



(12) 发明专利申请

(10) 申请公布号 CN 103891150 A

(43) 申请公布日 2014. 06. 25

(21) 申请号 201180074550. 2

(51) Int. Cl.

(22) 申请日 2011. 10. 01

H03M 7/30 (2006. 01)

(85) PCT国际申请进入国家阶段日

H04L 29/10 (2006. 01)

2014. 04. 29

G06F 13/14 (2006. 01)

(86) PCT国际申请的申请数据

PCT/US2011/054487 2011. 10. 01

(87) PCT国际申请的公布数据

W02013/048531 EN 2013. 04. 04

(71) 申请人 英特尔公司

地址 美国加利福尼亚州

(72) 发明人 I · 帕多 I · 索菲尔 D · 瑞弗

D · 达斯莎玛 A · 佩特

(74) 专利代理机构 上海专利商标事务所有限公司 31100

代理人 张东梅

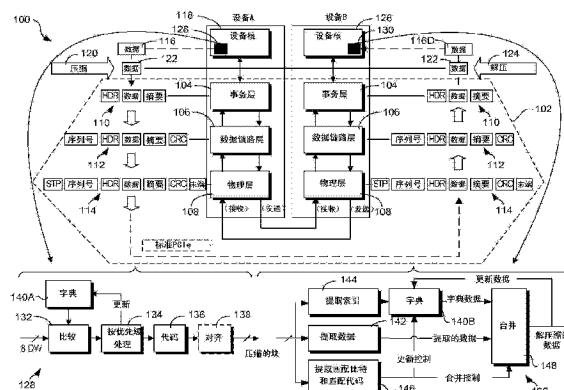
权利要求书2页 说明书11页 附图10页

(54) 发明名称

用于高带宽字典压缩的压缩格式

(57) 摘要

使用基于字典的高带宽无损压缩的方法、设备以及系统。通过压缩器和解压缩器中的逻辑，实现具有被同步和编码以支持压缩和解压缩操作的条目的一对字典。压缩器 / 解压缩器逻辑可以协作方式操作，包括实现相同字典更新方案，导致相应的字典中的数据被同步。字典还配置有可替换的条目，替换策略是基于通过链路传输的数据组内的数据的匹配字节实现的。为条目替换公开了各种方案，以及延迟的字典更新技术。这些技术使用并行操作，支持线速度压缩和解压缩，导致基本上没有等待时间的开销。



1. 一种系统，包括：

发射器；

通过串行链路可通信地耦合到所述发射器的接收器，

在所述发射器中实现的压缩器，所述压缩器被配置成使用基于字典的压缩 / 解压缩方案来从原始数据生成要通过所述串行链路传输到所述接收器的压缩数据，根据所述方案，所述原始数据被处理为多个数据值的块，所述多个数据值的块被并行地编码，以生成包括固定格式部分和可变格式部分的压缩块；以及

在所述接收器中实现的解压缩器，所述解压缩器被配置成使用所述基于字典的压缩 / 解压缩方案来解压缩所述压缩数据，以通过并行地解压缩所述压缩数据，来提取所述原始数据。

2. 如权利要求 1 所述的系统，其特征在于，所述串行链路使用物理层、数据链路层以及事务层，并且其中压缩和解压缩操作是在所述事务层上面的层中执行的。

3. 如权利要求 1 所述的系统，其特征在于，所述压缩器和解压缩器中的每一个都维持用于压缩和解压缩操作的具有以同步方式更新的可替换的条目的字典的相应的副本。

4. 如权利要求 3 所述的系统，其特征在于，所述压缩器被配置成利用原始数据的对应的部分替换字典未命中上的字典条目，并生成以标识所述被替换的字典条目和原始数据的所述对应的部分的方式编码的压缩数据，并且其中所述解压缩器被配置成解码所述压缩数据，并利用原始数据的所述对应的部分来替换所述字典的其副本中的相同字典条目。

5. 如权利要求 4 所述的系统，其特征在于，所述压缩器和解压缩器中的每一个都被配置成并行地替换多个字典条目。

6. 如权利要求 1 所述的系统，其特征在于，所述串行链路包括 PCI Express 链路。

7. 如权利要求 1 所述的系统，其特征在于，所述原始分组被并行地编码为包括所述压缩数据的压缩块的多个双字。

8. 如权利要求 7 所述的系统，其特征在于，所述解压缩器被配置成并行地提取所述多个双字。

9. 如权利要求 1 所述的方法，其特征在于，压缩 / 解压缩方案并行地处理 256 比特的数据。

10. 一种方法，包括：

将原始数据压缩为根据基于字典的压缩 / 解压缩方案编码的压缩数据；以及  
解压缩所述压缩数据以提取所述原始数据，

其中所述原始数据包括包含多个字的多个块，且压缩和解压缩所述原始数据是通过按块地对所述多个字执行并行操作来实现的，以生成包括固定格式部分和可变格式部分的压缩块。

11. 如权利要求 10 所述的方法，还包括：

在被配置成将所述原始数据压缩为压缩数据并传输要由接收器通过串行链路接收的所述压缩数据的发射器上维持与压缩操作一起使用的第一字典；以及

在所述接收器上维持第二字典，所述第二字典与解压缩操作一起使用，

其中所述第一和第二字典中的每一个都包括可替换的条目，其中条目以同步方式在所述第一和第二字典中被替换。

12. 如权利要求 11 所述的方法,还包括:

并行地替换所述第一字典中的多个条目;

通过所述串行链路,向所述接收器传输包含标识所述第一字典中的被替换的所述条目的信息的编码数据;以及

并行地替换所述第二字典中的相同多个条目。

13. 如权利要求 10 所述的方法,还包括并行地将包括所述原始数据的多个字编码为包括所述压缩数据的压缩块。

14. 如权利要求 10 所述的方法,还包括:

通过具有线速率的串行链路,传输所述压缩数据;以及

实时地执行所述压缩和解压缩操作,以便以线速率支持和平均化所述原始数据的传输速率。

15. 如权利要求 14 所述的方法,其特征在于,所述串行链路包括 PCIExpress 链路。

16. 一种设备,包括:

具有带有多个可替换的条目的字典并具有被配置成通过使用基于字典的压缩 / 解压缩方案,使用所述字典,将原始数据压缩为压缩数据的逻辑的压缩器,其中所述原始数据包括被并行地按块地压缩以生成包括固定格式部分和可变格式部分的数据的压缩块的多个双字。

17. 如权利要求 16 所述的设备,其特征在于,压缩数据被编码为标识所述多个双字中的每一个的字典匹配或未命中条件。

18. 如权利要求 16 所述的设备,其特征在于,所述多个字包括八个 32 比特双字。

19. 如权利要求 16 所述的设备,其特征在于,所述压缩器被配置为:

对于所述多个双字中的每一个;

将字数据与所述字典中的字典条目中的数据进行比较以确定字典匹配或未命中条件;

对于匹配条件,编码所述压缩数据以标识匹配条件和对应的字典条目;以及

对于未命中条件,编码所述压缩数据以标识未命中条件,并在所述压缩数据中包括所述双字数据的副本。

20. 如权利要求 19 所述的设备,其特征在于,按字节地将所述双字数据与所述字典条目中的数据进行比较,且匹配条件对应于匹配字典条目中的对应的字节的所述双字数据的至少两个字节。

## 用于高带宽字典压缩的压缩格式

### 技术领域

[0001] 发明的领域一般涉及计算机系统中的高带宽压缩,但不排他地涉及用于基于字典的高带宽无损压缩的技术。

### [0002] 背景信息

[0003] 计算机系统通常使用一个或多个互连来促进系统各组件间的通信,诸如在处理器和存储器之间。也可以使用互连和 / 或扩展接口来支持内嵌的和附加设备,诸如 I/O(输入 / 输出)设备和扩展卡等等。许多年来,在引入个人计算机之后,互连的主要形式是并行总线。并行总线结构用于内部数据传输和扩展总线,诸如 ISA(工业标准体系结构)、MCA(微通道体系结构)、EISA(扩展工业标准结构)和 VESA 局部总线。在 1990 年代初,Intel 公司引入了 PCI(外围组件互连)计算机总线。PCI 通过不仅提高总线速度,而且还使用共享地址和数据线,引入了自动配置和基于事务的数据传输,改进了以前的总线技术。

[0004] 随着时间的推移,计算机处理器时钟速率以比并行总线时钟速率更快的速度增大。结果,计算机工作负荷常常受到互连瓶颈而并非处理器速度的限制。虽然并行总线支持利用每一个周期传输大量的数据(例如,根据 PCI-X,32 或者甚至 64 比特),但是,它们的时钟速率受采样歪斜考虑的限制,导致对最大总线速度造成实际限制。为克服此问题,开发了高速串行互连。早期的串行互连的示例包括串行 ATA、USB(通用串行总线)、FireWire,以及 RapidIO。

[0005] 另一广泛地使用的标准串行互连是 PCI Express,也叫做 PCIe,这是在 2004 年根据 PCIe1.0 标准中推出的。PCIe 被设计成替换旧的 PCI 和 PCI-X 标准,同时提供旧式支持。PCIe 使用点对点串行链路,而非共享的并行总线体系结构。每一个链路都使用一个或多个通道,来支持两个 PCIe 端口之间的点对点通信信道,每一个通道都包括双向串行链路。通道在物理上使用纵横开关体系结构来路由,该体系结构支持同时在多个设备之间的通信。由于其固有的优点,在当今的个人计算机中,PCIe 取代了 PCI 而作为最流行的互连。PCIe 是由 PCI-SIG(特殊兴趣组)管理的行业标准。如此,许多 ASIC 和硅供应商提供 PCIe 簿(pad)。

[0006] 根据摩尔定律,处理器和存储器继续发展,虽然最近的处理器速度提高主要基于具有多个核而并非基于时钟速率的提高。然而,互连的速率,特别是诸如 PCIe 之类的串行链路的速率未能够赶上。这鉴于当前技术,部分地由于对时钟速率的有限的限制。相应地,并非聚焦于基于较高时钟速率来提高速度,最近引入或提出了其他方案。例如,PCIe3.0 规范通过使用提高的时钟速率和从 PCIe 的标准 8b/10b 编码切换到 8 比特编码(对于每 8 比特的可使用的数据,10 比特的编码数据)的组合,使 PCIe2.x 互连带宽翻倍。

[0007] 用于提高有效链路带宽的另一种方法是使用数据压缩。有多个用于各种类型的数据传输的可行的数据压缩方案。然而,与这些方案中的许多相关联的开销和计算要求使它们用于诸如 PCIe 之类的高速串行互连不切实际。例如,为了有益,平均速度改进增益必须大于由于压缩 / 解压缩操作导致的平均开销增大(就传输等待时间而言)。由于 PCIe 协议有效负载大小限制只是 4KB(对于实际实现的大多数分组有效负载通常仅限于 256 字节

