[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 17/16

[12] 发明专利申请公开说明书

[21] 申请号 01819979.8

[43] 公开日 2004 年 2 月 25 日

[11] 公开号 CN 1478234A

[22] 申请日 2001.10.5 [21] 申请号 01819979.8

[30] 优先权

[32] 2000.10.6 [33] US [31] 09/680,665

[86] 国际申请 PCT/US01/42537 2001.10.5

[87] 国际公布 WO02/029611 英 2002.4.11

[85] 进入国家阶段日期 2003.6.3

[71] 申请人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 D·叶林 D·雷尼斯 A·朵

[74] 专利代理机构 上海专利商标事务所

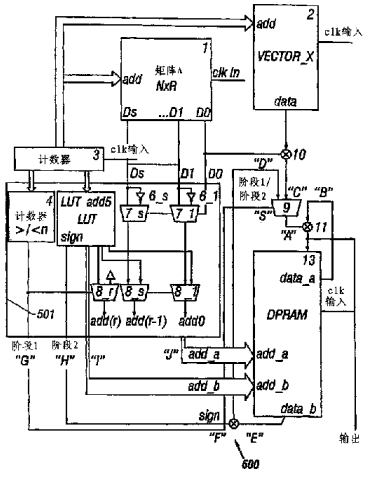
代理人 张政权

权利要求书 3 页 说明书 64 页 附图 5 页

[54] 发明名称 用于有效地执行线性变换的方法和装置

[57] 摘要

一种用于执行线性变换的改进了的方法和装置，具有减少算术运算次数、简化的电路和低功率消耗等特点。 该方法执行 CDMA 应用中出现的复数 U－变换。 该装置允许同时分析扩展因子不同的输入信号。



ISSN1008-4274

1 一种改进线性变换性能效率的方法，该变换用实数或复数或有限字段上的至少一个 n-维输入向量的 r×n 矩阵来表示，其特征在于包括：

在存储器中存储每个被省去的行和每个被选择的行之间的比；

省去所述矩阵的零值行和输入向量的相应标量分量；

标准化所述矩阵的每一列；

从标准化的矩阵中相等列的组中产生经修改的向量；

产生经修改的矩阵；以及

获得输出向量。

2．如权利要求 1 所述的方法，其特征在于还包括把变换矩阵分割为几个子矩阵，并且通过合并从每个子矩阵的乘积产生的输出向量而获得输出向量。

3．如权利要求 2 所述的方法，其特征在于，经修改的矩阵包含所述变换矩阵的行的子集。

4．如权利要求 3 所述的方法，其特征在于还包括把输入向量分割为几个子向量，以便每个子向量对应于一个子矩阵，并且其中输出向量是通过把从每个子矩阵的乘积产生的输出向量相加而获得。

5．如权利要求 1 所述的方法，还包括把经修改的矩阵分割为几个子矩阵，其特征在于，输出向量是由相应的子向量通过把从每个子矩阵的乘积产生的输出向量相加而获得。

6．如权利要求 1 所述的方法，其特征在于还包括通过把所述矩阵的每一列乘以先导元素的倒数而使各列标准化。

7．如权利要求 1 所述的方法，其特征在于，输出向量是矩阵和输入向量的乘积。

8. 如权利要求 1 所述的方法，其特征在于还包括标识标准化矩阵中各相等列的组，并把唯一的位置附加到每个被标识的组。

9. 一种用于执行线性变换的装置，其特征在于包括：

接收输入数据和预定数据的第一和第二输入端；

作用在输入数据和预定数据上的变换电路；

连接于第一存储器的控制和地址发生电路，它为访问所述存储器的单元产生相应的地址，并且用于控制数据接收模式和数据处理模式间的选择，其中数据接收模式中，数据通过所述第一输入端被接收，而在数据处理模式中，经由第一输入端的输入数据的到达被锁定；以及

用于控制装置的操作定时的计数器电路。

10. 如权利要求 9 所述的装置，其特征在于，变换电路用变化数据的相应元素与输入数据的每个元素相乘。

11. 如权利要求 10 所述的装置，其特征在于，变换电路包括存储乘法结果的存储器。

12. 如权利要求 9 所述的装置，其特征在于，变换电路包括加法和累加电路。

13. 如权利要求 9 所述的装置，其特征在于还包括在数据接收模式和数据处理模式间选择的多路复用器电路。

14. 如权利要求 9 所述的装置，其特征在于，控制和地址生成电路包括：

存储预先编程的处理和控制数据的第二存储器；

在数据接收模式和数据处理模式间切换的比较器电路。

15. 如权利要求 14 所述的装置，其特征在于，控制和地址生成电路还包括：

第一组多路复用器，每个都具有至少一个用于接收变换数据的直接输入端，以及变换数据通过相应的反相器被送入的另一个输入端，所述第一组被控制，以所

述变换数据提供的预定值来传送变换数据或者经倒置的变换数据；

第二组多路复用器，每个都具有至少一个连接到从所述第一组多路复用器中选出的相应的多路复用器的输出端的输入端，以及另一个连接到所述第二存储器的输入端，所述第二组由所述比较器电路控制，以便通过把来自所述第一组的每个多路复用器的输出传送到来自所述第二组的相应多路复用器的输出端来把第一地址提供给第一存储器，或者通过传送存储在所述第二存储器中的数据来把第二地址的至少一部分提供给第一存储器；以及

多路复用器，在所述数据处理模式中结合所述第二组多路复用器进行操作，具有一个未连接的输入端和一个连接到所述第二存储器的输入端，并且由所述比较器电路控制，从而提供所述第二地址的剩余部分。

用于有效地执行线性变换的方法和装置

技术领域

本发明涉及数字信号处理领域。它引入了一种用于执行线性变换的改进的方法和装置，它具有减少的算术运算次数、简化的电路和低功耗。

发明背景

在信号或数据处理中经常使用到线性变换。线性变换是从被认为是“输入向量”的 n-维向量和被认为是“输出向量”的 r-维向量产生。在许多应用中，输入向量由需要处理的给定信号的数字采样组成。不过线性变换的应用并不限于任何特定的领域，它对于每个科学技术领域都是重要的并且要求它有高效的执行性能。

从一组 n-维向量集合（实数、复数或从任何标量字段）到一组 r-维向量（在同一标量字段）的广义线性变换可以用 r×n 阶矩阵表示：

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{r1} & & & a_{rn} \end{bmatrix}$$

n-维广义输入列向量书写形式如下：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{P1}$$

从输入向量 x 产生输出 r 维列向量 y 的线性变换是通过矩阵与下列公式给出的向量的乘积获得：

$$y=\begin{bmatrix}y_1\\y_2\\\cdot\\\cdot\\\cdot\\y_r\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & \cdot & \cdot & \cdot & \cdot & a_{1n}\\a_{21} & a_{22} & \cdot & \cdot & \cdot & \cdot & a_{2n}\\\cdot & & & & & & \\\cdot & & & & & & \\\cdot & & & & & & \\a_{r1} & & & & & & a_m\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\cdot\\\cdot\\\cdot\\x_n\end{bmatrix}=$$

$$=\begin{bmatrix}a_{11}x_1+ & a_{12}x_2+ & \cdot & \cdot & \cdot & \cdot & +a_{1n}x_n\\a_{21}x_1+ & a_{22}x_2+ & \cdot & \cdot & \cdot & \cdot & +a_{2n}x_n\\\cdot & \cdot & & & & & \cdot\\\cdot & \cdot & & & & & \cdot\\\cdot & \cdot & & & & & \cdot\\a_{r1}x_1 & a_{r2}x_2+ & \cdot & \cdot & \cdot & \cdot & +a_mx_n\end{bmatrix}$$

线性变换的直接执行需要 r×n 次乘法和 r×(n-1)次加法。

二进制矩阵在本发明的发展和应用中特别重要。二进制双极矩阵在其元素中仅含有±1 的值。今后它们将被称为 U-矩阵。由 U-矩阵表示的线性变换将被称作 U-变换。在上述表示中，如果给定的变换为 U-变换，则为了其直接执行需要 r×(n-1)次加法/减法。另外一种类型的二进制矩阵为其元素中仅包含 0，1 值的矩阵。它们将被称作 0-1 矩阵。由 0-1 矩阵表示的线性变换被称为 0-1 变换。在上述表示中，对于 0-1 变换的直接执行平均将需要 r×(n-1)/2 次加法。

在本文中，术语“二进制变换”将由上述二种形式的变换组成：U-变换和 0-1 变换。为完成术语介绍，应该说明术语 U-向量是指分量为±1 值的向量，同样，分量为 0，1 值的向量被称作 0-1 向量。

二进制变换的处理包括对输入向量的分量的加法和减法。它们在硬件上以电子专用集成电路（ASIC）来实现，需要诸如加法器和减法器这类元件。这类元件的使用是昂贵并且耗能的，它们的构造需要占用珍贵的区域。在许多技术领域均不断地需要一种能够实现二进制变换的硬件而仅耗费少量资源。

U-变换在数字信号处理的各方面均有广泛用途。它包括多种诸如图像处理和无线通信等通信技术。

当前对于直接序列（DS）码多分址（CDMA）扩频通信系统在全球范围内引起日益增加的兴趣。IS-95 标准［TIA/EIA/IS-95-A，“Mobile Station Base Station Compatibility Standard for Dual-Mode Wideband Spread spectrum Cellular System”，1996 年 2 月 27 日］是开发 DS-CDMA 系统的一个例子。

在 CDMA 传输技术中，多用户数据在复合信号中被发出，然后在传输前与伪随机噪声（PN）码相乘，伪随机噪声（PN）码是一种具有随机噪声特性（诸如低度的交叉关系）的 U-序列。扩展特性使传输能抵抗包括多通道噪声、抖动或检测在内

的噪声。也施加了比信道编码长的扰频码。第二代（IS-95-B）和第三代（3G）宽带（WB）CDMA 标准的传输方式要求接收机执行多编码检测。这是一种同时对组合信道（其中各信道原先按照不同信道编码扩展）进行去扩展的任务。这通过其中组成变换矩阵的扩展码通常是 HADAMARD 矩阵的行的 U-变换的应用来完成。这是为了完成低资源消耗的计算任务所期望的有效 U-变换技术的许多实例之一。

存在几种改进特定类型的线性变换的已知机械装置。这里将提及从非常广泛范围内选出的少数相关实例。执行卷积并且在 DSP 中用作滤波器的 Toplitz 变换可以通过采用要求$O(n\cdot\log_2(n))$次加法和乘法操作的典型快速傅里叶变换（FFT）算法有效地完成，其中 n 是域空间的维数。相比较而言，标准、直接的方法需要$O(n^2)$次加法和乘法。详细可见：[James W. Cooley 和 John W. Tukey 著“An algorithm for the machine calculation of complex Fourier series”，*Mathematics of computation* 杂志，19（90）：297-301，1965 年 4 月]

在包括 CDMA 技术的数字信号处理中使用的 U-变换特定形式是可以用 HADAMARD 矩阵表示 Walsh-Harmard 变换。HADAMARD 矩阵可以递归地定义如下：

$$H_1=[1],\quad H_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},$$

且对于每个整数 n、幂为 2 为：$H_{2n}=\begin{bmatrix}H_n & H_n\\ H_n & -H_n\end{bmatrix}$

执行这种变换的一种低复杂度和能量耗费算法是快速 Walsh-Hadmard 变换（FWT），它把用于 n×n HADAMARD 矩阵的加法/减法次数减少到$n\cdot\log_2(n)$次。这在运算中提供了最佳的节省。在这种方法中基本原理相似于 FFT 的基本原理。

U-变换一方面被本发明的另一特征所改进以适应执行 TOPLITZ U-变换的有效方法和装置。这些变换代表与给定 U-序列或复数 U-序列的全部或部分卷积。Toplitz U-变换的无线通信应用包括初始时间同步和使用与实数或复数伪随机（PN）或黄金、U-序列卷积的搜索器。黄金序列是由两个 PN 序列的 Z_2 和组成。

以上本发明的二进制方面通过本发明的另一方面广义化为一种执行线性变换的有效方法和装置，它由具有相对较少数量的不同元素的 r×n 矩阵来表示。本发明的这种先进方面的应用包括复数 U-变换和由带有{0，1，-1}元素的矩阵表示的线性变换。本发明这种广泛优选实施例将被称作广义消元法（GEM）。

宽带 CDMA 和先进的 DSP 未来技术的其它可能的分支将得益于同时分析不同扩展因子下的输入信号。这将为可以同时接收不同信息（诸如传真、一般电话交谈和

互联网数据）而不使网络过载的多编码信道所取代。本发明的另一方面是对于这些类型的线性变换提供有效的方法和装置。它包括本发明带有附加处理器的 U-二进制方面的经修改的版本。为了发现在全局低速率加法中的各分布中应用 U-二进制方法的配置，该附加处理器使用与所述 U-二进制方法所需的加法次数有关的信息。

本发明其它的主要应用包括无线多媒体系统个人卫星移动系统、基于 GPS 卫星的定位系统等等。在移动通信和其它移动、基于计算的系统中，减少用于线性处理的当前损耗是根本的。本发明在移动电话技术中的应用可以延长电池寿命、减少电路并且并且缩短响应时间。

附图说明

通过下面提出的结合附图的详细描述，本发明的特征、性质和优点将变得更加明显，附图中相同的符号具有相同的标识，其中：

图 1 示意地阐明了更新执行使用按照本发明优选实施例的 0-1 二进制变换矩阵的变换的装置所使用的存储器的内容的操作。

图 2 示意地阐明了更新执行使用按照本发明优选实施例的 U-变换矩阵的变换的装置所使用的存储器的内容的操作。

图 3 示意地阐明了更新执行使用按照本发明优选实施例的 Toplitz 变换矩阵的变换的装置所使用的存储器的内容的操作。

图 4 是按照本发明的优选实施例用于执行加法次数减少的线性变换的示例性装置的框图。

图 5 阐明了按照本发明的一个实施例实现 r×n U-矩阵 A 和 n-维向量 X 的乘积的实例。

详细说明

在描述本发明的方法中有两个术语是基本的。第一个是相等向量。两个同样维数的非零向量中的一个是另一个的积，则它们在下文中被称作相等。该术语可被应用于给定矩阵中的两行或两列。第二个术语是给定矩阵中一列的首元素。本发明全文中使用的定义是在所述矩阵的每列中最上方下标的非零成份。该定义依赖于在给定矩阵中一致地使用的预先定义的列下标阶。随着新先导元素的重新定义产生新阶的重新定义。例如某一给定矩阵的先导元素可能是各列中最底下或最上方的非零元素，这取决于为此目的看来最方便的顺序。

0-1-矩阵

本发明的现有优选实施例提供了一种执行 0-1-二进制变换的有效方法。今后它也将被称为“0-1-方法”。下列例子是对 0-1-方法的介绍和其主要思想的概述。

例子：给出 0-1-二进制 4×14 矩阵 A：

$$A=\begin{bmatrix}0&1&1&0&1&1&0&0&1&0&0&0&1&1\\1&0&0&1&0&1&1&0&0&0&1&1&1&0\\0&0&1&1&1&1&0&0&0&0&1&0&1&1\\1&1&1&1&0&1&1&0&1&0&1&1&1&1\end{bmatrix},$$

和 14 维输入向量，

$$x=\begin{bmatrix}x_1\\x_2\\\cdot\\\cdot\\\cdot\\x_{14}\end{bmatrix}$$

假定期望计算 4 维“输出”向量：

$$y=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}=\begin{bmatrix}0&1&1&0&1&1&0&0&1&0&0&0&1&1\\1&0&0&1&0&1&1&0&0&0&1&1&1&0\\0&0&1&1&1&1&0&0&0&0&1&0&1&1\\1&1&1&1&0&1&1&0&1&0&1&1&1&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\cdot\\\cdot\\\cdot\\x_{14}\end{bmatrix}=A\bullet x.$$

按照本发明的优选实施例，首先检查矩阵 A 是否有相等的行。由于没有两行是相等的，则程序进入下一步。接着，输出向量 y 表达为各列之和，其系数为相应的输入向量坐标。因此：

$$y=x_1\begin{bmatrix}0\\1\\0\\1\end{bmatrix}+x_2\begin{bmatrix}1\\0\\0\\1\end{bmatrix}+x_3\begin{bmatrix}1\\0\\1\\1\end{bmatrix}+x_4\begin{bmatrix}0\\1\\1\\1\end{bmatrix}+x_5\begin{bmatrix}1\\0\\1\\0\end{bmatrix}+x_6\begin{bmatrix}1\\1\\1\\1\end{bmatrix}+x_7\begin{bmatrix}0\\1\\0\\1\end{bmatrix}+x_8\begin{bmatrix}0\\0\\0\\0\end{bmatrix}+$$

$$x_9\begin{bmatrix}1\\0\\0\\1\end{bmatrix}+x_{10}\begin{bmatrix}0\\0\\0\\0\end{bmatrix}+x_{11}\begin{bmatrix}0\\1\\1\\1\end{bmatrix}+x_{12}\begin{bmatrix}0\\1\\0\\1\end{bmatrix}+x_{13}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}+x_{14}\begin{bmatrix}1\\0\\1\\1\end{bmatrix}.$$

在此阶段，零列与其系数一起被删除。由关于输入向量的本发明的优选实施

例完成的第一步是收集任何循环非零列的系数并且相加。这样在耗费 6 次加法后可以得到下列简化的向量方程式：

$$y=(x_1+x_7+x_{12})\begin{bmatrix}0\\1\\0\\1\end{bmatrix}+(x_2+x_9)\begin{bmatrix}1\\0\\0\\1\end{bmatrix}+(x_3+x_{14})\begin{bmatrix}1\\0\\1\\1\end{bmatrix}+(x_4+x_{11})\begin{bmatrix}0\\1\\1\\1\end{bmatrix}+x_5\begin{bmatrix}1\\0\\1\\0\end{bmatrix}$$

$$+(x_6+x_{13})\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

此表达式可以用下列定义简化：

$w_1=x_1+x_7+x_{12}$，$w_2=x_2+x_9$，$w_3=x_3+x_{14}$，$w_4=x_4+x_{11}$，$w_5=x_5$，$w_6=x_6+x_{13}$，

这意味着：

$$y=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}=w_1\begin{bmatrix}0\\1\\0\\1\end{bmatrix}+w_2\begin{bmatrix}1\\0\\0\\1\end{bmatrix}+w_3\begin{bmatrix}1\\0\\1\\1\end{bmatrix}+w_4\begin{bmatrix}0\\1\\1\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\\1\\0\end{bmatrix}+w_6\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

这等于最初的问题，但具有较少的列数。不过现在这种简化已经用尽。为了获得更多的收益，该向量方程式分裂为下列相当的两个向量方程式组：

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}=w_1\begin{bmatrix}0\\1\end{bmatrix}+w_2\begin{bmatrix}1\\0\end{bmatrix}+w_3\begin{bmatrix}1\\0\end{bmatrix}+w_4\begin{bmatrix}0\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\end{bmatrix}+w_6\begin{bmatrix}1\\1\end{bmatrix}$$

$$\begin{bmatrix}y_3\\y_4\end{bmatrix}=w_1\begin{bmatrix}0\\1\end{bmatrix}+w_2\begin{bmatrix}0\\1\end{bmatrix}+w_3\begin{bmatrix}1\\1\end{bmatrix}+w_4\begin{bmatrix}1\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\end{bmatrix}+w_6\begin{bmatrix}1\\1\end{bmatrix}$$

这两方程式中每一个将如该优选实施例的第一步那样分别被处理。这样相同非零列的系数将被收集并相加。因此该方法在这一阶段通过附加的 6 次加法的代价得到下列两个向量方程式组：

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}=(w_1+w_4)\begin{bmatrix}0\\1\end{bmatrix}+(w_2+w_3+w_5)\begin{bmatrix}1\\0\end{bmatrix}+w_6\begin{bmatrix}1\\1\end{bmatrix}$$

$$\begin{bmatrix}y_3\\y_4\end{bmatrix}=(w_1+w_2)\begin{bmatrix}0\\1\end{bmatrix}+(w_3+w_4+w_6)\begin{bmatrix}1\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\end{bmatrix}$$

现在很明显只需要 4 次附加的加法来完成输出向量 y 的计算。因此，期望的结果总共用 16 次加法完成。可以容易地检查到，完成该计算过程的常规现有技术方法需要 28 次加法。上面对本发明优选实施例的描述将揭示，当矩阵维数增加时该方法的相对效率也增加。

一般而言，本发明的0-1-二进制方面是n维向量x与r×n维的二进制0-1-矩阵A的乘法的高效的方法和装置。

该矩阵为：

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & \cdot & a_{2n} \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ a_{r1} & & & & & & a_{m} \end{bmatrix}$$

其中对所有 $1 \le i \le r$ ， $1 \le j \le n$， $a_{ij}=0,1$

而输入向量为：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

输入向量的元素可以是实数或复数或属于任何标量字段（例如 Z_2）。其目的在于计算矩阵A和向量x的乘积。结果将表示为：

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_r \end{bmatrix}=\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & \cdot & a_{2n} \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ a_{r1} & & & & & & a_{m} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

将要描述的过程是递归的，并且下列步骤的顺序或其中一部分将在各次迭代中重复。按照本发明的优选实施例，第一步是检查是否有相等的行。当有可能时，该步骤应该作为预先准备在开始处理任何输入向量前完成。如果发现i-行等于j-行，则可以推理 y_i 等于 y_j。因此两相等行之一将被省去。为清楚起见，将永远消除具有较大下标的行。如此，如果 $j > I$，则省去j-行。该最初操作在本发明的最后执行步骤中被修正，其中这一阶段已知的 y_j（由于它等于yi）被插回向量y中适当的位置。相等行的省去继续进行，直到矩阵A中没有相等的两行为止。实际上，该阶段在很多情况中被跳过。它在一旦有相等行合理的可能性时而被执行。它总是在 $\log_2(r) > n$ 时被执行，因为该条件确保相等行的存在。

为避免麻烦的标识，从这一筛选过程得到的经修改的矩阵将具有与原来的矩阵同样的名称 A ，r×n 维数的称呼也将保留。在下一阶段，相加过程将被引入来消除相等的列。该过程将以最小加法代价来减少经修改的矩阵中的列数。首先，矩阵和向量的积 y = A·x 被分解为 A-列与相应的 x 个成分的乘积之和。因此：

$$y=A\bullet x=x_1\begin{bmatrix}a_{11}\\a_{21}\\\cdot\\\cdot\\\cdot\\a_{r1}\end{bmatrix}+x_2\begin{bmatrix}a_{12}\\a_{22}\\\cdot\\\cdot\\\cdot\\a_{r2}\end{bmatrix}+\cdots\cdots\cdots\cdots+x_n\begin{bmatrix}a_{1n}\\a_{2n}\\\cdot\\\cdot\\\cdot\\a_{m}\end{bmatrix}$$

