[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/30
G06F 9/318 G06F 9/302

[12] 发明专利说明书

[21] ZL 专利号 99816312.0

[45] 授权公告日 2004 年 9 月 8 日

[11] 授权公告号 CN 1165838C

[22] 申请日 1999.12.8 [21] 申请号 99816312.0
[30] 优先权
[32] 1998.12.30 [33] US [31] 09/223457
[86] 国际申请 PCT/US1999/029179 1999.12.8
[87] 国际公布 WO2000/041069 英 2000.7.13
[85] 进入国家阶段日期 2001.8.22
[71] 专利权人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 S·马金尼 S·基姆恩
G·B·多施 R·A·戈利维
审查员 袁丽颖

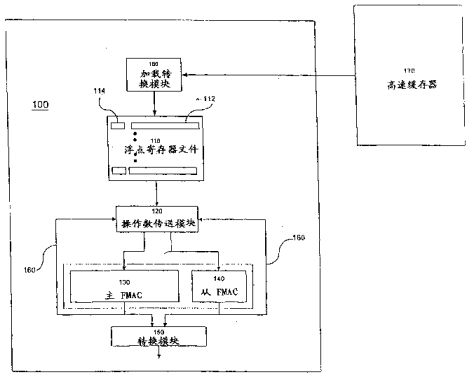
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 吴立明 王忠忠

权利要求书 2 页 说明书 14 页 附图 5 页

[54] 发明名称 用于在寄存器文件中存储分组操作数的方法和处理器

[57] 摘要

这一部分提供了把浮点操作数加载到拥有一个或者是一个以上的寄存器文件入口的方法。分组浮点操作数包括多个分量操作数。每一个分量操作数的有效位和幂位被拷贝到寄存器入口的相应字段中，幂位被检验以确定分量操作数是否被格式化。当分量操作数被格式化，分量操作数相应的隐含位就会被设置。



ISSN1008-4274

1. 一种方法，包括:

为寄存器文件入口检索分组浮点操作数;

将有效、阶和符号位从分组操作数的每个操作数写入到寄存器文件入口;

测试每个浮点操作数的阶位; 和

对于每个操作数，设置寄存器文件入口相关的一位对应于操作数的阶位的一个状态。

2. 权利要求 1 的方法，其中，测试每个浮点操作数的阶位包括从阶位中确定该浮点操作数为非标准或者 0。

3. 权利要求 2 的方法，其中，测试浮点操作数的阶位包括:

确定每个阶位是否为 0。

4. 权利要求 1 的方法，其中，寄存器入口是一个 82 位的寄存器入口，它容纳两个 32 位的浮点操作数并且如果阶位表示相关操作数为非标准，则每个位被设置成第一状态。

5. 一种将分组的操作数加载到浮点寄存器文件入口中的方法，该方法包括:

将分组操作数的每个分量操作数的有效位和阶位复制到寄存器文件入口中的相应字段中;

确定每个分量操作数是否为标准化的; 和

将与每个标准化的分量操作数相关的位设置为第一逻辑状态。

6. 权利要求 5 的方法，其中，确定每一个分量操作数是否为标准化包括测试分量操作数的阶位。

7. 权利要求 6 的方法，其中，测试阶位包括:

确定任何阶位是否是非 0; 和

当包括至少一个非 0 的阶位的时候，确认该分量操作数为标准化的。

8. 权利要求 7 的方法，进一步包括将与每个非标准化的分量指数相关的位设置为第二逻辑状态。

9. 一种处理器，包括:

一个数据高速缓冲存储器;

一个包括多个入口的浮点寄存器文件，其中每个入口有一个或多个相关联的位，表示入口的操作数是否为标准化的；和

一个加载转换器，它将分组浮点操作数中多个操作数中的每一个操作数的有效位和阶位从数据高速缓存器复制至寄存器文件入口之一，该加载转换器包括一个 OR 门以测试分组浮点操作数中的每一个操作数的阶位和以根据测试结果将相应的位设置为第一或者第二逻辑状态的逻辑。

10. 权利要求 9 的处理器，其中，加载转换器包括一个 OR 门，它拥有数目等于阶位数的输入端以测试每个操作数的阶位。

11. 权利要求 9 的处理器，其中，加载转换器包括每个分量操作数的测试逻辑。

用于在寄存器文件中存储分组操作数的方法和处理器

相关专利申请

本申请与美国专利申请 09/169.865“执行 SIMD 的标量硬件”，它于 1998 年 10 月 12 日提交，并转让给相同的受让人。

发明背景

技术领域 本发明涉及处理数据的系统，尤其是通过单指令多数据(SIMD)操作来处理数据的系统。

背景技术 处理器设计人员一直致力于研究可以增强多处理器运行性能的办法。并行处理多个操作数可以从当前高度优化的处理器中获取额外的运行性能。在某些常见的数学运算和图形运算中，相同的运算在每一个操作数上被重复执行。举例来说，在矩阵乘法中，第一矩阵的行元素与在第二矩阵上相应的列元素相乘，乘积被相加（乘－累加）。通过提供正确的时序排列和执行资源，乘－累加运算可能在多组行－列操作数上同时实现。这种方式叫做向量处理或者是单指令，多数据流处理，以便把它与标量或者是单指令，单数据流（SISD）处理区分开来。

为了有效实现单指令多数据流的运算，在通常情况之下，数据被提供到在“分组”数据格式中的执行资源中。举例来说，64 位处理器可能在分组数据块上运行，该数据块包括两个 32 位的操作数。在该示例中，向量乘－累加指令，V-FWA（f_1, f_2, f_3）将存储在寄存器 f_1 中的两个 32 位操作数与存储在寄存器 f_2 中相应的 32 位操作数相乘，并把乘积添加到存储在寄存器 f_3 中的两个运行乘积和中。换而言之，数据以分组格式存储在寄存器 f_1，f_2，f_3 中，该数据提供两个来自于每一个寄存器入口的操作数。如果处理器拥有足够的资源，它可能会同时处理两个或者是两个以上的分组数据块，例如四个或者是四个以上的 32 位操作数。32 位操作数被发送到并行处理并随后被分组的不同执行单元中。

甚至是在图形密集化和科学化的编程中，也并非所有的运算都是 SIMD 运算。大部分由通用处理器执行的软件包括执行标量运算的指令。

也就是说，每一个由指令指定的源寄存器存储一个操作数，每一个由指令指定的目标寄存器会接收一个操作数。在上述示例中，标量浮点乘－累加指令，S-FWA（f_1, f_2, f_3）可能把存储在寄存器 f_1 中的单个 64 位操作数与存储在寄存器 f_2 中相应的 64 位操作数相乘，并把乘积添加到存储在寄存器 f_3 中的运行乘积和中。由 S-FWA 指令处理的每一个操作数都以未被分组的格式提供到 FWAC 部件中。

