(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106339202 A
(43)申请公布日 2017.01.18

(21)申请号 201610726151.4

(22)申请日 2015.06.24

(30)优先权数据
62/020,246 2014.07.02 US
62/173,808 2015.06.10 US

(62)分案原申请数据
201580003388.3 2015.06.24

(71)申请人 上海兆芯集成电路有限公司
地址 201203 上海市浦东新区张江高科技园区金科路2537号301室

(72)发明人 汤玛士·艾欧玛

(74)专利代理机构 北京林达刘知识产权代理事务所(普通合伙) 11277
代理人 刘新宇

(51)Int.Cl.
G06F 7/483(2006.01)
G06F 7/499(2006.01)
G06F 7/544(2006.01)
G06F 9/30(2006.01)
G06F 9/38(2006.01)

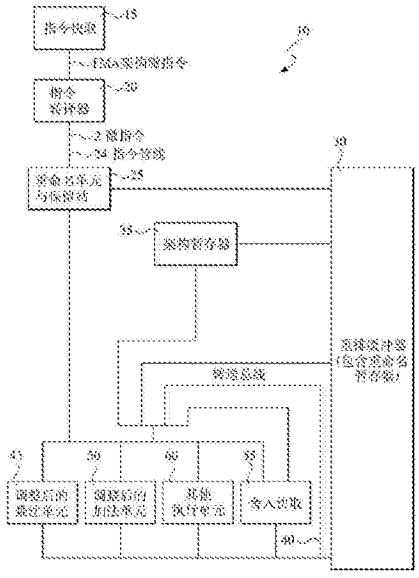
权利要求书3页 说明书29页 附图10页

(54)发明名称

微处理器及其方法

(57)摘要

一种微处理器及其方法，该微处理器包括一指令管线、一共享存储器、以及在指令管线中的第一与第二算术处理单元；其中，每一处理单元自共享存储器读取运算元并写入结果；第一算术处理单元执行一数学运算的一第一部分、以产生一中间结果向量，其不为数学运算的一完全与最终的结果；第一算术处理单元亦产生多个非架构运算控制指标，以指明自中间结果向量产生一最终结果的接续运算该如何进行；第二算术处理单元执行数学运算的一第二部分，并依据运算控制指标产生数学运算的一完全与最终的结果。



CN 106339202 A

1.一种微处理器,包括:

一指令管线;

一共享存储器;以及

在指令管线中的第一算术处理单元与第二算术处理单元;

其中,每一该第一算术处理单元与该第二算术处理单元自共享存储器读取运算元并写入结果;该第一算术处理单元执行一数学运算的一第一部分,以产生一中间结果向量,该中间结果向量不为该数学运算的一完全与最终的结果;

该第一算术处理单元亦产生多个非架构运算控制指标,以指明自该中间结果向量产生一最终结果的接续运算该如何进行;

该第二算术处理单元执行该数学运算的一第二部分,并依据该运算控制指标产生该数学运算的一完全与最终的结果。

2.根据权利要求1所述的微处理器,其中,该中间结果向量为一未舍入值,且该完全与最终的结果为一舍入值。

3.根据权利要求1所述的微处理器,其中,该数学运算为一±A*B±C形式的融合浮点乘积-相加运算,其中该A、该B与该C为浮点输入运算元,且在该C相加至该A与该B的乘积前,并未发生舍入运算。

4.根据权利要求1所述的微处理器,其中,该数学运算为一乘法乘积-相加运算,且该微处理器还包括一翻译器或一ROM,该翻译器或该ROM将一不可切割的、一体的乘积-相加指令转换为至少第一微指令与第二微指令,其中该第一微指令的执行能够产生该中间结果向量,而该第二微指令的执行则能够使用该中间结果向量产生该完全与最终的结果。

5.根据权利要求1所述的微处理器,其中,该运算控制指标包括舍入指标,用以提供充足信息使该第二算术处理能于在该中间结果向量上执行该数学运算的该第二部分后、产生一正确的舍入完全与最终的结果。

6.根据权利要求1所述的微处理器,其中,该第一算术处理单元将该中间结果向量存入至一暂存器中,并将该运算控制指标存入至一运算控制指标快取中,而该第二算术处理单元则自该暂存器下载该中间结果向量,且自该运算控制指标快取下载该运算控制指标。