该和的各元素由矩阵 A 的列向量再乘以向量 x 的相应标量系数而组成。该和可以更紧凑地写成：

$$y=A\cdot x=x_1v_1+x_2v_2+\ldots+x_nv_n$$

其中 v_j（对于每个 j）是 A 的第 j 列：

$$v_j=\begin{bmatrix}a_{1j}\\a_{2j}\\\cdot\\\cdot\\\cdot\\a_{rj}\end{bmatrix}$$

按照本发明的优选实施例，在此阶段零列及其系数被省去。其次，全等的非零列将组合在一起并重新安排，其中各明显不同的列成为共同因子，并与其相应标量系数的和相乘。因此，形成的总和包括通过省去重复列而从原始非零列提取的所有不同的非零列 w_1，……，w_m 的子序列。每列 w_j 与系数 t_j 相乘，t_j 是等于该列的所有列的相应原有系数之和。可以清楚地知道 m≤n，且间隙 n-m 是原始序列 v_1，v_2，……，v_n 的重复次数。上述过程可以用下列数学描述公式化：

$$y=A\bullet x=\square_{1\le j\le n}x_jv_j$$

$$=\square_{1\le k\le m}\square_{j\text{以便：}v_j=w_k}x_jv_j$$

$$=\square_{1\le k\le m}\left(\square_{j\text{以便：}v_j=w_k}x_j\right)w_k$$

现在对于所有 1 ≤ k ≤ m 定义：

$$t_k=\square_{j\text{以便：}v_j=w_k}x_j$$

到现在为止，计算任务是计算作为原有 x_1，……，x_n 之和的新系数 t_1，……，t_m。这花费 n-m 次加法。从而乘积 y = A·x 由下式给出：

$$y=A\bullet x=\square_{1\le k\le m}t_kw_k$$

这一计算的重新安排方面和决定所需总和的参与部分可以在输入向量到来之前作为预先准备工作按每矩阵一次完成。总之，设 B 为各列分别为 w_1，……，w_m 的 r×m 矩阵，并设：

$$t=\begin{bmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ \cdot \\ t_m \end{bmatrix}$$

于是 y＝A·x＝B·t。这样上述过程使原来把 r×m 矩阵 A 乘以 n-向量 x 的问题简化为把 r×m 矩阵 B 乘以 m-向量 t 的问题。当 m 较小时获得的收益较大。

下一步是水平分裂上一阶段的 B 矩阵至两个或更多的子矩阵，其中各行保持不断。各子矩阵将具有减少的行数，并且将在下一迭代中增加上述列-采集的收益。矩阵 B 的各行将用 $u_1, u_2 \ldots\ldots, u_r$ 表示。所得乘积将标示为：

$$y=B\cdot t=\begin{bmatrix} u_1 \cdot t \\ u_2 \cdot t \\ \cdot \\ \cdot \\ \cdot \\ u_r \cdot t \end{bmatrix}$$

$u_1 \cdot t$

式中各行与向量 t 乘为标量积。由于这一表达式，计算 y = B·t 的任务相当于计算 r 个标量积：$u_1 \cdot t$，$u_2 \cdot t, \ldots\ldots$，$u_r \cdot t$。

按照本发明优选实施例这可以通过计算矩阵与向量的乘积 $B_1 \cdot t$ 和 $B_2 \cdot t$ 来完成，其中矩阵 B_1 和 B_2 各包含矩阵 B 中行的子集，并且这两个子集互相不相交并且它们的并集包括 B 中所有行的集合。通常，除去第一次迭代，两个子-矩阵具有同样或几乎同样的行数。不过按照 A 矩阵的特性在有些特殊情况可能分割成为两个以上矩阵。当 A 矩阵没有特殊内部顺序时就是这种情况，并且其元素可以认为是任意选择的。本发明优选实施例将意味着第一次行的分割将使所有子矩阵的状态为 log2(列数) > 行数。接着的迭代将分割各行成为几乎相等的两半。认为是任意的矩阵情况基本上是最坏的情况。

另一可被替代地或在上述水平分割之前插入的步骤是垂直分割。按照本发明的优选实施例，上述总和：

$$y=\square_{1\le k\le m} t_k w_k$$

被分割为两部分：

$$y = \square_{1 \le k \le p} t_k w_k + \square_{p+1 \le k \le m} t_k w_k$$

式中 p 是在 1 和 m-1 之间选出。这样对于列集合为 $w_1, \ldots\ldots, w_p$ 的矩阵 B_1，以及对于列集合为 $w_{p+1}, \ldots\ldots, w_m$ 的矩阵 B_2 而言，成立：

$$y = B_1 \cdot t' + B_2 \cdot t''$$

式中相对应的向量为 $t' = (t_1, \ldots\ldots, t_p)$ 以及 $t'' = (t_{p+1}, \ldots, t_m)$。因此通过本发明优选实施例，两个较低维数的乘积 $B_1 \cdot t'$ 和 $B_2 \cdot t''$ 将被分别单独计算，最终的结果为两个 r 向量被加在一起。B 列 $w_1, \ldots, w_m$ 的下标的预先重排可以促进这一步骤的有效性。

垂直分割与其它过程相比不经常使用。其主要用途是在行数实质上大大超过列数的情况，这在 DSP 应用中是少见的。这一步骤的基本缺点是需要对两乘积 $B_1 \cdot t'$ 和 $B_2 \cdot t''$ 求和，这包括附加的 r 次标量和的集合，与上述水平分割没有相同之处。相似于上述水平分割，依照矩阵 A 的特性，垂直分割可以成为多于两个矩阵。

最后各上述步骤应用于重复迭代，从而形成一种递归机制。

其次，说明上述方法促成节约的限度。对于 r×n 的 0-1-矩阵 A，其中 $r < \log2(n)$，用 s*(n,r)表示的最坏情况下加法的次数由下式表达：

$$s^*(n, r) = n + s^*(r).$$

下面的表格给出对于具有少量行数的矩阵的加法次数的限度。

$s^*(2) = -1$	$s^*(n, 2) = n - 1$
$s^*(3) = 2$	$s^*(n, 3) = n + 2$
$s^*(4) = 7$	$s^*(n, 4) = n + 13$
$s^*(5) = 22$	$s^*(n, 5) = n + 49$

对于最坏情况为下列：

$$s^*(r) < 2^r + 2^{r/2+2} - r$$

标准（常规工艺）需要 u(A) – r 次加法，式中 u(A)是在矩阵 A 中的 1'ns 数。平均（当 A 是任意时的期望值）u(A) = (n – 2)·r/2。由于 s*(n,r)/n 当 n 趋向于无穷大而 r 为常数（或 $r \leqslant c \cdot \log2(n)$，式中 $1 > c > 0$）时接近于 1,这方法渐近地为最佳（对于有限的 r 和无穷的 n）。

当 A 为 r×n 0-1-矩阵并且对矩阵 A 的维数 r×n 没有任何假定，则本发明为计算乘积 A·x 所需加法次数限定在 $(1 + \square)n \cdot r/\log2(n)$，式中 $1 > \square > 0$，并且当 r 和 n 均趋向于无穷时，□趋向于 0。如果 r > n 则存在更紧的限度（相对这一情

况），这由于应用垂直分割而造成：

$(1 + \square)n \cdot r/\log_2(r)$, where $1 > \square > 0$,

并且当 r 和 n 均趋向于无穷时，□趋向于 0。

为评估这一过程的效率，应该注意到本发明增加了管理上的操作而减少了加法。不过加法是复杂度的主要部分而本发明的采用是其减少的主要原因。此外，多数管理上的操作在数据向量处理开始以前对于每一矩阵完成一次。这样，所建议的方法实质上节约了功率消耗和电路。当给定的矩阵具有某种程度的稀疏性时上述 0-1-二进制方面可能特别有效。当在计算广义实数矩阵与本发明的实数矩阵中描述的向量的乘积时结合本实施例的应用与分散算术时，这是经常遇到的。

此外，本发明这一方面可应用于代数编码（见：[Shu Lin, Daniel J. Costello, Jr "Error Control Coding Fundamentals and Applications", Prentice Hall, Inc., Englewood Cliffs, N.J., 1983]），其中在编码及解码过程中，Z_2-矩阵与 Z_2-向量（或 Z_{2n}-向量）相乘。这在任何 Z_2-矩阵与 Z_{2n}-向量相乘的情况下均为真。结果，以上 0-1-二进制实施例和下一个 U-二进制实施例在除去特性值为 2 的字段之外的每一向量字段中都很容易互换。在每种特定情况下可以选择较有效的实施例。

U-矩阵

为描述本发明的 U-二进制优选实施例（今后称作"U-方法"）所需初步概念是向量相当的概念。如果两个同样维数的非零向量中一个是另一个的乘积，则它们被称作相当。在本实施例的上下文中，应注意如果两个 U-向量相等或者其中之一是另一个与(-1)的乘积，则它们是相当的。下列优选实施例在大多数细节上大致相似于上述的实施例。主要区别是在下文中消元是对相当（列或行）向量而非对相等向量完成的。这加速了消元速度，因此提高效率。该实施例通过一个说明其概念本质的例子引入：

例：考虑下列 5×13 U-矩阵 A 和 13-维输入向量 x 的乘积。

$$A=\begin{bmatrix} -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \end{bmatrix}$$

而输入向量由下式给出：

$$x=\begin{bmatrix}x_1\\x_2\\\cdot\\\cdot\\\cdot\\x_{13}\end{bmatrix}$$

乘积的结果可写为：

$$y=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\\y_5\end{bmatrix}=A\cdot x$$

过程中的第一步是检查相当行。实际上它是一种在输入向量到达之前的一次操作。扫视发现在所给出的矩阵中有一处相当行，即行 2 相当于行 4，实际上行 4 等于行 2 乘以（-1）之积。由此 $y_4 = -y_2$。因此不需要既计算 y_2 又计算 y_4，故可以消除计算 y_4。这样行 4 从矩阵 A 中消除。作为结果的矩阵 A' 为：

$$A'=\begin{bmatrix}-1&1&-1&-1&1&1&-1&1&1&-1&1&-1&1\\1&-1&-1&1&1&-1&1&-1&-1&1&-1&-1&1\\1&1&-1&1&-1&-1&-1&1&-1&-1&1&1&1\\-1&1&-1&-1&1&1&-1&1&1&-1&1&1&1\end{bmatrix}$$

相应输出向量为：

$$y'=\begin{bmatrix}y_1\\y_2\\y_3\\y_5\end{bmatrix}$$

因此：y'=A'·x。

这是第一次简化。在下一步中 A'·x 分解为 A' 列乘以相应 x 分量之和。因此：

$$\begin{bmatrix}y_1\\y_2\\y_3\\y_5\end{bmatrix}=x_1\begin{bmatrix}-1\\1\\1\\-1\end{bmatrix}+x_2\begin{bmatrix}1\\-1\\1\\1\end{bmatrix}+x_3\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}+x_4\begin{bmatrix}-1\\1\\1\\-1\end{bmatrix}+x_5\begin{bmatrix}1\\1\\-1\\1\end{bmatrix}+x_6\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}+x_7$$

$$\begin{bmatrix}-1\\1\\-1\\-1\end{bmatrix}+x_8\begin{bmatrix}1\\-1\\1\\1\end{bmatrix}+x_9\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}+x_{10}\begin{bmatrix}-1\\1\\-1\\-1\end{bmatrix}+x_{11}\begin{bmatrix}1\\-1\\1\\1\end{bmatrix}+x_{12}\begin{bmatrix}-1\\-1\\1\\1\end{bmatrix}+x_{13}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

其次，对在上述总和中各列进行标准化，使各经修改的向量最上端元素之值为 1。如此，如果最上端元素为（-1），向量及其系数均被乘以（-1）。不过如果向量的最上端元素为 1 则无需改变。上述总和的标准化形式推导如下：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_5 \end{bmatrix} = (-x_1)\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + x_2\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + (-x_3)\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + (-x_4)\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + x_5\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} + x_6\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} +$$

$$(-x_7)\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

$$+ x_8\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + x_9\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + (-x_{10})\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + x_{11}\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + (-x_{12})\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + x_{13}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

在此标准化的总和中各向量的最上端元素为 1 而不同向量的数目减少。下一步为收集并求同一列中各系数之和。这样在花费 8 次加法后可获得下列方程式：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_5 \end{bmatrix} = ((-x_1) + (-x_4) + x_6 + x_9)\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + (x_2 + (-x_7) + x_8 + (-x_{10}) + x_{11})\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + ((-$$

$$x_3) + x_{13})\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + + x_5\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} + (-x_{12})\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

现在定义新系数为：

$w_1 = (-x_1) + (-x_4) + x_6 + x_9$，　$w_2 = x_2 + (-x_7) + x_8 + (-x_{10}) + x_{11}$，

$w_3 = -x_3 + x_{13}$，　$w_4 = x_5$，　$w_5 = -x_{12}$

因此上述方程式可以写成下列形式：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_5 \end{bmatrix} = w_1\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + w_2\begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} + w_3\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + w_4\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} + w_5\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

系数：w_1，w_2，w_3，w_4，w_5 在此阶段对于处理器是已知的。在下一阶段，此向量方程将被水平地分割为下列两方程组：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = w_1\begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_2\begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_3\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_4\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_5\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} y_3 \\ y_5 \end{bmatrix} = w_1\begin{bmatrix} -1 \\ 1 \end{bmatrix} + w_2\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_3\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_4\begin{bmatrix} -1 \\ 1 \end{bmatrix} + w_5\begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

现在将以与第一次迭代相同的方式分别处理这些方程的每一个。由于上面的方程保持前述方程的标准态，因此它无须标准化，从而只需对第二个方程进行标准化。因此产生的方程组为：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = w_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_2 \begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_5 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} y_3 \\ y_5 \end{bmatrix} = (-w_1) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-w_4) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + (-w_5) \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

在下一步，收集同一向量的系数并相加之。这样经过 6 次附加的加法后结果为：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (w_1 + w_2) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + (w_3 + w_4 + w_5) \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} y_3 \\ y_5 \end{bmatrix} = (-w_1 - w_4) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + (w_2 + w_3 + -w_5) \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

很清楚，向量 y'用 4 次附加的加法求得。为获得原有输出向量 y 现在只需要记得 $y_4=-y_2$。这样这一过程总共需要 18 次加法。应注意为计算输出向量的常规现有技术方法需要总共需要 60 次加法。

本发明的 U-二进制优选实施例为一种有效计算 r×m U 矩阵 A 与 n-维输入向量 x 的乘积的方法。它通常比 0-1-优选实施例提供更多的收益，并且有更广泛的应用。

矩阵 A 表示如下：

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{r1} & & & a_{m} \end{bmatrix}$$

式中对于所有 $1 \leqslant i \leqslant r$, $1 \leqslant j \leqslant n$, $a_{ij} = \pm 1$

而输入向量可写为：

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

向量的元素可以是实数或复数输入或属于具有特征值大于 2 的标量字段。目的是计算下列乘积的结果：

$y = A \cdot x$。

第一步是通过消除相当行来修改给定矩阵，这一步可能是初步准备工作的一部分。如发现第 i-行相当于第 j-行，则可以推断 y_i 等于$\pm y_i$。因此本发明第一步的优选实施例决定两相当行中之一将被消除。为清楚起见，原则上将消除具有较大下标的行。这样，如果假定 $j>i$ 则第 j 行将被消除。该初始操作在过程的最后一步中被倒置，其中此阶段已知 y_i 被放回到它在向量 y 中的位置。这一消除过程一直继续到经修改的矩阵中没有相当的两行为止。

只要实际上可能存在相当的两行，这一阶段将被执行。在 U-方法的多乘积应用（本发明的另一方面）中经常发生这种情况。当 $\log_2(r) \geqslant n$ 时总是执行这一阶段，因为这种情况保证相当行的存在。不过还有其他情况也保证相当行的存在，而应该加以考虑。例如，在 Hadamard 矩阵中的子矩阵。为避免不便的表注，所产生的经修改的矩阵将具有同原有矩阵同样的名称 A，而维数 r×n 的名称也将保留。

下一步为消除相当的行。这将以最少加法次数的代价来减少经修改的矩阵的水平维数（即列数）。如在 0-1-方面中一样，所有与这一步有关的管理通常作为数据向量到达前的初始准备工作而每矩阵执行一次。首先，$y = A \cdot x$ 的乘积表达为 A-列乘以相应 x-分量之和。

令 v_j(对于每一 j) 为 A 的第 j 列：

$$v_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdot \\ \cdot \\ \cdot \\ a_{nj} \end{bmatrix}$$

于是：

$$A \cdot x = x_1 v_1 + x_2 v_2 + \cdots\cdots + x_n v_n$$

其次，矩阵 A 的各列向量的最上方元素被检查，并且如果它还不是+1 就被标准化为+1 值。因此，当 $a_{ij} = +1$ 时则不需要标准化，而若 $a_{ij} = -1$ 则进行标准化而相应的第 j-列向量和它的系数 x_j 均被乘以-1。上述总和的新表达式就这样获得，其中各向量的最上方元素等于+1。在该表达式中相当的列永远是相等的。

全等的列然后被组合在一起，如上面的 0-1-实施例一样，并且以各列作为公共因子被重新排列，再乘以相应的标量系数。因此，所得到的总和包括所有明显不同的、经标准化的列向量 $w_1, \ldots\ldots, w_m$（没有重复）的结果，该向量通过筛选重复的标准化列后从标准化列的列向量中提取。各明显不同的标准化列 w_j 被乘以系数 t_j，它是所有等于该列的标准化列中的相应标准化系数的总和。很清楚 $m \leqslant n$，

而其差数 n－m 为标准化列的原始序列：$a_{11}v_1$，$a_{12}v_2$，……，$a_{1n}v_n$ 中的重复次数。

数学上该过程为：

$$A \cdot x = \square_{1 \le j \le n} x_j v_j$$

$$= \square_{1 \le k \le m} \square_{j \text{以便} a_{1j} v_j = w_k} (x_j a_{1j}) a_{1j} v_j$$

$$= \square_{1 \le k \le m} \square_{j \text{以便} a_{1j} v_j = w_k} a_{1j} x_j) w_k$$

对于所有 1≤ k≤ m 定义：$t_k = \square_{j \text{以便} v_j = w_k} a_{1j} x_j$

在发明的这一部分中牵涉的主要计算任务为计算作为经标准化的原始系数之和的新系数 t_1，……，t_m。加法次数的代价为 n-m。乘积 A·x 这样就给出如下：

$$A \cdot x = \square_{1 \le k \le m} t_k w_k$$

列消元过程的收益取决于原有矩阵 A 的结构。这在一般被用于无线通信中信号向量的编码和解码的某些矩阵中很重要，诸如某些形式的 Hadamard 矩阵的子矩阵或循环 PN 矩阵。当矩阵 A 中的行数 r 相对于列数 n 很小时也很显著。特别当 r ≤ log(n)时，在这种情况下 $m-n \geq n-2^{r-1} > 0$。这些观察对于 0-1-矩阵同样也大部分正确。

本发明的该优选实施例的剩余部分，水平和垂直分割、迭代相当于其 0-1-对等部分。

要了解采用本发明所得的节约，首先注意加法次数是复杂度的主要部分。对于 r×n U-矩阵 A，其中 r ≤ log(n)，以 s（n,r）表示的最坏情况下加法的次数为：

$$s(n, r) = n + s(r)$$

其中：

s（1）= 1	s（n,1）= n - 1
s（2）= 0	s（n,2）= n
s（3）= 3	s（n,3）= n + 3
s（4）= 8	s（n,4）= n + 8
s（5）= 19	s（n,5）= n + 19
s（6）= 38	s（n,6）= n + 38
s（7）= 75	s（n,7）= n + 75
s（8）= 144	s（n,8）= n + 144
s（9）= 283	s（n,9）= n + 283

下列界限范围总是有效的：

$$s(r) < 2^{r-1} + 2^{r/2+1} - r$$

现有技术常规方法需要 (n-1)·r 次加法。由于当 n 趋向于无穷而 r 保持常数时 s(n,r)/n 趋向于 1，因此这一方法渐近地接近于最佳。以后将予描述的本发明的更复杂多乘积实施例对于加法次数最坏情况需要一个精确的限度范围。这可以用下列递归公式获得：

对于偶数 r：$s(r) = 2^{r-1} + 2s(r/2)$

对于奇数 r：$s(r) = 2^{r-1} + s((r+1)/2) + s((r-1)/2)$

当 A 为 r×n U-矩阵并且对于 A 矩阵维数 r×n 没有限制时，则在最坏情况下本发明计算 A·x 乘积的加法次数永远被限于下列范围：

(1 + □)n·r/log2(n)，式中 1 > □ > 0，而当 r 和 n 均趋向于无穷时，□趋向于零。如果 r > n，则存在更严格的界限（与本情况比较），这在应用垂直分割时产生：

(1 + □)n·r/log2(r)，式中 1 > □ > 0，

且当 r 和 n 均趋向于无穷时，□趋向于零。

然而，如果在 A 矩阵中存在特定形式的结构，则本发明优选实施例计算 A·x 乘积所需要加法次数大大下降。它的最普遍情况超过本文范围，但可以提到有些例子。在 A 是 r×n Hadmard 或循环伪随机（PN）矩阵的情况，只需要 $n \cdot \log_2(n)$ 次加法和 n 个标量的存储器，在这方面是最佳的。这一说法对于 0-1-矩阵也是正确的。

工业应用的例子

有几种技术可以从上述发明优选实施例的实现中得益。其中一种技术是图像处理，它包含几个包括 U-矩阵与向量相乘的过程。在无线通信 CDMA IS-95 和更先进的第三代宽带 CDMA 中，有几个使用 U-矩阵与向量乘积的过程。这些过程都可以通过本发明的使用用较少的功耗、电路和有时运行时间来完成。

在 IS-95-B 中的多编码正文中，Hadamard-64 中的 8 行包括被乘以向量的 U-矩阵，该向量的元素为信号采样。另一应用是由去扩展器完成的邻近检测。这里也有由 Hadamard 的几行组成的矩阵与数据向量的乘积。另一应用是搜索器。它通过计算互相的向量乘积搜索序列的高度相关性。相关器序列是从 U-序列的伪随机（PN）序列中抽出，而它们的数据向量的标量积是所需要的计算。初始询问是搜索器的一个例子。

广义消元法

上述发明的二进制方面的特征在于下列过程的迭代：消除零值行和相当行（即一行是另一行的标量积）；消除零列和与其关联输入向量的标量成分；组合相当列并相加在一起和在这两种功能用尽后把矩阵水平或垂直分割。按照本发明优选实施例这些步骤的迭代顺序可以在任何标量字段应用于任何矩阵与向量之积。这一更广泛的发明优选实施例将被称作广义消元法（GEM）。