(对于服务器芯片集) 和 128 字节(对于客户端芯片集)), 如果在 PCIe 链路上使用常规压缩编码技术, 则一般没有好处(事实上, 通常会有损害)。相应地, 对高速互连实现高带宽无损压缩 / 解压缩方案, 没有由于处理开销导致的传输等待时间或传输等待时间最少, 将是有利的。

[0008] 附图简述

[0009] 通过参考与附图一起进行的下面的详细描述, 本发明的前述的方面和许多伴随的优点, 将变得更加轻松地被理解, 其中, 在各个视图中, 相同参考编号表示相同部件, 除非另作说明:

[0010] 图 1 示出了根据一实施例的实现为对标准化的 PCIe 体系结构的增强的使用基于字典的压缩和解压缩的高带宽无损压缩数据传输体系结构的总览;

[0011] 图 2 示出了对应于基于字典的压缩 / 解压缩实现的一个实施例的有选择性的操作以及逻辑的细节;

[0012] 图 3 示出了根据一实施例的示例性压缩块编码方案的细节;

[0013] 图 4 示出了根据一实施例的说明用于基于字典的压缩 / 解压缩实现中的示例性编码组的表;

[0014] 图 5 是示出了根据一实施例的与字典匹配相关联的操作的组合方框示意图和流程图;

[0015] 图 6 是示出了根据一实施例的与字典未命中 (miss) 相关联的操作的组合方框示意图和流程图;

[0016] 图 7 是示出了与对 8 个双字压缩块的处理相关联的并行操作的组合方框示意图和流程图;

[0017] 图 8a 和 8b 是示出了根据一实施例的与延迟的字典更新一起执行的操作的组合方框示意图和流程图;

[0018] 图 8c 是示出了在发射器和接收器处使用阴影字典副本所依据的延迟的字典更新的组合方框示意图和流程图。

## 具体实施方式

[0019] 此处描述了使用高带宽基于字典的无损压缩的方法、设备, 以及系统的实施例。在下面的描述中, 阐述了很多具体细节(诸如使用 PCIe 的示例性实现), 以便全面地理解本发明的各实施例。然而, 那些精通相关技术的人将认识到, 本发明可以在没有一个或多个具体细节的情况下实现, 或利用其他方法、组件、材料等等来实现。在其他情况下, 没有显示或详细描述已知的结构、材料、或操作, 以避免模糊本发明的某些方面。

[0020] 说明书中对“一个实施例”、“实施例”的引用意味着结合该实施例所描述的特定特征、结构或特性被包括在本发明的至少一个实施例中。如此, 在整个说明书中的不同位置出现短语“在一个实施例中”或“在实施例中”不一定都是指同一个实施例。此外, 在一个或多个实施例中, 特定特征、结构或特性可以以任何合适的方式组合起来。

[0021] 根据此处所公开的各实施例的各方面, 提供了使用新颖的基于字典的压缩 / 解压缩方案支持高带宽无损压缩的技术。技术可以一般性地以诸如高速串行链路之类的链路和互连, 以及支持高带宽无损失数据压缩的实现来实现。此外, 传输带宽改进是在没有对基础

链路物理传输的修改的情况下实现的。

[0022] 图 1 示出了根据一实施例的实现为对标准化的 PCIe 体系结构的增强的高带宽无损压缩数据传输体系结构 100 的总览。通过虚线框 102 描绘了描绘用于实现标准 PCIe 方面的框和流程逻辑的图示部分,且操作以及逻辑对应于支持包括图示的其余部分的压缩和解压缩操作的增强。

[0023] 根据 PCIe, 数据以分组化的形式在两个 PCIe 端点或端口之间传输,如由设备 A, 以及设备 B 所描绘的。PCIe 链路包括单或多通道点对点互连。此外,点对点互连是在相对的方向以单向的点对点互连对配置的,以便每一个链路对都支持双向通信。传输起源于发送方或发射器,并由接收器接收。正在被发送的数据通过传输 (Tx) 路径传输,对应的数据通过接收 (Rx) 路径接收。在 PCIe 设备之间交换的分组的两个主要类别是高级事务层分组 (TLP), 以及低级链路维护分组,叫做数据链路层分组 (DLLP)。总起来说,各种 TLP 和 DLLP 使两个 PCIe 设备能可靠地执行存储器、IO, 以及配置空间事务,并使用消息来发起电源管理事件、生成中断,报告错误等等。

[0024] PCIe 标准的三个较低层包括事务层、数据链路层以及物理 (PHY) 层。相应地,设备 A 和 B 中的每一个都被描绘成包括事务层 (框) 104、数据链路层 (框) 106 以及物理层 (框) 108。起源设备 (例如,此示例中的设备 A),生成要向接收方设备 (例如,设备 B) 发送的数据,该数据然后被 PCIe 层分组化,并通过链路传输。在 PCIe 事务的上下文中,起源方被称为请求者,接收方被称为完成方。

[0025] 在事务层,数据被分组为一个或多个分组,其具有分组格式 110,包括标头 (HDR)、分组有效负载数据 (数据),以及摘要 (Digest)。在数据链路层,将序列号 (SeqNum) 添加到分组的开始处,与附加到末端的 CRC 一起,形成分组格式 112。分组格式在物理层通过添加 STP 和末端控制 (K) 字符以形成分组格式 114,来进一步增强。然后,分组通过链路发送以便由接收器 (设备 B) 使用逆序列处理,最终得到原始数据。

[0026] 根据此处所公开的各实施例的各方面,向事务层上面的分组有效负载数据应用压缩和解压缩方案。在一个实施例中,在开放系统互连参考模型 (OSI-RM) 的层 6 (表示层) 中执行压缩和解压缩操作。通过使压缩和解压缩操作在事务层上执行,没有对包括事务层以及其下面的任何层的增强。如此,压缩 / 解压缩方案可以使用标准 PCIe 库和簿来实现。此外,类似的实现可以与其他类型的串行链路一起使用,而不要求对基础链路结构的修改。

[0027] 在图 1 中,数据 116 包括起源于设备 A 的设备核 118 的原始 (非压缩的) 数据。然后,原始数据被压缩,如由压缩操作 120 和压缩数据 122 所描绘的。然后,由设备 A 和 B 的 PCIe 工具过适用的 PCIe 链路,应用常规分组处理和传输操作。在设备 B 的事务层框 104 的常规处理时,最初以其压缩形式 (压缩数据 122) 提取分组有效负载数据,然后,解压缩,如由解压缩操作 124 所描绘的,以产生原始数据 116 的副本,然后,由设备 B 的设备核 126 接收。

[0028] 压缩和解压缩操作是使用基于字典的编码和解码方案来实现的,该方案是通过相应的压缩器和解压缩器组件使用设备核 118 和 126 中的压缩器逻辑 128 和解压缩器逻辑 130 来实现的。在图 1 底部描绘了压缩器逻辑 128 和解压缩器逻辑 130 的块级别的细节。如图所示,压缩器逻辑 128 包括比较块 132、优先级块 134、代码块 136、对齐块 138 (可以可任选地使用,取决于实现) 以及字典 140A。同时,解压缩器逻辑 130 包括字典 140B、提取数

据块 142、提取索引块 144、提取匹配位和匹配代码块 146 以及合并块 148。如通过压缩器 / 解压缩器逻辑和下面的相关联的操作的实现的进一步的细节变得显而易见的,由压缩器 / 解压缩器逻辑同步地对相关的数据执行互补操作。通过使用此方案,减少了需要被添加以实行编码的数据的量。此外,还可以并行地执行操作,使压缩 / 解压缩操作能以链路速度执行,就数据传输等待时间而言,基本上没有开销。

[0029] 在此处的各实施例中,实现了基于字典的方案,用于通过使用编码 / 解码的数据格式来执行压缩和解压缩。虽然以前已经使用了基于字典的方法,但是,它们通常会带来大量的等待时间(对于带有较大的词汇表的字典),或者它们支持有限的线速度带宽,该带宽小于当今的高速互连所需的带宽。另外,这样的现有的方案一次仅处理数据的单一部分(例如,单字或双字)。相比之下,此处的各实施例通过使用并行操作,支持高带宽数线速度,且带有低等待时间。

[0030] 在图 2-7 以及 8a-c 中,示出了涉及新颖的字典配置、编码 / 解码逻辑以及字典替换策略的使用的压缩器 / 解压缩器逻辑的各方面。例如,图 2 示出了与使用基于字典的压缩和解压缩(在接收器一侧)的事务的发射器一侧的压缩操作结合的字典格式和编码的示例。当原始数据作为比特模式输入流 200 被接收时,它以对齐的双字(DW)流 202 分割为数据的相等部分,在参考编号 204 处示出了每一个双字输入的处理的细节。每一个双字都包括 4 个字节,从 LSB(最低有效字节)到 MSB(最高有效字节)编号,字节 0、字节 1、字节 2 以及字节 3,每一个字节都包括 8 个比特,总共 32 个比特。在一个实施例中,使用具有 16 通道的宽度的 PCIe 链路,对于每一个 PCIe 时钟周期,使 16 比特的数据能并行地传输。可任选地,对于对应的分组处理,可以使用较大的或较小数量的通道,如那些精通本技术的普通人员所知的。此外,还可以按类似的方式实现其他类型的串行传输链路或互连。

[0031] 图 2 进一步示出了包括 16 行(即字典条目)32 比特数据的字典 140(逻辑地被分成四个字节部分)的一个实施例。在接收到每一个双字时,逐行地将 DW 的数据内容与字典 140 中的对应的字节数据进行比较,以确定匹配 / 非匹配条件。即,对于给定行,比较字节 0 的值,比较字节 1 的值,比较字节 2 的值,并比较字节 3 的值。比较操作的结果被编码为匹配比特 206 和一组 8 个类型 / 数据比特 208。匹配比特 206 指出是否存在匹配。在一个实施例中,如果两个或更多字节值匹配,则存在匹配。值“1”标识匹配,而值“0”标识非匹配或未命中。如果存在匹配,则头四个比特(从左到右)[7:4]被编码为对应于按字节的比较结果的匹配模式。同时,第二四个比特[3:0]被编码为字典 140 中的匹配行的索引(即,地址)。相应地,在所示出的示例中,匹配模式是“0111”,且匹配行索引是对应于第一行的“0000”。如果比较操作产生非匹配结果(即,字典未命中),则匹配比特 206 被编码为“0”,而类型 / 数据比特 208 被简单地编码为 DW 数据的第一字节(LSB 或字节 0)。在图 3 以及图 4 的表 400 中示出了对于类型 / 数据比特 208 的结构和编码逻辑的进一步的细节。