7.根据权利要求1所述的微处理器,其中,该微处理器将该中间结果向量传送至该第二算术处理单元。

8.根据权利要求1所述的微处理器,其中,该数学运算的该第一部分包括二输入运算元的至少一乘积。

9.根据权利要求8所述的微处理器,其中,该数学运算的该第一部分与该第二部分其中之一还包括一具有一第三运算元的相加运算,只要该二输入运算元与该第三运算元的数值运算元满足一或多个预定状况中的至少一个。

10.根据权利要求1所述的微处理器,其中,该数学运算的该第二部分包括至少一舍入子运算。

11.根据权利要求1所述的微处理器,其中,该数学运算的该第一部分与该第二部分其中之一还包括一相加运算子运。

12.一种微处理器中的方法,用以执行一±A*B±C形式的融合乘积-相加运算,其中的该A、该B与该C为输入运算元,该方法包括:

计算该运算元A与该运算元B的部分乘积；

自下列之一中的第一相加运算产生产生一原始结果：(a)该运算元A与该运算元B的该部分乘积、或(b)具有该运算元A与该运算元B的该部分乘积的该运算元C；

选择该原始结果的多个最重要位，以加入一未舍入中间结果向量；

将多个最不重要位缩减至一或多个舍入指标；

如果该第一相加运算并未包括该运算元C，则执行具有该未舍入中间结果向量的该运算元C的一第二相加运算；以及

使用该舍入指标产生该乘积-相加运算的一最终舍入结果。

13.根据权利要求12所述的方法，还包括将该一或多个舍入指标存入一舍入快取的步骤。

14.根据权利要求13所述的方法，其中，该一或多个舍入指标包括保护位、舍入位与/或黏位。

15.根据权利要求12所述的方法，还包括将该未舍入中间结果向量存入至一能够供多个指令执行单元存取的共享储存空间中。

16.根据权利要求12所述的方法，还包括：如果包括该运算元C与该未舍入中间结果向量的该第一相加运算为正值，则产生一端回进位值，以指明尚有一端回进位修正悬而未决。

17.根据权利要求12所述的方法，其中，该未舍入中间结果向量包括一中间尾数结果与一中间结果指数值，该中间结果指数值为该C的一指数与该运算元A和B的指数值的总和的函数中较大者的正常化表示。

18.根据权利要求17所述的方法，其中，该未舍入中间结果向量亦包括一中间符号指标，该中间符号指标作为一是否该第一相加运算包括该运算元C、该乘积-相加运算是否为一有效减法、及是否有悬而未决的端回进位的函数而产生。

19.根据权利要求17所述的方法，还包括产生中间不足位与中间溢位指标，以指明该中间结果指数值是否在一能够代表或所欲指数范围之上或之下。

20.根据权利要求12所述的方法，还包括产生一中间符号指标，该中间符号指标作为一包括是否该第一相加运算包括该运算元C、该乘积-相加运算是否为一有效减法、及是否有悬而未决的端回进位的函数而产生。

21.根据权利要求12所述的方法，还包括：在执行一具有该运算元C的第一相加运算前，将一乘数单元部分乘积加总树中的该运算元C的选择性补足尾数对准。

22.根据权利要求12所述的方法，其中，该未舍入中间结果向量以一浮点数字用的标准IEEE格式来表示，该未舍入中间结果向量为一尾数，且具有一相等于该融合乘积-相加运算一目标结果位数的位数。

23.根据权利要求12所述的方法，在执行任何用以产生该最终舍入结果的第二相加运算前，还包括产生一或多个额外舍入指标。

24.根据权利要求23所述的方法，其中，该一或多个额外舍入指标中的一个为一端回进位值，用以指明一端回进位修正悬而未决且正使用该端回进位值。

25.根据权利要求23所述的方法，其中，该一或多个额外舍入指标中的一个指明具有该C的相加运算是否在该第一相加运算中执行。

26.根据权利要求23所述的方法，其中，该一或多个额外舍入指标中的二个为不足位与

中间溢位指标。

27.一种微处理器中的方法，使用多个算术处理单元于多个运算元上以执行一复合算术运算，该方法包括：

通过执行该复合算术运算的至少一第一算术运算，使用一第一算术单元产生一未舍入结果；以及

使用该第一算术单元从该未舍入结果输出一储存空间格式中间结果；

其中，该储存空间格式中间结果包括该未舍入结果的多个最重要位、但排除该未舍入结果的多个最不重要位；

其中，该储存空间格式中间结果亦包括多个舍入指标，以使一第二算术单元自该储存空间格式中间结果产生一最终舍入结果；

使用该第二算术单元完成该复合算术运算，并从该储存空间格式中间结果产生该最终舍入结果。

微处理器及其方法

[0001] 本申请是申请日为2015年06月24日、申请号为201580003388.3(国际申请号PCT/US2015/037508)、发明名称为“使用第一和第二子运算的分路融合乘积-累加运算”的申请的分案申请。

