(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104885052 A
(43)申请公布日 2015.09.02

(21)申请号 201380066018.5

(22)申请日 2013.11.07

(30)优先权数据
13/720,532 2012.12.19 US

(85)PCT国际申请进入国家阶段日
2015.06.17

(86)PCT国际申请的申请数据
PCT/US2013/068939 2013.11.07

(87)PCT国际申请的公布数据
WO2014/099169 EN 2014.06.26

(71)申请人 高通股份有限公司
地址 美国加利福尼亚

(72)发明人 D·P·阿里 B·巴布
T·N·埃拉拉 A·库马尔

(74)专利代理机构 永新专利商标代理有限公司
72002
代理人 张立达 王英

(51)Int.Cl.
G06F 3/06(2006.01)
G06F 12/02(2006.01)
G11C 29/00(2006.01)

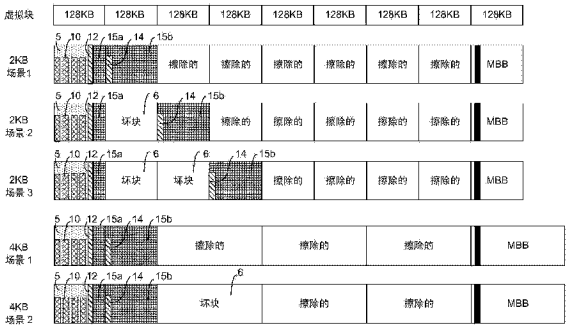
权利要求书8页 说明书14页 附图10页

(54)发明名称

读写存储设备的数据映像中的虚拟边界码

(57)摘要

本发明提供了用于配置具有数据映像的读写存储设备的方法、系统和设备。该方法包括：基于针对读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布。将该数据映像分割成一个或多个数据映像部分，其中向这些数据映像部分中的至少一个添加虚拟边界码。将这些数据映像部分存储在上述一系列的虚拟块的各个虚拟块中，跳过该读写存储设备中的任何坏块（即使是虚拟块之间）。



CN 104885052 A

1. 一种在读写存储设备上存储数据映像的方法，所述方法包括：

基于针对第一读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布，所述第一读写存储设备包括一系列实际块；

向数据映像第一部分添加虚拟边界码，所述虚拟边界码代表用于定位所述数据映像第一部分的标记；以及

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在所述第一读写存储设备上的所述一系列虚拟块中的虚拟块里。

2. 根据权利要求 1 所述的方法，其中，所述虚拟边界码被添加在所述数据映像第一部分的开始处。

3. 根据权利要求 1 所述的方法，其中，所述一系列实际块中的每一个实际块的大小可被所述虚拟块大小整除，而没有余数。

4. 根据权利要求 1 所述的方法，还包括：

将数据映像第二部分存储在所述第一读写存储设备上的所述一系列虚拟块中的另一个虚拟块里，其中，所述数据映像被分割成一个以上的部分，所述一个以上的部分包括所述数据映像第一部分和所述数据映像第二部分。

5. 根据权利要求 4 所述的方法，其中，所述第一读写存储设备中的坏块位于所述数据映像第一部分和所述数据映像第二部分之间。

6. 根据权利要求 4 所述的方法，其中，所述数据映像第一部分和所述数据映像第二部分被存储在所述第一读写存储设备上的所述一系列实际块中的共享实际块里。

7. 根据权利要求 4 所述的方法，其中，所述数据映像第二部分占据所述第一读写存储设备的块 0。

8. 根据权利要求 4 所述的方法，其中，所述数据映像第二部分包括映像头，其中所述映像头包含关于以下各项中的至少一项的信息：所述第一读写存储设备上的所述虚拟边界码的位置以及所述数据映像的大小。

9. 根据权利要求 4 所述的方法，还包括：

向数据映像第三部分添加虚拟边界码；以及

将所述数据映像第三部分与添加到其的所述虚拟边界码存储在所述一系列虚拟块中的第三虚拟块里。

10. 根据权利要求 1 所述的方法，还包括：

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在第二读写存储设备上的虚拟块中，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的。

11. 一种用于在读写存储设备上存储数据映像的计算设备，所述计算设备包括：

存储器；以及

处理器，其耦合到所述存储器并且配置有处理器可执行指令以执行操作，所述操作包括：

基于针对第一读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布，所述第一读写存储设备包括一系列实际块；

向数据映像第一部分添加虚拟边界码，所述虚拟边界码代表用于定位所述数据映像第

一部分的标记；以及

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在所述第一读写存储设备上的所述一系列虚拟块中的虚拟块里。

12. 根据权利要求 11 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述虚拟边界码被添加在所述数据映像第一部分的开始处。

13. 根据权利要求 11 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述一系列实际块中的每一个实际块的大小可被所述虚拟块大小整除，而没有余数。

14. 根据权利要求 11 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

将数据映像第二部分存储在所述第一读写存储设备上的所述一系列虚拟块中的另一个虚拟块里，其中，所述数据映像被分割成一个以上的部分，所述一个以上的部分包括所述数据映像第一部分和所述数据映像第二部分。

15. 根据权利要求 14 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述第一读写存储设备中的坏块位于所述数据映像第一部分和所述数据映像第二部分之间。

16. 根据权利要求 14 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述数据映像第一部分和所述数据映像第二部分被存储在所述第一读写存储设备上的所述一系列实际块中的共享实际块里。

17. 根据权利要求 14 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述数据映像第二部分占据所述第一读写存储设备的块 0。

18. 根据权利要求 14 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，使得所述数据映像第二部分包括映像头，其中所述映像头包含关于以下各项中的至少一项的信息：所述第一读写存储设备上的所述虚拟边界码的位置以及所述数据映像的大小。

19. 根据权利要求 14 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

向数据映像第三部分添加虚拟边界码；以及

将所述数据映像第三部分与添加到其的所述虚拟边界码存储在所述一系列虚拟块中的第三虚拟块里。

20. 根据权利要求 11 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在第二读写存储设备上的虚拟块中，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的。

21. 一种用于在读写存储设备上存储数据映像的计算设备，所述计算设备包括：

用于基于针对第一读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布的单元，所述第一读写存储设备包括一系列实际块；

用于向数据映像第一部分添加虚拟边界码的单元，其中所述虚拟边界码代表用于定位

所述数据映像第一部分的标记；以及

用于将所述数据映像第一部分与添加到其的所述虚拟边界码存储在所述第一读写存储设备上的所述一系列虚拟块中的虚拟块里的单元。

22. 根据权利要求 21 所述的计算设备，其中，所述虚拟边界码被添加在所述数据映像第一部分的开始处。

23. 根据权利要求 21 所述的计算设备，其中，所述一系列实际块中的每一个实际块的大小可被所述虚拟块大小整除，而没有余数。

24. 根据权利要求 21 所述的计算设备，还包括：

用于将数据映像第二部分存储在所述第一读写存储设备上的所述一系列虚拟块中的另一个虚拟块里的单元，其中，所述数据映像被分割成一个以上的部分，所述一个以上的部分包括所述数据映像第一部分和所述数据映像第二部分。

25. 根据权利要求 24 所述的计算设备，其中，所述第一读写存储设备中的坏块位于所述数据映像第一部分和所述数据映像第二部分之间。

26. 根据权利要求 24 所述的计算设备，其中，将所述数据映像第一部分和所述数据映像第二部分被存储在所述第一读写存储设备上的所述一系列实际块中的共享实际块里。

27. 根据权利要求 24 所述的计算设备，其中，所述数据映像第二部分占据所述第一读写存储设备的块 0。

28. 根据权利要求 24 所述的计算设备，其中，所述数据映像第二部分包括映像头，其中所述映像头包含关于以下各项中的至少一项的信息：所述第一读写存储设备上的所述虚拟边界码的位置以及所述数据映像的大小。

29. 根据权利要求 24 所述的计算设备，还包括：

用于向数据映像第三部分添加虚拟边界码的单元；以及

用于将所述数据映像第三部分与添加到其的所述虚拟边界码存储在所述一系列虚拟块中的第三虚拟块里的单元。

30. 根据权利要求 21 所述的计算设备，还包括：

用于将所述数据映像第一部分与添加到其的所述虚拟边界码存储在第二读写存储设备上的虚拟块中的单元，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的。

31. 一种其上存储有处理器可执行软件指令的非临时性计算机可读存储介质，其中所述处理器可执行软件指令被配置为使处理器执行操作，以管理具有存储器的计算设备上的存储器，所述操作包括：

基于针对第一读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布，所述第一读写存储设备包括一系列实际块；

向数据映像第一部分添加虚拟边界码，所述虚拟边界码代表用于定位所述数据映像第一部分的标记；以及

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在所述第一读写存储设备上的所述一系列虚拟块中的虚拟块里。

32. 根据权利要求 31 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述虚拟边界码被添加在所述数据映像第

一部分的开始处。

33. 根据权利要求 31 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述一系列实际块中的每一个实际块的大小可被所述虚拟块大小整除，而没有余数。

34. 根据权利要求 31 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

将数据映像第二部分存储在所述第一读写存储设备上的所述一系列虚拟块中的另一个虚拟块里，其中，数据映像被分割成一个以上的部分，所述一个以上的部分包括所述数据映像第一部分和所述数据映像第二部分。

35. 根据权利要求 34 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述第一读写存储设备中的坏块位于所述数据映像第一部分和所述数据映像第二部分之间。

36. 根据权利要求 34 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述数据映像第一部分和所述数据映像第二部分被存储在所述第一读写存储设备上的所述一系列实际块中的共享实际块里。

37. 根据权利要求 34 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述数据映像第二部分占据所述第一读写存储设备的块 0。

38. 根据权利要求 34 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，使得所述数据映像第二部分包括映像头，其中所述映像头包含关于以下各项中的至少一项的信息：所述第一读写存储设备上的所述虚拟边界码的位置以及所述数据映像的大小。

39. 根据权利要求 34 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

向数据映像第三部分添加虚拟边界码；以及

将所述数据映像第三部分与添加到其的所述虚拟边界码存储在所述一系列虚拟块中的第三虚拟块里。

40. 根据权利要求 31 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

将所述数据映像第一部分与添加到其的所述虚拟边界码存储在第二读写存储设备上的虚拟块中，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的。

41. 一种从读写存储设备中读取数据映像的方法，包括：

访问第一读写存储设备的第一虚拟块中的第一数据，所述第一读写存储设备的实际块大小可被虚拟块大小整除，而没有余数；

针对指定所述第一虚拟块之后的第二虚拟块的第二虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第二虚拟块中的所述虚拟边界码，访问所述第二虚拟块中的第二数据；以及

基于所述第一数据和所述第二数据来读取第一设备数据映像。

42. 根据权利要求 41 所述的方法，还包括：

响应于确定所述第一数据和所述第二数据不包括所述数据映像的完整映像，针对指定所述第二虚拟块之后的第三虚拟块的第三虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第三虚拟块中的虚拟边界码，访问所述第三虚拟块中的第三数据；以及

进一步基于所述第三数据来读取所述数据映像。

43. 根据权利要求 41 所述的方法，还包括：

基于所述第一数据和所述第二数据来执行引导装载程序。

44. 根据权利要求 41 所述的方法，还包括：

响应于在所述实际块中没有检测到所述虚拟边界码，跳过所述第一读写存储设备的、所述第一虚拟块之后的实际块。

45. 根据权利要求 41 所述的方法，还包括：

响应于在所述第一读写存储设备上的、与所述第一虚拟块的大小相对应的中间距离中没有检测到所述虚拟边界码，跳过所述中间距离。

46. 根据权利要求 45 所述的方法，还包括：

扫描所述第一虚拟块中的所述第一数据里包含的头信息；以及

基于所述头信息，来确定所述数据映像和所述虚拟块大小中的至少一个的大小。

47. 根据权利要求 41 所述的方法，还包括：

访问第二读写存储设备的第三虚拟块中的第三数据，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的；

针对所述虚拟边界码，扫描所述第二读写存储设备；

响应于识别出所述虚拟边界码，访问所述第二读写存储设备的第四虚拟块中的第四数据；以及

基于所述第三数据和所述第四数据来读取第二设备数据映像。

48. 一种计算设备，包括：

存储器；以及

处理器，其耦合到所述存储器并且配置有处理器可执行指令以执行操作，所述操作包括：

访问第一读写存储设备的第一虚拟块中的第一数据，所述第一读写存储设备的实际块大小可被虚拟块大小整除，而没有余数；

针对指定所述第一虚拟块之后的第二虚拟块的第二虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第二虚拟块中的所述虚拟边界码，访问所述第二虚拟块中的第二数据；以及

基于所述第一数据和所述第二数据来读取第一设备数据映像。

49. 根据权利要求 48 所述的计算设备，其中，所述处理器配置有处理器可执行指令以

执行操作，所述操作还包括：

响应于确定所述第一数据和所述第二数据不包括数据映像的完整映像，针对指定所述第二虚拟块之后的第三虚拟块的第三虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第三虚拟块中的所述虚拟边界码，访问所述第三虚拟块中的第三数据；以及

