[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G11C 16/10 (2006.01)
G06F 13/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 200710074652.X

[45] 授权公告日 2009 年 3 月 11 日

[11] 授权公告号 CN 100468576C

[22] 申请日 2007.5.30
[21] 申请号 200710074652.X
[73] 专利权人 忆正存储技术(深圳)有限公司
地址 518057 广东省深圳市南山区高新技术产业园南区 W2A 栋三楼
[72] 发明人 黄 河
[56] 参考文献
JP2000231793A 2000.8.22
CN1474415A 2004.2.11
CN1480953A 2004.3.10
CN1249084A 2000.3.29
审查员 陈安安

[74] 专利代理机构 深圳鼎合诚知识产权代理有限公司
代理人 陈俊斌

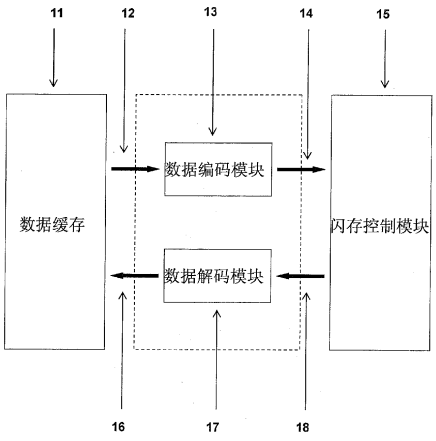
权利要求书 2 页 说明书 8 页 附图 6 页

[54] 发明名称

闪存数据读写处理方法

[57] 摘要

本发明公开了一种闪存数据读写处理方法，包括以下步骤：1)在闪存数据的存入过程中，先将待存入的二进制数据进行编码处理，且编码处理后的二进制数据中 0 的个数比编码前的二进制数据中 0 的个数少；然后将编码后的二进制数据写入闪存存储单元；2)在闪存数据的取出过程中，先将闪存存储单元中编码后的二进制数据读出，然后对读出的二进制数据进行与步骤 1)所述的编码处理相对应的解码处理。 采用本发明的方法可以减小写入和擦除操作对闪存芯片的损耗，延长闪存芯片的使用寿命；还可以提高写入和擦除操作的效率，减小操作时间；以及减小闪存操作的功耗。



1．一种闪存数据读写处理方法，其特征在于，所述方法包括以下步骤：

1)在闪存数据的存入过程中，先将待存入的二进制数据进行编码处理，且编码处理后的二进制数据中0的个数比编码前的二进制数据中0的个数少；然后将编码后的二进制数据写入闪存存储单元；

2）在闪存数据的取出过程中，先将闪存存储单元中编码后的二进制数据读出，然后对读出的二进制数据进行与步骤 1）所述的编码处理相对应的解码处理。

2．根据权利要求1所述的闪存数据读写处理方法，其特征在于，所述步骤 1）具体为在系统主机侧、闪存控制器侧和闪存芯片侧中任一侧对待存入的二进制数据进行编码处理。

3．根据权利要求1所述的闪存数据读写处理方法，其特征在于，所述步骤 2）具体为在系统主机侧、闪存控制器侧和闪存芯片侧中任一侧对读取的二进制数据进行解码处理。

4．根据权利要求1—3任一项所述的闪存数据读写处理方法，其特征在于，所述步骤 1）中将待存入的二进制数据进行编码处理具体为将数据进行分段编码处理成多组二进制数据，而且编码处理后的每一组二进制数据中0的个数都不多于编码前的对应二进制数据中0的个数，且所述步骤2）中采用与之对应的解码处理。

5．根据权利要求4所述的闪存数据读写处理方法，其特征在于，所述步骤 1）的编码处理具体为对于0的个数大于1的数据进行取反的编码处理；所述步骤 2）中的解码处理具体为对于在编码处理中被取反的数据进行取反后获得原始数据信息。

6．根据权利要求4所述的闪存数据读写处理方法，其特征在于，所述步骤 1）的编码处理具体为：在数据区后增加冗余区，将数据区中所有数据组合跟加入冗余区后的新数据区中所有数据的组合中0的个数少的数据组合建立新的映射关系；所述步骤 2）的解码处理具体为：对于编码后的数据根据新的映射关系，对其解码得到原始数据。

7．根据权利要求1—3任一项所述的闪存数据读写处理方法，其特征在于，所述步骤 1）中将待存入的二进制数据进行编码处理具体为将数据

进行分段编码处理成多组二进制数据；而且在设定周期内，编码处理后的各组二进制数据中 0 的总数小于编码前的各组二进制数据中 0 的总数，且所述步骤 2）中采用与之对应的解码处理。

8．根据权利要求 7 所述的闪存数据读写处理方法，其特征在于，所述步骤 1）的编码处理具体为：将二进制数据进行数据压缩编码；所述步骤 2）的解码处理具体为：对读取的压缩数据进行相应的解压缩运算。

9．根据权利要求 8 所述的闪存数据读写处理方法，其特征在于，所述数据压缩编码处理具体为：对数据进行压缩运算，使得写入闪存记忆单元的数据总数以及数据中 0 的个数都小于压缩运算前数据总数和 0 的个数。

