



[12] 发明专利申请公开说明书

[21] 申请号 03815012.3

[43] 公开日 2005 年 8 月 31 日

[11] 公开号 CN 1663208A

[22] 申请日 2003.6.27 [21] 申请号 03815012.3

[30] 优先权

[32] 2002.6.27 [33] KR [31] 10-2002-0036216

[32] 2003.6.27 [33] KR [31] 10-2003-0042357

[86] 国际申请 PCT/KR2003/001268 2003.6.27

[87] 国际公布 WO2004/004265 英 2004.1.8

[85] 进入国家阶段日期 2004.12.27

[71] 申请人 三星电子株式会社

地址 韩国京畿道

共同申请人 亚洲大学校产学协力团

[72] 发明人 鲜于明勋

[74] 专利代理机构 北京市柳沈律师事务所

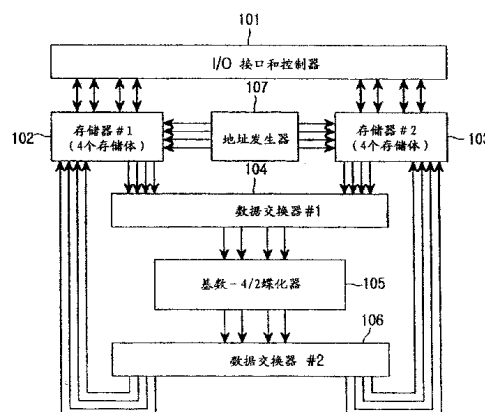
代理人 郭定辉 黄小临

权利要求书 2 页 说明书 13 页 附图 16 页

[54] 发明名称 利用混合基数快速付里叶变换的调制设备

[57] 摘要

本发明公开了作为 OFDM(正交频分多路复用)或 DMT(离散多频声)MODEM 的核心模块的 FFT(快速付里叶变换)处理器。FFT 处理器通过将原地算法(in-place algorithm)应用于混合基数多存储体存储器,同时进行顺序输入和输出,从而只利用含有 4 个存储体的 $2N$ -字存储器就可以实现连续处理。FFT 处理器在满足高速计算要求的同时,使其复杂度降到最低。



1. 一种利用混合基数快速付里叶变换 (FFT) 的调制设备, 包括:

5 输入/输出接口, 用于从用于输入/输出和 FFT 计算的存储器当中选择用于输入/输出的存储器, 选择所选存储器的 4 个存储体之一, 并且对所选存储体进行输入和输出;

两个 N-字存储器, 每一个都含有用于针对输入/输出接口的输入/输出和 FFT 计算的 4 个存储体;

10 第一数据交换器, 用于从用于针对输入/输出接口的输入/输出和 FFT 计算的存储器当中选择用于 FFT 计算的存储器, 并且将指定给每个蝶式输入/输出的存储体与用于原地计算的蝶式计算电路的 4 个输入端相连接;

蝶式器, 用于利用一个电路, 根据第一数据交换器提供的输出, 进行基数-4 蝶化和基数-2 蝶化, 并且形成对称反向输出;

15 第二数据交换器, 用于从用于针对输入/输出接口的输入/输出和 FFT 计算的存储器当中选择用于 FFT 计算的存储器, 并将指定给每个蝶式输入/输出的存储体与用于原地计算的蝶式计算电路的 4 个输出端相连接; 和

地址发生器, 用于生成在多存储体存储器结构中进行原地计算的存储体索引和地址。

2. 根据权利要求 1 所述的调制设备, 其中, 蝶式器是能够通过交换基数-4 蝶式器和两个基数-2 蝶式器的输出形成对称反向输出, 同时进行顺序输入和输出的基数-4/2 蝶式计算电路, 从而能够利用每一个包括 4 个存储体的两个 N-字存储器进行连续处理。

3. 根据权利要求 1 所述的调制设备, 其中, 通过修改基于基数-4 算法多存储体存储器结构的原地算法确定原地算法。

25 4. 根据权利要求 1 所述的调制设备, 其中, 原地算法在生成存储体索引的过程中的 2^n -点计算期间, 在模-2 相加中单独包括第 $(n-2)$ 位。

5. 一种利用混合基数快速付里叶变换 (FFT) 的调制设备, 包括:

每一个都含有 4 个存储体的两个存储器, 用于写入输入码元或 FFT 计算码元;

30 蝶式器, 用于按照从存储器输出的码元的个数, 以基数-2 模式或基数-4 模式进行蝶式计算, 并对称反向地输出计算值;

第一数据交换器，用于从存储器之一的每个存储体中读取一个码元，并将读取的码元输出到蝶式器；

第二数据交换器，用于匹配从蝶式器输出的计算码元，以便将码元写入与从第一数据交换器中读取码元的地址相同的地址中；和

- 5 地址发生器，用于控制第二数据交换器的输出，以便当在计算之后，通过第二数据交换器输出从第一数据交换器中读取的码元时，使从第一数据交换器中读取码元的存储体和地址与第二数据交换器的输出存储体和输出地址相同。

6. 根据权利要求 5 所述的调制设备，进一步包括在输入/输出数据和两个存储器之间进行交接的输入/输出接口。

7. 根据权利要求 5 所述的调制设备，其中，蝶式器具有基数-4 蝶式结构和被设计成当需要基数-2 计算时，通过包括在蝶式器中的多路复用器进行两次基数-2 计算。

8. 根据权利要求 5 所述的调制设备，其中，对于蝶式器的对称反向，按两个位将通过经过蝶式计算的码元的总数确定的二进制输出计数值转换成地址。

9. 根据权利要求 8 所述的调制设备，其中，当二进制输出计数值的数字是奇数时，通过以数字的中心位为基准进行对称转换确定地址。

10. 根据权利要求 5 所述的调制设备，其中，地址发生器利用通过对与计算的码元相对应的二进制输出计数值进行模-4 计算确定的值来确定存储体。

11. 根据权利要求 10 所述的调制设备，其中，当二进制输出计数值的数字是奇数时，地址发生器通过交换两个最高有效位的位置，通过低两个位进行模-4 计算，然后对位置交换最高有效位进行模-4 计算来确定存储体。

利用混合基数快速付里叶变换的调制设备

5 技术领域

本发明一般涉及调制数据的设备，尤其涉及基于正交频分多路复用（下文称为“OFDM”）技术或离散多频声（下文称为“DMT”）技术的调制设备。

背景技术

10 一般说来，在数字数据通信系统中，在发送之前对数据加以调制，在接收之后对数据加以解调。这样的调制和解调由可能随其调制方案而改变的MODEM（调制解调器）来完成。通常，用于数据通信的调制方案包括码分多路复用（CDM）、频分多路复用（FDM）、OFDM和DMT方案。

下文对OFDM和DMT调制方案加以描述。

15 OFDM方案是为无线通信系统中多路信道上的高速数据传输提出的。在提出OFDM方案之前，单载波传输方案用于数据传输。也就是说，利用OFDM方案之前的调制方案的无线通信系统调制串行传输数据，然后，利用信道的整个频带传输每个调制码元。OFDM方案或DMT方案将调制数据串-并转换成与载波数目一样多的数据码元，并且利用相应的副载波调制转换数据码元。
20 这种利用副载波的调制是利用离散付里叶变换（下文称为“DFT”）实现的。但是，对于实际的硬件设计，为了减少计算（或运算）次数，利用副载波的调制是利用快速付里叶变换（下文称为“FFT”）算法实现的，而不是利用DFT或离散付里叶逆变换（下文称为“IDFT”）算法实现的。当应用于OFDM系统时，处理FFT算法的处理器高度复杂，并且，需要高速计算。因此，难以实现处理FFT算法的处理器。
25

在需要高速计算的领域中，主要使用具有流水线结构的FFT处理器。但是，流水线结构需要与级数一样多的计算器，这样，点数的增加引起其硬件尺寸的增大。因此，为了解决与硬件尺寸增大有关的问题，人们引入了利用存储器结构和单个蝶式计算器的处理器。

30 利用基数-2 FFT算法的、基于存储器FFT处理器是这种处理器的典型例子。由于基于存储器FFT处理器可以将基数-2 FFT算法应用于存储器结构，

可以使乘法器数目达到最少。因此，基于存储器 FFT 处理器可以用在实现小型 FFT 处理器中。

但是，利用基数-2 FFT 算法的基于存储器 FFT 处理器需要许多个计算周期，这增加了计算时间。因此，基于存储器 FFT 处理器不适合于需要高速计算的 OFDM 系统或 DMT 系统，并且，为了满足高速计算要求，基于存储器 FFT 处理器需要极高工作频率。因此，在 OFDM 系统或 DMT 系统中，基数-4 算法一般用来取代基数-2 算法。现在对基于基数-4 算法的现有 FFT 处理器加以描述。