进一步基于所述第三数据来读取所述数据映像。

50. 根据权利要求 48 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

基于所述第一数据和所述第二数据来执行引导装载程序。

51. 根据权利要求 48 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

响应于在所述实际块中没有检测到所述虚拟边界码，跳过所述第一读写存储设备的、所述第一虚拟块之后的实际块。

52. 根据权利要求 48 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

响应于在所述第一读写存储设备上的、与所述第一虚拟块的大小相对应的中间距离中没有检测到所述虚拟边界码，跳过所述中间距离。

53. 根据权利要求 52 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

扫描所述第一虚拟块中的所述第一数据里包含的头信息；以及

基于所述头信息，来确定所述数据映像和所述虚拟块大小中的至少一个的大小。

54. 根据权利要求 48 所述的计算设备，其中，所述处理器配置有处理器可执行指令以执行操作，所述操作还包括：

访问第二读写存储设备的第三虚拟块中的第三数据，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的；

针对所述虚拟边界码，扫描所述第二读写存储设备；

响应于识别出所述虚拟边界码，访问所述第二读写存储设备的第四虚拟块中的第四数据；以及

基于所述第三数据和所述第四数据来读取第二设备数据映像。

55. 一种用于从读写存储设备中读取数据映像的计算设备，所述计算设备包括：

用于访问第一读写存储设备的第一虚拟块中的第一数据的单元，其中，所述第一读写存储设备的实际块大小可被虚拟块大小整除，而没有余数；

用于针对指定所述第一虚拟块之后的第二虚拟块的第二虚拟块边界的虚拟边界码，扫描所述第一读写存储设备的单元；

用于响应于识别出所述第二虚拟块中的所述虚拟边界码，访问所述第二虚拟块中的第二数据的单元；以及

用于基于所述第一数据和所述第二数据来读取第一设备数据映像的单元。

56. 根据权利要求 55 所述的计算设备，还包括：

用于响应于确定所述第一数据和所述第二数据不包括所述数据映像的完整映像，针对指定所述第二虚拟块之后的第三虚拟块的第三虚拟块边界的虚拟边界码，扫描所述第一读写存储设备的单元；

用于响应于识别出所述第三虚拟块中的虚拟边界码，访问所述第三虚拟块中的第三数据的单元；以及

用于进一步基于所述第三数据来读取所述数据映像的单元。

57. 根据权利要求 55 所述的计算设备，还包括：

用于基于所述第一数据和所述第二数据来执行引导装载程序的单元。

58. 根据权利要求 55 所述的计算设备，还包括：

用于响应于在所述实际块中没有检测到所述虚拟边界码，跳过所述第一读写存储设备的、所述第一虚拟块之后的实际块的单元。

59. 根据权利要求 55 所述的计算设备，还包括：

用于响应于在所述第一读写存储设备上的、与所述第一虚拟块的大小相对应的中间距离中没有检测到所述虚拟边界码，跳过所述中间距离的单元。

60. 根据权利要求 59 所述的计算设备，还包括：

用于扫描所述第一虚拟块中的所述第一数据里包含的头信息的单元；以及

用于基于所述头信息，来确定所述数据映像和所述虚拟块大小中的至少一个的大小的单元。

61. 根据权利要求 55 所述的计算设备，还包括：

用于访问第二读写存储设备的第三虚拟块中的第三数据的单元，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的；

用于针对所述虚拟边界码，扫描所述第二读写存储设备的单元；

用于响应于识别出所述虚拟边界码，访问所述第二读写存储设备的第四虚拟块中的第四数据的单元；以及

用于基于所述第三数据和所述第四数据来读取第二设备数据映像的单元。

62. 一种其上存储有处理器可执行软件指令的非临时性计算机可读存储介质，其中所述处理器可执行软件指令被配置为使处理器执行操作，以读取具有存储器的计算设备上的存储器，所述操作包括：

访问第一读写存储设备的第一虚拟块中的第一数据，所述第一读写存储设备的实际块大小可被虚拟块大小整除，而没有余数；

针对指定所述第一虚拟块之后的第二虚拟块的第二虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第二虚拟块中的所述虚拟边界码，访问所述第二虚拟块中的第二数据；以及

基于所述第一数据和所述第二数据来读取第一设备数据映像。

63. 根据权利要求 62 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

响应于确定所述第一数据和所述第二数据不包括所述数据映像的完整映像，针对指定所述第二虚拟块之后的第三虚拟块的第三虚拟块边界的虚拟边界码，扫描所述第一读写存储设备；

响应于识别出所述第三虚拟块中的所述虚拟边界码，访问所述第三虚拟块中的第三数据；以及

进一步基于所述第三数据来读取所述数据映像。

64. 根据权利要求 62 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

基于所述第一数据和所述第二数据来执行引导装载程序。

65. 根据权利要求 62 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

响应于在所述实际块中没有检测到所述虚拟边界码，跳过所述第一读写存储设备的、所述第一虚拟块之后的实际块。

66. 根据权利要求 62 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

响应于在所述第一读写存储设备上的、与所述第一虚拟块的大小相对应的中间距离中没有检测到所述虚拟边界码，跳过所述中间距离。

67. 根据权利要求 66 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

扫描所述第一虚拟块中的所述第一数据里包含的头信息；以及

基于所述头信息，来确定数据映像和所述虚拟块大小中的至少一个的大小。

68. 根据权利要求 62 所述的非临时性计算机可读存储介质，其中，所存储的处理器可执行软件指令被配置为使处理器执行操作，所述操作还包括：

访问第二读写存储设备的第三虚拟块中的第三数据，其中，特定于页、块和设备的坏块算法特性中的至少一个，在所述第一读写存储设备和所述第二读写存储设备之间是不同的；

针对所述虚拟边界码，扫描所述第二读写存储设备；

响应于识别出所述虚拟边界码，访问所述第二读写存储设备的第四虚拟块中的第四数据；以及

基于所述第三数据和所述第四数据来读取第二设备数据映像。

读写存储设备的数据映像中的虚拟边界码

背景技术

[0001] 闪存是可以进行电擦除和重新编程的非易失性计算机存储芯片。NAND 闪存（还称为 NAND 存储器）是可以按照块或页来编程和读取的高密度型的读写存储器。NAND 存储器用于存储卡、U 盘、固态硬盘和类似产品中，以进行通常的存储和数据传输，以及在众多的数字设备中用于存储配置数据。

[0002] 只读存储器 (ROM) 是在计算机和其它电子设备中使用的一种类型的存储介质。传统的 ROM 是只读存储器，不能够进行修改，或者只能慢速地或困难地进行修改，所以其主要用于分配固件。固件是指专门被设计用于特定硬件的软件。设备的固件可能很少或者从未在其经济寿命期间被改变。更新固件的通常原因包括 :修复问题或者向使用它的硬件添加功能特征。用于更新固件的一种方式是利用结合 ROM 使用的特殊程序闪存进行重新编程。与 ROM 不同，NAND 存储器可以被多次擦除和重新编程。

[0003] NAND 存储器通常被以多个块来进行组织，每个块包含多个页。当 NAND 存储器的实际的页或块的长度由制造商进行设置时，它就被永久固定在该长度。“块”（由于其涉及计算机存储器，特别是 NAND 存储器）包括具有标称长度的存储字节或比特序列。该长度称为块大小。将数据存储到块中的过程，通常是一次完成整个页，而擦除数据则是以块为单位进行。

[0004] 每个块还可以被组织成多个页，其中页的大小通常为 2,048 字节（～ 2KB) 或 4,096 字节（～ 4KB)。与每一个块的至少一页相关联的是少许字节（通常是数据大小的 1/32)，该少许字节可以用于存储纠错码 (ECC) 校验和。典型的块大小包括 :

[0005] •64 页，每一页具有 2,048 字节（～ 2KB) 和 64 字节的备用区，其意味着 128KB（对应于页数据）加 4KB（对应于诸如纠错码、坏块信息、元数据等等之类的备用数据）的块大小 ;

[0006] • 64 页，每一页具有 4,096 字节（～ 4KB) 和 128 字节的备用区，其意味着对应于页数据的 256KB 的块大小 ;以及

[0007] •128 页，每一页具有 2,048 字节（～ 2KB) 和 128 字节的备用区，其意味着对应于页数据的 256KB 的块大小。

[0008] 在 NAND 存储器的制造中经常发生制造缺陷，其可能使一些存储单元不具有功能和不可用。包括有缺陷的存储单元的块称为“坏块”。由于制造过程，许多 NAND 设备出厂时就具有一些坏块。例如，通过允许一些坏块，与所有块都必须被验证是好的相比，制造商实现了高得多的产量。这显著地降低了 NAND 闪存成本，并且只稍微地减少这些部件的存储容量。尽管如此，NAND 存储器的制造一般保证初始块 (Block 0) 是好的，但并不保证剩余的数据块。

[0009] 一些坏块管理方案在 NAND 存储器上设置用于标识该芯片上的坏块的信息。这样的坏块信息可以包括 :指示要访问的用于存储映像的好块或者要略过的坏块的列表。但是，NAND 存储器自身的特性的变化，以及特定于经销商和 / 或制造商的坏块检测方案 / 格式，往往需要定制的 NAND 存储器处理和编程。

[0010] 另外，计算设备中的 ROM 固件设计与软件相比受到某种程度的限制，这是由于一旦该计算设备被制造完成，则这些例行程序在稍后的开发周期中几乎不能被改写，并且通常是永久设定的。从而，由于闪存上的数据映像的全部或一部分可以被改写并因此进行更新，所以闪存被用来运行引导加载例程。如本文所使用的，术语“数据映像”是指读写存储设备（例如，NAND 存储设备）的一个或多个存储单元中所保存的信息。对于趋于复杂的当代引导装载程序来说，数据映像会耗尽闪存，这是由于它们试图适应特定于 NAND 设备的特性，而如上所述，这些特性在经销商和 / 或制造商之间有很大的不同。

发明内容

[0011] 各个实施例包括在诸如闪存之类的读写存储器上存储数据映像，其中该读写存储器接纳坏块，并使计算设备使用单一算法而不管存储器制造商，在从该存储器读取数据时能够跳过坏块。该方法包括：配置数据映像与数据的好块的边界处的码或特殊值一起保存在读写存储器中。本文将这些码或特殊值称为“虚拟边界码”，当计算设备识别出这些码或特殊值时，其指示所标识的块包含有效数据。可以将数据映像以一个或多个虚拟块的方式保存在存储器中，其中所述一个或多个虚拟块具有与指示虚拟块的起始或结束的虚拟块边界相对应的虚拟块大小。与 NAND 设备的实际块大小相比，每一个虚拟块的大小可以更小或者相等。可以将虚拟块大小设置成预期该产品将支持的最低公共实际块大小。用于在 NAND 存储器上存储数据映像的系统，使用该存储器制造商所规定的算法来识别坏块，随后将数据映像存储在好块中，同时跳过这些坏块。虚拟边界码包含于用于存储数据映像的每一个好块的虚拟块边界处。因此，数据映像包括虚拟边界码，其中当计算设备访问读写存储器时，可以查找虚拟边界码以识别要读取的数据块，而无需必须遵循特定于经销商的参数来识别该存储器芯片中的好块或坏块。

[0012] 另一个方面包括可以在计算设备中实现的、用于从读写存储器中读取数据映像的方法。该方法可以包括：访问存储器的第一虚拟块中的数据，并针对虚拟边界码，对该存储设备进行扫描。该虚拟边界码指示第一虚拟块之后的第二虚拟块的边界。此外，该方法还可以包括：响应于识别出虚拟边界码，访问第二虚拟块中的数据。此外，该方法还可以包括：至少基于在第一虚拟块中访问的数据和在第二虚拟块中访问的数据，读取数据映像。此外，该方法还可以包括：响应于在其它虚拟块中识别的虚拟边界码，访问其它虚拟块中的数据。

[0013] 另外的方面包括具有处理器的计算设备，其中该处理器配置有处理器可执行指令，以执行与上面所讨论的方法相对应的各种操作。

[0014] 另外的方面包括计算设备，其中该计算设备具有用于执行与上面所讨论的方法操作相对应的功能的各种单元。

[0015] 另外的方面包括其上存储有处理器可执行指令的非临时性处理器可读存储介质，其中所述处理器可执行指令被配置为使处理器执行与上面所讨论的方法操作相对应的各种操作。

附图说明

[0016] 被并入本文并且构成本说明书一部分的附图，描绘了本发明的示例性实施例，并且连同上面给出的概括描述以及下面给出的详细描述一起来解释本发明的特征。

[0017] 图 1 是根据一个实施例的读写存储设备封装的图。

[0018] 图 2 示出了用于配置图 1 的读写存储设备的处理步骤。

[0019] 图 3 根据一个实施例，示出了五种读写存储设备封装场景。

[0020] 图 4 根据一个实施例，示出了两个另外的读写存储设备封装场景。

[0021] 图 5 是示出用于在读写存储器上保存数据映像的一个方面方法的处理流程图。

[0022] 图 6 是从读写存储设备中读取引导装载程序并进行执行的一个方面方法的处理流程图。

[0023] 图 7 示出了用于在读写存储设备上重写映像的处理步骤。

[0024] 图 8 是在读写存储设备上重写映像的一个方面方法的处理流程图。

[0025] 图 9 是适合于结合各种实施例使用的计算设备的部件图。

[0026] 图 10 是适合于结合各种实施例使用的另一种计算设备的部件图。

[0027] 图 11 是适合于结合各种实施例使用的另一种计算设备的部件图。

具体实施方式

[0028] 现在参照附图来详细地描述各个实施例。在可以的地方，贯穿附图使用相同的附图标记来指代相同或者类似的部件。对于特定示例和实现的引用只是用于说明目的，而不是旨在限制本发明或者权利要求的保护范围。在不脱离本发明的保护范围的基础上，可以设计替代性的实施例。此外，为了避免造成本发明的相关细节的模糊，没有详细描述或者省略了本发明的一些公知单元。

[0029] 本文所使用的“示例性的”一词意味着“用作例子、实例或说明”。本文中描述为“示例性”的任何实施方式不应被解释为比其它实现更优选或更具优势。另外，为了清楚地区分多个描述的组成元素，本文使用“第一”、“第二”、“第三”、“主要”、“辅助”、“第三级”之类的词语或者类似的措词，但其并不旨在将本发明限于元素的特定顺序或者层次。如本文所使用的术语“访问”或“访问到”指代与电子存储设备进行交互的动作，或者当指定其特定部分时，用于扫描或读取其上的数据或信息。此外，如本文所使用的术语“扫描”或“扫描到”意味着检查、解读或查看数据或信息，特别是电子存储设备中的数据或信息。相比而言，如本文所使用的术语“读取”或“读取到”意味着获得、提取、复制或获取数据或信息，特别是电子存储设备中的数据或信息。

