(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 102067606 A
(43)申请公布日 2011.05.18

(21)申请号 200880104691.2

(22)申请日 2008.09.26

(30)优先权数据
11/861,804 2007.09.26 US

(85)PCT申请进入国家阶段日
2010.02.26

(86)PCT申请的申请数据
PCT/US2008/077988 2008.09.26

(87)PCT申请的公布数据
WO2009/042943 EN 2009.04.02

(71)申请人 高通股份有限公司
地址 美国加利福尼亚州

(72)发明人 拉戈哈文德拉·C·纳加拉杰
徐迪藻 斯蒂芬·莫洛伊

(74)专利代理机构 北京律盟知识产权代理有限责任公司 11287
代理人 刘国伟

(51)Int.Cl.
H04N 7/30(2006.01)

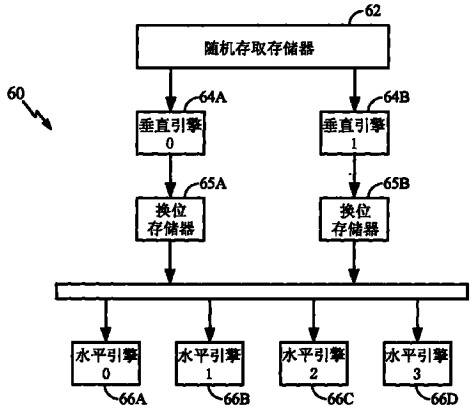
权利要求书 3 页 说明书 20 页 附图 12 页

(54)发明名称

用于视频编码的高效变换技术

(57)摘要

本发明描述可在视频编码中使用的高效变换技术。明确地说，在第二视频数据块的变换中，再次使用与第一视频数据块的变换相关联的计算的中间结果。可在运动估计过程期间使用所述技术，其中，在该运动估计过程中，变换搜索空间的视频块，但本发明并不限于此方面。可使用管线输送技术来加速高效变换技术，且可实施换位存储器以促进高效的管线输送。



CN 102067606 A

1. 一种装置，其包含对视频数据块执行变换的视频编码器，其中在执行所述变换时，所述视频编码器在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

2. 根据权利要求 1 所述的装置，其中所述视频编码器执行来自由整数和离散余弦变换 (DCT) 组成的群组的变换。

3. 根据权利要求 1 所述的装置，其中所述视频编码器包括运动估计器，且所述运动估计器包括执行所述变换的变换引擎。

4. 根据权利要求 3 所述的装置，其中所述第一视频数据块和所述第二视频数据块包含与搜索空间相关联的视频数据，其中所述变换引擎还对待编码的视频数据块执行变换。

5. 根据权利要求 1 所述的装置，其中所述视频编码器对搜索空间内所定义的四个或四个以上视频数据块执行变换，其中所述视频编码器再次使用所述搜索空间内所定义的所述视频数据块中的至少三个视频数据块的所述变换中的计算。

6. 根据权利要求 1 所述的装置，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个行进行的 1 维变换以产生中间结果，以及对所述中间结果的一列进行的 1 维变换；且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

7. 根据权利要求 6 所述的装置，其中所述视频编码器包括执行所述变换的一个或一个以上水平引擎以及一个或一个以上垂直引擎。

8. 根据权利要求 7 所述的装置，所述视频编码器进一步包含位于所述一个或一个以上水平引擎与所述一个或一个以上垂直引擎之间的一个或一个以上换位存储器，以缓冲并解决来自所述引擎中的一者的输出数据相对于到达所述引擎中的一者的输入数据的时序。

9. 根据权利要求 8 所述的装置，其中所述变换相对于一组视频数据块以管线方式执行，且其中所述水平引擎与所述垂直引擎针对所述变换的至少一部分同时操作。

10. 根据权利要求 8 所述的装置，其中所述视频编码器：

相对于所述输出数据，跟踪与所述输入数据相关联的索引值；且

经由所述索引值，相对于所述输出数据重新排序所述输入数据。

11. 根据权利要求 7 所述的装置，所述视频编码器进一步包含位于所述一个或一个以上水平引擎与所述一个或一个以上垂直引擎之间的一组换位寄存器，以缓冲并解决来自所述引擎中的一者的输出数据相对于到达所述引擎中的一者的输入数据的时序。

12. 根据权利要求 11 所述的装置，其中所述变换相对于一组视频数据块以管线方式执行，且其中所述水平引擎与所述垂直引擎针对所述变换的至少一部分同时操作。

13. 根据权利要求 11 所述的装置，其中所述视频编码器：

相对于所述输出数据，跟踪与所述输入数据相关联的索引值；且

经由所述索引值，相对于所述输出数据重新排序所述输入数据。

14. 根据权利要求 1 所述的装置，其中所述装置包含无线通信装置。

15. 根据权利要求 1 所述的装置，其中所述装置包含高分辨率电视 (HDTV)。

16. 根据权利要求 1 所述的装置，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个列进行的 1 维变换以产生中间结果，以及对所述中间结果的一行进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

17. 一种方法，其包含对视频数据块执行变换，其中执行所述变换包括在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

18. 根据权利要求 17 所述的方法，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个行进行的 1 维变换以产生中间结果，以及对所述中间结果的一列进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

19. 根据权利要求 17 所述的方法，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个列进行的 1 维变换以产生中间结果，以及对所述中间结果的一行进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

20. 一种装置，其包含 ：

用于对视频数据块执行变换的装置 ；以及

用于在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算的装置。

21. 根据权利要求 20 所述的装置，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个行进行的 1 维变换以产生中间结果，以及对所述中间结果的一列进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

22. 一种计算机可读媒体，其包含当在视频编码装置中执行时致使所述装置对视频数据块执行变换的指令，其中在执行所述变换时，所述指令致使所述装置在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

23. 根据权利要求 22 所述的计算机可读媒体，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个行进行的 1 维变换以产生中间结果，以及对所述中间结果的一列进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

24. 一种电路，其经配置以对视频数据块执行变换，其中在执行所述变换时，所述电路在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

25. 根据权利要求 24 所述的电路，

其中所述视频数据块包含 4 乘 4 像素块，

其中所述变换包含对所述 4 乘 4 像素块的多个行进行的 1 维变换以产生中间结果，以及对所述中间结果的一列进行的 1 维变换，且

其中所述被再次使用的计算包含所述中间结果中的至少一些中间结果。

用于视频编码的高效变换技术

技术领域

[0001] 本发明涉及数字视频处理，且更明确地说，涉及对视频数据的基于块的编码。

背景技术

[0002] 可将视频能力并入各式各样的装置中，包括数字电视、数字直播系统、无线通信装置、个人数字助理 (PDA)、膝上型计算机、桌上型计算机、数码相机、数字记录装置、蜂窝式或卫星无线电电话、视频游戏顾问 (counsel)、手持式游戏装置等。数字视频编码可在创建、修改、传输、存储、记录和播放全运动多媒体序列方面提供优于常规模拟系统的显著改进。广播网络可使用视频编码来促进向无线订户装置广播多媒体（音频－视频）序列的一个或一个以上信道。还使用视频编码来支持视频电话 (VT) 应用，例如通过蜂窝式无线电电话召开视频会议。

[0003] 已针对编码数字视频序列建立了许多不同的编码标准。举例来说，运动图片专家组 (MPEG) 已开发出包括 MPEG-1、MPEG-2 和 MPEG-4 在内的许多标准。其它标准包括国际电信联盟 (ITU)H.263 标准和 H.264 标准、加利福尼亚州库珀蒂诺市的苹果公司 (Apple Computer) 开发的 QuickTime™ 技术、华盛顿州雷蒙德市的微软公司 (MicrosoftCorporation) 开发的 Windows™ 的视频、英特尔公司 (Intel Corporation) 开发的 Indeo™、来自华盛顿州西雅图市的真实网络公司 (RealNetworks，Inc.) 的 RealVideo™ 以及 SuperMac 公司开发的 Cinepak™。此外，新的标准不断涌现和演化。MPEG-4 的第 10 部分“高级视频编码 (AVC)”中也陈述 ITU H.264 标准。

[0004] 多数视频编码技术利用基于块的编码，其将视频帧划分为像素块，且使所述块与视频序列中的其它帧的块相关。通过对当前块与另一帧的预测块之间的差异进行编码，可实现数据压缩。通常使用术语“宏块”来定义视频帧的与搜索空间（其通常为视频序列的前一或后一帧的子组）进行比较的离散块。还可将宏块进一步再分为多个分区或子分区。ITU H.264 标准支持 16 乘 16 宏块、16 乘 8 分区、8 乘 16 分区、8 乘 8 分区、8 乘 4 子分区、4 乘 8 子分区以及 4 乘 4 子分区。其它标准可支持不同大小的块、宏块、分区和 / 或子分区。

[0005] 对于视频帧中的每一块（宏块、分区或子分区），编码器将一个或一个以上紧接在前的视频帧（和 / 或后续帧）的大小类似的块进行比较，以识别类似块，其被称作“预测块”或“最佳匹配”。将当前视频块与其它帧的视频块进行比较的过程通常被称作运动估计。一旦针对待编码的给定块识别出“最佳匹配”，编码器就可对当前块与最佳匹配之间的差异进行编码。此对当前块与最佳匹配之间的差异进行编码的过程包括被称作运动补偿的过程。运动补偿包含创建差异块（称作残差），其包括指示待编码的当前块与最佳匹配之间的差异的信息。明确地说，运动补偿通常涉及使用运动向量取得最佳匹配且接着从输入块中减去最佳匹配以产生残差的动作。可对残差执行例如熵编码等额外编码步骤以进一步压缩位流。

发明内容

[0006] 本发明描述可在视频编码中使用的高效变换技术。明确地说，在计算与第二视频

数据块的变换相关联的计算的中间结果时，再次使用与第一视频数据块的变换相关联的计算的中间结果。可在运动估计过程期间使用高效变换技术，其中搜索空间的视频块被变换，但本发明不必在此方面受限。根据本发明，搜索空间可拆分为多个不同的 4 乘 4 像素块，且所述不同的 4 乘 4 像素块可彼此重叠。

[0007] 可对 4 乘 4 像素块的多个行执行一维变换以产生中间结果，且接着可对所述中间结果的一列执行一维变换。或者，可首先对多个列执行一维变换，且接着对中间结果的一行执行一维变换。在任意情况下，在已知搜索空间中的不同 4 乘 4 像素块之间的重叠的情况下，可再次使用（例如，与后面的变换共享）所述中间结果中的至少一些结果，而无需执行相同计算。还揭示一种用于实施本文中所描述的技术的高效架构。

[0008] 在一个实例中，本发明提供一种包含对视频数据块执行变换的方法，其中执行变换包括在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

[0009] 在另一实例中，本发明提供一种装置，其包含对视频数据块执行变换的视频编码器。在执行变换时，视频编码器在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

[0010] 在另一实例中，本发明提供一种装置，其包含：用于对视频数据块执行变换的装置；以及用于在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算的装置。

[0011] 本文中所描述的技术可以硬件、软件、固件或其任意组合实施。如果以软件实施，那么所述软件可在数字信号处理器 (DSP) 或其它类型的处理器或装置中执行。执行所述技术的软件最初可存储在计算机可读媒体中，且加载在处理器或其它装置中并在其中执行，以允许使用本文中所描述的技术来进行视频编码。

[0012] 因此，本发明还预期一种计算机可读媒体，其包含当在视频编码装置中执行时致使所述装置对视频数据块执行变换的指令，其中在执行变换时，所述指令致使所述装置在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

[0013] 此外，本发明预期一种经配置以对视频数据块执行变换的电路，其中在执行变换时，所述电路在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。

[0014] 另外，如下文中更详细地描述，可使用管线输送技术来加速高效变换技术，且可实施换位存储器以促进高效管线输送。附图和以下描述内容中陈述各种实施例的额外细节。根据所述描述内容、附图及所附权利要求书，本发明的其它特征、目标和优势将变得显而易见。

附图说明

[0015] 图 1 是说明可实施本发明的技术的视频编码装置的示范性视频编码器的框图。

[0016] 图 2 和图 3 是说明可实施包括变换的运动估计过程的一部分的组件的框图。

[0017] 图 4 是说明用于执行一维变换的蝶形实施方案的图。

[0018] 图 5 是说明可将计算共享技术用于如本文所述的变换的架构的框图。

[0019] 图 6 是说明可根据本发明而实现的计算节省的曲线图。

[0020] 图 7 是说明作为可并行搜索的搜索点数目的函数的变换引擎数目的曲线图。

[0021] 图 8 是说明示范性垂直引擎的框图。

[0022] 图 9 是说明可用以变换搜索空间的第一 4 乘 4 像素块的示范性水平引擎的框图。

[0023] 图 10 是说明可用以在由图 8 所示的水平引擎所执行的变换之后变换搜索空间的其余 4 乘 4 像素块的示范性水平引擎的框图。

[0024] 图 11 是搜索空间内的块的概念图，其说明到达垂直引擎中的输入块。

[0025] 图 12 是说明水平引擎与垂直引擎之间的数据流的图。

[0026] 图 13 是说明可位于水平引擎与垂直引擎之间的换位寄存器中的时序和数据流的框图。

具体实施方式

[0027] 本发明描述可在视频编码中有用的高效变换技术。如下文中更详细地描述，在第二视频数据块的变换中再次使用与第一视频数据块的变换相关联的计算的中间结果。所述技术可能尤其对在变换搜索空间的视频块的运动估计过程期间执行的整数变换或正向离散余弦变换有用。然而，可在与视频编码相关联的其它变换情况中使用所述技术。实际上，所述技术可对任意类型的线性变换、整数变换以及可能其它变换情况有用。

[0028] 根据本发明，可将（任意大小的）搜索空间拆分为例如 4 乘 4 像素块等多个不同视频块。搜索空间内所界定的不同 4 乘 4 像素块可彼此重叠。举例来说，5 乘 5 像素搜索空间可界定四个不同的 4 乘 4 像素块，但可使用对部分分辨率的内插来在 5 乘 5 像素搜索空间内界定更多的 4 乘 4 像素块。当将 4 乘 4 像素块从像素域变换到空间频域时，可使用搜索空间。在从一个域向另一域进行此变换期间，通常对 4 乘 4 像素块执行两遍一维变换。对多个列执行第一遍一维变换以产生水平空间频率分量（称作中间结果），且对一个或一个以上行执行第二遍一维变换以产生垂直空间频率分量。所属领域的技术人员将认识到，可轻易地对多个行执行第一遍一维变换，且可对一列执行第二遍一维变换。

[0029] 可对 4 乘 4 像素块的多个列执行一维变换以产生中间结果，且接着可对所述中间结果中的一行执行一维变换。在已知搜索空间中的不同 4 乘 4 像素块之间的重叠的情况下，可再次使用中间结果中的至少一些结果，而无需执行相同计算。以此方式，可避免计算以促进效率。还揭示示范性硬件架构，其可实现本文中所描述的技术的高效实施。在此情况下，可使用管线输送技术来加速一组视频数据块的高效变换技术，且可实施换位存储器以促进高效管线输送。

[0030] 图 1 是说明可实施本发明的技术的视频编码装置的示范性视频编码器 10 的框图。实际上，多种装置可实施可受益于本发明的教示的视频编码器。举例来说，视频编码器 10 可用于数字电视、数字直播系统、无线通信装置（例如，手持机）、个人数字助理 (PDA)、膝上型计算机、桌上型计算机、数码相机、数字记录装置、蜂窝式或卫星无线电电话、视频游戏顾问、手持式游戏装置等。广播网络可使用视频编码来促进向无线订户装置广播多媒体（音频 - 视频）序列的一个或一个以上信道。还使用视频编码来支持视频电话 (VT) 应用，例如通过蜂窝式无线电电话召开视频会议，以及多种其它应用。

[0031] 如图 1 所示，视频编码器 10 接收输入宏块 (MB)。通常使用术语“宏块”来定义视

频帧的与搜索空间进行比较并被编码的离散块。还可将宏块进一步再分为分区或子分区。ITU H.264 标准支持 16 乘 16 宏块、16 乘 8 分区、8 乘 16 分区、8 乘 8 分区、8 乘 4 子分区、4 乘 8 子分区以及 4 乘 4 子分区。其它标准可支持不同大小的块、宏块、分区和 / 或子分区。在任意情况下，虽然可使用术语“宏块”来描述本发明的方面，但本文中所描述的技术可在对包括宏块、分区、子分区或其它视频块大小的任意大小的视频数据块进行编码的过程中有用。