[0032] 图 3 示出了根据一实施例的示例性压缩块编码方案的细节。此处所公开的各实施例的一个方面是并行地处理数据的多个部分的能力,由此显著地缩短与压缩 / 解压缩操作相关联的延迟。例如,图 3 中所示出的实施例示出了用于并行地处理 8 个双字的编码方案。数据流(例如,比特模式输入流 200)中的原始数据被分割为对齐的双字流 202,8 个 DW 被并行地压缩 / 解压缩。8 个 DW 的编码的形式被描绘成 8DW 压缩块 300。

[0033] 8DW 压缩块 300 的第一字节占用压缩块的第一槽,并包括 8 个双字的类型 / 数据

比特 208,每一个 DW 的一个比特都指出是否有匹配 (“1”) 或未命中 (“0”)。接下来八个槽包括相应的类型 / 数据比特 208 条目,每一个 DW,一个字节。在一个实施例中,用于编码类型 / 数据比特 208 的规则存储在包括数据结构等等的表中,诸如由图 4 的表 400 例示的。响应于匹配判断,标识表中的对应的行,使用由行数据所定义的编码格式来编码将要被编码的八个比特。在字典匹配的情况下,头四个比特将被编码为匹配模式,接下来的四个比特将包括字典索引。如果有未命中,则 8 个比特将对应于 DW 的 LSB(字节 0)。下面讨论了还可以适用于可替换的字典条目的未命中的额外的可能的编码。

[0034] 8DW 压缩数据块 300 的其余部分包括用于存储每一个 DW 的 0-3 字节的可变长度槽,用于给定 DW 和槽的字节的数量取决于匹配条件。例如,对于两个字节的匹配,对应的数据 n(0-7) 槽将包括不匹配的两个字节。如果有三个字节的匹配,则数据 n 数据将包括其余非匹配字节。对于四个字节的匹配,Data n 数据将是空值 (无数据)。同时,对于未命中,数据 n 数据将包括字节 1、字节 2 以及字节 3。相应地,与字节 0 数据相结合 (在对应的类型 / 数据比特 208 槽中),DW 的完全数据具有在 8DW 压缩数据块 300 中的编码的未命中,作为未经压缩的数据 (有效地)。

[0035] 图 5 是示出了对应于字典匹配的操作的组合方框示意图 / 逻辑流程图。值得注意的是,此示例和图 6 中的示例对应于对单 DW 的处理,与八个 DW 相关联的并行处理通过使用如图 7 所示的 8DW 压缩块。此时假设,字典 140 具有整套十六个 32 比特条目,但是,为简单起见,只示出了填充了数据的顶部三行。

[0036] 为说明,图 5 和 6 描绘了发射器一侧的 DW 索引 500A 以及接收器一侧的 DW 索引 500B。为更好地理解技术,使用 DW 索引来指定哪些单个 DW 在对应的图形中正在被处理。在与并行地处理八个双字相关联的一个实施例中,通过 8DW 压缩块 300 以及对应的编程逻辑等等的结构,实现 DW 索引,以便没有被使用的单独的索引。

[0037] 对于图 5 中的示例,DW 索引 500A 和 500B 具有值 0,表示原始数据 116 (“A003232C”) 对应于 8DW 压缩块 300 中的数据 0。如以前一样,逐字节地将每一个双字中的数据与字典 140 中的对应的条目进行比较,查找最佳匹配,这是利用带有最多按字节匹配的行定义的 (如果有的话)。在对于最佳匹配的区块 (tie) 的情况下,发射器可以选择匹配的 (或部分地匹配的) 条目中的任何一个作为所选一个。在一个实施例中,在并行支持按字节的比较操作的相应的寄存器中编码字典条目。相应地,可以在一个时钟周期或固定数量的时钟周期在整个字典中搜索匹配,不管字典大小如何。

[0038] 如上文所讨论的,同步地维持链路端点的字典中的数据 (当更新条目时,发送方和接收器字典之间的延迟非常小)。这是通过在发射器和接收器处使用相关联的互补操作来实现的,以便对发射器字典的任何更新都也应用于接收器字典。结果,预先已知,任何匹配的行在相对的端点处的同步的字典中的相同的行中将具有相同数据值。如此,并非发送原始数据,而是编码对于匹配的传输的数据,以便标识对于该匹配的行,以及匹配模式。然后,使用匹配模式来标识哪些字节是匹配的,哪些字节不是匹配的。基于根据图 4 中的表 400 的匹配模式和编程的逻辑,解压缩器逻辑知道如何编码数据,如此,知道如何提取相关数据字节以重组原始数据。

[0039] 返回到图 5,该图示出了原始数据 116 与具有地址或索引“0”的第一行 (“0000”) 中的数据的最佳匹配。值得注意的是,这里使用匹配第一行只是为方便,并且不使图形模

糊,因为任何一个行都可能会产生最佳匹配。在标识带有最佳匹配的行时,生成 8DW 压缩块中的对应的条目 300-0 数据,从设置为“1”的匹配比特 206 起,接下来是包括匹配模式(“0111”)和行地址(“0000”的类型 / 数据比特 208。

[0040] 基于匹配模式“0111”,压缩器 / 解压缩器逻辑使用基于表 400 中的对应的行的数据编码方案。对应于第三行的匹配条目在表 400 中是粗体,表示此行包含对于匹配模式“0111”(示为“ $xmmm$ ”,其中,“ $x$ ”代表未命中,“ $m$ ”代表匹配)的适用的逻辑。表逻辑表示,数据字节 3,对于数据 n 字节,将被编码(参见图 3 中的 8DW 压缩块 300 细节)。数据字节 3 是非匹配字节,并具有值“F2”。相应地,数据 0 槽处的单一字节利用对应的二进制值“11110010”编码。

[0041] 包含条目 300-0 数据的 8DW 压缩块作为 PCIe 分组中的有效负载数据传输到接收器,在那里,它被解压缩器逻辑 130 解压缩。接收器具有其自己的字典副本,标记为字典 140B。在接收到 8DW 压缩块时,解压缩器逻辑 130(参见图 1)解码压缩数据,提取数据,如通过提取数据块 142 所描绘的,还提取匹配比特模式和匹配代码(框 146)。每一个 DW 的匹配比特都被用来判断是否有该 DW 的匹配。在标识对于给定 DW 存在匹配时,执行字典提取和数据合并操作,如通过提取索引块 144、字典 140B 和合并块 148 所描绘的。基于对于匹配的编码,逻辑知道,类型 / 第一数据 0 条目的开头四个比特标识匹配模式,后面的四个比特包括对应于匹配的行的行索引的字典地址。此外,基于匹配模式比特“0111”,逻辑被配置成从匹配的行的字典条目检索字节 0、字节 1,以及字节 2,并将字节 3 与存储在数据 n 槽(在此示例中,数据 0,带有 F2 值)处的数据的单字节合并。如此,具有值“F203010B”的解压缩的数据 116D 与原始数据 116 相同。

[0042] 按类似的方式处理其他匹配 DW 值。在发射器一侧,首先,通过字典查询,检测匹配。如果得到匹配,则利用对应的 DW 索引的匹配比特来编码 8DW 压缩块,并将适用的匹配模式比特和行索引比特插入到适用的类型 / 第一数据 n 槽中。取决于特定匹配模式,以预先确定的顺序,利用非匹配的字节的副本来自编码数据 n 槽。(如上文所讨论的,如果所有字节都匹配,则数据 n 槽将是空的。)在接收器一侧,解码压缩数据,对带有其匹配比特的对应的“1”的每一个 DW 的处理继续,基于匹配模式和行索引,进行适用的字节的字典提取,并与合适的数据 n 槽中的 0-2 个其余数据字节合并。

[0043] 图 6 的框示意图 / 流程图示出了现有的字典条目被替换为原始数据 116A 的非匹配情况的示例。在此示例中,原始数据 116A 具有值“F203010B”,带有 DW 索引“0110”(即,6)。如所描绘的,字节 0-3 中一个也不匹配字典 140A 中的字节 0-3 的任何对应的条目。然而,非匹配条件的定义一般不要求所有字节都不匹配(虽然可以这样,取决于特定实现),而是要求匹配字节的数量小于字节匹配的阈值数量(在此示例中,阈值匹配数是两个)。相应地,如果有小于两个的按字节的匹配,那么,发生非匹配或未命中条件。如以前一样,这可以通过并行地搜索字典条目来检测。

[0044] 响应于未命中,逻辑可以被配置成利用对应的数据来更新(即,替换)字典条目。这可以在各种情况下有利,诸如访问存储器的相同地址或 I/O 事务。通过将对应于地址的数据添加到字典中,当在对应的事务中引用地址或附近的地址时,数据压缩可用。

[0045] 用于将条目添加到字典中的逻辑可能会变化,取决于特定实现和有针对性的用途。例如,可以使用用于将条目添加到缓存中的常规逻辑,诸如 FIFO(先进先出)和 LRU(最

近最少使用的) 算法。然而, 测试和建模示出了, 使用这些类型的替换算法的无限制的并行字典更新可能会产生次最佳的结果, 取决于使用上下文。

[0046] 并非对于字典条目使用全局更换策略, 一个实施例使用基于对应的 DW 序列(例如, 8DW 压缩块的适用的 DW 索引), 将新条目限制到字典条目的子集的方法。在第一级别, 可以只向可替换的字典条目的子集写入未命中的每一个输入双字。然后, 可以使用第二级别方案来从条目的子集中选择特定条目, 诸如 FIFO、LRU 或其他替换算法。

[0047] 所提出的方法类似于在某些缓存设计中所使用的已知的组相关联的技术。可以分配未命中请求, 并向由缓存组所定义的缓存位置的子集写入未命中请求。这里的差异是双重的。首先, 组相关联的高速缓存中的分配是基于请求的地址比特的子集执行的; 这里, 在一个实施例中, 分配是由输入流内的文本符号位置(时间方面的)所定义的。其次, 由于特定数据可能出现在任何文本符号位置, 因此, 为检测高速缓存命中, 将输入文本符号中的每一个与所有字典条目进行比较, 不管分配如何。在这种意义上讲, 比较相当于全相联高速缓存, 而替换相当于组相关联的高速缓存。当应用于字典上下文时, 方案被称为“N 路组分配”字典更换方案。

[0048] 在 N 路组分配字典更换方案的一个实施例中, 一组替换字典条目与块中的数据的每一部分的索引相关联, 索引此处一般是指数据索引以将它与字典索引区别。例如, 如果数据的每一个部分包括双字, 则索引是 DW 索引(此处在附图中示出), 块中的索引的数量等于块中的 DW 的数量。此外, 在一个实施例中, 每一个数据索引的替换字典条目的组是唯一的, 以便在各组之间没有重叠。通过如此配置组, 可以对于给定块的数据的所有部分并行地执行并行操作, 对于该块中的数据的各部分的匹配 / 未命中条件的任何组合, 促进对应的字典更新。