寄存器文件把源操作数提供到执行单元中，并接收来自于执行单元的结果，会消耗大量的处理器晶圆(die)。可以取用的晶圆是大多数处理器芯片上非常宝贵的稀有资源。正是基于这个原因，在通常情况之下，处理器包括每一个主要数据类型的寄存器文件。举例来说，一般而言，处理器拥有一个存储分组和未分组浮点操作数的浮点寄存器。因此，虽然分组操作数包括两个或者是两个以上的分量操作数，还是设计了分组和未分组操作数，以适应相同大小的寄存器入口。

为分组和未分组操作数提供执行资源，就出现了对运行性能和费用的挑战。提供高性能的标量和向量处理的方式包括单个标量和向量执行单元这种方式的优势在于每一个标量和向量执行单元都可以以诸如分组和未分组的相应方式优化处理数据，而该方式的问题在于，额外的执行单元会消耗硅晶圆，而硅晶圆是相对而言比较宝贵的资源。

除了可以提供正确的执行资源，高性能处理器还必须包括有效传输分组和未分组操作数数据的结构。这种结构既包括那些把操作数数据传输到诸如高速缓冲存储器之类的处理器存储器层中的寄存器文件的结构，也包括把来自于寄存器文件中的操作数数据传输到执行资源中的结构。

本发明描述了当前可取用的单指令多数据流系统相关的各方面内容。

发明概述

本发明提供这样的一种系统，它通过为“活跃”操作数（operand "on-the-fly"）(也就是操作数被加载到寄存器文件入口中)设置隐含位以支持浮点操作数的有效处理。

根据本发明，浮点操作数被检索加载进入寄存器文件入口中。浮点操作数的选定位被检验，并且，当选定位处于第一种状态中，寄存器文件中的隐含位被设置。

在本发明的一种实施方案中，浮点操作数是包括两个或多个分量操作数的分组操作数，寄存器文件则包括每一个分量操作数的隐含位。当选定位显示出分量操作数被标准化的时候，分量操作数的隐含位就会被设置。

因此，当操作数被加载进入寄存器文件并通过相应寄存器文件入口中的隐含位被追踪的时候，本发明允许确定操作数的标准/非标准状态。这样就不再需要使用操作数传输模块中的状态确定逻辑部件，其中操作数传输模块把操作数从寄存器文件中传输到执行单元中。由于操作数传输模块位于执行单元的关键（旁路）通路上，处理器的运行性能就可以大大提高。

附图概述

以下视图可以作为更好地了解本发明地辅助参考，在这些视图中，相同的部件由相同的标号标识。这些视图用于说明本发明的选定实施方案，并不限制发明的范围。

图 1 表示根据本发明的浮点执行系统的方框图。

图 2A 和 2B 分别表示图 1 的寄存器文件中入口里的未分组和分组的操作数。

图 3 表示在分组操作数上的操作数传输模块的运算的方框图。

图 4 表示图 1 中操作数传输系统的一个实施方案的电路方框图。

图 5 表示图 1 中输出转换模块的一个实施方案的电路方框图。

发明详述

接下来的内容将详细介绍本发明的各方面细节，以便让读者对本发明有全面透彻的了解。但是，那些受益于公开内容的本领域的技术人员可以理解，本发明即使不依赖于这些特定细节也一样可以执行。此外，为了着重介绍本发明的特性，这里并没有对一些非常有名的方法，程序，部件和线路进行的详细介绍。

在通常情况之下，处理器结构指定在芯片上资源中存储数据的格式，诸如寄存器文件格式。选择这种寄存器文件格式以便适应处理器执行资源处理的各种不同数据类型，以及任何用于处理数据的补充信息。可适应的数据类型可以是，例如，IEEE 754－1985 指定的二进制浮点算法 IEEE 标准。寄存器文件格式支持这样一种有效处理，通过以随时准备被执行资源访问和处理的格式，来存储操作数数据。

对于标量处理来说，每一个操作数都是以寄存器文件格式作为未分组操作数存储的。在这里，“未分组”指的是允许寄存器文件入口中的数据表示不超过一个操作数这种数据格式。例如，处理器可能容纳一个单精度型未分组操作数，一个双精度型未分组操作数，或者是处理器寄存器文件格式的每一个寄存器文件入口一个双精度扩展型未分组操作数。对于向量处理来说，有多个分量操作数被提供到适应单个寄存器文件入口的分组操作数中。适应在一种尺寸大小的寄存器入口中的分组和未分组操作数意味着操作数被映射到不同的寄存器入口中。不同的映射拷贝会反映到从高速缓冲存储器传输到寄存器文件中的资源上，以及那些把操作数从寄存器文件传输到执行资源的结构中。

关于标量和向量运算的不同操作数格式还可能反映在它们自身的执行资源中。举例来说，一个有两个 32 位分量操作数的分组操作数可能适应两个 32 位执行单元来处理。而在相同系统中的未分组操作数可能由一个 64 位执行单元作为单个 64 位操作数来处理。在该示例中，每一个执行管道可以使用三个不同的执行单元，两个 32 位标量执行单元和一个 64 位标量执行单元，但是执行管道只能并行处理两个操作数。额外的执行单元会消耗大量宝贵的晶圆和资源。

提供两个标量执行单元的另一种方法就是修改标量部件，让它既可以处理标量操作数，也可以适应向量操作数。这种方式不需要使用向量执行单元。然而，以该种方式修改标量部件会降低它在未分组操作数上的运行性能。

本发明提供了这样一种系统，它可以有效处理分组和未分组格式的数据，而无需增加处理器的硅晶圆，或者是降低处理器在未分组操作数上的运行性能。加载转换模块确定活跃的分组操作数分量操作数的隐含位，就像把分组操作数加载进入在诸如高速缓冲存储器之类的存储器位置中的寄存器文件入口一样。隐含位与寄存器文件入口有关，它们表示相应的分量操作数是标准，非标准，或者是 0（标准化状态）。还可以确定未分组操作数的隐含位，尽管它们并不应用于这些操作数的随后处理中。

当指令指向寄存器文件入口的时候，操作数传输结构把分量操作数从分组操作数转换为适合标量执行单元处理的格式。操作数传输系统可能通过位处理来实现操作数转换，以避免用额外额外的逻辑门加

载系统。这会大大降低操作数转换对系统运行性能的影响，同时还能够保持标量执行单元在未分组数据上的运行性能。

在本发明的一个实施方案中，标量执行单元与向量执行单元结合在一起运行来处理分组操作数。转换过的操作数和未经转换的分量操作数分别被应用于标量和向量执行单元进行处理。在发明的实施方案中，操作数传输结构包括位处理踪迹，以及无需加载额外的执行资源就可以把分量操作数转换为标量格式的反向器。反过来，它可以保持处理器在标量运算上的运行性能。