[0032] 如图 1 所示，对于每一输入 MB，产生一预测块（在图 1 中标记为“Pred”）。预测块有时被称作最佳匹配。经由单元 12 从输入 MB 中减去预测块以产生残差（在图 1 中标记为“Res”）。残差包含数据块，其指示输入 MB 与用以对输入 MB 进行编码的预测块之间的差异。预测块可由运动向量（或用于帧帧内编码的帧内向量）识别。对于帧内编码，预测块与输入 MB 位于相同的帧内。对于帧间编码，预测块与输入 MB 位于不同的帧内。运动向量（或帧内编码的帧内向量）识别编码时所使用的预测块。帧间编码可为预测性的 (P)，此表示预测块是基于前一帧的；或双向的 (B)，此表示预测块是基于视频序列的前一或后一帧的。

[0033] 在创建残差 (Res) 后，对残差执行变换和量化。变换单元 14 和量化单元 16 分别执行变换和量化。还可执行熵编码以产生输出位流。熵编码单元 18 执行熵编码，此可实现进一步压缩。熵编码可包括将代码指派给位组，以及使码长与概率匹配。视频编码中众所周知且常见各种类型的熵编码。

[0034] 在视频编码器 10 的预测回路中，逆量化单元 22 和逆变换单元 24 对残差执行逆量化和逆变换，以本质上使单元 12 和 14 所执行的变换和量化反向。通过加法器单元 26 将预测块加回到经重构的残差。这本质上在预测回路中重新创建输入 MB。可通过去块单元 28 来对经重构的 MB 的边缘进行滤波，且将其存储在存储器 30 中。

[0035] 量化原则上涉及减小经变换信号的动态范围。动态范围的减小影响通过熵编码产生的位的数目（速率）。此还在残差中引入损失，其可导致原始 MB 与经重构的 MB 略有不同。这些差异通常被称作量化误差或失真。量化的强度由量化参数决定。较大的量化参数导致较高的失真，但可降低编码率。

[0036] 预测回路可为帧内预测回路或帧间预测回路。MPEG-4 和 ITU H.263 通常仅支持帧间预测。ITU H.264 支持帧内预测和帧间预测两者。在图 1 的视频编码器 10 中，控制信号 32 可选择回路为帧内预测还是帧间预测。然而，本发明的技术还可在仅支持帧内编码或帧间编码的系统中起作用。

[0037] 在帧内预测中，单元 34 执行空间估计和空间补偿。在此情况下，单元 34 将经重构的 MB 与同一视频帧内的相邻宏块进行比较，以产生帧内预测器块。帧内预测器块本质上是经重构的 MB 的最佳匹配，其将导致残差中的良好压缩。帧内预测可有助于减少空间冗余。

[0038] 在帧间预测中，单元 36 执行运动估计和运动补偿。运动估计将经重构的 MB 与先前或未来帧的块进行比较，以产生帧间预测器。帧间预测器是经重构的 MB 的最佳匹配，但不同于帧内预测器，帧间预测器来自不同的视频帧。帧间预测可有助于减少时间冗余。通常，与利用空间冗余相比，利用时间冗余可对视频序列的压缩产生较大的影响。换句话说，MB 的帧间编码通常实现优于帧内编码的压缩。

[0039] 本发明的技术大体与例如正向离散余弦变换等变换有关。可在运动估计过程期间实施所述技术，但本发明在此方面不受限制。为描述起见，本发明将所述技术描述为在运动

估计期间执行，但还可在执行变换的其它情况下使用这些技术或类似技术。

[0040] 运动估计是可由视频编码器执行的计算密集型过程。大量计算可能归因于运动估计中可能考虑的大量潜在预测器。实际上，运动估计通常涉及在包含一个或一个以上先前帧（或后续帧）的子组的搜索空间中搜索帧间预测器。可基于成本函数或量度来检查来自搜索空间的候选者，其通常通过执行差异计算来界定，例如绝对差和 (SAD)、平方差和 (SSD)、绝对经变换差和 (SATD) 或平方经变换差和 (SSTD)。一旦计算出用于搜索空间中的所有候选者的量度，就可选择使所述量度减到最小的候选者作为帧间预测器。因此，影响运动估计的主要因素可为搜索空间的大小、搜索方法以及各种成本函数。成本函数本质上量化了当前帧的原始块与搜索区域的候选块之间的冗余。可在准确率和失真方面量化冗余。

[0041] 图 2 展示基于变换的量度框架。图 2 中所示的块可包含由运动估计器（例如图 1 所示的单元 36）执行的功能。图 2 中说明示范性位长度，但本发明的技术不限于任意特定位长度。此外，运动估计涉及将待编码的块与搜索空间内的多个候选者进行比较，以找出所述搜索空间内与待编码的块最匹配的一个块。根据本发明，可在速率和失真意义上界定最佳匹配。

[0042] 为了完成运动估计，分析待编码的块与给定搜索空间块之间的残余能量。通过从相应搜索空间块的像素中减去待编码的块的对应像素的过程来获得每一残余候选者。这是差异 (Diff) 模块 42 在图 2 中（例如）经由 SAD、SSD、SATD 或 SSTD 技术完成的过程。接着，使用可执行正向离散余弦变换的正向变换引擎 (FTE)44 将残余块变换到频域中。可接着分析经变换的残余块的速率和失真特性。明确地说，速率失真估计 (RDE) 模块 46 本质上估计在给定量化参数 QP 下针对此给定搜索空间块将导致的速率 (R) 和失真 (D)。RDE 模块 46 接着基于拉格朗日原理将 R 和 D 组合成单个成本量度（$= D+\lambda R$），鉴于量度至少部分地依据 R 和 D，所述成本量度可被称作速率 - 失真量度。可将所有候选者搜索空间块的成本进行比较，且选择导致最小成本的任一候选者来进行编码。

[0043] 应注意，上文所计算的成本量度对应于正被编码的块以及具有 4 乘 4 个像素的大小的搜索空间块。对于大于 4 乘 4 个像素的块大小，可使用多个 4 乘 4 像素块来跨越较大的块。在此情况下，通过累加跨越较大块的所有 4 乘 4 个单元块的成本量度来计算所述较大块的成本。以下描述集中于 4 乘 4 个像素的块大小，但所述技术可应用于其它大小的块。

[0044] 正向变换引擎 (FTE)44 通常为基于变换的量度计算的任意基础模块。由于变换的线性本质，可切换差异模块 42 与 FTE 44 的级。根据本发明，切换差异模块 42 与 FTE 44 的次序可允许变换期间的计算节省。对所描绘的块的输入和输出的记法为输入的（列）×（行）×（用以表示值的位的数目）。

[0045] 图 3 展示图 2 的替代方案，其中图 2 的差异模块 42 与 FTE 44 的次序被切换。图 3 的组件形成运动估计器单元的一部分。出于示范性目的而说明位长度，但可使用其它位长度。在图 3 中，两个不同的 FTE 52A 和 52B（例如）经由整数变换或正向离散余弦变换来变换待编码的块（“编码块”）以及正被考虑的搜索空间块。在此情况下，在变换之后由差异单元 54 来执行差异计算。RDE 模块 56 接着估计在给定量化参数 QP 下针对此给定搜索空间块将导致的速率 (R) 和失真 (D)。RDE 模块 56 可基于拉格朗日原理，将 R 和 D 组合成单个成本（$= D+\lambda R$）。接着可在运动估计单元（例如，图 1 的单元 36）中，比较所有候选者搜索空间块的成本，且可选择导致最小成本的任一候选者来进行编码。

[0046] 运动估计的基本问题是从搜索空间 (s) 中找出待编码的块 (编码块，e) 的“最佳匹配”。可将编码块 (e) 和搜索空间 (s) 定义为 :

[0047]

$$e=\begin{bmatrix} e_{00} & e_{01} & e_{02} & e_{03} \\ e_{10} & e_{11} & e_{12} & e_{13} \\ e_{20} & e_{21} & e_{22} & e_{23} \\ e_{30} & e_{31} & e_{32} & e_{33} \end{bmatrix} \quad s=\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} & s_{04} \\ s_{10} & s_{11} & s_{12} & s_{13} & s_{14} \\ s_{20} & s_{21} & s_{22} & s_{23} & s_{24} \\ s_{30} & s_{31} & s_{32} & s_{33} & s_{34} \\ s_{40} & s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix} \tag{1}$$

[0048] 如可看到，可相对于 s 中的四个搜索点来匹配 e，其可描绘为 :

[0049]
$$s=\begin{bmatrix} s(0,0) & s(0,1) \\ s(1,0) & s(1,1) \end{bmatrix} \tag{2}$$

[0050] 其中

[0051]

$$s(0,0)=\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \tag{3}$$

[0052]

$$s(0,1)=\begin{bmatrix} s_{01} & s_{02} & s_{03} & s_{04} \\ s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \end{bmatrix} \tag{4}$$

[0053]
$$s(1,0)=\begin{bmatrix} s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \\ s_{40} & s_{41} & s_{42} & s_{43} \end{bmatrix} \tag{5}$$

[0054]
$$s(1,1)=\begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix} \tag{6}$$

[0055] 注意，(搜索) 点 (例如，s(0,0)、s(0,1)、s(1,0) 或 s(1,1)) 被描绘为具有相等的水平与垂直维度的块。s(0,0) 可被称作块 00，s(0,1) 可称作块 01，s(1,0) 可称作块 10，且 s(1,1) 可称作块 11。搜索点还可描绘具有不相等的水平与垂直维度的块。图 11 展示具有

出于说明性目的而展示的与此实例一致的一些所界定的搜索点和块的 8×8 搜索区域。为在 s 中找出 e 的最佳匹配，可将残余块 r(x, y) 计算为：

[0056] $r(x,y) = e-s(x,y) \quad x,y \in 0,1$ (7)

[0057] 接着，经由变换矩阵 $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$ 将残余块 r(x, y) 变换为空间频域。

[0058] $$\begin{bmatrix} R'_{0v}(x,y) \\ R'_{1v}(x,y) \\ R'_{2v}(x,y) \\ R'_{3v}(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} r_{0v}(x,y) \\ r_{1v}(x,y) \\ r_{2v}(x,y) \\ r_{3v}(x,y) \end{bmatrix} \quad v \in 0,1,2,3 \qquad (8)$$

[0059]

$$\begin{bmatrix} R_{h0}(x,y) \\ R_{h1}(x,y) \\ R_{h2}(x,y) \\ R_{h3}(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} R'_{h0}(x,y) \\ R'_{h1}(x,y) \\ R'_{h2}(x,y) \\ R'_{h3}(x,y) \end{bmatrix} \quad h \in 0,1,2,3 \qquad (9)$$

[0060] 在等式 8 中，变量 v 表示垂直列，且在等式 9 中，变量 h 表示水平行。变换矩阵由整阵列成，在一些情况下，其被称作整数变换，且在其它情况下，其被称作离散余弦变换。离散余弦变换 (DCT) 可为整数变换或“实数”变换。所属领域的技术人员将认识到，变换矩阵可由整数或实数形成。可注意到，可使用 4 点一维 (1-D) 变换来产生视频块的空间频域分量。可首先对所有列应用 4 点 1-D 变换以产生第一遍中间结果，且接着可在第二遍中对中间结果的所有行应用 4 点 1D 变换。图 4 中展示 1-D 变换的“蝶形实施方案”，其使用一组加法器以不同方式组合输入从而产生不同输出。蛮力法将针对 R(0,0)、R(0,1)、R(1,0) 或 R(1,1) 的计算使用八个单独的 1-D 变换。在蛮力法中，独立地处理每一搜索点，且不存在对中间结果 R′ (0,0)、R′ (0,1)、R′ (1,0)、R′ (1,1) 的再次使用。对于高分辨率电视 (HDTV)，如果仅使用一个垂直引擎和一个水平引擎来搜索整个帧，那么通过量可能过慢而无法促进视频帧的实时呈现。因此，如下文中更详细地描述，对于 HDTV 或其它应用，为了加快搜索速度，可并行使用多个垂直引擎和水平引擎。

[0061] 如图 4 所示，可将一个 1-D 四输入变换视为具有若干个级。变量 x_0、x_1、x_2 和 x_3 表示变换的输入，且 y_0、y_1、y_2 和 y_3 表示变换的四个经变换的输出。值“a”和“c”为乘法常数。因此，上方具有“c”的箭头表示所述箭头的输出是使输入值（箭头左侧）与值“c”相乘的结果。值“p”表示字长，即，进入各级中的位的数目。

[0062] 如果如图 3 所示使用经修改的框架，那么可再次使用变换的中间结果。明确地说，与等式 (7)、(8) 和 (9) 形成对比，可如下获得经二维变换的 R(x, y)。

[0063] $R(x,y) = E-S(x,y) \quad x,y \in 0,1$ (10)

[0064] 其中

[0065]

$$\begin{bmatrix} S'_{0v}(x,y) \\ S'_{1v}(x,y) \\ S'_{2v}(x,y) \\ S'_{3v}(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} s_{0v}(x,y) \\ s_{1v}(x,y) \\ s_{2v}(x,y) \\ s_{3v}(x,y) \end{bmatrix} \quad v \in 0,1,2,3 \qquad (11)$$

[0066]

$$\begin{bmatrix} S_{h0}(x,y) \\ S_{h1}(x,y) \\ S_{h2}(x,y) \\ S_{h3}(x,y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} S'_{h0}(x,y) \\ S'_{h1}(x,y) \\ S'_{h2}(x,y) \\ S'_{h3}(x,y) \end{bmatrix} \quad h \in 0,1,2,3 \qquad (12)$$

[0067]

$$\begin{bmatrix} E'_{0v} \\ E'_{1v} \\ E'_{2v} \\ E'_{3v} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} e_{0v} \\ e_{1v} \\ e_{2v} \\ e_{3v} \end{bmatrix} \quad v \in 0,1,2,3 \qquad (13)$$

[0068]

$$\begin{bmatrix} E_{h0} \\ E_{h1} \\ E_{h2} \\ E_{h3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} E'_{h0} \\ E'_{h1} \\ E'_{h2} \\ E'_{h3} \end{bmatrix} \quad h \in 0,1,2,3 \qquad (14)$$

[0069] 因此，变换可由 r(x,y) 的变换变为 s(x,y) 的变换。此允许利用 s(0,0)（等式 3）与 s(0,1)（等式 4）的列重叠，且利用 s(1,0)（等式 5）与 s(1,1)（等式 6）的列重叠。可再次使用 S′ (0,0) 的第一遍中间结果（即，与列相关联的结果）以避免对 S′ (0,1) 的第一遍中间结果（即，与列相关联的结果）的重复计算。类似地，可再次使用 S′ (1,0) 的第一遍中间结果（即，与列相关联的结果）以避免对 S′ (1,1) 的第一遍中间结果（即，与列相关联的结果）的重复计算。下文更详细地阐释对中间结果的此再次使用（也称作共享）。

[0070] 为了说明再次使用（例如，共享）概念，作为实例考虑，可将经二维变换的 s(0,0) 计算为：

[0071]

$$\begin{bmatrix} S'_{0v}(0,0) \\ S'_{1v}(0,0) \\ S'_{2v}(0,0) \\ S'_{3v}(0,0) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} s_{0v}(0,0) \\ s_{1v}(0,0) \\ s_{2v}(0,0) \\ s_{3v}(0,0) \end{bmatrix} \quad v \in 0,1,2,3 \qquad (15)$$

[0072]

$$\begin{bmatrix} S_{h0}(0,0) \\ S_{h1}(0,0) \\ S_{h2}(0,0) \\ S_{h3}(0,0) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} S'_{h0}(0,0) \\ S'_{h1}(0,0) \\ S'_{h2}(0,0) \\ S'_{h3}(0,0) \end{bmatrix} \quad h \in 0,1,2,3 \qquad (16)$$

[0073] 此外，可将 S(0,1) 计算为：

[0074]

$$\begin{bmatrix} S'_{0v}(0,1) \\ S'_{1v}(0,1) \\ S'_{2v}(0,1) \\ S'_{3v}(0,1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} s_{0v}(0,1) \\ s_{1v}(0,1) \\ s_{2v}(0,1) \\ s_{3v}(0,1) \end{bmatrix} \quad v \in 0,1,2,3 \qquad (17)$$

[0075]

$$\begin{bmatrix} S_{h0}(0,1) \\ S_{h1}(0,1) \\ S_{h2}(0,1) \\ S_{h3}(0,1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} S'_{h0}(0,1) \\ S'_{h1}(0,1) \\ S'_{h2}(0,1) \\ S'_{h3}(0,1) \end{bmatrix} \quad h \in 0,1,2,3 \qquad (18)$$

[0076] 根据等式 (3) 和 (4) 可注意到：

[0077] $s_{v(n)}(0,0) = s_{v(n-1)}(0,1)$ $v \in 0,1,2,3$ $n \in 1,2,3$ (19)

[0078] 因此，得出：

[0079] $S'_{v(n)}(0,0) = S'_{v(n-1)}(0,1)$ $v \in 0,1,2,3$ $n \in 1,2,3$ (20)