闪存数据读写处理方法

技术领域

本发明涉及一种闪存数据读写处理方法，具体的涉及一种通过对数据进行编解码处理优化闪存操作的方法。

背景技术

闪存是一种重要的存储设备，闪存因为具有可多次进行数据读写，擦除，同时具有高密度、大容量、较低的读写操作耗时，以及非易失性，低功耗等特点而越来越广的被用于个人电脑，各种数字电子设备以及其他各种的数字存储设备领域；近年来，其工艺技术日趋成熟，成本价格逐渐降低，后端应用技术的日益完善，这些都大大的刺激了闪存市场的发展，使其逐渐在存储领域与硬盘的地位平分秋色。但是，闪存由于其自身制造工艺的一些问题，使其从产生就存在一些不可避免的缺陷，这类缺陷限制了其进一步的发展和应用。一方面，一般的闪存芯片都存在一个使用寿命，这是由于闪存单元自身的存储原理所决定的，闪存单元的操作方式通常是，首先将记忆单元的悬置栅放电也就是常说的擦除，使其到一个通用的状态，然后在写数据过程中，再将浮置栅极充电也就是常说的编程，使它们达到存储数据的需要状态，随着擦除和编程次数的逐渐增多，悬置栅在这种隧道效应中会逐渐被积累一些隧穿电子，从而导致需要更大的正向电压，记忆单元才能再次进行隧道效应。同时，在擦除过程中，绝缘体介质会在反复的隧道效应作用下老化，最终而不能够起到势垒的作用，也就是常说的势垒被击穿，这两种情况都会导致记忆单元不能够正常操作，也就是常说的结束了它的使用周期。对于 NAND 型闪存擦除或者重新编程次数一般是十万次左右，而对于 NOR 型闪存被擦除或者重新编程次数更少一般为万次左右。另一方面，闪存芯片写入和擦除操作比较特殊，闪存芯片是以页面为单位进行写入操作，以区块为单位进行擦除操作，闪存的写入和擦除操作时间一般需要占用很久的时间，每个页面数据从闪存芯片内部缓存写入的时间为 200us—700us，区块擦除操作时间为 2ms，同时，这些操作的时间跟闪存芯片的工艺密切相关，闪存的擦除操作，将区块内的存储单元

全都配置到状态 1，在数据写入过程中，如果需要写入的数据为 1 则不需要对该数据对应的闪存记忆单元位重新编程，如果是 0 则需要重新编程即对悬置栅充电，同时，在下一次对该页面写入数据之前，又必须先对其作一次擦除操作，那些被写作 0 的记忆单元又会被放电，因此，一次操作过程中页面数据中 0 的个数越多，页面需要被损耗的记忆单元也越多。第三方面，根据闪存存储单元的制造工艺，闪存写入操作的时间跟写入的数据值有关，每次操作页面写入的 0 的个数越多，操作需要的时间越久，同时，对于擦除操作其结论也是一样的。第四，闪存芯片的功耗跟写入的内容有关，闪存单元进行写入操作时，如果要写入的数据是 1，因为擦除操作之后数据位就是 1，所以不需要对悬置栅进行充电操作，反之，如果是 0，则需要对悬置栅进行充电操作，因此，页面数据写入的 0 的个数越少，记忆单元需要充电操作的也越少，功耗也会越低。

现有解决闪存使用寿命的方法主要为：尽量将写和擦除操作平均的分配到每一个区块，使得闪存芯片在使用过程中每一个区块被均衡的损耗，这是目前被闪存厂商广泛使用的方法。

发明内容

本发明解决的技术问题是提出了一种闪存数据读写处理方法，降低存储数据时对闪存记忆单元写入和擦除操作的次数，提高闪存的寿命和存储效率，降低功耗。

本发明提出的闪存数据读写处理方法，包括以下步骤：

1)在闪存数据的存入过程中，先将待存入的二进制数据进行编码处理，且编码处理后的二进制数据中 0 的个数比编码前的二进制数据中 0 的个数少；然后将编码后的二进制数据写入闪存存储单元。

2）在闪存数据的取出过程中，先将闪存存储单元中编码后的二进制数据读出，然后对读出的二进制数据进行与步骤 1）所述的编码处理相对应的解码处理。

优选的，所述步骤 1）具体为在系统主机侧、闪存控制器侧和闪存芯片侧中任一侧对待存入的二进制数据进行编码处理。

优选的，所述步骤 2）具体为在系统主机侧、闪存控制器侧和闪存芯片侧中任一侧对读取的二进制数据进行解码处理。

优选的，所述步骤 1）中将待存入的二进制数据进行编码处理具体为将数据进行分段编码处理成多组二进制数据，而且编码处理后的每一组二进制数据中 0 的个数都不多于编码前的对应二进制数据中 0 的个数，且所述步骤 2）中采用与之对应的解码处理。

优选的，所述步骤 1）的编码处理具体为对于 0 的个数大于 1 的数据进行取反的编码处理；所述步骤 2）中的解码处理具体为对于在编码处理中被取反的数据进行取反后获得原始数据信息。

优选的，所述步骤 1）的编码处理具体为：在数据区后增加冗余区，将数据区中所有数据组合跟加入冗余区后的新数据区中所有数据的组合中 0 的个数少的数据组合建立新的映射关系；所述步骤 2)的解码处理具体为：对于编码后的数据根据新的映射关系，对其解码得到原始数据。

优选的，所述步骤 1）中将待存入的二进制数据进行编码处理具体为将数据进行分段编码处理成多组二进制数据；而且在设定周期内，编码处理后的各组二进制数据中 0 的总数小于编码前的各组二进制数据中 0 的总数，且所述步骤 2）中采用与之对应的解码处理。

优选的，所述步骤 1）的编码处理具体为：将二进制数据进行数据压缩编码；所述步骤 2）的解码处理具体为：对读取的压缩数据进行相应的解压缩运算。

优选的，所述数据压缩编码处理具体为：对数据进行压缩运算，使得写入闪存记忆单元的数据总数以及数据中 0 的个数都小于压缩运算前数据总数和 0 的个数。

本发明的闪存数据读写处理方法主要通过编解码的方式降低闪存单元数据中 0 的个数，从而减小写入和擦除操作对闪存芯片的损耗，延长闪存芯片的使用寿命；其次，通过该算法减少闪存芯片中 0 数据写入的个数，还提高写入和擦除操作的效率，减小操作时间；第三，通过这种编解码操作的方法，减小闪存操作的功耗。