标量执行单元可能是用于优化处理寄存器文件格式（RFF）中未分组操作数的浮点乘－累加模块（FMAC）。标量执行单元可能是优化用于处理由分组数据格式（PDF）的分组操作数的分量操作数的浮点乘－累加模块。操作数传输模块可能包括拥有额外引线的MUX，该引线可以用于把位重新处理踪迹传输到标量执行单元中，把分量操作数的选定位作为未分组操作数来修改的反向器。

图1是适合实现本发明的浮点系统100的方框图。系统100包括浮点寄存器文件110，操作数传输模块120，主浮点乘－累加模块130，从浮点乘－累加模块140，一个输出转换模块150和一个加载转换模块160。该视图中还有通过加载转换模块160把操作数数据提供导寄存器文件110中的高速缓冲存储器170。高速缓冲存储器170表示存储由浮点系统100和其他处理器资源（未被显示出）来处理的数据的分级存储器系统中的结构。在通常情况之下，数据以上述IEEE浮点标准指定的数据格式存储在存储器系统中。

在系统100的公布实施方案中，主FMAC130处理寄存器文件（“未分组”）格式的未分组操作数。从FMAC140处理分组操作数的分量操作数。操作数传输模块120把寄存器110中的数据以合适格式连接至FMAC130和FMAC140上。

在系统100的公开实施方案中，寄存器文件110包括多个寄存器入口112。在一个实施方案中，每一个入口112都有一个相关隐含位114，该隐含位被设置位表示存储在相关隐含位114中的数据是否被标准化。举例来说，在IEEE标准754中，隐含位被定义。隐含位可以与分组操作数中的分量操作数结合起来表示有待处理的操作数。举例来说，隐含位可以被设置为表示数据是标准格式。

在标量运算中，操作数传输模块 120 从寄存器文件 110 提供未分组浮点操作数至主 FMAC130。未分组操作数以寄存器文件格式被存储于寄存器文件 110 中。在向量运算中，操作数传输模块 120 检索来自于寄存器文件 110 的分组操作数，把分量操作数转变为未分组操作数，并把它提供到主 FMAC130。第二分量操作数被提供到第二 FMAC140 上。这样，主 FMAC130 就在标量运算和向量运算之间被共享，而第二 FMAC140 则为标量运算提供额外的执行资源。

由 FMAC130 和 140 生成的结果被连接到输出转换模块 150 上，以便再一次结合成分组数据格式。在本发明的一个实施方案中，旁路 160 将来自于主 FMAC130 的输出数据在重分组以前连接至操作数传输模块 120 上。被旁路数据可能会由主 FMAC130 进行额外处理，而无需由转换模块 150 对其进行重新分组，以及随后由操作数传输模块 120 对其进行解分组。旁路循环避免了来自于旁路通路的输入转换（分组到未分组数据中）。

视图中的系统 100 拥有一个浮点管道，但是本发明并没有仅仅只局限于单个管道。它可能被复制到一个或者是一个以上的管道中以提供超标量浮点单指令多数据运算。换而言之，两个或者是两个以上的向量指令可能会被并行处理。广而言之，本领域的技术人员可以理解本发明可以在不同于系统 100 的配置的浮点系统上实现。

图 2A 显示了可供本发明使用的，寄存器文件格式的（RFF 操作数）未分组操作数 200 的实施方案。RFF 操作数 200 包括一个有效字段 210，幂字段 220，和符号字段 230。在本发明的实施方案中，有效字段 210 是 64 位，幂字段 220 是 17 位，符号字段 230 是 1 位。

图 2B 显示了可供本发明使用的，分组文件格式的（PDF 操作数）分组操作数 250 的实施方案。PDF 操作数 250 的公布实施方案分别包括第一和第二分量操作数 260（a）和 260（b）。在接下来的讨论中，除非需要索引来区别特殊分量操作数 260，否则它将会被删除。在 PDF 操作数 250 中还显示了不使用的位 270。不使用位 270 被添加到分量操作数 260 中以填满寄存器文件入口。每一个分量操作数 260 包括尾数字段 262，幂字段 264 和符号字段 266。在一个实施方案中，尾数字段 262 是 23 位宽，幂字段 264 是 8 位宽，符号字段 266 是 1 位宽。在公布的实施方案中，每一个分量操作数都以诸如 IEEE 标准 754-1985

所指定的单精度浮点格式出现。

图 2A 和图 2B 中还显示了（一个或者多个）隐含位 114，与每一个寄存器文件入口 112（图 1）相关。隐含位 114 可以应用于表示分量操作数是 0 还是非标准。在 FP 寄存器文件 110 的一个实施方案中，隐含位可以被确定为写入寄存器文件 110 中的寄存器入口 112 中的数据，也就是活跃的。这样就不再需要使用在操作数传输模块 120 中的额外逻辑部件以确定操作数的标准/非标准状态了，因为这只会减缓向量操作数的处理。举例来说，衡量将要传输到 FMAC140 的分量操作数的标准/非标准状态需要使用在操作数传输系统 120 的关键通路上的 OR 门。

对于分别存储在图 2A 和图 2B 中的未分组和分组操作数的寄存器文件入口 112 来说，隐含位 114，例如当数据被写入寄存器文件 110，位 82 和位 83 就会被设置为以下形式：

IF(DATA[62:55]=' 0::8),THEN data[83]=' 0,ELSE' 1

IF(DATA[30:23]=' 0::8),THEN data[82]=' 0,ELSE' 1

这里，'1 和'0 分别表示二进制 1 和二进制 0。

在系统 100（图 1）的一个实施方案中，加载转换模块 160 可能使用两个 8 输入 OR 门来实现上述运算。举例来说，加载转换模块 160 的一个实施方案包括传输阶位，有效位和符号位从高速缓冲存储器 170 传输到它们在寄存器文件的目标入口 112 中相应的位字段的踪迹。每一个 OR 门的输入数据都被连接到分量操作数的一个阶位踪迹上；而每一个 OR 门的输出数据则被连接到与分量操作数相关的隐含中。在这个实施方案中，如果分量操作数的阶位是非 0 的话，隐含位将会被设置为 1。

隐含位可以由操作数传输模块 120 使用以提供以适合处理的格式的分量操作数 260。举例来说，分量操作数 260 的尾数字段 262 容纳 23 位。24 位的有效字段就可能通过把相关的隐含位附加到 23 位尾数，作为最有效字段。在公布的实施方案中，这种转换并不需要 RFF 数据 200。但是，逻辑部件通过确定每一个被写入寄存器文件 110 的操作数，并在 RFF 数据 200 被访问的时候忽略它们得以简单化。

在本发明的一个实施方案中，一个分量操作数 260 被转换为 RFF 操作数，由主 FMAC130 处理，而另一个分量操作数被提供到第二 FMAC140 中而处理。