[0080] 此表示对于 $n \in 1,2,3$，可再次使用与 $S'(0,0)$ 相关联的第一遍中间结果来计算与 $S'(0,1)$ 相关联的第一遍中间结果。可注意到，再次使用可消除原本在计算 $S'(0,1)$ 时所需要的八个 1-D 列变换中的三个 1-D 列变换。因此，在第一遍期间可实现 1-D 变换的至多达 37.5 个百分比的节省。应指出，还可通过使 s(0,0) 的行与 s(1,0) 的行重叠来处理视频块，在所述情况下，可在第一遍时对行应用 1-D 行变换，之后对第一遍所得到的列进行第二遍 1-D 列变换。换句话说，可不考虑水平变换与垂直变换的次序而应用本发明的技术。

[0081] 一般来说，为了在 N×M 搜索空间中搜索 4×4 块的最佳匹配，总共存在 (N-3)×(M-3) 个搜索点。其中 N 表示水平方向上的总像素计数，且 M 表示垂直方向上的总像素计数。为了确保能够针对实时视频编码及时完成搜索最佳匹配的任务，可设计并同时运行多个 1D 引擎。假定并行使用“j+1”($0 \leq j \leq N-3$) 个水平引擎单元以加速搜索，且如果经垂直变换的数据被高效地共享，那么仅需要“k+1”个垂直引擎单元。其中 $k = (j+3)/4$ 的整数。在设计视频编码架构时，功率消耗和硅面积是变换引擎的性能的重要折衷因素。

[0082] 图 5 是说明可使用本文所述的计算再次使用技术进行变换以及管线输送技术的架构的框图。可使用图 5 所示的架构实例来实现 FTE 52B。此处，使用四个水平引擎 ($j = 3$)，且因此为了具有在此设计架构中共享的高效数据，可使用两个垂直引擎 ($k = 2$)。如

图 5 所示，FTE 60 包括随机存取存储器 (RAM)62、两个垂直引擎 (VE(0)64A 和 VE(1)64B)、两个换位存储器 TM 65A 和 TM 65B 以及四个水平引擎 (HE(0)66A、HE(1)66B、HE(2)66C 和 HE(3)66D)。

[0083] FTE 60 可在垂直和水平方向上实施 4 乘 4 正向变换。通常，首先执行垂直变换还是水平变换无关紧要。换句话说，可在其它实施方案中切换变换的序列（垂直和水平）。因此，在其它实施方案中，可在垂直变换之前执行水平变换。在图 5 中，垂直变换是第一变换运算，接着是水平变换。

[0084] 如图 5 所示，所述架构利用四个水平引擎 66A 到 66D 以及两个垂直引擎 64A 和 64B。RAM 62 可含有供应到垂直引擎 64A 和 64B 的搜索空间像素。垂直引擎 64A 和 64B 连接到换位存储器 65A 和 65B，且换位存储器 65A 和 65B 向水平引擎 66A 到 66D 供应数据。可以 1D 脉动阵列的形式对水平引擎 66A 到 66D 进行配置。在下文更详细描述的一些实例中，可基于 4×4 变换来设计水平引擎和垂直引擎。然而，变换大小可为任意大小（例如，N×N），且不同大小的变换可改变硬件引擎设计，但仍实现与再次使用中间计算相关联的优势。

[0085] 如上文中所注意，本文中所描述的技术能够在所执行的 1D 变换的数目方面实现至多达 37.5 个百分比的缩减。然而，图 5 所示的架构可实现仅 25 个百分比的缩减。随着水平引擎的数目增加，因为可能发生更多共享，因此计算缩减量可改进。为了维持十六个水平引擎，可能需要五个垂直引擎。在此情况下，计算缩减量可变为 1-[(5+16)/(16+16)] = 34.4%。一般来说，对于 N 个水平引擎，可能需要 1+Ceil((N-1)/4) 个垂直引擎。Ceil() 向正无穷大 (+ ∞) 舍入。可通过下式给出计算缩减的通用式：

[0086] 1-[（垂直引擎的数目 + 水平引擎的数目）/

[0087] （水平引擎的数目 + 水平引擎的数目）]

[0088] 图 6 是说明可根据本发明而实现的计算节省的曲线图。明确地说，图 6 用曲线表示可基于（在 x 轴上）并行搜索的点的数目而（在 y 轴上）实现的计算节省的百分比。一般来说，搜索点的数目可等于水平引擎的数目（假定首先执行垂直变换，且接着执行水平变换）。举例来说，16 个水平引擎应覆盖 16 个搜索点。因此，在图 6 的 x 轴上的值 16 处，此对应于曲线图上的 34.4% 节省。

[0089] 图 7 是说明作为可并行搜索的搜索点数目的函数的变换引擎的数目的曲线图。图 7 用曲线表示针对可（在 x 轴上）并行搜索的搜索点的不同数目的水平引擎的数目 (y 轴上)、垂直引擎的数目 (y 轴上) 以及水平引擎与垂直引擎的组合数目 (y 轴上)。

[0090] 此外，可在 FTE 中使用多个水平引擎以改进变换通过量和数据共享。为了确保 FTE 引擎可促进相对较小的区域内的高效数据共享和低功率消耗，可以不同方式设计 FTE 的垂直引擎和水平引擎。明确地说，为了节省时钟功率，通过以像素时钟速率的二分之一或四分之一进行计时来设计引擎中的大多数数据寄存器。表 1、表 2 和表 3（下文）说明可使用的计时方案。

[0091] 图 8 说明示范性垂直变换引擎 80（也称作垂直引擎）。垂直引擎 80 包括各种寄存器 R0(81A)、R1(81B)、P0(81C)、P1(81D)、P2(81E) 和 Rf(81F)；多路复用器 82A、82B、82C、82D、82E 和 82F；以及加法器 85A 和 85B。垂直引擎 80 直接从 RAM 62（图 5）接收像素数据。可用垂直引擎 80 来实施技术，其中垂直引擎 80 对输入像素数据进行重新定序、重新布置内部数据路径，且将所广播的像素输入数据锁存到指定的寄存器。

[0092] 表 1 展示垂直引擎 80 的针对搜索点 s(0,0) 以及搜索点 s(1,0) 的一部分（其变换在时钟周期十六处开始）的示范性数据流和时序。明确地说，在已知连续时钟周期内的输入 (I/P) 的情况下，表 1 展示这些不同寄存器的内容。输出 (O/P) 可对应于寄存器 Rf(81F) 的内容。此外，可使用“被划分的”时钟信号，在所述情况下，垂直引擎 80 的等效计时能力在每个时钟周期期间处理 3 个寄存器。

[0093] 一般来说，对于搜索点 s(i, j)，来自 RAM 62 的像素数据如下被重新定序 :s(i, j+1), s(i+3, j+1), s(i+1, j+1), s(i+2, j+1) ;1 ∈ 0,1,2,3。在时钟 2n 周期处启用寄存器 R0(81A)，以锁存像素数据 s(i, j+1), s(i+1, j+1), 1 ∈ 0,1,2,3。寄存器 R1(81B) 每两个时钟周期锁存输入像素数据。在时钟 2n+1 周期处启用寄存器 R0(81A) 时钟，以锁存像素数据 s(i+3, j+1), s(i+2, j+1), 1 ∈ 0,1,2,3。对于用以保存中间变换数据的寄存器 P081C、P181D、P281E，在周期 4n+1 处启用 P0(81C) 时钟，以锁存 s(i, j+1), s(i+3, j+1), P1(81D) 在周期 4n+2 处锁存 s(i,j+1)-s(i+3,j+1)，且 P2(81E) 在周期 4n+3 处锁存 s(i+1, j+1)+s(i+2, j+1)，在周期 4n+4 处锁存 s(i+1, j+1)-s(i+2, j+1)，此处 n ＝ 0、1、2、...，1 ∈ 0,1,2,3。使用寄存器 Rf(81F) 来保存最终变换结果，且在每个时钟周期启用寄存器 Rf(81F)。对于搜索点 s(i, j)，经垂直变换的输出序列为 S′ (i, j+1), S′ (i+2, j+1), S′ (i+1, j+1), s(i+3, j+1), 1 ∈ 0,1,2,3。

[0094] 表 1

[0095]

时间	I/P
0	s00
1	s00
1	s30
2	s30
2	s10
3	s10
3	s20
4	s20
4	s01
5	s01
5	s31
6	s31
6	s11
7	s11
7	s21
8	s21
8	s02
9	s02
9	s32
10	s32

垂直引擎 VE(0)输入和输出时序表

R1	R0	P0	P1	P2	垂直 O/P
	s00				
	s00				
s30	s00	s00+s30			
s30	s00	s00+s30			
s30	s10	s00+s30	s00-s30		
s30	s10	s00+s30	s00-s30		
s20	s10	s00+s30	s00-s30	s10+s20	s00+s30+s10+s20 (S'00)
s20	s10	s00+s30	s00-s30	s10+s20	s00+s30+s10+s20 (S'00)
s20	s01	s00+s30	s00-s30	s10-s20	s00+s30-s10-s20 (S'20)
s20	s01	s00+s30	s00-s30	s10-s20	s00+s30-s10-s20 (S'20)
s31	s01	s01+s31	s00-s30	s10-s20	2s00-2s30+s10-s20 (S'10)
s31	s01	s01+s31	s00-s30	s10-s20	2s00-2s30+s10-s20 (S'10)
s31	s11	s01+s31	s01-s31	s10-s20	s00-s30-2s10+2s20 (S'30)
s31	s11	s01+s31	s01-s31	s10-s20	s00-s30-2s10+2s20 (S'30)
s21	s11	s01+s31	s01-s31	s11+s21	s01+s31+s11+s21 (S'01)
s21	s11	s01+s31	s01-s31	s11+s21	s01+s31+s11+s21 (S'01)
s21	s02	s01+s31	s01-s31	s11-s21	s01+s31-s11-s21 (S'21)
s21	s02	s01+s31	s01-s31	s11-s21	s01+s31-s11-s21 (S'21)
s32	s02	s02+s32	s01-s31	s11-s21	2s01-2s31+s11-s21 (S'11)

[0096]

10	s12		s32	s02	s02+s32	s01-s31	s11-s21	2s01-2s31+s11-s21 (S'11)
11	s12		s32	s12	s02+s32	s02-s32	s11-s21	s01-s31-2s11+2s21 (S'31)
11	s22		s32	s12	s02+s32	s02-s32	s11-s21	s01-s31-2s11+2s21 (S'31)
12	s22		s22	s12	s02+s32	s02-s32	s12+s22	s02+s32+s12+s22 (S'02)
12	s03		s22	s12	s02+s32	s02-s32	s12+s22	s02+s32+s12+s22 (S'02)
13	s03		s22	s03	s02+s32	s02-s32	s12-s22	s02+s32-s12-s22 (S'22)
13	s33		s22	s03	s02+s32	s02-s32	s12-s22	s02+s32-s12-s22 (S'22)
14	s33		s33	s03	s03+s33	s02-s32	s12-s22	2s02-2s32+s12-s22 (S'12)
14	s13		s33	s03	s03+s33	s02-s32	s12-s22	2s02-2s32+s12-s22 (S'12)
15	s13		s33	s13	s03+s33	s03-s33	s12-s22	s02-s32-2s12+2s22 (S'32)
15	s23		s33	s13	s03+s33	s03-s33	s12-s22	s02-s32-2s12+2s22 (S'32)
16	s23		s23	s13	s03+s33	s03-s33	s13+s23	s03+s33+s13+s23 (S'03)
16	s10		s23	s13	s03+s33	s03-s33	s13+s23	s03+s33+s13+s23 (S'03)
17	s10		s23	s10	s03+s33	s03-s33	s13-s23	s03+s33-s13-s23 (S'23)
17	s40		s23	s10	s03+s33	s03-s33	s13-s23	s03+s33-s13-s23 (S'23)
18	s40		s40	s10	s10+s40	s03-s33	s13-s23	2s03-2s33+s13-s23 (S'13)
18	s20		s40	s10	s10+s40	s03-s33	s13-s23	2s03-2s33+s13-s23 (S'13)
19	s20		s40	s20	s10+s40	s10-s40	s13-s23	s03-s33-2s13+2s23 (S'33)
19	s30		s40	s20	s10+s40	s10-s40	s13-s23	s03-s33-2s13+2s23 (S'33)
20	s30		s30	s20	s10+s40	s10-s40	s20+s30	s10+s40+s20+s30 (S'10)
20	s11		s30	s20	s10+s40	s10-s40	s20+s30	s10+s40+s20+s30 (S'10)
21	s11		s30	s11	s10+s40	s10-s40	s20-s30	s10+s40-s20-s30 (S'30)
21	s41		s30	s11	s10+s40	s10-s40	s20-s30	s10+s40-s20-s30 (S'30)

[0097] 在表 1 中，第四列中所说明的方波表示系统时钟。垂直引擎 80 中的所有数据可寄存在此系统时钟的上升沿处。

[0098] 为了使水平引擎可高效地共享经垂直变换的数据，其必须以循序次序取用经 4×4 垂直变换的数据，使高效数据共享起作用的不同次序只有少数几种，此处所示的实例仅为其中一种。为了使功率消耗减到最小，期望水平引擎立即消耗经垂直变换的数据。换位寄存器 TM 65A、65B 经设计以临时存储并改组经垂直变换的数据。图 12 描绘垂直和水平引擎的输入和输出序列。图 13 描绘对水平引擎的经垂直变换的数据进行重新定序所需的最少 TM 寄存器。

[0099] 为了水平引擎（例如图 5 的引擎 66A 到 66D）中的高效处理，期望使对存储器的存取减到最少。此外，期望在弃用所有被存取的数据之前对其进行处理。为了高效地适应这些目标，可使用两种不同类型的水平引擎 HE(0)90 和 HE(j)100。HE(0)90 可对应于水平引擎 66A（图 5），且 HE(j)100 可对应于引擎 66B 到 66D 中的每一者。

[0100] 每一水平引擎取用四个数据串，其为数据序列 0、数据序列 1、数据序列 2 和数据序列 3。如下描绘所述序列：

[0101] 序列 0：S′ (i, j)，S′ (i+1, j)，S′ (i+2, j)，S′ (i+3, j)

[0102] 序列 1：S′ (i, j+1)，S′ (i+1, j+1)，S′ (i+2, j+1)，S′ (i+3, j+1)

[0103] 序列 2：S′ (i, j+2)，S′ (i+1, j+2)，S′ (i+2, j+2)，S′ (i+3, j+2)

[0104] 序列 3：S′ (i, j+3)，S′ (i+1, j+3)，S′ (i+2, j+3)，S′ (i+3, j+3)

[0105] S′ (i, j) 表示经垂直变换的像素数据。对于水平引擎 HE(0)，所有四个序列数据均从垂直引擎 VE(0) 输入。

[0106] 对于水平引擎 HE(1)100，其序列 0 和序列 1 输入数据来自 HE(0)90 的寄存器 91B 和 91C 的数据，序列 2 数据来自 HE(0)90 的 92H 的共享数据输出，且 HE(1)100 的序列 3 数据直接来自垂直引擎 VE(1)80。

[0107] 对于水平引擎 HE(2)100，其序列 0 数据输入来自 HE(0)90 的寄存器 91C，序列 1 数据来自 HE(0) 的 92H 的共享数据输出，序列 2 数据来自 HE(1)100 的共享数据输出 92H，且 HE(2) 的序列 3 数据直接来自垂直引擎 VE(1)80。

[0108] 对于水平引擎 HE(j)100（其中 $j \geq 3$），其输入数据序列 0、序列 1 和序列 2 分别使用其相邻水平引擎 HE(j-3)、HE(j-2) 和 HE(j-1) 的共享数据输出多路复用器 102H。HE(j) 的序列 3 数据始终直接来自垂直引擎 VE(k) 的输出。其中 $k = (j+3)/4$ 的整数。

[0109] 下文的表 2 和表 3 展示水平引擎 HE(0)90 和 HE(1)100 的输入、输出和内部控制时序。注意，每四个时钟周期仅启用一次 R0(91A)、R1(91B)、R2(91C)、R3(91D) 和 R0(101A)，且中间寄存器 P0(91E,101B)、P1(91F,101C) 每两个时钟周期锁存数据。