[0030] 本文所公开的各个方面提供用于将数据映像存储到读写存储器（例如，闪存），并从这些存储器中读取数据映像的机制，其避免了对定制化的和 / 或特定于设备的坏块检测 / 管理例程的需要。所公开的技术的一个方面包括：将数据映像存储到读写存储设备（例如，闪存设备）的方法。该方法包括：基于针对读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布。这些虚拟块是根据本发明的方面来生成的，而实际块是该读写存储器（如，闪存）的固定参数。读取存储设备的实际块大小可被虚拟块大小整除，而没有余数。用此方式，一个或多个虚拟块将精确地等于实际块。如果数据映像大于虚拟块大小，则将其分割成多个部分，以便适应用于存储该数据映像的读写存储器的块大小。向这些数据映像部分中的至少一个添加虚拟边界码。可以将虚拟边界码添加到数据映像部分的开始处，作为虚拟块的初始边界。数据映像可以是引导启动进程中计算设备所使用的数据和软件的一部分。可以根据虚拟块大小和整个数据映像的大小，按照需要将数据映像分割成尽

可能多的部分。数据映像的全部或一部分，可以仍然存储在 NAND 存储器的块 0 中。此外，该方法还包括：将该数据映像部分存储在所述一系列虚拟块中的一个虚拟块里。在存储数据映像部分时，跳过任何中间的坏块。用此方式，一个或多个坏块被布置在各个虚拟块中所保存的存储的数据映像部分之间。可以将数据映像第一部分和第二部分布置在读写存储设备的共享实际块中。该数据映像的后续部分中的每一个部分可以具有添加到自身的虚拟边界码。这种方式利用虚拟边界码在读写存储器上存储数据映像，其中计算设备可以查找虚拟边界码以定位要读取的存储器的虚拟块，而无需必须依赖好块 / 坏块列表。这避免了对计算设备进行编程，以适应变化非常大的各种好块 / 坏块方案的需求。用此方式，可以将相同的数据映像部分（包括那些具有添加到自身的虚拟边界码的数据映像部分）存储在另外的读写设备上。这些另外的读写设备甚至可以彼此之间具有不同的特定于页、块和 / 或设备的坏块算法特性。

[0031] 计算设备可以实现从读写存储器中读取数据映像的方面的方法，其中该读写存储器利用使用上述方法所生成的数据映像。当使用读写存储器来存储引导加载程序映像时，可以在针对计算设备所执行的引导加载例程中实现该方法。该引导加载程序映像不需要是针对计算设备执行的专有引导加载程序。该方法包括：访问读写存储器的虚拟块中的数据。读取存储设备的实际块大小可被虚拟块大小整除，而没有余数，其包括实际块大小等于虚拟块大小的场景。如果在该虚拟块中不包含被访问的完整的数据映像，则该方法可以在读写存储器中向前扫描以查找虚拟边界码，其中虚拟边界码指示另一个虚拟块的边界（例如，开始）。当识别出虚拟边界码时，计算设备数据装载例程（例如，引导加载例程）可以访问该其它虚拟块中的数据。由于存储器是通过上述方法来装载的，因此虚拟边界码指示虚拟块（在该虚拟块中标识出该虚拟边界码）是好块，并包括可以读取的该数据映像的一部分。在数据的其它块中进行读取之后，如果仍然还没有访问完整的数据映像，则计算设备数据装载例程继续针对另外的虚拟边界码和数据块对存储器进行扫描，直到读取了完整的数据映像为止。可以根据映像头信息来确定数据映像的量 / 大小。当数据装载例程认识到已装载了完整的数据映像时，该过程结束。通过针对虚拟边界码，对存储器进行扫描，将自动地跳过坏块（由于它们不包括虚拟边界码），并且只从好块中读取数据，而无需计算设备提前确定哪些块是坏块或好块。根据另一个方面，该方法可以基于第一块中（例如，在数据映像的前导部分中）存储的信息，来推断该数据映像的大小或者虚拟块大小。可以根据数据映像的第一块或者前导部分，来推断该读写存储器的页大小，转而其可以用于确定每一虚拟块的页数量。

[0032] 另外的方面包括用于向闪存重写更新的或者新的数据映像的方法（例如，当更新或者替换存储器中存储的现有数据映像时）。该方法包括读写存储器的至少一个初始虚拟块中的初始数据映像。此外，从所述至少一个初始虚拟块向前扫描该读写存储器中可用的至少一个后续虚拟块。可以将新数据映像的第一部分存储在后续的虚拟块中，该虚拟块包括添加到其的虚拟边界码。此外，该方法还包括：在存储了完整的新数据映像之后，从所述至少一个初始虚拟块中擦除初始数据映像。可以将新数据映像重写到所述至少一个初始虚拟块中。事实上，可以将新数据的重写和旧数据的擦除执行为一个动作。但是，不需要将新数据映像写入到第一虚拟块中，可以将其写入到后面跟着的一个或多个虚拟块中。一旦将完整的新数据映像写入到所述至少一个初始虚拟块或者第一虚拟块后面跟着的一个或多

个虚拟块中，则作为该方法的一部分，该方法还可以包括：从新数据映像临时写到的后续虚拟块中擦除该新数据映像。

[0033] 当计算设备第一次加电时，该计算设备的处理器通常并不具有 ROM 或 RAM 中装载的操作系统。因此，处理器初始时执行 ROM 固件中存储的程序代码（其称为引导加载程序），该程序开始启动该系统的进程。引导加载程序的工作是将处理器随后执行的其它数据和程序装载到随机存取存储器 (RAM) 中。主引导加载程序通常存储在 ROM 中，并在设备加电或者该设备重置时执行。其后，在操作性地耦合到 CPU 的 NAND 设备上存储的一个或多个辅助引导加载程序可以允许装载和执行新的和 / 或额外的应用代码。传统情况下，使用多阶段引导加载程序，在此期间，越来越复杂的几个程序在链式加载过程中一个接一个地加载。但是，在 NAND 闪存中具有单一可变大小的引导加载程序是有利的，其中该引导加载程序在存储器的不同的页面和 / 或块大小和坏块特性中具有可扩展性。这种 NAND 存储器引导加载程序可以仍然是辅助引导加载程序，和 / 或与其它引导加载程序一起工作。

[0034] NAND 存储器制造商和经销商并不使用统一标准格式来识别它们的存储设备中的坏块。尽管如此，计算设备必须知道如何处理存储器中的坏块，否则它将无法正常工作或持续。因此，坏块软件例程通常被包含于数据加载例程（其包括辅助引导加载程序）中，坏块软件例程被配置为适应坏块，并提供处理每一个制造商 / 经销商的 NAND 存储设备的方法，其中预期这些存储设备将结合该计算设备进行使用。但是，NAND 存储芯片可能是从多个制造商或经销商中的任何一个购买的，并且经销商可能改变它们的坏块检测信息 / 方法。当代坏块软件例程通过将多种不同的例程包含到被编程进它们的计算设备的引导加载程序中，来处理多种不同的坏块检测方案。这通过尝试处理经销商的多变的要求和规定来实现。适应坏块例程中的这种变化，可能使引导加载程序的 ROM 固件变得复杂，通常并不希望对这些例程进行改变来适应经销商改变。此外，当设计第一引导加载程序完全地位于 NAND 存储器的最小典型块 0 之内时（其中保证该块是好块），这可能使系统的整体启动变得复杂，并常常使驱动程序和引导加载程序重复。本文的一个方面减少了这种复杂度和 / 或重复性，并使 ROM 能够加载跨度超过第一好块的引导加载程序。

[0035] 所公开的实施例包括：用于限制和 / 或消除对 NAND 存储器经销商针对将数据映像装载到存储芯片上的过程，而实现的不同的坏块检测方案的需求的方法。通过增加虚拟边界码以使计算设备能够识别要装载的数据块，而无须使用资源来查找坏块。针对正在处理（即，装载该数据映像）的读写存储器来实现通用的坏块检测 / 处理例程，可以实现为机器设置的一部分。

[0036] 在各个方面，要在读写存储器上存储的数据映像配置有一个或多个虚拟边界码，每一个虚拟边界码位于该数据映像中与虚拟块相对应的位置。这些虚拟块的大小与计算设备中实现的读写存储器的块大小相兼容。虚拟块大小可以匹配存储器的实际块大小。但是，为了适应存储器的不同配置（即，具有不同大小的实际块的存储芯片），可以将虚拟块大小设置成：在该计算设备中可能实现的所有存储器的实际块的最小公分母的大小。这使得本发明的方面方法能支持不同页大小（例如，2K 和 4K）的 NAND 设备，而不会针对每一种类型的存储器都要求单独的数据映像。通过将虚拟边界码添加在虚拟块内的固定位置（至少在初始块之后），可以在能够实现的任何存储器的好块中存储同一个数据映像，并且跳过任何中间的坏块。可以将这些虚拟边界码添加在虚拟块中的几乎任何位置（例如，开始位置）。

[0037] 当将数据映像存储到读写存储器（例如，NAND 存储器）上时，使用特定于经销商的坏块检测方法，以避免坏块，并且只使用存储器中的好块。但是，传统的系统需要接下来读取该数据映像的例程也使用特定于经销商的坏块检测方法。相比而言，所公开的实施例的方面最小化或者消除了在读取数据映像时对使用这些特定于经销商的方法的需求，提供适用于实质上任何读写存储设备的更通用方法。本文所公开的方面将数据映像存储在读写存储器上的一个或多个虚拟块中。可以将数据映像存储在读写存储器的一个或多个虚拟块中。如果需要一个以上的虚拟块来容纳整个的数据映像，则可以使用接下来的可用虚拟块（或者多个块，如果需要的话）来存储该数据映像的剩余部分，其中跳过任何中间的坏块，直到存储了全部的数据映像为止。虚拟块通过添加在放置在每个虚拟块中的数据映像的部分中的虚拟边界码来限定和划分界线。一个实施例将虚拟边界码放置在各个虚拟块的起始位置。此外，第一虚拟块可以不需要虚拟边界码，特别是由于通常都保证设备上的第一块是好块。此外，如果将虚拟边界码放置在下一个和 / 或后续虚拟块的起始位置，则它们标识数据映像的部分存储的位置。只将数据映像与添加到其的虚拟边界码存储在存储器的好的实际块中，因此虚拟边界码对应于存储器的好的实际块内的好的虚拟块。结果，引导加载程序可以使用根据本文的方面来使用单一映像读取处理，以适应实质任何读写存储器的方式，来识别虚拟边界码并确定要读取的存储器的块，而不用管制造商或经销商。本发明的方面方法可以用于从 NAND 存储器来启动设备（如上所述的编程），例如，智能电话或者具有计算能力的其它电子产品。此外，这些方面方法也可应用于处理辅助引导加载程序的故障保护更新。

[0038] 这些方面方法特别适用于存储在闪存中，并由计算设备（例如，智能电话或其它移动计算设备）使用为在加电或重启时装载的初始软件的初始引导软件映像。一般情况下，这种引导软件被操作系统、处理器或设备制造商紧紧地控制。控制引导软件的公司通常将主引导加载映像“烧制”到 ROM 存储器中，随后作为计算设备的引导例程的一部分，计算设备访问读写存储器（例如，NAND 存储器）来装载初始软件映像。

[0039] 本发明的方面方法、系统和设备提供坏块处理，从实质任何 NAND 设备实现单一辅助引导加载程序（本文称为引导加载程序并简称为“BL”）体系结构。在当代的系统中，辅助引导加载程序的大小可以通过内部 SRAM 存储器的大小来规定。所以，例如，如果在 NAND 设备中 SRAM 大小是 256KB，则减去通常放置在块 0 中的前导的千字节，并减去用于各种各样的其它数据（例如，证书 (Cert) 或填充 (PAD)）的额外千字节，在该初始实际块中可以剩下大约 240KB。该 240KB 大小对应于多个单独的引导加载程序的最大大小，并大于如今通常在读写存储器中实现的 128KB 的最小 NAND 块大小。本文的一个方面将虚拟块大小设置成与 128KB 的当代最小 NAND 块大小相对应。因此，为了从 SRAM 执行 240KB BL，则需要主引导加载程序读取一个以上的虚拟块。尽管一些当代闪存设备具有大于 128KB 最小值的块大小（其中一些具有 256KB，并且正在开发更大的存储器），但这些更大的实际块可以包括两个或更多个 128KB 的虚拟块。此外，本文方法的方面也可以扩展到更小的块大小（例如，32KB）。

[0040] 应当理解的是，本文将 128KB 的分段大小使用成例子，其只是用于说明目的；根据考虑的具体 NAND 设备，这些分段可以更大或者更小。例如，如果使用 128KB 虚拟块大小，则可能不支持每个块小于 64 页或者每个块少于 128KB 的 NAND 设备。但是，由于随着虚拟块

大小变得越来越小，所需要的虚拟块的数量增加，因此对于更小的块大小具有一些实际限制。更多的虚拟块意味着：为了找到有效的数据映像需要进行更多的扫描，这可能转换成增加启动时间。

[0041] 本发明的方面方法将放置在NAND设备上的BL映像分割为可以放置在128KB虚拟块大小中的分段。由于128KB虚拟块大小对应于NAND设备的当代最小值，因此选择该虚拟块大小，但也可以根据需要使用不同的虚拟块大小。通过在至少第二128KB分段中添加标记来生成虚拟块。第一128KB分段通常包括BL前导和其它可读代码，并因此不需要包括虚拟边界码。虚拟边界码所标记的每一个分段，规定了虚拟块及其边界（虚拟块的两端——本文还将其称为“虚拟块边界”）。对BL映像的剩余部分进行分割，并分布到所需要的多个虚拟块以容纳整个映像。可以增加封装的数据，以填充不完整的数据虚拟块。

