都有其各自的检索算法，因此就需要根据不同的法则在树型间穿行。例如，对于固定匹配（FM）树型结构，其数据结构为 Patricia 树型结构。当发现一个叶之后，该叶就是唯一可能与输入端密钥相匹配的对象。而在软件处理树型结构中，就会有多个叶被链锁在同一链接表中。此时，需要用输入端密钥检查该链中所有的叶，直到发现一个匹配项，或是直到该链已全部检查过。所谓的“最终比较”操作是将输入端密钥与储存在叶中的格式相比较，它可以检验该叶是否与输入端密钥真的匹配。如果找到了相应的叶并且实现了匹配，检索的结果就为“OK”，如果对于所有其它情况，检索的结果就为“KO”。

检索运算的输入端由下列参数组成：

密钥(128 位)。密钥必须是在搜索(或插入/删除)之前由专门的微微码指令来建立。只有一个密钥寄存器。不过,在树形搜索开始之后,密钥寄存器就会通过微微码来建立用于下一个检索的密钥,这一操作是在 TSE 进行搜索的同时进行的。这是因为 TSE 将密钥散列并且将结果存储在内部 HashedKey 寄存器中(这样,实际上就有两个密钥寄存器)。

KeyLength(7 位)。该寄存器包含密钥长度的各个位。密钥长度是在建立密钥的过程中由硬件自动更新的。

LUDefIndex(8 位)。它是到 LUDefTable 的指针,LUDefTable 中包含全部树型结构的定义,搜索就是在这里进行的。关于 LUDefTable 下文会有详述。

TSRnr (1 位)。搜索结果可以存储在树形搜索结果区 0 (TSR0),也可存储在 TSR1。这是由 TSRnr 来确定的。在 TSE 执行搜索的同时,微微码可以访问另一个 TSR 以解析先前搜索的结果。

VectorIndex(6 位)。对于那些可启用 VectorMask (这是在 LUDefTable 中确定的)的树型结构,VectorIndex 就会在 VectorMask 中标志一个位。当到达搜索结尾时,该位的值就被送回供微微码使用。

输入端密钥被散列为 HashedKey,如图 14 所示。有 6 种固定的散列算法可以使用(一种“算法”执行非散列函数)。使用哪种算法在 LUDefTable 中有规定。还有一个可编程的散列函数可用来增加挠性。

散列函数的计算输出总是一个 128 位的数,其属性为在原输入端密钥与输出端的散列函数之间一一对应。正如在下文中要说明的,该属性

使得在直接表之后开始的树型结构的深度达到最小。

如果对树型结构启用了颜色，如图 14 中所示的情形，则会将 16 位的彩色寄存器插入 128 位的散列函数计算输出中。该插入是在直接表之后直接发生的。也就是说，如果直接表中包含 2^N 个条目，则在第 N 个位的位置插入 16 位色值，如图所示。散列函数的计算输出与插入的色值（当启用时）一起被储存在 HashedKey 寄存器中。

散列函数是这样定义的，即让计算输出中的多数熵驻留在最高位。HashedKey 寄存器中的 N 个最高位可用作进行直接表(DT)的指针。

搜索从进入直接表开始：从直接表中读取一个 DTEntry。用于读取 DTEntry 的地址是从 HashedKey 中的 N 个最高位计算而来的，用于 LUDefTable 中所定义的树型结构属性的地址也是这样得来的。这一点将在下面详细介绍。DTEntry 可以被看作是树型结构的根。使用哪种树型结构的数据结构取决于树型结构的类型。在这一点上，我们完全可以认为，Patricia 树型结构的数据结构可用于 FM 树型结构，而 Patricia 树型结构的各种延展则可用于 LPM 和 SMT 树型结构。

图 15 中所示为一个 8 个条目的 DT 的使用举例。可以看出，搜索时间（即 必须访问的 PSCB 的数目）可以由 DT 来减少。这样，通过增加 DT 的容量，可以实现存储器的使用与搜索性能之间的平衡。

从图 15 可以看出，DTEntry 中可以包含下列信息：

空。该 DTEntry 与任何叶都不相连。

指向叶的指针。有一个叶与该 DTEntry 相连接。

指向 PSCB 的指针。有多个叶与该 DTEntry 相连接。

DTEntry。DTEntry 定义树型结构的根。

软件管理树型结构的搜寻算法与生成该树型结构的算法见美国专利申请第 09/312, 148 号。

有一种名为“选择位算法”的算法，它是根据从一套或多套法则中挑选出来的名为“法则”项运用某种量度来建立一个二元搜索树。我们这里所讲的例子都是依据因特网协议（IP）首标，但也可使用任何一种固定格式的首标来代替。

在 IP 中，每个法则都与特定的密钥相匹配，这些密钥可由下列子段组合而成：源地址(SA)、目的地址(DA)、源端口(SP)、目的端口(DP)和协议(P)。这些数据分别为 32、32、16、16 和 8 个位长，这样，要测试的密钥就包含 104 位。选择位算法会找到那些特别有用的 104 位。通过测试少数几个有效的位，可以在众多可能有用的应用程序中将多数剔除，而只留下一个或几个有用的法则。对于某些法则，也可以通过简单比较法来测试不等式。位测试和比较是在二元树中通过逻辑来组织进行的。树型结构被映射到一个启用硬件的结构中，在那里以高速测试位。通过测试得出可能与密钥相吻合的一种或少数几种法则（称为叶链）。在前一种情况下，再由该法则对该密钥进行完全测试。在后一种情况下，再运用比较和完全法则测试的方法对密钥进行点阵测试。

每组法则中的每个法则都伴随着一个动作，当该法则与该密钥相吻合且有最高优先权时，就会执行该动作。各种法则也会出现交叉（一个密钥与两个或两个以上的法则相吻合）。在这种情况下，就要为法则设定优先级 1,2,3,...，从而使两个交叉的法则有不同的优先权（如果密钥与两个或两个以上的法则相吻合，就需要由管理员来宣布以哪个法则为主）。这样，如果在位测试和比较之后，还剩下不只一个法则需要测试，就按照这些法则的优先顺序来进行测试。优先级的数字越小，所表示的法则的优先级越高。

如果一个相吻合的也找不到，就需要做一些默认规定。

有关最长前缀匹配法的搜寻算法公布于美国专利第 5,787,430 号。该方法要求从数据库的一个节点(根节点)进入；确定从一个节点经上述树形数据库到另一个节点的搜索路径，其方法是，依次对那些只包含识别下一个（子）节点所需条目的搜索变元的各段进行处理，同时依次处理二级链接信息，直至将上述各段都处理完或是到达一个无二级链接信息的（叶）节点；将上述搜索变元与储存在搜索路径末端节点中的条目相比较；如果在该节点上连搜索变元与所述条目之间的局部匹配都没找到，就通过处理当前节点的一级链接信息退回到上述搜索路径；重复前两个步骤，直到至少找到局部匹配或是达到根节点。

图 16 所示为主交换结构装置的一个实施例。每个接口设备集成电路块中最好集成有两个以上的并行-串行口，这些口可接收并行数据并其转换为高速串行数据流，然后再将这些数据通过串行链接转发到交换结构装置。把从高速串行链接上的交换结构装置接收到的数据通过另一个 DASL 转换为并行数据。在这里，我们以一个称为数据调整串行链接（DASL）的串行器/解串器为实施例来进行说明。

至少有一个 DASL 将交换结构装置连接到串行链接。来自串行链接的数据被转换为并行数据后，然后传送到交换结构装置。与此类似，来自交换结构装置的并行数据被转换为串行数据，然后转发到串行链接。也可将串行链接集合起来，以提高处理量。

仍然参看图 16，交换系统中包括交换结构 11、输入交换适配器 13(13-1... 13-k)，它们与交换结构输入端口 15(15-1...15-k)相连接，以及输出交换适配器 17(17-1...17-p)，它们与输出端口 19(19-1...19-p)交换结构相连接。