图 1 是例示 Amphion 公司引入的基于基数-4 算法 FFT 处理器的方块图。与基数-2 算法相比，基数-4 算法使级数减少一半，并且，还使每一级的蝶式计算次数减少一半。因此，基数-4 算法在蝶式计算次数上比基数-2 算法少得多。下表 1 所示的是基于 FFT 长度的基数-2 算法、基数-4 算法和如后所述的混合基数算法的计算次数。

表 1

FFT 长度	基数-2	基数-4	混合基数
256	1,024	256	-
512	2,304	-	640
1,024	5,120	1,280	-
2,048	11,264	-	3,072
4,096	24,576	6,144	-
8,192	53,248	-	14,336

如表 1 所示，基数-4 算法适合于 FFT 长度只为 4^n （其中， n 是整数）的 FFT 计算，而基数-2 算法适合于所有 FFT 长度 2^n 的 FFT 计算。例如，对于 2^8 的 FFT 长度 256 ($256 = 2^8$)，基数-2 算法和基数-4 算法两者都可以进行 FFT 计算。但是，对于 2^9 的 FFT 长度 512 ($512 = 2^9$)，基数-4 算法不能进行 FFT 计算，而基数-2 算法可以进行 FFT 计算。因此，为了对所有 FFT 长度 2^n 进行 FFT 计算，需要与另一种基数算法一起使用基数-4 算法的混合基数算法。表 1 的最后一列示出了当使用混合使用基数-4 算法和基数-2 算法的混合基数算法时，蝶式计算的次数。表 1 中通过混合基数算法进行的计算的次数等于用 Amphion 公司提供的 FFT 处理器进行的计算的次数。现在参照图 1 描述

Amphion 公司制造的 FFT 处理器。

参照图 1，利用混合基数算法的 FFT 处理器通过有选择地使基数-4 蝶式器（butterfly）和基数-4/基数-4 蝶式器工作，进行基数-4、基数-8 和基数-16 计算的混合基数计算。输入/输出接口和控制器 11 对从外部接收的输入数据 X 进行 FFT 计算，并且将 FFT 计算结果数据输出到 FFT 处理器的外部。输入/输出接口和控制器 11 的输入数据 X 和输出数据 Y 可以成为 OFDM 码元或 DMT 码元。存储器控制器 12 控制存储器 13 的地址生成，以便读写计算过程中的数据和从输入/输出接口和控制器 11 接收的用于 FFT 计算的数据。存储器 13 是利用 1024-字双端口存储器实现的，并且读写从外部接收的数据和存储器控制器 12 指定的地址中 FFT 计算的中间数据和结果数据。

蝶式计算器 10 由基数-4 蝶式器（butterfly）14、旋转因子查用表（下文称为“LUT”）16 和复数乘法器 15 组成。基数-4 蝶式器 14 进行基数-4 蝶式计算当中的相加和相减计算。旋转因子 LUT 16 是存储计算过程中数据的旋转因子和输出旋转因子值的存储表。复数乘法器 15 进行基数-4 蝶式计算当中复数相乘，生成复数相乘结果值。基数-4/基数-2 选择性蝶式器 17 有选择地进行基于 FFT 长度的最后计算。例如，当基于 FFT 长度的最后计算需要基数-2 计算时，选择基数-2 蝶式器进行基数-2 计算。但是，当最后计算需要基数-4 计算时，选择基数-4 蝶式器进行基数-4 计算。结果，通过将整个 FFT 计算与蝶式计算器 10 的基数-4 蝶式计算结合在一起，可以进行基数-8 计算或基数-16 计算。因此，FFT 处理器包括只有在末级中选择基数-4/基数-2 选择性蝶式器 17 和在其它级中选择基数-4 蝶式计算器 10 的多路复用器（MUX）18。基数-4 算法是利用具有 4 个输出端和 4 个输出端的蝶式器实现的。因此，在一个周期内必须进行 4 次输入和 4 次输出，以便使计算周期数最少。为了在一个周期内进行 4 次输入和 4 次输出，必须将存储器划分成多个存储体。但是，图 1 的 FFT 处理器没有多存储体结构。因此，图 1 的 FFT 处理器需要多个计算周期，未能利用基数-4 计算。

图 2 是例示 Drey Enterprise 公司引入的、拥有混合基数算法和多存储体结构的 FFT 处理器的方块图。如图 2 所示，Drey Enterprise 公司引入的 FFT 处理器也拥有存储器结构。在图 2 的 FFT 处理器中，在两个输入存储器（RAM）21 和 22 之一存储来自外部的输入数据的同时，另一个输入 RAM 用于 FFT 计算。MUX 23 确定将从输入 RAM 21 和 22 之一接收蝶式输入，还是从输出

RAM 28 和 29 之一接收蝶式输入。基数-2 计算器 26 和 27 的每一个在基数-2 计算级中进行基数-2 计算，生成基数-2 计算结果。MUX 24 多路复用从基数-2 计算器 26 和 27 接收的计算结果值，以便将基数-2 计算结果值写入输入 RAM 21 和 22 的某一个或输出 RAM 28 和 29 的某一个中。基数-2/基数-4 公用计算器 25 在基数-4 计算级中进行基数-4 计算，并且在基数-2 计算级中进行基数-2 计算。在两个输出 RAM 28 和 29 之一用于 FFT 计算的同时，另一个 RAM 将 FFT 计算结果输出到外部。图 2 的结构使用了基数-4 算法和基数-2 的算法的混合算法，还使用了多存储体存储器结构。多存储体存储器结构的使用有助于使计算时钟周期达到最少。

- 10 但是，图 2 的结构未能应用将蝶式输出写入蝶式输入已被访问的存储位置中的原地算法。因此，图 2 的结构将两个 N-字存储器用于 FFT 计算。也就是说，仅仅对于 FFT 计算，只需要两个 4-存储体存储器。但是，为了进行连续处理，必须将多于两个的 4-存储体存储器用于输入和输出。因此，在图 2 中，总共使用了 4 个存储器。存储器是占据 FFT 处理器最多区域的模块之一。
- 15 因此，存储器数目的增加引起 FFT 处理器的存储器复杂度、硬件尺寸和成本增加。

- 图 3 例示了 L.G.Johnson 引入使存储器结构的存储器复杂度达到最小的原地算法的 16-点 FFT。原地算法用在将存储器划分成多个存储体的时候。对于基数-4 蝶式计算，必须同时访问 4 个码元，并且必须同时将 4 个蝶式计算结果写入访问位置中。为了这个目的，必须将主存储器划分成 4 个存储体，即，存储体 #0、存储体 #1、存储体 #2 和存储体 #3，并且必须进行适当寻址，以便不同时从一个存储体访问几个数据码元。图 3 例示了 16-点 FFT 的原地存储器寻址，其中配备了进行第 1 到第 8 蝶式计算的结构。在每次蝶式计算中，一次挑选 4 个输入。这里，4 个输入是从不同存储体读取的。在第 1 蝶式计算中，从存储体 #0 的地址 0、存储体 #1 的地址 1、存储体 #2 的地址 2 和存储体 #3 的地址 3 中读取 4 个输入，并且将蝶式计算结果写入同一存储体的同一地址中。在第 2 蝶式计算中，从存储体 #1 的地址 0、存储体 #2 的地址 1、存储体 #3 的地址 2 和存储体 #0 的地址 3 中读取 4 个输入，并且将蝶式计算结果写入同一位置中。在图 3 中，通过对将输入数据计数位划分成 2-位数字确定的值进行模-4 相加，可以简单地计算出指示正在使用的存储体的存储体索引 i。由于图 3 的 FFT 是 16-点 FFT，使用 4-位计数器，以便计数

16 个数据位。将 4 个位划分成 2 个高位和 2 个低位，在对 2 个高位和 2 个低位进行模-4 相加的方法中计算存储体索引 i 。

但是，上述原地算法是为固定基数系统，而不是为混合基数系统提出的。因此，如果不作修改，原地算法 (in-place algorithm) 不能应用于混合基数系统。

接着，对传统连续处理结构加以描述。R.Radhouane 提出了通过在存储器结构中同时进行输入和输出，只利用两个 N -字存储器就能够进行连续处理的基于存储器 FFT 处理器。这种结构在根据当基数-2 算法进行 DIF (频度抽选) 计算和 DIT (时间抽选) 计算时，其输出和输入具有位反向特性的事实，交替进行 DIF 计算和 DIT 计算的方法中实现了连续处理。下表 2 是利用两个存储器的连续处理结构的计算方法。

表 2

OFDM 码元	存储器 # 1		存储器 # 2	
	存储速率	FFT-I/O 模式	存储速率	FFT-I/O 模式
0	C	DIF	I/O	NAT
1	I/C	BR	C	DIF
2	C	DIT	I/C	BR
3	I/C	NAT	C	DIT