[0042] 各个方面允许ROM固件的开发人员不再必须适应NAND存储器经销商所规定的坏块检测方案。各个方面不再依赖于预编程的保存坏/好块表的页。所公开的技术简化了闪速工具和构建管理的设计，而以前针对不同的NAND设备则需要不同的工具/构建。另外，所公开的设备、系统和方法实现了引导加载程序的故障保护更新以及其处理。此外，该NAND设计可扩展以支持不同的页/大小的NAND设备。

[0043] 图1根据各个方面，示出了具有单独的引导加载程序15的NAND设备100的封装。在该实施例中，使用128KB虚拟块大小，其工作于2KB或4KB NAND设备（如上所述）。因此，为了建立128KB虚拟块，可以通过虚拟边界码14来指出虚拟块边界。这种虚拟边界码14是唯一的和可识别的数据模式，其可以如12字节数据一样小。举例而言，虚拟边界码14可以是十六进制数字：844bdc56 73531014 d48b54c6。虽然可以使用比这种情况更大或更小的虚拟边界码14，但其不需要很大，这是由于只需要很小数量的数据来标定虚拟块边界。此外，还可以考虑SRAM中为主引导加载程序栈、共享区域和块0的开始处的其它数据所保留的其它部分。因此，与实际SRAM大小相比，可用的SRAM可能更小。

[0044] 图1还示出了前导5，其包括可以称为前导存储码的一个或多个存储码10。即使包括几个前导内存码10，也可以假定前导不超过10KB。因此，可以在NAND设备100的块0的前10KB之后，开始写入BL 15。在图1所示的例子中，BL 15的前80字节包括映像头12（其指定引导加载程序第一部分15a的起始位置），并可以包括额外的信息（例如，针对虚拟块设置的大小）。其后，可以通过将虚拟边界码14添加在128KB边界处（跟着是下一个128KB中的引导加载程序第二部分15b），来标记虚拟块边界。虚拟边界码14可以位于典型的辅助引导加载程序的中间，因此BL 15被分割成一个以上的部分15a、15b。通过知道前导大小、映像头大小和虚拟块的大小，可以确定第一引导加载程序部分15a的大小。随后，可以将引导加载程序的剩余部分包括在引导加载程序第二部分15b中，或者甚至包括在额外的虚拟块中的另外部分中（如果需要的话）。因此，在对被分割的BL 15的每一个部分多大进行确定时，可以包括一些已知的或者假定的变量。例如，考虑下面的参数：

[0045] • SRAM大小＝256KB

[0046] • 保留/使用的SRAM＝～16KB

[0047] • 虚拟块大小＝128KB

[0048] • 前导(5)大小（其包括前导存储码10）＝10KB

[0049] • 引导加载程序映像头(12)大小＝80B

[0050] • 签名和证书 (16) 加填充 (18) = 6KB

[0051] • 虚拟边界码 (VBC) 大小= 12B。

[0052] 从 SRAM 大小中减去所使用的上面字节的总数，在 NAND 存储器中剩下大约 223KB 用于引导加载程序，该引导加载程序可以存储在两个 128KB 虚拟块中。

[0053] 图 2 示出了用于考虑引导加载程序 15 的间隔和分布的另一种方式。首先在 20 处，可以按照 128KB 建立虚拟块大小，因此在 256KB SRAM 中包括两个这种的虚拟块。在 22 处，可以考虑 BL 15 的大小，其跨度虚拟块中的至少两个。随后，在 24 处，向 BL 15 映像的总长度增加映像头 12 和证书 16。此外，在 26 处，向该映像的总长度增加前导 5（其包括前导存储码 10）和填充 18。此外，在 28 处，在 BL 15 的两个部分 15a、15b 之间，添加虚拟边界码，在该实施例中，其对应于第一 128KB 虚拟边界。在考虑这些参数的情况下，实际设备的页大小决定将使用多少实际块，以及将使用多少虚拟块。例如，考虑页大小为 4K 的 NAND 设备。该设备具有 256KB 的实际块，在块 0 中大约留下 223KB 用于引导加载程序。因此，4K NAND 设备只需要一个好块来保存该 BL。如果使用 128KB 的虚拟块大小，那么一个好的实际块包括两个虚拟块。现在考虑页大小仅为 2K 的 NAND 设备。该设备将需要两个好的实际块。这些实际块的大小与 128KB 的虚拟块大小一致。因此，2K NAND 的块 0 只留下近似 118KB 的空间用于第一 BL 部分 15a(128KB 减去 10KB 前导 5，减去 80B 的映像头 12)。然后，可以将剩余的第二 BL 部分 15b 放置在通过虚拟边界码 14 所指定的下一个好块中。

[0054] 图 3 示出了各种场景中，示例性引导加载程序如何分布。所描述的五种场景均使用 128KB 虚拟块大小（在这五种场景上，示出为一系列虚拟块）。三种场景考虑 2KB NAND 设备，另两种场景考虑 4KB NAND 设备。在 2KB 场景中，虚拟块大小与 128KB 的实际块大小相一致。在 4KB 场景中，虚拟块是实际块大小的一半。

[0055] 在 2KB 场景 1 中，引导加载程序的分割和分布是简单明了的，其中虚拟边界码 14 放置在块 1 的开始位置处，其与该设备上标记的第一虚拟块边界相一致。应当注意的是，在 2KB 场景中，虚拟块与实际块相一致。关于上面所描述的 NAND 存储器的编码，块 0 还包括前导 5（其具有前导存储码 10）、引导加载映像头 12 和第一引导加载程序部分 15a。第二引导加载程序部分 15b 跟着添加的虚拟边界码 14，其中在该实施例中，虚拟边界码 14 标记虚拟块的开始。这种 2KB 场景 1 应用于块 1 是好块的 NAND 设备，所以第一和第二虚拟块是连续的好块。但是，制造商通常只保证块 0 是好的，因此考虑 2K NAND 设备的第二和第三场景。在 2KB 场景 2 中，块 1 是坏的，但块 2 是好的。因此，由于块 1 是坏的，所以虚拟边界码 14 不能写到该块中。相反，虚拟边界码 14 以及第二引导加载程序部分 15b 写入的是下一个好块（即，块 2）。2KB 场景 3 具有跟着块 0 的两个初始坏块（块 1 和块 2）。因此，在该情况下，在这两个坏块中的最后一个坏块之后的边界处（其是下一个好块（块 3）的开始位置）写入虚拟边界码 14。类似地，在该场景下，将该 BL 的剩余部分 (BL 第二部分 15b) 写入到块 3 中。

[0056] 在 4KB 场景下，其实质上并不关心块 1 是好块还是坏块，这是由于引导加载程序通常可以被完全容纳在块 0 中，块 0 足够大到包括两个虚拟块。因此，4KB 场景 1 示出了块 1 是好块的情形，但其是不需要的，这是由于 BL 第一部分 15a 和 BL 第二部分 15b 二者均被完全地写入到块 0 中。此外，4KB 场景 2 示出了两部分的 BL 15a、15b 如何仍然能够完全地写入到其块 0 中，而与在该场景中，块 1 是坏的无关。

[0057] 图 4 示出了引导加载程序分布的另外两个场景。图 4 中的第一场景是 4KB 场景 3，该场景包括需要分割成三个部分 15a、15b、15c 的更大 BL。由于第二实际块（块 1）是好的，因此该数据映像扩展到连续的虚拟 / 实际块。但是，当块 1 是坏块时，第二虚拟边界码 14 和 BL 第三部分 15c 将被存储在块 2 中。图 4 中的第二场景是 2KB 场景 4。再一次使用需要被分割成三个部分 15a、15b、15c 的更大 BL，但在这里，第三块（块 2）是坏的。因此，使用第四块（块 3）来存储第二虚拟边界码 14 和 BL 第三部分 15c。

[0058] 图 5 示出了在读写存储设备上存储数据映像的方面方法 500。该数据映像可以是根据本文的方面的引导加载程序。在方框 510 中，装载读写设备（例如，NAND 设备）并准备在其上进行数据存贮。在方框 520 中，基于针对该读写存储设备指定的一系列虚拟块的虚拟块大小，来确定数据映像分布以便对该数据映像进行封装。作为该确定操作的一部分，如果整个数据映像对于单一虚拟块来说太大的话，需要将该数据映像分割成至少数据映像第一部分和数据映像第二部分。此外，可以向数据映像第一部分和数据映像第二部分中的至少一个添加虚拟边界码。在该实施例中，将虚拟边界码添加到数据映像第二部分的开始处，因此在方框 530 中，将数据映像第一部分存储在一系列虚拟块中的第一虚拟块里。在方框 540 中，存储虚拟边界码 (VBC)，形成下一个虚拟块（并因此指出一系列虚拟块中的第二虚拟块），其中该下一个虚拟块自然地与好的实际块相一致。在存储 VBC 和指明虚拟块之后，该读写存储器中的第一虚拟块和第二虚拟块之间的任何中间坏块都被跳过。用于物理地写入数据映像的编程工具，可以执行针对于工厂所标记的坏块的特定于经销商的检查，并根据需要跳过这些坏块。这些编程工具的目标是存储数据映像和 / 或将 VBC 合并在数据映像中。因此，编程工具不需要识别或者甚至处理 VBC，这是由于将 VBC 作为数据映像的一部分，以生成可以独立于经销商的坏块管理方案来读取的模式。这些编程工具也可以具有 NAND 驱动器，但与 ROM 中的 NAND 驱动器不同，编程工具 NAND 驱动器不需要在大小或复杂度方面受到限制。这允许编程工具遵循修订或者对产品的更新来修复错误，甚至处理新的特定于 NAND 经销商的坏块检查方案。用此方式，与 ROM 中的 NAND 驱动器相比，可以对这些编程工具中的功能进行增加、删除、更改或者增强。在方框 550 中，可以将数据映像的第二（下一个）部分存储在与 VBC 相同的虚拟块中，其中该虚拟块是该系列虚拟块的第二虚拟块。判断框 560 判断是否存储了完整的数据映像。如果没有，则该方法返回到方框 540，在下一个可用的虚拟块存储另一个 VBC，同样在方框 550 中，存储该数据映像的下一个部分。继续该循环，直到存储了该数据映像的所有部分为止。一旦在判断框 560 中做出了已存储了完整的数据映像（例如，引导加载程序映像）的肯定判断，则针对在方框 510 中开始 / 准备的读写设备装载，该处理在 570 中结束。因此，该处理可以与其它读写设备串行地或并行地重复执行。

[0059] 将 VBC 放置在虚拟块的开始位置，有助于立即检测到好块，并可以优化引导装载进程。虚拟块的单一通道读取可以检测 VBC，并同时读取在该块中包含的引导加载程序数据映像。替代地，不需要将虚拟边界码放置在虚拟块的开始位置。例如，可以将 VBC 放置在每一个虚拟块（其包括块 0）的结束位置，或者实质上可以根据期望放置在任何位置。关于一个或多个 VBC 的位置信息，可以包括在上面所讨论的映像头中。

[0060] 一旦 NAND 设备具有向其写入的一个或多个虚拟边界码，则主引导加载程序可以解析该设备，以找到这些虚拟边界码，并使用类似于路标的虚拟边界标记来读取、认证和运

行完整的辅助引导加载程序。图 6 示出了从读写设备中读取数据映像的一个方面方法。

[0061] 图 6 示出了从读写存储器中读取数据映像的一个方面方法 600。例如，主引导加载程序中的 NAND 解析器可以实现方法 600，来读取、装载和认证辅助引导加载程序。在方框 605 中，以块 0 中的前导（其还与该设备的页 0 相一致）为起始，对该读写设备进行读取。该前导可以包含可以由数据装载例程使用的设备宽度和页大小信息。在方框 610 中，可以执行纠错编码 (ECC) 检测，这是由于当前方法优选地启用 ECC。该操作可以缺省为 4 比特 BCH 纠错编码或者其它编码。如果对编程的页的读取不成功，从而没有检测到 ECC，则在方框 612 中，可以检查另外的 ECC 配置。例如，可以使用 8 比特 BCH ECC，也可以使用 4 比特 BCH ECC（如果其不是缺省的话）。因此，可以检查一个或多个额外的 ECC 配置，使得如果在方框 614 中也没有检测到 ECC，则在方框 616 中结束，其意味着没有找到引导加载程序 (BL)，该处理退出。替代地，该方法可以使用 AUTO_DETECT 例程和加权的算法检查，来检查有效的前导。否则，如果在方框 610 或方框 614 中检测到 ECC，则该方法转到方框 620 处，在其中执行页大小检测。在方框 620 中，可以读取连续的页来验证虚拟边界码大小和 / 或位置，以便计算实际 NAND 页大小。如果在判断框 620 处没有检测到页大小，则在另外的判断框 622 处，判断是否检测到任何虚拟边界码 (VBC)。如果在 622 处检测到 VBC，则在 624 处，读取指出其偏移（位置和大小）的 VBC，并移动到下一个页，返回到页大小检测判断框 620。如果在 622 中没有检测到 VBC，则在方框 616 中结束，其意味着没有找到引导加载程序 (BL)，该处理退出。如果在判断框 620 中检测到页大小，则通过将虚拟块大小除以页大小，来确定每一个虚拟块的页数量。ROM 可以使用该确定结果，来了解在转到下一个虚拟块之前，需要读取多少固定的页。一旦检测到页大小，则可以在方框 630 中访问第一虚拟块中的数据映像，甚至进行复制。因此，可以读完该第一虚拟块中的剩余页，其使得该数据映像的第一虚拟块中的所有数据都被访问到。随后，该处理可以转到针对 VBC，扫描下一个虚拟块。如果在 640 处没有检测到 VBC，则下一个虚拟块是坏的和 / 或损坏的。因此，在方框 642 中，该处理针对虚拟边界码，扫描下一个虚拟块。对于可以检查多少虚拟块，可以使用门限（其还称为超时功能），以便确保该处理不会变成为死循环。因此，可以使用诸如 15 个虚拟块之类的门限，如果在 644 处达到该门限，则这可能意味着没有能够找到 BL，该处理在 616 处结束。如果没有达到虚拟块门限，则该循环继续进行 / 返回到方框 640 处，直到在方框 640 中检测到虚拟边界码为止。一旦在 640 处检测到 VBC，则在 650 中，可以访问下一个虚拟块中的所有数据。该下一个虚拟块对应于找到 VBC 并且可以访问的虚拟块。在 660 处，可以判断是否读取了完整的数据映像（例如，完整的 BL）。如果在 660 处还没有读取完整的 BL，则在方框 642 中，该处理针对虚拟边界码来扫描下一个虚拟块，如上面所述的进行进一步处理。如果在方框 660 处读取了完整的 BL，则在 670 处，可以对该 BL 进行认证和执行，因此根据由第一虚拟块和第二虚拟块中的数据映像所表示的数据来装载该 BL。BL 装载可以跳过（不需要包括）虚拟边界码字节，这些字节仅仅表示标记，不需要进行装载。