相当列的相加基本上是下列过程的重复。令 $v_1, v_2, \ldots\ldots, v_n$ 变换矩阵 A 的各列而 $x=(x_1, x_2, \ldots\ldots, x_n)$ 为输入向量，其目标是计算 A·x。假定在经过适当的下标重新安排以后各列 $v_{k+1}, \ldots\ldots, v_n$ (对于某些 $2 \leq k < n$)为 k 列的标量积，即对于 $k<j \leq n$: $v_j = z_j v_k$。然后 n-维向量 x 被简化了的 k-维向量所替代：

$$x'=(x_1, x_2, \ldots\ldots, x_{k-1}, x'_k) \quad \text{其中} \quad x'_k = x_k + x_{k+1} \cdot z_{k+1} + \ldots\ldots + x_n \cdot z_n$$

而 r×n 矩阵被精简的 r×k 矩阵 A' 所更换，其列为 $v_1, v_2, \ldots\ldots, v_k$。下式成立：A'·x' = A·x。按照本发明优选实施例，这一过程，如果需要的话包括下标变化，一直重复到没有两列相当为止。实际上这一过程仅仅是在本发明的 U-矩阵实施例中进行的过程标准化的广义化而已。实际上所有上述相当行的精简可以同时进行。当这一阶段完成后应该被视作整个过程的第一次迭代。然后该矩阵被水平分割，而在每一次分割中上述列消元工作以递归方式重复，如在以上实施例中所做。

如以上 U-实施例和下面例子所阐明一种有效消除相当列的方法是通过首先把各列除以最上方元素并由此把相应系数乘以同一最上方元素。结果是在每个经修改的列中的最上方元素为 1。不过有时在应用 GEM 中变换矩阵中一列的最上方分量为零，使除法成为不可能。在这种情况简单解决办法是把个非零列除以最上方的非零元素，并且相应地用同一最上方非零元素乘以相应的系数。结果是每个经修改的列的最上方非零元素为 1。这使得在经修改的矩阵中只有当两列相等时两列才相当，是一种令人满意的结果。

至于本发明优选实施例增进效率的情况，考虑一种其元素在相对小的有限集 S 中的矩阵。实际中，该 S 集合可能是有限字段、字段的乘法组的子组或者在乘法下封闭字段的有限子集，并且在较大的广义性中为字段的任何有限子集。它包括上面讨论的二进制情况，其中 S={0,1}，或者 S={1,-1}及其它非常普通的情况 S={0,1,-1} 或 S={1,-1,j,-j}或 S={0,1,-1,j,-j}。收益随 S 大小而减少，而一般法则保持 GEM 以因子 $\log_{|S|}(n)$ 来减少加法次数，式中 n 为给定变换矩阵中的列

数。当可行时，GEM 应该与下面将描述的复数和通用实施例比较效率。

为阐明这一方法如何工作，以下将说明一种矩阵其元素取自集合：U_1 =def={1,-1,j,-j}，它被称作 U_1-矩阵。 在实际中这种形式的矩阵经常出现。不过应该再次强调，GEM 不限于应用于任何特定形式的矩阵。对于 r×n U_1-矩阵与向量的乘积，加法次数是由 $C = n+4^{r-1}+O(4^{r-1})$ 限定。

例：考虑下列 3×15 U_1-矩阵 A 和 15 维复数输入向量的乘积。矩阵为：

$$A=\begin{bmatrix} 1 & -j & j & -1 & 1 & -j & -1 & j & -1 & j & -1 & 1 & -j & 1 & j \\ -1 & j & -1 & j & 1 & j & 1 & 1 & j & -j & -j & -j & -j & -1 & -j \\ j & 1 & -1 & j & j & j & 1 & -1 & 1 & -1 & -1 & -j & 1 & j & -1 \end{bmatrix}$$

而输入向量为：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_{15} \end{bmatrix}$$

乘积的结果可写为：

$$y=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = A \cdot x$$

所有该例中完成的加法均为复数加法。第一步是检查相当行。没有发现相当行。第二步中乘积 A·x 分解为 A-列乘以相应的 x-分量之和。

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = x_1\begin{bmatrix} 1 \\ -1 \\ j \end{bmatrix} + x_2\begin{bmatrix} -j \\ j \\ 1 \end{bmatrix} + x_3\begin{bmatrix} j \\ -1 \\ -1 \end{bmatrix} + x_4\begin{bmatrix} -1 \\ j \\ j \end{bmatrix} + x_5\begin{bmatrix} 1 \\ 1 \\ j \end{bmatrix} + x_6\begin{bmatrix} -j \\ j \\ j \end{bmatrix} +$$

$$x_7\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} +$$

$$x_8\begin{bmatrix} j \\ 1 \\ -1 \end{bmatrix} + x_9\begin{bmatrix} -1 \\ j \\ 1 \end{bmatrix} + x_{10}\begin{bmatrix} j \\ -j \\ -1 \end{bmatrix} + x_{11}\begin{bmatrix} -1 \\ -j \\ -1 \end{bmatrix} + x_{12}\begin{bmatrix} 1 \\ -j \\ -j \end{bmatrix} + x_{13}\begin{bmatrix} -j \\ -j \\ 1 \end{bmatrix} + x_{14}$$

$$\begin{bmatrix} 1 \\ -1 \\ j \end{bmatrix} + x_{15}\begin{bmatrix} j \\ -j \\ -1 \end{bmatrix}$$

其次，进行上面总和中每列向量的标准化。这通过用每个向量的最上方元素的相反数与其相乘，并从而用相应向量的（同一）最上方元素与每个系数相乘而完成。因此每个经修改的向量的最上方元素为 1。如果向量的最上方元素已经是 1 则不需要修改。可以从上述总和导出下列标准化形式。

$$\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}=x_1\begin{bmatrix}1\\-1\\j\end{bmatrix}+(-j)\cdot x_2\begin{bmatrix}1\\-1\\j\end{bmatrix}+j\cdot x_3\begin{bmatrix}1\\j\\j\end{bmatrix}+(-1)x_4\begin{bmatrix}1\\-j\\-j\end{bmatrix}+x_5\begin{bmatrix}1\\1\\j\end{bmatrix}+(-j)x_6\begin{bmatrix}1\\-1\\-1\end{bmatrix}$$

$$+(-1)x_7\begin{bmatrix}1\\-1\\-1\end{bmatrix}+$$

$$+j\cdot x_8\begin{bmatrix}1\\-j\\j\end{bmatrix}+(-1)x_9\begin{bmatrix}1\\-j\\-1\end{bmatrix}+j\cdot x_{10}\begin{bmatrix}1\\-1\\j\end{bmatrix}+(-1)x_{11}\begin{bmatrix}1\\j\\1\end{bmatrix}+x_{12}\begin{bmatrix}1\\-j\\-j\end{bmatrix}+(-j)x_{13}\begin{bmatrix}1\\1\\j\end{bmatrix}+$$

$$x_{14}\begin{bmatrix}1\\-1\\j\end{bmatrix}+j\cdot x_{15}\begin{bmatrix}1\\-1\\j\end{bmatrix}$$

在该标准化的总和中，各向量最上方元素为 1，当（在通用设置中）列数 n 相对于行数 r 足够大时（要求为：$n>4^{r-1}$），不同向量的数目大大减少。下一步为收集并把相同列的系数相加。这样在付出 7 次加法的代价后可获得下列简化：

$$\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}=(x_1+(-j)\cdot x_2+j\cdot x_{10}+x_{14}+j\cdot x_{15})\begin{bmatrix}1\\-1\\j\end{bmatrix}+j\cdot x_3\begin{bmatrix}1\\j\\j\end{bmatrix}+((-1)x_4+x_{12})\begin{bmatrix}1\\-j\\-j\end{bmatrix}$$

$$+(x_5+(-j)x_{13})\begin{bmatrix}1\\1\\j\end{bmatrix}+\left((-j)x_6+(-1)x_7\begin{bmatrix}1\\-1\\-1\end{bmatrix}\right)+j\cdot x_8\begin{bmatrix}1\\-j\\j\end{bmatrix}+(-1)x_9\begin{bmatrix}1\\-j\\-1\end{bmatrix}+(-1)x_{11}\begin{bmatrix}1\\j\\1\end{bmatrix}$$

新系数现定义如下：

$w_1=x_1+(-j)\cdot x_2+j\cdot x_{10}+x_{14}+j\cdot x_{15}$，　$w_2=j\cdot x_3$

$w_3=(-1)x_4+x_{12}$，　$w_4=x_5+(-j)x_{13}$，　$w_5=(-j)x_6+(-1)x_7$

$w_6=j\cdot x_8$，　$w_7=(-1)x_9$，　$w_8=(-1)x_{11}$

因此方程式可以写成：

$$\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix}=w_1\begin{bmatrix}1\\-1\\j\end{bmatrix}+w_2\begin{bmatrix}1\\j\\j\end{bmatrix}+w_3\begin{bmatrix}1\\-j\\-j\end{bmatrix}+w_4\begin{bmatrix}1\\1\\j\end{bmatrix}+w_5\begin{bmatrix}1\\-1\\-1\end{bmatrix}+w_6\begin{bmatrix}1\\-j\\j\end{bmatrix}+w_7\begin{bmatrix}1\\-j\\-1\end{bmatrix}+w_8\begin{bmatrix}1\\j\\1\end{bmatrix}$$

在此阶段系数 w_1, ... w_8 对于处理器是已知的。其次，该向量方程将被水平分割成为下列两个方程组：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = w_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_2 \begin{bmatrix} 1 \\ j \end{bmatrix} + w_3 \begin{bmatrix} 1 \\ -j \end{bmatrix} + w_4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_5 \begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_6 \begin{bmatrix} 1 \\ -j \end{bmatrix} + w_7 \begin{bmatrix} 1 \\ -j \end{bmatrix} + w_8 \begin{bmatrix} 1 \\ j \end{bmatrix}$$

$$[y_3] = w_1[j] + w_2[j] + w_3[-j] + w_4[j] + w_5[-1] + w_6[j] + w_7[-1] + w_8[1]$$

一般两者均为向量方程。不过在本示范例中第二方程已简化为标量方程。现在的法则是这些方程的每一个将以与第一次迭代相同的方式被分别处理。上面的方程不需标准化，因为它保持前面方程的标准化状态，这样，在任何未简化的应用中只有第二向量方程需要标准化。因此上面的向量方程简化的结果是：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (w_1+w_5) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + (w_2+w_8) \begin{bmatrix} 1 \\ j \end{bmatrix} + (w_3 + w_6 + w_7) \begin{bmatrix} 1 \\ -j \end{bmatrix} + w_4 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

这一步需要 4 次更多的加法。现在计算是以直接的方式完成，两个方程共需花费 13 次附加的加法。这样，在此例子中本发明实施例用去总共 24 次加法。硬性计算将需要 42 次加法。在大规模情况节约将更加显著。

TOPLITZ 矩阵

为说明本发明的该优选实施例将展示一个例子。

例：假定序列长度为 8：u = (1, 1, -1, -1, 1, -1, 1, -1)，并且期望检查数据向量 x = $(x_1, x_2, \ldots . x_{10})$ 的 3 个连续假设。目的是搜索最大相关性。需要计算下列三个和：

$$y_1 = 1 \cdot x_1 + 1 \cdot x_2 + (-1) \cdot x_3 + (-1) \cdot x_4 + 1 \cdot x_5 + (-1) \cdot x_6 + 1 \cdot x_7 + (-1) \cdot x_8$$

$$y_2 = 1 \cdot x_2 + 1 \cdot x_3 + (-1) \cdot x_4 + (-1) \cdot x_5 + 1 \cdot x_6 + (-1) \cdot x_7 + 1 \cdot x_8 + (-1) \cdot x_9$$

$$y_3 = 1 \cdot x_3 + 1 \cdot x_4 + (-1) \cdot x_5 + (-1) \cdot x_6 + 1 \cdot x_7 + (-1) \cdot x_8 + 1 \cdot x_9 + (-1) \cdot x_{10}$$

这可以用下列 TOPLITZ 矩阵积表示：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_{10} \end{bmatrix}$$

$$= x_1\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + x_2\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + x_3\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + x_4\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + x_5\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + x_6\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + x_7\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} +$$

$$x_8\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + x_9\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} + x_{10}\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

下一步为收集补充向量，这样：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \left(x_1\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + x_9\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}\right) \left(x_2\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + x_{10}\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}\right) + x_3\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + x_4\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} +$$

$$x_5\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + x_6\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + x_7\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + x_8\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

在此考虑每个括号内的元素，并利用引理：如果 x, y 为标量且 u, v 为向量，则：

$$xv + yu = \frac{1}{2}(x+y)(v+u) + \frac{1}{2}(x-y)(v-u)$$

因此：

$$x_1\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + x_9\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \frac{1}{2}(x_1+x_9)\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}\right) + \frac{1}{2}(x_1-x_9)\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}\right)$$

$$= \frac{1}{2}(x_1+x_9)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \frac{1}{2}(x_1-x_9)\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

同样：

$$x_2\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + x_{10}\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \frac{1}{2}(x_2+x_{10})\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}(x_2-x_{10})\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

因此在花费 4 次加法后结果是：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \frac{1}{2}(x_1+x_9)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \frac{1}{2}(x_1-x_9)\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}(x_2+x_{10})\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}($$

$$x_2-x_{10})\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + x_3\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + x_4\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + x_5\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + x_6\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + x_7\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + x_8\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} =$$

但现在可以应用上面的 U-二进制方法。这样，下一步是可以使各列的最上方元素等于 1 的标准化过程。

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \frac{1}{2}(x_1+x_9)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \frac{1}{2}(x_1-x_9)\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}(x_2+x_{10})\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}(x_2-$$

$$x_{10})\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + (-x_3)\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + (-x_4)\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + x_5\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + (-x_6)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + x_7\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + (-x_8)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

下一步为收集及求公共列的系数的和。这样，在花费 6 次附加的加法后获得下列等式：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \left(\frac{1}{2}(x_1+x_9) + (-x_6) + x_7 + (-x_8)\right)\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + \left(\frac{1}{2}(x_1-x_9) + \frac{1}{2}(x_2+x_{10}) + (-\right.$$

$$\left.x_4)\right)\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \frac{1}{2}(x_2-x_{10})\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \left((-x_3)+x_5\right)\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

这样，为方便起见定义：

$$w_1=\frac{1}{2}(x_1+x_9)+(-x_6)+x_7+(-x_8), \qquad w_2=\frac{1}{2}(x_1-x_9)+\frac{1}{2}(x_2+x_{10})+(-x_4)$$

$$w_3=\frac{1}{2}(x_2-x_{10}), \qquad w_4=(-x_3)+x_5$$

于是等式可以写成：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = w_1\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} + w_2\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + w_3\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + w_4\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

在下一阶段该向量方程式将被水平分割为下列两个方程组：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = w_1\begin{bmatrix} 1 \\ -1 \end{bmatrix} + w_2\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_3\begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_4\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$y_3 = w_1 + (-w_2) + w_3 + (-w_4)$$

在第一方程式中收集完全相同的列，我们可以用两次附加的加法获得：

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (w_1 + w_4)\begin{bmatrix} 1 \\ -1 \end{bmatrix} + (w_2 + w_3)\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

现在可以用 5 个附加的加法得到期望的输出向量 y。这样，包含本发明的方法需要 17 次加法完成这一计算。硬性计算方法需要 21 次加法。显然在该示范例获得的收益并不大，但在大规模计算中我们将看到所得的收益可以与上述“平方”（非 TOPLITZ）U-方法比拟。

其次为发明的这方面总设置的描述。假定 U 序列为：

$$u_1, u_2, \ldots\ldots\ldots\ldots, u_n$$

而实数或复数标量数据的序列为：

$$x_1, x_2, \ldots\ldots\ldots\ldots, x_{n+m-1}$$

数据序列的元素可以是实数或复数或属于任何特征值大于 2 的标量字段。假定希望计算下列和：

$$y_1 = u_1 \cdot x_1 + u_2 \cdot x_2 + \ldots\ldots\ldots\ldots + u_n \cdot x_n$$

$$y_2 = u_1 \cdot x_2 + u_2 \cdot x_3 + \ldots\ldots\ldots\ldots + u_n \cdot x_{n+1}$$

.

.

.

.

$$y_m = u_1 \cdot x_m + u_2 \cdot x_{m+1} + \ldots\ldots\ldots\ldots + u_n \cdot x_{n+m-1}$$

如果 $m<\log_2(n)$ 则令 r=m，否则如果 $m\geq\log_2(n)$，则本发明优选实施例将通过把 m 个和的组分成 r 个连续和的块而开始，其中 $r<\log_2(n)$。所有块的处理方法完全相同，因此该方法可以在第一块上阐明。最初的 r 个和可以用 TOPLITZ 矩阵与向量的乘积来表示。然后考虑 rx(n+r-1)TOPLITZ 矩阵：

$$A=\begin{bmatrix} u_1 & u_2 & u_3 & u_4 & \cdots & u_n & 0 & \cdots & 0 & 0 \\ 0 & u_1 & u_2 & u_3 & \cdots & u_n & 0 & \cdots & 0 & \cdot \\ 0 & 0 & u_1 & u_2 & \cdots & u_n & 0 & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & u_1 & \cdots & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & & & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & & & & \cdot & & & \cdot & 0 & 0 \\ \cdot & & & & & \cdot & & \cdot & \cdot & 0 \\ 0 & 0 & \cdots & 0 & u_1 & u_2 & \cdots & & & u_n \end{bmatrix}$$

以及(n+r-1)-维向量：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_{n+r-1} \end{bmatrix}$$

于是最初 r 个和由以下向量给出：

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_r \end{bmatrix}$$

其中： $y = A \cdot x$。

本发明的该优选实施例的根本思想在于，重新组织给定的问题以便使上述 U-方法可以应用。令 v_1, v_2, ..., v_{n+r-1} 为矩阵 A 按照其阶数的列。注意所有下列“中间”列向量均为 U-向量：

$$v_r=\begin{bmatrix} u_r \\ u_{r-1} \\ \cdot \\ \cdot \\ \cdot \\ u_1 \end{bmatrix}, \quad v_{r+1}=\begin{bmatrix} u_{r+1} \\ u_r \\ \cdot \\ \cdot \\ \cdot \\ u_2 \end{bmatrix}, \ldots\ldots\ldots\ldots\ldots\ldots, v_n=\begin{bmatrix} u_n \\ u_{n-1} \\ \cdot \\ \cdot \\ \cdot \\ u_{n-r+1} \end{bmatrix}$$

也要注意下列“边”列向量的“配合”对的和及差也是 U-向量。

$$v_1+v_{n+1}=\begin{bmatrix} u_1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}+\begin{bmatrix} 0 \\ u_n \\ \cdot \\ \cdot \\ \cdot \\ u_{n-r+2} \end{bmatrix}=\begin{bmatrix} u_1 \\ u_n \\ u_{n-1} \\ \cdot \\ \cdot \\ u_{n-r+2} \end{bmatrix}, \qquad v_1-v_{n+1}=\begin{bmatrix} u_1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}-\begin{bmatrix} 0 \\ u_n \\ \cdot \\ \cdot \\ \cdot \\ u_{n-r+2} \end{bmatrix}=$$

$$\begin{bmatrix} u_1 \\ -u_n \\ -u_{n-1} \\ \cdot \\ \cdot \\ -u_{n-r+2} \end{bmatrix}$$

$$v_2 + v_{n+2} = \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_n \\ \cdot \\ \cdot \\ u_{n-r+3} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_n \\ \cdot \\ \cdot \\ u_{n-r+3} \end{bmatrix}, \qquad v_2 - v_{n+2} = \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ u_n \\ \cdot \\ \cdot \\ u_{n-r+3} \end{bmatrix} =$$

$$\begin{bmatrix} u_1 \\ u_2 \\ -u_n \\ \cdot \\ \cdot \\ -u_{n-r+3} \end{bmatrix}$$

.....