输入和输出传输链路 21 (21-I...21-q) 和 23(23-I...23-r)分别通过线路(链接)适配器 25(25-I...25-q)和 27(27-I...27-r)连接到交换系统。传输链路执行由电路交换的通信量或是由包交换的通信量与所连接的单元(如工作站、受话器组等由 WS 指定的链接)之间的往来传送、与局域网(由局域网路指定的链接)之间的往来传送、与综合服务数字网综合服务数字网(由 ISDN 指定的链接)之间的往来传送,或是与所有其它通信系统的往来传送。传输链路执行由电路交换的通信量或是由包交换的通信量与所连接的单元(如工作站、受话器组等由 WS 指定的链接)之间的往来传送、与局域网(由局域网路指定的链接)之间的往来传送、与综合服务数字网综合服务数字网(由 ISDN 指定的链接)之间的往来传送,或是与所有其它通信系统的往来传送。此外处理器还可连接到交换适配器 13 和 17。线路适配器(LA)与交换适配器(SA)共用同一接口。

在输入端交换适配器上,来自分组交换和电路交换接口的各种服务被集中起来,然后转换为统一的迷你包(其长度为固定的几种之一),其首标中包含路由选择信息,该信息用以指定交换机的输出端口(以及外向链路)。在下一个部分当中,将详细介绍迷你包的格式、迷你包在输入端交换适配器上的生成以及在输出端交换适配器上的解包过程。

通过快速自我路由互连网络,交换机网可将迷你包从任意一个输入端路由到任意一个输出端。自我路由网络的结构可以使迷你包在内部同时路由而不发生任何冲突。

交换系统的中心部分是交换结构。这里,我们研究两种不同的实现方式,并分别予以说明。在第一种实现方式中,交换结构中的每个输入端口都有一个自我路由的二元树,分别将对应的输入端口与所有输出端口相连接;交换结构中有 k 个这样的树型结构(如果能提供 k 个输入端口),它们相互组合在一起。在另一种实现方式中,有一个带有输出端 RAM 的

作为每个输出端口的一个片段, 它将所有的输入端口连接到相应的输出端口; 交换结构中有 P 个这样的片段组合在一起(如果能提供 P 个输出端口)。

有关 DASL 的说明详见 1999 年 6 月 11 日的申请, 其序号为 09/330,968。DASL 接口从 CMOS ASIC 这样的并行接口上接收数据, 并将来自并行接口的位分割为数目较少的并行位流。这些较小数目的并行位流被转换为一个高速串行流, 然后经由传输介质传送到另一个模块中的接收器。一个带有控制阻抗的差分驱动器将上述数据的串行位流输送到传输介质中。

DASL 对数据流进行解析, 将 N 位并行的数据流变成若干段, 每段中有 n 位, 其中 n 为 N 的一部分; 将数据流的各个 n 位段串起来; 将每个串行的部分通过多个并行信道行中的一个传输出去; 将所传输的每个数据流的部分进行串并转换, 使其恢复为并行的 N 位数据流。