图 3 是由操作数传输模块 120 实现的运算示意图，以便把正确的格式化数据提供到 FMAC130 和 FMAC140 的框架。在公布的实施方案中，隐含位 114 与分量操作数 260（a）中的数据结合在一起，该分量操作数被转换为 RFF 操作数以备主 FMAC 处理。本发明的一个特性在于这个转换处理可以无需加载主 FMAC130 而实现，这样，就不会降低它在标量运算上的运行性能。分量操作数 260（b）被提供到第二 FMAC 而处理，第二 FMAC 可以优化而处理分量格式的操作数。FMAC130 并不需要显示分组操作数的上层分量操作数（260（a））。广而言之，分组操作数的任何分量操作数都可能会被选择发送到 FMAC130 中。

在本发明的一个实施方案中，操作数传输模块 120 通过把把分组操作数位处理成未分组操作数 200 来实现数据转换。举例来说，来自于尾数字段 262（a）和隐含位 114（I1）的数据可能被转换到 RFF 有效数据（有效字段 210）。同样的，来自于幂字段 264 的数据可能被转换到 RFF 幂数据（幂字段 220）。符号字段 266 中的数据可能被映射到 RFF 符号位（符号字段 240）。在接下来的内容中，将使用在图 2A 和图 2B 中的示例数据来详细介绍各种不同的转换步骤。

有效转换 分量操作数 260（a）的示例实施方案包括 23 位尾数字段 262（a）。在本发明的一个实施方案中，来自于尾数字段的数据通过：（1）预先把相关隐含位 114 挂起到来自于部件尾数字段 262 的 23 位尾数；以及（2）把二进制零附加到尾数的最不有效位上以便形成 64 位的 RFF 有效字段这两种方式转换为 RFF 有效字段。表 1 概括了尾数的位处理，它表示出到寄存器入口 112 中主要（1）FMAC130 的位（一个或者是多个）输入数据，位的特性，标量指令的源位，以及向量指令的源位。

表 1

第一 FMAC 输入处的位	位功能	标量指令的源位	向量指令的源位
[63]	隐含位	[63]	[83]
[62: 40]	上有效位	[62: 40]	[54: 32]
[39: 0]	下有效位	[39: 0]	0: : 40
[81]	符号位	[81]	[63]

正如我们上面所介绍的那样，当这些操作数共享寄存器文件 110 中同样的印记（footprint），也就是寄存器入口 112，它们的操作数就会被映射到寄存器入口 112 的不同位上。

在 RFF 操作数 200 和 PDF 操作数 250 的公布实施方案中，隐含位分别由位 63 和位 83 指定。为了适应这两种数据格式，操作数模块 110 会使用一个 2:1 的复用器（MUX），选择输入 FMAC130 的正确源位。把额外的 2:1MUX 介绍到在寄存器文件 110 和 FMAC130 之间的逻辑链，加载操作数传输模块 120，会减缓到 FMAC130 的数据传输速度。加载过程会削减标量和向量运算的运行性能。在本发明的实施方案中，是通过向操作数传输模块 110 中的现存的 3:1MUX（图 4）提供旁路数据（旁路 4:1MUX）的额外通路实现位处理的。这样就不再需要把额外的 MUX 添加到在寄存器文件 110 和 FMAC130 之间的数据通路了。

幂转换 在公布的实施方案中，RFF 和 PDF 幂根据不同的偏差来表示。在把 PDF 幂（字段 264）转换为 RFF 幂（字段 220）的时候，就需要考虑偏差之间的区别了。在一种实施方案中，RFF 偏差值是 FFFFh，PDF 偏差值是 7Fh，也就是 IEEE 标准中单精度实数的幂偏差。这些值之间的区别是 FF80h。

进行幂转换的一种方式就是把 FF80h 添加到 PDF 幂字段 264 中的 8 位的幂值，以获取 RFF 幂。问题在于它使用的是在寄存器文件 110 和主 FMAC130 之间数据通路上的加法器。操作数传输模块 120 上的额外的选通延迟会降低系统 100 的标量和向量运算性能。而使用另一种去调整幂偏差的方式就可以不必使用加法器。

表 2 概括了当偏差区别是 FF80h，把 PDF 幂转换为 RFF 幂的时候，所执行的调整处理。在这里，通过 E7 的 E0 是 PDF 幂字段 264（a）上的 8 位，并且如果是以二进制方法表示的话，0111111111000000 是 FF80h。正如在表 2 中所表示的那样，幂转换可以通过反转 PDF 幂字段 264（a）上的第 8 阶位（E7->E7），在 RFF 幂（字段 220）的下一个 9 位对它进行复制，并把未反转的第 8 位（ $\overline{E7}$ ）拷贝到 RFF 幂中最高位位置上。这些操作可以通过一个反向器和适当的路由踪迹来完成。这里并不需要加法器，并且在操作数传送模块 120 和主 FMAC130 上的反向器的性能影响并不重要。

表 2

位位置	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIMD 幂										E7	E6	E5	E4	E3	E2	E1	E0
常数	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
结果	E7	-- E7	-- E7	-- E7	-- E7	-- E7	-- E7	-- E7	-- E7	-- E7	E6	E5	E4	E3	E2	E1	E0

在表 2 中总结的方法并不用于单精度非标准化数字的操作。非标准数值可以由关联的隐含位 114 的值定义，而转换过程可以使用一个软件处理程序来实现。表 3 概括了上面所讨论的阶位处理。

表 3

第一 FMAC 输入处的位	位功能	标量指令的源位	向量指令的源位
[6: 0]	下阶位	[70: 64]	[61: 55]
[15: 7]	中阶位	[79: 71]	[62#: : 9]
[16]	上阶位	[80]	[62]

在这里，“#”指的是该位是被反转了。主 FMAC 130 的阶位是从幂字段的第一位开始计算的。

图 4 表现了操作数传送模块 120 的一个实施方案，它适合于在分别拥有第一和第二浮点管道，pipe0 和 pipe1，的处理器中使用。本发明并不仅限于在处理器中采用的管道的特殊的号码。操作数传送模块 120 的公布的实施方案包括一个 4: 1 的先旁路 MUX（EBPM）410，一个 6: 1 的中旁路 MUX（MBPM）420，以及一个 4: 1 的后旁路 MUX（LBPM）430。对于上面所描述的实例数据格式，操作数传送模块 120 将了一个 32 位的分量操作数 260（a）从分组的或者 PDF 操作数 250 转换为 82 位复原的或者 RFF 操作数 200。

EBPM 410 在输入端 A 和 B 处接收所存储的 pipe0 和 pipe1 的数据。从 pipe0 和 pipe1 的后端写回数据可以通过输入端 C 和 D 连接至操作数传送模块 120。EBPM 410 的一个控制输入端从输入端 A，B，C 或 D 中选择提供给 MBPM 420 的数据。MBPM 420 在输入端 E 处接收来自 EBPM 410 的数据。来自 pipe0 和 pipe1 的一个级的数据被分别从输入端 F 和 G 处接收。来自于 pipe0 和 pipe1 的加载转换器的数据分别从输入端 H 和 I 处被接收。来自寄存器文件 110 的数据在输入端 J 处被接收。