[0062] 图 6 中所示出的方面方法示出了如何跳过坏块，使得当在第一虚拟块和第二虚拟块之间或者后续的虚拟块之间存在坏块时，数据装载例程（例如，主引导加载程序）能够读取整个的数据映像（例如，辅助引导加载程序），而不用首先识别该存储器中的好块或坏块。因此，数据装载例程将跳过跟着第一虚拟块的坏块，这是因为在坏的虚拟块中不会检测到虚拟边界码。此外，可以剩下第一虚拟边界码，而不由辅助引导加载程序执行（这是由于

只需要将这些字节作为标记）。

[0063] 另外的方面包括故障保护更新提供，例如通过针对无线设备的空中（OTA）下载。在该方面，包括读写存储器的很多设备（例如，包括闪存的智能电话）通常需要对固件进行升级或者更新。所公开的坏块管理设计方案使用上面所描述的技术的扩展，以可靠／故障保护方式来实现这些升级。初始时，系统检测当前引导加载程序所在的块。随后，提前转到可用于写入新的／替代的引导加载程序的一个或多个其它好块。其后，可以将新的好块位置用作备份，对新的引导加载程序进行编程，以防止某些事情中断该升级／更新过程。随后，可以在初始位置处（例如，其以块 0 开始），擦除旧引导加载程序，并在块 0 和后续的好块处，重新编程新的辅助引导加载程序（如果需要的话）。

[0064] 图 7 根据各个方面，示出了对 NAND 设备上的映像进行修订的方法的例子。这种升级方法与使用上面所描述的虚拟边界码的虚拟块指定完全兼容。在初始状态 70 下，系统访问当前引导加载程序所在的第一块内的数据。这些块可以包括前导 5（其具有前导存储码 10）和映像头 12。在所示出的例子中，将引导加载程序分割成布置在两个初始块中的两个部分 15a、15b，其中根据 NAND 设备，这两个初始块可以对应于虚拟块或实际块，并且根据本文的方面，其可以包括虚拟边界码 14。替代地，根据现有技术（可以根据本发明的方法来替代该技术），原始引导加载程序可以是单一的连续引导加载程序。在该阶段，系统可以确定实际块的大小（或者虚拟块的大小（如果存在的话）），以便进一步确定新数据映像（其包括任何引导加载程序部分 15a’、15b’、新前导 5’（其具有新的前导内存码 10’）、新映像头 12’和任何必需的新 VBC 14’）需要多少块。系统可以在闪存设备上向前扫描，以便寻找下一个可用的好块。如果新的引导加载程序需要一个以上的好块，则系统可以继续在闪存设备上向前扫描，直到找到足够的可用好块来容纳该新数据映像为止。在 72 处所示出的阶段中，系统随后在所识别的可用的好块中存储新的数据映像（其包括引导加载程序部分 15a’、15b’、具有新的前导内存码 10’的新前导 5’、新映像头 12’和新 VBC 14’）。根据本发明的一个方面，可用的好块不需要是紧接着的下一个顺序可用的好块。在所示出的例子中，作为该方法的一部分，为了在旧块和改变的新块之间提供缓冲，跳过了至少两个块。留下至少一对块，也使得该方法能考虑初始引导加载程序的大小的未来成长，与当代系统所需要的一个或两个初始块相比，未来可能占用更多的块。此外，留下一些空间还可以避免设备的这些初始块中有坏块。在擦除旧的引导加载程序之前，可以对新数据映像执行验证检查。在 72 处所示出的阶段，在新数据映像没有正确地装载、被损坏或者不可使用的情况下，系统仍然从旧的引导加载程序部分 15a、15b 进行引导，以作为故障保护。一旦装载了新数据映像（并可选地进行验证），则在 74 所示出的状态中，可以擦除旧的引导加载程序或者至少其初始部分。具体而言，如果擦除旧前导 5、旧映像头 12 或者引导加载程序 15a 的初始部分中的至少一个，则寻找这些元素的主引导加载程序可以跳过旧的引导加载程序。虽然没有将 74 处所示出的第二块示出为被擦除，但可以在转到下一步之前，对该块进行擦除。其后，在 76 所示出的状态中，系统随后在第一块中存储新的数据映像，其实质上与先前在阶段 72 处所存储的新数据映像相同。因此，在阶段 76 处，可以在 NAND 设备上的两个地方找到新数据映像，在标记第一块出现错误的情况下，这可以用作备份。一旦对初始块中的数据映像进行了认证，则该修订方法可以结束。可以在 NAND 设备上留下新数据映像的第二复本作为备份。替代地，在 78 处所示出的状态中，该方法可以进一步从 72 处示出的状态中所标记的好块

里，擦除新数据映像的第二复本。

[0065] 在图 7 中，与目标数据映像无关的各个块被指示成“擦除的”。但是，这些块并不需要是空白的，或者是被擦除的。根据本文的方面，这些方法可以使用前导和虚拟边界码来确定引导加载程序的大小和位置，因此被注释成“擦除的”那些块旨在指示该数据块与引导加载程序映像无关。

[0066] 在图 8 中，示出了在读写存储器上重写数据映像的方面方法 800。在方框 810 中，在读写存储器的至少一个第一虚拟块中，访问初始数据映像。所述至少一个第一虚拟块可以包括一个以上的第一虚拟块。在方框 820 中，该方法针对该读写存储器中可用的至少一个第二虚拟块，从所述至少一个第一虚拟块向前扫描。因此，如果包括的所述至少一个第一虚拟块超过一个虚拟块，则如方法 800 中所规定的“第二虚拟块”将跟着这些在先的第一虚拟块。在方框 830 中，将新数据映像的第一部分存储在第二虚拟块中。在方框 840 中，该方法从第二虚拟块向前扫描到第三虚拟块。一旦定位到第三虚拟块，则在 850 处，将新数据映像的第二部分存储在第三虚拟块中。方框 860 包括 :擦除所述至少一个第一虚拟块的至少一部分中的初始数据映像。在方框 860 中，可以擦除整个的初始数据映像或者仅仅其一部分。例如，在方框 860 中，可以从一个以上的第一虚拟块中擦除初始数据映像。在 870 处，将包括 VBC 的新数据映像第一部分和第二部分存储在所述至少一个第一虚拟块中。在方框 880 中，从第二虚拟块中擦除新数据映像第一部分。在方框 890 中，从第三虚拟块中擦除新数据映像第二部分。

[0067] 前导块可以用于获取各种设备特性。如果初始没有发现有效的前导块，则该处理可以继续穿过这些块以尝试找到一个前导块。可以通过在作为闪存读取命令的一部分所读取的块的初始字节（例如，前 12 字节）中，检查特定的编码（例如，虚拟边界码）来检测前导块。根据所公开的方面，可以期望通用的 NAND 设备宽度处理技术。可以使用一种算法来计数所检测到的被指定的虚拟块。如果使用这一技术没有获得虚拟块计数，则该处理可以退出，并以没有找到辅助引导加载程序来结束。否则，在这种方式下，可以使用虚拟边界码来确定虚拟块和 NAND 设备自身的大小。此外，可以将 ECC 检测和页大小检测实现成本文所公开的方法的一部分。例如，自动页大小检测算法可以读取利用特定的虚拟边界码来标记的页的数量，以便确定在该设备上有多少页，并因此确定其页大小。

[0068] 另外，各个实施例可以使用和 / 或利用各种各样的移动计算设备中的任何一种来实现，在图 9 中以蜂窝电话的形式示出了其一个例子。典型的移动计算设备 900 通常都具有图 9 中所示出的部件。例如，移动计算设备 900 可以包括耦合到内部存储器 902 的处理器 901 和触摸表面输入设备 / 显示器 903（例如，电阻式感应触摸屏 904、电容感应触摸屏、红外线感测触摸屏、声学 / 压电感应触摸屏等等）。移动计算设备 900 可以具有用于发送和接收电磁辐射的无线装置 / 天线 906，后者连接到无线数据链路和 / 或耦合到处理器 901 的蜂窝电话收发机 902。移动计算设备 900 还可以包括耦合到处理器 901 的 GPS 接收机，以确定该设备的位置。此外，移动计算设备 900 还可以包括用于接收用户输入的物理按键 908。

[0069] 各个实施例可以使用和 / 或利用各种各样的计算设备中的任何一种（例如，平板计算机）来实现，图 10 示出了其一个例子。例如，无线设备 1000 可以包括耦合到内部存储器 1004 和 1006 的处理器 1002。内部存储器 1004 和 1006 可以是易失性存储器或非易失性存储器，还可以是安全和 / 或加密存储器，或者非安全和 / 或非加密存储器、或者其任意组合。

此外，处理器 1002 还可以耦合到用户接口，例如触摸屏显示器 1016（例如，电阻式感应触摸屏、电容感应触摸屏、红外线感测触摸屏等等）或者传统的按键（例如，1012a 和 1012b）和非触摸屏显示器。另外，无线设备 1000 可以包括一个或多个网络收发机，其被配置为使处理器 1002 能够通过一个或多个有线或无线网络与其它计算设备进行通信的。举一个特定的例子，无线设备 1000 的网络收发机可以包括用于发送和接收电磁辐射的一付或多付天线 1018，后者可以连接到一个或多个无线数据链路收发机和 / 或耦合到处理器 1002 的蜂窝电话收发机 1010。此外，无线设备 1000 还可以包括用于接收用户输入的物理按键 1012a 和 1012b。

[0070] 上面所描述的各个实施例还可以在各种各样的个人计算设备（例如，如图 11 中所示的膝上型计算机 1100）之中实现和 / 或利用各种各样的个人计算设备来实现。很多膝上型计算机包括用作该计算机的指取设备的触摸板触摸表面 1107，因此可以接收拖拽、滚动和轻打手势（其类似于在装备有触摸屏显示器和上面所描述的移动计算设备上所实现的那些）。通常，膝上型计算机 1100 包括处理器 1101，后者耦合到易失性存储器和大容量非易失性存储器（例如，闪存设备 1102）。此外，膝上型计算机 1100 还可以包括耦合到处理器 1101 的软盘驱动器和激光光盘 (CD) 驱动器。此外，膝上型计算机 1100 还可以包括耦合到处理器 1101 的多个网络收发机或网络连接器端口，其配置为使处理器 1101 能够通过一个或多个有线或无线网络与其它计算设备进行通信。举一个特定的例子，膝上型计算机 1100 的网络收发机可以包括以太网、USB 或火线®连接器插座 / 收发机、耦合到一付或多付天线用于发送和接收电磁辐射的一个或多个无线调制解调器收发机（例如，Wi-Fi 和 / 或蜂窝数据网络收发机）。此外，膝上型计算机 1100 还可以包括用于将处理器 1101 耦合到未来可能开发的网络的其它类型的网络连接电路。在笔记本配置中，计算机壳体包括触摸板触摸表面 1107、键盘 1108 和显示器 1109，所有这些部件都耦合到处理器 1101。如本领域所公知的，计算设备的其它配置可以包括耦合（例如，通过 USB 输入）到处理器的计算机鼠标或者跟踪球，它们也可以结合各种实施例进行使用。

[0071] 本文描述的各种实施例中的处理器可以是能通过软件指令（应用）配置，以执行各种功能（其包括上面所描述的各种实施例的功能）的任何可编程微处理器、微计算机或者多处理器芯片或芯片集。在一些设备中，可以提供多个处理器，例如，一个处理器专用于无线通信功能，一个处理器专用于运行其它应用。通常，在访问软件应用并将它们装载到处理器之前，可以将这些软件应用存储在内部存储器中。处理器可以包括足够用于存储这些应用软件指令的内部存储器。在很多设备中，内部存储器可以是易失性存储器或者非易失性存储器（例如，闪存）、或者二者的混合。为了便于说明目的，对于存储器的引用通常指代处理器可访问的存储器，其包括内部存储器或者插入在设备之中的可移除存储器、以及处理器自身中的存储器。

[0072] 上述的方法描述和处理流程图仅仅是用作为说明性例子，而不是旨在要求或者隐含着必须以所给出的顺序来执行各个实施例的模块。如本领域普通技术人员所应当理解的，可以以任何顺序来执行上述的实施例中的模块顺序。

[0073] 诸如“其后”、“转而”、“接着”等等之类的词语，并不旨在限制这些模块的顺序；这些词语仅仅只是用于引导读者遍历该方法的描述。此外，任何对权利要求元素的单数引用（例如，使用冠词“一个 (a)”、“某个 (an)”或者“该 (the)”），不应被解释为将该元素限制