本发明优选实施例中的附图和说明中, 尽管也使用了一些专用名词, 但总体上讲都是取术语的普通的、叙述性的含义, 决无限制之意。

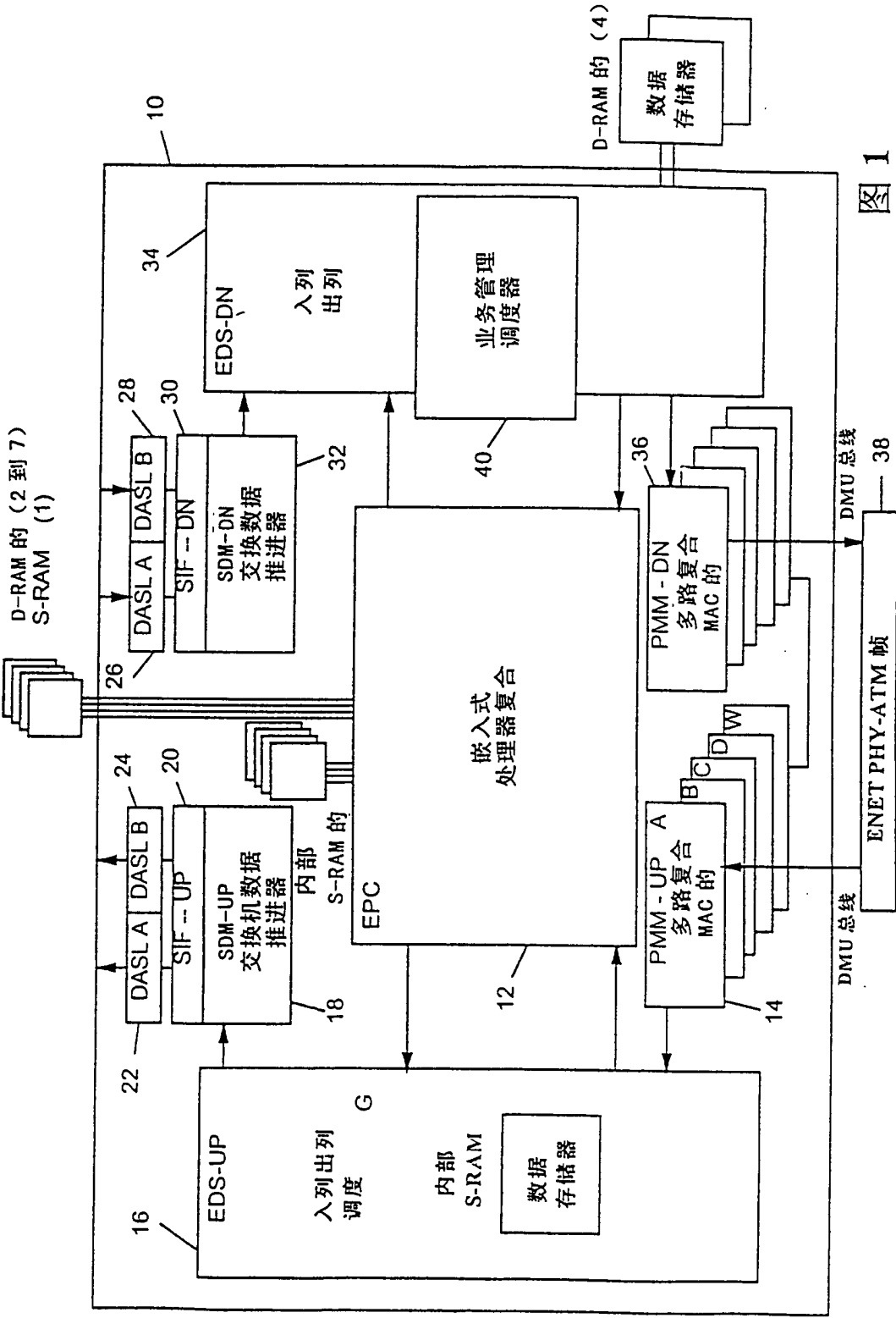


图 1

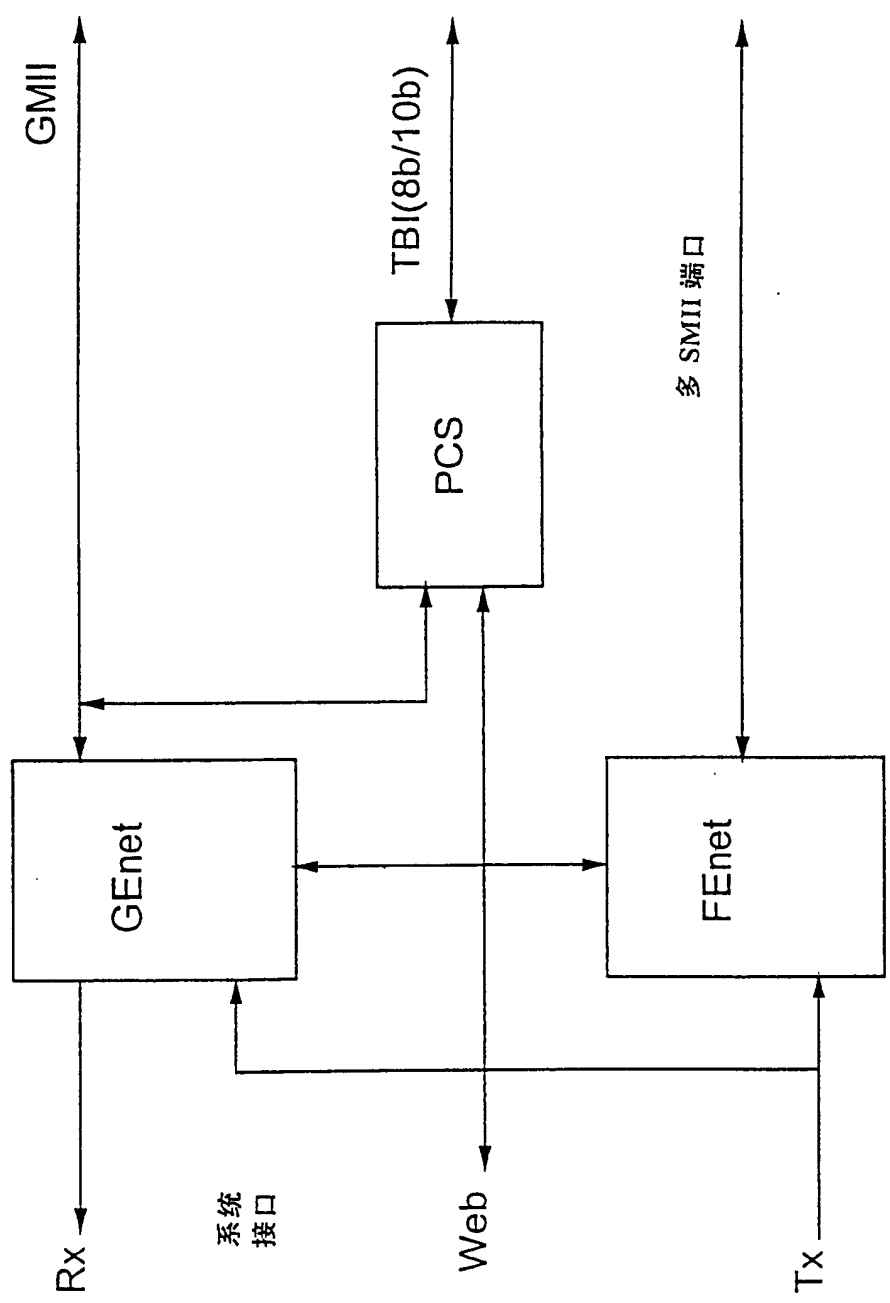


图 1A

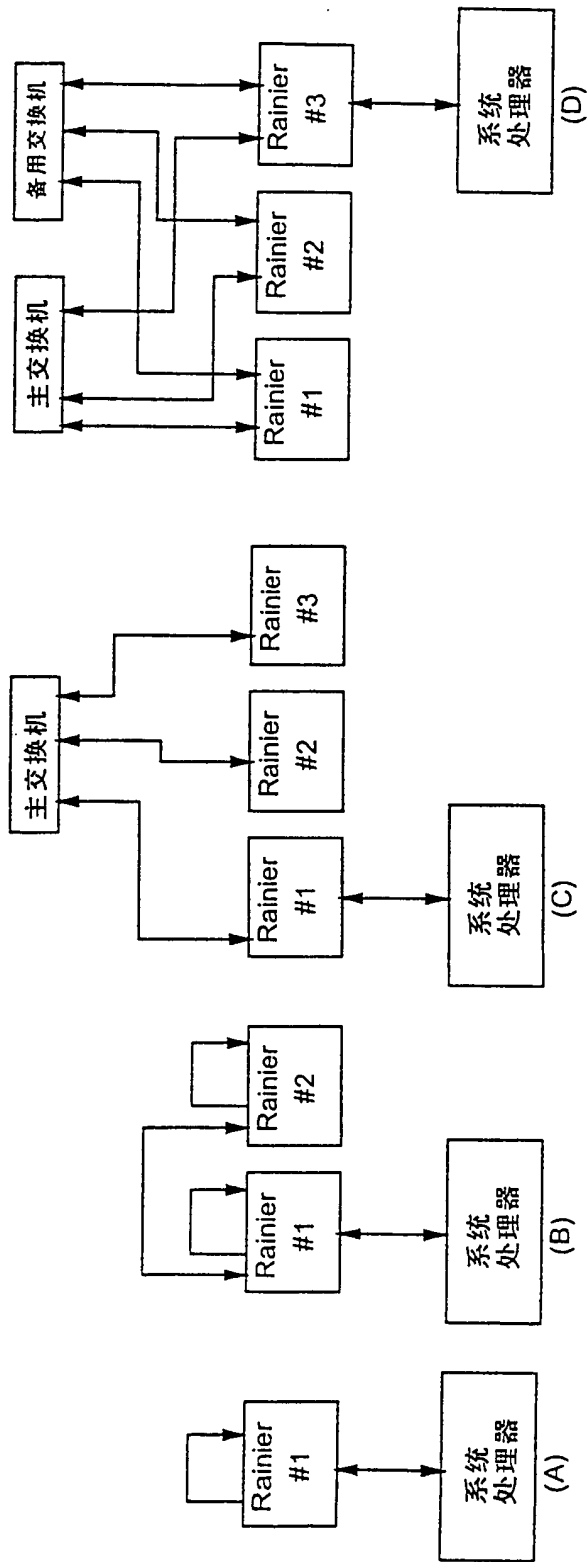


图 2 系统配置

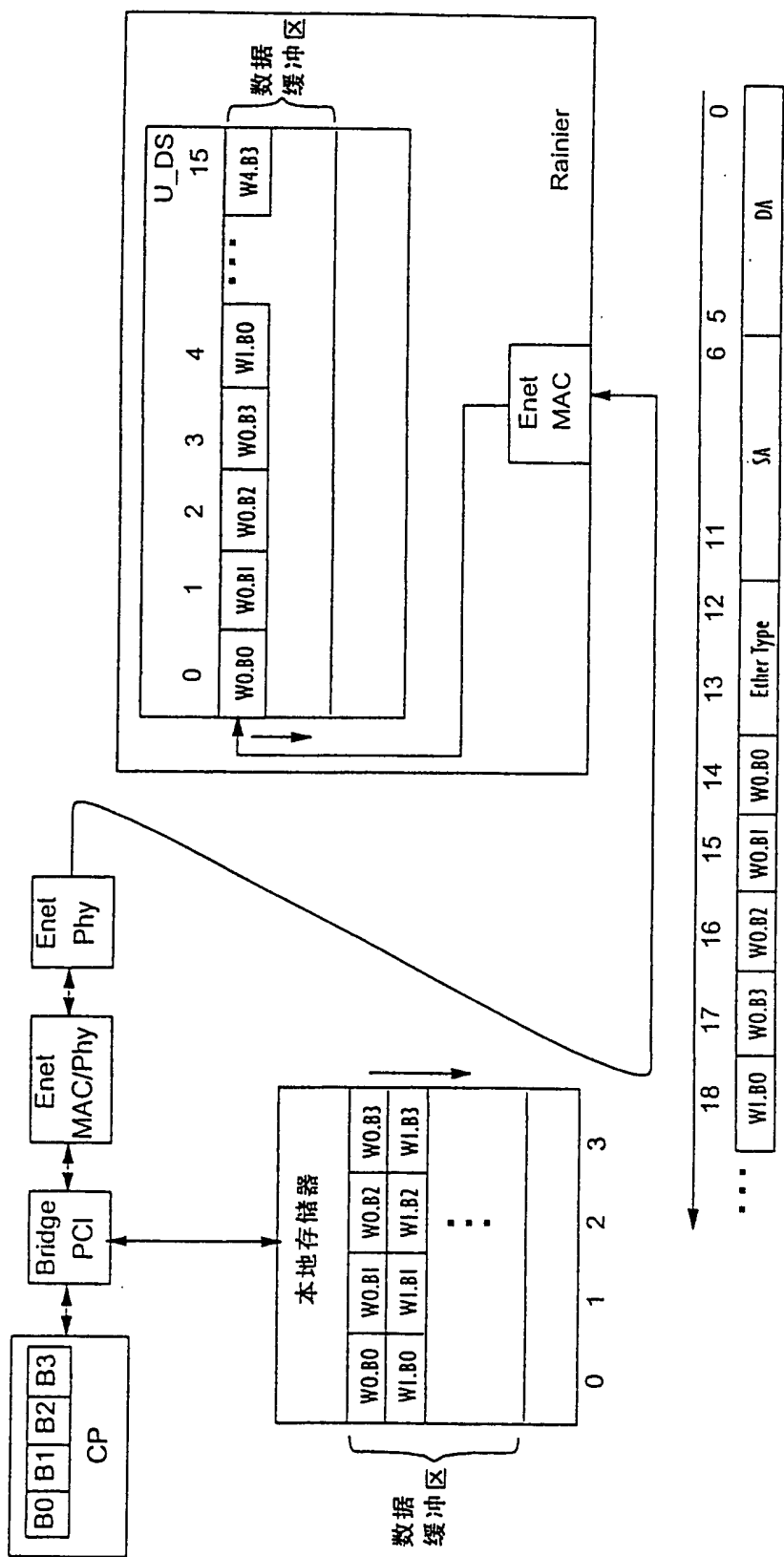


图 3 引导业务数据流（以太网端口）

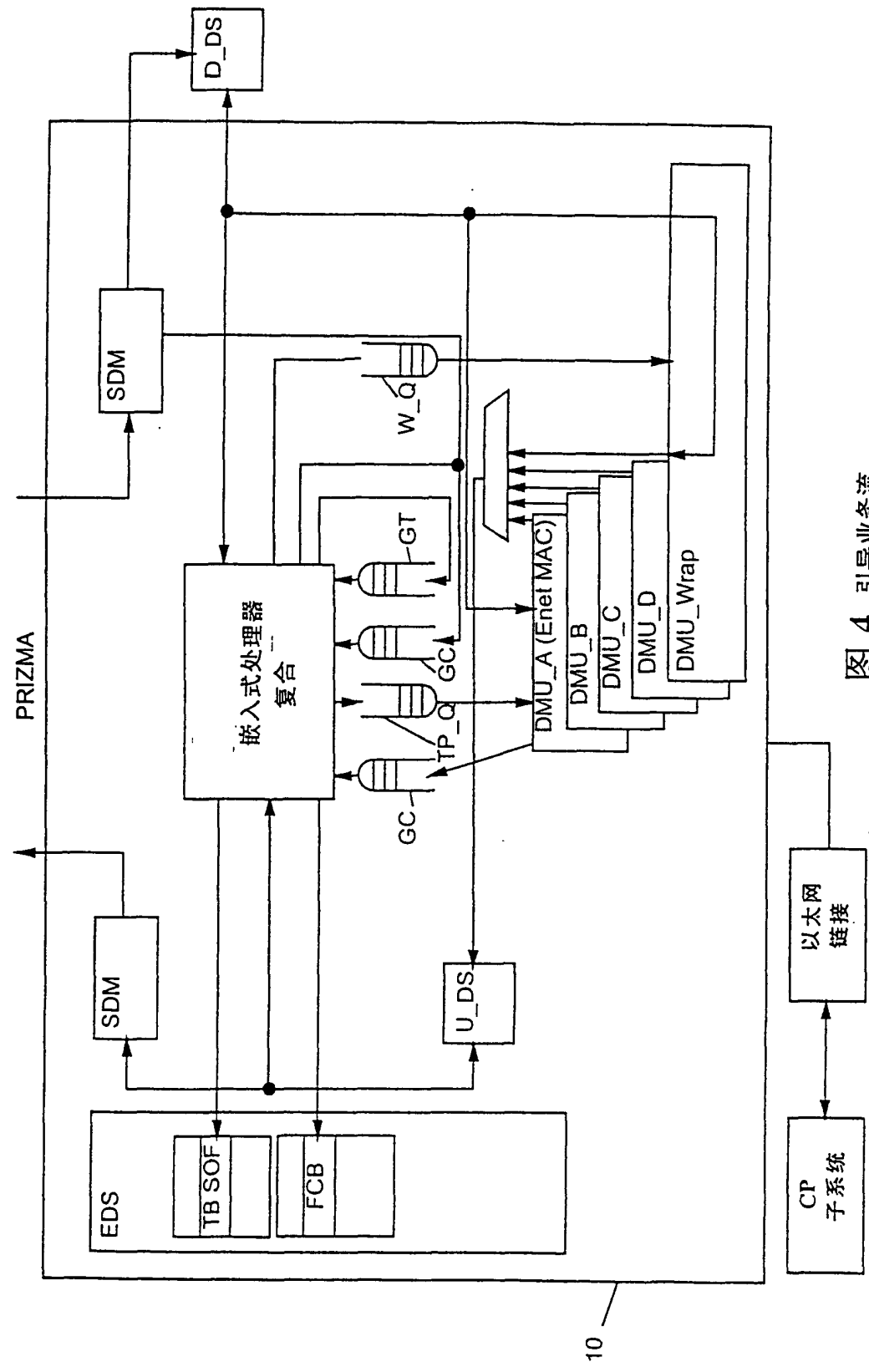
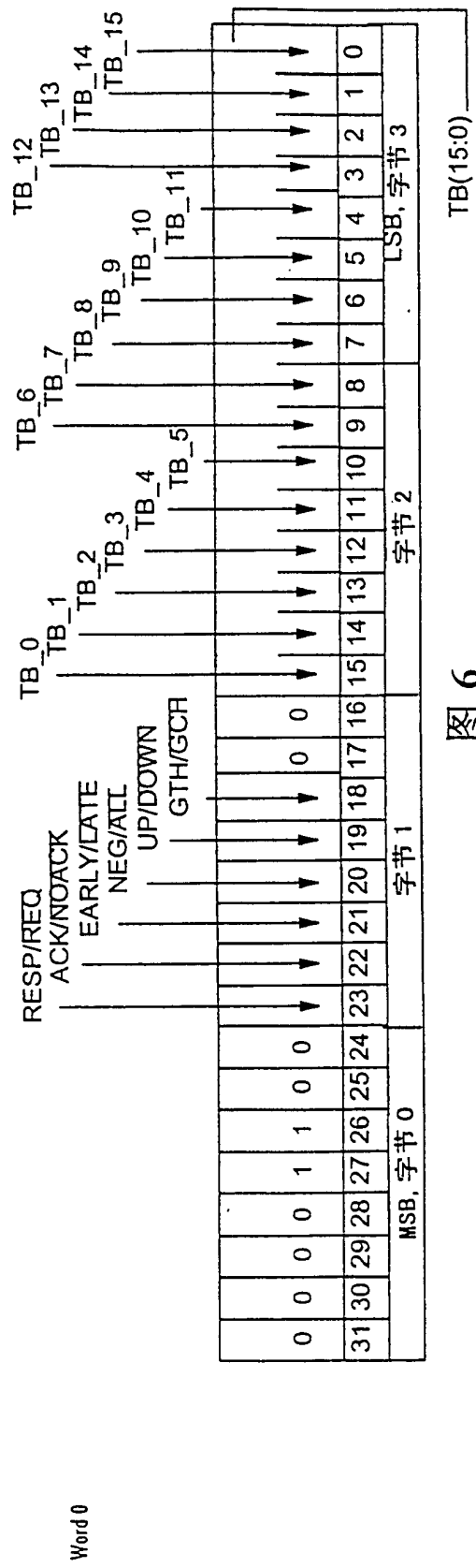


图 4 引导业务流

Word 0	帧控制信息																															
Word 1	相关子																															
Word 2 Cmd Word 0	命令控制信息																															
Word 3 Cmd Word 1	地址信息																															
Word 4 Cmd Word 2	操作对象																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSB, 字节 0								字节 1								字节 2								LSB, 字节 3							

5
[Xen]