MBPM 420 的一个控制输入端从输入端 E，F，G，H，I 和 J 之一选择提供给 LBPM 430 的数据。

LBPM 430 从输入端 M 和 N 处分别接收来自于 pipe0 和 pipe1 的另一个级的旁路数据。LBPM 430 的输入端 K 通过位处理块 440 接收来自 MBPM 420 的数据，该位处理块可以为矢量操作实现上面所描述的转换。在上面的例子中，为处理块 440 包括一个反向器和位处理踪迹来将数据从上级的分量操作数 269（a）重新标准化为 RFF 数据。举例来说，位处理块包括一个逻辑部件和踪迹将 23 位 PDF 尾数位转换为一个 64 位 PFF 有效位，并将 8 位 PDF 阶位转换为一个 17 位带有经过再调整偏差的 RFF 阶位。对于标量运算来说，LBPM 430 的输入端 L 可以无需介入位处理或者反向而接收来自于 MBPM 420 的数据。来自于输入端 K 或者 L 之一的数据依据正在处理的分别是分组数据还是写回数据被提供给主 FMAC 130。

在图 4 的实施方案中，转换（位处理）数据和未转换数据通过分别选择适当的输入端 K 或者 L 被提供给主 FMAC 130。这允许来自寄存器文件 110 的数据或者与先旁路级(early bypass stage)相关的数据在 LBPM 430 处被转换。在公布的实施方案中，在输入端 M 和 N 处的末级旁路数据可以通过使用相应的位处理块 440 复制每个输入端来进行转换。然而，如果位于输入端 M 和 N 处的旁路数据并不优先于后旁路(late bypass)被分组的话，就不需要对它进行分组了。这样就可以避免在 LBPM 430 上使用额外的支路和位处理块 440。

对于操作数传送模块 120 的一个实施方案来说，后旁路数据可以通过旁路 160（图 1）提供给 LBPM 430。在被重分组为例如 PDF 的格式 250 之前，旁路 160 分别捕获主和从 FMAC 130 和 140 的输出。在这个实施方案中，对于旁路数据来说并不需要 LBPM 430 或者它的输出端的扩展。

图 5 显示了一个输出转换模块 150 的实施方案。公布的输出转换模块实施方案包括一个主输出 MUX 510 和位处理块 520。主输出 MUX 510 包括用来接收来自主 FMAC 130 的结果的输入端和不同的资源（FMISC）154，以及一些特殊的结果编码。对于实例的数据格式，FMAC 130 为标量和矢量运算提供结果以作为未分组（RFF）操作数 200（82 位）。在后面的情况下，位处理块 520 将来自于主 FMAC 130 的未分组操作数

与来自于次级 FMAC 140 的分量操作数结合在一起并形成一个分组操作数。

提供给主输出 MUX 510 的特殊编码显示着寄存器文件格式中的特殊结果，诸如 0、无穷大、或者最大实数（最大可描述的浮点值）。对于公布的实施方案来说，这些编码可以使用上面所描述位处理运算在必要时转换为矢量运算。如果 需要对 0、无穷大、和最大实数结果进行更精确的转换，它们可以在操作数传送模块 120 的初级阶段中实现，因为这里的必要信息在浮点执行管道中能够相对较早地被利用。在关键通路中没有引入附加的延迟。

对于矢量运算来说，主 FMAC 130 和次级 FMAC 140 的结果被重分组以提供结果作为 PDF 操作数。在公布的实施方案中，82 位入口的低 32 位接收来自次级 FMAC 140 分量操作数结果。接下来最高的 32 位由来自 FMAC 130 的未分组操作数通过位处理提供。高 18 位接收由执行情况决定的内容，在公布的实施方案中，这些是 '1&1003Eh。

由主 FMAC 130 提供的分组（RFF）指数有一个相对于分组操作数 250 中的分量操作数采用的 IEEE 单精度格式的 FF80h 的超额偏差。当转换为分组的分量操作数结果的时候，超额偏差转换就会从未分组操作数结果中被减去。在实例实施方案中，这可以通过下面的方法来实现：将 2 的补码与 17 位 RFF 指数相加，使用结果的低 8 位作为分量操作数的阶位。如以前一样，完成这些为加载浮点管道加载附加的加法器，使用位处理运算就可以了。具体如表 4 所示。

表 4

位位置	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFF 幂	E16									E7	E6	E5	E4	E3	E2	E1	E0
常数	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
结果		--	--	--	--	--	--	--	--	-- E7	E6	E5	E4	E3	E2	E1	E0

如在表 4 中所显示的那样，分量操作数的阶位可以通过由主 FMAC 130 提供的未分组操作数获得。在上面的实例中，它们是由未分组操作数结果的低 8 位阶位提供的，其中 8 位是被反向了的。在转换运算中忽略的高 8 位在表 2 中并没有显示。

表 5 概括了将来自于主 FMAC 130 的未分组操作数结果与来自于次

级 FMAC 140 的分量操作数结果分组为矢量运算提供分组操作数结果的位处理运算。

表 5

分组操作数位	PO 位功能	来自第一 FMAC 的结果位	来自第二FMAC 的结果位
[83]	第一 CO 的隐含位	[63]	-
[82]	第二 CO 的隐含位	-	[23]
[81]	未使用常数（RFF 中的符号位）	-	-
[80: 64]	未使用常数（RFF 中的阶位）	-	-
[63]	第一 CO 的符号位	[81]	
[62]	第一 CO 幂的 MSB	-- [71]	-
[61: 55]	第一 CO 的剩余阶位	[70: 64]	-
[54: 32]	第一 CO 的尾数	[62: 40]	-
[31]	第二 CO 的符号位	-	[32]
[30: 23]	第二 CO 的阶位	-	[31: 24]
[22: 0]	第二 CO 的尾数	-	[22: 0]

在这里，“ CO”指的是分量操作数，“来自于第一 FMAC 的结果位”指的是由主 FMAC 130 产生的分组操作数结果，“来自于第二 FMAC 的结果位”指的是由次级 FMAC 130 产生的分量操作数结果。

对于这样的由 FMAC 执行的指令来说，附加的控制信号被提供给 FMAC 来更改单元的运算。举例来说，FMAC 可以被用来把浮点操作数转换为整型操作数。转换预算可以对矢量和标量操作数同时适用。矢量和标量所要求的移位操作是不同的，因此一个分开的控制信号被提供来调整 FMAC 运算。前面讨论过的多重累加运算，它包括矩阵乘法和协同转换，它可以无需这样的 FMAC 单元内部的更改而进行处理。

提供给系统进行处理 SIMD 或者矢量运算的就是使用标量和矢量执行单元的结合。必须减少必须添加到系统中的附加执行硬件总数，而标量执行硬件的性能必须被保持。该系统采用一个寄存器文件，该文