[0110] 图 9 说明示范性水平变换引擎 HE(0)90（还称作水平引擎 HE(0)）。图 10 说明示范性水平变换引擎 HE(j)100。图 9 的水平引擎 90 包括标记为 R0(91A)、R1(91B)、R2(91C)、R3(91D)、P0(91E)、P1(91F) 和 Rf(91G) 的各种寄存器；多路复用器 92A、92B、92C、92D、92E、92F、92G、92H；以及加法器 95A 和 95B。图 10 的水平引擎 100 包括标记为 R0(101A)、P0(101B)、P1(101C) 和 Rf(101D) 的各种寄存器；多路复用器 102A、102B、102C、102D、102E、102F、102G、102H 和 102I；以及加法器 105A 和 105B。应注意，图 8、图 9 和图 10 所示的寄存器可包含每一相应引擎中的物理结构，但存储结构可彼此类似。换句话说，例如，图 8 中的寄存器 R0(81A) 与图 9 中的寄存器 R0(91A) 不同。

[0111] 水平引擎 HE(0)90 可经设计以变换第一 4 乘 4 像素列块（来自 N×M 大小的搜索区域的搜索点 s(0,0)、s(1,0)、...、s(M-3,0)）。相比之下，可重复使用水平引擎 HE(j)100（图 10 所示）来变换列块中的剩余部分（搜索点 s(0, j)、s(1, j)、...、s(M-3, j)，($1 \leq j \leq N-3$)）；图 9 的水平引擎 HE(0)90 可对应于图 5 的 HE(0)66A，且图 9 的水平引擎 HE(j)100 可对应于图 5 的 HE(1)66B、HE(2)66C 和 HE(3)66D 中的每一者。

[0112] 水平引擎 HE(0)90 将从垂直引擎 VE(0)（图 5）广播的循序像素数据锁存到四个四分之一像素计时数据寄存器 R0(91A)、R1(91B)、R2(91C) 和 R3(91D) 中，执行中间变换功能，接着将结果存储在寄存器 P0(01E) 和 P1(91F) 中。寄存器 P0(01E) 和 P1(91F) 存储数据以用于第二蝶形运算，且将结果传递到最终寄存器 Rf(91G)。

[0113] 锁存在水平引擎 90 的寄存器 R0(91A)、R1(91B)、R2(91C) 和 R3(91D) 中的数据可由接下来的三个水平引擎 HE(1)、HE(2)、HE(3)（见 66B 到 66D 或图 5）共享。一般来说，对于每一水平引擎 HE(j)100（其中 $j = 1$、2 或 3），输入数据来自对 HE(0) 的输出、VE(1) 的输出和“先前”HE(j) 引擎的多路复用。

[0114] 表 2 展示时序以及如何在水平引擎 HE(0) 到 HE(3) 之间共享数据。明确地说，表 2 展示对搜索点 s(0,0)、s(1,0)、s(2,0)、... 等起作用的水平引擎 HE(0)66A 的时序信息。在周期 0 处开始的像素数据 S′ 00 是与搜索点 s(0,0) 相关联的 4 乘 4 块的第一像素。在周期 16 处开始的数据 S′ 10 指代搜索点 s(1,0) 的 4 乘 4 块的第一像素，等等。通过将类似的时序关系应用于表 2 所呈现的搜索点，表 2 可轻易地延伸到结束搜索点 s(M-3,0)。

[0115] 表 2

[0116]

水平引擎 HE(0)输出时序表

时间	I/P	HE0						
		R0	R1	R2	R3	P1	P0	水平引擎 O/P
0	S'00	S'00						
1	S'00	S'00						
1	S'01	S'00	S'01					
2	S'01	S'00	S'01					
2	S'02	S'00	S'01	S'02		S'01+S'02		
3	S'02	S'00	S'01	S'02		S'01+S'02		
3	S'03	S'00	S'01	S'02	S'03	S'01+S'02	S'00+S'03	S'00+S'30+S'10+ S'20 (S00)
4	S'03	S'00	S'01	S'02	S'03	S'01+S'02	S'00+S'03	S'00+S'30+S'10+ S'20 (S00)
4	S'10	S'10	S'01	S'02	S'03	S'00-S'03	S'00+S'03	S'00+S'30-S'10-S'20 (S02)
5	S'10	S'10	S'01	S'02	S'03	S'00-S'03	S'00+S'03	S'00+S'30-S'10-S'20 (S02)
5	S'11	S'10	S'11	S'02	S'03	S'00-S'03	S'01-S'02	2S'00-2S'30+S'10-S'20 (S01)
6	S'11	S'10	S'11	S'02	S'03	S'00-S'03	S'01-S'02	2S'00-2S'30+S'10-S'20 (S01)
6	S'12	S'10	S'11	S'12	S'03	S'11+S'12	S'01-S'02	S'00-S'30-2S'10+ 2S'20 (S03)
7	S'12	S'10	S'11	S'12	S'03	S'11+S'12	S'01-S'02	S'00-S'30-2S'10+ 2S'20 (S03)
7	S'13	S'10	S'11	S'12	S'13	S'11+S'12	S'10+S'13	S'10+S'13+S'11+ S'12 (S10)
8	S'13	S'10	S'11	S'12	S'13	S'11+S'12	S'10+S'13	S'10+S'13+S'11+ S'12 (S10)
8	S'20	S'20	S'11	S'12	S'13	S'10-S'13	S'10+S'13	S'10+S'13-S'11-S'12 (S12)
9	S'20	S'20	S'11	S'12	S'13	S'10-S'13	S'10+S'13	S'10+S'13-S'11-S'12 (S12)
9	S'21	S'20	S'21	S'12	S'13	S'10-S'13	S'11-S'12	2S'10-2S'13+S'11-S'12 (S11)
10	S'21	S'20	S'21	S'12	S'13	S'10-S'13	S'11-S'12	2S'10-2S'13+S'11-S'12 (S11)
10	S'22	S'20	S'21	S'22	S'13	S'21+S'22	S'11-S'12	S'10-S'13-2S'11+ 2S'12 (S13)
11	S'22	S'20	S'21	S'22	S'13	S'21+S'22	S'11-S'12	S'10-S'13-2S'11+ 2S'12 (S13)
11	S'23	S'20	S'21	S'22	S'23	S'21+S'22	S'20+S'23	S'20+S'23+S'21+ S'22 (S20)
12	S'23	S'20	S'21	S'22	S'23	S'21+S'22	S'20+S'23	S'20+S'23+S'21+ S'22 (S20)
12	S'30	S'30	S'21	S'22	S'23	S'20-S'23	S'20+S'23	S'20+S'23-S'31-S'32 (S22)
13	S'30	S'30	S'21	S'22	S'23	S'20-S'23	S'20+S'23	S'20+S'23-S'31-S'32 (S22)
13	S'31	S'30	S'31	S'22	S'23	S'20-S'23	S'21-S'22	2S'20-2S'23+S'21-S'22 (S21)
14	S'31	S'30	S'31	S'22	S'23	S'20-S'23	S'21-S'22	2S'20-2S'23+S'21-S'22 (S21)
14	S'32	S'30	S'31	S'32	S'23	S'31+S'32	S'21-S'22	S'20-S'23-2S'21+ 2S'22 (S23)
15	S'32	S'30	S'31	S'32	S'23	S'31+S'32	S'21-S'22	S'20-S'23-2S'21+ 2S'22 (S23)
15	S'33	S'30	S'31	S'32	S'33	S'31+S'32	S'30+S'33	S'30+S'33+S'31+ S'32 (S30)
16	S'33	S'30	S'31	S'32	S'33	S'31+S'32	S'30+S'33	S'30+S'33+S'31+ S'32 (S30)
16	S'10	S'10	S'31	S'32	S'33	S'30-S'33	S'30+S'33	S'30+S'33-S'31-S'32 (S32)
17	S'10	S'10	S'31	S'32	S'33	S'30-S'33	S'30+S'33	S'30+S'33-S'31-S'32 (S32)
17	S'11	S'10	S'11	S'32	S'33	S'30-S'33	S'31-S'32	2S'30-2S'33+S'31-S'32 (S31)
18	S'11	S'10	S'11	S'32	S'33	S'30-S'33	S'31-S'32	2S'30-2S'33+S'31-S'32 (S31)
18	S'12	S'10	S'11	S'12	S'33	S'11+S'12	S'31-S'32	S'30-S'33-2S'31+ 2S'32 (S33)
19	S'12	S'10	S'11	S'12	S'33	S'11+S'12	S'31-S'32	S'30-S'33-2S'31+ 2S'32 (S33)
19	S'13	S'10	S'11	S'12	S'13	S'11+S'12	S'10+S'13	S'10+S'13+S'11+ S'12 (S10)
20	S'13	S'10	S'11	S'12	S'13	S'11+S'12	S'10+S'13	S'10+S'13+S'11+ S'12 (S10)
20	S'20	S'20	S'11	S'12	S'13	S'10-S'13	S'10+S'13	S'10+S'13-S'11-S'12 (S12)
21	S'20	S'20	S'11	S'12	S'13	S'10-S'13	S'10+S'13	S'10+S'13-S'11-S'12 (S12)
21	S'21	S'20	S'21	S'12	S'13	S'10-S'13	S'11-S'12	2S'10-2S'13+S'11-S'12 (S11)
22	S'21	S'20	S'21	S'12	S'13	S'10-S'13	S'11-S'12	2S'10-2S'13+S'11-S'12 (S11)
22	S'22	S'20	S'21	S'22	S'13	S'21+S'22	S'11-S'12	S'10-S'13-2S'11+ 2S'12 (S13)
23	S'22	S'20	S'21	S'22	S'13	S'21+S'22	S'11-S'12	S'10-S'13-2S'11+ 2S'12 (S13)
23	S'23	S'20	S'21	S'22	S'13	S'21+S'22	S'11-S'12	S'10-S'13-2S'11+ 2S'12 (S13)

[0118] 表 3 显示搜索点 s(0,1) 的 4×4 匹配区域的 HE(1) 引擎的 16 像素时序以及从周期十七开始的搜索点 s(1,1) 的像素时序的一部分。表 3 中的数据序列 0、序列 1 和序列 3 是来自水平引擎 HE(0) 的 R1、R2 和 R3 寄存器值的拷贝，其在水平引擎 HE(1) 到 HE(3) 之间

共享。表 3 中列出寄存器值以说明 HE(0) 与 HE(1) 之间的时序关系和数据流。值得注意的是，通过使用“被划分的”像素时钟频率，可通过使用更多寄存器来实现等效的计时能力。举例来说，HE(1) 在一种设计中可具有四个物理寄存器，其中计时能力等效于 2.25 个寄存器，且 HE(0) 可具有七个寄存器，其中计时能力等效于三个寄存器。

[0119] 表 3

[0120] 水平引擎 HE(1) 输出时序表

[0121]

时间	I/P	序列 0	序列 1	序列 2	R0	P1	P0	水平引擎 O/P
0								
1	S'01							
2	S'01	S'01						
2	S'02	S'01						
3	S'02	S'01	S'02					
3	S'03	S'01	S'02					
4	S'03	S'01	S'02	S'03		S'02+S'03		
4	S'04	S'01	S'02	S'03		S'02+S'03		
5	S'04	S'01	S'02	S'03	S'04	S'02+S'03	S'01+S'04	S'01+S'04+S'02+S'03 (S01)

[0122]

5	S'11	S'01	S'02	S'03	S'04	S'02+S'03	S'01+S'04	S'01+S'04+S'02+S'03 (S01)
6	S'11	S'11	S'02	S'03	S'04	S'01-S'04	S'01+S'04	S'01+S'04-S'02-S'03 (S03)
6	S'12	S'11	S'02	S'03	S'04	S'01-S'04	S'01+S'04	S'01+S'04-S'02-S'03 (S03)
7	S'12	S'11	S'12	S'03	S'04	S'01-S'04	S'02-S'03	2S'01-2S'04+S'02-S'03 (S02)
7	S'13	S'11	S'12	S'03	S'04	S'01-S'04	S'02-S'03	2S'01-2S'04+ S'02-S'03 (S02)
8	S'13	S'11	S'12	S'13	S'04	S'12+S'13	S'02-S'03	S'01-S'04-2S'02+ 2S'03 (S04)
8	S'14	S'11	S'12	S'13	S'04	S'12+S'13	S'02-S'03	S'01-S'04-2S'02+ 2S'03 (S04)
9	S'14	S'11	S'12	S'13	S'14	S'12+S'13	S'11+S'14	S'11+S'14+S'12+S'13 (S11)
9	S'21	S'11	S'12	S'13	S'14	S'12+S'13	S'11+S'14	S'11+S'14+S'12+S'13 (S11)
10	S'21	S'21	S'12	S'13	S'14	S'11-S'14	S'11+S'14	S'11+S'14-S'12-S'13 (S13)
10	S'22	S'21	S'12	S'13	S'14	S'11-S'14	S'11+S'14	S'11+S'14-S'12-S'13 (S13)
11	S'22	S'21	S'22	S'13	S'14	S'11-S'14	S'12-S'13	2S'11-2S'14+ S'12-S'13 (S12)
11	S'23	S'21	S'22	S'13	S'14	S'11-S'14	S'12-S'13	2S'11-2S'14+ S'12-S'13 (S12)
12	S'23	S'21	S'22	S'23	S'14	S'22+S'23	S'12-S'13	S'11-S'14-2S'12+ 2S'13 (S14)
12	S'24	S'21	S'22	S'23	S'14	S'22+S'23	S'12-S'13	S'11-S'14-2S'12+ 2S'13 (S14)
13	S'24	S'21	S'22	S'23	S'24	S'22+S'23	S'21+S'24	S'21+S'24+S'22+S'23 (S21)
13	S'31	S'21	S'22	S'23	S'24	S'22+S'23	S'21+S'24	S'21+S'24+S'22+S'23 (S21)
14	S'31	S'31	S'22	S'23	S'24	S'21-S'24	S'21+S'24	S'21+S'24-S'22-S'23 (S23)
14	S'32	S'31	S'22	S'23	S'24	S'21-S'24	S'21+S'24	S'21+S'24-S'22-S'23 (S23)
15	S'32	S'31	S'32	S'23	S'24	S'21-S'24	S'22-S'23	2S'21-2S'24+ S'22-S'23 (S22)
15	S'33	S'31	S'32	S'23	S'24	S'21-S'24	S'22-S'23	2S'21-2S'24+ S'22-S'23 (S22)
16	S'33	S'31	S'32	S'33	S'24	S'32+S'33	S'22-S'23	S'21-S'24-2S'22+ 2S'23 (S24)
16	S'34	S'31	S'32	S'33	S'24	S'32+S'33	S'22-S'23	S'21-S'24-2S'22+ 2S'23 (S24)
17	S'34	S'31	S'32	S'33	S'34	S'32+S'33	S'31+S'34	S'31+S'34+S'32+S'33 (S31)
17	S'11	S'31	S'32	S'33	S'34	S'32+S'33	S'31+S'34	S'31+S'34+S'32+S'33 (S31)
18	S'11	S'11	S'32	S'33	S'34	S'31-S'34	S'31+S'34	S'31+S'34-S'32-S'33 (S33)
18	S'12	S'11	S'32	S'33	S'34	S'31-S'34	S'31+S'34	S'31+S'34-S'32-S'33 (S33)
19	S'12	S'11	S'12	S'33	S'34	S'31-S'34	S'32-S'33	2S'31-2S'34+ S'32-S'33 (S32)
19	S'13	S'11	S'12	S'33	S'34	S'31-S'34	S'32-S'33	2S'31-2S'34+ S'32-S'33 (S32)
20	S'13	S'11	S'12	S'13	S'34	S'12+S'13	S'32-S'33	S'31-S'34-2S'32+ 2S'33 (S34)
20	S'14	S'11	S'12	S'13	S'34	S'12+S'13	S'32-S'33	S'31-S'34-2S'32+ 2S'33 (S34)
21	S'14	S'11	S'12	S'13	S'14	S'12+S'13	S'11+S'14	S'11+S'14+S'12+S'13 (S11)
21	S'21	S'11	S'12	S'13	S'14	S'12+S'13	S'11+S'14	S'11+S'14+S'12+S'13 (S11)
22	S'21	S'21	S'12	S'13	S'14	S'11-S'14	S'11+S'14	S'11+S'14-S'12-S'13 (S13)
22	S'22	S'21	S'12	S'13	S'14	S'11-S'14	S'11+S'14	S'11+S'14-S'12-S'13 (S13)
23	S'22	S'21	S'22	S'13	S'14	S'11-S'14	S'12-S'13	2S'11-2S'14+ S'12-S'13 (S12)
23	S'23	S'21	S'22	S'13	S'14	S'11-S'14	S'12-S'13	2S'11-2S'14+ S'12-S'13 (S12)

[0123] 表 4 展示 VE(0)、VE(1)、HE(0)、HE(1)、HE(2) 和 HE(3) 之间的示范性时序关系。明确地说，表 4 说明垂直引擎 VE(k) 与水平引擎 HE(j) (k = 0、1。j = 0、1、2、3) 之间的输入时序关系以及如何在水平引擎 HE(j) 之间共享经垂直变换的数据。表 4 仅提供搜索点 s(i, j) 的时序信息，其中 i = 0、1、2、...、4 ;j = 0、1、...、3。可轻易地针对所有 s(i, j) 搜索点开发出类似的时序图案，其中 i = 0、1、2、...、M-3 ;j = 0、1、...、N-3。