9

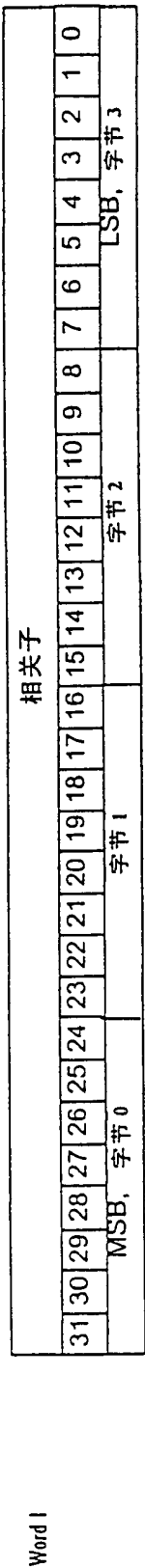


图 7 相关子

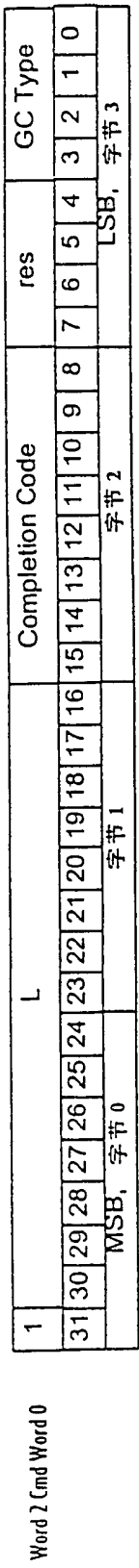


图 8 命令控制信息格式

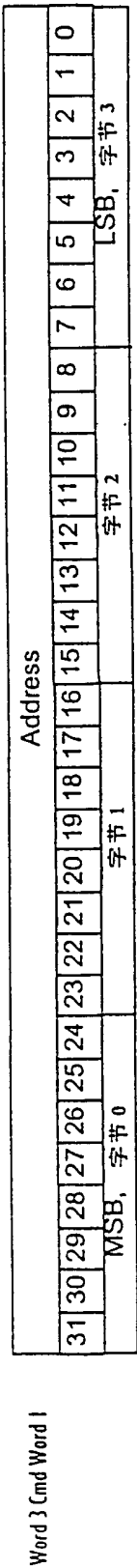


图 9 地址信息格式

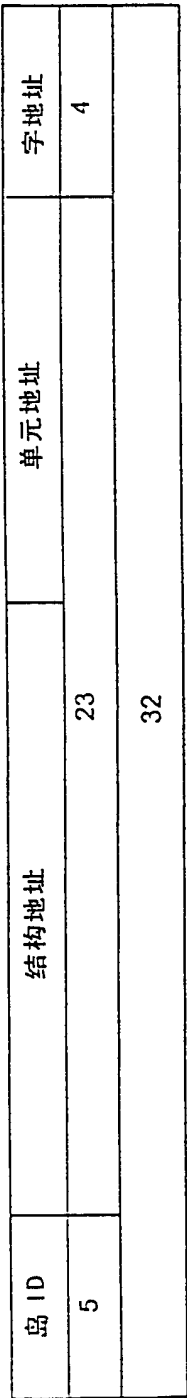


图 10 结构编址的一般格式

Island_ID	岛名称
'00000'b	UpData Store*
'00001'b	Up PMM
'00010'b	Up EDS
'00011'b	Up SDM
'00100'b	嵌入式处理器复合
'00101'b	SPM
'00110-00111'b	保留
'01000'b	控制存储器*
'01001-01111'b	保留
'10000'b	下行数据存储设备*
'10001'b	下行 PMM
'10010'b	下行 EDS
'10011'b	下行 SDM
'10100'b	配置寄存器
'10101'b	DASL
'10110-11111'b	保留
* 这些岛不能通过 Web 接口来访问。	