件有每个分组操作数的分量操作数的隐含位。一个加载转换单元包括一个逻辑部件，它可以在加载运算期间测试每个分量操作数为非标准化的还是 0，它还可以根据这个来设置相关的隐含位。为每个系统中的管道提供了一个标量和一个矢量执行单元。一个操作数传送模块将来自于寄存器文件的数据提供给标量和矢量执行单元。对于 SIMD 运算来说，操作数传送单元从寄存器文件中检索分组操作数，通过位处理将分量操作数转换为未分组操作数，将该未分组操作数提供给标量执行单元以备处理，另一个分量操作数被提供给矢量执行单元。位处理逻辑部件包括轨迹路由选择和反向器，它为标量和矢量运算而对系统性能造成的影响是最小的。

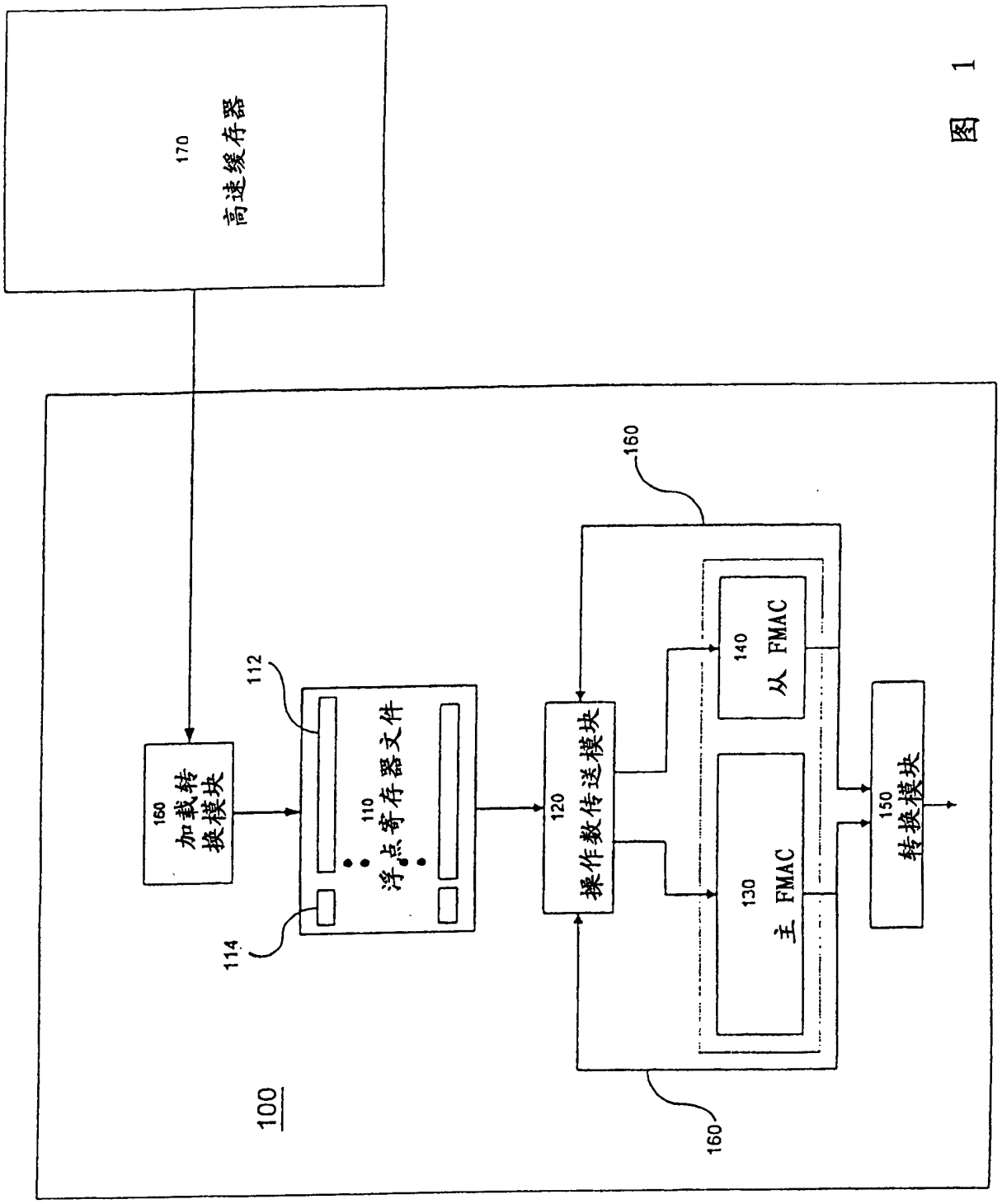

170
高速缓存器
160
加载转换模块
112
110
浮点寄存器文件
114
120
操作数传送模块
160
140
从 FMAC
130
主 FMAC
150
转换模块
160
100