在表 2 中，“OFDM 码元”指的是与 FFT 计算的长度相对应的数据。例如，在 256-点 FFT 计算中，一个 OFDM 码元指的是 256 个数据位。在表 2 中，“C”指的是 FFT 计算，并且“I/O”指的是进行输入/输出。而且，“NAT”指的是通过按地址 0, 1, 2, 3, ..., $N-1$ 的正序进行存储器寻址，进行输入/输出，并且“BR”指的是通过位反向寻址进行存储器输入/输出。另外，在表 2 的第 0 个 OFDM 码元中，存储器 # 1 通过 DIF 进行计算，而存储器 # 2 通过进行 NAT，即，正序的存储器寻址进行输入/输出。接着，在第 1 个 OFDM 码元中，存储器 # 1 通过 BR，即，位反向寻址进行输入/输出，而存储器 # 2 通过 DIF 进行计算。在第 2 个 OFDM 码元中，存储器 # 1 通过 DIT 进行计算，而存储器 # 2 通过 BR，即，位反向寻址进行输入/输出。接着，在第 3 个 OFDM 码元中，存储器 # 1 通过 NAT，即，正序的存储器寻址进行输入/输出，而存储器 # 2 通过 DIT 进行计算。从接着的第 4 个 OFDM 码元开始，重复对第 0 到第 3 个

OFDM 码元的一系列计算。为了利用两个存储器进行连续处理，在一个存储器进行计算的同时，另一个必须能够同时进行有序的输入和输出。在这种结构中，只利用两个存储器就可以进行连续处理，从而两个存储器交替进行输入/输出和 FFT 计算。

- 5 虽然 Alcatel 公司引入的传统结构通过利用三个存储器实现了连续处理，但上述传统连续处理结构通过只利用两个存储器，可以使存储器复杂度降到最低。

但是，上述传统连续处理结构只是为使用基数-2 算法的情况设计的。不利的是，由于连续处理结构只进行基数-2 计算，它需要许多计算周期和高工作频率。

10

发明内容

因此，本发明的一个目的是提供一种能够使其尺寸最小和通过提供高速计算满足高性能要求的 FFT 处理器和复杂度最小的电路。

- 15 本发明的另一个目的是提供一种在提供基数-2 算法和基数-4 算法的混合基数算法的同时，能够降低其尺寸和复杂度的 FFT 处理器。

本发明的又一个目的是提供一种在提供基数-2 算法和基数-4 算法的混合基数算法的同时，能够进行快速处理的 FFT 处理器。

- 本发明的再一个目的是提供一种在提供基数-2 算法和基数-4 算法的混合基数算法的同时，能够进行连续处理的 FFT 处理器。
- 20

为了实现上面和其它目的，本发明提供了利用混合基数快速付里叶变换 (FFT) 的调制设备，它包括输入/输出接口，用于从用于输入/输出和 FFT 计算的存储器当中选择用于输入/输出的存储器，选择所选存储器的 4 个存储体之一，并且对所选存储体进行输入和输出；两个 N-字存储器，每一个都含有用于针对输入/输出接口的输入/输出和 FFT 计算的 4 个存储体；第一数据交换器，用于从用于针对输入/输出接口的输入/输出和 FFT 计算的存储器当中选择用于 FFT 计算的存储器，并且将指定给每个蝶式输入/输出的存储体与用于原地计算的蝶式计算电路的 4 个输入端相连接；蝶式器，用于利用一个电路，根据第一数据交换器提供的输出，进行基数-4 蝶化 (butterfly) 和基数-2 蝶化，并形成对称反向输出；第二数据交换器，用于从用于针对输入/输出接口的输入/输出和 FFT 计算的存储器当中选择用于 FFT 计算的存储器，并且

25

30

将指定给每个蝶式输入/输出的存储体与用于原地计算的蝶式计算电路的4个输出端相连接；以及地址发生器，用于生成在多存储体存储器结构中进行原地计算的存储体索引和地址。

- 5 另外，本发明提供了另一种利用混合基数快速付里叶变换（FFT）的调制设备，它包括每一个都含有4个存储体的两个存储器，用于写入输入码元或FFT计算码元；蝶式器，用于按照从存储器输出的码元的个数，以基数-2模式或基数-4模式进行蝶式计算，并且对称反向地输出计算值；第一数据交换器，用于从存储器之一的每个存储体中读取一个码元，并且将读取的码元输出到蝶式器；第二数据交换器，用于匹配从蝶式器输出的计算码元，以便
- 10 将码元写入与从第一数据交换器中读取码元的地址相同的地址中；以及地址发生器，用于控制第二数据交换器的输出，以便当在计算之后，通过第二数据交换器输出从第一数据交换器中读取的码元时，使从第一数据交换器中读取码元的存储体和地址与第二数据交换器的输出存储体和输出地址。

15 附图说明

通过结合附图，进行如下详细描述，本发明的上面和其它目的、特征和优点将更加清楚，在附图中：

图1是例示没有使用多存储体结构的传统混合基数FFT处理器的结构的方块图；

- 20 图2是例示没有使用原地算法的传统混合基数FFT处理器的结构的方块图；

图3是例示用于传统多存储体存储器的基数-4原地算法的图形；

图4是例示按照本发明实施例的具有连续处理结构的基于原地算法混合基数FFT处理器的结构的方块图；

- 25 图5是例示按照本发明实施例的32-点混合基数FFT处理器的计算的信号流图；

图6A是例示用在图4的FFT处理器中的基数-4/基数-2蝶式电路的方块图；

图6B是例示图6A的基数-2蝶式器的等效蝶式器对的图形；

- 30 图7是例示对于只有基数-4算法用在图4中的情况，实现连续处理的FFT计算的图形；

图 8A 是例示对于只有基数-4 算法用于 4^n -点 FFT 计算的情况, 数字反向输出序列的图形;

图 8B 是例示对于混合基数算法与基数-2 算法一起用于 2^n -点 FFT 计算(其中, $n=3, 5, 7, 9, \dots$) 的情况, 不对称反向输出序列的图形;

5 图 9 是例示示出 32-点混合基数 FFT 计算的图 5 的列 # 4 的不对称反向输出序列的图形;

图 10 是例示示出 32-点混合基数 FFT 计算的图 5 的列 # 3 的对称反向输出序列的图形;

10 图 11 是例示用于 2^n -点 FFT 计算(其中, $n=3, 5, 7, 9, \dots$) 的混合基数算法的对称反向输出序列的图形;

图 12 是例示生成图 5 的列 # 3 的对称反向输出序列的数据交换器的图形;

图 13 是例示图 5 的 32-点混合基数 FFT 计算的存储体索引生成方法的图形; 和

15 图 14 是例示 2^n -点 FFT 计算(其中, $n=3, 5, 7, 9, \dots$) 的混合基数存储体索引生成方法的图形。

具体实施方式

现在参照附图描述本发明的几个优选实施例。在如下的描述中, 为了简洁起见, 省略包含在本文中的那些众所周知功能和配置的详细描述。

20 图 4 是例示与按照本发明实施例的混合基数结构一起, 通过应用原地算法和同时进行输入和输出, 利用 $2N$ -字存储器进行连续处理的 FFT 处理器。

参照图 4, 为了对输入数据进行 FFT 计算, 输入/输出接口 101 选择存储器 102 和 103 之一, 选择所选存储器的 4 个存储体之一, 并且将输入数据写入所选存储体中。当对输入数据的 FFT 计算完成时, 输入/输出接口 101 选择
25 写入 FFT 计算数据的存储器, 选择所选存储器的 4 个存储体之一, 并且从所选存储体中读取数据。每一个都含有 4 个存储体的两个 N -字存储器 102 和 103 写入从输入/输出接口 101 接收的数据, 将写入的数据提供给第一数据交换器 104, 供 FFT 计算用, 并且写入从第二数据交换器 106 接收的数据。将来自存储器 102 和 103 的 FFT 计算数据提供给输入/输出接口 101。第一数据交换器 104 按照从地址发生器 107 输出的地址值交换读取的数据, 并且将其输出
30 提供给基数-4/2 蝶式器 105。基数-4/2 蝶式器 105 按照从第一数据交换器 104

接收的数据，在基数-4 模式或基数-2 模式下工作。基数-4/2 蝶式器 105 由一个电路组成，并且含有对称反向输出端。将基数-4/2 蝶式器 105 计算的数据提供给第二数据交换器 106。第二数据交换器 106 选择要将从数-4/2 蝶式器 105 输出的值写入其中的存储器和存储器的存储体，并且将输出值写入所选存储器的存储体中。地址发生器 107 生成在多存储体存储器结构中，按照本发明进行原地计算的存储体索引和地址。