图 11 编址、岛编码

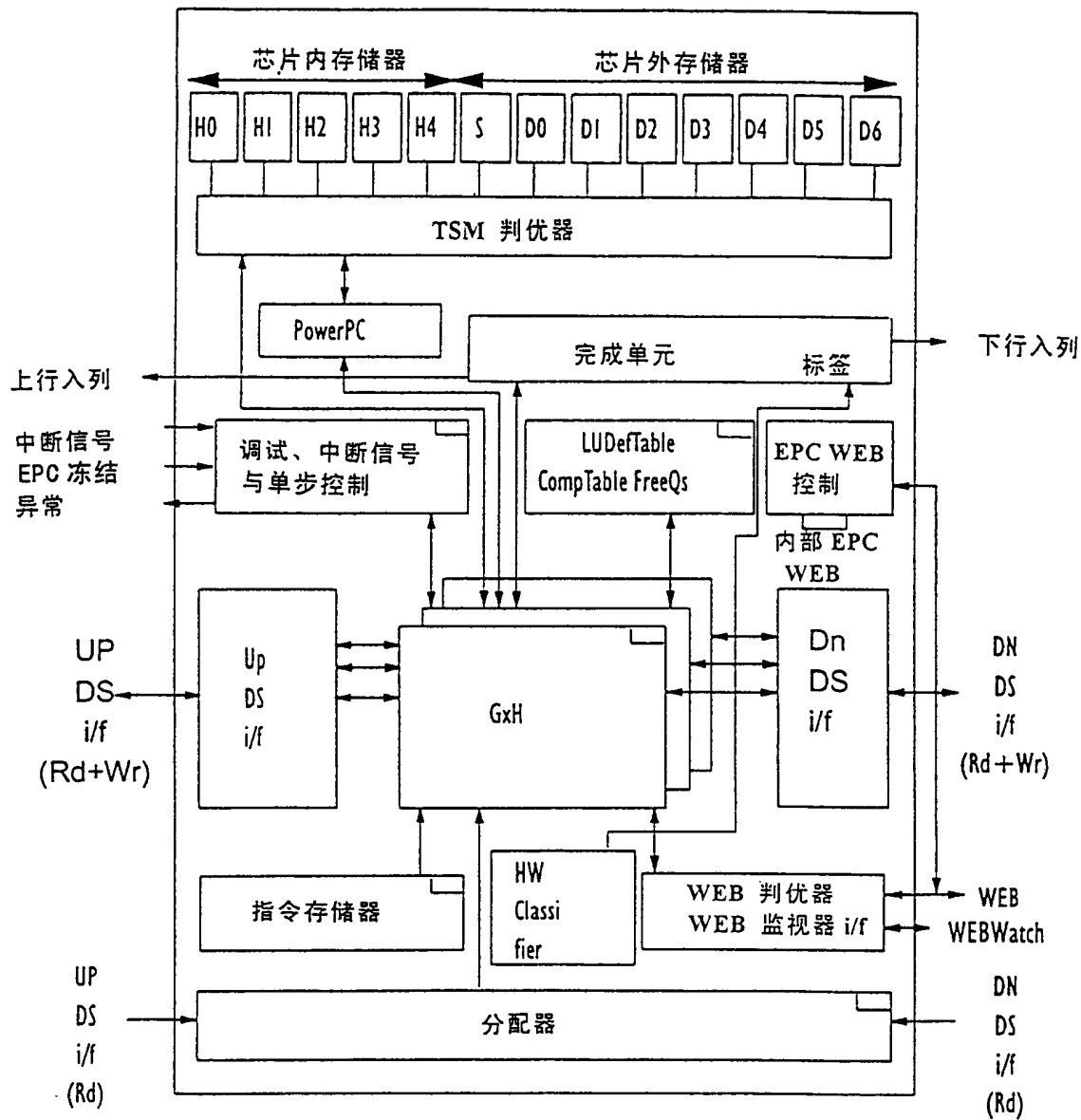


图 12A EPC 方框图

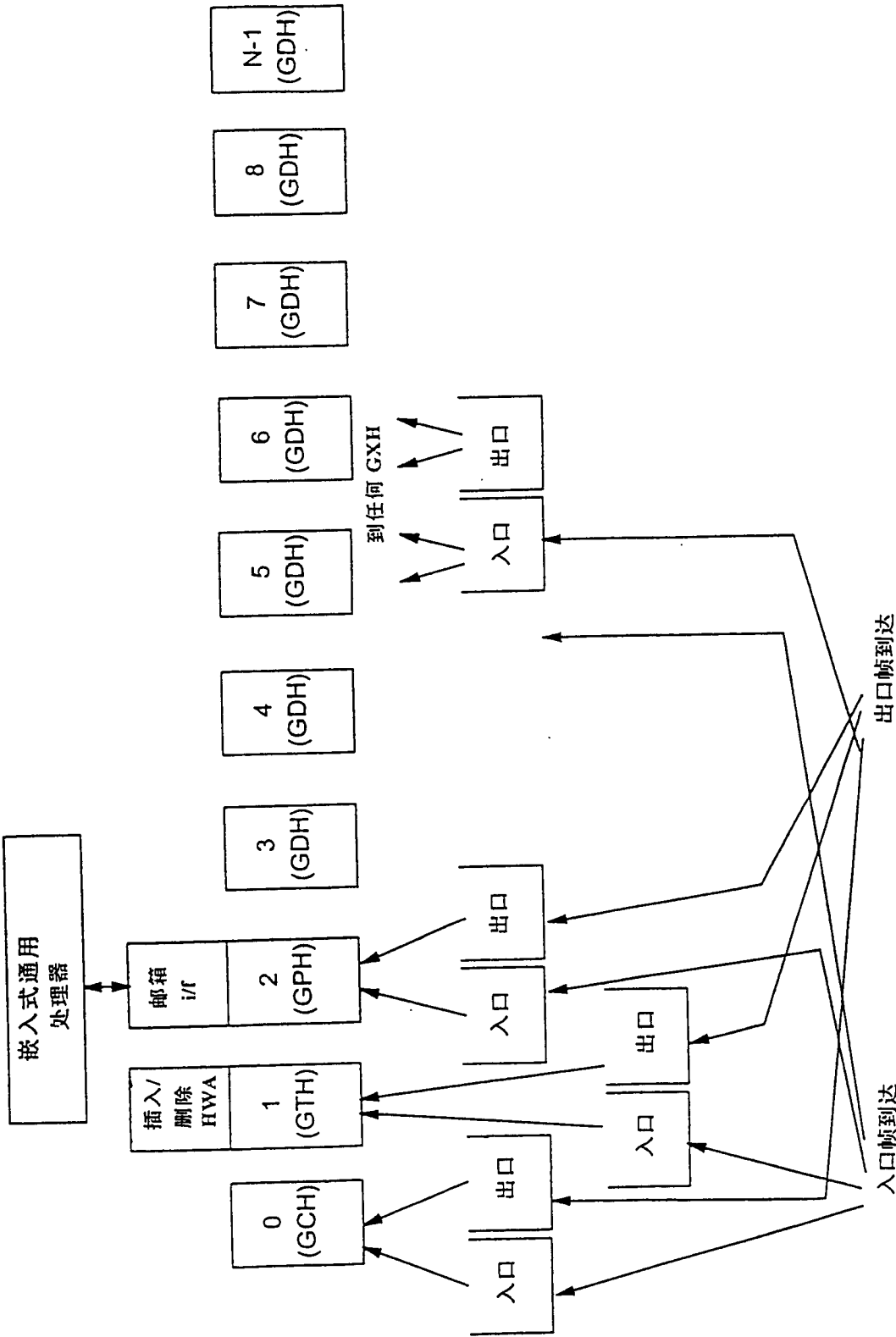


图 12B 10 个 GxHs