[0002] 相关申请

[0003] 本申请主张申请日为2014年7月2日的美国专利第62/020,246号临时申请“Non-Atomic Split-Path Fused Multiply-Accumulate with Rounding cache”与申请日为2015年6月10日的美国专利第62/173,808号临时申请“Non-Atomic Temporally-Split Fused Multiply-Accumulate Apparatus and Operation Using a Calculation Control Indicator Cache and Providing a Split-Path Heuristic for Performing a Fused FMA Operation and Generating a Standard Format Intermediate Result”的优先权。该些优先权案的全文并入本申请以供参考。

[0004] 本申请还关联于下列与本申请同时申请的申请:标题为“Temporally Split Fused Multiply-Accumulate Operation”的美国第14/748,870号申请;标题为“Calculation Control Indicator Cache”的美国第14/748,924号申请;标题为“Calculation Control Indicator Cache”的美国第14/748,956号申请;标题为“Standard Format Intermediate Result”的美国第14/749,002号申请;标题为“Split-Path Heuristic for Performing a Fused FMA Operation”的美国第14/749,050号申请;标题为“Subdivision of a Fused Compound Arithmetic Operation”的美国第14/749,088号申请;与标题为“Non-Atomic Split-Path Fused Multiply-Accumulate”的美国第14/748,817号申请。这些申请的全文并入本申请以供参考。

技术领域

[0005] 本发明有关于一种执行算术运算的微处理器设计,尤其是融合浮点乘积-累加(FMA)运算的微处理器设计。

背景技术

[0006] 在现代计算机设计中,从大约1990年起,融合浮点乘积-累加(floating-point multiply-accumulate,FMA)运算就已经成为一个受到商业瞩目与学术关注的领域。融合FMA运算是一种算术运算,其形式为±A*B±C,其中,A、B与C是浮点输入运算元(分别是一个被乘数(multiplicand)、一个乘数(multiplier)、与一个累加数(accumulator)),并且在C累加至A与B的乘积前不存在舍入(rounding)运算。±A*B±C可包含,但不限于,下列例子:(a)A*B+C;(b)A*B-C;(c)-A*B+C;(d)-A*B-C;(e)A*B(即C设为零);与(f)A+C(即B设为1.0)。

[0007] 在大约1990年,此算术运算即以一原子(atomic)或不可分割(inseparable)运算的形式商业实现于IBM的精简指令集(RISC)系统/6000。而后续设计进一步最佳化浮点乘积累加运算。

[0008] 在其2004年的文献“Floating-Point Multiply-Add-Fused with Reduced

Latency”中,Tomas Lang与Javier D.Bruguera(“Lang et al.”)提出与最佳化FMA设计有关的许多重要课题。这些课题包括,指数差值与累加器移位/对准量的预计算,累加器与相乘阵列的平行对准,必要时使用2’补数累加器(2’s complement accumulator),和向量与进位向量的条件反转,在最终相加/舍入模组前对于和向量与进位向量的标准化处理,LZA/LOA与标准化移位的重迭运算,进位位、舍入位、保护位与粘(sticky)位的分别运算,以及在合一的相加/舍入模组中具有1m宽度的双总和加法器的使用(其中,m是其中一个运算元的尾数(mantissa)宽度)。

[0009] 在其2005年的文献“Floating-Point Fused Multiply-Add:Reduced Latency for Floating-Point Addition”中,Tomas Lang与Javier D.Bruguera(“Lang et al.”)提出利用分离数据路径(或双数据路径)把对准方向从标准化的情况移开,其中,“近(close)”的数据路径是用以从{2,1,0,-1}有效减去指数差值(此概念在本申请详细说明中会有更进一步的发展与改进),而“远(far)”的数据路径则是用来处理其他情况。Lang等人还提出在远的数据路径使用双对准平移器来处理相乘阵列的进位储存输出,以及在近的数据路径使用非常有限对准平移的方法。