具有图 4 所示的结构 FFT 处理器具有如图 5 所示的信号流程图。图 5 例示了 32-点混合基数 FFT 计算的例子，其中，级 #1 和级 #2 经受基数-4 计算，并且末级 #3 经受基数-2 计算。正如结合图 4 所述的那样，在本发明提出的结构中，2 个存储器各自被划分成 4 个存储体，这样，可以同时访问 4 个数据码元。因此，利用 2 个数据码元的基数-2 蝶式器可以同时进行 2 个蝶式计算。另外，在级 #3 中，显示在用细实线画出的方框中的 2 个基数-2 蝶式器代表可以同时进行蝶式计算的蝶式器对，从而有助于计算周期的减少。

在 FFT 处理器中，基数-2 蝶式结构不是利用分立蝶式器实现的，而是将数据开关电路加入基数-4 蝶式器中实现的。这种情况例示在图 6A 中。在图 6A 中，通过多路复用器选择信号‘Radix-2’实现一个基数-4 蝶式器和两个基数-2 蝶式器。参照图 6A，将输入数据码元 $x(0)$ 和 $x(2)$ 供应给构成一个基数-2 蝶式器的加法器，并且将输入数据码元 $x(1)$ 和 $x(3)$ 供应给构成另一个基数-2 蝶式器的加法器。构成基数-2 蝶式器的加法器每一个都将它们的输出分支（或划分）成两个输出，并且将一个输出提供给每个多路复用器的输入端。将另一个分支输出加入来自构成基数-2 蝶式器的其它加法器的输出中或从来自构成基数-2 蝶式器的其它加法器的输出中减去另一个分支输出，并且将结果提供给多路复用器的另一个输入端。此后，如图所示，按原样输出每个多路复用器的输出，或通过其它多路复用器再次分支或选择它。按照本发明将这样的输出值映射到原地地址。下面将更详细地对此加以描述。图 6A 的等效电路图例示在图 6B 中。图 6B 例示了与基数-4 蝶式电路一起实现的基数-2 蝶式器等效的基数-2 蝶式器对。

现在对按照本发明在存储器结构中进行连续处理的结构加以描述。正如上面在现有技术的描述中所述的那样，R.Radhouane 提出的连续处理结构是用于基数-2 算法的结构。但是，本发明提出的结构是用于与基数-4 算法混合的混合基数算法的结构。另外，传统结构交替地进行 FFT 计算和 DIT 计算。但

是，本发明提出的结构只进行 DIF 计算和通过简单控制存储器寻址进行连续处理。

为了进行如本发明所提出那样的连续处理，必须将要重新解码的码元写入要解码的码元的位置中。通过读取要解码的码元和将要重新解码的码元写入读取位置中的操作，同时进行读取和写入操作。为了满足基数-4 蝶式器，4-存储体存储器的每个存储体必须只生成一个输出。当通过每个级的计算完成时，必须将码元写入读取位置中。因此，务必以这种方式生成地址。只有当使用基数-4 算法时，上面的方法才可以简单实现。因此，图 7 例示了只使用基数-4 算法的 16-点 FFT 的例子。

在图 7 中，列 #1 和列 #2 代表写入数据码元的存储器存储体和地址，和列 #3 代表数字反向输出序列。在图 7 中，将首先从外部接收的输入数据码元写入列 #2 的存储体和地址中。输入数据码元在第 1 级中经受 4 次蝶式计算，和在第 2 级中经受 4 次蝶式计算。也就是说，在图 7 中，(1) 到 (4) 代表在第 1 级中进行 4 次蝶式计算，和 (5) 到 (8) 代表在第 2 级中进行 4 次蝶式计算。应该注意到，在每一级的每次蝶式计算中，从构成一个存储器的 4 个存储体的每一个中读取一个数据码元。在第 1 次蝶式计算中，读取存储体 #0 的地址 0 中的数据、存储体 #1 的地址 1 中的数据、存储体 #2 的地址 2 中的数据和存储体 #3 的地址 3 中的数据。因此，从每个存储体中读取一个数据码元。如果以这种方式完成了 4 次蝶式计算。在第 2 级中再次进行 4 次蝶式计算。以这种方式计算的输出具有如图 7 的列 #3 中所示的输出序列。

当以上面的方式进行 FFT 计算时，以列 #3 的数字反向序列形成输出，并且，对于存储器写入位置，由于进行的是原地计算，保持了列 #2 的存储体和地址。通过数字反向寻址按顺序输出计算结果，同时，当按顺序接收下一个数据码元时，将新输入数据码元写入列 #1 的存储体和地址中。由于列 #3 的第 0 输出被写入列 #2 中存储体 #0 的地址 0 中，将用于下一次 FFT 计算的新数据的第 0 码元写入进行第 0 输出的存储体和地址中。接着，将列 #3 的第 1 输出写入列 #2 中存储体 #1 的地址 1 中，并将新数据的第 1 码元写入进行第 1 输出的位置中。将第 2 输出写入列 #2 中存储体 #2 的地址 2 中，并将新数据的第 2 码元写入进行第 2 输出的位置中。当以这种方式写入新输入时，形成列 #1 的存储体和地址。在对此进行 FFT 计算之后，如果按顺序输出 FFT 计算结果和按顺序写入输入，那么，通过列 #2 的存储体和地址的指

定得到恢复。因此，列 # 1 的存储体和地址指定和列 # 2 的存储体和地址指定接替进行下去。

如果像上述那样可以按顺序同时进行输入和输出，那么，只利用两个存储器就可以进行连续处理，从而在一个存储器进行计算的同时，另一个存储器进行输入和输出。此刻，必须以两倍于输入/输出工作频率的工作频率进行 FFT 计算。这是因为，从表 1 可以看出，FFT 计算时钟周期大于 FFT 点数。也就是说，对于基数-4 算法，计算时钟周期从 1024-点 FFT 开始变得更长了，和对于基数-4/基数-2 算法的混合基数算法，计算时钟周期从 512-点 FFT 开始变得更长了。

混合基数算法需要分开管理连续处理的顺序输入/输出。在 32-点混合基数算法中，像图 5 的列 # 4 中所示那样提供输出。与图 7 中的列 # 3 中基数-4 算法的反向序列不同，这具有不对称反向形式。首先，针对只使用基数-4 算法的情况，对数字反向序列加以描述。基数-4 算法的 2^n -点 FFT 的数字反向序列例示在图 8A 中。由于需要 n 个位来计数 2^n 个数据码元，使用 n -位计数器。在图 8A 中，在第 $(n-1, n-2)$ 位对，第 $(n-3, n-4)$ 位对， $\dots$ ，第 $(3, 2)$ 位对，和第 $(1, 0)$ 位对每一个都是一个数字的假设下，进行反向排序。这里，以这种方式进行的反向排序被称为“数字反向”。从图 8A 中可以了解到，以数字的中心为基准对称地进行反向排序。

图 8B 例示了 2^n -点 FFT 的混合基数算法的反向序列。由于混合基数算法具有与诸如 2^3 ， 2^5 ， 2^7 和 2^9 之类 2 的奇数次幂的数字一样多的点数，并输出计数位是诸如 3，5，7 和 9 之类的奇数，混合基数算法不能像基数-4 算法那样，只利用 2-位数字进行反向排序。必须分开地使最低有效位反向，由于这个原因，提供了不对称反向形式。作为图 8B 的例子，与 $32 (= 2^5)$ -点 FFT 相对应的图 5 的列 # 4 具有如图 9 所示的不对称反向形式。当提供不对称反向输出时，不能像由基数-4 算法组成的图 7 的结构那样，形成列 # 1 和列 # 2 的存储体和地址重复的结构。即使在混合基数算法中，为了像只利用基数-4 算法的图 7 的结构那样，为连续处理提供重复列 # 1 和列 # 2 的结构，输出必须具有对称反向形式。为此，在混合基数算法中，进行数据交换，以便将如图 5 的列 # 4 所示的不对称输出序列转换成如列 # 3 所示的对称输出序列，结果，在图 5 的列 # 3 中示出了 $32 (= 2^5)$ -点对称反向输出序列，和可以像图 0 所示那样生成它。现在描述一般化 2^n -点混合基数 FFT，输出的对称反向序

列例示在图 11 中。这里，高两位 $(n-1, n-2)$ 和低两位 $(1, 0)$ 经受数字反向 (reverse)，并且中间位 $(n-3, n-4, \dots, 3, 2)$ 经受位反向。总之，通过数据交换过程将图 8B 的原不对称反向形式转换成图 11 的对称反向形式。