图 1

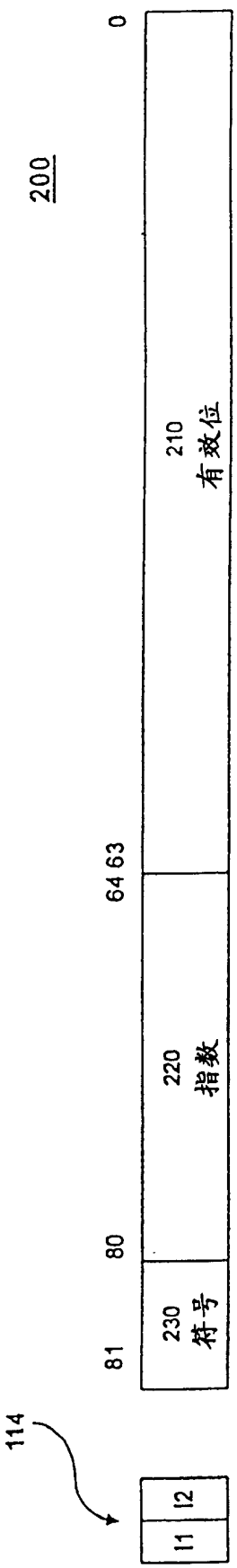

200
0
64 63
80
81
210
有效位
220
指数
230
符号
114
I1
I2

图 2A

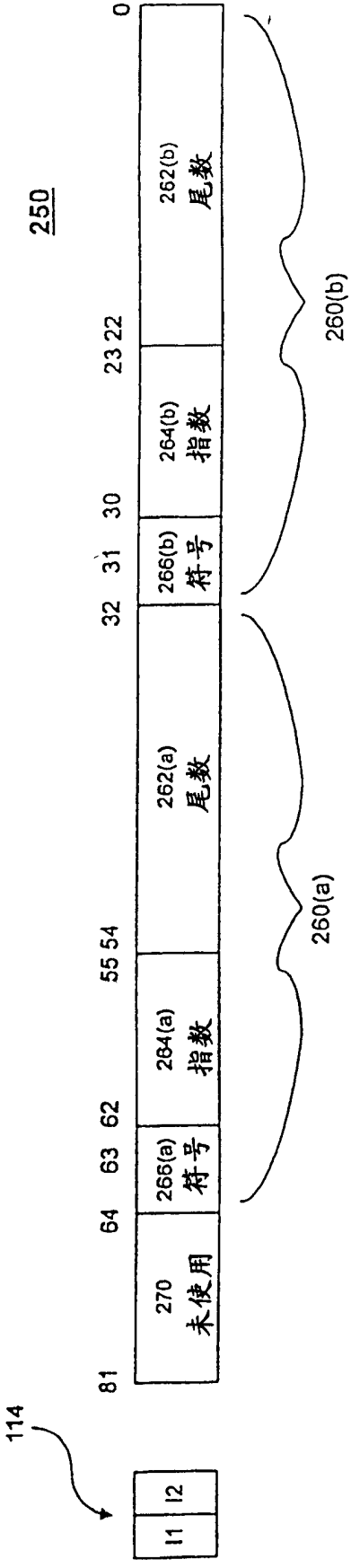

250
0
23 22
30
31
32
55 54
62
63
64
81
262(b)
尾数
264(b)
指数
266(b)
符号
260(b)
262(a)
尾数
264(a)
指数
266(a)
符号
260(a)
270
未使用
114
I1
I2

图 2B

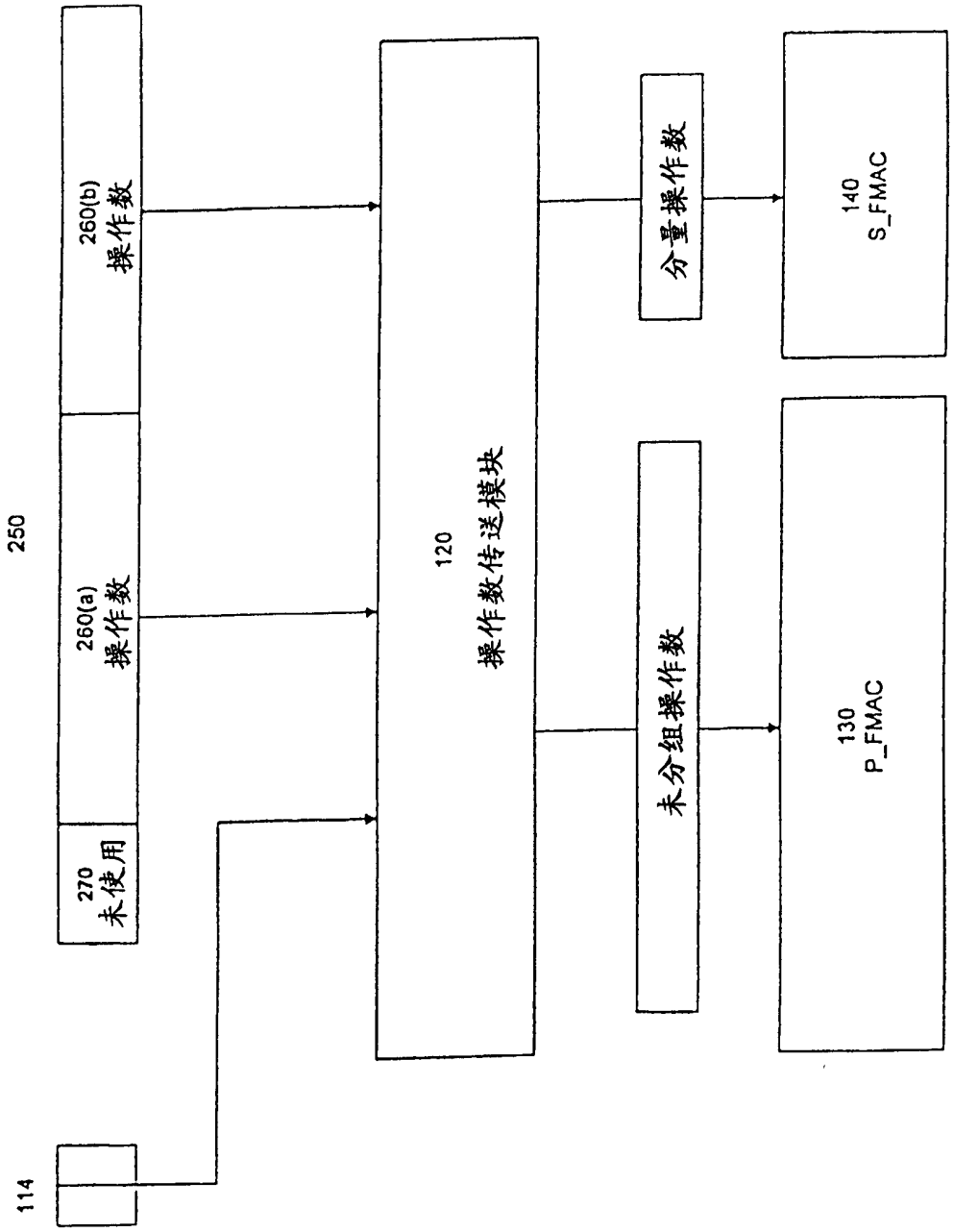

250
260(b)
操作数
260(a)
操作数
270
未使用
114
120
操作数传送模块
分量操作数
未分组操作数
140
S_FMAC
130
P_FMAC