[0010] 在2004年的文献“Multiple Path IEEE Floating-Point Fused Multiply-Add”中,Peter-Michael Seidel(“Seidel”)提出对于FMA设计的不同改进方法,即使用多个平行运算路径。Seidel并提出关闭未使用的路径门;从指数差值与有效运算确认多个运算路径;使用两个不同运算路径,其中一个提供给容易产生块消去(mass cancellation)的小指数差值使用,另一个则是用来处理其他情况;对于具有效减法的小指数差值的情况,在有效乘积运算中插入累加器数值。

[0011] 在现今随处可见个人行动运算装置来提供扩充媒体分发与网路内容存取的情况下,更需要去设计一个的低制作成本、少功耗,但又能以高效能执行指令的FMA逻辑电路。

[0012] 执行FMA运算的主要方法涉及合一的乘积-累加单元的使用,以执行整个FMA运算,包含对结果进行舍入。大部分学术提案与商业应用都描述一个单块的,或是不可分割的功能方块,其具有将两个数相乘,将未进行舍入的乘积与一第三运算元、加数或累加器进行相加操作、以及对运算结果进行舍入运算。

[0013] 另一种替代方案使用传统的乘法单元来执行A*B的子运算,然后使用传统的加法单元来将A与B的乘积与C相加。不过,此传统分工处理的方法会牺 牲运算速度以及在同一个单元内将C与A和B的部分乘积累加运算所能得到的效能增益。传统分工处理的方法也涉及两个舍入运算,因A与B的乘积会进行舍入运算,而后续C与乘积的加总也会进行舍入运算。因此,相较于合一处理的方法,传统分工处理的方法有时候会产生不同且较不准确的结果。同样地,因为重复舍入运算的存在,此传统分工处理的方法无法执行“融合(fused)”FMA运算,而未能符合IEEE 754技术标准关于浮点运算的要求。

[0014] 因为FMA硬件可同时用于多个运算目的并遵从IEEE 754技术标准,在现代的产品中,计算机设计者往往会选择利用不可分割的FMA执行单元来完全取代传统分离的乘积与加法功能方块。然而,此方法会带来许多不利影响。

[0015] 第一,相较于使用独立的加法与乘法功能单元,FMA硬件的设置成本较高,也较复杂。第二,在执行简单的加法或乘法运算时,相较于使用独立的加法与乘法功能单元,使用FMA硬件的延迟较长,且通常会消耗较多能量。第三,对于超纯量(superscalar)计算机设计

而言，将乘法与加法的功能合并于单一个功能方块会减少算术指令可发送端口的数量，而削减计算机针对源码、机械层级或软体开发平行处理的能力。

[0016] 前述第三个不利影响可加入更多功能单元来处理，例如一个独立的加法功能单元，但这会增加设置成本。基本上，一个额外的加法器(举例)就是为了在提供不可分割的FMA功能时，维持可接受的指令层级平行处理所需付出的代价，这又会增加整个硬件的设置尺寸，并会增加寄生电容与电阻。随着半导体制造技术朝向小尺寸发展，寄生电容与电阻会对于算术运算的时间延迟，或称延迟(latency)，有更显著的影响。此时间延迟有时会被模拟为因为“长导线”而造成的延迟。因此，为了减少实行不可分割的FMA运算对于指令层级平行处理影响所额外设置的功能方块，在考量所需的晶片尺寸、功耗与算术运算的延迟后，所能提供的改善其实相当有限。

[0017] 因此，最佳的提案与实施通常会(但不总是)提供正确的运算结果(对应于 IEEE的舍入与其他说明)，有时提供较高的指令产出(throughput)，但显然需要额外的硬件电路而增加设置成本，并且会需要增加功耗来在复杂的FMA硬件上执行简单的乘法与加法运算。

[0018] 现代FMA设计所要达成的目标仍未能完全达成。

发明内容

[0019] 本发明的一方面，提供一种微处理器，其包括一指令管线、一共享存储器、以及在指令管线中的第一与第二算术处理单元；其中，每一处理单元自共享存储器读取运算元并写入结果。第一算术处理单元执行一数学运算的一第一部分、以产生一中间结果向量，其不为数学运算的一完全与最终的结果。第一算术处理单元亦产生多个非架构运算控制指标，以指明自中间结果向量产生一最终结果的接续运算该如何进行。第二算术处理单元执行数学运算的一第二部分，并依据运算控制指标产生数学运算的一完全与最终的结果。