[0124] 表 4

[0125]

时间	V 引擎输出 在 TM 之后		H 引擎输入序列			
	VE(0)	VE(1)	HE(0)	HE(1)	HE(2)	HE(3)
0	S'0(0,0)		S'0(0,0)			

[0126]

1	S'0(0,1)		S'0(0,1)	S'0(0,1)		
2	S'0(0,2)		S'0(0,2)	S'0(0,2)	S'0(0,2)	
3	S'0(0,3)		S'0(0,3)	S'0(0,3)	S'0(0,3)	S'0(0,3)
4	S'0(1,0)	S'0(0,4)	S'0(1,0)	S'0(0,4)	S'0(0,4)	S'0(0,4)
5	S'0(1,1)	S'0(0,5)	S'0(1,1)	S'0(1,1)	S'0(0,5)	S'0(0,5)
6	S'0(1,2)	S'0(0,6)	S'0(1,2)	S'0(1,2)	S'0(1,2)	S'0(0,6)
7	S'0(1,3)	S'0(0,7)	S'0(1,3)	S'0(1,3)	S'0(1,3)	S'0(1,3)
8	S'0(2,0)	S'0(1,4)	S'0(2,0)	S'0(1,4)	S'0(1,4)	S'0(1,4)
9	S'0(2,1)	S'0(1,5)	S'0(2,1)	S'0(2,1)	S'0(1,5)	S'0(1,5)
10	S'0(2,2)	S'0(1,6)	S'0(2,2)	S'0(2,2)	S'0(2,2)	S'0(1,6)
11	S'0(2,3)	S'0(1,7)	S'0(2,3)	S'0(2,3)	S'0(2,3)	S'0(2,3)
12	S'0(3,0)	S'0(2,4)	S'0(3,0)	S'0(2,4)	S'0(2,4)	S'0(2,4)
13	S'0(3,1)	S'0(2,5)	S'0(3,1)	S'0(3,1)	S'0(2,5)	S'0(2,5)
14	S'0(3,2)	S'0(2,6)	S'0(3,2)	S'0(3,2)	S'0(3,2)	S'0(2,6)
15	S'0(3,3)	S'0(2,7)	S'0(3,3)	S'0(3,3)	S'0(3,3)	S'0(3,3)
16	S'1(1,0)	S'0(3,4)	S'1(1,0)	S'0(3,4)	S'0(3,4)	S'0(3,4)
17	S'1(1,1)	S'0(3,5)	S'1(1,1)	S'1(1,1)	S'0(3,5)	S'0(3,5)
18	S'1(1,2)	S'0(3,6)	S'1(1,2)	S'1(1,2)	S'1(1,2)	S'0(3,6)
19	S'1(1,3)	S'0(3,7)	S'1(1,3)	S'1(1,3)	S'1(1,3)	S'1(1,3)
20	S'1(2,0)	S'1(1,4)	S'1(2,0)	S'1(1,4)	S'1(1,4)	S'1(1,4)
21	S'1(2,1)	S'1(1,5)	S'1(2,1)	S'1(2,1)	S'1(1,5)	S'1(1,5)
22	S'1(2,2)	S'1(1,6)	S'1(2,2)	S'1(2,2)	S'1(2,2)	S'1(1,6)
23	S'1(2,3)	S'1(1,7)	S'1(2,3)	S'1(2,3)	S'1(2,3)	S'1(2,3)
24	S'1(3,0)	S'1(2,4)	S'1(3,0)	S'1(2,4)	S'1(2,4)	S'1(2,4)
25	S'1(3,1)	S'1(2,5)	S'1(3,1)	S'1(3,1)	S'1(2,5)	S'1(2,5)
26	S'1(3,2)	S'1(2,6)	S'1(3,2)	S'1(3,2)	S'1(3,2)	S'1(2,6)
27	S'1(3,3)	S'1(2,7)	S'1(3,3)	S'1(3,3)	S'1(3,3)	S'1(3,3)
28	S'1(4,0)	S'1(3,4)	S'1(4,0)	S'1(3,4)	S'1(3,4)	S'1(3,4)
29	S'1(4,1)	S'1(3,5)	S'1(4,1)	S'1(4,1)	S'1(3,5)	S'1(3,5)
30	S'1(4,2)	S'1(3,6)	S'1(4,2)	S'1(4,2)	S'1(4,2)	S'1(3,6)
31	S'1(4,3)	S'1(3,7)	S'1(4,3)	S'1(4,3)	S'1(4,3)	S'1(4,3)
32	S'2(2,0)	S'1(4,4)	S'2(2,0)	S'1(4,4)	S'1(4,4)	S'1(4,4)
33	S'2(2,1)	S'1(4,5)	S'2(2,1)	S'2(2,1)	S'1(4,5)	S'1(4,5)
34	S'2(2,2)	S'1(4,6)	S'2(2,2)	S'2(2,2)	S'2(2,2)	S'1(4,6)
35	S'2(2,3)	S'1(4,7)	S'2(2,3)	S'2(2,3)	S'2(2,3)	S'2(2,3)
36	S'2(3,0)	S'2(2,4)	S'2(3,0)	S'2(2,4)	S'2(2,4)	S'2(2,4)
37	S'2(3,1)	S'2(2,5)	S'2(3,1)	S'2(3,1)	S'2(2,5)	S'2(2,5)
38	S'2(3,2)	S'2(2,6)	S'2(3,2)	S'2(3,2)	S'2(3,2)	S'2(2,6)
39	S'2(3,3)	S'2(2,7)	S'2(3,3)	S'2(3,3)	S'2(3,3)	S'2(3,3)
40	S'2(4,0)	S'2(3,4)	S'2(4,0)	S'2(3,4)	S'2(3,4)	S'2(3,4)
41	S'2(4,1)	S'2(3,5)	S'2(4,1)	S'2(4,1)	S'2(3,5)	S'2(3,5)
42	S'2(4,2)	S'2(3,6)	S'2(4,2)	S'2(4,2)	S'2(4,2)	S'2(3,6)
43	S'2(4,3)	S'2(3,7)	S'2(4,3)	S'2(4,3)	S'2(4,3)	S'2(4,3)
44	S'2(5,0)	S'2(4,4)	S'2(5,0)	S'2(4,4)	S'2(4,4)	S'2(4,4)
45	S'2(5,1)	S'2(4,5)	S'2(5,1)	S'2(5,1)	S'2(4,5)	S'2(4,5)
46	S'2(5,2)	S'2(4,6)	S'2(5,2)	S'2(5,2)	S'2(5,2)	S'2(4,6)
47	S'2(5,3)	S'2(4,7)	S'2(5,3)	S'2(5,3)	S'2(5,3)	S'2(5,3)
48	S'3(3,0)	S'2(5,4)	S'3(3,0)	S'2(5,4)	S'2(5,4)	S'2(5,4)
49	S'3(3,1)	S'2(5,5)	S'3(3,1)	S'3(3,1)	S'2(5,5)	S'2(5,5)
50	S'3(3,2)	S'2(5,6)	S'3(3,2)	S'3(3,2)	S'3(3,2)	S'2(5,6)
51	S'3(3,3)	S'2(5,7)	S'3(3,3)	S'3(3,3)	S'3(3,3)	S'3(3,3)
52	S'3(4,0)	S'3(3,4)	S'3(4,0)	S'3(3,4)	S'3(3,4)	S'3(3,4)
53	S'3(4,1)	S'3(3,5)	S'3(4,1)	S'3(4,1)	S'3(3,5)	S'3(3,5)
54	S'3(4,2)	S'3(3,6)	S'3(4,2)	S'3(4,2)	S'3(4,2)	S'3(3,6)
55	S'3(4,3)	S'3(3,7)	S'3(4,3)	S'3(4,3)	S'3(4,3)	S'3(4,3)
56	S'3(5,0)	S'3(4,4)	S'3(5,0)	S'3(4,4)	S'3(4,4)	S'3(4,4)
57	S'3(5,1)	S'3(4,5)	S'3(5,1)	S'3(5,1)	S'3(4,5)	S'3(4,5)

[0127]

58	S'3(5,2)	S'3(4,6)	S'3(5,2)	S'3(5,2)	S'3(5,2)	S'3(4,6)
59	S'3(5,3)	S'3(4,7)	S'3(5,3)	S'3(5,3)	S'3(5,3)	S'3(5,3)
60	S'3(6,0)	S'3(5,4)	S'3(6,0)	S'3(5,4)	S'3(5,4)	S'3(5,4)
61	S'3(6,1)	S'3(5,5)	S'3(6,1)	S'3(6,1)	S'3(5,5)	S'3(5,5)
62	S'3(6,2)	S'3(5,6)	S'3(6,2)	S'3(6,2)	S'3(6,2)	S'3(5,6)
63	S'3(6,3)	S'3(5,7)	S'3(6,3)	S'3(6,3)	S'3(6,3)	S'3(6,3)
64	S'4(4,0)	S'3(6,4)	S'4(4,0)	S'3(6,4)	S'3(6,4)	S'3(6,4)
65	S'4(4,1)	S'3(6,5)	S'4(4,1)	S'4(4,1)	S'3(6,5)	S'3(6,5)
66	S'4(4,2)	S'3(6,6)	S'4(4,2)	S'4(4,2)	S'4(4,2)	S'3(6,6)
67	S'4(4,3)	S'3(6,7)	S'4(4,3)	S'4(4,3)	S'4(4,3)	S'4(4,3)
68	S'4(5,0)	S'4(4,4)	S'4(5,0)	S'4(4,4)	S'4(4,4)	S'4(4,4)
69	S'4(5,1)	S'4(4,5)	S'4(5,1)	S'4(5,1)	S'4(4,5)	S'4(4,5)
70	S'4(5,2)	S'4(4,6)	S'4(5,2)	S'4(5,2)	S'4(5,2)	S'4(4,6)
71	S'4(5,3)	S'4(4,7)	S'4(5,3)	S'4(5,3)	S'4(5,3)	S'4(5,3)
72	S'4(6,0)	S'4(5,4)	S'4(6,0)	S'4(5,4)	S'4(5,4)	S'4(5,4)
73	S'4(6,1)	S'4(5,5)	S'4(6,1)	S'4(6,1)	S'4(5,5)	S'4(5,5)
74	S'4(6,2)	S'4(5,6)	S'4(6,2)	S'4(6,2)	S'4(6,2)	S'4(5,6)
75	S'4(6,3)	S'4(5,7)	S'4(6,3)	S'4(6,3)	S'4(6,3)	S'4(6,3)
76	S'4(7,0)	S'4(6,4)	S'4(7,0)	S'4(6,4)	S'4(6,4)	S'4(6,4)
77	S'4(7,1)	S'4(6,5)	S'4(7,1)	S'4(7,1)	S'4(6,5)	S'4(6,5)
78	S'4(7,2)	S'4(6,6)	S'4(7,2)	S'4(7,2)	S'4(7,2)	S'4(6,6)
79	S'4(7,3)	S'4(6,7)	S'4(7,3)	S'4(7,3)	S'4(7,3)	S'4(7,3)

[0128] FTE 变换架构不仅由引擎决定，而且由在垂直引擎与水平引擎之间流动的数据决定。由于所述架构可包含像素速率管线式设计，因此引擎之间的任意偏斜数据流均可使管线停止。为了避免此问题，可在引擎之间使用换位存储器 65A 和 65B 来缓冲并解决数据的时序。

[0129] 本发明的技术允许以管线方式相对于一组视频数据块而执行变换。明确地说，水平引擎与垂直引擎可针对变换的至少一部分同时操作，使得数据被管线输送通过 FTE 60（图 5）。举例来说，如图 5 所示，数据可由垂直引擎 64A 和 64B 处理，且输出可存储在换位存储器 65A 和 65B 中。在处理与第一视频块相关联的所有垂直数据后，垂直引擎 64A 和 64B 可接着处理与第二视频块相关联的垂直数据（在适当时再次使用中间结果）。换位存储器 65A 和 65B 允许垂直引擎 64A 和 64B 的输出在需要时存储，之后此数据可由水平引擎 66A 到 66D 中的一者处理。以此方式，数据被管线输送通过 FTE 60。

[0130] 一般来说，垂直引擎 64A 或 64B 中的一者的数据产生与一个或水平引擎 66A 到 66D 的消耗之间的时序偏斜的最大量界定所需的换位存储器 (TM) 的最小量。为了在转换之间具有最少 TM，通常建议：

[0131] 1. 使馈送器引擎（产生器）与接受器（消耗）引擎之间的数据时序偏斜减到最小，且

[0132] 2. 尽快消耗所产生的馈送器引擎数据。

[0133] 实施 TM 的一种方式是使用随机存取存储器 (RAM)。基于 RAM 的设计的缺点是：

[0134] 1. RAM 的地址大小（等于时序偏斜且小于 16 个时钟周期）较小，且 RAM 的形状在物理设计中可能并不高效。值得注意的是，在两个 4 乘 4 变换之间，最大时序偏斜可能为 15 个时钟周期。

[0135] 2. 在每一 4 乘 4 变换的 16 个时钟周期期间，可在同一时钟周期期间多次从存储器读取和写入到存储器。这意味着 RAM 必须以像素时钟速率的两倍运行，除非使用两端口 RAM。

[0136] 3. 对于较小的基于 RAM 的设计，存储器测试逻辑抵消了其面积优势。

[0137] 在已知这些因素的情况下，当硅上的物理实施方案不遭遇困难时，可考虑在垂直引擎与水平引擎之间使用 RAM 作为换位存储器。

[0138] 另一 TM 设计方法是基于寄存器的设计。可不考虑使用换位存储器还是换位寄存器而使用相同的管线输送技术。图 11 说明可如何在搜索空间中界定输入块。图 12 说明垂直引擎与水平引擎之间的输入和输出数据流。以图 12 所示的方式跟踪索引，以便促进相对于水平引擎输入数据重新排序垂直引擎输入数据和垂直引擎输出数据可改进变换过程的效率。具有 s00、s30、s10、s20、...、s01、...、s23 序列中的像素输入的垂直引擎设计可使用最少的组件。在此情况下，VE 变换输出遵循 S′ 00、S′ 20、S′ 10、S′ 30、...、S′ 33 格式。这可通过跟踪索引到阵列中以记住输入垂直引擎数据到输出垂直引擎数据的重新排序，以纯逻辑方式完成。跟踪索引允许数据存储在一个位置中而无需更新存储器，且消除了不必要的写入操作。所属领域的技术人员还将认识到，还可完成通过将数据重新写入到存储器中来进行重新排序。

[0139] 为了共享输入数据，期望水平引擎输入呈循序格式。在此情况下，可使用管线输送技术，藉此水平引擎与垂直引擎相对于视频数据的管线并行起作用。对于高效的管线式数据处理，只要垂直引擎产生的第一数据可用，水平引擎就可输入所述第一数据。因此，水平输入可呈 S′ 00、S′ 01、S′ 02、S′ 03、...、S′ 33 序列格式。将图 12 中的垂直引擎输出序列（垂直 O/P 序列）与水平引擎输入序列（水平 I/P 序列）进行比较，最大对应差异为 13-4 ＝ 9，其指示在此情况下可能需要九个换位寄存器。所属领域的技术人员将认识到，较大的块大小可能需要不同数目的换位寄存器。所述数目与数据可用于处理的最早时间有关。通过选择合适数目的换位寄存器，可以高效方式支持管线输送，使得额外寄存器被避免。在此情况下，九个换位寄存器可能足以存储所有必要数据达足够时间，以允许此数据尽快在数据处理管线中使用，而无需额外或过量的存储器或寄存器。

[0140] 图 13 描绘可在垂直引擎与水平引擎之间使用的换位寄存器 (TR) 的数目。图 13 的左侧的数字为时标。因此，在此实例中，垂直引擎在时间 T1 输出 S′ 00，其在 T2 输出 S′ 20，...，依此类推。图 13 中的阴影区域表示垂直引擎（馈送器）输出在被水平（接受器）引擎消耗之前，在 TR 中存储多久。此处，水平引擎从时间 T10 开始消耗垂直引擎的输出，且在时间 T25 完成。图 12 的右侧的数字表示每一时标处所需的 TR 的数目，其等于灰色区域的总量。图 13 所估计的总 TR 也是九个。注意，时间 13 处的像素数目被直接馈入水平引擎中，而无需锁存到 TR 中。