图 3

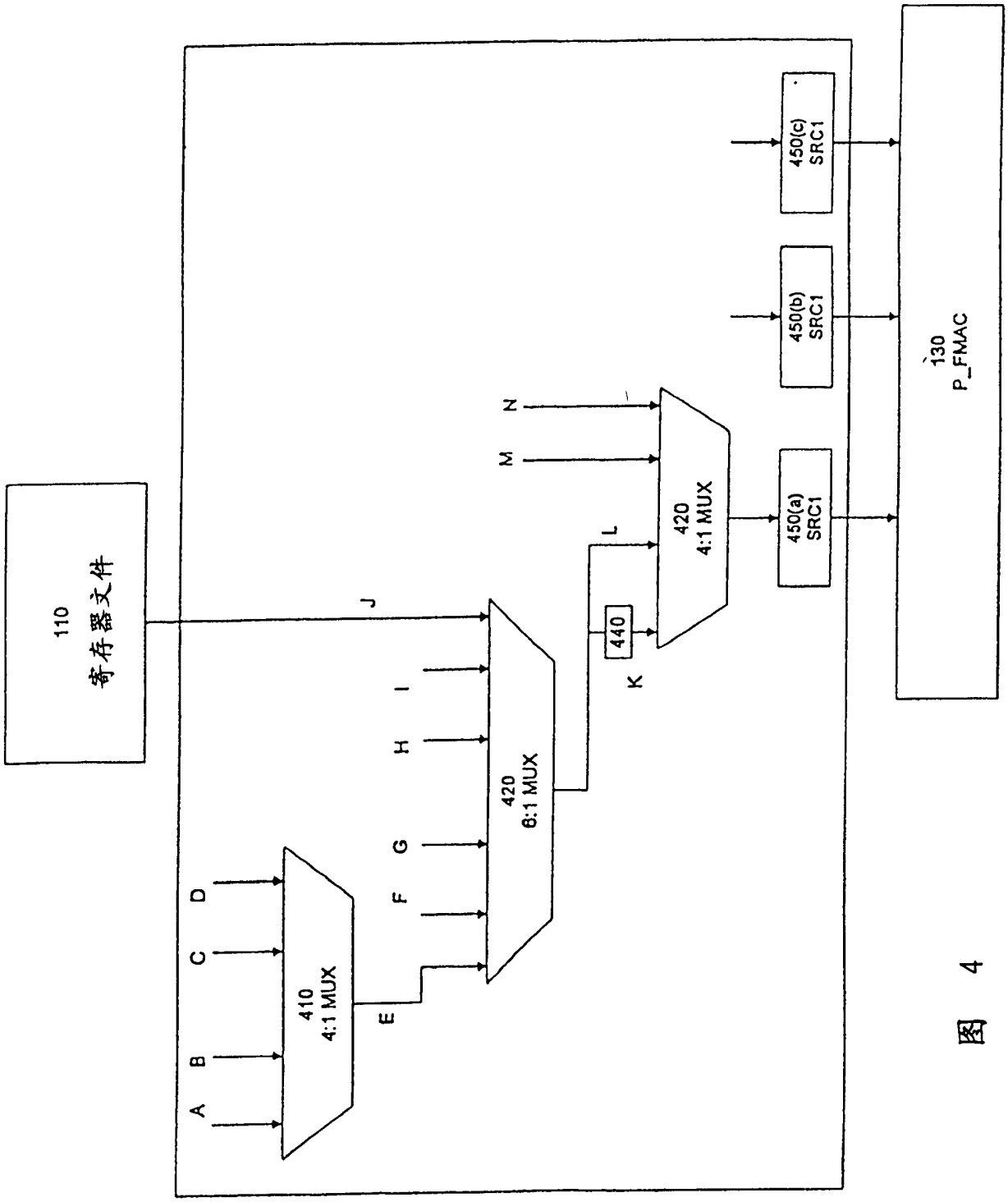

110
寄存器文件
J
A
B
C
D
410
4:1 MUX
E
F
G
H
I
420
8:1 MUX
K
440
L
M
N
420
4:1 MUX
450(a)
SRC1
450(b)
SRC1
450(c)
SRC1
130
P_FMAC

图 4

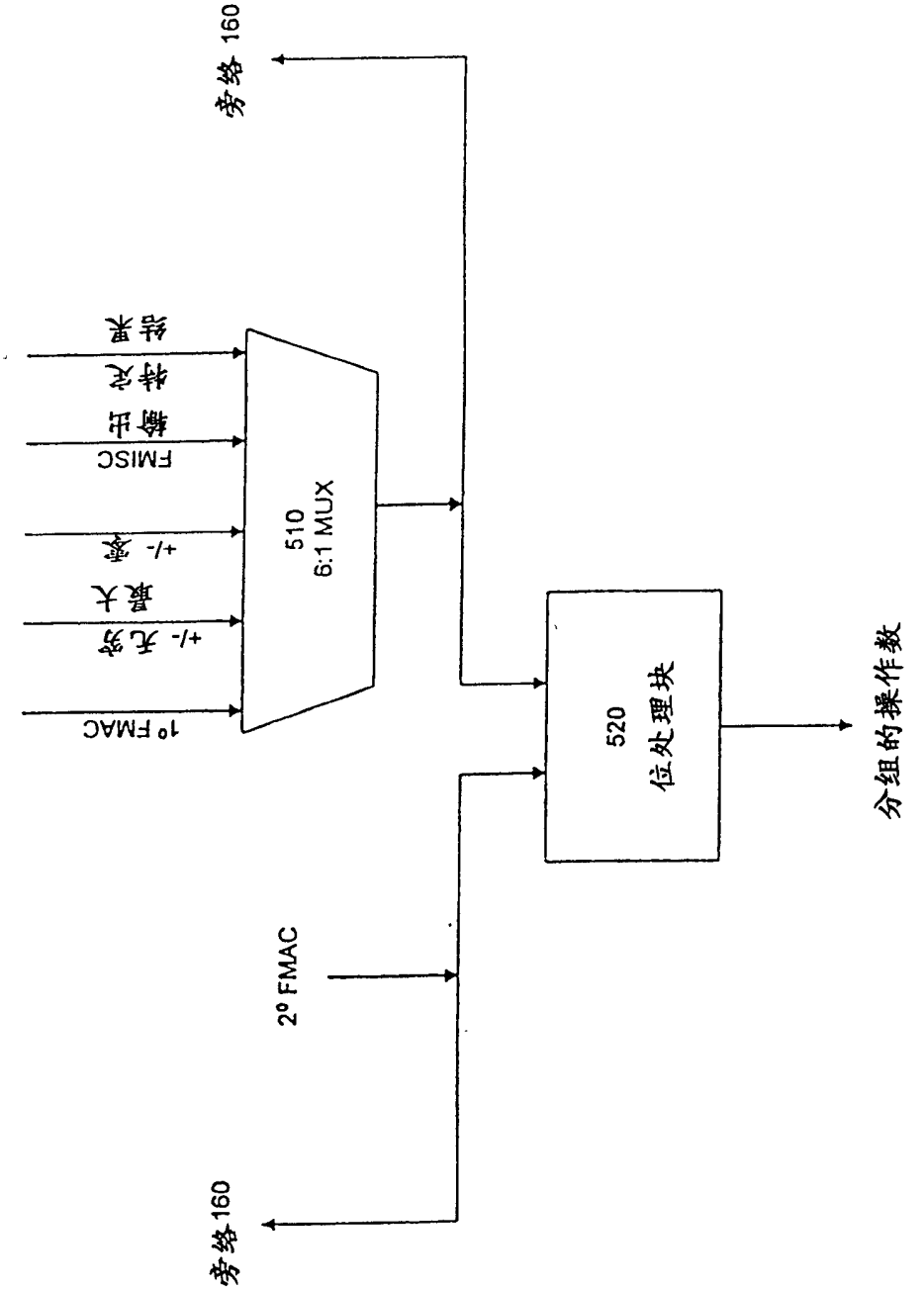

旁络 160
510
6:1 MUX
结果
特定
FMISC
输出
+/- 零
最大
+/- 无穷
1° FMAC
520
位处理块
分组的操作数
2° FMAC
旁络160

图 5