附图说明

图 1 为本发明在闪存控制器侧进行编解码处理的实施例原理图；

图 2 为本发明在闪存控制器侧进行编解码处理实施例写入操作示意图；

图 3 为本发明在闪存控制器侧进行编解码处理实施例读操作示意图；

图 4 为采用本发明实施例方法处理的闪存芯片页面数据操作示意图；

图 5 为未采用本发明实施例方法处理的闪存芯片页面数据操作示意图；

图 6 为本发明映射编解码方式实施例中 4 位数据的组合示意图；

图7为本发明映射编解码方式实施例中增加了2位冗余区之后的数据组合示意图；

图8为本发明映射编解码方式实施例中4位原始数据和增加冗余区之后的映射关系示意图；

图 9 为本发明压缩编解码方式实施例中未经压缩的原始数据示意图；

图 10 为本发明压缩编解码方式实施例中原始数据与压缩编码的对应关系示意图；

图 11 为本发明压缩编解码方式实施例中压缩后的编码数据示意图。

具体实施方式

下面通过具体实施例并结合附图对本发明进行详细说明。

根据闪存区块内记忆单元的操作特点，影响闪存记忆单元寿命的主要原因是写入过程中由 1 到 0 的操作和擦除过程中由 0 到 1 的操作对记忆单元造成的损耗，因此，如果在区块作擦除操作之后，每次操作的页面写入 0 的数目越少，擦除过程中被操作的记忆单元数目也越少，单个区块的使用次数就会随之被延长，运用统计方法可以得到，整个闪存芯片的寿命也会随之延长；同时，还可以通过控制写入数据值的方法减少操作的时间，提高写入和擦除操作的效率，同时，也可以通过控制写入数据值的方法减小闪存操作的功耗。本发明采用编解码方式通过优化写入数据的方法达到延长闪存寿命，优化闪存操作和降低闪存功耗的目的。

本发明采用的编码方式可以包括很多种，可以直接通过算法产生编码，使编码后 0 的个数低于编码之前的原始数据中的 0 的个数；还可以通过数据压缩的方法，通过减少写入的数据量，来减少数据中 0 的个数。本发明的编解码的算法可以包括多种，其主要目的是在写入操作过程中，对写入数据进行编码，使产生的编码中尽量少的含有 0，以减少写入和擦除操作

对存储单元的损耗，同时在读操作过程中通过相应的解码方法，又可以实现把原始数据还原出来。

本发明所提出的闪存读写编解码算法实现方法比较广泛，可以通过以下描述方法实现：1.通过软件的方法，当主机在向闪存设备发送数据时，直接对数据作编码运算，然后向闪存设备接口发送经过编码后的数据；当主机在读取闪存设备时，直接对接口读取的闪存数据作解码运算，然后再传给其他的存储设备。2.可以在闪存设备中的控制器模块中，加一个编解码模块，通过硬件的方法实现，在写操作过程中，当设备收到设备接口发来的数据后，对数据作编码运算，然后，将编码之后的数据发给闪存芯片进行存储；在读操作过程中，控制器模块将闪存芯片中的数据读出来，接着作解码运算，然后，把解码之后的数据通过接口传给主机。3.也可以在闪存芯片的内部，加一个编解码模块；当闪存芯片收到外部控制器模块发来的数据之后，直接对数据作编码运算，然后，再把编码之后的结果写入相应地址的记忆单元；当控制器模块对闪存芯片做读操作时，闪存芯片内的编解码模块先对从相应地址记忆单元读取的数据作解码运算，然后，再将结果送到控制器模块。编码模块和解码模块可以分别独立设置在不同的设备侧，可以设置的设备侧包括：系统主机侧、闪存控制器侧和闪存芯片侧。

如图 1 所示为本发明在闪存控制器侧进行编解码处理的实施例原理图；图中 11 所示为设备内数据缓存，其主要用来在操作过程中缓存数据；图中 12 所示为在设备写操作过程中，数据从数据缓存模块写入数据编码模块；图中 13 所示为编解码模块中的编码模块，其主要作用是对缓存中写入的数据做编码运算，并将相应编码信息写入冗余区指定位；图中 14 所示为数据经过编解码器中的编码模块处理后写入闪存控制模块；图中 15 所示为闪存控制模块，主要用来控制闪存的操作，同时将编码后数据传给闪存芯片；图中 18 所示为读操作过程中，闪存控制器将读取的闪存芯片内存储数据传给解码模块；图中 17 所示为编解码器中的解码模块，其主要作用是根据数据冗余区中记录的编码算法，对数据区的数据作相应的解码运算，然后如图中 16 将解码后的结果传给数据缓存。

图 2 所示本发明实施例算法对某页面写操作过程中的编码操作，如图所示，将数据操作页面大小设为 8Byte，冗余区设为 1Byte，即每次闪存写

操作单位为 9 个字节，假设任意选定一个页面，其中的数据值如图中左边方框内所示：“01100000 10000100 01000100 10101001 01001001 00101000 00000100 00100001”，冗余区为“xxxxxxx1”，其中冗余区的前 7 位记录了其他的数据信息，第 8 位是指定的编码信息，经过编码模块的统计其中 0 的个数为 46，1 的个数为 18，经过其判断，需要对数据进行编码运算，经过本发明实施例的算法，得到图右方框内数据所示：“10011111 01111011 10111011 01010110 10110110 11010111 11111011 11011110”，并将编码信息写入冗余区“xxxxxxx0”，经过编码模块对数据区的统计其中 0 的个数为 18，1 的个数为 46，将数据写入闪存芯片，同时，将冗余区内指定位写入相应标记（这里我们定义数据位为 0 时，代表经过求反运算；数据位为 1 时，表示没有经过求反运算），来记录页面数据经过求反运算。