[0141] 已描述许多技术和实例。所描述的技术可以硬件、软件、固件或其任意组合实施。如果以软件实施，那么本文中所描述的技术可包含于包含指令的计算机可读媒体中，所述指令当在装置中执行时执行上文所描述的技术中的一者或一者以上。举例来说，所述指令在被执行时可致使视频编码装置对视频数据块执行变换，其中执行变换包括在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的计算。

[0142] 计算机可读媒体可包含例如同步动态随机存取存储器 (SDRAM) 的随机存取存储器 (RAM)、只读存储器 (ROM)、非易失性随机存取存储器 (NVRAM)、电可擦除可编程只读存储器 (EEPROM)、快闪存储器、磁性或光学数据存储媒体等。所述指令可由一个或一个以上处理器或其它机器执行，例如一个或一个以上数字信号处理器 (DSP)、通用微处理器、一个或一

个以上专用集成电路 (ASIC)、一个或一个以上现场可编程逻辑阵列 (FPGA) 或其它等效的集成或离散逻辑电路。在一些实施例中，可在经配置以用于对音频或多媒体信息进行编码和解码的专用软件模块或硬件单元内提供本文中所描述的功能性，或将其并入组合式多媒体编码器 - 解码器 (CODEC) 中。

[0143] 如果以硬件电路实施，那么本发明可针对一种经配置以对视频数据块执行变换的电路，其中在执行变换时，所述电路在第二视频数据块的第二变换中再次使用与第一视频数据块的第一变换相关联的一个或一个以上计算。举例来说，所述电路可包含集成电路或形成芯片组的电路组。所述电路可包含 ASIC、FPGA、各种逻辑电路、集成电路或其组合。

[0144] 这些和其它实施例在所附权利要求书的保护范围内。

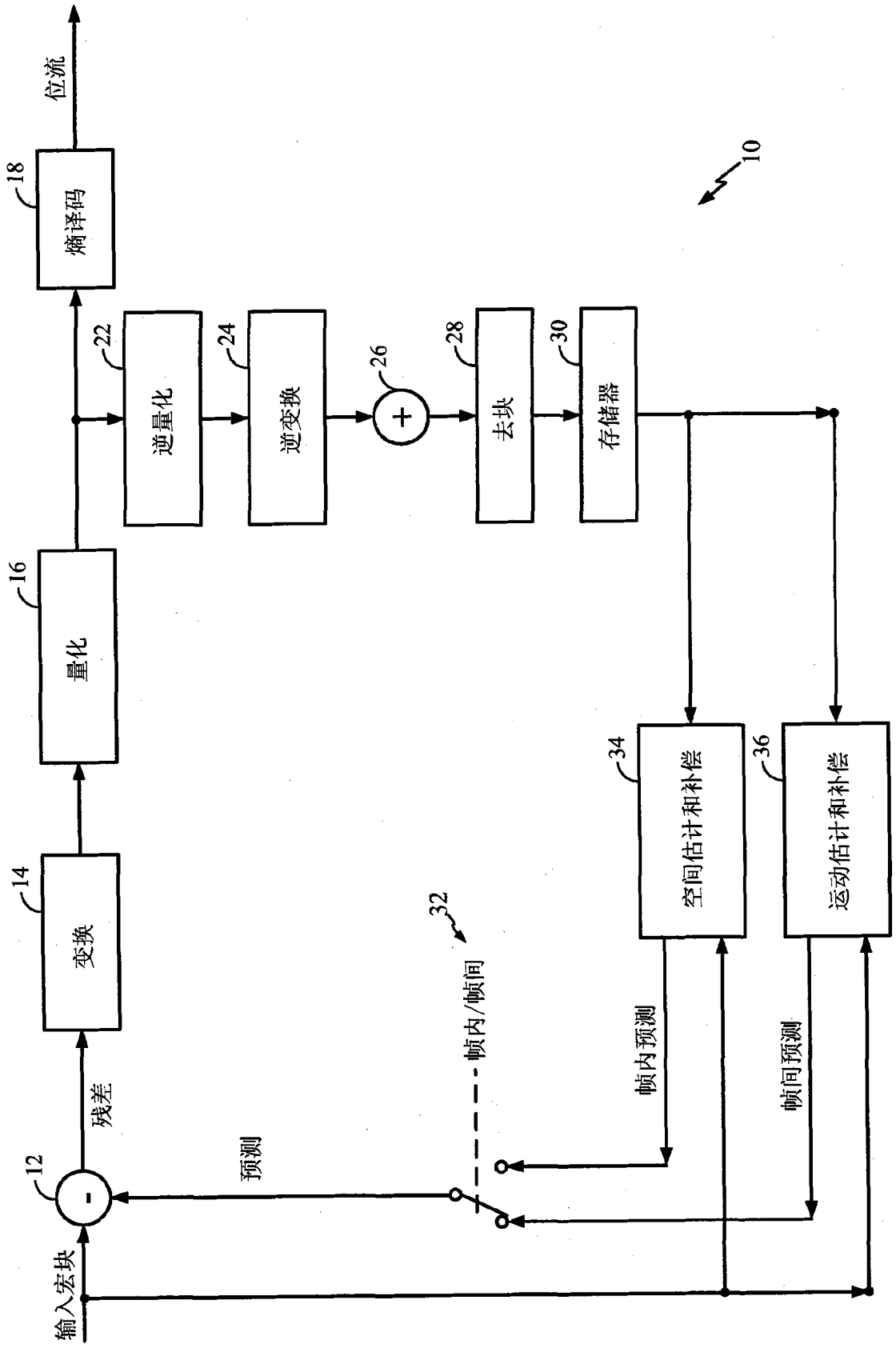

10
位流
18
熵译码
22
逆量化
24
逆变换
26
28
去块
30
存储器
16
量化
14
变换
12
残差
预测
32
帧内/帧间
34
空间估计和补偿
36
运动估计和补偿
帧内预测
帧间预测
输入宏块

图 1

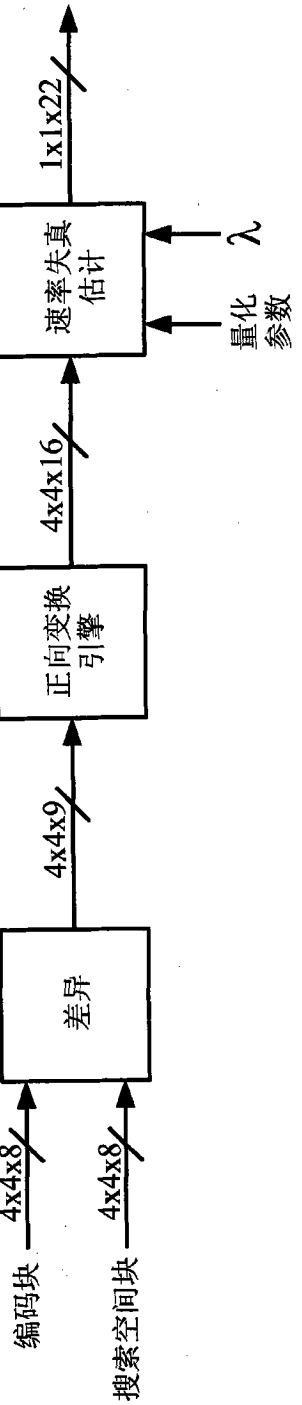

编码块
搜索空间块
4x4x8
4x4x8
42
差异
4x4x9
44
正向变换引擎
4x4x16
46
速率失真估计
1x1x22
λ
量化参数

图 2

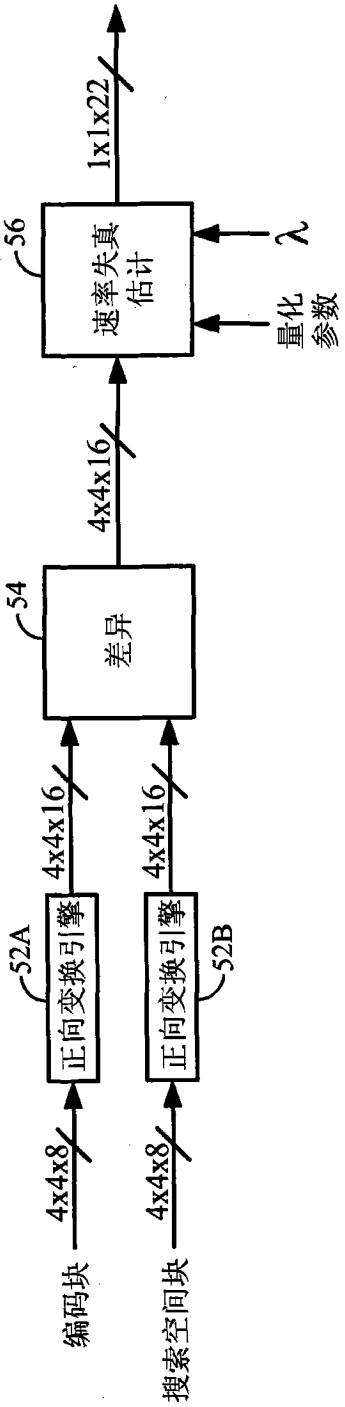

编码块
搜索空间块
4x4x8
4x4x8
52A
正向变换引擎
52B
正向变换引擎
4x4x16
4x4x16
54
差异
4x4x16
56
速率失真估计
1x1x22
λ
量化参数

图 3

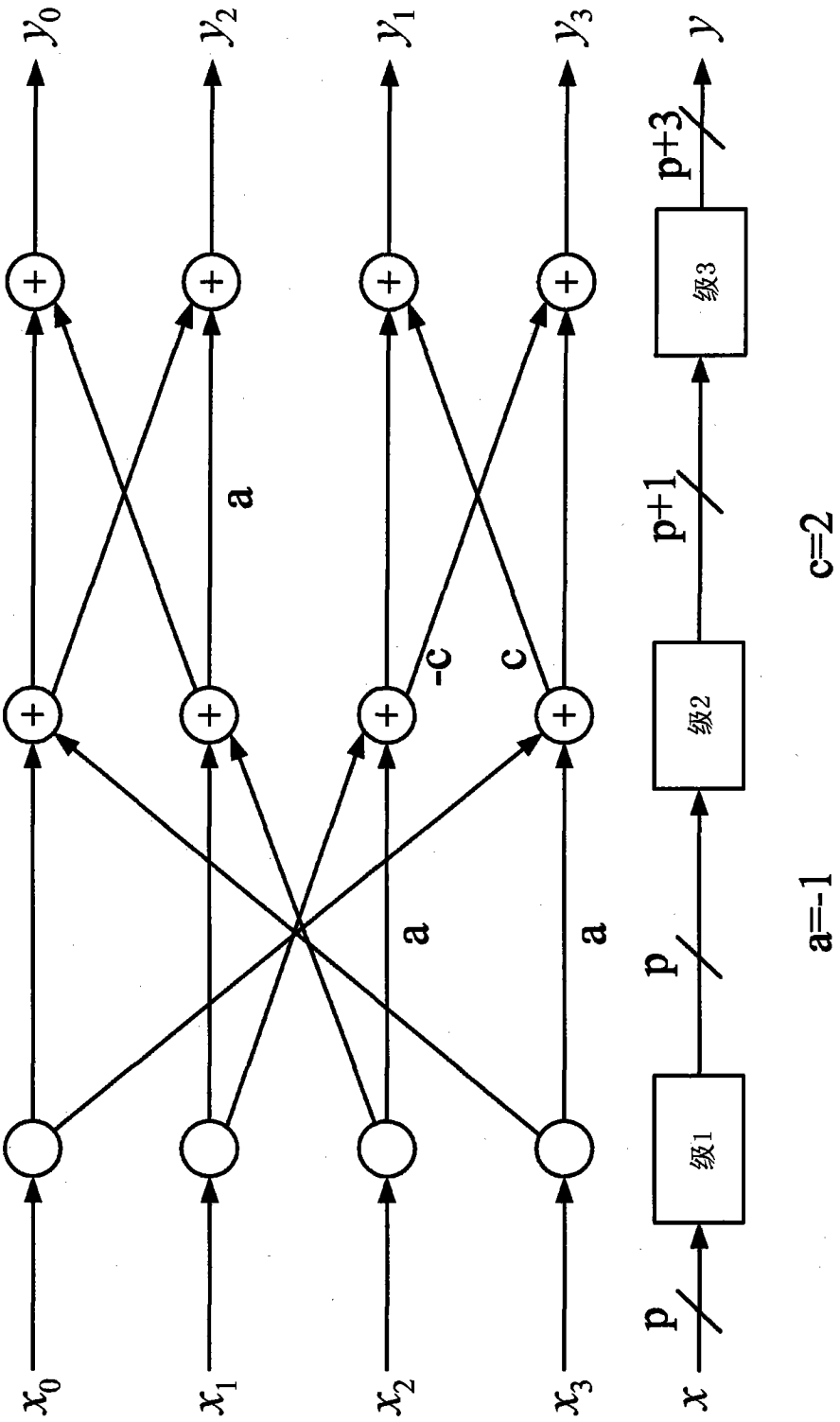


图 4

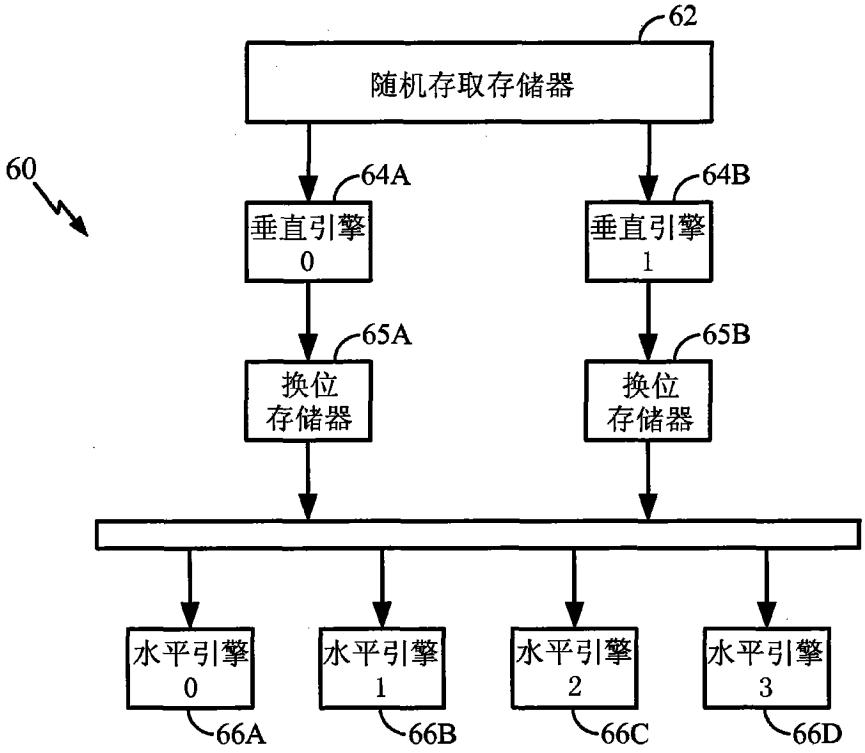

62
随机存取存储器
60
64A
垂直引擎
0
64B
垂直引擎
1
65A
换位
存储器
65B
换位
存储器
水平引擎
0
66A
水平引擎
1
66B
水平引擎
2
66C
水平引擎
3
66D

图 5

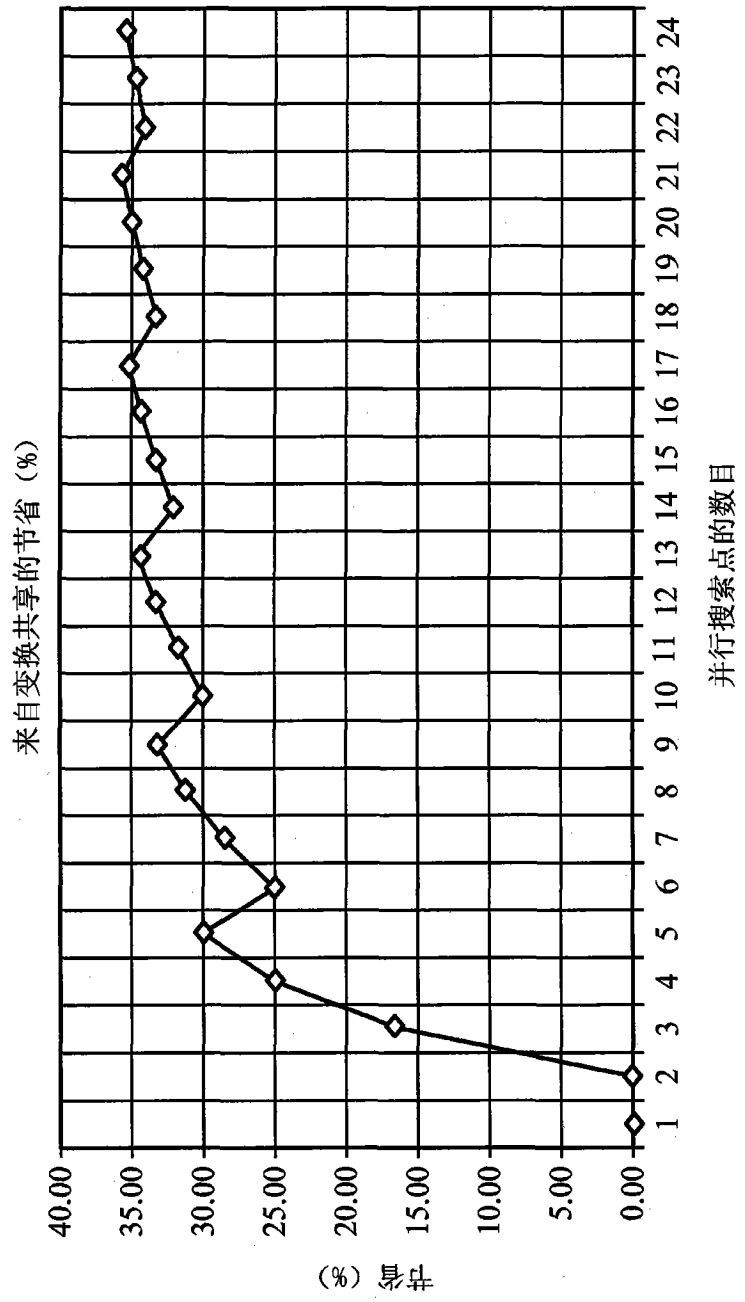

来自变换共享的节省（%）
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
并行搜索点的数目
节省（%）