为单数形式。

[0074] 结合本文所公开的实施例描述的各种示例性的逻辑框和处理流程图模块均可以实现成电子硬件、计算机软件或二者的组合。为了清楚地表示硬件和软件之间的这种可交换性，上面对各种示例性的部件、框、模块、电路和模块均围绕其功能进行了总体描述。至于这种功能是实现成硬件还是实现成软件，取决于特定的应用和对整个系统所施加的设计约束条件。熟练的技术人员可以针对每个特定应用，以变通的方式实现所描述的功能，但是，这种实现决策不应解释为背离本发明的保护范围。

[0075] 用于执行本文所述功能的通用处理器、数字信号处理器 (DSP)、专用集成电路 (ASIC)、现场可编程门阵列 (FPGA) 或其它可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件部件或者其任意组合，可以用来实现或执行结合本文所公开实施例描述的用于实现各种示例性的逻辑、逻辑框、模块和电路的硬件。通用处理器可以是微处理器，或者，该处理器也可以是任何常规的处理器、控制器、微控制器或者状态机。处理器也可以实现为计算设备的组合，例如，DSP 和微处理器的组合、若干微处理器、一个或多个微处理器与 DSP 内核的结合，或者任何其它此种结构。替代地，一些模块或方法可以由特定于给定的功能的电路来执行。

[0076] 在一个或多个示例性方面，本文所述功能可以用硬件、软件、固件或它们任意组合的方式来实现。当在软件中实现时，可以将这些功能存储成非临时性计算机可读存储介质或者非临时性处理器可读存储介质上的一个或多个指令或代码。本文所公开的方法或算法的步骤，可以体现在处理器可执行软件模块中，后者可以位于非临时性计算机可读或者处理器可读存储介质上。非临时性计算机可读或者处理器可读存储介质可以是计算机或处理器能够存取的任何存储介质。举例而言，但非做出限制，这种非临时性计算机可读或处理器可读介质可以包括 RAM、ROM、EEPROM、闪存、CD-ROM 或其它光盘存储器、磁盘存储器或其它磁存储设备、或者能够用于存储具有指令或数据结构形式的期望的程序代码并能够由计算机进行存取的任何其它介质。如本文所使用的，磁盘和光盘包括压缩光盘 (CD)、激光光盘、光盘、数字通用光盘 (DVD)、软盘和蓝光光盘，其中磁盘通常磁性地复制数据，而光盘则用激光来光学地复制数据。上述的组合也应当包括在非临时性计算机可读和处理器可读介质的保护范围之内。另外，一种方法或算法的操作可以作为一个代码和 / 或指令集或者其任意组合，位于非临时性处理器可读介质和 / 或计算机可读介质上，其中该非临时性处理器可读介质和 / 或计算机可读介质可以并入到计算机程序产品中。

[0077] 本领域普通技术人员应当认识到，可以使用所公开实施例的方面的多种可能修改和组合，同时仍然利用相同的基本底层机制和方法。为了便于说明起见，参照特定的实施例来给出前面的描述。但是，上面的这些示例性讨论并不是详尽的，或者并不旨在将本发明限于所公开的精确形式。在了解了上面的教示内容之后，可以实现多种修改和变型。选择和描述这些实施例以解释本发明的基本原理以及它们的实际应用，使本领域其它普通技术人员能够通过各种修改以适合于预期的具体使用，来最佳地利用本发明和各种实施例。因此，本发明并不限于这些实施例和本文所示出和描述的所公开技术的各个方面，而是与所附权利要求书和本文公开的原理和新颖性特征的最广范围相一致。

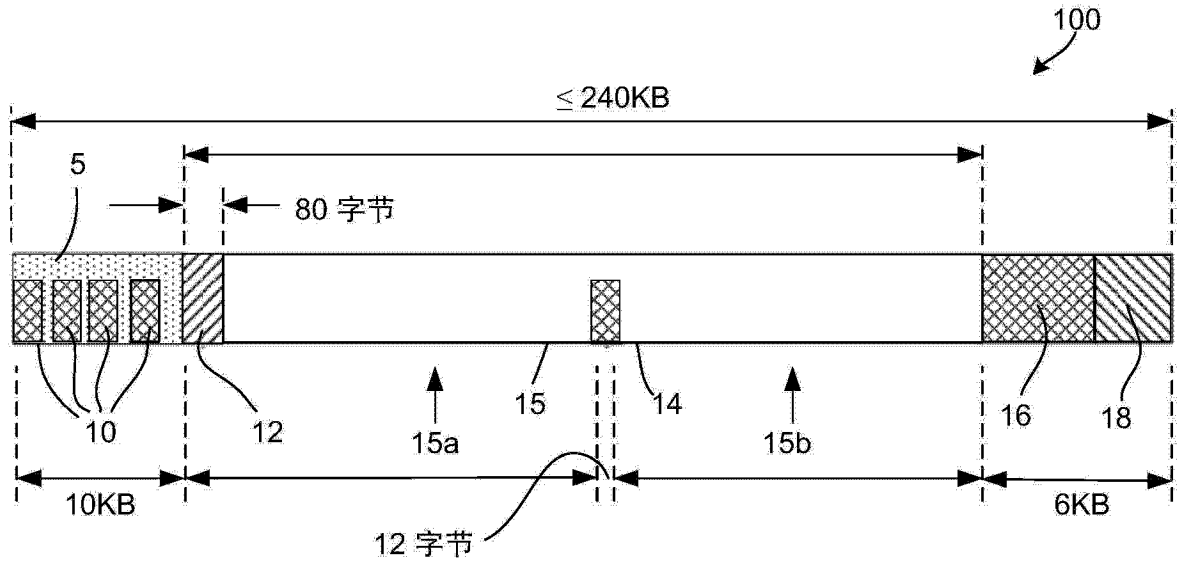

100
≤ 240KB
5
80 字节
10
12
15a
15
14
15b
16
18
10KB
12 字节
6KB

图 1

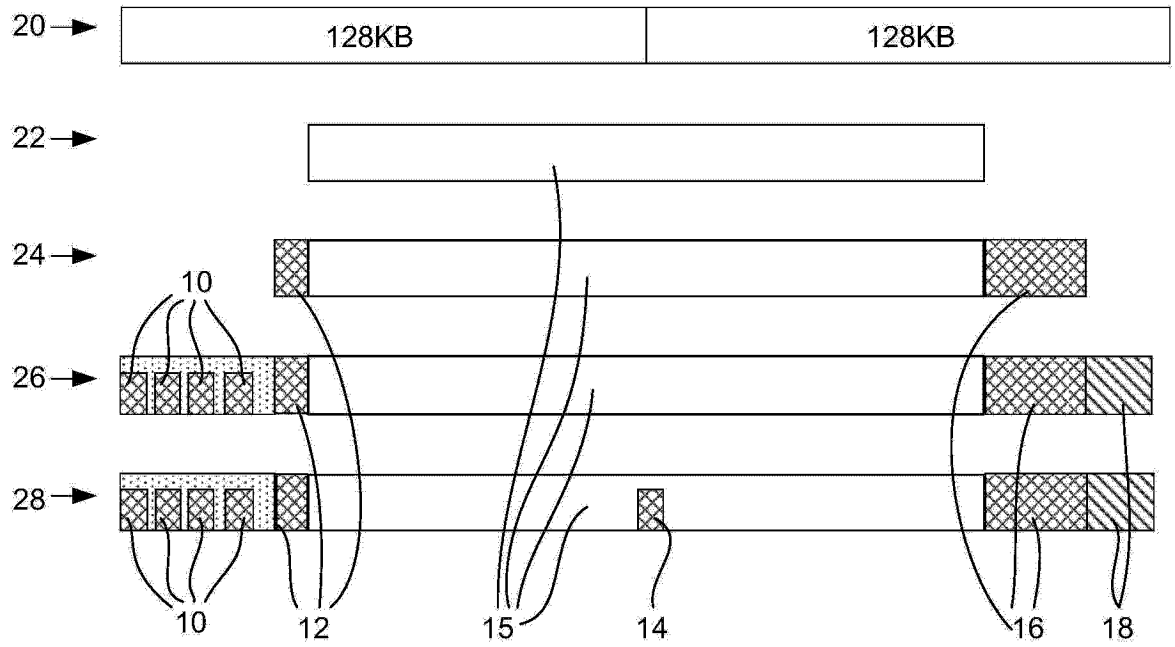

20
128KB
128KB
22
24
10
26
28
10
12
15
14
16
18

图 2

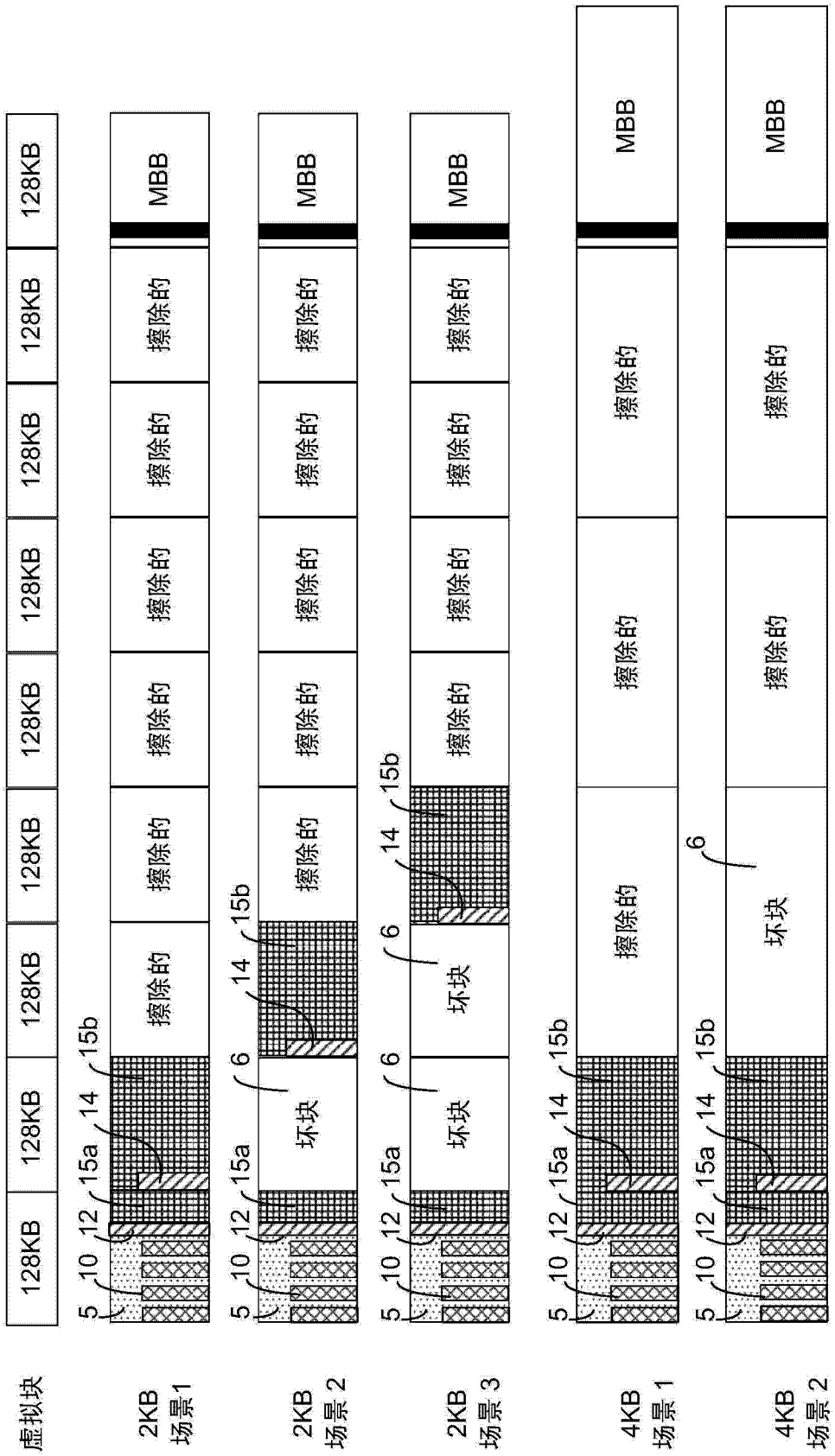

虚拟块
128KB
128KB
128KB
128KB
128KB
128KB
128KB
128KB
2KB 场景 1
擦除的
擦除的
擦除的
擦除的
MBB
5
10
12
15a
14
15b
2KB 场景 2
坏块
擦除的
擦除的
擦除的
MBB
6
2KB 场景 3
坏块
擦除的
擦除的
擦除的
MBB
4KB 场景 1
擦除的
擦除的
擦除的
MBB
4KB 场景 2
坏块
擦除的
擦除的
擦除的
MBB