[0049] 根据此并行替换组实施例, N 路组分配字典更换策略的替换组可以被一般化以适用于对于各种大小字典和数据块的字典替换策略, 如下列公式 1 所定义的,

$$[0050] \quad U_{i=0}^{k/N-1} j + N * i.$$

[0051] 其中, U 是合并操作, 数据的每一部分都具有数据索引 j, 块中的数据的部分的数量是 N(也等于数据索引的数量和组的数量), 而字典中的可替换的字典条目的数量是 k。例如, 如果一个块被分成 8 部分数据(即, N = 8), 且字典中的可替换的条目的数量(k)是 16, 则数据索引 j 的可替换的条目的组将是 {j, j+8}, 对应于 8 路组分配, 每一个组都包括两个字典条目。在图 6 中所示出的实施例中描绘了此结果。其他类似的更换方案也可以使用适用的 N 路组分配来实现, 如公式 1 所定义的。

[0052] 在一个实施例中, 使用 LRU 更换方案来选择要被按组地替换的字典条目。例如, 对于每一组, 存储了标识该组的最近最少使用的条目的信息。对于带有两个成员的组, 可以使用简单得如触发器或单一逻辑比特的一些东西来跟踪 LRU 条目。其他已知的 LRU 方案可以用于具有多于两个的成员的组。在另一个实施例中, 使用 FIFO 方案。也可以使用其他组更换方案, 诸如加权的循环, 伪随机等等。

[0053] 返回到图 6, 在所示出的示例中, 组关联是使用 j+8 更换策略的八路组关联。要被替换的可能的条目是 j, DW 索引, 以及 j+8; 在此示例中, j = 6, 可能的条目是 6 和 14。在所示出的示例中, LRU 算法将行 6 标识为要替换的适用的行(在可能的行 6 和 14 之间), 如

此,行 6 中的字节 0-3 数据被替换为原始数据 116A 的字节 0-3,即,非匹配 DW 数据。

[0054] 字典未命中在发射器一侧导致两个动作。首先,如刚刚所讨论的,字典条目被替换为非匹配数据,如通过指向字典 140A 的更新数据箭头所描绘的。其次,整个 DW 数据未经压缩地传输(根据 8DW 压缩块编码方案)以便由接收器处的解压缩器逻辑 130 处理。这是通过将匹配比特 206 设置为 0,将字节 0 值复制到合适的类型 / 第一数据 n 槽,以及将字节 1、字节 2 以及字节 3 数据复制到应用程序数据 n 槽来实现的。

[0055] 如上文所讨论的,字典 140A 和 140B 被保持同步。相应地,由于传输一侧字典 140A 是使用 DW 索引值来更新的,DW 索引数据在 8DW 压缩块中编码,因此,对于字典条目替换算法的适用的 DW 索引被接收器解码,如提取索引块 144 所描绘的。结果,当接收器解压缩器逻辑处理压缩块时,将在接收器处执行正好相同的字典更新,如此,使发射器和接收器处的字典数据保持同步。

[0056] 如以前一样,编码数据 300-6 在被分组化的并通过链路发送到的接收器的 8DW 压缩块中编码,在接收器处,它根据解压缩器逻辑 130 被解压缩。匹配比特 206 的“0”值表示,将有字典条目替换,替换值的字节 0 存储在适用的类型 / 第一数据 n 槽处(在此示例中,类型 / 第一数据 6 槽),并从适用的数据 n 槽(数据 6 槽)中提取字节 1、字节 2 以及字节 3。相应地,行 6 被利用原始数据 116A 数据(也对应于解压缩的数据 116AD 的值)更新。

[0057] 图 7 示出了使用并行压缩 / 解压缩操作并利用 8DW 压缩块 300 实现的数据传输操作的各阶段的示例输入数据和字典数据。并行地接收和处理第一组 8DW,如由 8DW 组 202-1 所描绘的。第二组 8DW 被描绘成 8DW 组 202-2;这些数据对应于要被处理的下一组 8DW。发射器一侧的字典 140A 被示为处于处理 8DW 组 202-1 之前的状态。同时,接收器一侧的字典 140B 被示为处于处理 8DW 组 201-1 之后的状态,更新的字典条目以粗体示出。图 7 进一步示出了与 8DW 压缩块 300 中的槽相邻的各种数据值。这些包括匹配比特,每一个类型 / 第一数据 n 槽的适用的值,以及存储在数据 n 槽中的数据字节。

[0058] 如在图 7 中所描绘的,匹配字节被以框架箱(framed boxes)示出,所产生的匹配比特是“10100100”。结果,对应于要被替换的字典条目的适用的 DW 数据 n 是数据 1、数据 3、数据 4、数据 6 以及数据 7。为了方便起见,对应的行 1,3,4,6,以及 7 的字典条目被示为在字典 140B 中被替换(更新)。然而,应该认识到,在一个实施例中,可以使用如前所述的 j+8 更换方案。结果,可能的替换行是 1 或 9,3 或 11,4 或 12,6 或 14,以及 7 或 15,每一组中的适用的行依赖于正在被使用的组替换算法以及适用的以前的数据使用和 / 或替换历史(例如,对于 LRU)。将进一步认识到,相同行将在字典 140A 中被替换(替换未示出)。

[0059] 除具有可替换的字典条目之外,字典还可以包括固定的(即,预定义的)条目,或可以通过嵌入的逻辑或其他手段,实现适用的规则,以支持类似的操作。例如,图 4 的表 400 中的数据的编码的格式化的一部分对应于数据值包含具有值 0 的一个或多个字节的规则。如带有标题“类型 / 第一数据 [7:4] 数据”的列所定义的,为包含带有零值的两个或更多字节的数据定义了 0x0 的编码(即,“0000”)。在“格式”列中定义了对应的匹配模式。另外,表 400 中的最后四个条目对应于相应的没有输入数据,或不比较的输入数据的一个、两个或三个字节的情况。在后三种情况的每一种情况下,规则指定原始数据的副本通过 8DW 压缩块 300 的数据 n 字节部分来传输。

[0060] 根据字典更新操作的其他实施例,在字典更新之间实现了延迟,字典中的全部可

替换的条目被替换或者可替换的条目的子集被替换。例如，在一个实施例中，以类似于上面的方式执行操作，但是，不是更新未命中上的字典条目，而是将对应的替换条目添加到发射器处的字典的临时的“替换”副本中（被称为延迟的更新字典），同时让发射器和接收器处的原始字典数据保持不变。也如以前一样，数据的副本以其原始形态发送到接收器，在那里，提取它。在预定义的延迟之后，诸如 M 个周期，延迟的更新字典中的更新的条目被复制到发射器处的“活动”字典，条目的副本被传输到接收器，以便两个字典都被利用适用的更新的条目组更新。在一个实施例中，延迟的字典更新是使用一个或多个 8DW 压缩块（或其他大小的 DW 压缩块）完成的，在这种情况下每一个数据 n 条目都被编码为未命中，对字典条目的更新以上文对于未命中所描述的方式来处理。另外，在一个实施例中，同步地发起将字典条目从延迟的更新字典复制到发射器和接收器两者处的字典副本中。在另一个实施例中，发射器和接收器中的每一个都在两个端点处使用相同替换逻辑来维护延迟的更新字典的相应的副本。

[0061] 此方法有多种可能的变种。首先，可以使用各种不同的替换算法来替换临时存储在延迟的更新字典中的字典条目，包括 FIFO、LRU、加权循环、伪随机等等。这些可以使用上文所讨论的完全关联或者组关联的替换方案来实现。另外，延迟的字典更新操作的时序或调用可以根据各种方案来实现。例如，可以使用诸如简单计数器之类的计时器来在预定的时间段之后（诸如上文所讨论的 M 个周期）发起延迟的字典更新。可任选地，可以维持监测字典的延迟的更新副本中的已更新的行的数量的计数。一旦数字达到一个阈值，就可以发起延迟的更新。一般而言，阈值可以对应于字典中的全部可替换的行的数量，或其一部分。

[0062] 图 8a 和 8b 示出了前述的讨论的各方面。图 8a 示出了响应于对于单一数据 n 条目的字典未命中执行的操作，认识到将同步地执行对应于 DW 压缩块中的多个数据 n 条目的匹配结果的未命中操作。如以前一样，字典 140A 和 140B 中的数据是相同的，并且在字典更新方面之外，以上文参考图 5 和 6 所讨论的方式处理对于未命中和匹配的数据传输。图 8a 还示出了延迟的更新字典 141，该字典 141 最初将与紧随延迟的字典更新操作之后的字典 140A 和 140B 包含相同数据。为示出此状态，在时间 0 示出了字典 140A 和 140B，该时间表示最近的延迟的字典更新操作的完成时间。

[0063] 在正在进行的操作过程中将发生未命中。响应于每一个未命中，在延迟的更新字典 141 中，对应的行条目将被更新（即，替换），如在图 8a 中所描绘的，但在发射器字典 140A 或者接收器字典 140B 中不更新。虽然这示出了上文呈现的组相关联的替换策略的示例，但是，这只是示例性的，因为可以使用其他类型的替换策略以及相关联的替换算法。

[0064] 为延迟的更新字典 141 的视图指出的时间是时间 M，该时间 M 可以表示自从最后一次延迟的字典更新以来的 M 个周期，或 M 可表示自从最后一次延迟的字典更新以来的时间段。在一个实施例中，对 M 个周期或时间段的发生的检测可以使用计数器 800 来实现。

[0065] 在时间 M，发起延迟的字典更新，如在图 8b 中所描绘的。首先，利用来自延迟的更新字典 141 的数据，来更新字典 140A，如在时间 M+1 所描绘的。在一个实施例中，在单一周期内，并行地更新字典的内容。这可以，例如，通过将每一个字典行数据存储到寄存器中来实现，且用于每一行的相邻的寄存器存储用于该行的延迟的更新字典数据，其中相邻的寄存器共同地包括延迟的更新字典。也可以使用其他已知数据镜像方案。进一步指出，延迟的更新字典的大小只须对应于将被替换为每一个更新的行的最大数量（通常将等于字典

中的可替换的行的数量,但是,可以少一些)。

[0066] 此时,或与字典 140 的更新同时发起,每一行的副本(或部分字典更新的适用的行)通过通信链路传输到接收器处的更新字典 140B。例如,在一个实施例中,替换所有十六个字典条目,阴影字典 141 中的行 0-7 在第一 8DW 压缩块中发送,行 8-15 在第二 8DW 压缩块中发送,每一个压缩块的数据 n 被编码为未命中。可任选地,可以使用其他方案来将延迟的更新字典 141 的副本传输到接收器字典 140B。在延迟的字典更新操作完成时,字典 140B 中的数据再次匹配字典 140A 中的数据。此时间被描绘成时间 M+1+L,“L”表示用于执行字典数据副本传输的等待时间。