图 6

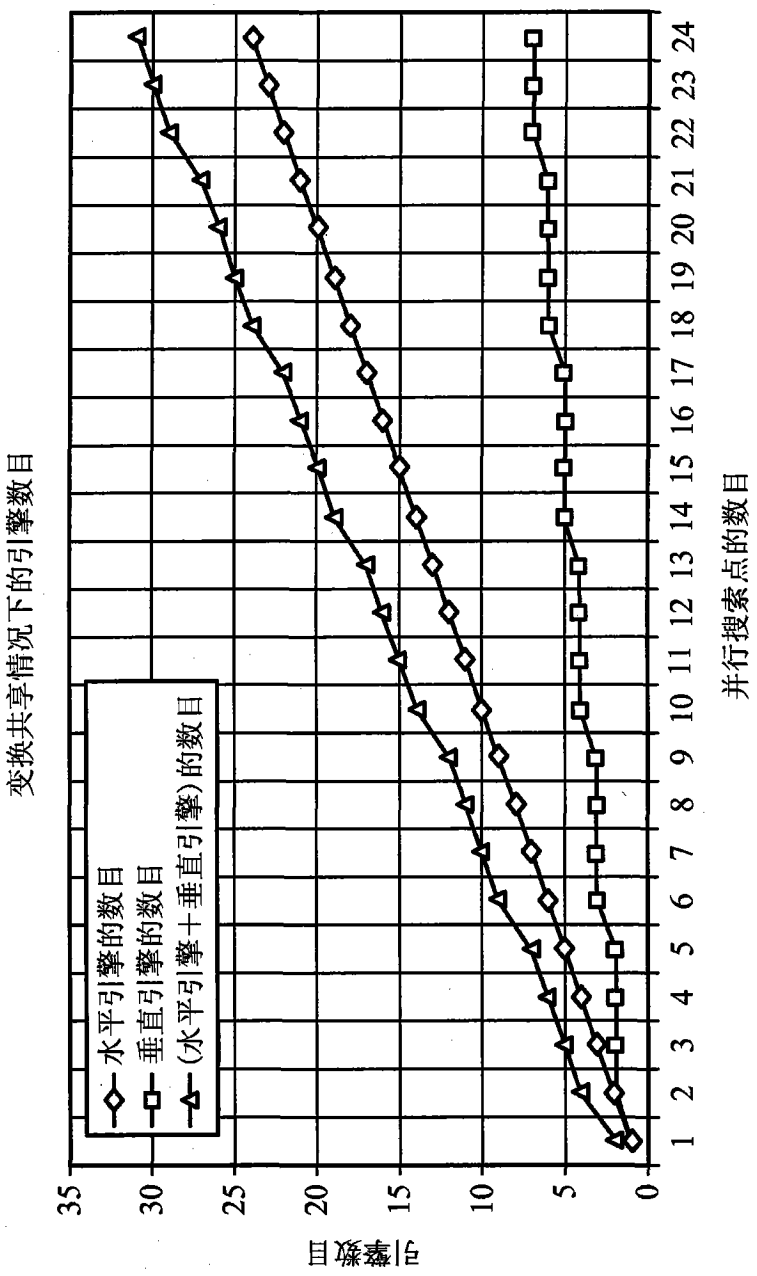

变换共享情况下的引擎数目
水平引擎的数目
垂直引擎的数目
(水平引擎+垂直引擎)的数目
引擎数目
并行搜索点的数目
0
5
10
15
20
25
30
35
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

图 7

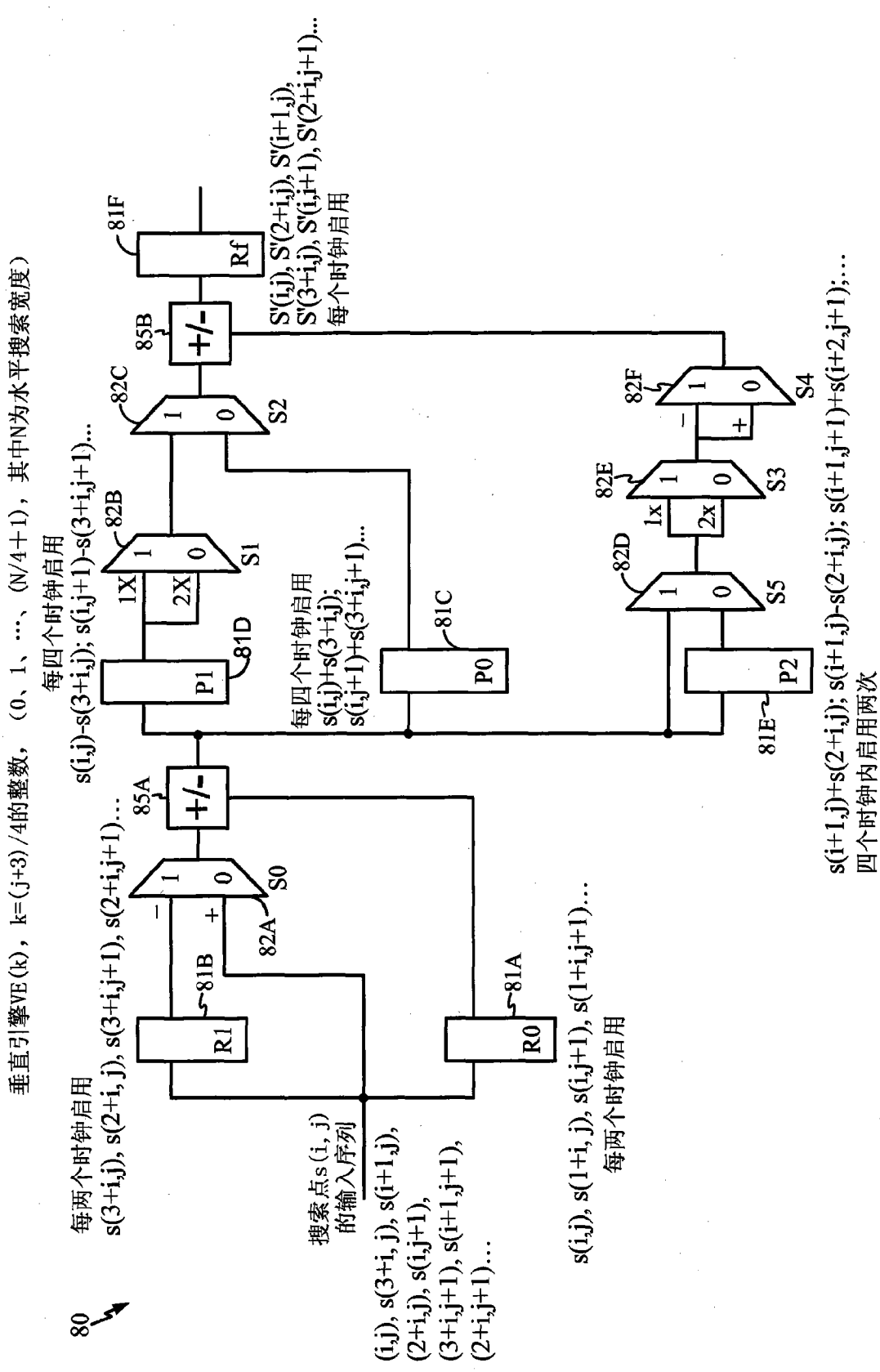

垂直引擎VE(k)，k=(j+3)/4的整数，（0、1、…、(N/4+1)，其中N为水平搜索宽度）
80
每两个时钟启用
s(3+i,j), s(2+i, j), s(3+i,j+1), s(2+i,j+1)…
每四个时钟启用
s(i,j)-s(3+i,j); s(i,j+1)-s(3+i,j+1)…
搜索点s(i, j)的输入序列
(i,j), s(3+i, j), s(i+1,j),
(2+i,j), s(i,j+1),
(3+i,j+1), s(i+1,j+1),
(2+i,j+1)…
s(i,j), s(1+i, j), s(i,j+1), s(1+i,j+1)…
每两个时钟启用
每四个时钟启用
s(i,j)+s(3+i,j);
s(i,j+1)+s(3+i,j+1)…
每四个时钟启用
S'(i,j), S'(2+i,j), S'(i+1,j),
S'(3+i,j), S'(i,j+1), S'(2+i,j+1)…
每个时钟启用
s(i+1,j)+s(2+i,j); s(i+1,j)-s(2+i,j); s(i+1,j+1)+s(i+2,j+1);…
四个时钟内启用两次
R0
R1
P0
P1
P2
Rf
81A
81B
81C
81D
81E
81F
82A
82B
82C
82D
82E
82F
85A
85B
S0
S1
S2
S3
S4
S5
+/-
1X
2X
1x
2x

图 8

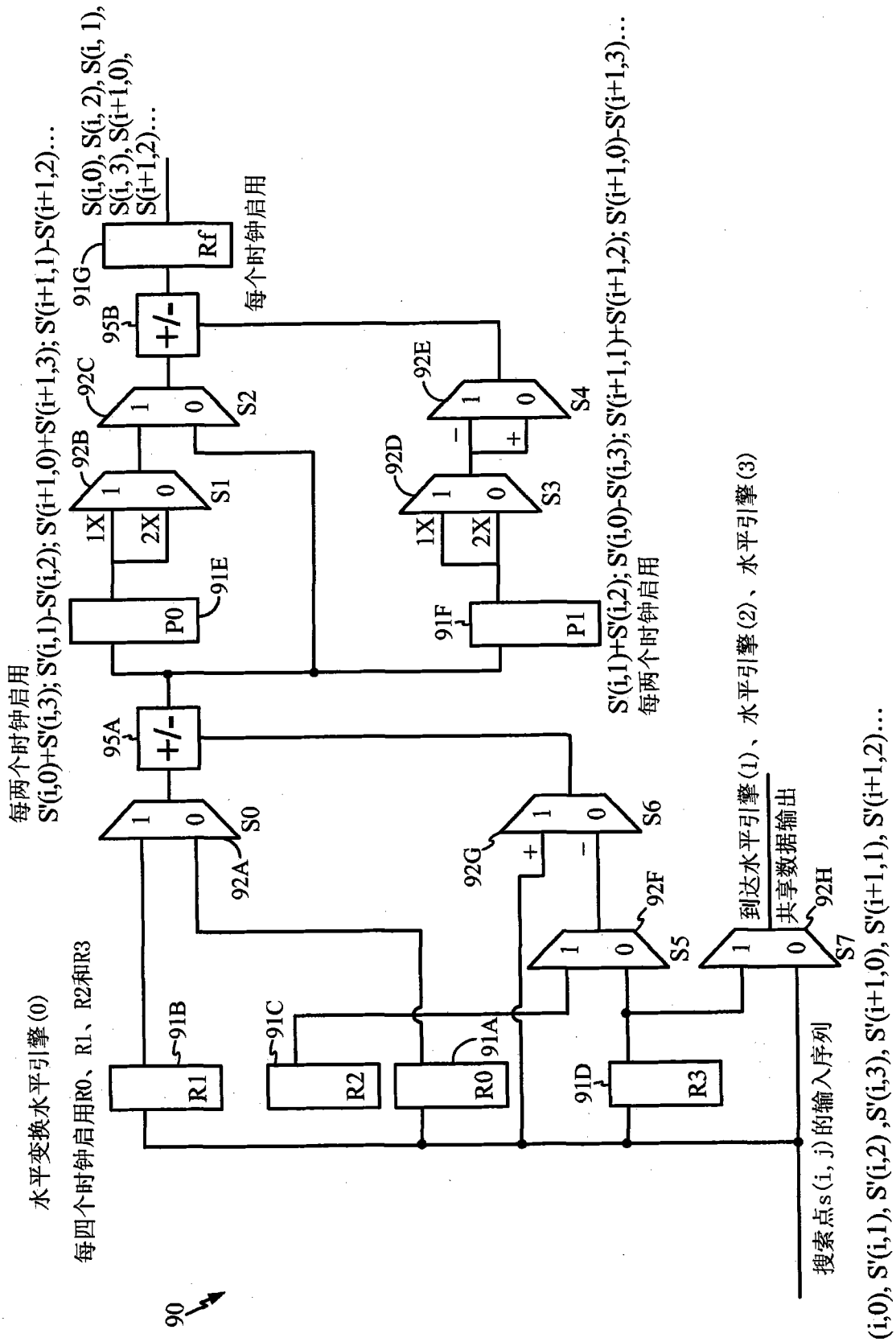

90
水平变换水平引擎(0)
每四个时钟启用R0、R1、R2和R3
每两个时钟启用
S'(i,0)+S'(i,3); S'(i,1)-S'(i,2); S'(i+1,0)+S'(i+1,3); S'(i+1,1)-S'(i+1,2)…
S(i,0), S(i, 2), S(i, 1), S(i, 3), S(i+1,0), S(i+1,2)…
每个时钟启用
每两个时钟启用
S'(i,1)+S'(i,2); S'(i,0)-S'(i,3); S'(i+1,1)+S'(i+1,2); S'(i+1,0)-S'(i+1,3)…
到达水平引擎(1)、水平引擎(2)、水平引擎(3)
共享数据输出
搜索点s(i, j)的输入序列
(i,0), S'(i,1), S'(i,2) ,S'(i,3), S'(i+1,0), S'(i+1,1), S'(i+1,2)…
R0
R1
R2
R3
P0
P1
Rf
S0
S1
S2
S3
S4
S5
S6
S7
91A
91B
91C
91D
91E
91F
91G
92A
92B
92C
92D
92E
92F
92G
92H
95A
95B
+/-
1X
2X