图 3

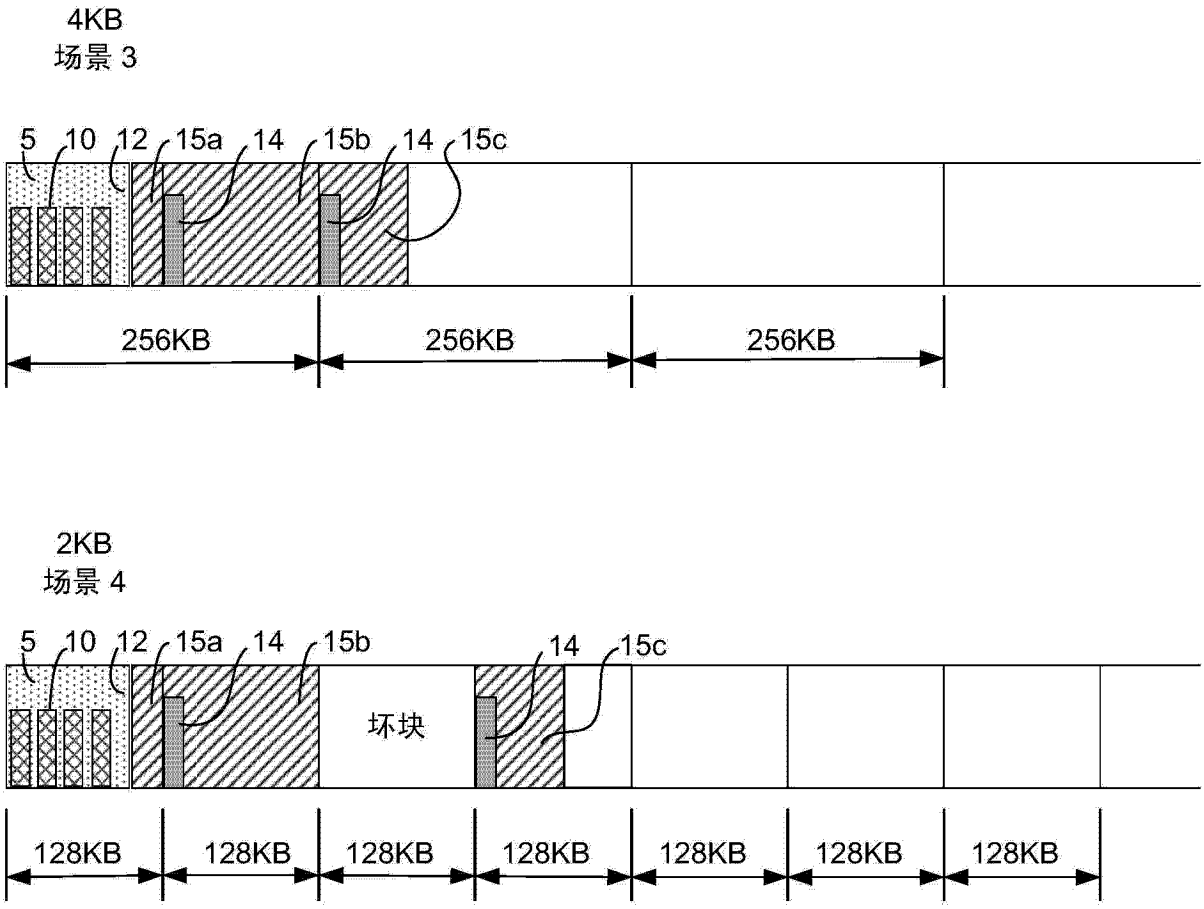

4KB
场景 3
5
10
12
15a
14
15b
14
15c
256KB
256KB
256KB
2KB
场景 4
5
10
12
15a
14
15b
坏块
14
15c
128KB
128KB
128KB
128KB
128KB
128KB
128KB

图 4

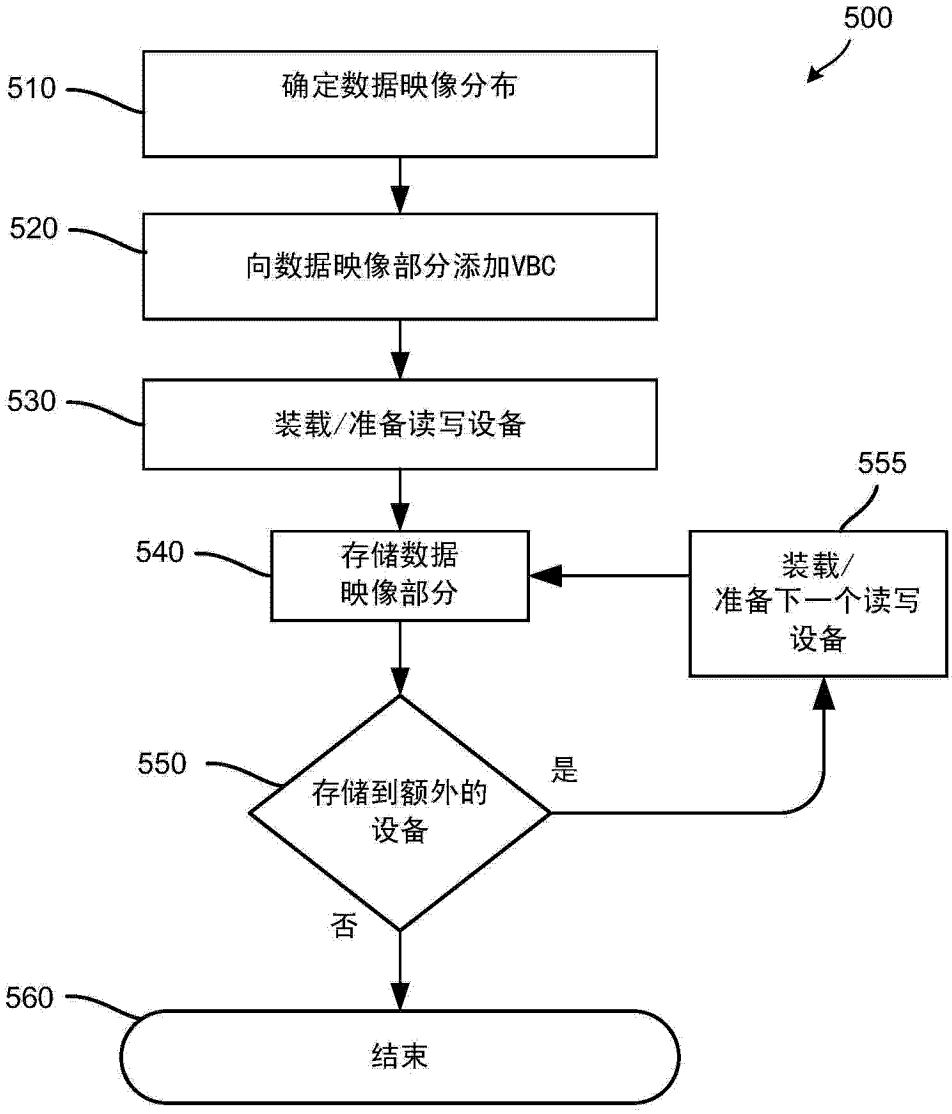

500
510
确定数据映像分布
520
向数据映像部分添加VBC
530
装载/准备读写设备
555
540
存储数据
映像部分
装载/
准备下一个读写
设备
550
存储到额外的
设备
是
否
560
结束

图 5

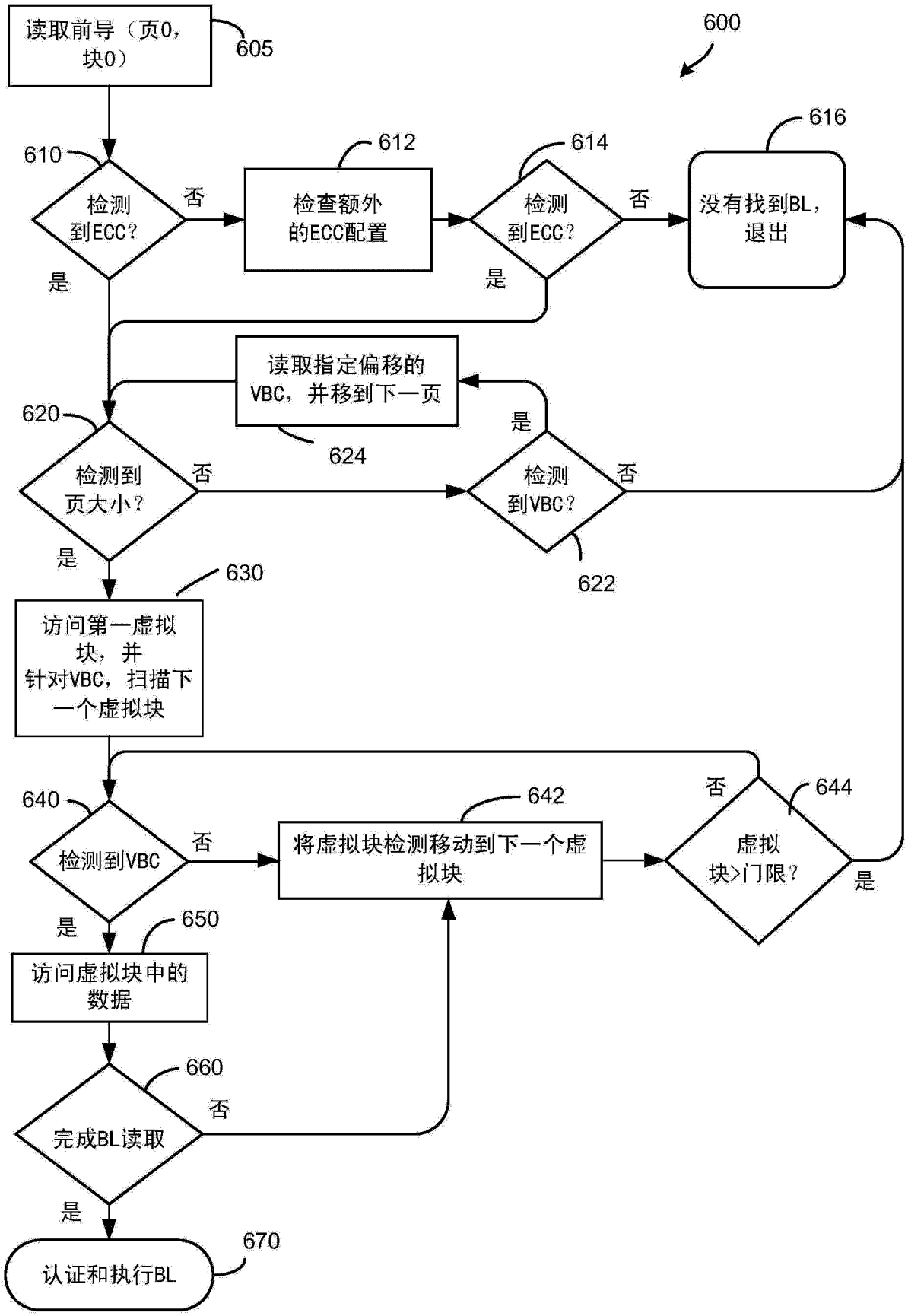

600
读取前导（页0，块0）
605
610
检测到ECC？
否
612
检查额外的ECC配置
614
检测到ECC？
否
616
没有找到BL，退出
是
是
读取指定偏移的VBC，并移到下一页
624
620
检测到页大小？
否
是
检测到VBC？
否
是
622
630
访问第一虚拟块，并针对VBC，扫描下一个虚拟块
640
检测到VBC
否
642
将虚拟块检测移动到下一个虚拟块
否
644
虚拟块>门限？
是
是
650
访问虚拟块中的数据
660
完成BL读取
否
是
670
认证和执行BL