图 12 例示了实现如图 10 所示的 32-点对称反向序列和图 5 的列 #3 的数据交换过程。图 12 对应于图 5 中用粗实线画出的方框中的 8-点 DFT 部分。如图 12 所示，在级 #2 中，交换基数-4 蝶式器的 2 个第 2 输出 $g'2(n)$ 和 2 个第 3 输出 $g'3(n)$ 的写入位置，并且，在级 #3 中，交换基数-2 蝶式器对的输出 $X'2(n)$ 和 $X'3(n)$ 的写入位置。当以这种方式形成时，如图 12 的列 #1 所示，提供对称反向序列。在图 12 中，列 #2 示出了对于不交换写入位置的情况的不对称反向序列。这样的数据交换可以通过控制图 6A 的蝶式电路中的 'Exchange' 信号来完成。在所有 2^n -点 FFT (其中， $n=1, 3, 5, 7, 9, \dots$) 中，以及在作为例子给出的 32-点 FFT 中，通过在除了第 1 级之外的其它级中交换基数-4 蝶式器的第 2 和第 3 输出写入位置，并且在末级中交换两个基数-2 蝶式器的第 2 和第 3 输出写入位置，可以形成对称反向输出。

最后，对在混合基数算法中生成存储体索引的方法加以描述。按照本发明的存储体索引生成方法生成如图 5 的列 #1 和列 #2 所示的存储体索引。如果 FFT 长度是 2^n ，利用 n -位计数器生成存储体索引和每个存储体的地址。在现有技术部分中所述的基数-4 算法存在 2^2 、 2^4 、 2^6 或 2^8 个点。因此，可以通过对 2-位数字进行模-4 相加生成存储体索引 i 。但是，混合基数算法存在 2^3 、 2^5 、 2^7 或 2^9 个点，这样，输入计数位的个数变成诸如 3、5、7 和 9 之类的奇数。因此，不可以简单地通过对 2-位数字进行模-4 相加生成存储体索引 i 。在本发明的混合基数算法中，当输入计数位的数字是奇数时，通过如下过程进行生成存储体索引的方法。首先，交换两个最高有效位的位置。其次，从低位开始将位置交换计数值划分成 2-位数字，和划分的 2 个位进行模-4 相加。然后，保留一个最高有效位，并让通过对其余最高有效位进行模-4 相加计算的值再次经受模-4 计算。

图 13 例示了形成 $32 (= 2^5)$ -点 FFT 的 2-位数字和 1 个位方法。如果以上述方式交换和计算输入计数位的位置，当进行模-4 计算时，最后剩下的一个位对应于输入数据计数位当中的第 3 位，而从低位开始通过 2-位数字经受模-2 相加计算的位的匹配对应于 $(4, 2)$ 和 $(1, 0)$ 。当这样生成存储体索引时，在 FFT 计算期间可以从其它存储器中读取数据，并且，当在图 5 中作出

从列 # 2 到列 # 1 的改变时，可以保持存储体索引序列不变。

图 14 例示了通过推广用于 32-点混合基数 FFT 的存储体索引生成方法获得的生成用于 2^n -点混合基数 FFT 的存储体索引的方法。在图 14 中，输入数据计数位当中一个单独位的位置是第 (n-2) 位，在对 2-位数字进行模-4 相加的过程中包括单独位。

如上所述，本发明可以利用基于基数-4 算法的混合基数算法进行高速计算。另外，本发明通过将原地计算应用于混合基数算法，和利用每一个包括 4 个存储体的 2 个 N-字存储器，通过同时输入和输出进行连续处理，使存储器占据的空间达到最小。

10 下表 3 所示的是建议 FFT 处理器的计算周期和传统 FFT 处理器的计算周期之间的比较。从表 3 中可以了解到，与利用基数-2 算法的传统存储器结构相比，建议存储器结构使其计算周期缩短到 1/4。

表 3

结构	时钟周期	N=2,048	N=4,096
传统结构	$\text{Radix-2} \frac{N}{2} \log_2 N + 2$	11,266	24,578
建议结构	$\text{Radix-4} \frac{N}{4} \log_4 N + 6$	-	6,150
	Mixed-radix(Radix-4, $\text{Radix-2} \frac{N}{4} \log_4 2N + 6$)	3,078	-

15 另外，当将本发明与不利用多存储体结构的传统基于基数-4 算法混合基数 FFT 处理器相比时，传统 FFT 处理器消耗大约是建议 FFT 处理器 4 倍长的计算周期。不采用原地算法和同时输入/输出结构的传统混合基数 FFT 处理器需要每一个由 4 个存储体组成的 4 个 N-字存储器，具有相当于需要 2 个 N-字存储器的建议 FFT 处理器的存储器尺寸 2 倍的存储器尺寸。因此，本发明

20 满足了高速计算要求和低硬件复杂度要求，因此，可以简单应用于 OFDM 或 DMT 系统。

虽然通过参照本发明的某些优选实施例，已经对本发明进行了图示和描述，但本领域的普通技术人员应该明白，可以在形式上和细节上对其作各种各样的改变，而不偏离所附权利要求书所限定的本发明的精神和范围。