图 9

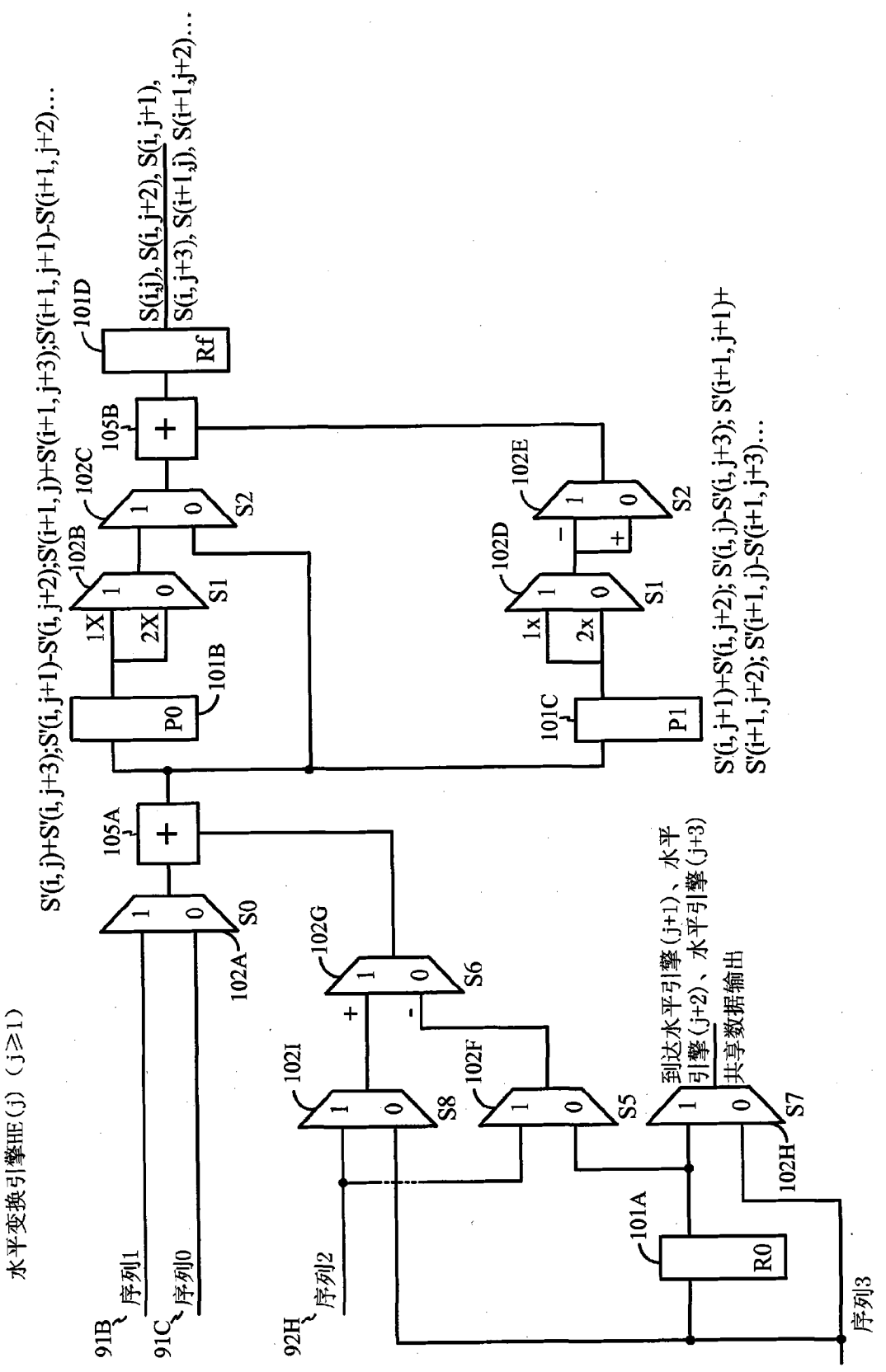

水平变换引擎HE(j)（j≥1）
91B
序列1
91C
序列0
92H
序列2
序列3
S'(i,j)+S'(i,j+3);S'(i,j+1)-S'(i,j+2);S'(i+1,j)+S'(i+1,j+3);S'(i+1,j+1)-S'(i+1,j+2)…
S(i,j), S(i,j+2), S(i,j+1),
S(i,j+3), S(i+1,j), S(i+1,j+2)…
S'(i,j+1)+S'(i,j+2); S'(i,j)-S'(i,j+3); S'(i+1,j+1)+
S'(i+1,j+2); S'(i+1,j)-S'(i+1,j+3)…
到达水平引擎(j+1)、水平引擎(j+2)、水平引擎(j+3)共享数据输出
105A
105B
101A
101B
101C
101D
102A
102B
102C
102D
102E
102F
102G
102H
102I
S0
S1
S2
S5
S6
S7
S8
P0
P1
R0
Rf
1X
2X
1x
2x

图 10

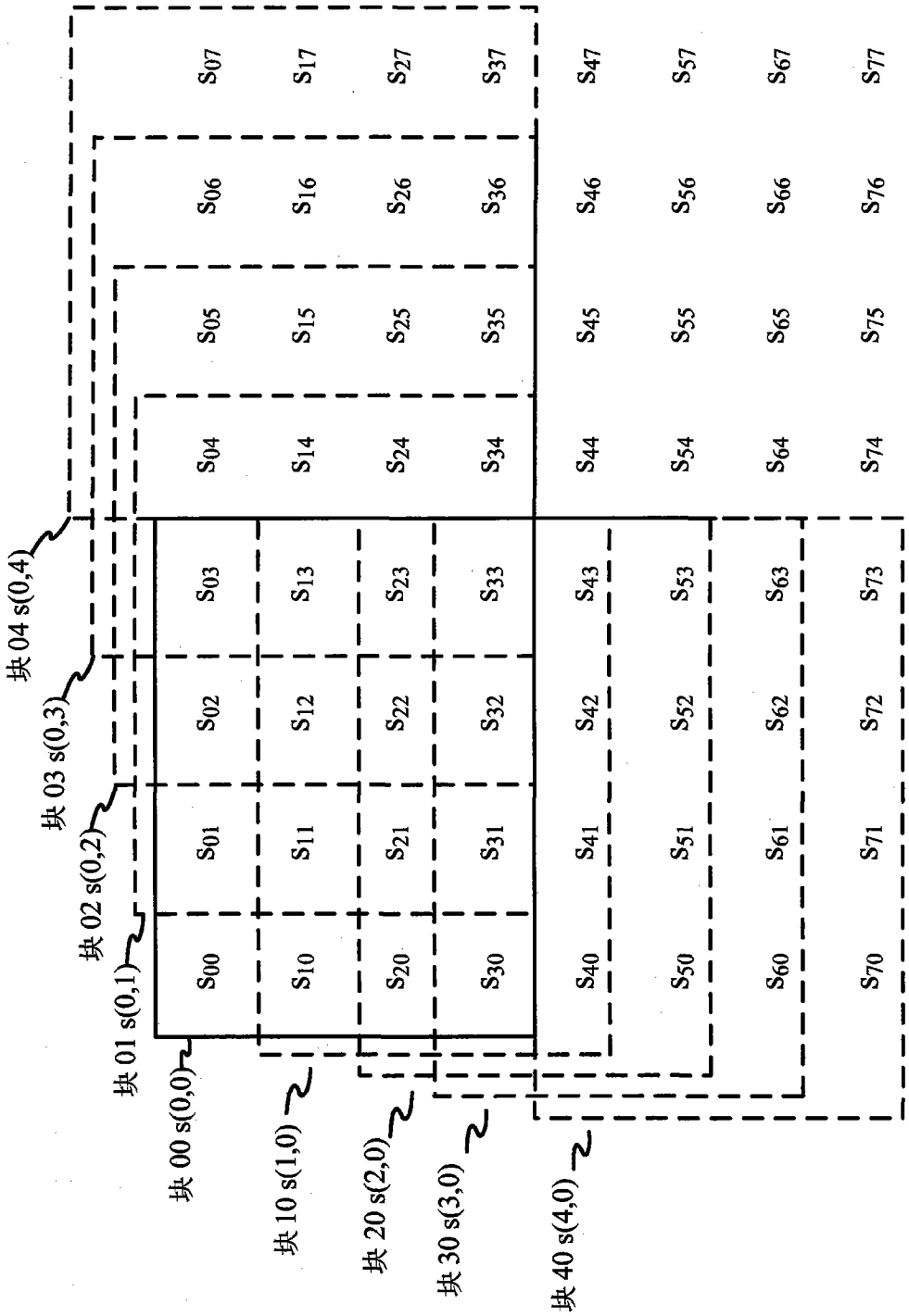

块 04 s(0,4)
块 03 s(0,3)
块 02 s(0,2)
块 01 s(0,1)
块 00 s(0,0)
块 10 s(1,0)
块 20 s(2,0)
块 30 s(3,0)
块 40 s(4,0)
S00 S01 S02 S03 S04 S05 S06 S07
S10 S11 S12 S13 S14 S15 S16 S17
S20 S21 S22 S23 S24 S25 S26 S27
S30 S31 S32 S33 S34 S35 S36 S37
S40 S41 S42 S43 S44 S45 S46 S47
S50 S51 S52 S53 S54 S55 S56 S57
S60 S61 S62 S63 S64 S65 S66 S67
S70 S71 S72 S73 S74 S75 S76 S77

图 11

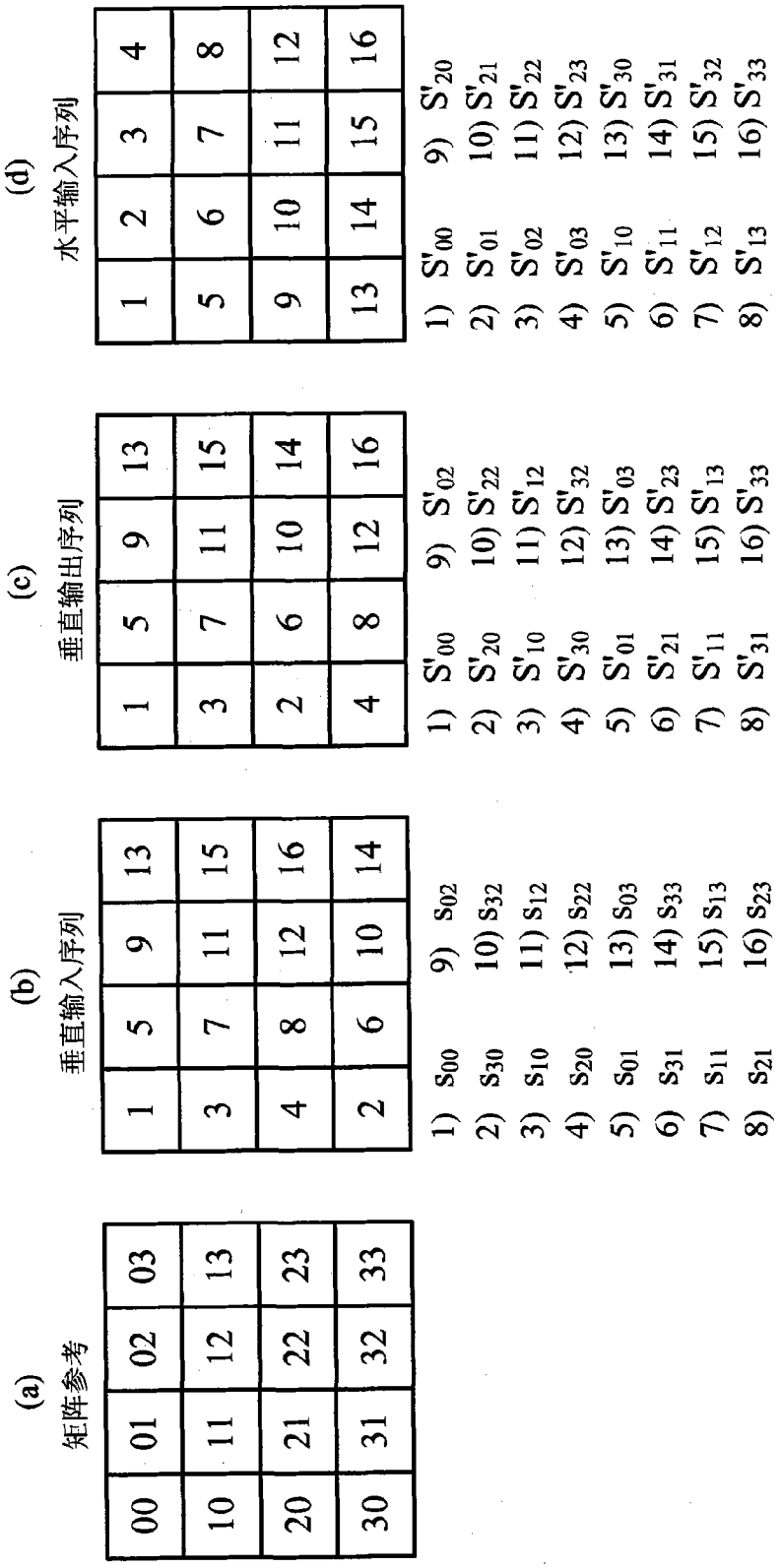

(a)
矩阵参考
00 01 02 03
10 11 12 13
20 21 22 23
30 31 32 33
(b)
垂直输入序列
1 5 9 13
3 7 11 15
4 8 12 16
2 6 10 14
1) s_{00} 9) s_{02}
2) s_{30} 10) s_{32}
3) s_{10} 11) s_{12}
4) s_{20} 12) s_{22}
5) s_{01} 13) s_{03}
6) s_{31} 14) s_{33}
7) s_{11} 15) s_{13}
8) s_{21} 16) s_{23}
(c)
垂直输出序列
1 5 9 13
3 7 11 15
2 6 10 14
4 8 12 16
1) S'_{00} 9) S'_{02}
2) S'_{20} 10) S'_{22}
3) S'_{10} 11) S'_{12}
4) S'_{30} 12) S'_{32}
5) S'_{01} 13) S'_{03}
6) S'_{21} 14) S'_{23}
7) S'_{11} 15) S'_{13}
8) S'_{31} 16) S'_{33}
(d)
水平输入序列
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
1) S'_{00} 9) S'_{20}
2) S'_{01} 10) S'_{21}
3) S'_{02} 11) S'_{22}
4) S'_{03} 12) S'_{23}
5) S'_{10} 13) S'_{30}
6) S'_{11} 14) S'_{31}
7) S'_{12} 15) S'_{32}
8) S'_{13} 16) S'_{33}

图 12

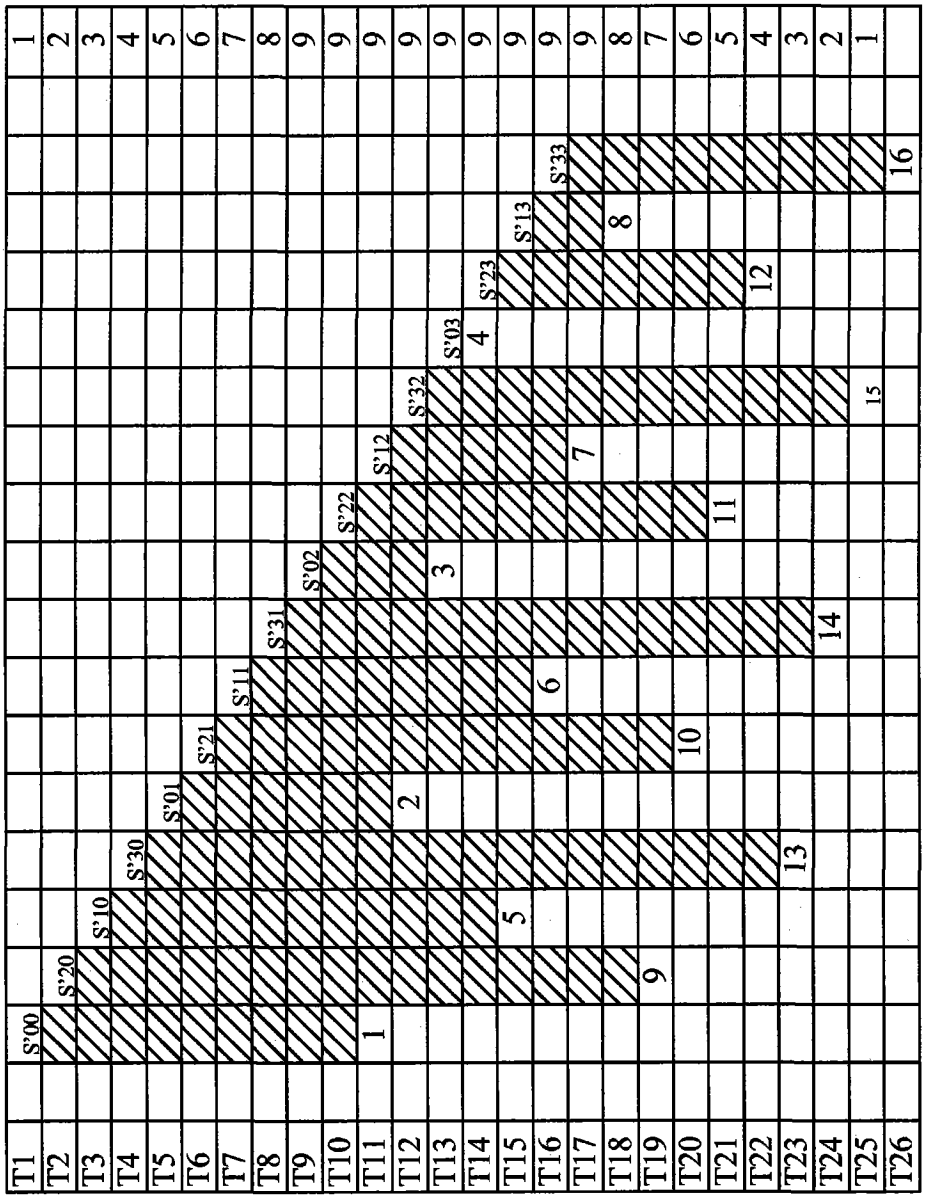

T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17
T18
T19
T20
T21
T22
T23
T24
T25
T26
S'00
S'20
S'10
S'30
S'01
S'21
S'11
S'31
S'02
S'22
S'12
S'32
S'03
S'23
S'13
S'33
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

图 13