.....

$$v_{r-1} + v_{n+r-1} = \begin{bmatrix} u_{r-1} \\ u_{r-2} \\ \cdot \\ \cdot \\ u_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ u_n \end{bmatrix} = \begin{bmatrix} u_{r-1} \\ u_{r-2} \\ \cdot \\ \cdot \\ u_1 \cdot \\ u_n \end{bmatrix}, \qquad v_{r-1} - v_{n+r-1} = \begin{bmatrix} u_{r-1} \\ u_{r-2} \\ \cdot \\ \cdot \\ u_1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ u_n \end{bmatrix} =$$

$$\begin{bmatrix} u_{r-1} \\ u_{r-2} \\ \cdot \\ \cdot \\ u_1 \cdot \\ -u_n \end{bmatrix}$$

在完成必要的初步准备工作后可以引入该方法。按照以上重新排列：

$$A \cdot x = \square_{1 \le j \le n+r-1} \; x_j v_j$$

$$= \square_{1 \le j \le r-1} \; x_j v_j + x_{n+j} v_{n+j} + \square_{r \le j \le n} \; x_j v_j$$

然后，本发明采用下列法则，如果 x，y 为标量且 u，v 为向量则：

$$xv + yu = \frac{1}{2}(x+y)(v+u) + \frac{1}{2}(x-y)(v-u).$$ 这样：

$$A \cdot x = \square_{1 \le j \le r-1} \; \frac{1}{2}(x_j + x_{n+j})(v_j + v_{n+j}) + \frac{1}{2}(x_j - x_{n+j})(v_j - v_{n+j}) + \square_{r \le j \le n}$$

$x_j v_j$

这一过程花费 2r-2 次加法而完成的形式实际上是 rx(n+1)的 U矩阵与 n+r-1-向量的乘积。这是因为上面所有向量：

$v_r, v_{r+1}, \ldots\ldots, v_n,\ v_1 + v_{n+1}, \ldots\ldots\ldots\ldots, v_{r-1} + v_{n+r-1},\ v_1 - v_{n+1}, \ldots\ldots\ldots\ldots, v_{r-1} - v_{n+r-1}$

均为 U-向量。这样，计算的其余部分可以用 U-方法完成。所有这一对 U-形式的修改，除去实际加法/减法：$x_j \pm x_{n+j}$，都作为“一次性工作”在输入向量到达前完成。

在最坏情况下的加法次数由以下表达式给出：

$$s_t(n, r) = s(n+r-1, r) + 2r-2 = n+3r-3 + s(r)$$

在通常设置中，其中计算 m 个和并且 $m>\log_2(n)$，在最坏情况中的加法次数是由(1+□)$(n+3\log_2(n)\cdot m/\log_2(n)$所界限，其中 1>□>0，且当 m 和 n 均趋向无穷时，□趋向于零。

按照本发明另一实施例在上述方法中可以综合 GEM 分量。在这种情况下，某些“边”列在耦合边列的第一阶段可以保持不被处理。

（0，1，-1）-矩阵

TOPLITZ 矩阵是矩阵更广义类别的一部分：这些矩阵的元素是 0，1，或 -1。这种矩阵在此将被称作（0，1，-1）-矩阵。关于上述 TOPLITZ-方法的更广泛的概念现在将予以开发。假定 A 是 r×n 维的（0，1，-1）矩阵且 x 是 n 维输入向量，并且期望计算乘积：A·x。该乘积可以表示为 A-列乘以相应 x 分量之和。这样，令 v_j（对于每一 j）表示 A 的第 j 列，则：

$$A \cdot x = x_1 v_1 + x_2 v_2 + \cdots\cdots + x_n v_n$$

某些或可能全部 A 的列都包含 0-元素。为简化标注，可以假定下标被排列（当然是方法上，不派给处理器该任务）以便最初 k 列 $v_1, v_2, \ldots\ldots\ldots\ldots, v_k$ 的每一列都含有零元素并且所有剩余的 n-k 列（如果有）$v_{k+1}, v_{k+2}, \ldots\ldots\ldots\ldots, v_n$ 均为 U-向量（即没有 0 元素）。很清楚各“混合”向量：$v_1, v_2, \ldots\ldots\ldots\ldots, v_k$ 为 2 个 U-向量的平均值。因此对于每个 $1 \leqslant j \leqslant k$ 都存在两个 U-向量 u_j, w_j，使：

$v_j = 1/2(u_j + w_j)$。作为每矩阵一次的初步准备工作，本发明的优选实施例求出向量 $u_1, w_1, \ldots\ldots\ldots\ldots, u_k, w_k$。因此，从上可得：

$$A \cdot x = \frac{1}{2} x_1(u_1 + w_1) + \frac{1}{2} x_2(u_2 + w_2) + \ldots\ldots\ldots\ldots + \frac{1}{2} x_k(u_k + w_k) + x_{k+1} v_{k+1} + \ldots\ldots\ldots\ldots + x_n v_n$$

然而，在打开括号以后，我们得到 r×(n+k) U 矩阵和(n+k)-向量的乘积的表达式。按照本发明优选实施例，这一任务将由上述发明的 U-方面所完成。上述发明的 TOPLITZ 方面是这一较广泛方面的特例。

工业应用实例

搜索器通常由 TOPLITZ 矩阵和数据向量的乘积表示。这在 CDMA 及宽带 CDMA 中也是如此。

实数矩阵

按照本发明另一优选实施例，计算操作的次数对于任何实数线性变换可以通过采用分布算术而被减少。在前面的文中它被称为“实数矩阵方法”。线性变换由元素为实数、不一定是整数、具有固定数目的二进制位数的矩阵来表示。矩阵可以被分解为具有二进位系数的二进制矩阵之和。在下一阶段将应用发明的二进制实施例。为引入这一方法先考虑下列例子：

例： 考虑以下 3×8 U-矩阵 A，它具有（为本例子的简单起见）整数元素并且写成十进制：

$$A=\begin{bmatrix}2 & 1 & 5 & 4 & 3 & 0 & 4 & 5\\ 5 & 0 & 4 & 1 & 4 & 7 & 1 & 2\\ 2 & 7 & 3 & 1 & 4 & 5 & 0 & 3\end{bmatrix}$$

8-维输入向量表示为：

$$x=\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\\ x_7\\ x_8\end{bmatrix}$$

期望计算 3-维输出向量：

$$y=\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix}=A\cdot x.$$

由于正常情况在二进制基础上工作，因此矩阵 A 将以此为基础被表示为：

$$A=\begin{bmatrix}010 & 001 & 101 & 100 & 011 & 000 & 100 & 101\\ 101 & 000 & 100 & 001 & 100 & 111 & 001 & 010\\ 010 & 111 & 011 & 001 & 100 & 101 & 000 & 011\end{bmatrix}$$

该表达式可以看到有可能使用分布算术表达矩阵为三个二进制矩阵的和：

$A = \square\cdot A[0] + \square'\cdot A[1] + \square^{\square}\cdot A[2]$

式中：

$$A[0]=\begin{bmatrix}0&1&1&0&1&0&0&1\\1&0&0&1&0&1&1&0\\0&1&1&1&0&1&0&1\end{bmatrix}$$

$$A[1]=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&1\\1&1&1&0&0&0&0&1\end{bmatrix}$$

$$A[2]=\begin{bmatrix}0&0&1&1&0&0&1&1\\1&0&1&0&1&1&0&0\\0&1&0&0&1&1&0&0\end{bmatrix}$$

由本发明实施例采取的第一步是建立二进制矩阵 A[0]，A[1]，A[2]，包括按照有效位 A 元素的位元。在考虑中的优选实施例将为等式的结果：

$$A\bullet x = \square\bullet A[0]\bullet x + \square'\bullet A[1]\bullet x + \square^{\square}\bullet A[2]\bullet x$$

下一步为组成由 A[0]，A[1]，A[2]的水平链形成的 24×3 二进制矩阵 A*。

$$A^*=$$

$$=\begin{bmatrix}0&1&1&0&1&0&0&1\\1&0&0&1&0&1&1&0\\0&1&1&1&0&1&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&1\\1&1&1&0&0&0&0&1\end{bmatrix}$$

$$\begin{bmatrix}0&0&1&1&0&0&1&1\\1&0&1&0&1&1&0&0\\0&1&0&0&1&1&0&0\end{bmatrix}$$

同时也要形成 24-维列向量 x*，它包括向量 x 的 3 个复制件，每个都具有从上面的和中导出的相应二进制权重。这在输入向量的到达开始前完成。

$$x^*=\begin{bmatrix}2^0x_1\\\vdots\\2^0x_8\\2^1x_1\\\vdots\\2^1x_8\\2^2x_1\\\vdots\\2^2x_8\end{bmatrix}$$

注意用 2^1 次乘法仅需下标移位操作。此实施例的关键是确定：

$$y=A\cdot x=A^{*}\cdot x^{*}$$

这样随后的工作为通过 0-1 方法计算 $A^{*}\cdot x^{*}$。因此下一工作是收集和把共同非零列的系数相加，产生新系数并从而减少矩阵的大小。

因此以下成立：

$$y=w_1\begin{bmatrix}1\\0\\0\end{bmatrix}+w_2\begin{bmatrix}0\\1\\0\end{bmatrix}+w_3\begin{bmatrix}1\\1\\0\end{bmatrix}+w_4\begin{bmatrix}0\\0\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\\1\end{bmatrix}+w_6\begin{bmatrix}0\\1\\1\end{bmatrix}$$

式中：

$$w_1=2^2\cdot x_7+2^2\cdot x_8+2^1\cdot x_4+2^1\cdot x_5+2^0\cdot x_5$$

$$w_2=2^2\cdot x_1+2^1\cdot x_6+2^0\cdot x_1+2^0\cdot x_7$$

$$w_3=2^2\cdot x_3$$

$$w_4=2^2\cdot x_2+2^1\cdot x_2+2^1\cdot x_3$$

$$w_5=2^1\cdot x_5+2^0\cdot x_2+2^0\cdot x_1+2^0\cdot x_3+2^0\cdot x_8$$

$$w_6=2^2\cdot x_1+2^1\cdot x_6+2^0\cdot x_1+2^0\cdot x_7$$

这是在花费 16 次加法后完成。其次，产生的矩阵进行行分割。因此：

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}=w_1\begin{bmatrix}1\\0\end{bmatrix}+w_2\begin{bmatrix}0\\1\end{bmatrix}+w_3\begin{bmatrix}1\\1\end{bmatrix}+w_5\begin{bmatrix}1\\0\end{bmatrix}+w_6\begin{bmatrix}0\\1\end{bmatrix}$$

和：

$$y_3=w_1+w_3+w_5$$

现在，在第一方程式中把公共列的系数相加，得到：

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}=(w_2+w_5)\begin{bmatrix}1\\0\end{bmatrix}+(w_2+w_6)\begin{bmatrix}0\\1\end{bmatrix}+w_3\begin{bmatrix}1\\1\end{bmatrix}$$

这一步花费 2 次加法。其余再用 2 次加法完成。在该例子中总共需要 20 次加法来计算变换过程。常规方法的使用将需要相当于 29 次加法的操作，其中“相当”意味着包括在乘法内的加法也计算在内。

显然这一例子并未得到显著的节约。这不过是为了简单地引入发明实施例的概念。不过当矩阵的参数很多时（维数和各元素的位数）可获得相当大的节约。

一般而言，研究中的发明优选实施例涉及一种有效计算实数、无穷制、线性变换方法。表示变换的 r×n 矩阵 A 可写成：

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & \cdot & a_{2n} \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ a_{r1} & & & & & & a_{m} \end{bmatrix}$$

式中矩阵的元素为实数。期望计算 A·x，其中 x 为实数或复数标量元素的 n-维向量，可写为：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

在此点前后假定矩阵的元素写为二进制并具有固定位数。提出发明的两种优选实施例作为可选解决方案，而它们之间的选择取决于变换矩阵的结构。第一种称作 0-1-分解，而第二种称作 U-分解。

假定矩阵 A 元素的界限为 2^{m1} 并对于适当的正整数 m_1 及 m_2 在此点以外具有 m 个二进位精确度。这一假定并不限制发明的范围，因为处理器遭遇的每一标量在此点前后具有有限位数。令 $m=m_1+m_2+1$。将应用分布算术以解开乘积 A·x 成为上述例子中处理过形式的 m 个二进制乘积。

首先本发明在下列情况描述：矩阵的元素为非负而分解是基于 0-1 制。在此点外由 2^{m1} 范围限制的实数 t 并具有 m_2 个二进位可以下形式表示：

$$t = \square_{-m_2 \le k \le m_1} t_k \cdot 2^k$$

其中各 t_k 为 0 或 1。这是标准二进制表达式。通过典型分布算术论点 A 可以分解为 m 个 r×n 0-1-二进制矩阵之和，其方式如下：

$$A = \square_{-m_2 \le k \le m_1} 2^k \cdot A[k]$$

在此总和中 $A[m_1]$ 为最大有效位（MSB）0-1-矩阵，$A[-m_2]$ 为最小有效位（LSB）0-1-矩阵。一般，各矩阵 A[k]为 0-1-矩阵，其元素由 A 的元素中 k 有效位组成，其中各位放置在相应的元素中。通过分布法则可得出：

$$A \cdot x = \square_{-m_2 \le k \le m_1} 2^k \cdot A[k] \cdot x.$$

这是当前发明实施例的基础。

其次令人 A*为 rx(m·n) 0-1-矩阵，由上述总和的二进制矩阵的水平行组成，其次序按总和中表示。然后，

$$A^* = [\,A[-m_2],\ldots\ldots,A[0],\ldots\ldots,A[m_1]\,]$$

这将在输入向量到达开始前，对于任何给定的矩阵一次被完成。

此外，令 m·n-维列向量 x*由下式给出：

$$x^* = \begin{bmatrix} 2^{-m_2}x \\ 2^{-m_2+1}x \\ \cdot \\ \cdot \\ 2^{0}x \\ \cdot \\ \cdot \\ \cdot \\ 2^{m_1}x \end{bmatrix}$$

考虑中本实施例所含关键内容为：

$$A \cdot x = A^* \cdot x^*$$

后者是 rx(m·n)的 0-1-矩阵与 (m+1)·n-维向量之积，该向量是由原向量移位 m 次的复制品组成。

A*·x*积的计算用本发明的 0-1-二进制实施例完成。由于各用 2 的整数幂的乘法可以用移位实现，因此仅添加很少复杂度。

本发明的该实施例可以显著地减少乘积的复杂度。令：

$$l = \log_2(m \cdot n) = \log_2(m) + \log_2(n)$$

并且令 C(A)为本发明计算 A*·x*需要的加法次数。如果 l≥r 则：

$$C(A) < m \cdot n + 2^r + 2^{r/2+1} - r.$$

特别可以看到在此情况粗略的界限：

$$C(A) < 2m \cdot n.$$

当假定 l≥r，则：

$$C(A) < (1+\square) \cdot m \cdot r \cdot n / l$$

式中 1 > □ > 0 并且当 m·n 及 r 趋向于无穷时□趋向于零。

计算 A·x 积的常规（硬性）现有技术方法平均需要相当于 n·r·m/2 次加法的，其中包括乘法操作中所需的加法。

在某些情况下，特别当 r=1，或当 r 较小时，可以采用上述实施例的另一种变化以便进一步节约。注意在 r=1 的情况问题简化为两向量之间的标量积的问题。

这本身在科学技术上是相当普通的计算，并且它的有效性能非常有用。按照此变化矩阵 A**由一连串矩阵形成，

$A[-m_2]$, $A[-m_2]$, ..., A[0], ..., $A[m_1]$按垂直顺序。这样，r·(m+1)xn 的 0-1-矩阵为：

$$A^{**} = \begin{bmatrix} A[-m_2] \\ \cdot \\ \cdot \\ \cdot \\ A[0] \\ \cdot \\ \cdot \\ \cdot \\ A[m_1] \end{bmatrix}$$

其次，A**·x 的计算用本发明的 0-1-矩阵优选实施例完成。乘积 A**·x 含有所有乘积：$A[-m_2]\cdot x$, ..., A[0]·x, ..., $A[m_1]\cdot x$。因此对于二进制乘法应用移位，期望的结果用执行总和完成：

$$y = \square_{-m_2 \le k \le m_1} 2^k \bullet A[k] \bullet x$$

这推断出带有非负元素的矩阵部分。

当矩阵 A 为实数并且其元素具有两种符号时，矩阵 A 可以表示为两个具有非负元素的矩阵之差：$A = A_1 - A_2$。按照此表达式，讨论中的发明通过首先用上述方法分别地或以组合形式计算各乘积 $y_1 = A_1 \cdot x$ 和 $y_2 = A_2 \cdot x$ 来执行 $y = A \cdot x$。然后最后步骤为执行减法：$y = y_1 - y_2$

本发明优选实施例的关于实数矩阵的 0-1-二进制选择方案在下列分解二进制矩阵：$A[-m_2]$, $A[-m_2+1]$, ..., A[0], ... $A[m_1]$ 为较稀疏矩阵时特别有效。这可以是大小不同并且在此点外不具有非一致二进制位数目 A-元素。在这种情况下，上述一致数字位格式所需的零填充造成较高的稀疏性。

下面将根据 U-二进制分布算术来描述本发明优选实施例的另一种形式。本发明的这一形式具有更适于具有两种符号的矩阵并且基于更快的 U-方法的优点。实际中，当在矩阵元素的大小和精确度方面有某些一致性时，它比上面的 0-1-方法更加有效。下列方法的主要特征类似于 0-1-方法。

本发明的随后描述需要下列守则。被 2^{m1} 限定并在此点外具有 m_2 个位数的实数 t 可以用以下方式表示为 U-二进制和：

$$t = \square_{-m_2-1 \le k \le m_1-1} s_k \bullet 2^k + s_{m_1} \bullet (2^{m1} - 2^{-m2-1})$$

其中所有 s_k 对于 $m_2-1 \leqslant k \leqslant m_1-1$ 均为±1，且当 t 为非负时 $s_{m1}=1$，当 t 为负时 $s_{m1}=-1$。

因此具有两种符号 r×n 实数矩阵 A 可以用下列方式被分解为 U-矩阵的和：

$$A = \square_{-m_2-1 \le k \le m_1-1} 2^k \cdot A[k] + (2^{m_1} - 2^{-m_2-1}) \cdot A[m_1].$$

式中各矩阵 A[k]为 r×n U-矩阵。

令 A*为 rx((m=1)·n)的 U-矩阵：

$$A^* = [A[-m_2-1],\ A[-m_2],\ldots\ldots\ldots\ldots,A[0],\ldots\ldots\ldots\ldots,A[m_1-1],\ A[m_1]]$$

这一矩阵是在进入数据向量到达前每矩阵一次地被组成。

此外 (m+1)·n-维列向量 x*定义为：

$$x^* = \begin{bmatrix} 2^{-m_1-1}x \\ 2^{-m_2}x \\ \cdot \\ \cdot \\ 2^0 x \\ \cdot \\ \cdot \\ 2^{m_1-1}x \\ (2^{m_1} - 2^{-m_2-1}) \cdot x \end{bmatrix}$$

因此成立：

$$A \cdot x = A^* \cdot x^*$$

这是 rx((m+1)·n)的 U-矩阵与 (m+1)·n-维向量的乘积。此乘积的计算是应用本发明的 U-矩阵实施例完成。

如在上述关于 0-1-二进制分解的方法中，同样这里存在对于 r=1 或小 r 值的垂直方案。它完全类似于上面描述的情况并且没有必要重复细节。

令 l=log((m+1)·n)=log(m+1)+log(n)。成立对于 l≥r，执行上述方法所需的加法次数由下式限制：

$$C(A) < (m+1) \cdot n + 2^{r-1} + 2^{r/2+1} - r.$$

如上所述，C(A)被定义为上面计算 A·x 的 U-实施例所需的加法次数。在更广义的情况下，成立：

$$C(A) < (1+\square) \cdot (m+1) \cdot r \cdot n/l$$

式中 1 > □ > 0 并且当(m+1)·n 及 r 趋向于无穷时□趋向于零。

工业应用的例子

线性变换普遍地应用于科学技术的每一领域。在通信技术中本发明实数矩阵的应用方面包括多用户检测器（MUD）矩阵（诸如反相关器或最小均方误差（MMSE）矩阵）与去扩展器的输出向量的乘积。它也可用于最小平方的计算。有限脉冲响应（FIR）滤波器，其中离散傅里叶变换（DFT）全部或部分被计算。特别是部分 DFT 的 FIR 滤波器以及那些其中 FFT 不太有效的中小尺寸的 FIR 滤波器是本发明特征可以应用的字段合。离散余弦变换 DCT 是另一种线性变换形式，它的计算可以由本发明来改进。当仅部分地被计算或者当其尺寸并不太大尤其如此，因此较高维数的快速算法不很有效。

在一些数字信号处理应用中，诸如应用 FIR 的处理电路，需要两个相对较长的向量的相关。其中一个向量可以表示运行在第二向量（代表应该滤波的输入）上的 FIR 滤波器的分接头。包括部分卷积的滤波操作由 TOPLITZ 矩阵和向量的乘积来表示。这可以有效地通过本发明实数矩阵方面来完成。

复数矩阵

例： 假定要求计算下列和：

$$y_1 = (1+j)\cdot x_1 + (1-j)\cdot x_2 + (-1-j)\cdot x_3 + (-1+j)\cdot x_4 + (1-j)\cdot x_5 + (-1+j)\cdot x_6$$

$$y_2 = (-1+j)\cdot x_1 + (1+j)\cdot x_2 + (1+j)\cdot x_3 + (-1-j)\cdot x_4 + (-1-j)\cdot x_5 + (1-j)\cdot x_6$$

$$y_3 = (1-j)\cdot x_1 + (-1-j)\cdot x_2 + (-1-j)\cdot x_3 + (1+j)\cdot x_4 + (-1+j)\cdot x_5 + (1+j)\cdot x_6$$

式中输入向量 $x_1, x_2, \ldots\ldots, x_6$ 为复数。

常规的现有技术方法需要 66 次实数加法。当考虑到因子(±1±j)与复数的乘积需要 2 次实数加法而两个复数的一次加法需要两次实数加法，就可以理解这一点。

为完成此计算的发明优选实施例的两个主要可选方案提出如下。发明的第一优选实施例将称作相位旋转加 GEM，它应用下列事实：

$$\frac{1}{2}(1+j)\cdot(1+j) = j$$

$$\frac{1}{2}(1+j)\cdot(1-j) = 1$$

$$\frac{1}{2}(1+j)\cdot(-1+j) = -1$$

$$\frac{1}{2}(1+j)\cdot(-1-j) = -j$$

因此通过把上述用 1/2(1+j)乘以所有上述和，这里该行为被称作相位旋转，我们可以从集合{1, -1, j, -j}中获得系数如下：

$\{1, -1, j, -j\}$:

$$\frac{1}{2}(1+j)\cdot y_1 = j\cdot x_1 + 1\cdot x_2 + (-j)\cdot x_3 + (-1)\cdot x_4 + 1\cdot x_5 + (-1)\cdot x_6$$

$$\frac{1}{2}(1+j)\cdot y_2 = (-1)\cdot x_1 + j\cdot x_2 + j\cdot x_3 + (-j)\cdot x_4 + (-j)\cdot x_5 + 1\cdot x_6$$

$$\frac{1}{2}(1+j)\cdot y_3 = 1\cdot x_1 + (-j)\cdot x_2 + (-j)\cdot x_3 + j\cdot x_4 + (-1)\cdot x_5 + j\cdot x_6$$

按照发明优选实施例这些和数用 GEM 计算。由于这个例子规模很小，与常规方法相比得益是边际的（无区别），但在更大的维数时它将是显著的。当维数很小，如在此例子中，在相旋转步骤以后可以采用其它更常规的计算方法。最后各总和的结果被乘以(1-j)以获得期望的结果。注意：用 j 或(-1)是组织性操作，它需要适度的时间和能量。

优选实施例的第二种可选方案称作复数-U-方法。它通过开启各系数的括号来打开总和，其方法如下：

$$y_1 = x_1 + (jx_1) + x_2 - (jx_2) - x_3 - (jx_3) - x_4 + (jx_4) + x_5 - (jx_5) - x_6 + (jx_6)$$

$$y_2 = -x_1 + (jx_1) + x_2 + (jx_2) + x_3 + (jx_3) - x_4 - (jx_4) - x_5 - (jx_5) + x_6 - (jx_6) y_2$$

$$y_3 = x_1 - (jx_1) - x_2 - (jx_2) - x_3 - (jx_3) + x_4 + (jx_4) - x_5 + (jx_5) + x_6 + (jx_6)$$

其余应用 U-方法完成。由于 s(12,3)=12+3=15，其中这是复数加法次数，于是通过利用上述 s(r,n)表格，这最多需要 30 次实数加法。

在通过上述例子说明基本原理后，现在应该提出通常情况的详细描述。为描述发明优选实施例关于复数矩阵方面，考虑一种将要用作系数的集合：U_1= {1, -1, j, -j}和 U_2 = {1+j, 1-j, -1+j, -1-j}。一个 U_1-向量或者 U_1-矩阵的元素是矩阵 U_1 的元素。同样 U_2-向量或者 U_2-矩阵的元素是矩阵 U_2 的元素。这类矩阵和向量在无线应用中很普通。接下来应该考虑 U_2 数和复数的乘积需要 2 次实数加法，而 U_1 数和复数的乘积牵涉相对较少的复杂度。

第一个要解决的问题是：假定有 r×n U2-矩阵 A 和 n-维复数列输入向量 x，要求计算乘积 y = A·x。实际上是标量乘积的情况 r=1 被包括在内。现给出两种主要计算方法。各按适合情况选择。通过少许变化可以应用于实数数据向量。首先引入发明优选实施例的相旋转加上 GEM 方法。

令 $$B=\frac{1}{2}(1+j)\cdot A \quad \text{and} \quad z=\frac{1}{2}(1+j)y$$

于是 B 为 r×n 的 U_1-矩阵且 z=B·x。其次，z = B·x 的积通过 GEM 法计算。一旦 z 计算以后，输出向量通过乘积 y =(1-j)·z 求出。比较常规方法的得益是从初始的相旋转步骤获得，即节约 2r·(n-1)次加法。该得益即使在 r=1 的情况也是存在的，即标量积的情况。进一步的收益可以从应用 GEM 获得。