[0067] 图 8c 描绘了延迟的字典更新的替换实施例。在此实施例中,相应的延迟的更新字典 141A 和 141B 在发射器和接收器两者中实现。相应地,为执行延迟的字典更新,延迟的更新字典数据被以同步方式复制到相应的字典 140a 和 140b,如此提供两个字典的同步更新这可以通过在接收器处实行模式切换来实现,如此,解压缩器逻辑可以切换到延迟的字典更新模式,在该模式下,字典替换条目被输入到接收器的延迟的更新字典中,而并非其活动字典中。在一个实施例中,如上文所描述的,对 8DW 压缩块的编码保持不变,在接收器一侧的处理和以前一样,当处理编码的未命中时字典更新方面除外。

[0068] 通过使用并行操作,压缩 / 解压缩操作可以实时地以线速度以及以前根据 PCIe 不可能的传输带宽来执行。例如,在一个实施例中,实现使用 8GT/s 的 PCIe3.0 线速度的 x16(即,16 通道)PCIe 链路。为支持 8GT/s 的维持的传输速率,8DW(256 比特)的未压缩的数据将需要以 1GHz 的时钟速率处理。在上文所讨论的各实施例中,解压缩器逻辑能够以 1GHz 时钟速率处理 4DW 的压缩数据。如此,如果通过使用压缩 / 解压缩方案获得的平均压缩比是 2x(意味着,平均起来,8DW 的未压缩的数据被压缩成 4DW),可以获得 8GT/s 线速度。

[0069] 一般而言,此处所公开的各实施例的各方面可以在包括分离的组件的系统中,和 / 或在集成的组件和诸如片上系统(SoC)之类的系统中实现为独立组件。例如,当作为发射器和 / 或接收器来操作时,使用高速串行接口的外围设备可以被配置成支持压缩 / 解压缩操作。类似地,当作为发射器和 / 或接收器来操作时,诸如使用高速接口来支持处理器或 I/O 中枢和外围组件或控制器芯片(例如,图形控制器或 GPU)之间的通信的个人计算机或服务器之类的使用分离的组件的系统内的 PCIe 或其他串行链路端点可以被配置成支持压缩 / 解压缩操作。类似地,SoC 可以配置有用于在 SoC 上的功能块之间传输数据的高速串行链路。在其他实现中,包括压缩器或解压缩器的逻辑可以嵌入在 SoC 的功能块中,以作为一个端点来操作,而另一个端点在 SoC 外部的通过串行互连或串行链路链接到 SoC 中的功能块的组件中实现。

[0070] 虽然上文是在为通过链路来传输数据使用基于字典的压缩 / 解压缩方案的上下文中描述的,但是,这不旨在是限制性的。一般而言,此处的技术可以用于使用高带宽无损失数据压缩的任何应用。

[0071] 当压缩支持重复的和 / 或可预测的数据或数据序列的数据或数据流时,此处所公开的基于字典的压缩 / 解压缩技术的使用一般是有利的。例如,存储器和访问存储器块或数据的 I/O 事务可以使用相同或类似的地址和相同数据请求命令(编码为对应的数据)。由于从事务请求一侧传输的数据的合理的部分是重复的,因此,这些类型的传输是对于基于字典的压缩 / 解压缩的好候选。在 f1ip 一侧,涉及大部分的随机数据的传输不是好的候

选,因为命中率一般是差的。结果,与压缩 / 解压缩操作相关联的开销当平均考虑时将超出对于经压缩的小部分分组的带宽改进。对字典更新方案的选择也可以是应用或使用特定的。例如,延迟的字典更新的使用可以适用于某些使用情况,而不适用于其他情况。

[0072] 鉴于前面的内容,可以有这样的情况:串行链路端点之间的传输的一个方向使用压缩 / 解压缩,而相反的方向的链路不。此外,在某些实现中,通过使用对应的编码技术或其他措施,跨给定链路,启用和禁用压缩 / 解压缩是有利的。例如,某些类型的事务可以具有以结合对应的事务而有选择地启用或禁用压缩 / 解压缩操作的使用的方式编码的命令。

[0073] 在此处的实施例中使用 32 比特数据格式(例如,对于字典条目)是示例性的,而不是限制性的。然而,当实现使用其他数据宽度格式时,应该作出诸如链路传输参数(例如,通道宽度、带宽、编码等等),以及开销(包括延迟、电路逻辑复杂性和成本)之类的各方面的考虑。类似地,使用 16 个可替换的字典条目也是示例性的。对压缩数据的编码,例如,可以增强以支持更多字典条目,但是,应该认识到,这样的增强一般可以要求对相关的数据结构以及逻辑的修改。

[0074] 上文对本发明的所示出的本发明的各实施例的描述,包括在“摘要”所描述的,不是详尽的,或将本发明限于上文所公开的准确的形式。尽管为了说明,此处描述了本发明的具体实施例以及示例,但是,如那些精通相关技术的人所理解的,各种可能的等效的修改也都在本发明的范围内。

[0075] 可以根据上面的“详细描述”对本发明进行这些修改。后面的权利要求中所使用的术语不应该被理解为将本发明限制于说明书和附图中所公开的特定实施例。相反,本文的范围完全由下列权利要求书来确定,权利要求书根据权利要求解释的建立的原则来解释。