图 3 所示为本发明实施例算法对某页面记录数据读操作过程中的解码操作，如图所示，根据图 4 写操作过程中写入的冗余区标志位信息，解码模块对数据进行解码运算，读出原来写入设备的数据。

图 4 所示本发明实施例中某页面写入操作和最终擦除操作闪存芯片内的数据区记忆单元位变化情况，首先在写入数据之前需要先对要操作的该地址闪存页面做擦除操作，擦作之后如图中左方框所示，记忆单元都变成 1；然后，将数据写入，在写入过程中如果该位置数值为 1，则不需要对该位置存储单元充电，如果为 0，则需要充电操作，对应的将 0 位充电写入，如图中间方框内所示；最后对写入数据做擦除操作，则将所有位放电，如果位为 1，则在本次写和擦除操作没有被损耗，如果为 0，则在本次写和擦除操作被损耗 1 次。

经过统计，如果没有经过本发明实施例编解码运算如图 5 所示，数据区闪存记忆单元将被损耗 46 次，经过本发明实施例算法后，闪存记忆单元被损耗 18 次，大大降低了该页面记忆单元被损耗的次数，因此，通过统计运算，整个闪存芯片以及闪存设备都能够有效地延长使用寿命，同时，根据闪存写入和擦除操作特点，每次操作过程中写入或者擦除的数据中 0 的个数越少，操作时间越短，效率就会越高，同时，0 的个数越少，操作需要的能耗也越少。

上例中以一种求反运算这样一种最简单，快捷的算法为例，对本发明作了说明性的描述。此外，本发明所涉及的算法和实现方法可以包括很多，

既可以通过软件方法实现，也可以通过硬件方法实现，其目的最终都是尽可能的减少对闪存记忆单元产生操作，将写入闪存芯片中的数据对闪存芯片的损耗尽可能的减小，从而达到延长闪存芯片以及闪存设备的使用寿命，尽量缩短操作时间，提高闪存操作速率，以及减小闪存操作的功耗等优化的目的。

以下再介绍一种映射的编解码处理方式。如图 6 所示，本实施例以 4 位数据为例，图中为所有的 4 位数据的集合，其中数据中 0 个 0 的有 1 个，1 个 0 的有 4 个，2 个 0 的有 6 个，3 个 0 的有 4 个，4 个 0 的有 1 个。

如图 7 所示，为增加 2 位冗余区之后的所有数据的组合，其中包含 0 个 0 的有 1 个，1 个 0 的有 6 个，2 个 0 的有 15 个，3 个 0 的有 20 个，4 个 0 的有 15 个，5 个 0 的有 6 个，6 个 0 的有 1 个。

如图 8 所示，对 4 位数据组合和 6 位包含冗余区的数据组合之间建立一种新的映射关系，根据这种映射关系在数据写入过程中对数据进行编码运算之后，可以大大的减少数据中 0 的个数，图中经过影射之后的 6 位数据中，0 的个数为 0 的有 1 个，0 的个数为 1 的有 6 个，0 的个数为 2 的有 9 个。

对上述结果进行一个统计：由于数据是随机的产生，所以组合中的每一个值被记录进闪存芯片的几率是一样的，假设组合中的每个值被写入 n 次，则 0 被写入的次数为 4n+12n+12n+4n=32n；经过编码运算之后，实际写入的 0 的次数为 6n+18n=24n。经过统计，一共少写入 8n 次 0，进行了 16n 次 4 位写操作，所以经过本实施例写操作对于每位可以节省的损耗为 12.5%。

数据的读取过程中，从闪存存储单元读取数据编码后的码值，根据映射关系进行解码运算，其中解码是编码的逆运算，得到对应的原数数据。

根据上述的映射原理，每个闪存页面包括 2048B 的数据区和 64B 的冗余区，可以建立一种编码运算，其运算输入输出的数据组合分别为主机数据和需要记录入闪存记录单元的编码数据，由于冗余区的存在，所以编码后的数据位数大于主机原始数据的位数，这种编码运算将主机数据和位数扩大之后的数据组合建立新的映射关系，使得进行编码运算之后的数据 0 的个数小于主机原始数据中 0 的个数，从而达到减少对闪存芯片的损耗，

延长闪存设备寿命的目的。

以下对于本发明实施例采用的压缩编码方式进行详细说明。本发明采用一种编码做为实施例，其实施方法非常简单，统计一段数据中 16 进制数据出现的频率，按照出现频率来决定编码的位数，频率出现最多的码值为 0，接下来根据频率依次为 10，110，1110，…每次多一位，最高位增加一个 1；当遇到频率相同时，根据数据对应的 16 进制值的大小来决定码值，值小的，码值位数少，值大的，码值位数多。

如图 9 所示为未经过压缩运算的 128bit 原始数据，以 4bit 为运算单元进行压缩运算，经过统计，数据中 4 位数据单元所有组合的个数如图 10 所示，根据统计结果的数据写入频率设计一种压缩编码，其中数据与压缩编码对应关系如图 10 所示。经过统计，未经压缩运算的原始数据中 0 的个数为 72 个，1 的个数为 56 个。

将 128bit 原始数据进行压缩编码运算之后，可得到如图 11 所示的结果，压缩运算之后数据位数为 125bit 其中 0 的个数为 32 个，1 的个数为 93 个。与压缩前的数据作对比，可以得到，0 的个数减少了 40 个，数据总数减少了 3 个 bit，因此通过这种压缩编码运算可以有效地减少数据中 0 的个数，同时减少写入的数据位。