本发明的第二优选实施例称作 U-复数-方法。首先用 $A = A_1 + jA_2$ 表示 A，其中 A_1 和 A_2 为 U-矩阵。然后考虑等式：$A \cdot x = A_1 \cdot x + j\ A_2 \cdot x$。该等式意味着 A ·x 可以通过 r×2n U-矩阵 $A^* = [A_1, j\ A_2]$ 与 2n 维-复数-列向量：$x^* = \begin{bmatrix} x \\ jx \end{bmatrix}$ 的乘积计算。这表示在下列等式：$A \cdot x = A^* \cdot x^*$。现在乘积 $A^* \cdot x^*$ 将用 U-方法计算。在此发明优选实施例中有另一种选择，当 r 很小时是合理的。令 $A^{**} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$，这是 2r×n U-矩阵。然后应用 U-方法计算乘积：$A^{**} \cdot x$。这实际上是计算乘积 $y_1 = A_1 \cdot x$ 和 $y_2 = A_2 \cdot x$。该过程用加法：$y = y_1 + y_2$ 完成。

在某些应用中可能要对上述问题作一些变化，这些问题包括 CDMA 中以 TOPLITZ 矩阵代表 PN 相关器。在此布置中矩阵可以有零元素。相应地，令：$U'_1 = \{0, 1, -1, j, -j\}$ 以及 $U'_2 = \{0, 1+j, 1-j, -1+j, -1, -j\}$。$U'_1$ -向量或 U'_1 -矩阵具有在 U'_1 中的元素。相似地 U'_2-向量或 U'_2-矩阵具有在 U'_2 中的元素。给定一 U'_2-矩阵 A 和一 n-维复数-列输入向量 x。目的是计算乘积 $y = A \cdot x$。包括实际上是标量积的情况 r=1。

发明实施例相旋转加上 GEM 将首先讨论。令

$$B = \frac{1}{2}(1+j) \cdot A \qquad \text{以及} \qquad z = \frac{1}{2}(1+j)y$$

则 B 是 r×n 的 U'_1-矩阵和 $z = B \cdot x$。其次应用 GEM 计算 $z = B \cdot x$ 的积，或者当维数较小时应用更常规的方法。最后，一旦 z 已计算，输出向量 y 利用乘积 $y = (1-j) \cdot z$ 求出。

按照本发明另一实施例，A 首先用总和：$A = A_1 + jA_2$ 来表示，式中 A_1 及 A2 为（0，1，-1）-矩阵，很可能是 TOPLITZ 矩阵。于是通过等式：$A \cdot x = A_1 \cdot x + j\ A_2 \cdot x$，乘积 A ·x 可以通过 r×2n 的（0，1，-1）-矩阵 $A^* = [A_1, A_2]$ 与 2n-维-复数列向量 $x^* = \begin{bmatrix} x \\ jx \end{bmatrix}$ 的乘积求得。最后乘积 $A^* \cdot x^*$ 将用 TOPLITZ 或本发明更广义的(0,1,-1) 方面求得。

本发明的另一（可选）优选实施例，反映其非-TOPLITZ 对等部分，当 r 较小时：令 $A^{**} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$，此为 2r×n 的（0，1，-1）-矩阵。然后应用本发明的（0，1，-1）方面计算 A^{**}。这实际上是计算 $y_1 = A_1 \cdot x$ 和 $y_2 = A_2 \cdot x$。最后该过程用加法：$y = y_1 + jy_2$ 完成。

复数部分现在将用一种计算一个广义复数 r×n 矩阵 $A \in C^{r \times n}$ 与一个实数或复数

n-维向量 x 的乘积的方法作为结束。按照发明的一种优选实施例，首先以总和：A=A_1+ jA_2，式中 A_1 和 A_2 为实数矩阵。其次从等式 A·x = A_1·x+j A_2·x 可以推导得出：A·x 可以通过 2r×n 实数-矩阵 A*=[A_1, A_2]与 2n-维-列向量 $x^* = \begin{bmatrix} x \\ jx \end{bmatrix}$ 的乘积求得，因为 A·x=A*·x*。最后乘积 A*·x*将用实数矩阵方法计算。

按照发明另一（可选）优选实施例令 $A^{**} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$，此为 2r×n 实数-矩阵。然后应用实数矩阵方法计算乘积 A**·x。这是计算乘积 y_1=A_1·x 和 y_2=A_2·x。最后该过程用加法：y=y_1+jy_2 完成。

最后具有 U_2-系数的 TOPLITZ 矩阵与向量的乘积是通过应用上述 TOPLITZ 技术完成。

工业上应用 TOPLITZ 矩阵、(0,1,-1)-矩阵和复数矩阵技术的例子

IS-95 搜索器：IS-95 CDMA 系统容许一定数量的基站在一较小地理区域中同时使用同一段频谱以传输数据到移动接收器。从不同的基站输出的数据模式可以通过用来扩展传输数据的 PN 序列加以区分。各基站在 PN 序列的不同相位上传输。搜索器机构在移动接收器中的任务是通过调准它们的 PN 相位识别不同的由周围基站传输的先导信号。它也应用于区别从同一基站到达的几个多通道之间的信号（回声）。在初始同步程序中应用相似的过程。

搜索器用来通过收到信号的部分相关性试验一定数量的假设，对于各假设用当地产生的 PN 序列。然后对于每一假设序列被移位，并且对于每一固定数目的信号元素（芯片）执行相关操作。正常情况下搜索器需要搜索在给定窗口的所有假设，该窗口每次序列被移位 1。在 DS-CDMA 系统中搜索器可以通过建立矩阵 A 实现，该矩阵的行由上述移位后的 PN 结果组成。搜索结果然后然后在向量 y = A·x 给出，其中 x 代表在单个芯片时期采样的进入信号。按照发明的优选实施例上述有效线性变换用的发明性算法的实现可以耗费较少资源获得向量 y。本发明的众多应用可以在建议的宽带 CDMA 标准有用。

多 乘 积

本发明的另一优选实施例设计一种要求 U-矩阵和向量的乘积的部分和的情况。这可以发生在 CDMA 通信应用中，当具有不同速率（扩展因子）的几种编码被同时测试。对于已经完成掌握本发明的前面几方面的读者研究这一实施例特别有成

效。通过下列例子予以介绍，它给出该相当复杂的方法的初步概念而不需要大量的细节。不过没有一个合理大小的例子可以描述此实施例的所有方面。读者可以参考本发明的概要，第六项。

例子：考虑下列 5×8 U-矩阵

$$A=\begin{bmatrix} 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix}$$

和一个 8-维输入向量：

$$x=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

假如说（在多乘积术语中）行 1 及 2 的扩展因子是 2，行 3 及 4 的扩展因子是 4，而行 5 的扩展因子是 8。着意味在首先二行中，每相继二进制素相加，在第三及第四行中每相继四元素相加，而在第五行中整行元素均相加。规定扩展因子不可逐渐减少，即每一行的扩展因子等于或大于前一行的扩展因子。此术语将在以后精确定义。

从上意味应该计算下列和：

$x_1 + x_2$， $-x_3 + x_4$， $x_5 - x_6$， $-x_7 - x_8$

$-x_1 + x_2$， $-x_3 - x_4$， $x_5 + x_6$， $-x_7 + x_8$

$x_1 - x_2 + x_3 + x_4$， $x_5 - x_6 + x_7 - x_8$

$x_1 + x_2 - x_3 + x_4$， $-x_5 - x_6 + x_7 - x_8$

$-x_1 - x_2 + x_3 + x_4 - x_5 + x_6 - x_7 - x_8$

首先对于四个 4×2 U-矩阵在水平维数反映最低扩展因子处进行分割。然后应用 U-方法，在最基本水平上，以说明通过新的方面如何节约加法次数。如此，通过应用相等行的消元，只需要二次加法计算各行的所有初次和数：

$x_1 + x_2$

$-x_1 + x_2$

$x_1 - x_2$

$x_1 + x_2$

$-x_1 - x_2$

相似地对于各行的第二总和只需要二次加法：

$-x_3 + x_4$

$-x_3 - x_4$

$x_3 + x_4$

$-x_3 + x_4$

$x_3 + x_4$

如此等等。因此，这部分总共需要花费 8 次加法。必须有 4 次附加的加法完成第 3 和第 4 行所需的和，和另外 4 次附加的加法计算第五行需要的和。因此在应用发明的优选实施例中总共需要 16 次加法。常规的硬性方法将需要 28 次加法以完成同样的任务。

发明当前方面的环境包括 U-矩阵，它可能是大维数的并且如上述例子中一样需要求各行中相等间隔的次级和。输入向量可以是实数或复数。矩阵可以再次按行分割成几个子矩阵，其中每一个按照上述例子使用方法分别地和独立地计算。在此实施例中综合一种方法，它在节约加法次数前提下寻找最近-最佳的分割从而减少复杂度。它的工具是加法处理器或者基于动态编程的装置，通过使用 U-方法的表格、上下限、和复杂度 s(n,r)的递归公式，可以分析不同的次级分割。为开发该实施例需要有十分精确的公式表示。需要作出新的定义作为初步材料 。

令 $v=(v_1, v_1, \ldots\ldots, v_n)$ 是向量而 p 为可以除尽 n 的正整数（简言之 $p \mid n$）。定义 v[p]为由分割 v 成为长度为 p 的分段而形成的诸向量的向量。如此：

$$v[p]= ((v_1,v_2,\ldots\ldots,v_p),\ (v_{p+1},v_{p+2},\ldots\ldots,v_{2p}),\ldots\ldots\ldots\ldots\ldots\ldots,(v_n - p +1,v_n - p + 2,\ldots\ldots,v_n)).$$

分段表示为：

$$v[p,k]= (v_{(k-1)p + 1},v_2,\ldots\ldots\ldots\ldots\ldots,v_{kp}) \quad \text{对于} \quad 1 \leq k \leq n/p.$$

在文中整数 p 称作扩展因子。注意多向量是类似于矩阵结构的结构。下一元素，多向量的多-标量积，是矩阵乘积与通常标量乘积之间的交叉。取两个 n-维向量 $v=(v_1, v_2, \ldots\ldots, v_n)$ 和 $w=(w_1, w_1, \ldots\ldots, w_n)$，然后 v 和 w 的 p-多-标量-乘积义为：

$$v\bullet w[p]=(v[p,1]\bullet w[p,1],\ v[p,2]\bullet w[p,2],\ldots\ldots\ldots\ldots\ldots,v[p,n/p]\bullet w[p,n/p])$$

式中内部积：$v[p,1]\cdot w[p,1]$, $v[p,2]\cdot w[p,2]$, ...为普通标量积。注意此乘积的结果是 n/p-维向量。

令 A 为具有 $A_1, \ldots\ldots, A_r$ 行的 $r\times n$ 矩阵和 $p=(p_1, p_2, \ldots\ldots, p_r)$ 为正整数向量。

可以说如果在 1≤i≤r 时每一个 pi 除尽 n 则 p 除尽 n。这表示为：p | n。假定现在 p | n。取 n-维向量 x 并定义 A 和 x 的多乘积乘积为：A·x[p]，为诸向量的向量：

$$A \cdot x[p] = (A_1 \cdot x[p_1], A_2 \cdot x[p_2], \ldots\ldots, A_r \cdot x[p_r])$$

本发明的当前实施例在以后将描述的设置中将改进这类乘积的计算。

多乘积系统

多乘积系统（或简言之 MP-系统）是包括以整数 n,r 和正整数向量 p=(p_1, p_2, ..., p_r)作为参数的设置，并使：

$p_1 \mid p_2$, $p_2 \mid p_3$, $p_3 \mid p_4$, ..., $p_{r-1} \mid p_r$ 和 $p_r \mid n$ 。

整数 p_1, p_2, ..., p_r 称作系统的扩展因子。MP 系统的参数将以下列形式书写：

$$P = (r, n, p)$$

在这些参数上现在附着 r×n U-矩阵 A 和 n-维实数向量 x。目的是有效地计算乘积 A·x[p]。整个 MP 系统将写为：

$$S = (r, n, p, A, x)$$

当所有整数 p_1, p_2, ..., p_r, n 也为 2 的幂数，则系统称为二进制多乘积系统或简言之，BMP 系统。

M-1-方法

此发明的特征是直接使 U-方法适配于 MP-系统。它用上述例子代表。实际上它通常在以后将描述的接近-最佳分割后应用于 MP 系统的子系统。

给定一 MP-系统 S = (r, n, p, A, x)。该方法基于关于最小扩展因子 p_1 下矩阵水平分割为子矩阵，各子矩阵宽度为 p_1。它开始于用矩阵 A 的 U-系数计算向量 x 的首先 p_1 个实数的和。这同时在所有行用 U-二进制方法完成。然后进入下面的 p_1 列，完成同样的过程。它以此方式进行直至 n 个变数均完成。其次，进入各行，其中 $p_i > p_1$ 并以琐碎的方式完成取和过程。

在各子矩阵上 U-方法的第一步是扫描出那些具有其它行直接复制件或负复制件的行。因此，例如，如果 p_1=2，则不管 r 如何在每一区段至多需要二次加法。一般在每一区段利用 U-二进制方法不考虑大于 2^{p1-1} 行。此外本文打算的应用是当 A 为 HADAMARD 矩阵。在此情况在各区段中没有多于 p_1 不相当的行。这样插入另一段信息关于在各子矩阵中可能出现多少不相当行数的上下限。它包含在标识为 z

（以表格贮存） 函数中，z 为即将到来的矩阵类型的推论。它的参数是 p_1 及 r $z(p_1,r)$。例如当 A 是 HADAMARD 矩阵则 z(4,6)=4，z(8,5)=5，当是一般 U-矩阵时则 z(4,40)=8。永远有 $z(p_1,r)\leqslant\min\{2^{p1-1},r\}$，而当 A 是 HADAMARD 矩阵时：$z(p_1,r)=\min\{p_1,r\}$。

计算 M-1-方法的复杂度将基于表格和出现在上述 U-二进制方法中的递归公式。对于每一正整数 y 令 s'(y)=s(y)+y。回忆起 s'(y)<2y-1+2y/2+1 并且此不等式相当紧密。着给出对于下一公式复杂度直觉的概念。M-1-方法所完成的加法因此是由下式所限定：

$$C(n,r,p,z) = (n/p_1)(p_1 + s(z(p_1,r))) + (n/p_2)(p_2/p_1 - 1) + (n/p_3)(p_3/p_1 - 1)$$

$$+\ldots\ldots\ldots\ldots$$

$$\ldots\ldots\ldots\ldots + (n/p_r)(p_r/p_1 - 1)$$

$$= n\cdot\left(1 + (s(z(p_1,r)) + r - 1)/p_1 - (1/p_2 + 1/p_3 + \ldots\ldots\ldots\ldots + 1/p_r)\right) =$$

$$= n\cdot\left(1 + s'(z(p_1,r))/p_1 - (1/p_1 + 1/p_2 + 1/p_3 + \ldots\ldots\ldots\ldots + 1/p_r)\right)$$

注意此公式中一些琐碎但不重要的情况。首先当矩阵只有一行，即 r=1，则：

$$C(n,r,p) = n - n/p_1$$

第二是一致的扩展因子情况，即当 p1=p2=.....=pr，则：

$$C(n,r,p,z) = n\cdot\left(1 + s(z(p_1,r))/p_1\right)$$

显然，当 r 相对于 p1 较大时 M-1-方法并不十分有效。它主要是一种在其基础上发展的更灵巧方法的踏脚石。更强大的方法通过水平方向分割矩阵并对每一子矩阵分别地应用 M-1-方法。为了用较少的计算次数寻找一种承受较少加法总数的区段结构，最好成绩有上述公式的如下较短的形式。故定义：

$$C^*(n,r,p,z) = 1 + s'(z(p_1,r))/p_1$$

多系统的再分割和多向量的广义化标注

其次，作出规定使矩阵在水平行上分割。这一指令用再分割向量 r 代表。结果是把原来问题打开为几和具有同样宽度-n 的子问题，各自用上述M-1-方法解决。稍后这种再分割将附加在一个发明性算法上以使效率最大化。取一个 r-维正整数向量，$p=(p_1, p_2, \ldots\ldots, p_r)$，具有扩展因子，和一个 r×n U-矩阵：

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & \cdot & a_{2n} \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ a_{r1} & & & & & & a_{m} \end{bmatrix}$$

并取整数 k, m 其中 1≤k≤m≤r。首先定义向量 p 的一个区段。：

$$p(k,m)=(p_k,\ldots\ldots,p_m).$$

这是简单地取具有 k 到 m 下标的分量。定义矩阵 A 的一个区段：

$$A(k,m)=\begin{bmatrix} a_{k1} & a_{k2} & \cdot & \cdot & \cdot & \cdot & a_{kn} \\ a_{k+1,1} & a_{k+1,2} & \cdot & \cdot & \cdot & \cdot & a_{k+1,n} \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ a_{m1} & & & & & & a_{mn} \end{bmatrix}$$

同样，它意味取具有下标 k 到 m 的行。

其次，考虑一个整数向量：r=(r(1)，……，r(r+1)) 可满足：

$$k=r(1)<r(2)<\ldots\ldots\ldots<r(t)<r(t+1)=m+1,$$

它是分割区段的工具。首先，r 是把 p 的分段创造成为诸向量 p[r]的向量，即下列形式：

$$p[r]=\left((p_{r(1)},\ldots\ldots,p_{r(2)-1}),(p_{r(2)},\ldots\ldots,p_{r(3)-1},)\ldots\ldots\ldots\ldots,(p_{r(t)},p_2,\ldots\ldots,p_{r(t+1)-1})\right)$$

子向量表示为，

$$p[r,1]=(p_{r(1)},\ldots\ldots\ldots\ldots,p_{r(2)-1})$$

$$p[r,2]=(p_{r(2)},\ldots\ldots\ldots\ldots,p_{r(3)-1})$$

.

.

.

$$p[r,t]=(p_{r(t)},\ldots\ldots\ldots\ldots,p_{r(t+1)-1})$$

相似地矩阵 A 的行按照分割向量 r 再分割为下列形式：

对于所有整数 1≤q≤t： $A[r,q]=\left(a_{ij}: r(q)\le i<r(q+1),\ 1\le j\le n\right)$

给定区段的 M-方法

此节为形成一种寻找低复杂度区段的机制中主要步骤，这种机制为此实施例的核心。假定給定一种矩阵的行分割并在个子矩阵上分别地执行 M-方法。目的是

估计加法的总数使其在下一阶段中寻到用较少次数加法获得区段。固定一个 MP-系统 S=(r, n, p=(p_1, p_2, ..., p_r), A, x, z) 和区段向量 r =(r(1), ..., r(t)), 式中 1≤k=r(1)<r(2)<...<r(t)<r(t+1)=m+1≤r+1

对于 1≤q≤t S_q MP-子系统给出如下：

$$S_q=\left(r(q+1)-r(q),\ n,\ p[r,q],\ A[r,q],x,z\right)$$

所有子系统的加法总数为：

$$C(n,r,r,p,z) = \square_{1\le q\le t} C\left(n,\ r(q+1) - r(q),\ p[r,q],z\right) =$$

$$= n\cdot\left(\square_{1\le q\le t}\ s'\left(z(p_{r(q)},\ (r(q+1) - r(q))\right)/p_{r(q)} - (1/p_k + \ldots\ldots\ldots + 1/p_m) + t\right)$$

下一目的将是发展一种寻找可以使此元素最小化区段的有效方法。不在各阶段计算重复附加的子元素：1/pk+.....+1/pm 和 n-因子是一种完成该计算的有效方法。

如此，我们定义：

$$C^*(n,r,r,p,z) = \square_{1\le q\le t}\ s'\left(z(p_{r(q)},\ (r(q+1) - r(q))\right)/p_{r(q)} + t$$

接近最佳的分割

在此阶段应该应用再分割段的特性，它尽量减少由上述项 C(n, r, r, p, z) 给出的加法次数。首先给出一个对于加法次数的抽象公式，其中分割段在原来问题的给定子间隙上具有在最坏情况的上下限。

然后考虑 MP-系统 S = (r, n, p=(p_1, p_2, ..., p_r), A, x, z)和整数 k, m, 其中 1≤k≤m≤n。 定义：

$$h(k,m) = \min\{C(n,\ m - k + 1,\ r,\ p(k,m),z):\ \text{对于}\ r=(r(1),\ldots\ldots,r(t+1))$$

其中 $k=r(1)<r(2)<\ldots\ldots<r(t)<r(t+1)=m+1\}$

$$h^*(k,m) = \min\{C^*(n,\ m - k + 1,\ r,\ p(k,m),z):\ \text{对于}\ r=(r(1),\ldots\ldots,r(t+1))$$

其中 $k=r(1)<r(2)<\ldots\ldots<r(t)<r(t+1)=m+1\}$

下式成立：

$$h(k,m) = n\cdot\left(h^*(k,m) + (1/p_k + \ldots\ldots\ldots + 1/p_m)\right)$$

下列递归公式成立：

$$h(k,m) = \min\{C(n,\ m-k+1,\ p(k,m),\ z),\ \min\{h(k,q-1) + h(q,m):$$

对于其中所有 $k < q \le m\}\}$

其中 min(空集) =无穷大。

$h^*(k,m) = \min\{C^*(n, m-k+1, p(k,m), z)\ \min\{h^*(k,q-1) + h^*(q,m):$

对于其中所有 $k < q \leq m\}\}$

其中 min(空集) =无穷大。

现在这些表达式被下列动态编码所用，其中子结构为 k 和 m 之间的间隙。为减少此编码的复杂度 h*在行进中需要更换 h 以发现最佳结构。这对最终结果没有影响。为在最佳分割中寻找第一步，将计算可以使表达式 h(k,q-1)+h(q,k)对每一 k 和 m（其中 1≤k≤m≤n）最小的 q,这与使表达式 h*(k,q-1)+h*(q,k)最小的 q 相同。这样，定义：

q(k,m) = k 当 h*(k,m) = C*(n,r,p(k,m),z) 和否则：

q(k,m) = 最小 q,其中 k<q≤m 和 h*(k,m)=h*(k,q-1)+h*(q,k)

现在可以形成最佳分割区编码。

可以寻找最佳分割区的动态编程创造性编码

下列编码接受 MP-系统 P=(r,n,p,z)参数作为数据，并且产生 M-方法最佳地执行的分区 r 作为输出。此外它也返回最佳 M-方法的复杂度。

要有效地运行必须在事前计算和贮存表格 s'(r)。这是用可以递归公式计算此表格的快速编码完成：

$s'(r) = 2^{r-1} + s'(r(1)) + s'(r(2))$

最佳分割区表(n,r,p,z)

对于 b 从 1 到 r-1 执行

对于 k 从 1 到 r-b 执行

m:=k+b

h*(k,m)←---C*(n,m-k+1, p(k,m),z)

q(k,m) ←---k

对于 q 从 k+1 到 m 执行

d←---h*(k,q-1)+ h*(q,m)

如果 d<h*(k,m)则

h*(k,m)←---d 和

q(k,m) ←---q

返回表格 h*和 q。

现在剩下的是要从通过此程序建立的表格中获得最佳分割向量。这是由下列创造包含最佳向量 r 分量的集合 R 的编码完成。

最佳分割区向量(n,r,p)

集合：R:=空集 以及 k:=1 m:=r

寻找向量(k,m):

如果 q(k,m)>k 则

令 q:=q(k,m)

集合 R 增加 q

寻找向量(q,m)

寻找向量（k,q-1）

返回集合 R。

现在最佳分割区 r 已找到，并且 M-方法将在该分区运行。

.....