图 6

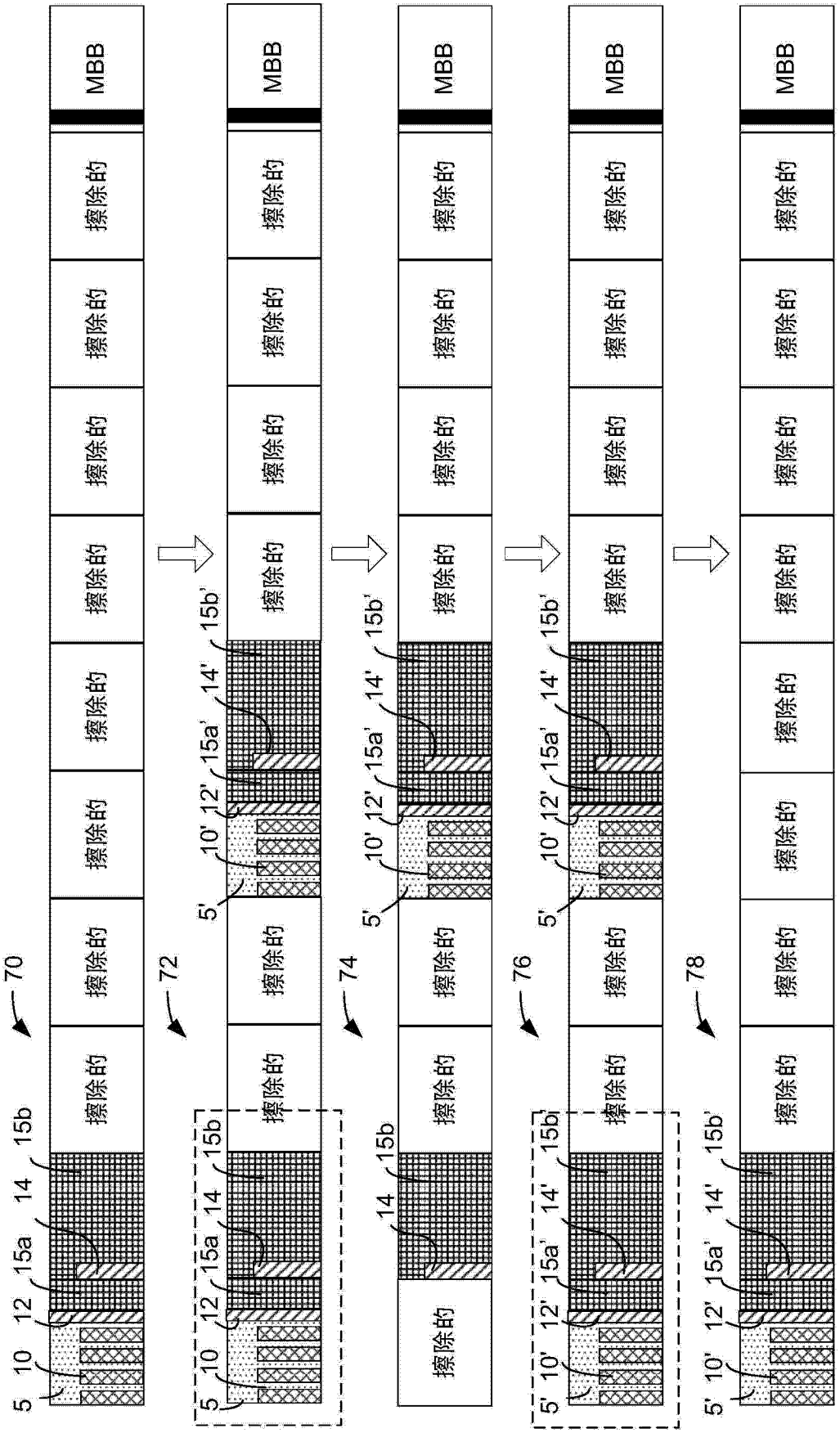


图 7

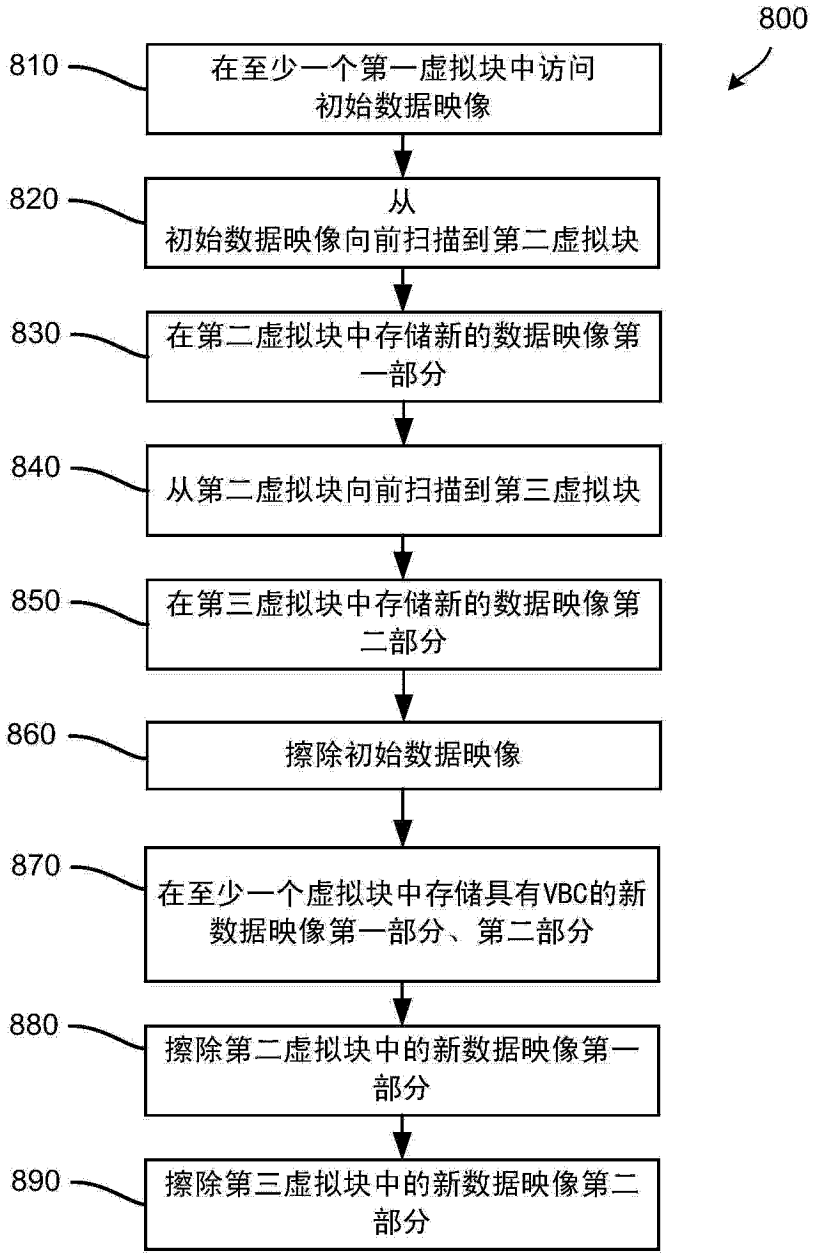

800
810
在至少一个第一虚拟块中访问初始数据映像
820
从初始数据映像向前扫描到第二虚拟块
830
在第二虚拟块中存储新的数据映像第一部分
840
从第二虚拟块向前扫描到第三虚拟块
850
在第三虚拟块中存储新的数据映像第二部分
860
擦除初始数据映像
870
在至少一个虚拟块中存储具有VBC的新数据映像第一部分、第二部分
880
擦除第二虚拟块中的新数据映像第一部分
890
擦除第三虚拟块中的新数据映像第二部分

图 8

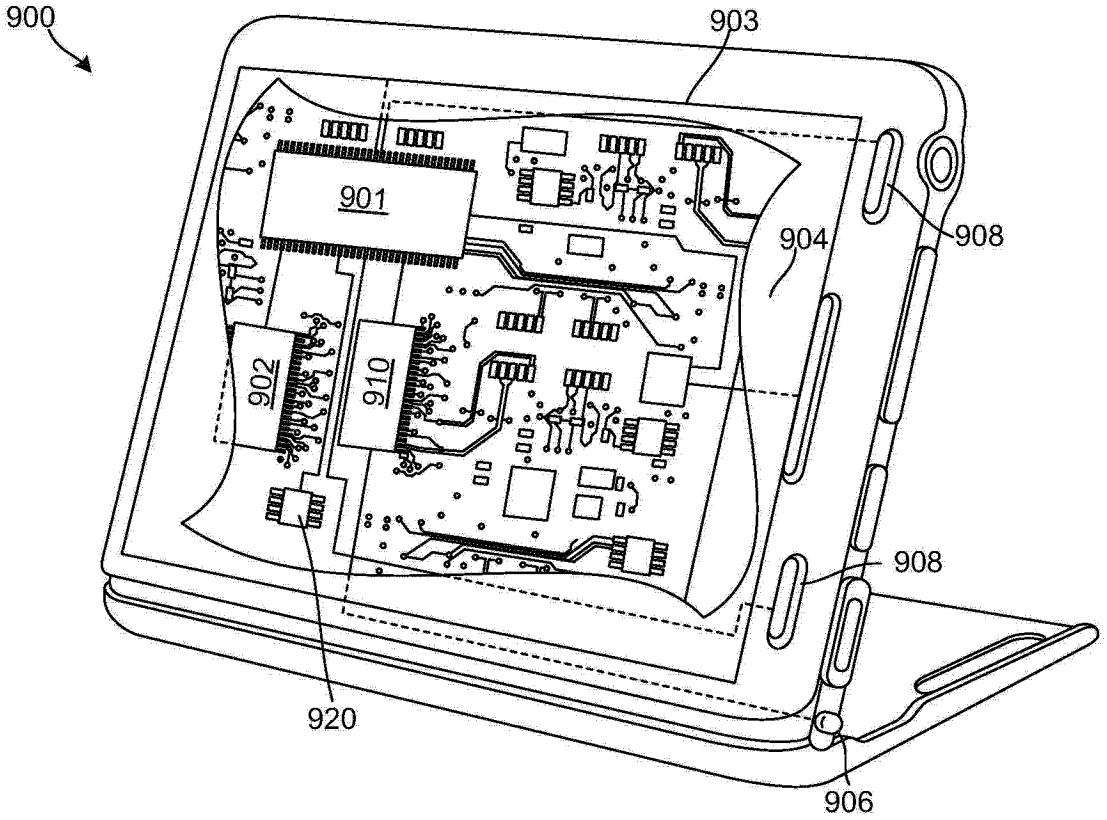


图 9

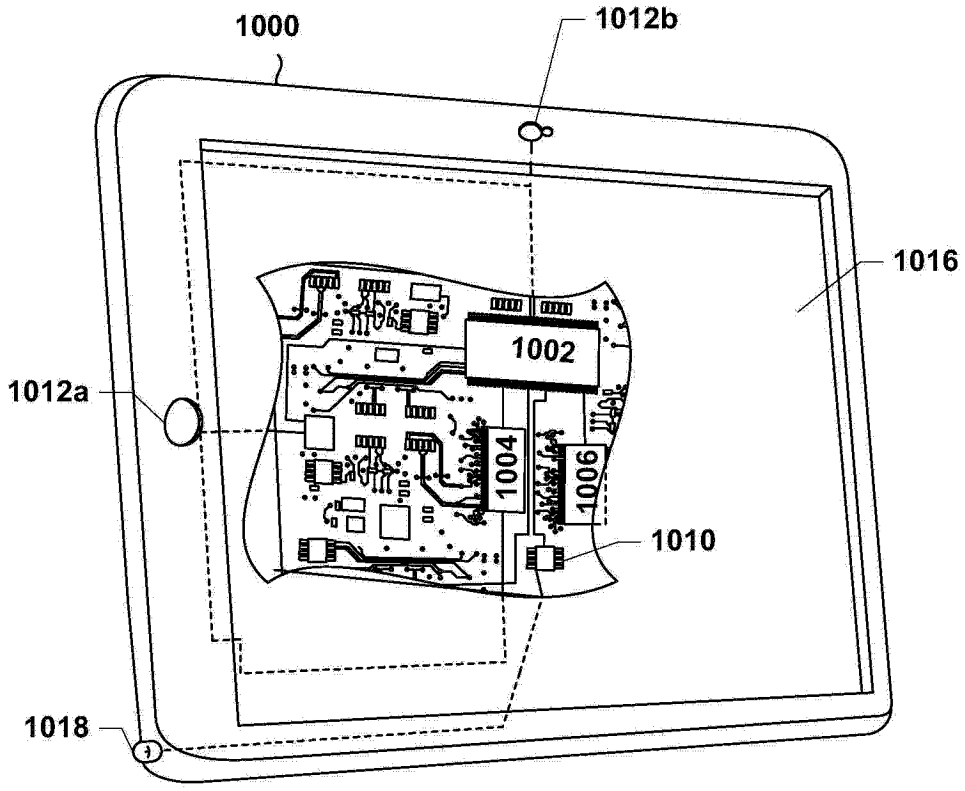


图 10

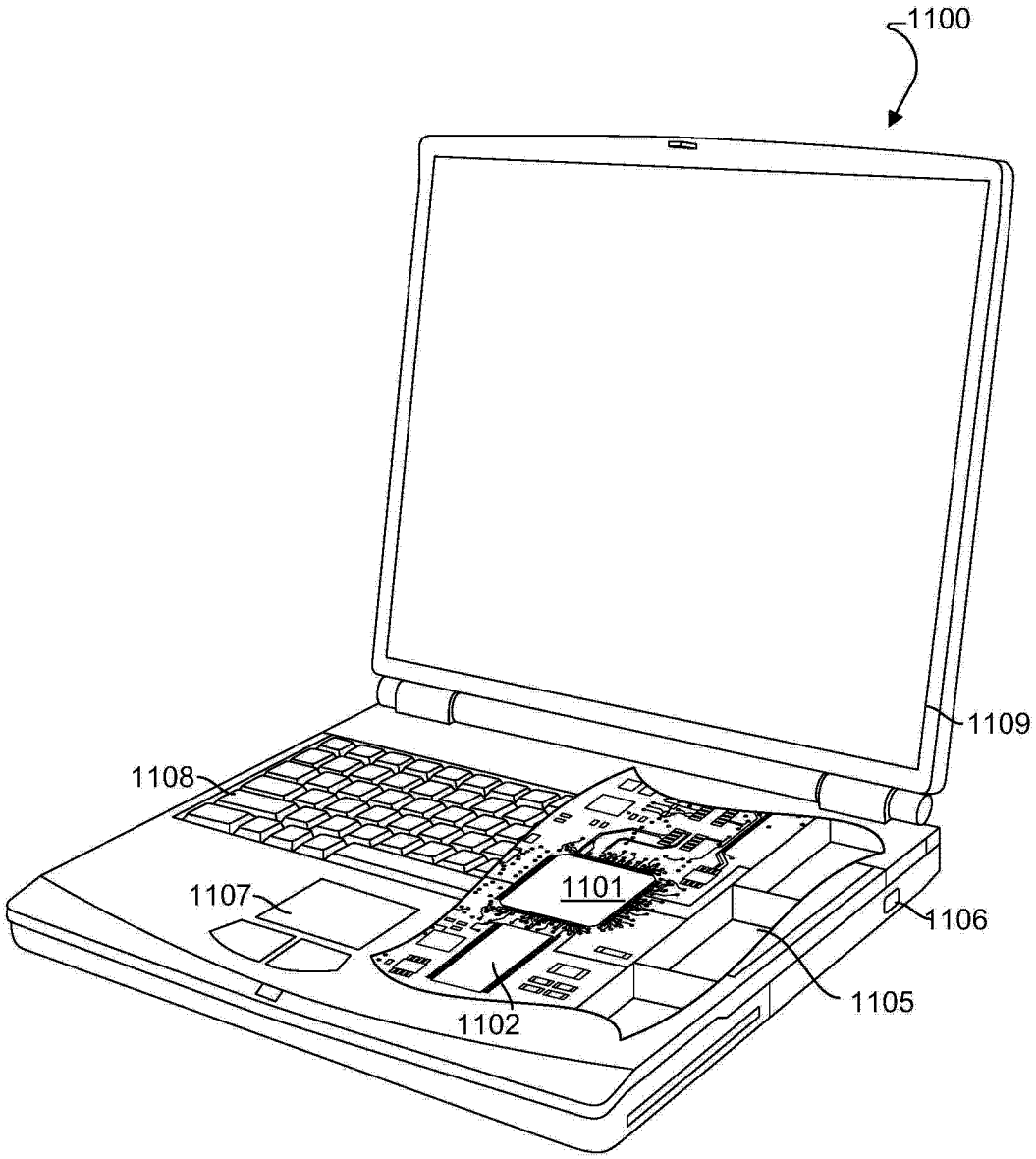


图 11