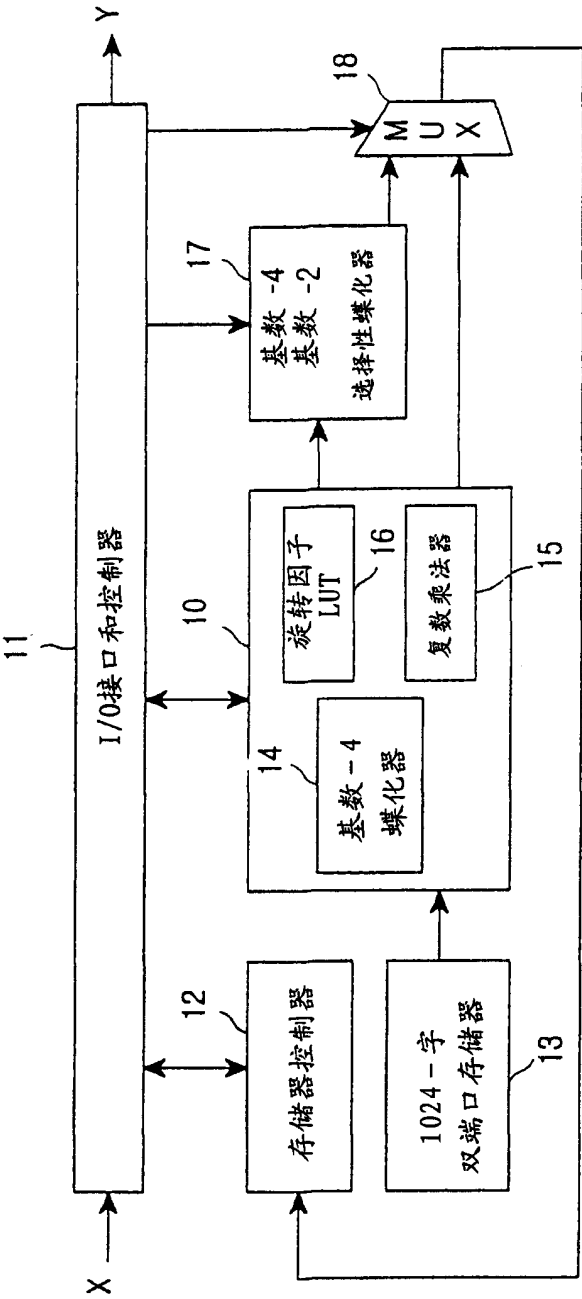


图 1

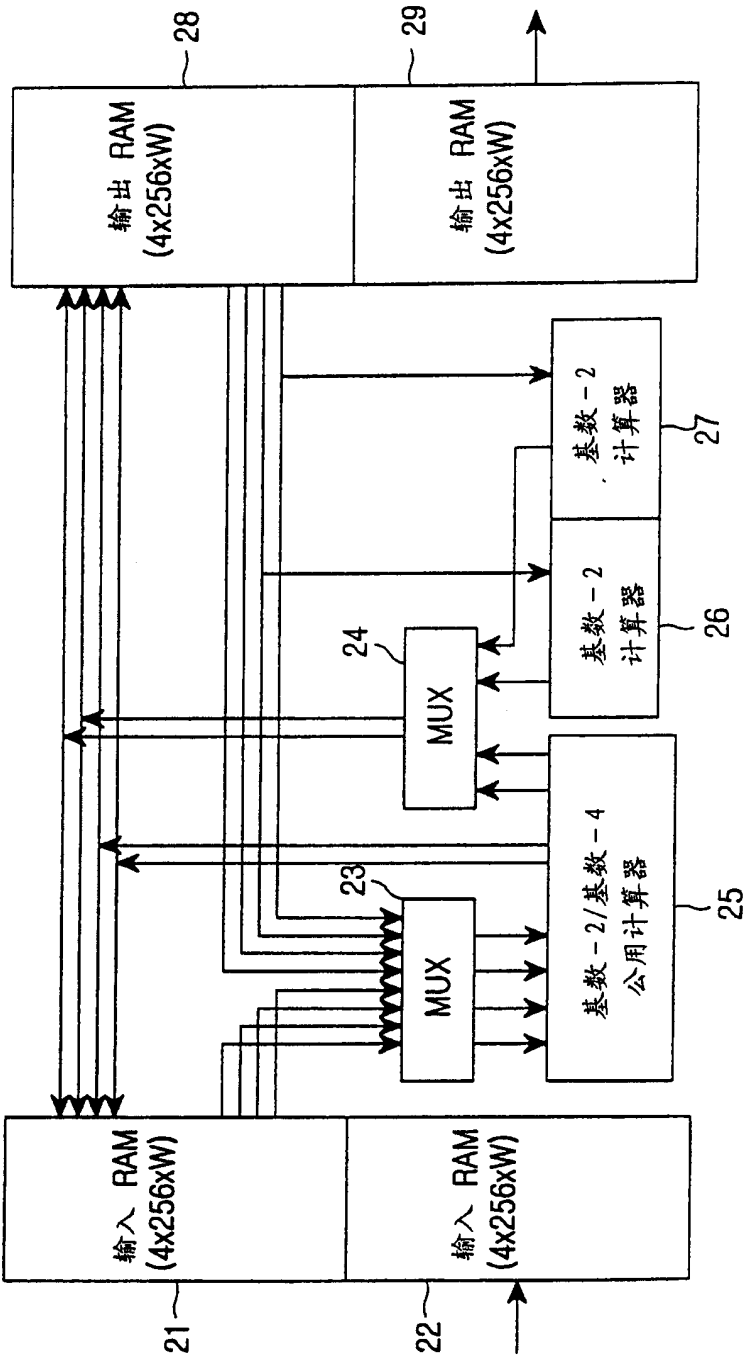


图 2

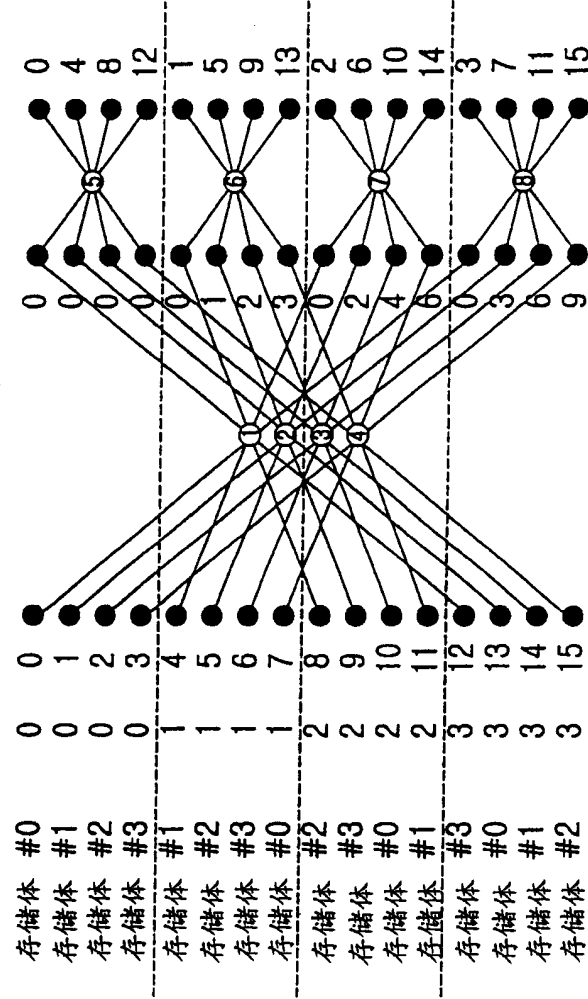


图 3

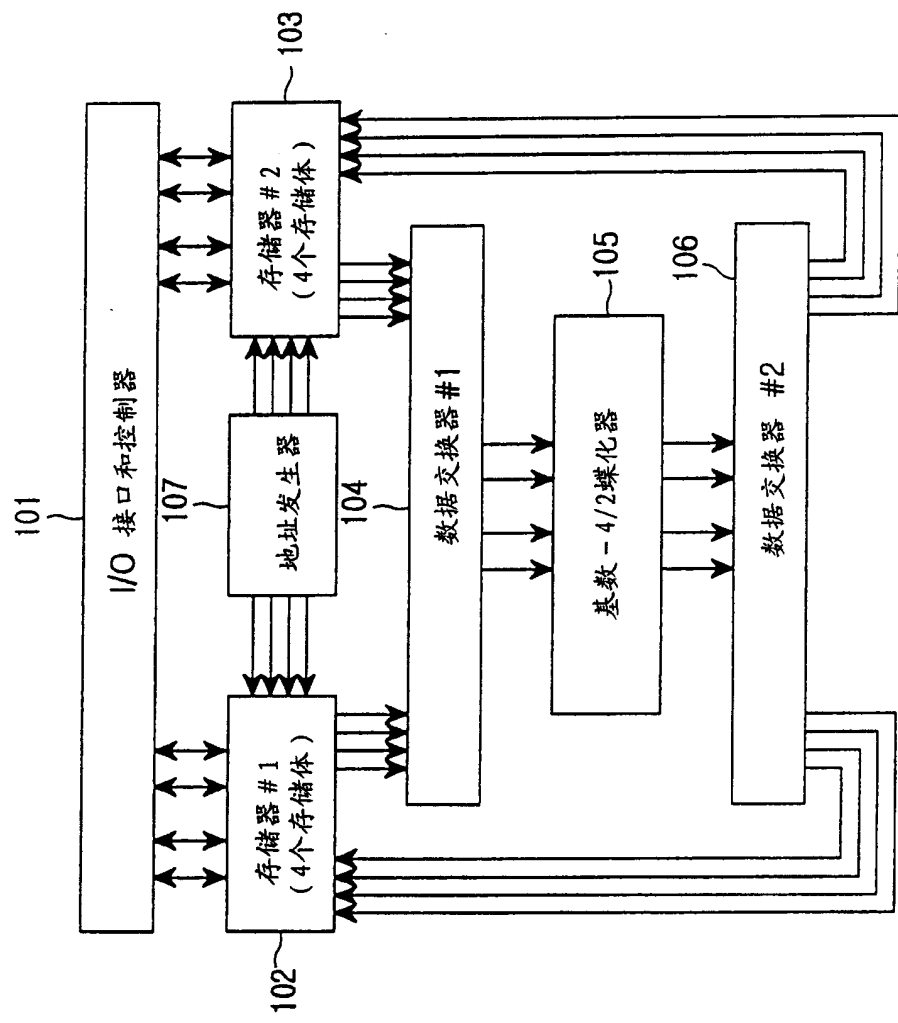


图 4

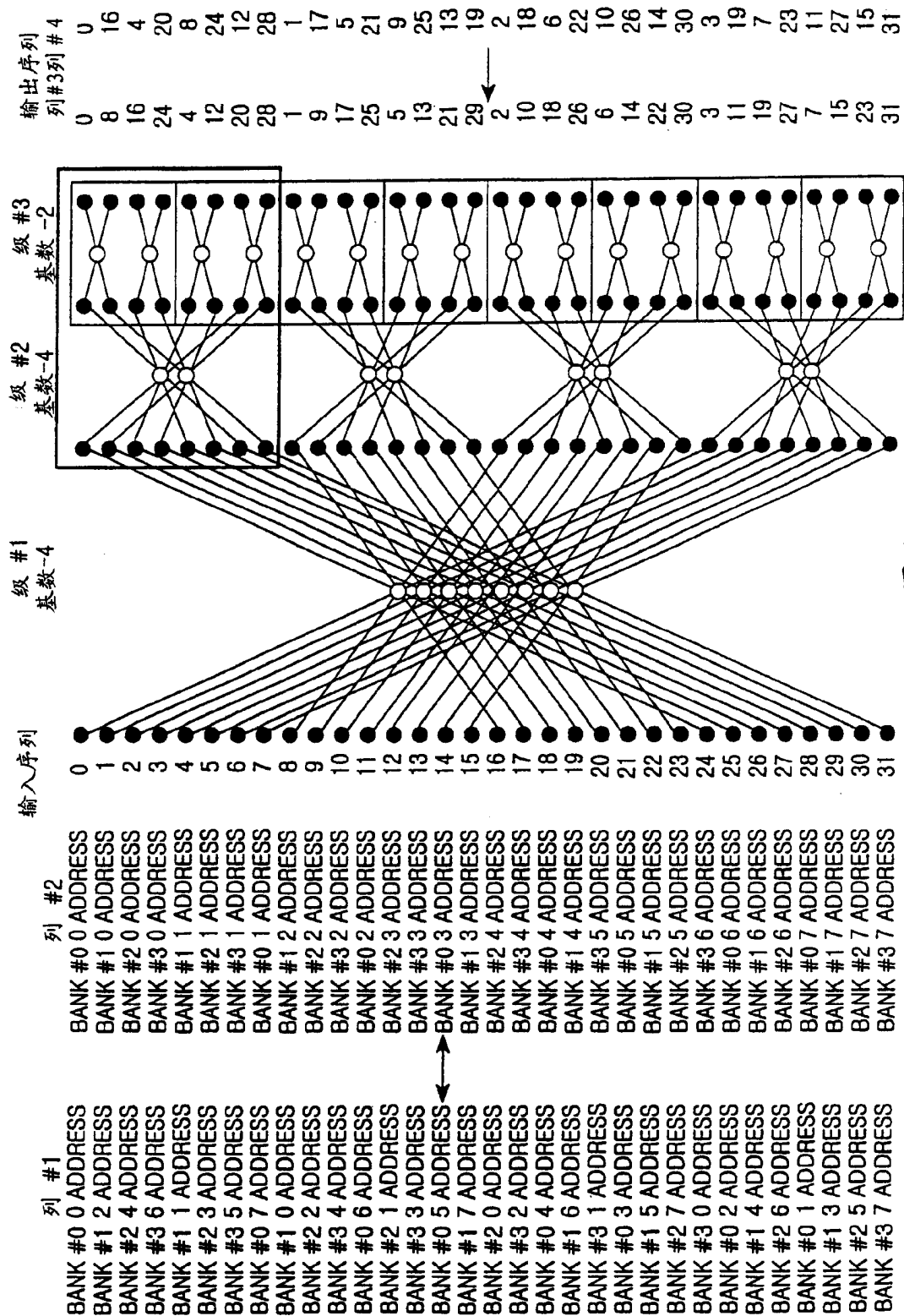


图 5