实现

下列发明的优选实施例提供作为上述方法详细实现工具的创造性算法。它们可促成节约需要执行上述方法的存储和能量资源。它们有增进上述方法和提供需要建立器件指令的双重目的。关于实现的重要设想是变换矩阵是一般的并可以认为是从具有给定元素集合的矩阵集合中任意地选出。另一点是与列数相比行数相当地小，以至上述基本界限：行数<log(列数) 成立。因此在其它情况下有些步骤是适用的，而在这里检查相当的行是多余的。

随后文中描述的映射使从一处到下一处的数据信道化。每一处指定一个二进制地址，对应于在给定迭代中矩阵的一列。如果是 U-矩阵，一列的（-1）分量对应于地址中的 1，而类似地一列的 1 分量对应于地址中的 0。在 U_1-矩阵中也定义了相似的对应性。

将要描述的实现包括第一步，其中进入的信号被加入到预定的由其相应列决定的目的地，在列中个 x_j 可被乘以一个符号和/或 2 的幂。在前面的迭代中，每当一列按照发明被分割后，两分割部分指定的地址各小于或等于所给列的地址。这使每一贮存在贮存器内某些位置的信息在被处理和丢失前送往目的地。此外，如果一列的分割出半部中之一为零，代表此列的地址将用作下一次迭代。这些特性是通过新颖的数据流映射建设完成，该映射设立为最大限度减少数据的移动。在每一次迭代中，地址最大限度地保持不动，包括所有当前内容可以为下次迭带使用的。

下述实现的机制最好通过上述发明的 GEM 优选实施例理解。考虑有限集合数，S，包括数 1，并且在文中具有 r 行的矩阵称作完全的 S-r-矩阵，如果它包括 r-维列向量（其成分属于集合 S）的所有非零配置，其中各配置正好出现一次，并且标准化法则是最底下的元素应该是 1。注意下列小维数的例子。

矩阵：

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

是完全的(0,1)-2-矩阵；

矩阵：

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

是完全的 U2-矩阵；

矩阵：

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

是完全的 U-3 矩阵；

矩阵：

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

是完全的{0,1}-3-矩阵；

矩阵：

$$\begin{bmatrix} 1 & -1 & j & -j \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

是完全的{1,-1,j,-j}-2-矩阵。

随后的实现是基于假定，即行数与列数比较是足够地小以至大部分可能的配置出现在变换矩阵列中。因此在所有前面的实现中的第一次迭代简化计算为完全 S-r-矩阵与改进向量的乘积。因此所有进一步的迭代是计算较低维数完全 S 矩阵与各自改进向量的乘积。

各列出现在上述例子的矩阵中的有序方式反映这些实现的另一地址设置特征。这是一种基于把列转化为一个 | S | -基地址的一致性法则的寻址方式，其中 MSB 为向量的底部分量，而 LSB 为顶端。

基本结果是在第一次迭代后有一个唯一的一致完整变换矩阵，并具有一致的（相当明显）地址-编号方式。该一致的矩阵依靠 S 及 r 而不依靠初始的变换矩阵。这在过程的其余部分导致一致性，因此创造一种数据结构，它是普遍的而不依赖第一阶段后的特定变换矩阵。这在硬件实现和基于本发明的器件制造中非常有用，这是本章的主要目的。

这些实现的额外和单独的方面是减少所需读写存储分配和对于适当的变换矩阵的加法次数。因此，执行过程的能量需要和器件成本减少。应该注意，当以数个输入向量处理同样的初始矩阵时随后的各编码效率提高。

最后，在正确的观点上，下面的实现应该理解为它们仅是本发明许多可能有效的实现中几个例子。

例 1：0-1-二进制矩阵

以下步骤序列描述发明的0-1-矩阵方面。数据包括 r×n的 0-1-矩阵 $A=(a_{ij}: 0\cdot i<r, 0\cdot j<n)$ 和输入的实数或复数向量 $x=(x_0,\ldots,x_{n-1})$。矩阵 A 的列表示为 $v_0,\ldots\ldots,v_{n-1}$。该步骤序列包括计算乘积 $y=(y_0,\ldots,y_{r-1})^T=A\cdot x$。在每次给定的 k-迭代上，各位置包含一个实数或复数，这取决于向量 x。分配的读写存储包含 2^r-1 个地址，标识为从 1 到 2^r-1，代表在各次迭代中参加过程的列。以下定义为本发明优选实施例的部分描述。

定义.

1）对于所有 k¬0，j¬1 定义： $l(k,j)=\lceil (j-1)\cdot(r/2^k)\rceil$

2）对于所有 0·m<r 定义 $Ym=2^m$。这些地址在过程结束时将包含输出向量 $y=(y_0,\ldots,y_{n-1})^T$，其中 Y_0 为 y_0 的地址，Y_2 为 y_1 的地址，如此等等。

3）对于 k¬ 0 和 j¬1 定义如下函数 Fk, j, Gk, j, 控制数据从各地址移动到下一处。首先令：l=l(k, j)， m=l(k+1, 2j)，h=l(k, j+1)-1。

其次对于各正整数 v¬ 0，定义：

$$F_{kj}(v)=2^m\cdot\lfloor v/2^m\rfloor \qquad \text{[方程 100]}$$

$$G_{kj}(v)=v \bmod 2^m \qquad \text{[方程 101]}$$

4）对于每一向量 $v=(v_0,\ldots\ldots,v_{r-1})\in\{0,1\}^r$ 定义：

$$\square(v)=\sum_{0\le j<r} 2^j v_j.$$

编码.

1. **初始化：**对每一从 1 到 2^r-1 的地址填入 0。

2. **第一阶段：**对于每一 j 从 0 到 n-1 执行

如果 $v_j\neq 0$ 则令 x_j 加到地址 $v=\square(v_j)$

3. **主要部分：**

对于 k 从 0 到 $\lceil\log(r)\rceil-1$ 执行

对于 j 从 1 到 2k 执行

如果 l(k, j+1)-1 > l(k, j) 则

对于 p 从 1 到 $2^{l(k,j+1)-l(k,j)}-1$ 执行

令 $v=2^{l(k,j)}\cdot p$

对于（源）地址 v 执行

如果 $F_{k,j}(v)\neq 0$ 和 $F_{k,j}(v)\neq v$ 则

把驻留在（源）地址 v 中的值加入（目的）地址 $G_{k,j}(v)$。

并且也把此值（源地址 v 的值）加入（目的）地址 $F_{k,j}$（v）中的值。

4．获得输出：

在此阶段，对于所有 0·i<r 每一地址 Y_i 含有输出分量 y_i 的值。

复杂度.

在本术语中基本步骤是从存储器中一个存储单元（称作源）读取数字，并将其加入存储器中另一存储单元（称作目的）。

基于上述编码的器件将需要下列数目的基本步骤以完成上述 A·x 计算的主要部分：

$$C^{0,1}_{r} \odot 2^{r+1} - 2r - 2.$$

此数仅取决于 r 。

A·x 的整个计算这样仅需要最多为下列总和：

$$C^{0,1}_{n,r} \odot n + C^{0,1}_{r}$$

等基本步骤。

图表 1：0-1-二进制矩阵

图 1 示意地阐明更新一种器件使用的存储器内容的操作，该器件执行上述编码的主要部分，这样在较佳的模式下利用包括 4 行的 0-1-二进制矩阵实现发明的 0-1-二进制方面。在各次迭代中各矩阵的列以二进制方式用存储器中一个地址代表。底部（0 或 1）分量（在水平表达时最右侧分量）为 MSB，而顶部（0 或 1）分量（在水平表达时最左侧分量）为 LSB。地址 Ym = 2^m, 0·m·3，在过程结束时含有输出向量 $y = (y_0, \ldots\ldots, y_3)^T$ 的分量，其中 Y_0 为 y_0 的地址，Y_1 为 y_1 的地址，如此等等。箭头表示取一个地址内容并且将其送往另一地址并相加的行为。这是按照迭代顺序和参加各次迭代的地址（递增）顺序（如上述编码所示）而完成。

例 2：U-矩阵

下列步骤序列描述发明的 U-矩阵方面的实现。数据包括一个 r×n U-矩阵 $A = (a_{ij}: 0 \cdot i < r, 0 \cdot j < n)$ 和一个输入实数向量 $x=(x_0, \ldots, x_{n-1})$。矩阵 A 的列用 $w_0, \ldots\ldots, w_{n-1}$ 表示。该步骤序列计算 $y = (y_0, \ldots, y_{n-1})^T = A \cdot x$。在每一次迭代中各位置含有取决于向量 x 的实数或复数。分配的读写存储器包含 $2^{r-1}+r-1$ 个地址，其标识从 0 到 $2^{r-1}+r-2$。下列定义和先前例子的定义需要用来描述发明的目前优选实施例。

定义.

1）对于各 r –维 U-向量，$u = (u_0, \ldots\ldots, u_{r-1})$ 定义：

$Sign(u)=u_{r-1}$

$h(u) = (u_0 \cdot u_{r-1}, \ldots\ldots\ldots\ldots, u_{r-1} \cdot u_{r-1})$.

2）定义双极二进制集合 U ={1, -1}和对数位二进制集合 B={0, 1}之间的对应关系，通过下列关系式建立对应关系：

$(-1)' = 1$

$1' = 0$.

从而对于 r-维 U-向量，$u = (u_0, \ldots\ldots, u_{r-1})$，定义：

$\pi(u)=\sum_{0\le j<r} 2^j u'_j$.

3）定义 $Y_0=0$，并且对于所有 1·m·r-1: $Ym = 2^{r-1}+m-1$。在过程结束时这些地址将含有输出向量 $y=(y_0, \ldots, y_{n-1})^T$ 的分量，其中 Y_0 为 y_0 的地址，Y_1 为 y_1 的地址，如此等等。

4）对于所有 k¬ 0 和 j¬1，映射 $F_{k,j}$, $G_{k,j}$, $Sign_{k,j}$ 定义如下。令 $l = l(k, j)$, $m=l(k+1, 2j)$, $h=l(k, j+1)-1$。对于每一整数 v¬ 0 定义：

$Sign_{kj}(v) = 1 - 2\cdot\lfloor (v \bmod 2^m)/2^{m-1} \rfloor$

$F_{kj}(v)= 2^m\cdot\lfloor v/2^m \rfloor$

$v \bmod 2^m$ 如果 $Sign_{kj}(v) = 1$

$G_{kj}(v)= 2^m - 2^l - (v \bmod 2^m)$ 如果 $Sign_{kj}(v) = -1$

编码：

1．**初始化**：对每一从 0 到 $2^{r-1}+r-2$ 的地址填入 0。

2．**第一阶段**：对于每一从 0 到 n-1 的 j 执行

把 $Sign(w_j)\cdot x_j$ 加到地址 $\pi(h(w_j))$

3．**主要部分**：

对于 k 从 0 到 $\lceil \log(r) \rceil - 1$ 执行

对于 j 从 1 到 2^k 执行

如果 $l(k, j+1)-1 > l(k, j)$ 则

1）把驻留在（源）地址 $Y_{l(k,j)}$ 中的 v 值加入（目的）地址 $Y_{l(k+1,2j)}$。

2）对于 p 从 1 到 $2^{l(k,j+1)-l(k,j)-1}-1$ 执行

令 $u = 2^{l(k,j)}\cdot p$ 和 对于源地址 u 执行

如果 $G_{k,j}(u)=u$ 则

把驻留在（源）地址 u 中的值加到（目的）地址 $Y_{1(k+1,2j)}$

否则如果 $F_{k,j}(u) = u$ 则

令存在（源）地址 u 中的值加到（目的）地址 $Y_1(k, j)$

否则如果 $G_{k,j}(u) = 0$ 令存在（源）地址 u 中的值乘上(-1)加到（目的）地址 $Y_1(k, j)$ 和也令存在地址 u 中的值加到地址 $F_{k,j}(u)$。

否则把与 $Sign_{k,j}(u)$相乘的驻留在源地址 u 中的值加入（目的）地址 $G_{k,j}(u)$ 中的值并且也把该值（地址 u 的值）加入地址 $F_{k,j}(u)$中的值。

4. 获得输出：

现在对于所有 $0 \cdot i < r$ 每一地址 Y_i 含有 y_i 的值。

复杂度.

i)用上述实现在 U-设置中，基本步骤是从存储器中一处（称作源）读取数字，乘以符号 1 或-1 并加到存储器中另一处（称作目的）的位置。

ii)复杂度可用下列项公式化。对于 $0 \cdot k \cdot \lceil \log(r) \rceil - 1$ 并且对于 $1 \cdot j \cdot 2^k$ ，令：

$$u_{kj} = 2^{l(kj+1)-l(kj)-1}$$

$$u_k = \sum_{1 \cdot j \cdot 2^k} u_{kj} = \sum_{1 \cdot j \cdot 2^k} 2^{l(kj+1)-l(kj)-1}$$

对于 $k = \lceil \log(r) \rceil - 1$ 并且对于 $1 \cdot j \cdot 2^k$ ，令：

$$u_{kj} = \lfloor 2^{l(kj+1)-l(kj)-1} \rfloor,$$

则：

$$u_k = \sum_{1 \cdot j \cdot 2^k} u_{kj} = r$$

iii) 基于上述编码的器件将需要下列数目的基本步骤以完成上述 A·x 计算：

（1）第一阶段需要 n 个基本步骤。

（2）对于所有 $0 \cdot k \cdot \lceil \log(r) \rceil - 1$ 和 $1 \cdot j \cdot 2^k$

$$2 \cdot u_{kj} - u_{k+1,2j} - u_{k+1,2j-1} + 1$$

基本步骤在(k, j)步完成。

（3）因此对于所有 $0 \cdot k \cdot \lceil \log(r) \rceil - 1$，主要部分的 k-迭代需要下列数目的基本步骤：

$$2^k + 2 \cdot u_k - u_{k+1}$$

（4）将需要下列数目的基本步骤以完成上述 A·x 计算的主要部分：

$$C^u_r \odot 2u_0 + u_1 + u_2 + \ldots\ldots + u_{\lceil \log(r) \rceil - 1} + 2^{\lceil \log(r) \rceil} - 1 - r.$$

此数只取决于 r。

（5）整个 A·x 计算需要总共：

$$C^{u}_{n,r}\copyright\ n+C^{u}_{r}$$

个基本步骤。

图表 2：U-矩阵

图 2 示意地阐明更新一种器件使用的存储器内容的操作，该器件执行上述编码的主要部分，这样在较佳的模式下利用包括 4 行 U-二进制矩阵实现发明的 U-二进制方面。在各次迭代中各矩阵的列用存储器中一个二进制地址代表，其中列的（-1）分量对应于地址中的 1，而列的 1-分量对应于地址中的 0。应用一种二进制解码，其中底部分量（在水平表达时最右侧分量）为 MSB，而顶部分量（在水平表达时最左侧分量）为 LSB。特别地址 $Y_0 = 0$， $Y_1 = 1$，$Y_2 = 9$， $Y_3 = 10$，在过程结束时含有输出向量 $y = (y_0, \ldots\ldots, y_3)^T$ 的分量，其中 Y_0 为 y0 的地址，Y_1 为 y1 的地址，如此等等。箭头表示取一个地址内容，乘以符号 1 或-1，并且将其送往另一地址并相加的行为。单箭头表示符号为 1 而双箭头表示符号为-1。这是按照迭代顺序和参加各次迭代的地址（递增）顺序（如上述编码所示）而完成。

例 3：U_1-矩阵

以下步骤序列描述在变换矩阵中的元素属于 $U_1\copyright\{1, -1, j, -j\}$ 集合情况中 GEM 方法的实现。这是在应用中经常出现的 GEM 的子情况之一。数据包括一个 r×n U1-矩阵 $A = (a_{ij}: 0\cdot i<r, 0\cdot j<n)$ 和一个输入复数向量 $x=(x_0, \ldots\ldots, x_{n-1})$。矩阵的列用 $w_0, \ldots\ldots, w_{n-1}$ 表示。该步骤序列计算

$y = (y_0, \ldots, y_{n-1})^T=A\cdot x$。在每一次迭代中各位置含有取决于向量 x 的实数或复数。分配的读写存储器包含 $4^{r-1}+r-1$ 个地址，其标记从 0 到 $4^{r-1}+r-2$。此例子使用先前例子的定义和下列定义：

定义.

1）在集合 U1 和集合{0, 1, 2, 3}之间定义对应关系（和反对应关系）：

$1' = 0,\quad 0^* = 1$

$(-1)' = 1,\quad 1^* = -1,$

$j' = 2,\quad 2^* = j,$

$(-j)' = 3,\quad 3^* = -j$

从而对于 r-维 U_1-向量，$u = (u_0, \ldots\ldots, u_{r-1})$，定义：

$\square(u)=\sum_{0\le j<r} 4^j u'_j.$

2)对于 r-维 U_1-向量，$u=(u_0, \ldots\ldots, u_{r-1})$，定义：

$$Sign(u)=u_{r-1}$$

$$g(u) = (u_0\cdot(u_{r-1})^{-1}, \ldots\ldots\ldots\ldots\ldots\ldots, u_{r-1}\cdot(u_{r-1})^{-1}).$$

3) 定义 $Y_0=0$，和对于所有 $1\cdot m\cdot r$： $Ym = 4^{r-1}+m-1$。这些将是输出向量分量的地址。

4）对于所有 $k \rightarrow 0$ 和 $j \rightarrow 1$，映射 $F_{k,j}, G_{k,j}$ 定义如下。令 $l=l(k, j)$，$m=l(k+1, 2j)$，$h=l(k, j+1)-1$。取整数 $v \rightarrow 0$，在 4-基础上由下式代表： $v = \Sigma_{0\leqslant j<r-2} 4^j\cdot v_j$ 并定义：

$$Sign_{kj}(v) = (v_{m-1})^* = (\lfloor (v \bmod 4^m)/4^{m-1} \rfloor)^*$$

$$F_{kj}(v) = 4^m\cdot\lfloor v/4^m \rfloor$$

$$G_{kj}(v) = \sum_{l\leq j\cdot m-1} 4^l\cdot((v_l)^*\cdot((v_{m-1})^*)^{-1})'$$

编码.

1. 初始化：对每一从 0 到 $4^{r-1}+r-2$ 的地址填入 0。

2. 第一阶段：对于每一 j 从 0 到 n-1

把 $Sign(w_j)\cdot x_j$ 加到地址 $\square(g(w_j))$

3. 主要部分：

对于 k 从 0 到 $\lceil \log(r) \rceil - 1$ 执行

对于 j 从 1 到 2^k 执行

如果 $l(k, j+1)-1 > l(k, j)$ 则

1) 把驻留在（源）地址 $Y_{l(k,j)}$ 中的值加到（目的）地址 $Y_{l(k+1,2j)}$。

2）对于从 1 到 $4^{l(k,j+1)-l(k,j)-1} - 1$ 的 p 执行

令 $u = 4^{l(k,j)}\cdot p$ 并且对于源地址 u 执行：

如果 $G_{k,j}(u) = u$ 则

把驻留在（源）地址 u 中的值加入（目的）地址 $Y_{l(k+1,2j)}$ 中的值

否则如果 $F_{k,j}(u) = u$ 则

把驻留在（源）地址 u 中的值加入（目的）地址 $Y_{l(k,j)}$ 中的值

否则如果 $G_{k,j}(u) = 0$ 令存在（源）地址 u 中的值乘上 $Sign_{k,j}(u)$ 加到（目的）地址 $Y_{l(k,j)}$ 和也令存在地址 u 中的值加到地址 $F_{k,j}(u)$。

否则令存在源地址 u 中的值乘上 $Sign_{k,j}(u)$ 加到（目的）地址 $G_{k,j}$ 和也令存在地址 u 中的此值加到地址 $F_{k,j}(u)$。

4. 获得输出：

现在对于所有 0·i<r 每一地址 Y_i 含有 y_i 的值。

复杂度.

i)用上述实现在 U_1-设置中基本步骤意味着从存储器中一个存储单元（称作源）读取数字，乘以符号 1 或-1 或 j 或-j 并把结果加到存储器中另一个存储单元（称作目的）中的复数。

ii)复杂度可用下列项公式化。对于 $0\cdot k\cdot\lceil\log(r)\rceil-1$ 和 $1\cdot j\cdot 2^k$

令：

$$w_{k,j}=4^{l(k,j+1)-l(k,j)-1}$$

$$w_k=\sum_{1\cdot j\cdot 2^k} w_{kj} = \sum_{1\cdot j\cdot 2^k} 4^{l(k,j+1)-l(k,j)-1}$$

对于 $k=\lceil\log(r)\rceil-1$ 并且对于 $1\cdot j\cdot 2^k$，令：

$$w_{kj}=\lfloor 4^{l(k,j+1)-l(k,j)-1}\rfloor$$

$$w_k=\sum_{1\cdot j\cdot 2^k} w_{kj} = r.$$

iii) 基于上述编码的器件将需要下列数目的基本步骤以完成上述 A·x 计算：

（1）第一阶段需要 n 个基本步骤。

（2）对于所有 $0\cdot k\cdot\lceil\log(r)\rceil-1$ 和 $1\cdot j\cdot 2^k$ 总共

$$2\cdot w_{kj} - w_{k+1,2j} - w_{k+1,2j-1} + 1$$

基本步骤在(k, j)步完成。

（3）因此对于所有 $0\cdot k\cdot\lceil\log(r)\rceil-1$，主要部分的 k-迭代需要下列数目的基本步骤：

$$2^k + 2\cdot w_k - w_{k+1}$$

（4）将需要下列数目的基本步骤以完成上述 A·x 计算的主要部分：

$$C^{U_1}_r \text{ © } + 2w_0 + w_1 + w_2 + \ldots\ldots + w_{\lceil\log(r)\rceil-1} + 2^{\lceil\log(r)\rceil} - 1 - r.$$

此数只取决于 r。

（5）整个 A·x 计算需要总共：

$$C^{U_1}_{n,r} \text{ © } n + C^{U_1}_r$$

个基本步骤

例 4：TOPLITZ U-矩阵

下列步骤序列描述一种发明的具有 U-系数的 TOPLITZ 矩阵方面。数据包括一

个 U-序列 $t_0, \ldots\ldots, t_{n-1}$ 和一个输入实数或复数向量 $x = (x_0, \ldots\ldots, x_{n+r-2})$。从 U-序列形成一个 rx(n+r-1) TOPLITZ- 矩阵 A ©(a_{ij}©t_{i-j}:0·i<r, 0·j·n+r-2), 其中对于所有 k<0 或 k¬ n, t_k©0。这些步骤计算 $y = (y_0, \ldots., y_{n-1})$=A·x 的乘积。只有第一阶段与 U-矩阵例子有区别，因此只需引入这一阶段。所有在实现前面的例子中列出的定义在此均可应用。

编码.