在对数据进行读取的操作过程中，先从闪存单元读取写入的数据编码，根据压缩运算中压缩码与数据的对应关系进行解码，其中解码运算是编码的逆运算，得到相应原始数据。

根据上述压缩编解码算法，其实施方法可以有很多，这里仅提供一种为例，其主要目的是通过压缩算法减少写入的数据位数，特别是数据中 0 的位数，从而达到减少闪存损耗，延长设备寿命以及优化写入速度，降低功耗的目的。

本发明中仅介绍了几种实施例做为说明，可实现的算法有很多，都是在本发明主导思想之下，这些对于技术人员来说都是显然的。此外，本发明提出的存储芯片不仅仅包括 NAND，NOR 等闪存芯片，其他的有相似写入损耗的半导体类存储芯片都不能认为是脱离本发明的主题思想和使用范围，对于以上这些对技术人员来说是显然的改变，都包含在本发明的范围内。

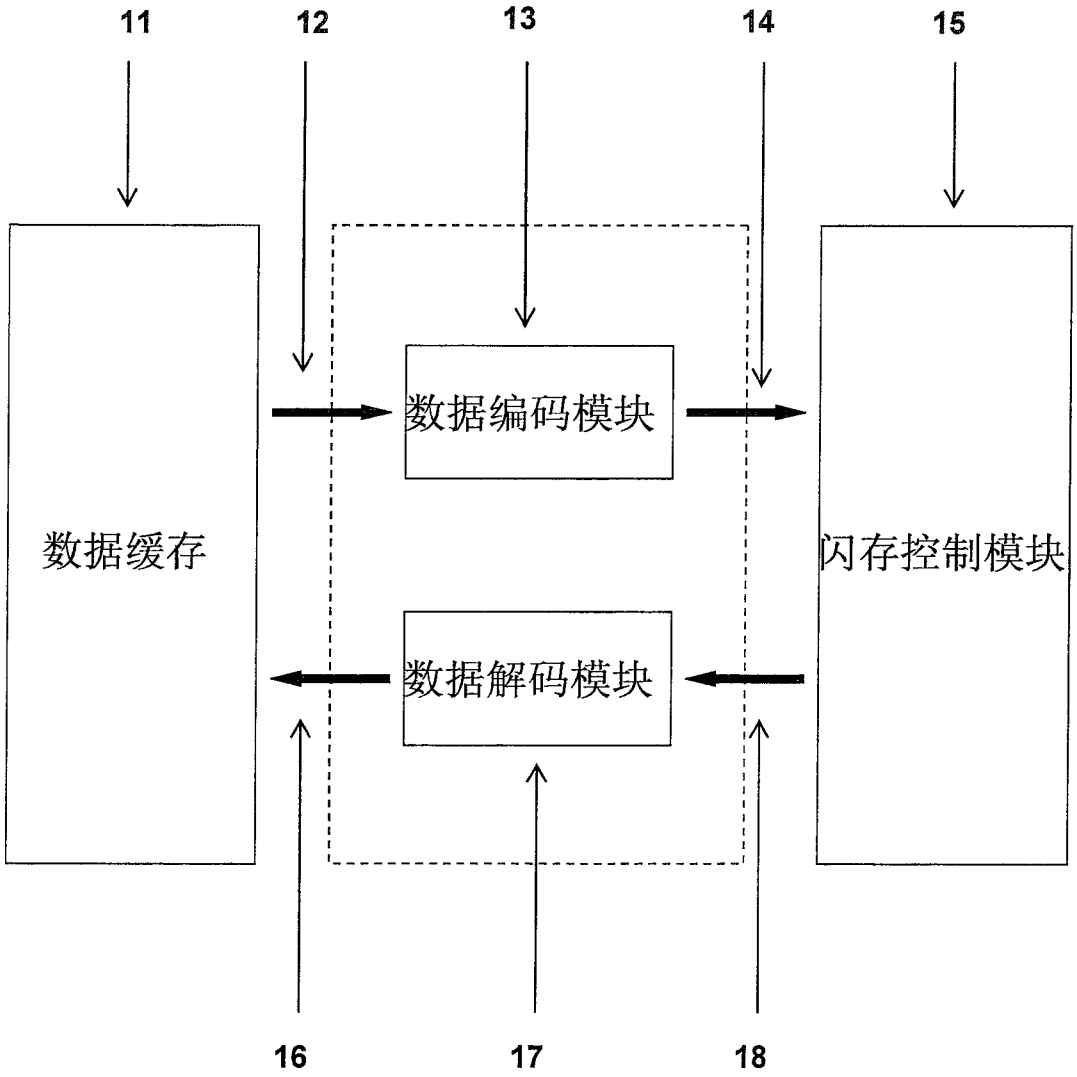


图 1