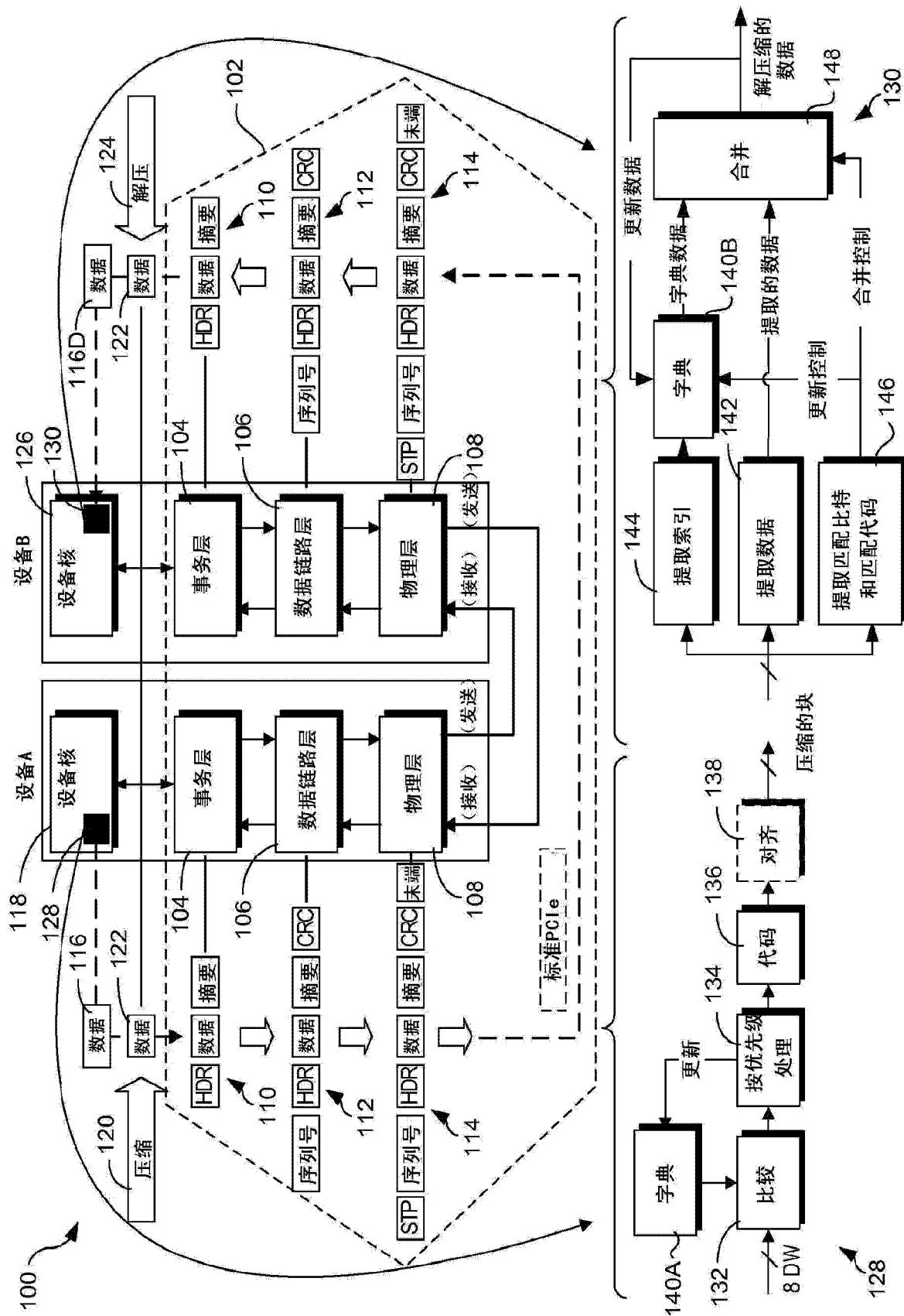


图 1

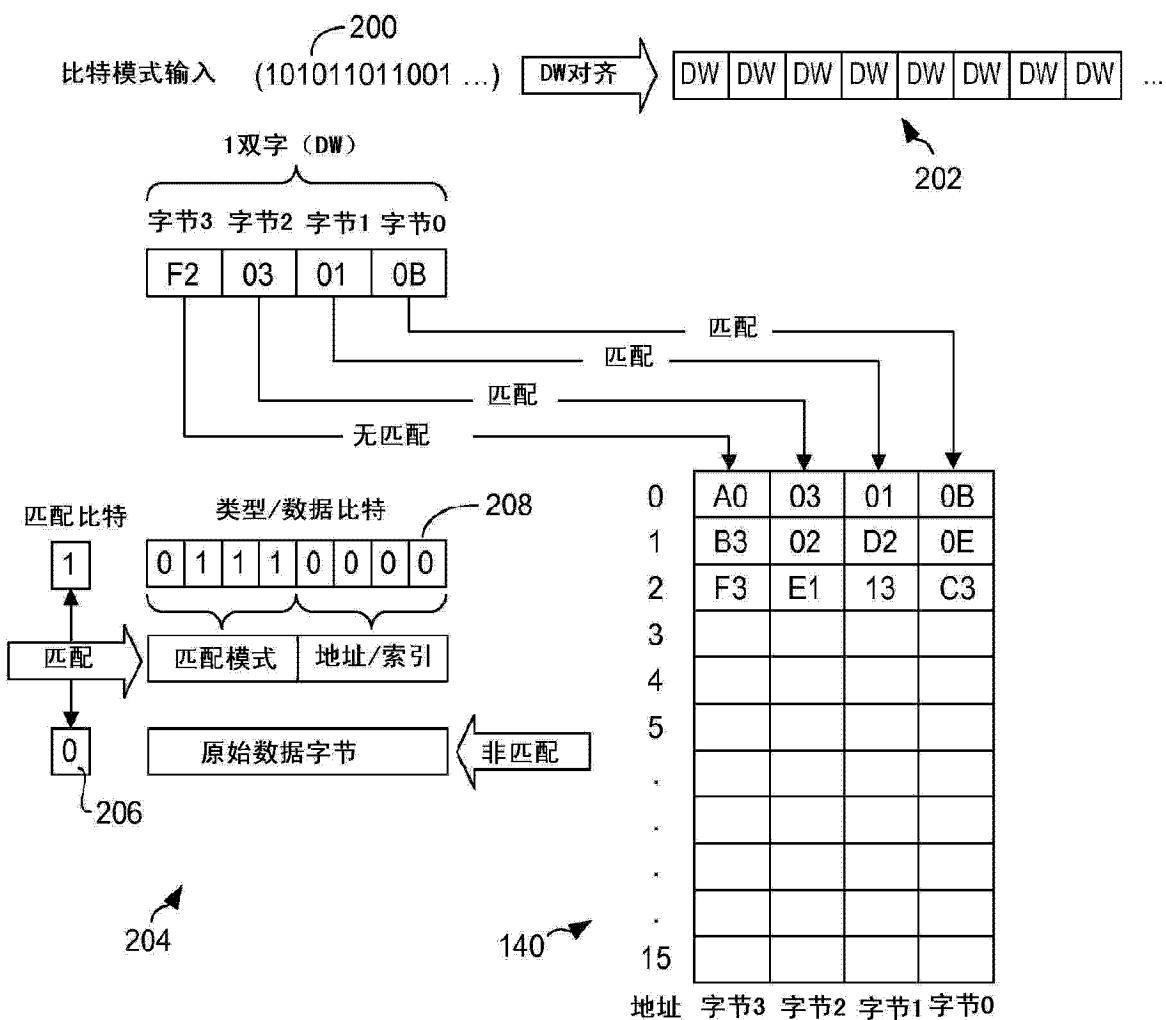


图 2

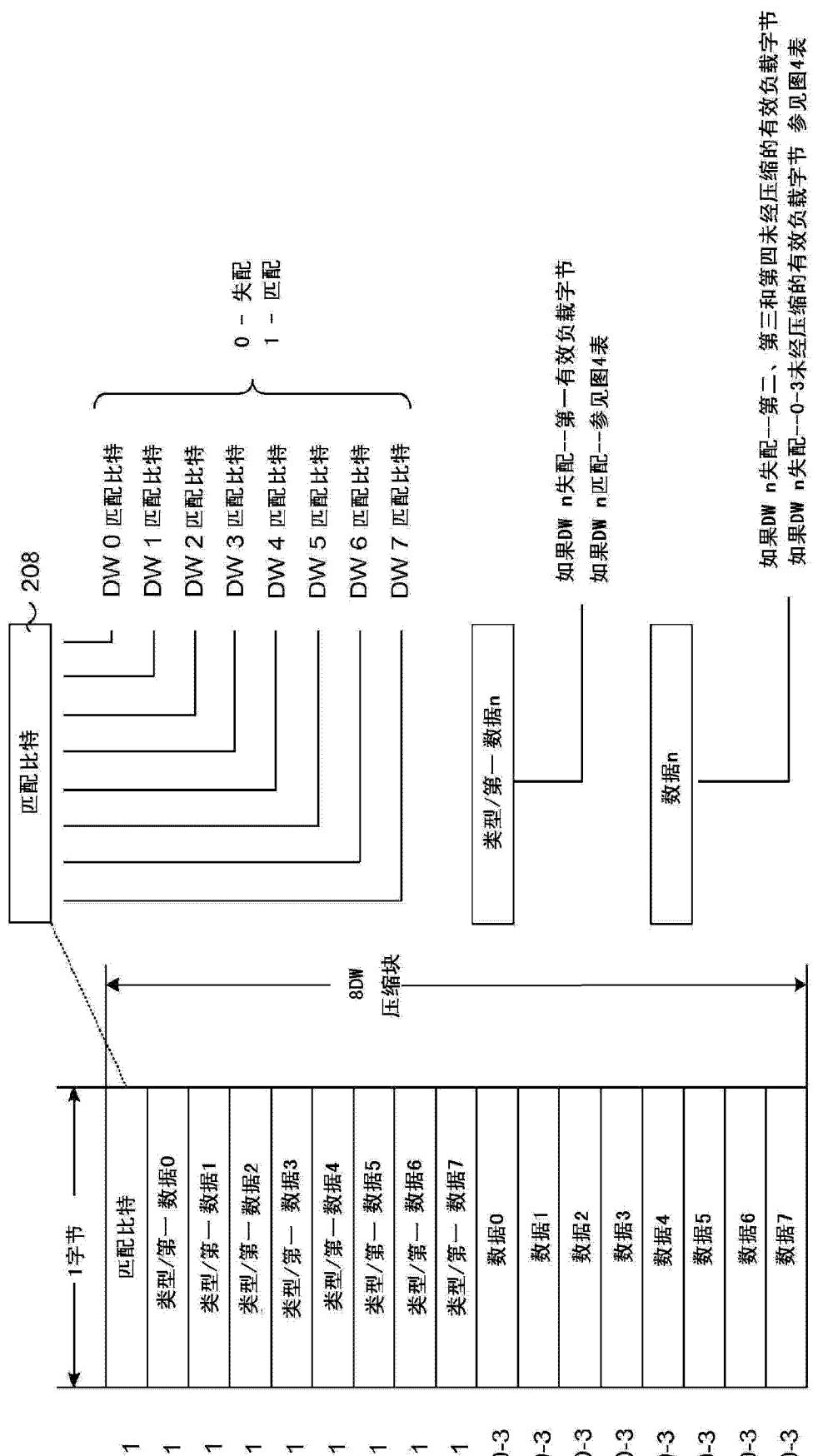


图 3

| 匹配比特 | 类型/第一数据[7:4] | 类型/第一数据[3:0] | 格式             | 数据n字节                      |
|------|--------------|--------------|----------------|----------------------------|
| 0    | 数据字节0[7:4]   | 数据字节0[3:0]   | 无匹配            | 数据字节 1<br>数据字节 2<br>数据字节 3 |
| 1    | 0xF - 1111   | 字典索引         | 4字节匹配 (mmmm)   | -                          |
| 1    | 0x7 - 0111   | 字典索引         | 3字节匹配 (xmmm)   | 数据字节 3                     |
| 1    | 0xB - 1011   | 字典索引         | 3字节匹配 (mxmm)   | 数据字节 2                     |
| 1    | 0xD - 1101   | 字典索引         | 3字节匹配 (mmxm)   | 数据字节 1                     |
| 1    | 0xE - 1110   | 字典索引         | 3字节匹配 (mmmx)   | 数据字节 0                     |
| 1    | 0x3 - 0011   | 字典索引         | 2字节匹配 (xxmm)   | 数据字节 2<br>数据字节 3           |
| 1    | 0xC - 1100   | 字典索引         | 2字节匹配 (mmxx)   | 数据字节 0<br>数据字节 1           |
| 1    | 0x9 - 1001   | 字典索引         | 2字节匹配 (mx xm)  | 数据字节 1<br>数据字节 2           |
| 1    | 0x6 - 0110   | 字典索引         | 2字节匹配 (xmmx)   | 数据字节 0<br>数据字节 3           |
| 1    | 0xA - 1010   | 字典索引         | 2字节匹配 (mxmx)   | 数据字节 0<br>数据字节 2           |
| 1    | 0x5 - 0101   | 字典索引         | 2字节匹配 (xm xm)  | 数据字节 1<br>数据字节 3           |
| 1    | 0x0          | 0xF - 1111   | 4零匹配 (zzzz)    | -                          |
| 1    | 0x0          | 0x7 - 0111   | 3零匹配 (x zzz)   | 数据字节 3                     |
| 1    | 0x0          | 0xB - 1011   | 3零匹配 (z xzz)   | 数据字节 2                     |
| 1    | 0x0          | 0xD - 1101   | 3零匹配 (zz xz)   | 数据字节 1                     |
| 1    | 0x0          | 0xE - 1110   | 3零匹配 (zzzx)    | 数据字节 0                     |
| 1    | 0x0          | 0x3 - 0011   | 2零匹配 (xx z z)  | 数据字节 2<br>数据字节 3           |
| 1    | 0x0          | 0xC - 1100   | 2零匹配 (z zxx)   | 数据字节 0<br>数据字节 1           |
| 1    | 0x0          | 0x9 - 1001   | 2零匹配 (zxxz)    | 数据字节 1<br>数据字节 2           |
| 1    | 0x0          | 0x6 - 0110   | 2零匹配 (x zzz)   | 数据字节 0<br>数据字节 3           |
| 1    | 0x0          | 0xA - 1010   | 2零匹配 (zx z x)  | 数据字节 0<br>数据字节 2           |
| 1    | 0x0          | 0x5 - 0101   | 2零匹配 (x z x z) | 数据字节 1<br>数据字节 3           |
| 1    | 0x0          | 0x0          | 无输入数据          | -                          |
| 1    | 0x0          | 0x1 - 0001   | 一个字节数据，不比较。    | 数据字节 0                     |
| 1    | 0x0          | 0x1 - 0010   | 两个字节数据，不比较。    | 数据字节 0<br>数据字节 1           |
| 1    | 0x0          | 0x1 - 0100   | 三字节数据，不比较。     | 数据字节 0<br>数据字节 1<br>数据字节 2 |