1．初始化：对每一从 0 到 2^{r-1}+r-2 的地址填入 0。

2．第一阶段：

1）对于 j 从 0 到 r-2 把(1/2)·$t_{n-r+j+1}$·x_j 加到地址:

$\pi(h(t_j, t_{j-1}, \ldots\ldots\ldots\ldots t_1, t_0, t_{n-1}, t_{n-2}, \ldots\ldots\ldots\ldots, t_{n-r+j+1}))$

并且也把-(1/2)·$t_{n-r+j+1}$·x_j 加入地址:

$\pi(h(t_j, t_{j-1}, \ldots\ldots\ldots\ldots t_1, t_0, -t_{n-1}, -t_{n-2}, \ldots\ldots\ldots\ldots, -t_{n-r+j+1}))$

2）对于 j 从 r-1 到 n-2 ，把 t_{j-r+1}·x_j 加入地址:

$\pi(h(t_j, t_{j-1}, \ldots\ldots\ldots\ldots\ldots, t_{j-r+1}))$

3）对于 j 从 n 到 n+r-2 把(1/2) ·t_{j-r+1}·x_j 加入地址:

$\pi(h(t_{j-n}, t_{j-n-1}, \ldots\ldots\ldots t_1, t_0, t_{n-1}, t_{n-2}, \ldots\ldots\ldots\ldots\ldots, t_{j-r+1}))$

并且也把(1/2)·t_{j-r+1}·x_j 加入地址:

$\pi(h(t_{j-n}, t_{j-n-1}, \ldots\ldots\ldots, t_1, -t_0, -t_{n-1}, -t_{n-2}, \ldots\ldots\ldots, -t_{j-r+1}))$

3．主要部分：

算法现在进行与 U-矩阵实现算法主要部分相同，并且输出贮存在相同地址。

复杂度.

i)在上述 TOPLITZ 实现中的基本步骤与 U-实现相同，即从存储器中一处（称作源）读取数字，乘以符号 1 或-1 并加到存储器中另一处（称作目的）的位置。

ii）基于上述编码的器件将需要下列数目的基本步骤以完成上述 A·x 计算：

（1）对于第一阶段需要下列

$$2\cdot(r-1) + n - r + 1 + 2\cdot(r-1) = n + 3r - 3$$

基本步骤。

（2）TOPLITZ 矩阵的主要部分计算 A·x 通过 U-编码主要部分的实现 r 行完成。因此需要 C_r^u 个基本步骤。

（3）整个 A·x 的 TOPLITZ 计算需要总共：

$$C^T_{n,r} \text{©} \; n + 3r - 3 \; + C^u_r$$

个基本步骤

图表 3：TOPLITZ-矩阵

图 3 示意地阐明输送进入的数据到器件所使用的适当储存器位置的操作，该器件利用包括 4 行的 TOPLITZ 矩阵执行上述编码的初始部分。除去这一初始部分，其它各方面均与 U-矩阵实现和器件完全相同，并且其描述可以同样在此应用。这里同样一个箭头代表符号为 1，而双箭头代表符号为-1。

例 5：TOPLITZ U_1-矩阵

下列发明的优选实施例为具有 U_1-系数的 TOPLITZ 矩阵方面的实现。数据包括一个 U_1-序列 $t_0, \ldots\ldots, t_{n-1}$ 和一个输入复数向量 $x = (x_0, \ldots\ldots, x_{n+r-2})$。从 U_1-序列形成一个 rx(n+r-1) TOPLITZ-矩阵

A ©(a_{ij}©t_{ij}:0·4<r, 0·j·n+r-2)，其中对于所有 k<0 或 k¬ n, t_k ©0。下列步骤序列计算 $y = (y_0, \ldots, y_{n-1}) = A \cdot x$ 的乘积。只有第一阶段与 U_1-矩阵例子有区别，因此只需描述这一阶段。所有在实现前面的例子中列出的定义在此均可应用。

编码：

1．初始化：对每一从 0 到 $4^{r-1}+r-2$ 的地址填入 0。

2．第一阶段：

1）对于 j 从 0 到 r-2 把 $(1/2)\cdot t_{n-r+j+1}\cdot x_j$ 加到地址：

□(g($t_j, t_{j-1}, \ldots\ldots, t_1, t_0, t_{n-1}, t_{n-2}, \ldots\ldots, t_{n-r+j+1}$))

并且也把 $-(1/2)\cdot t_{n-r+j+1}\cdot x_j$ 加入地址：

□(g($t_j, t_{j-1}, \ldots\ldots, t_1, t_0, -t_{n-1}, -t_{n-2}, \ldots\ldots, -t_{n-r+j+1}$))

2）对于 j 从 r-1 到 n-1 把 $t_{j-r+1}\cdot x_j$ 加入地址：

□(g($t_j, t_{j-1}, \ldots\ldots, t_{j-r+1}$))

3）对于 j 从 n 到 n+r-2 把 $(1/2)\cdot t_{j-r+1}\cdot x_j$ 加入地址：

□(g($t_{j-n}, t_{j-n-1}, \ldots\ldots, t_1, t_0, t_{n-1}, t_{n-2}, \ldots\ldots, t_{j-r+1}$))

并且也把 $(1/2)\cdot t_{j-r+1}\cdot x_j$ 加入地址：

□(g($t_{j-n}, t_{j-n-1}, \ldots\ldots, t_1, -t_0, -t_{n-1}, -t_{n-2}, \ldots\ldots, -t_{j-r+1}$))

并且也把 $-(1/2)\cdot t_j\cdot x_j$ 加入地址：

□(g($t_{j-n}, t_{j-n-1}, \ldots\ldots, t_1, -t_0, -t_n, -t_{n-1}, \ldots\ldots, -t_{j-r+2}$))

3．主要部分：

算法现在进行与U1-矩阵实现算法主要部分相同，并且数据接收方法相同。

复杂度.

i)在上述TOPLITZ实现中的基本步骤与U_1-实现相同，即从存储器中一处（称作源）读取数字，乘以符号1或-1或j或-j并把结果加到存储器中另一处（称作目的）复数的位置。

ii) 基于上述编码的器件将需要下列数目的基本步骤以完成上述A·x计算：

（1）对于第一阶段需要下列

$$2\cdot(r-1)+n-r+1+2\cdot(r-1)=n+3r-3$$

个基本步骤。

(2)TOPLITZ矩阵的主要部分计算A·x通过U_1-编码主要部分的实现r行完成。因此需要C_r^{U1}基本步骤。

（3）整个A·x 的TOPLITZ计算需要总共：

$$C^{T_1}{}_{n,r} \copyright\ n+3r-3 \quad + C^{U_1}{}_{r}$$

个基本步骤

例6：实数矩阵、二进制 0-1 表示

下列发明的优选实施例是发明的具有矩阵元素用二进制0-1-表示法的实数矩阵方面的实现。数据包括一个r×n 具有非零元素的U_1-矩阵 $A=(a_{ij}: 0\cdot i<r, 0\cdot j<n)$ 和一个输入复数向量 $x=(x_0, \ldots\ldots, x_{n-1})$。算法计算 $y=(y_0, \ldots, y_{n-1})=A\cdot x$。

假定矩阵的元素由下列0-1-二进制表示法给出，对于所有0·i<r, 0·j<n

$$a_{ij}=\square_{-m_2\le k\le m_1} t_{ijk}\bullet 2^k$$

式中：对于所有 $-m_2\cdot k\cdot m_1$， $t_{ijk}\in\{0,1\}$

只有第一阶段与0-1-矩阵实现有区别，因此仅该阶段需要描述。所有在此期间0-1-矩阵实现部分列出的定义在此均可应用。此外还定义下列r维的{0,1}-向量，对于所有0·j<n，$-m_2\cdot k\cdot m_1$：

$$v_{jk}=(t_{1jk}, t_{2jk}\ldots\ldots\ldots\ldots, t_{rjk})$$

1. **初始化：**对每一从1到 2^r-1 的地址填入0。

2. **第一阶段：**对于计数器j从0 到n-1 执行

对于k 从 $-m_2$ 到 m_1 执行

在地址□(v_{jk}) 的值上加 $2^k\cdot x_j$

3. 主要部分：进行过程如同（0，1）-矩阵实现对于 r 行算法主要部分，而在计算结束时输出储存在同样的地址中。

复杂度.

i)在上述实现中的基本步骤与 0-1 二进制-实现相同，即从存储器中一处（称作源）读取数字，并把结果加到存储器中另一处（称作目的）数的位置。

ii) 基于上述编码的器件将需要下列数目的基本步骤以完成 A·x 计算：

（1）对于第一阶段需要下列

$$(m_1 + m_2 + 1)\cdot n$$

个基本步骤。

（2)编码的主要部分计算通过上述实现具有 r 行的 0-1-二进制编码主要部分完成。因此需要 $C_r^{0,1}$ 基本步骤。

（3）整个 A·x 的计算需要总共：

$$C^{Re\text{-}0\text{-}1} \copyright\ (m_1 + m_2 + 1)\cdot n + C^{0,1}{}_r$$

个基本步骤

例 7：实数矩阵、二进制 U-表示

下列发明的优选实施例是发明的具有矩阵元素用二进制 U-表示法的实数矩阵方面的实现。数据包括一个 r×n 实数-矩阵 $A = (a_{ij}: 0\cdot i<r,\ 0\cdot j<n)$ 和一个输入实数或复数向量 $x=(x_0, \ldots\ldots, x_{n-1})$。算法计算 $y = (y_0, \ldots, y_{n-1}) = A\cdot x$。

假定矩阵的元素由下列 U-表示法给出，对于所有 0·i<r, 0·j<n

$$a_{ij} = \square_{-m_2-1 \le k \le m_1-1}\ t_{ijk}\cdot 2^k + t_{ijm_1}\cdot(2^{m_1} - 2^{-m_2-1})$$

式中对于所有 $-m_2-1\cdot k\cdot m_1$， $t_{ijk}\in U$ 。

此表示法的存在已经在发明的实数矩阵方面涉及。只有第一阶段与 U-矩阵的实现不同，因此仅该阶段将予描述。所有 U-矩阵实现部分列出的定义在此均可应用。此外还定义下列 U-向量，

对于所有 $0\cdot j<n$， $-m_2-1\cdot k\cdot m_1$：

$$u_{jk} = (t_{1jk}, t_{2jk}\ldots\ldots\ldots\ldots, t_{rjk})$$

1. 初始化：对每一从 0 到 $2^{r-1}+r-2$ 的地址填入 0。

2. 第一阶段：

对于 j 从 0 到 n-1 执行

对于 k 从 $-m_2-1$ 到 m_1-1 执行

把 $2^k \cdot Sign(u_{jk}) \cdot x_j$ 加入地址 $\pi(h(u_{jk}))$

对于 $k= m_1$ 执行

把 $(2^{m1} - 2^{-m2-1}) \cdot Sign(u_{jk}) \cdot x_j$ 加入地址 $\pi(h(u_{jk}))$

3. 主要部分：进行过程如同（0,1）-矩阵实现对于 r 行算法主要部分，而在计算结束时输出储存在同样的地址中。

复杂度.

i)在上述实现中的基本步骤与 U-实现相同，即从存储器中一处（称作源）读取数字，并乘以符号 1 或-1，再把结果加到存储器中另一处（称作目的）数的位置。

ii) 基于上述编码的器件将需要下列数目的基本步骤以完成 A·x 计算：

（1）对于第一阶段需要下列

$$(m_1 + m_2 + 2) \cdot n$$

个基本步骤。

（2）编码的主要部分计算通过上述实现具有 r 行的 U-编码主要部分完成。因此需要 C_r^u 基本步骤。

（3）整个 A·x 的计算需要总共：

$$C^{Re\text{-}U} \copyright (m_1 + m_2 + 1) \cdot n + C_r^u$$

个基本步骤。

图 4 是按照发明的优选实施例，用以减少的加法次数执行线性变换的示例性装置框图。装置 500 包括具有两个输入端及一个输出端的乘法器 10，用来选择其输入之一传送到其输出的多路复用器 9（MUX），具有两个输入及一个输出的加法器 11，随后是双端随机存取存储器（DPRAM）13，有两条地址总线，即“add_a”和“add_b”和两个输出端，即“data_a”和“data_b”。MUX 的活动是由地址发生器 501 进行控制，该发生器也启用 DPRAM 13 中的存储地址的存取。地址发生器的活动是由计数器 3 进行控制。

乘法器 10 的输出端被连接到 MUX 9 的一个输入端“C”。MUX 9 的输出连接到加法器 11 的输入端“A”。加法器 11 的输出连接到 DPRAM13 的输入。DPRAM 的一个输出“data_a”连接到加法器 11 的输入。DPRAM 13 的一个输出端“data_b”被连接到乘法器 12 的输入端“E”。乘法器 12 的另一输入端“F”连接到地址发生器 501 的输出端“sign”。乘法器 12 的输出端连接到 MUX 9 的另一输入端“D”。计

数器 3 连接到地址发生器 501 的两个输入端。发生器的 501 输出端“G”连接到 MUX 9 的控制输入端“S”。发生器 501 的输出端“J”连接到 DPRAM 13 的第一地址输入端“add_a”。发生器 501 的另一输出端“I”连接到 DPRAM 13 的第二地址输入端“add_b”。变换矩阵可以储存在可选的 RAM/ROM 1 中，它供给地址发生器 501 以系列代码（矩阵中一行比特）$D_0, D_1, \ldots\ldots, D_s$，并供给乘法器 10 以最高有效位 D_0。输入向量可以储存在另一可选的 RAM/ROM 2 中，它供给乘法器 10 输入信号的采样。或者，如果输入向量的元素和它们在变换矩阵中的相应元素同步地被提供给装置 500，则存储器 1 和 2 可被消除。例如，这些元素可以由 ADC 或序列发生器提供。所有装置 500 的分量由公共时钟通过“clock_in”输入端来控制。

相同的 U 矩阵（包括 n 行）用相同的输入集合$[x_1\ x_2\ \ldots\ldots x_n]$来表示相同的不同编码 $D_0, D_1, \ldots\ldots, D_s$，下面说明该装置的操作。装置 500 的操作可以被分成两个阶段，表示为“阶段 1”，在此时期内输入数据（即采样$[x_1\ x_2\ \ldots\ldots x_n]$）被接收，并且各分量和它的相应变换矩阵元素的乘积被计算并存储在 DPRAM 13 中；第二阶段中，称为“阶段 2”，接收的数据被处理，而锁住进一步来自加法器 9 的输入数据。计数器 3 计算操作的累计操作次数且计数（计数器的输出）被用来区别两个阶段。“阶段 1”中的操作数是输入向量的长度 n。超过该数的操作与“2”相关。

按照本发明优选实施例，地址发生器 501 包括比较器 4，它被连接到计数器 3 的输出端。比较器读取计数器 3 的当前输出，将其与输入向量的长度 n 相比较，并提供相应的信号，表示当前操作是否已在“阶段 1”或“阶段 2”中被执行。该信号用来控制 MUX 9 的输入端“S”，以便在输入 “C”及“D”之间切换。

地址发生器 501 也包括异步存储器 5（例如，ROM），它存储编程值，并被用作查找表（LUT），用于通过输入地址“add_a”和“add_b”来确定 DPRAM 13 中的地址，以便处理它们的内容。LUT 的大小为$(C-n)\times(1+2r)$，并且它的内容包括三个字段，“源”字段，“符号”字段和“目的”字段。对于各次操作（即，各时钟周期，由计数器 3 计数），有三个相应的源符号和目的值。源字段包括与被拆分（分割）的变换矩阵的每列相关的信息，以及与标准化特定的两部分相关的指示。源字段确定 DPRAM 13 上的输入“add_b”的值。符号字段表示在各分割行或子行中低分量。目的字段确定 DPRAM 13 上的输入“add_a”的值，按照该值选择相应地址的内容以供处理。

地址发生器也包括一组 s(s=r-1)个反相器，$6_1, 6_2\ldots\ldots, 6_s$，各相应地包括连接到一系列比特 $D_0, D_1, \ldots\ldots, D_s$ 的输入端。各反相器的输出连接到来自一组多路复用

器 $7_1, 7_2 \ldots\ldots, 7_s$ 的相应的 MUX 的一个输入端。这一系列比特 $D_0, D_1, \ldots\ldots, D_s$ 也被送入来自一组多路复用器 $7_1, 7_2 \ldots\ldots, 7_s$ 的相应的 MUX 的另一个输入端。来自一组多路复用器 $7_1, 7_2 \ldots\ldots, 7_s$ 的每个 MUX 的输出受到最高有效位 D_0 控制，以便使集合 $D_1, \ldots\ldots, D_s$ 的传送不变，或者被反相（即 $D'_1, \ldots\ldots, D'_s$ ）地送至 DPRAM 13 的输入端“add_a”。集合 $D_1, \ldots\ldots, D_s$（或 D'1，……，D's）被送入 s 个多路复用器的相应集合 $8_1, 8_2 \ldots\ldots, 8_s$，其中附加的多路复用器 8_r 由来自 LUT 的“add_a”的 MSB（第 r 个比特）送入。比较器 4 的输出使“add_a ”的选择从集合 $7_1, 7_2 \ldots\ldots, 7_s$ 或从 LUT 到达。输入“add_a ”（即目的）控制 DPROM 13 的第一输出“data_a ”，后者被送入加法器 11 的输入端“B”。从 LUT（即，源）取得的输入“add_b”控制 DPRAM 13 的第二输出“data_b”，后者被送入乘法器 12 的输入端“E”，乘法器 12 用从 LUT 析取的相应“符号”值乘以各“data_b”值并把乘积送入乘法器 12 的输入“D”。DPRAM 13 中的“写”操作是同步的，即，各单元的内容按照时钟速率被重写（例如当时钟信号出现时）。另一方面，DPRAM 13 中的“读”操作是异步的，即不考虑时钟，各输出在地址输入变化时被改变。

第一阶段处的操作：

在此阶段，计数器 3 开始计算第一码元时间（n 个时钟循环），在此期间执行第一阶段。在第一阶段中，MUX 9 使来自输入端“C”的数据流流入其输出端并且流入加法器 11 的输入“A”，而锁住输入端“D”。输入码元$[x_1\ x_2 \ldots\ldots x_n]$被提供给乘法器 10 的一个输入端。编码的 MSB D_0 被提供给乘法器 10 的另一输入端。在此阶段，由输入“add_a ”确定的目的（输出“data_a”）从 DPRAM 13 被析取并被加入输入向量的分量$[x_1\ x_2 \ldots\ldots x_n]$，再与 MSB D_0 相乘。已经计数当前码元时间的计数器 3 把指示提供给地址发生器比较器 4 和 ROM 5，告知当前码元时间已经结束，然后比较器 4 把 MUX 9 的输入端选择从输入端“C”切换到另一输入端“D”。同样，比较器 4 驱使多路复用器 $8_1, 8_2 \ldots\ldots, 8_s$ 从 LUT 选择数据，而不是从 RAM 1 选择数据。

第二阶段处的操作：

在此阶段，计数器 3 开始计数执行阶段 2 的下一码元时间。在阶段 2 中，MUX 9 使来自其输入端“D”的数据流流向其输出端，并流入加法器 11 的输入端“A”，而锁住输入端“C”。在此阶段，在每一时钟循环，DPRAM 13 中选定的地址通过表示源的“add_b”被访问。源数据从 DPRAM 13 的第二输出端“data_b”被送入乘法器 12 的输入端“E”，在乘法器 12 中它与从 LUT 中析取的相应“符号”值相乘。

该乘积被送入 MUX 9 的输入端“D”，从而出现在加法器 11 的输入“A”中。与此同时，当前目的值的内容出现在加法器 11 的输入“B”中。这两个值由加法器 11 相加，其结果存储在 DPRAM 13 中的同一地址处（对应于前一个目的值）。在该加法过程结束时，相应的 r 个变换点（y_i‘）被存储在 DPRAM 13 中地址号 0 以及地址号 2^{r-1} 到$(2^{r-1}+r-2)$的位置处。这一过程因此继续，在所有变换点（y_i‘）都被计算和存储在 DPRAM 13 中不同地址之后，一直到时钟指示阶段 2 被终止（按照预定的计数）。同时，当前码元时间也已被终止，且比较器 4 把 MUX 9 的输入选择从输入端“D”重新切换回另一输入端“C”，并且驱动多路复用器 $8_1, 8_2 \ldots\ldots, 8_s$ 从 RAM 1 选择数据，而不是从 LUT 选择数据，从而驱动装置 500 再次工作在阶段 1。

用来实现上述方法的装置的一个实施例在图 5 中示出。图 5 阐明如何实现 r×n 的 U-矩阵 A 与 n-维向量 X 的乘积。矩阵的表示包含 0 或 1，其中 0 对应于 1 而 1 对应于-1。

在该例中，向量是实数的且 v 个比特用于每个分量。在随后的建立中，矩阵 A 被存储在元件 1（RAM 或 ROM）中，而向量 X 被存储在元件 2（RAM 或 ROM）中。矩阵和向量可以从任何内部或外部设备产生，譬如存储器设备或同步源，如 ADC 或某些序列发生器等。

控制模块 1、2、3、13 的时钟使整个系统同步。模块 3 是从 0 计数到 C 的计数器，其中 C 的定义如下：

$$C = n + 2u_0 + u_1 + u_2 + \ldots\ldots + u_{\lceil \log(r) \rceil - 1} - r$$

$$u_k = \sum_{1 \le j \le 2^k} 2^{h(k,j)-l(k,j)}.$$

$$l(k,j) = \lceil (j-1) \cdot (r/2^k) \rceil$$

$$h(k,j) = \lceil j \cdot (r/2^k) \rceil - 1$$

在阶段 1 中，来自元件 1&2 的输入信号按照来自元件 3 的地址被插入元件 13，其中各地址包含 $\log_2 n$ 列的有效位。

此外，计数器用作大小为(C-n)×(1+r+r)的异步 ROM（元件 5）的地址发生器。计数器的所有比特进入元件 4，元件 4 是检查当前计数是否大于 n 的比较器。元件 5 包含三个字段：符号、源、目的。ROM 处的行序应该使阶段 2 的第一操作由计数器在 n 次循环后引发。

为语法的目的，我们将定义 s=r-1。数据总线从元件 1 的比特 D_1-D_s 分别进入元件 6_1-6_s（它们是反相器）。元件 71-7s 按照 D0 的状态在 D1-Ds 和 61-6s 之间进行选择。

(D0=0 => 7_1-7_s = D1-Ds, D0=1 => 71-7s = 61-6s).