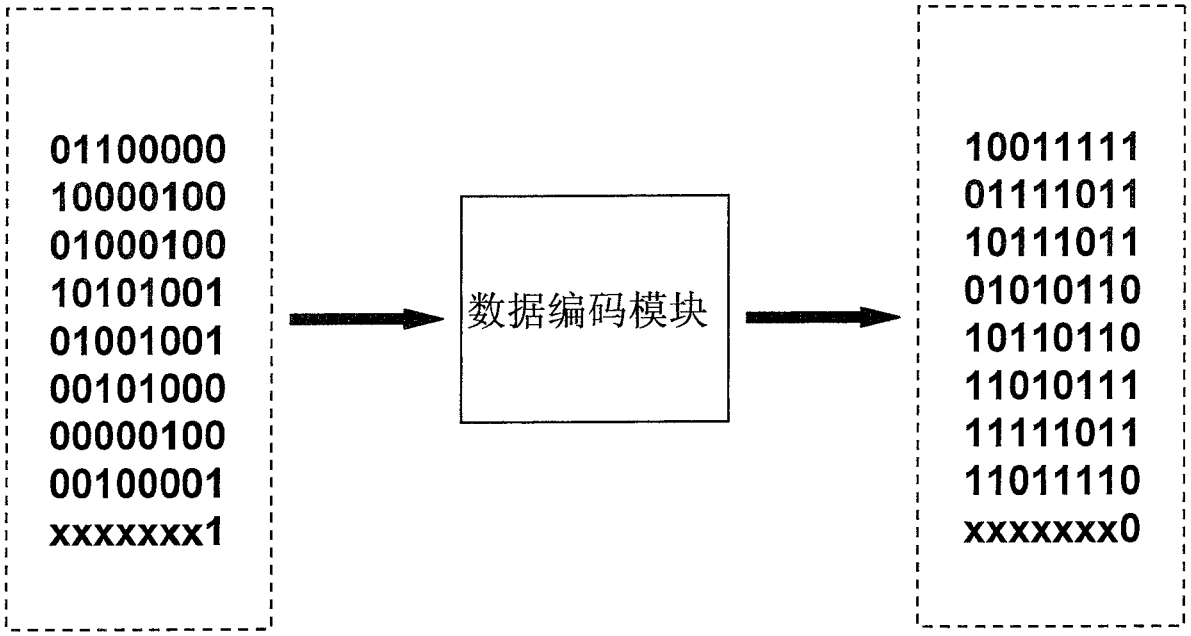

01100000
10000100
01000100
10101001
01001001
00101000
00000100
00100001
xxxxxxx1
数据编码模块
10011111
01111011
10111011
01010110
10110110
11010111
11111011
11011110
xxxxxxx0

图 2

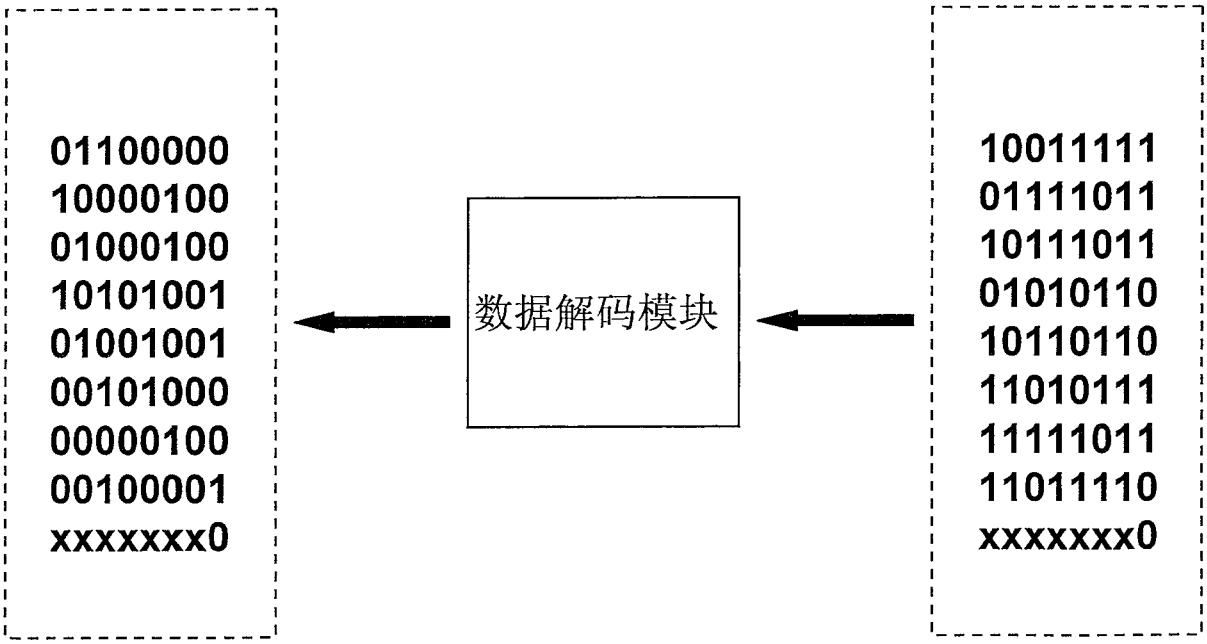

01100000
10000100
01000100
10101001
01001001
00101000
00000100
00100001
xxxxxxx0
数据解码模块
10011111
01111011
10111011
01010110
10110110
11010111
11111011
11011110
xxxxxxx0