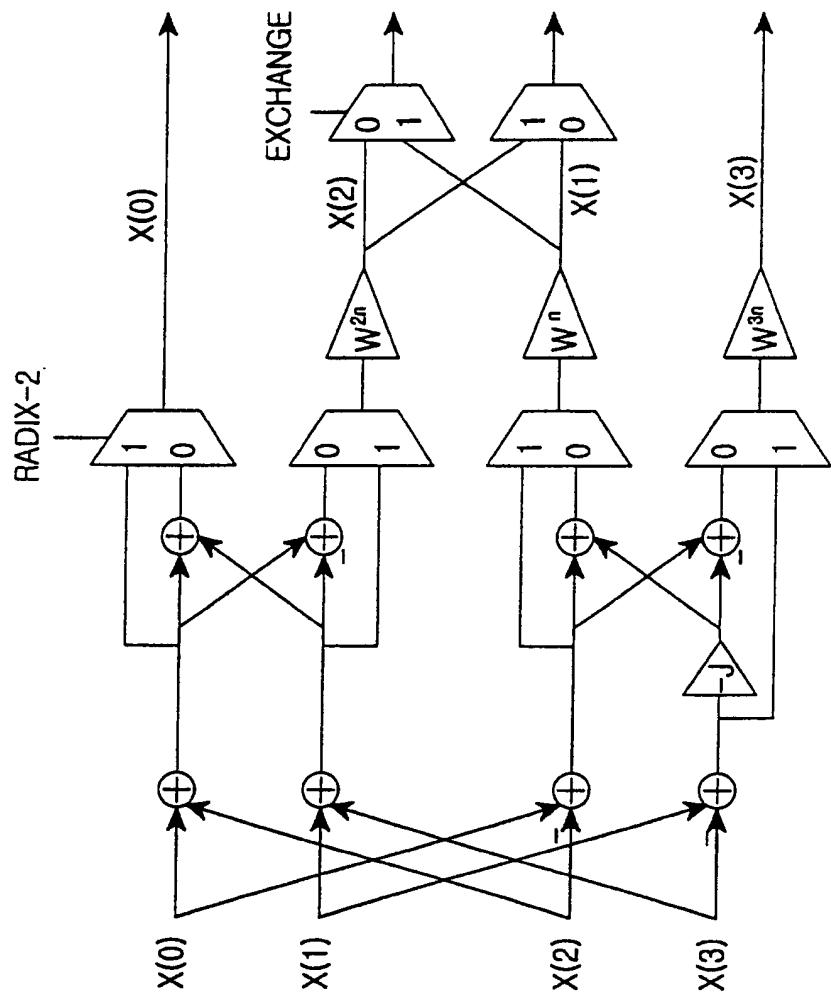


图 6A

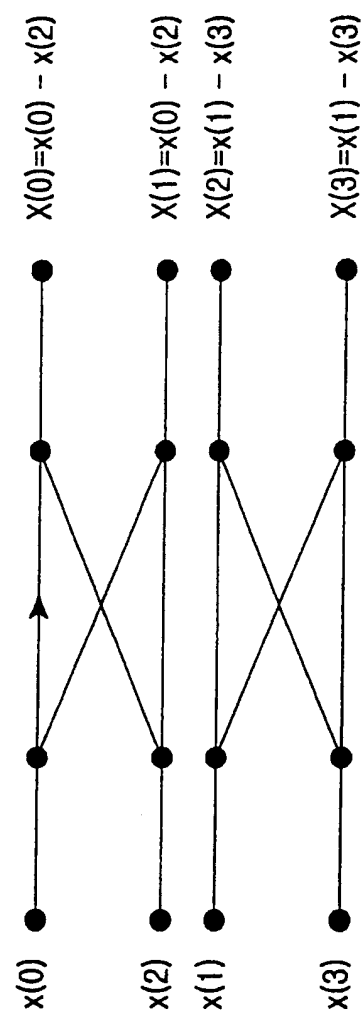


图 6B

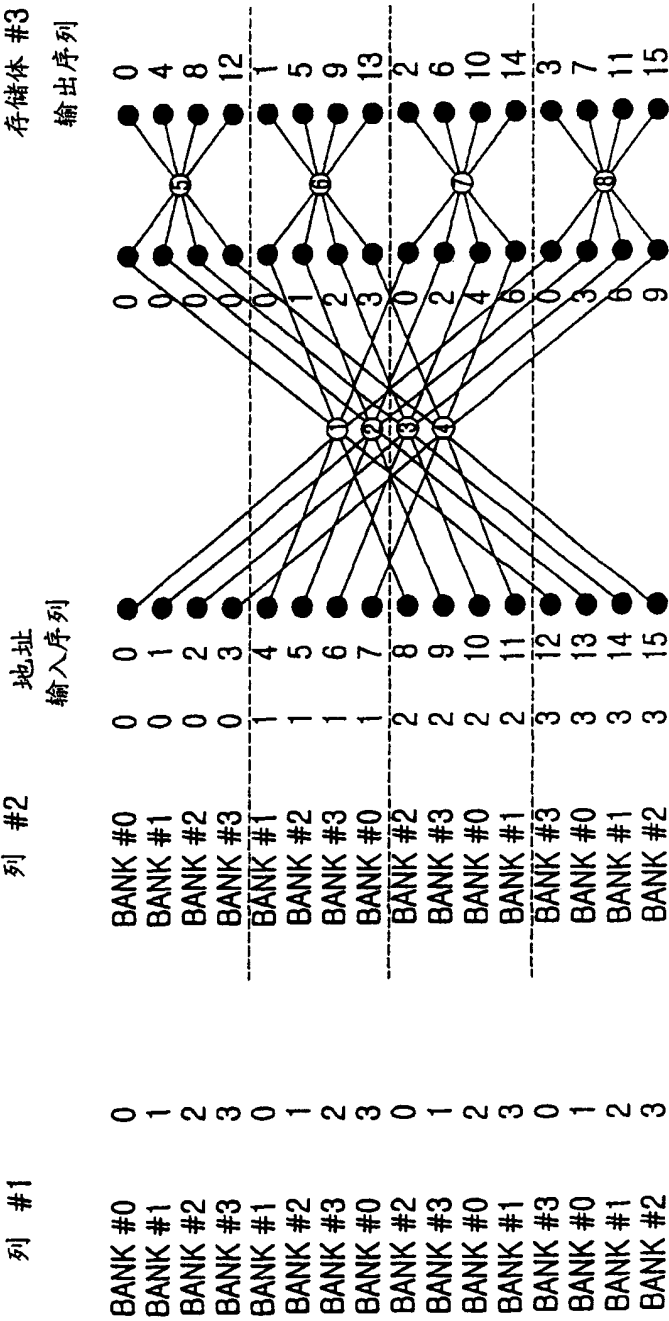


图 7

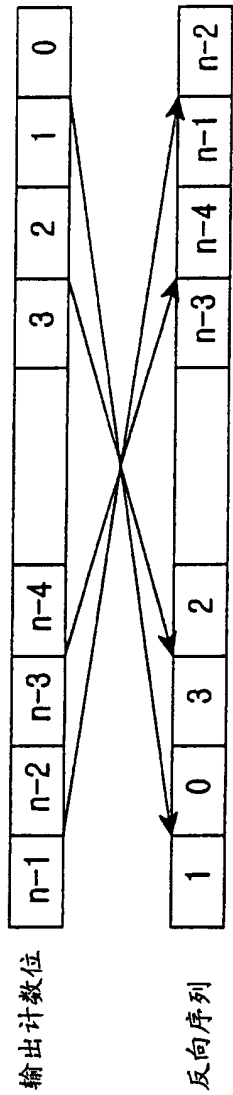


图 8A

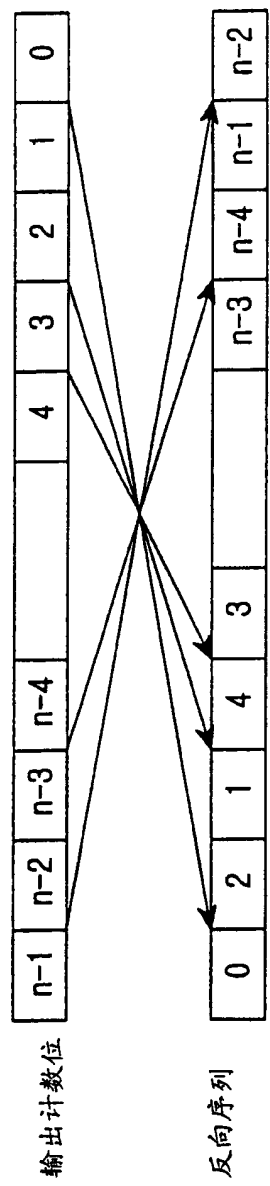


图 8B