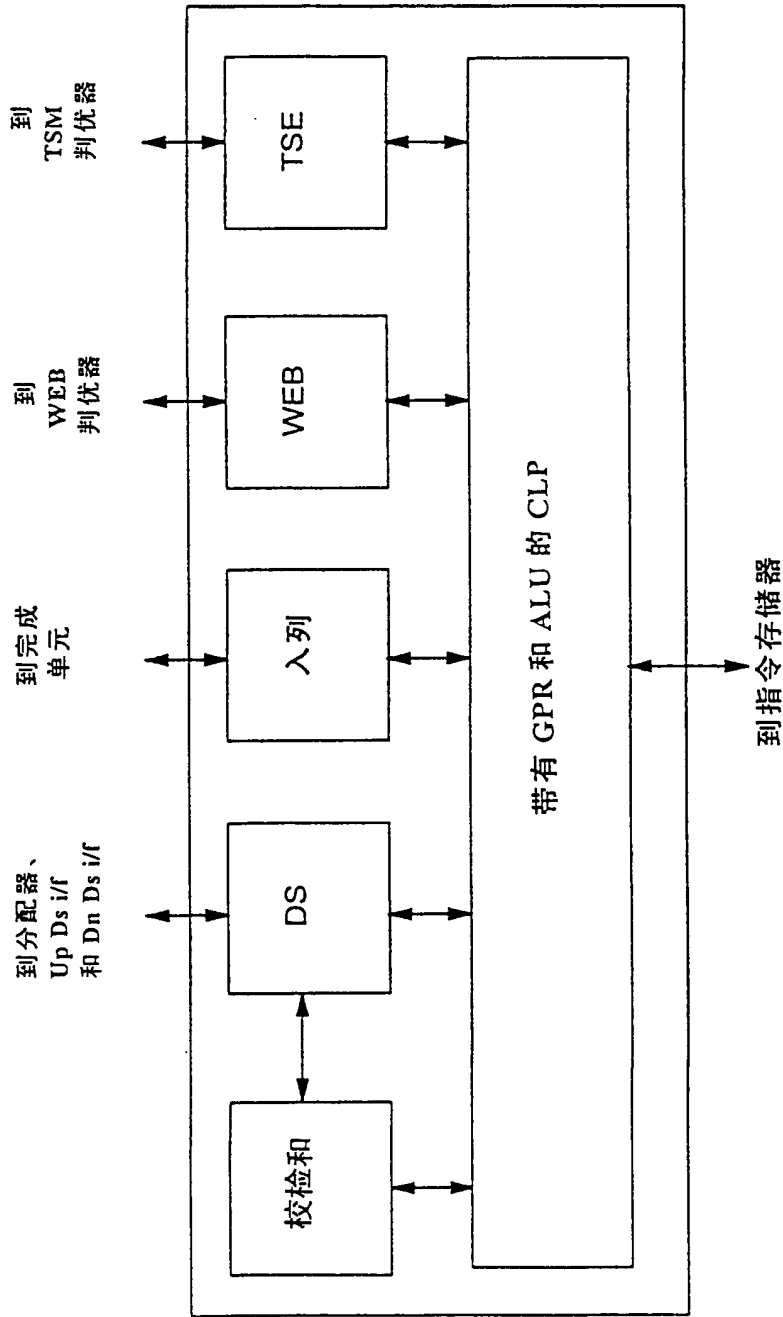


图 12C

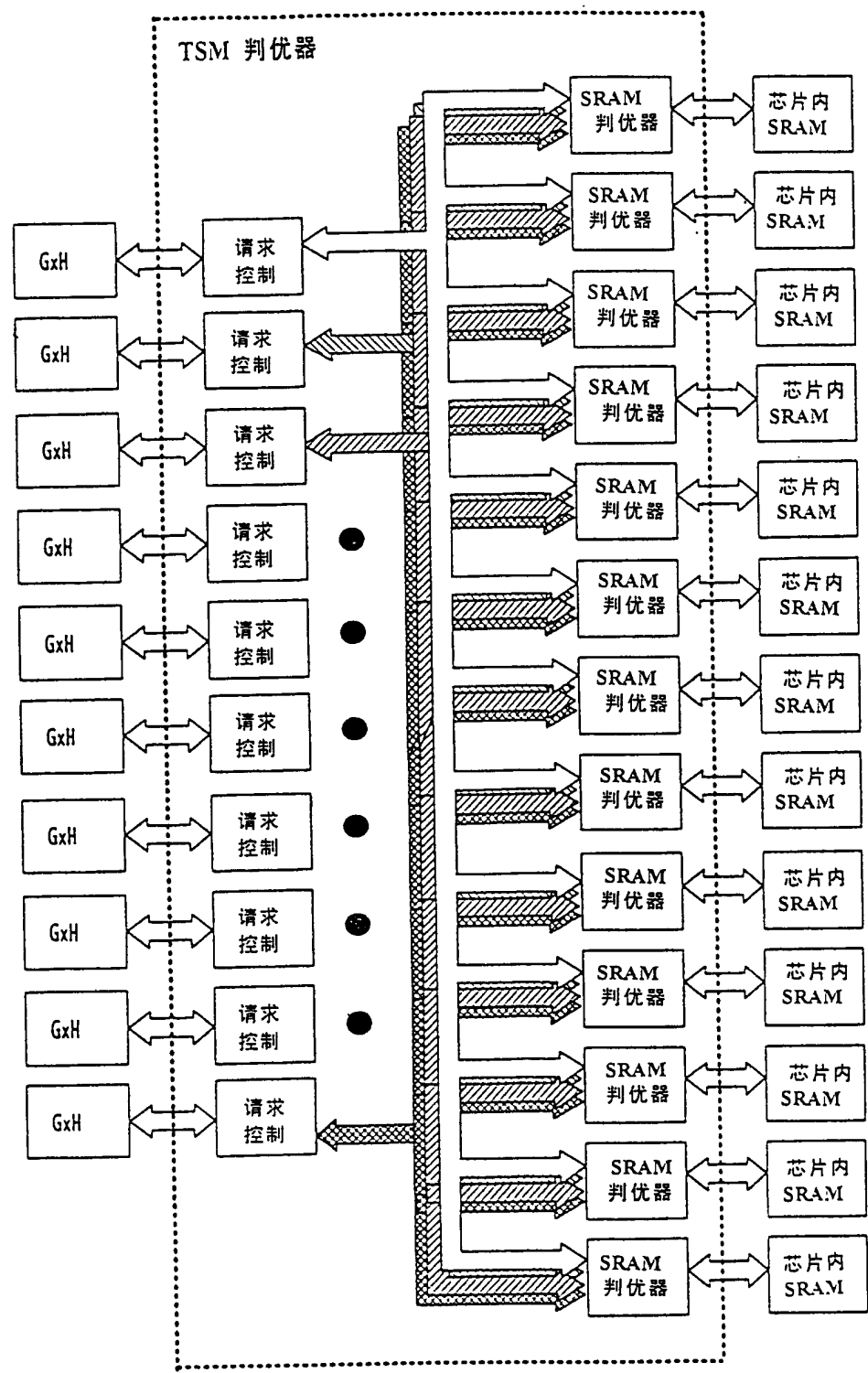
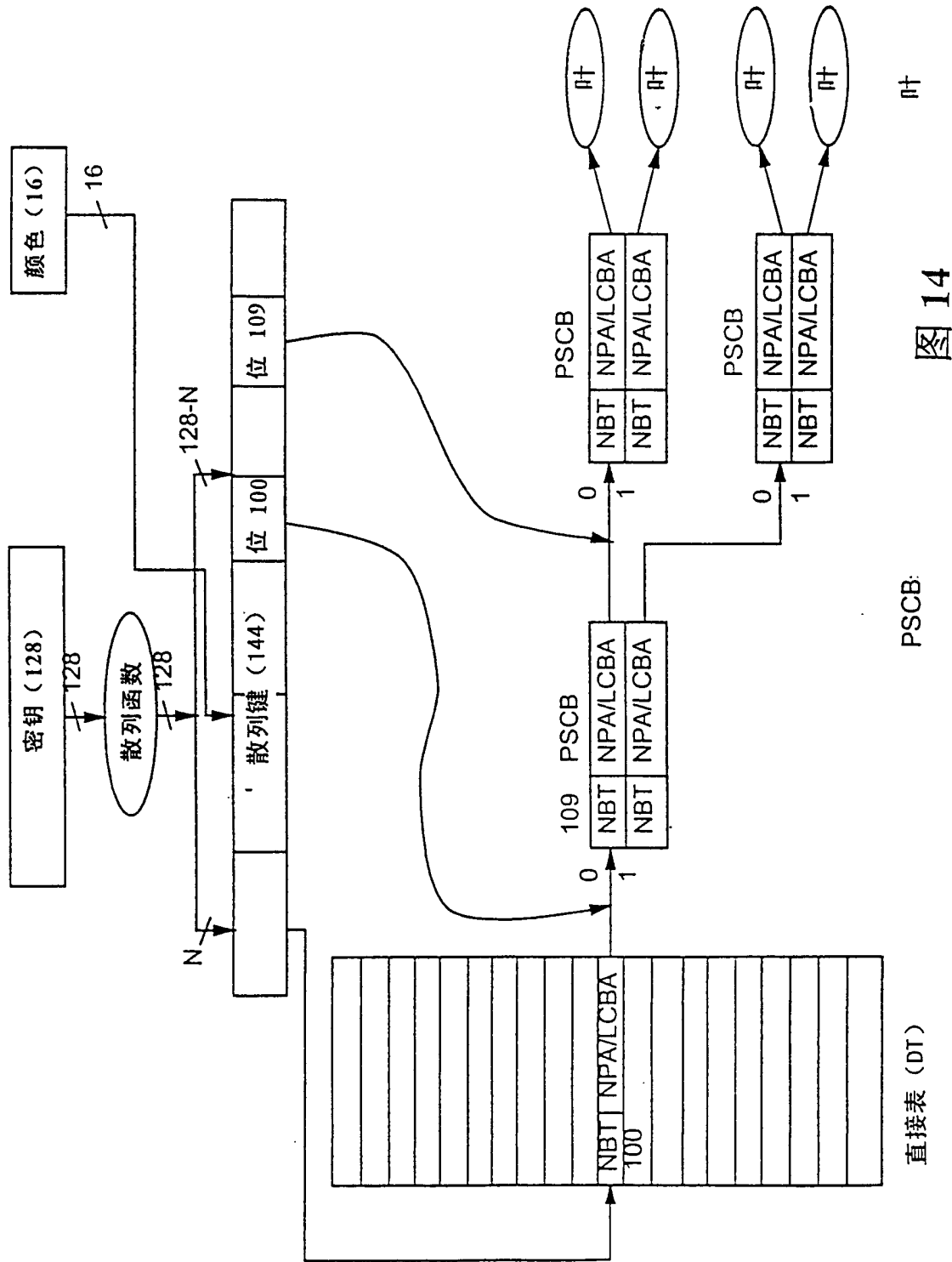