图 3

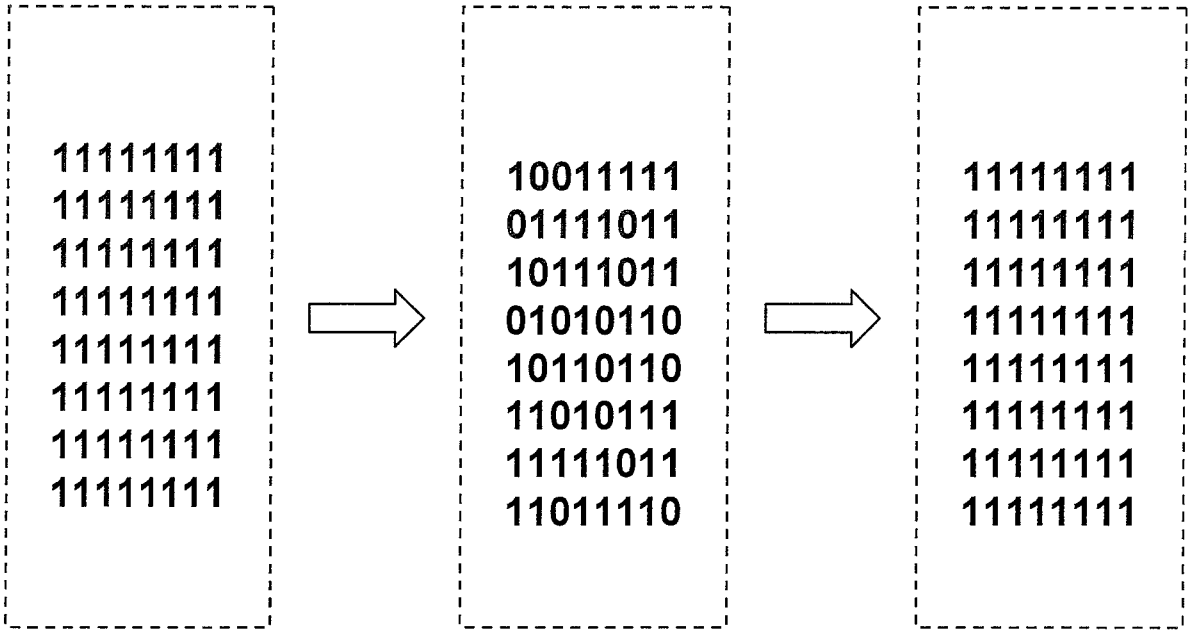

11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
10011111
01111011
10111011
01010110
10110110
11010111
11111011
11011110
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111

图 4

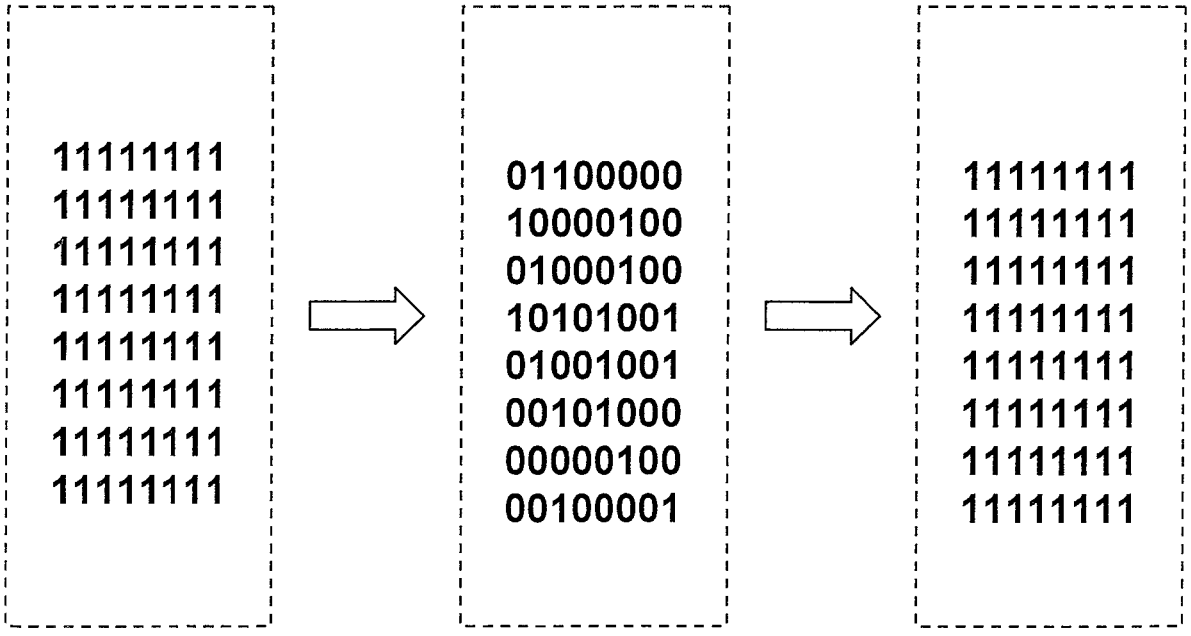

11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
01100000
10000100
01000100
10101001
01001001
00101000
00000100
00100001
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111

图 5

0000	1001
0001	1010
0010	0101
0100	0111
1000	1011
0011	1101
0110	1110
1100	1111

图 6

0000	11	1001	11
0001	11	1010	11
0010	11	0101	11
0100	11	0111	11
1000	11	1011	11
0011	11	1101	11
0110	11	1110	11
1100	11	1111	11

0000	01	1001	01
0001	01	1010	01
0010	01	0101	01
0100	01	0111	01
1000	01	1011	01
0011	01	1101	01
0110	01	1110	01
1100	01	1111	01

0000	10	1001	10
0001	10	1010	10
0010	10	0101	10
0100	10	0111	10
1000	10	1011	10
0011	10	1101	10
0110	10	1110	10
1100	10	1111	10

0000	00	1001	00
0001	00	1010	00
0010	00	0101	00
0100	00	0111	00
1000	00	1011	00
0011	00	1101	00
0110	00	1110	00
1100	00	1111	00