 400

图 4

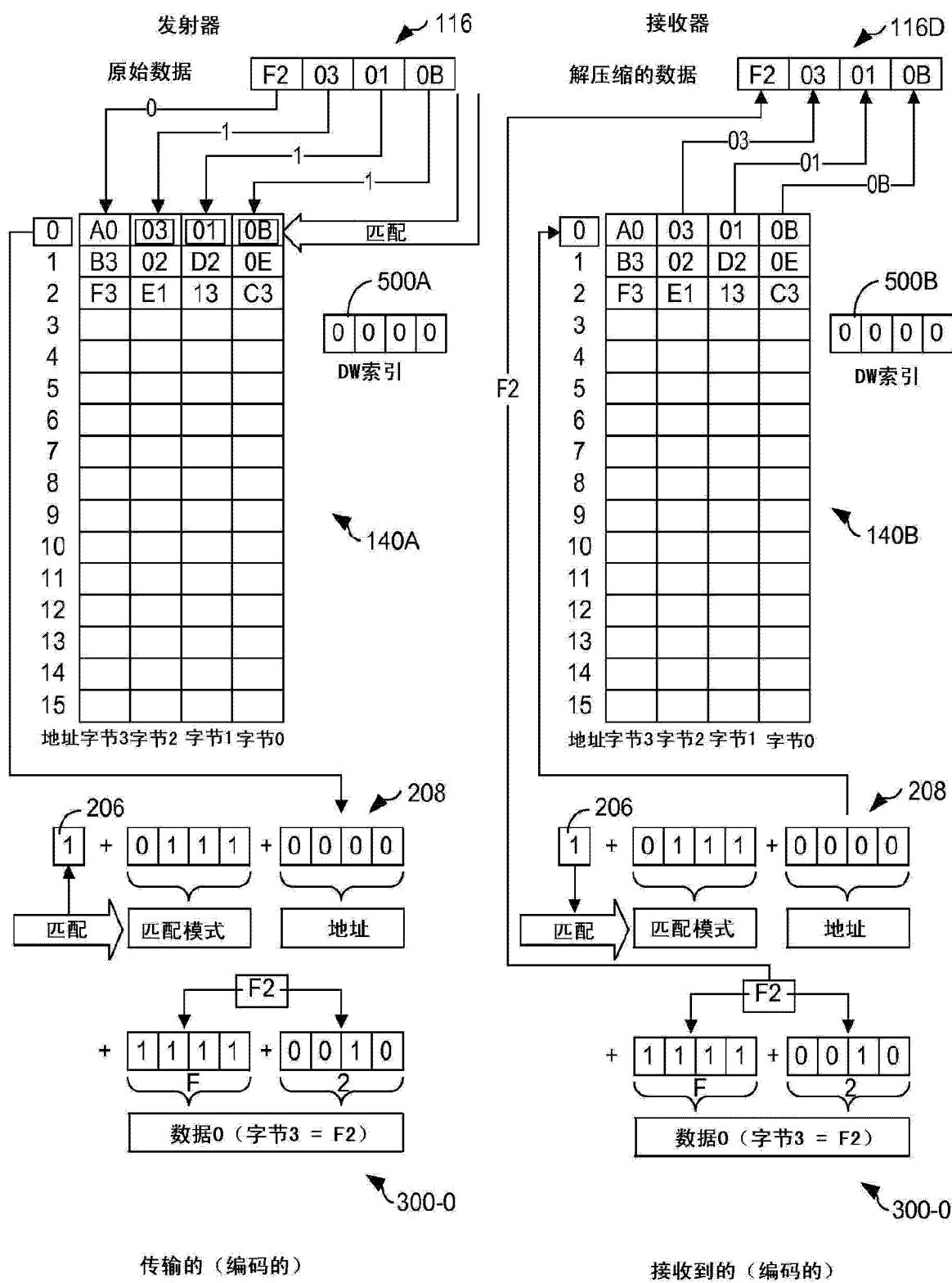


图 5

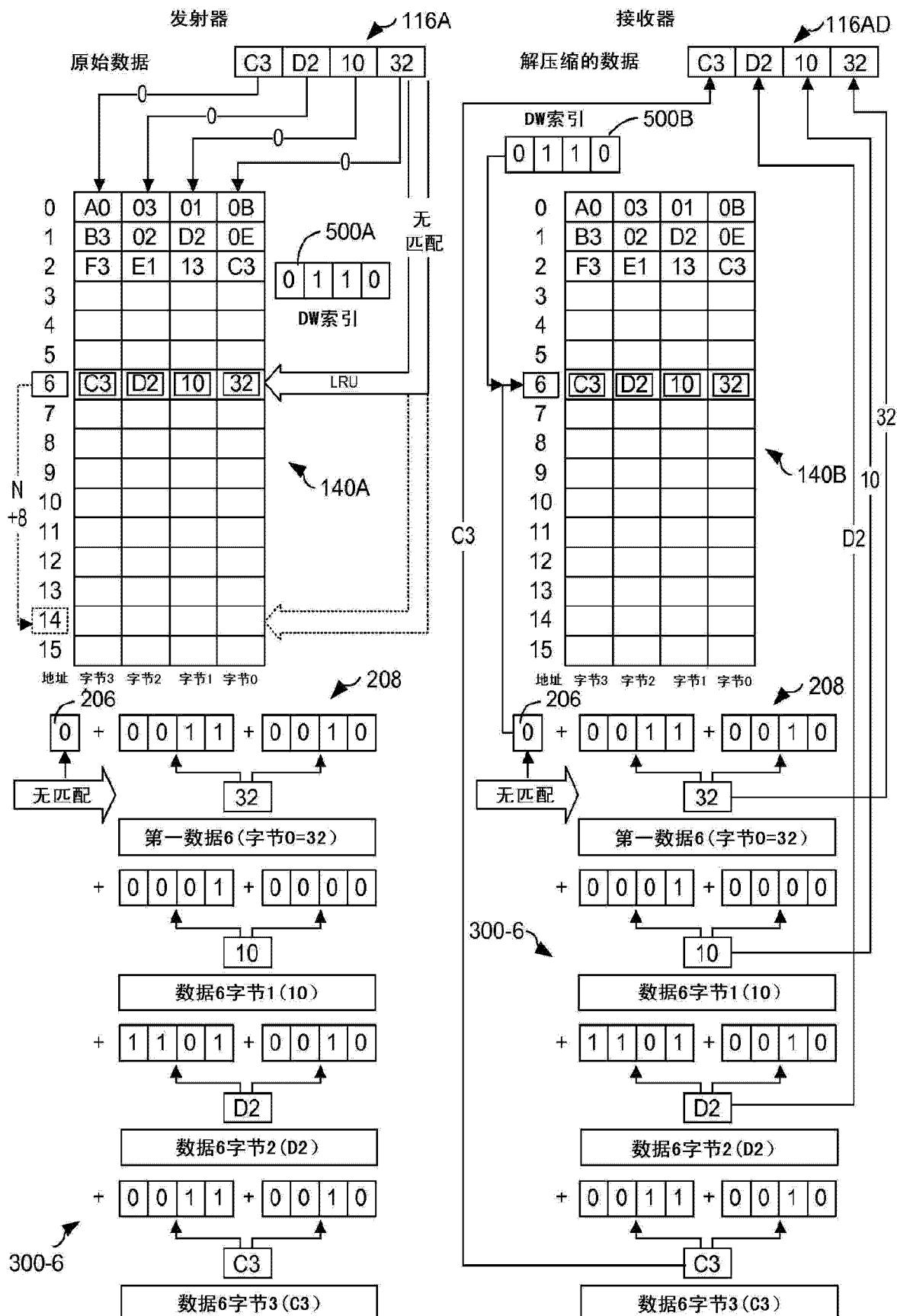


图 6

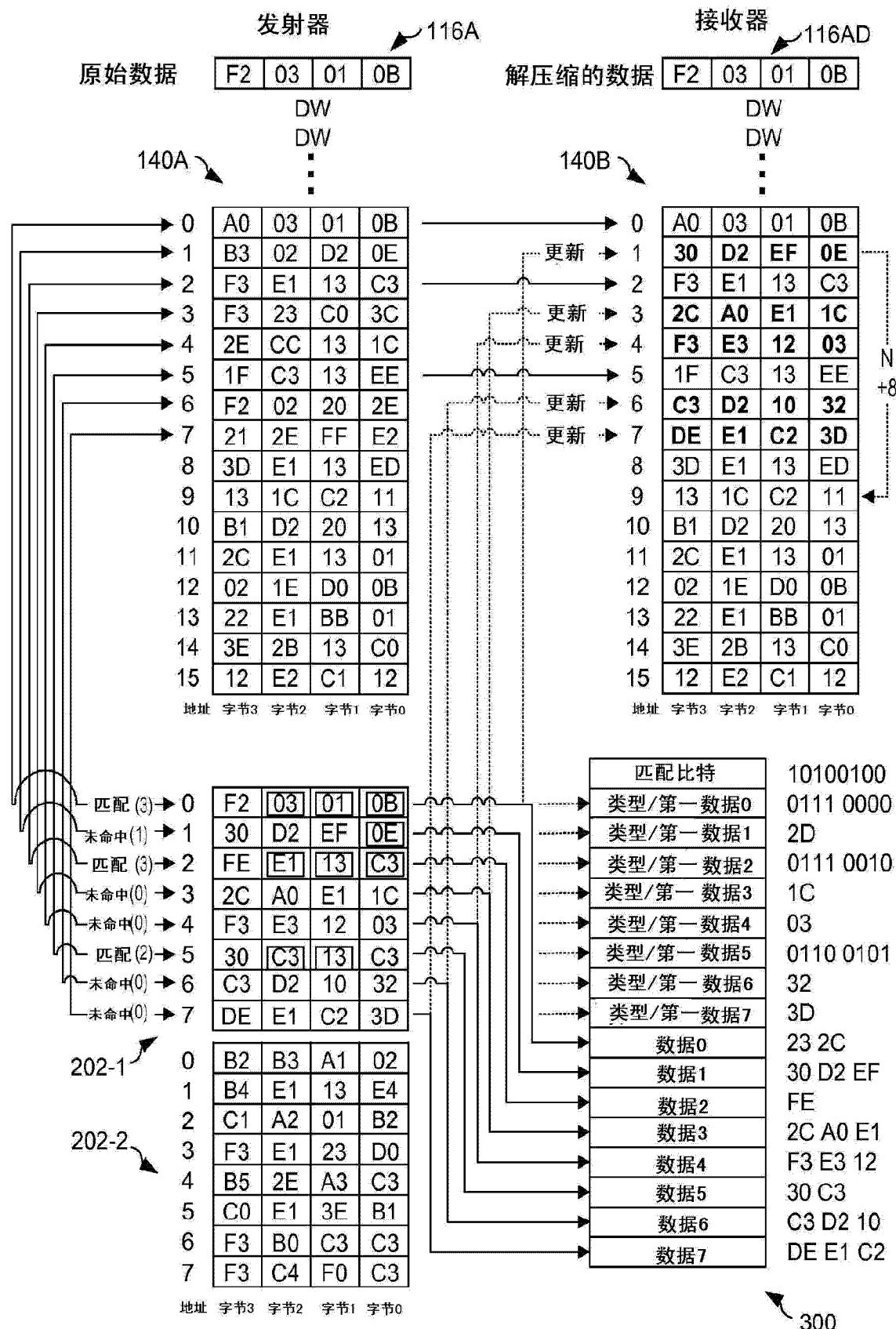