图 13 存储器复合



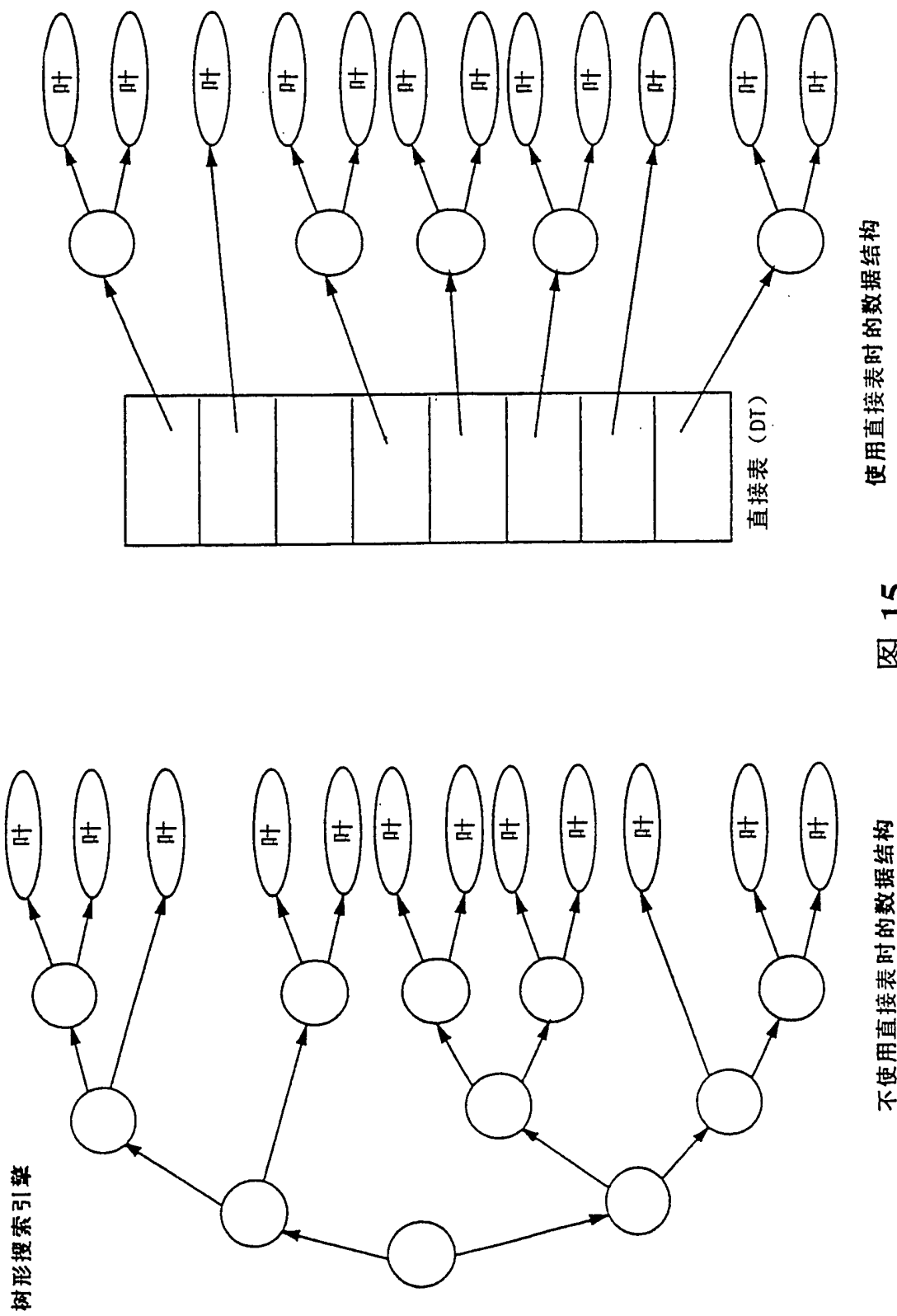


图 15

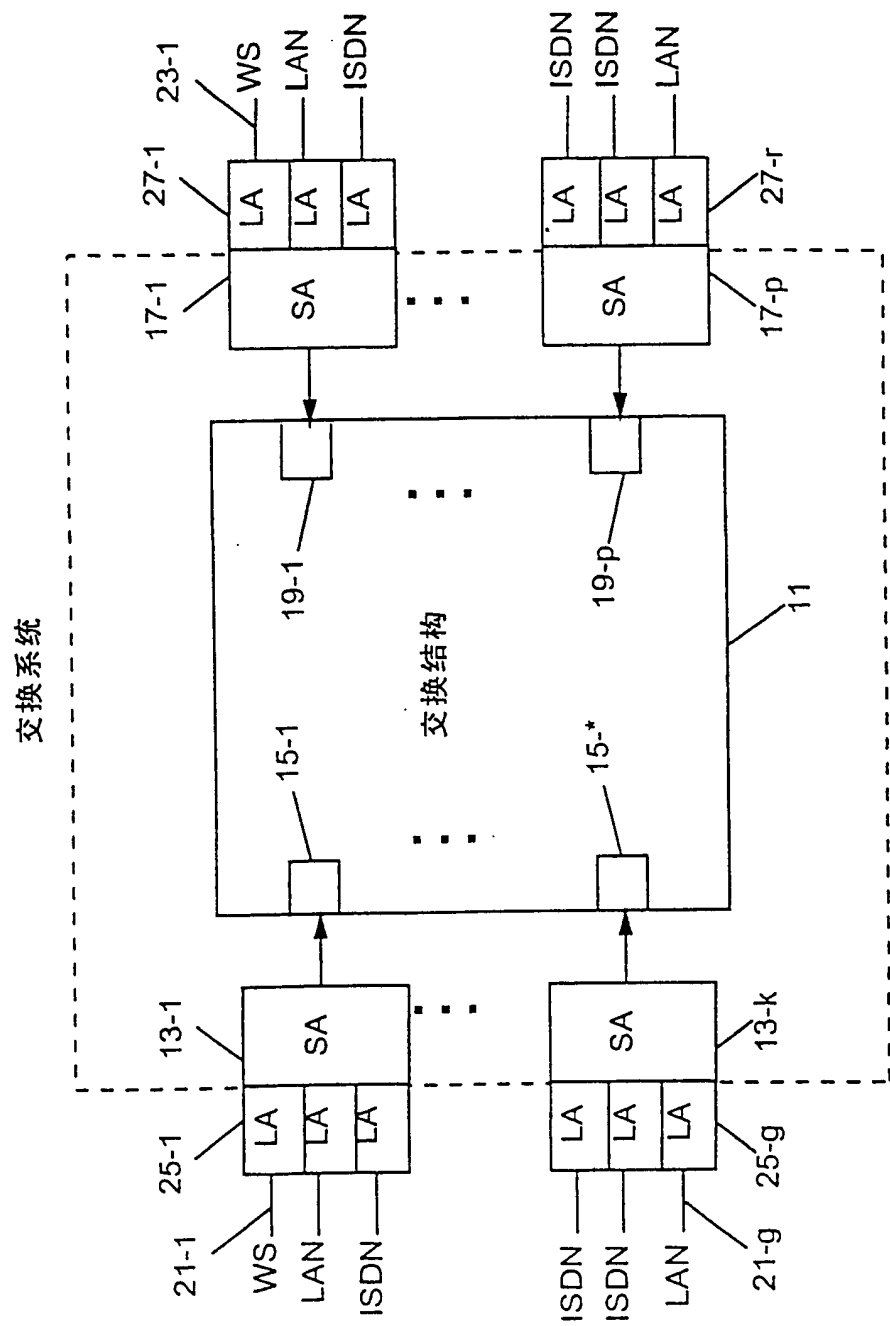


图 16

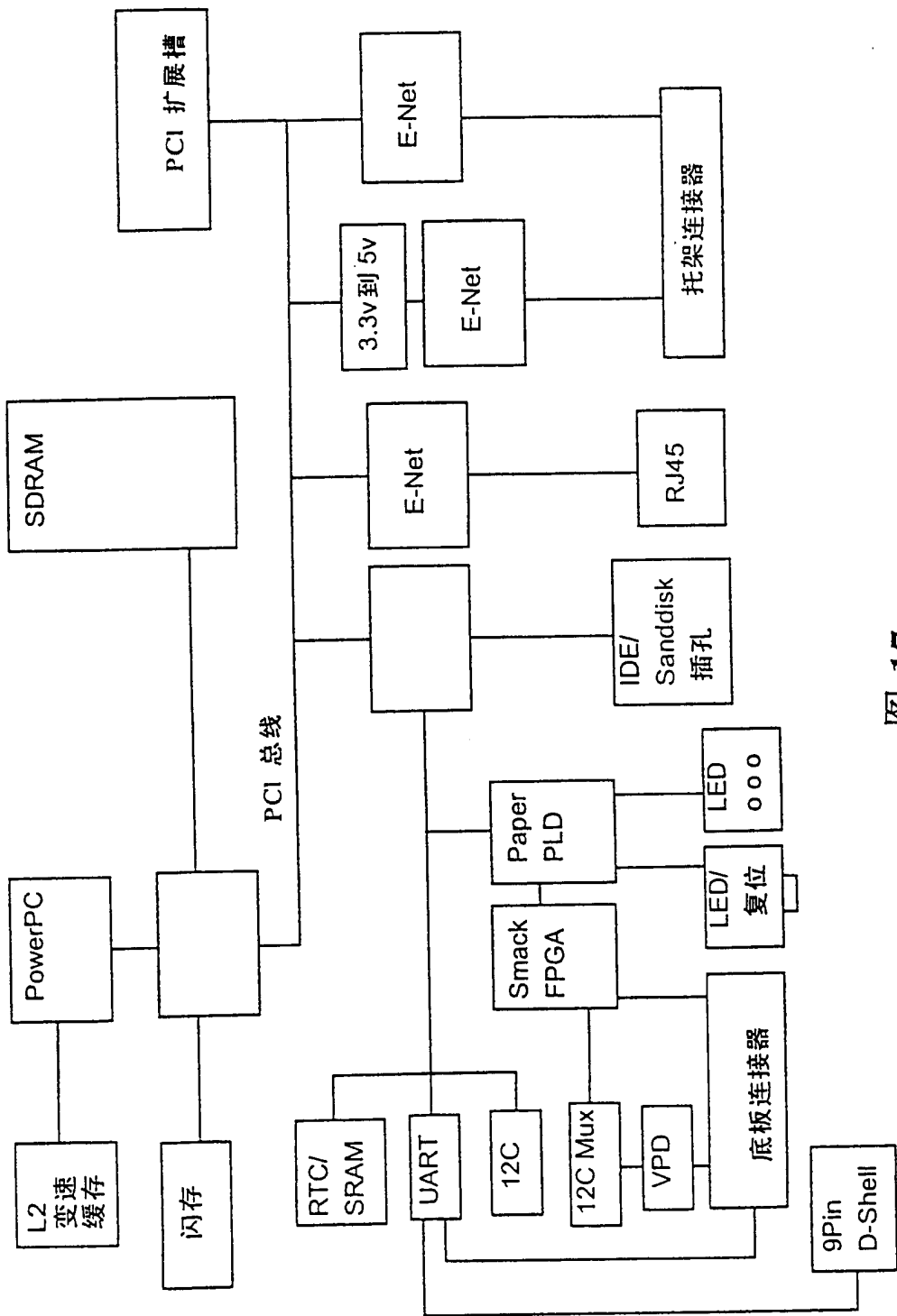


图 17