图 7

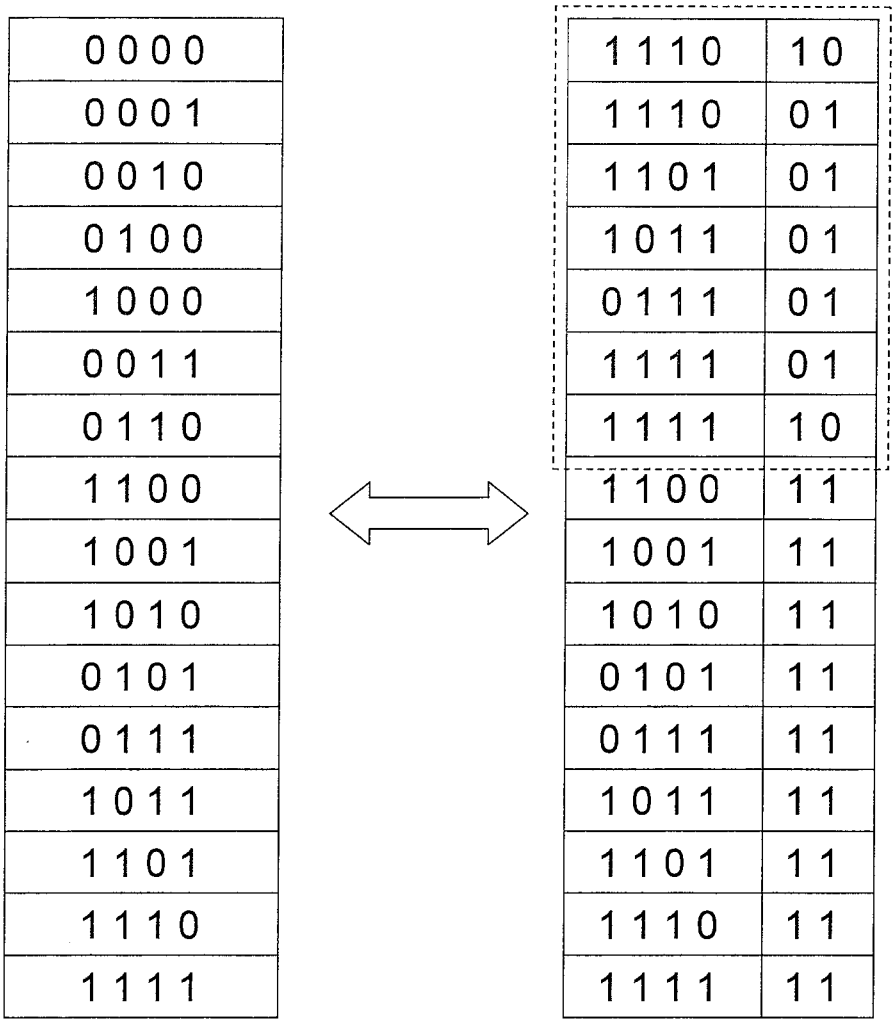


图 8

0011	0100	0101	0110
0001	1010	1010	1010
0010	1010	0100	0110
0100	1011	1010	0001
1000	0011	0100	0110
0011	1010	0011	0101
0110	1001	0100	1010
1100	1010	1001	0101

图 9

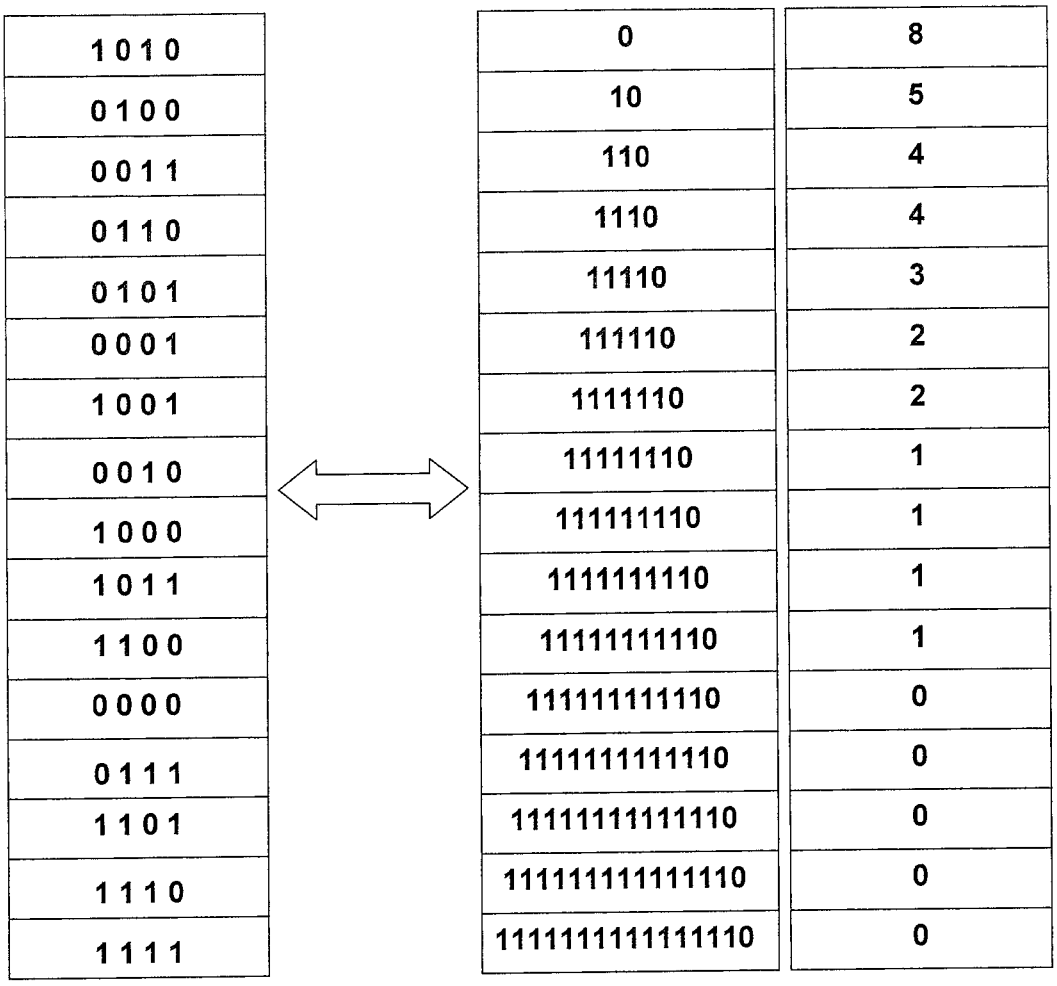


图 10

1101	0111	1011	1011
1110	0001	1111	1100
1011	1010	1111	1111
1001	1111	0111	1111
1011	0101	1101	1001
1011	1101	1101	1111
1010	0111	1111	1110
0111	1110	1111	0

图 11