元件 8_1-8_s 按照在元件 4（阶段 1_2n）中比较器输出端的状态在 7_1-7_s 和元件 5 的 $\log_2(n)$个 LSB 的输出（add_a=目的）之间进行选择。阶段 1_2n =0=>8_1-8_s = add_a, 阶段 1_2n=1=>8_1-8_s = 7_1-7_s。

元件 8_r 在元件 5 的“0”到 r 比特（add_a MSB）之间进行选择。阶段 1_2n=0=>8_r=add_a MSB, 阶段 1_2n=1=>8_r=0。来自 8_1-8_r 的输出被用作元件 13 的第一地址总线，该总线是大小为$(2^{r-1}+r-1)\times(V+\log_2 n)$的双端 RAM。该地址定义了 data_a 总线，它是来自元件 13 的输出，并且也是作为元件 13 输入的 data_in 总线的目的。

Add_b(元件 5 的输出(源字段))是元件 13 的第二地址总线(该地址控制作为元件 13 的输出的 data_b 总线)。

符号是从元件 13 输出的单个比特，它在元件 12(它是乘法器)中乘以 data_b。

作为关于元件 13 的输入的 data_in 总线从元件 11 到达，元件 11 是把 data_a 和元件 9 的输出相加在一起的加法器。data_in 以同步方式进入 add_a 目的单元，而 data_a 和 data_b 的读操作是异步的。

在元件 13 动作之前，它通过插入零被初始化。元件 9 在元件 10 和元件 12 的输出之间进行选择。元件 10 把来自元件 2 的数据总线与元件 1 的 LSB 相乘。在 C 次循环之后形成的向量被存储元件 13 中地址 0 以及地址 2^{r-1} 到$(2^{r-1}+r-2)$的位置处。

本发明的各种变化和修改对于本领域的技术人员来说是显而易见的。从而本发明可以以其它特定形式作为实施例，而无须背离本发明的精神或基本特性。详细实施例仅仅是说明性且非限制性的，因此本发明的范围由所附权利要求来表示，而非由上述说明表示。与权利要求相当的意义和范围中产生的变化在本发明的范围之内。

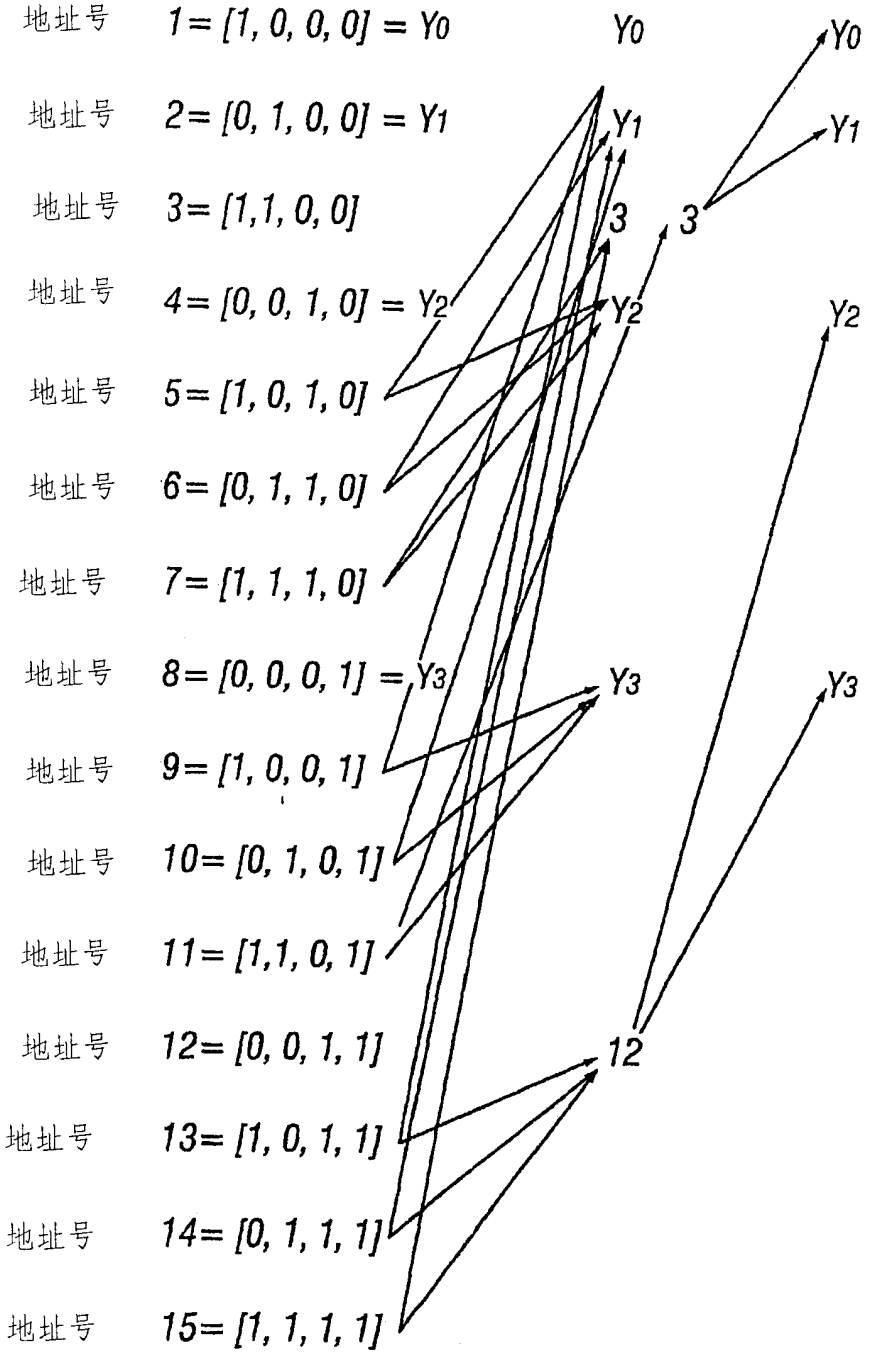

地址号 1= [1, 0, 0, 0] = Y0
地址号 2= [0, 1, 0, 0] = Y1
地址号 3= [1,1, 0, 0]
地址号 4= [0, 0, 1, 0] = Y2
地址号 5= [1, 0, 1, 0]
地址号 6= [0, 1, 1, 0]
地址号 7= [1, 1, 1, 0]
地址号 8= [0, 0, 0, 1] = Y3
地址号 9= [1, 0, 0, 1]
地址号 10= [0, 1, 0, 1]
地址号 11= [1,1, 0, 1]
地址号 12= [0, 0, 1, 1]
地址号 13= [1, 0, 1, 1]
地址号 14= [0, 1, 1, 1]
地址号 15= [1, 1, 1, 1]
Y0
Y1
3
Y2
Y3
12
Y0
Y1
3
Y2
Y3

图 1

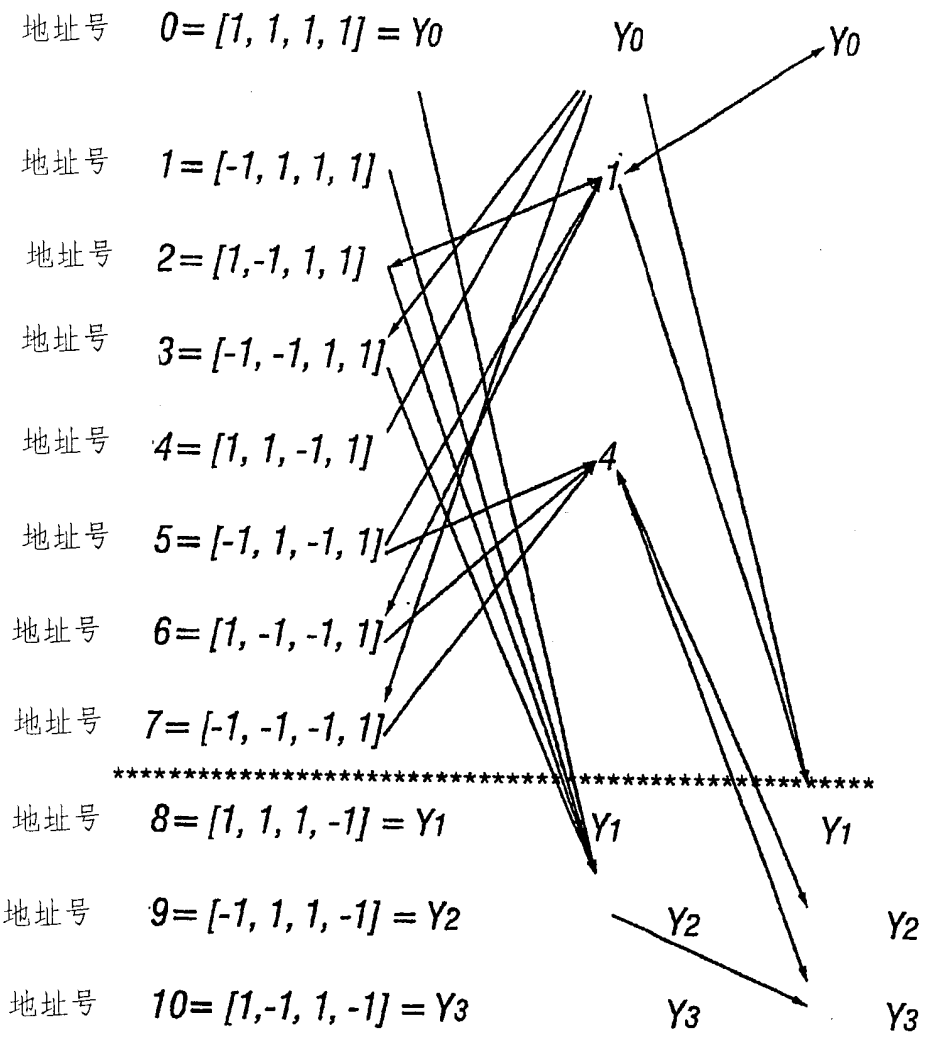

地址号 0= [1, 1, 1, 1] = Y0
地址号 1= [-1, 1, 1, 1]
地址号 2= [1,-1, 1, 1]
地址号 3= [-1, -1, 1, 1]
地址号 4= [1, 1, -1, 1]
地址号 5= [-1, 1, -1, 1]
地址号 6= [1, -1, -1, 1]
地址号 7= [-1, -1, -1, 1]
地址号 8= [1, 1, 1, -1] = Y1
地址号 9= [-1, 1, 1, -1] = Y2
地址号 10= [1,-1, 1, -1] = Y3
Y0
Y0
1
4
Y1
Y1
Y2
Y2
Y3
Y3

图 2

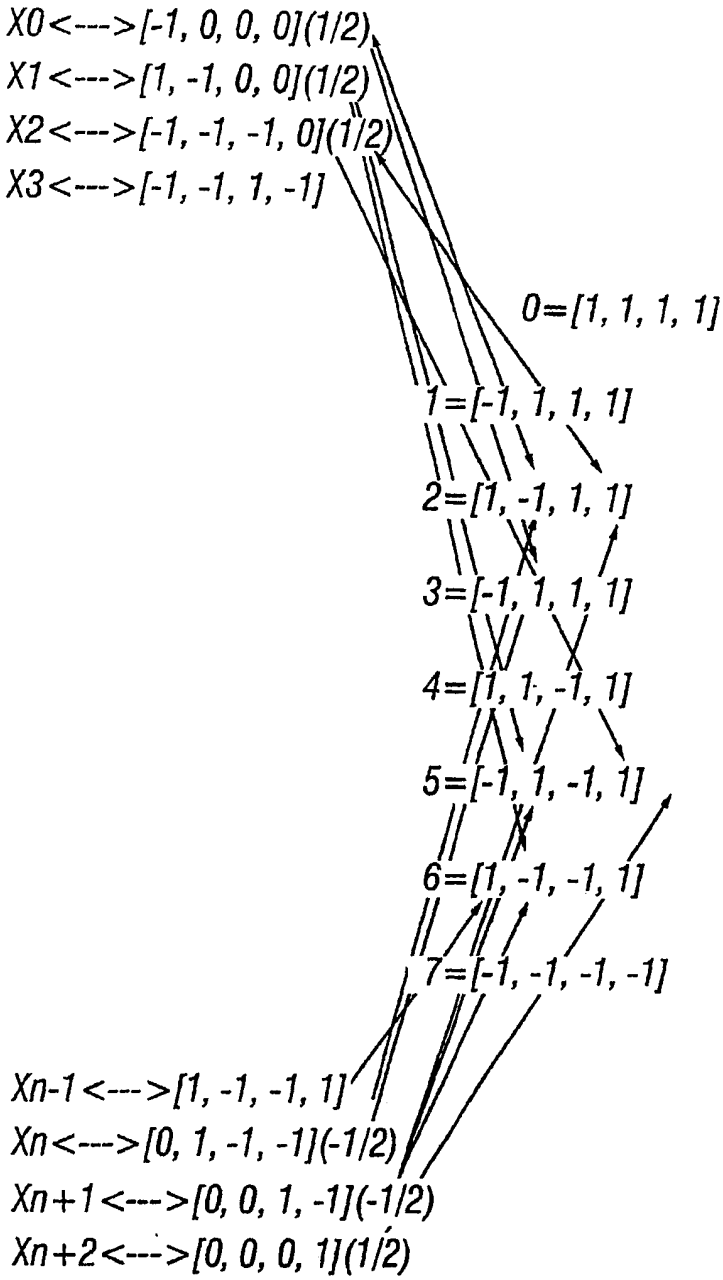

X0<--->[-1, 0, 0, 0](1/2)
X1<--->[1, -1, 0, 0](1/2)
X2<--->[-1, -1, -1, 0](1/2)
X3<--->[-1, -1, 1, -1]
0=[1, 1, 1, 1]
1=[-1, 1, 1, 1]
2=[1, -1, 1, 1]
3=[-1, 1, 1, 1]
4=[1, 1, -1, 1]
5=[-1, 1, -1, 1]
6=[1, -1, -1, 1]
7=[-1, -1, -1, -1]
Xn-1<--->[1, -1, -1, 1]
Xn<--->[0, 1, -1, -1](-1/2)
Xn+1<--->[0, 0, 1, -1](-1/2)
Xn+2<--->[0, 0, 0, 1](1/2)

图 3

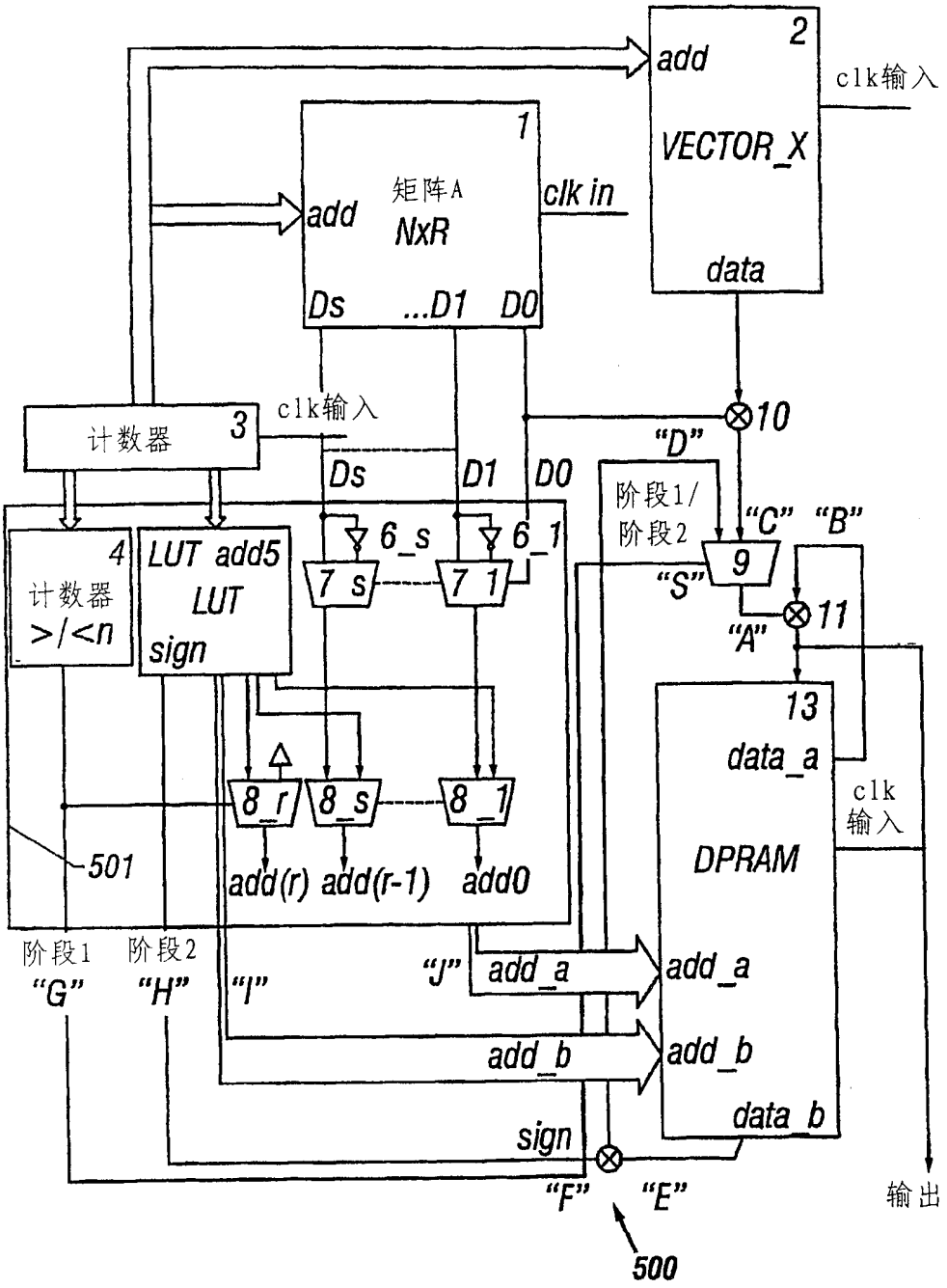

2
add
clk输入
VECTOR_X
data
1
矩阵A
NxR
add
clk in
Ds ...D1 D0
clk输入
计数器 3
10
"D"
Ds D1 D0
阶段1/
阶段2
"C" "B"
6_s 6_1
7_s 7_1
9
"S"
4
计数器
>/<n
LUT add5
LUT
sign
11
"A"
13
data_a
clk
输入
8_r 8_s 8_1
DPRAM
501
add(r) add(r-1) add0
阶段1 阶段2
"G" "H" "I"
"J" add_a
add_a
add_b
add_b
data_b
sign
"F" "E"
输出
500

图 4

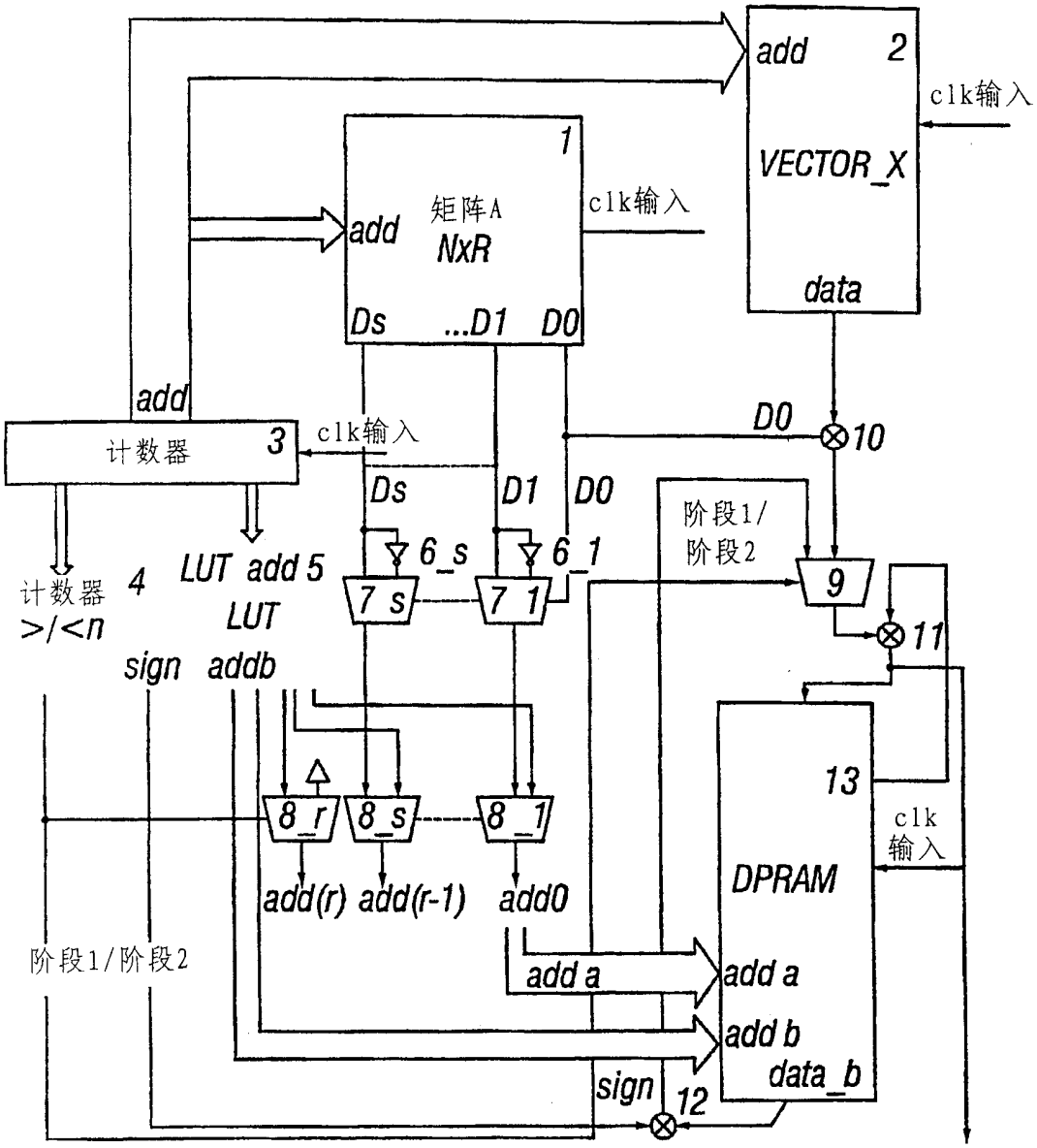

add
2
clk输入
VECTOR_X
data
1
矩阵A
NxR
add
clk输入
Ds
...D1
D0
add
计数器
3
clk输入
D0
10
Ds
D1
D0
阶段1/
阶段2
6_s
6_1
计数器
>/<n
4
LUT add 5
LUT
7_s
7_1
9
11
sign
addb
13
8_r
8_s
8_1
clk
输入
DPRAM
add(r)
add(r-1)
add0
阶段1/阶段2
add a
add a
add b
data_b
sign
12

图 5