图 7

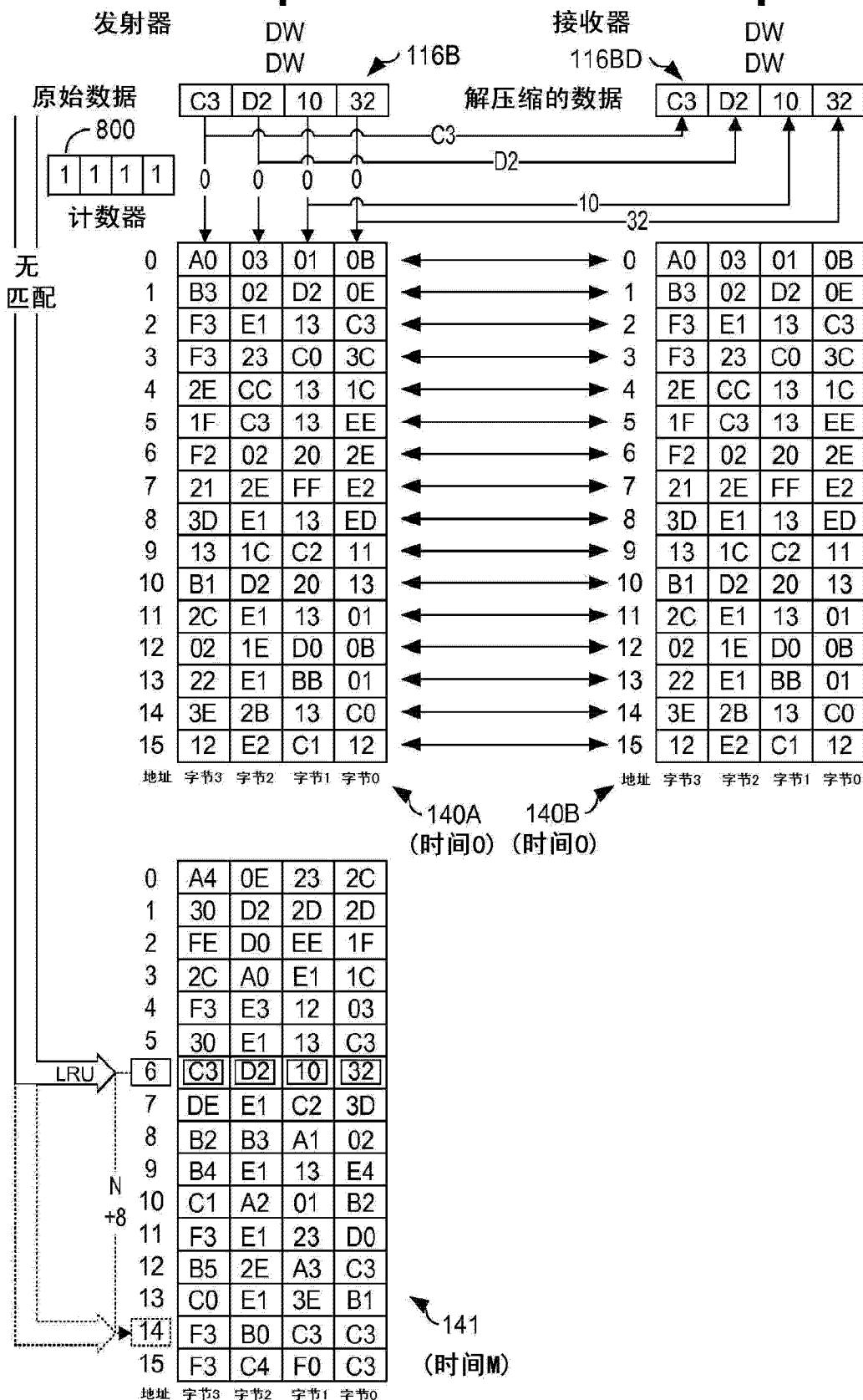


图 8a

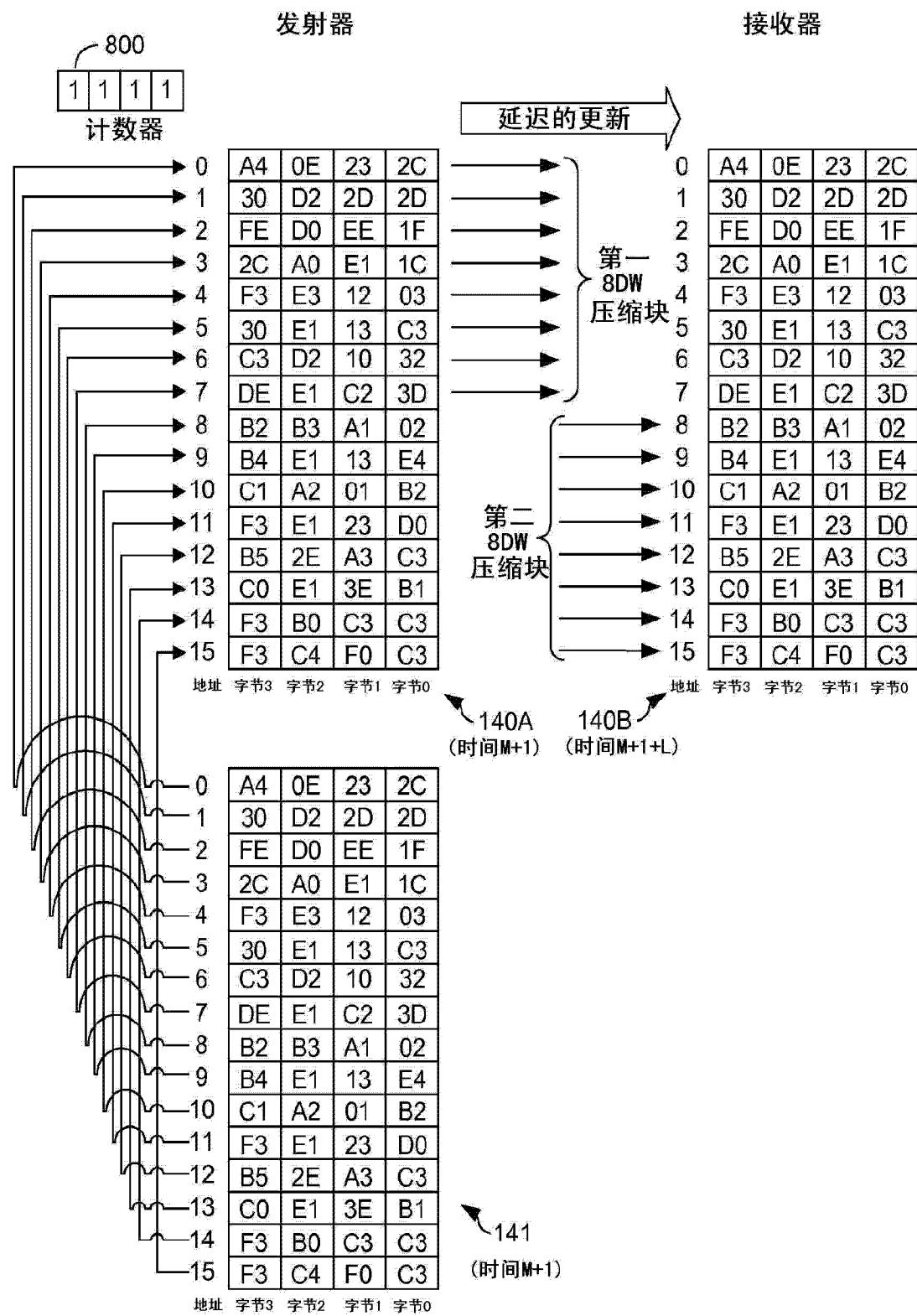


图 8b

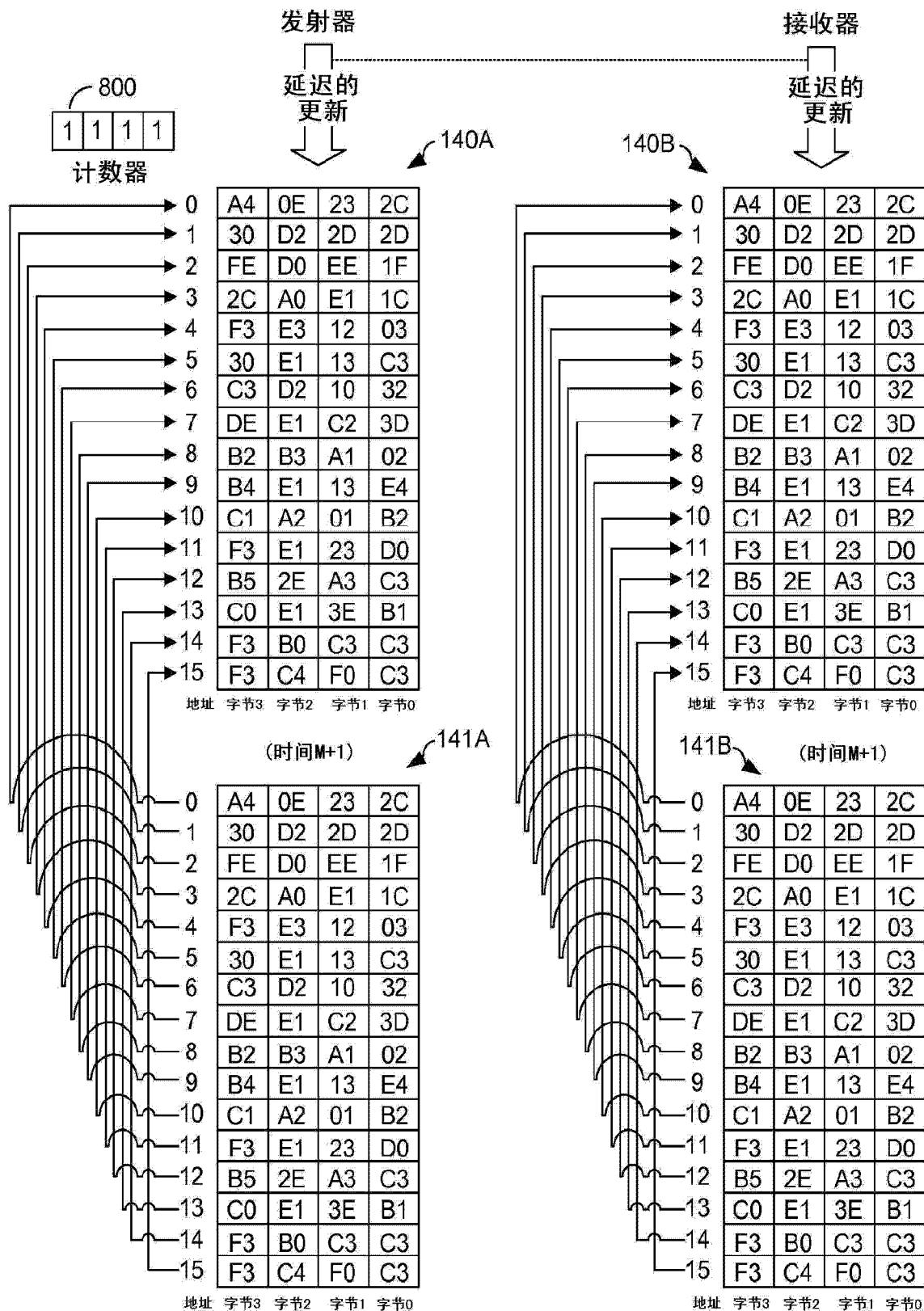


图 8c