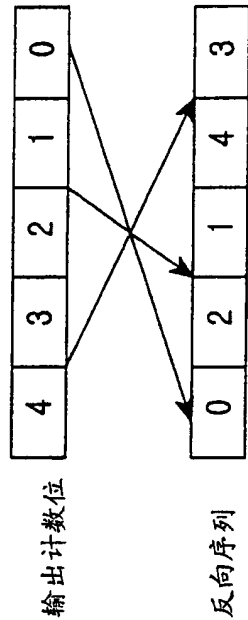


图9

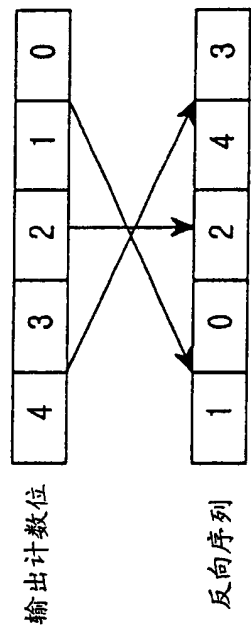


图 10

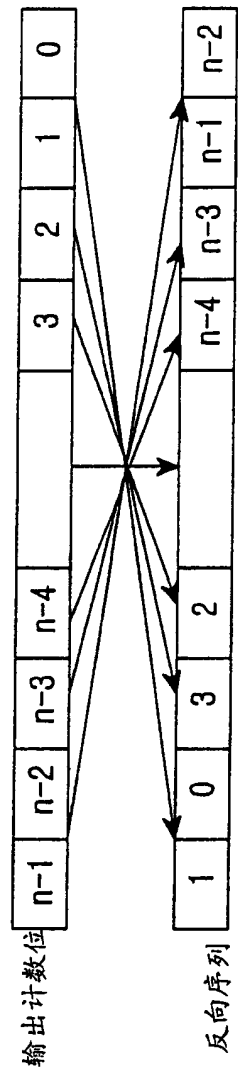


图 11

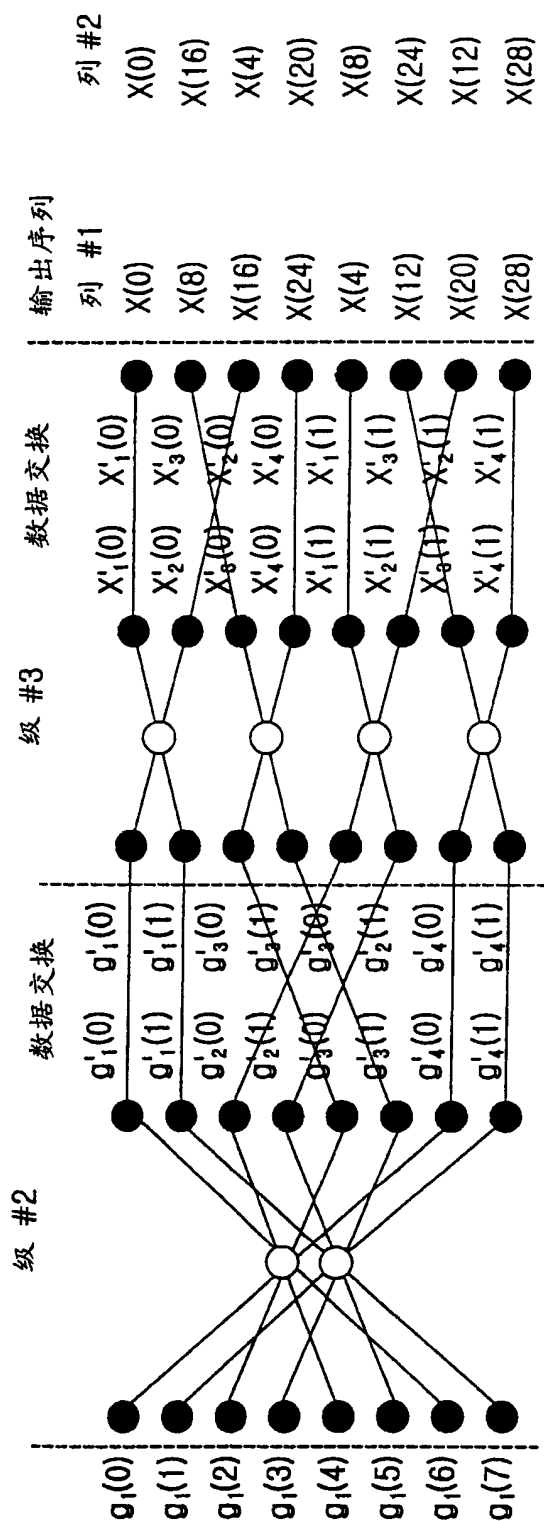


图 12

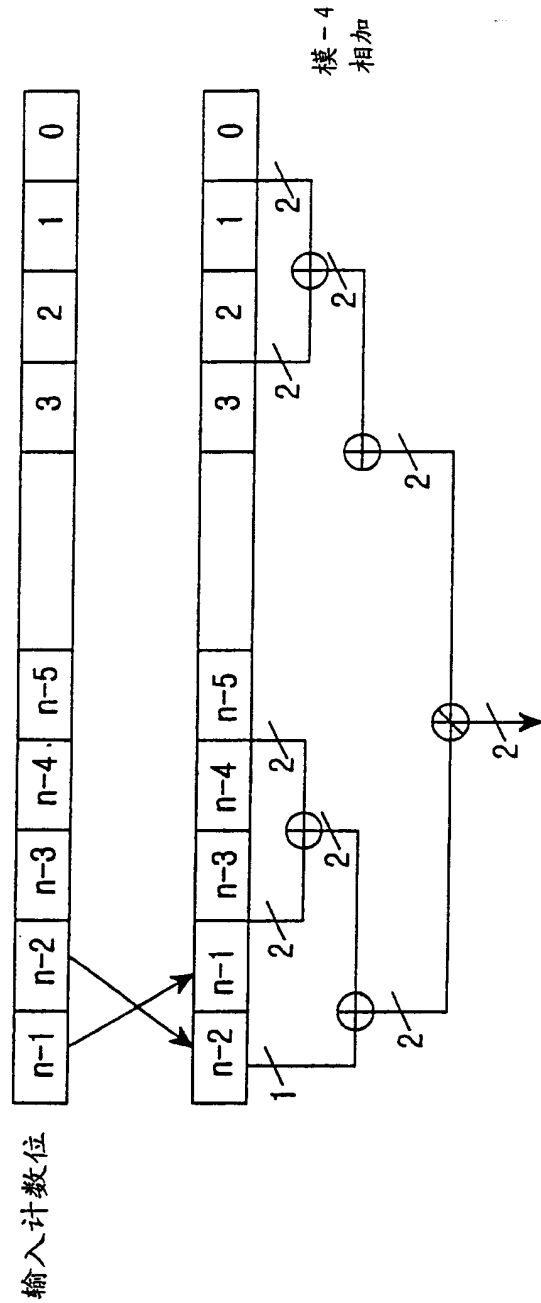


图 14