[0020] 依据本发明的一实施例，数学运算的第一部分包括至少二输入运算元的一乘积。在更进一步的实施例中，如果前二输入运算元与第三运算元的数值运算元满足一或多个预定状况中的至少一个时，其数学运算的第一部分还包括一具有一第三运算元的相加运算；若非，则仅有数学运算的第二部分包括一具有一第三运算元的相加运算，无论如何，至少数学运算的第二部分包括一舍入子运算。

[0021] 在一更特别的实施例中，数学运算为一乘法乘积-相加运算，且其微处理器还包括一翻译器或一ROM，其将一不可切割的、一体的乘积-相加指令转换为至少第一与第二微指令。更甚者，第一微指令的执行能够产生中间结果向量，而第二微指令的执行则使用此中间结果向量产生前述的完全与最终的结果。

[0022] 在一更加特别的实施例中，数学运算为一±A*B±C形式的融合浮点乘积- 相加(FMA)运算，其中A、B与C为浮点输入运算元，且在C乘积-相加至A与B的乘积前，不会发生舍入运算。

[0023] 依据本发明的一实施例，中间结果向量为一未舍入值，且完全与最终的结果为一舍入值。尚且，运算控制指标包括舍入指标，用以提供充足信息使第二算数处理能于在中间结果向量上执行数学运算的第二部分后、产生一正确的舍入完全与最终的结果。

[0024] 依据本发明的一实施例，第一算术处理单元将中间结果向量储存于一暂存器中，且将运算控制指标储存于一运算控制指标快取中，而第二算术处理单元则自暂存器下载中

间结果向量、且自运算控制指标快取下载运算控制指标。在本发明的另一实施例中，微处理器将中间结果向量传送至第二算术处理单元。

[0025] 在本发明的另一方面，所提供的微处理器中的方法用以执行一±A*B±C形式的融合乘积-相加运算，其中的A、B与C为输入运算元，该方法包括：计算该运算元A与该运算元B的部分乘积；自下列之一中的第一相加运算产生产生一原始结果：(a)该运算元A与该运算元B的该部分乘积、或(b)具有该运算元A与该运算元B的该部分乘积的该运算元C；选择该原始结果的多个最重要位，以加入一未舍入中间结果向量；将多个最不重要位缩减至一或多个舍入指标；如果该第一相加运算并未包括该运算元C，则执行具有该未舍入中间结果向量的该运算元C的一第二相加运算；以及使用该舍入指标产生该乘积-相加运算的一最终舍入结果。

[0026] 一或多个最不显著位则自未舍入结果到产生一未舍入中间结果向量时排除。依据本发明的一实施例，此未舍入中间结果向量以一浮点数字用的标准IEEE格式来表示，其为一尾数，其具有一相等于融合乘积-相加运算一目标结果位数的位数。

[0027] 在一更特别的实施例中，未舍入中间结果向量包括一中间尾数结果及一 中间结果指数(IRExp)，其中IRExp为C的一指数以及运算元A与B指数值总和的一函数的较大者的正常化表示。未舍入中间结果向量亦包括一中间符号指标，其产生由一是否第一相加运算包括运算元C、乘积-相加运算是否为一有效减法、及是否有悬而未决的端回进位的函数表示，亦即，此处的中间符号指标以一包括是否第一相加运算包括运算元C、乘积-相加运算是否为一有效减法、及是否有悬而未决的端回进位的函数产生。

[0028] 依据本发明的一实施例，此方法还包括产生一端回进位指标(E)。如果包括运算元C与未舍入中间结果向量的第一相加运算为正值，则此相加运算乃为一有效减法，此时则产生一E值、以指明尚有一端回进位修正悬而未决。在本发明的另一实施例中，此方法还包括产生中间不足位(U)与中间溢位(O)指标、以指明IRExp是否在一可代表或所欲指数范围之上或之下。

[0029] 未舍入结果所排除的最不显著位降低至一或多个舍入指标。依据本发明的一实施例，此一或多个舍入指标包括保护(G)、舍入(R)与/或黏(S)位。在本发明的另一实施例中，舍入指标(Z)的其中之一指明具有C的相加运算是否有于第一相加运算中执行。在本发明的又一实施例中，不足位(U)与溢位(O)指标为二舍入指标。在本发明的又一实施例中，舍入指标的其中之一(E)指明是否有一端回进位悬而未决。

[0030] 依据本发明的一实施例，此方法还包括将一或多个舍入指标存入一舍入快取中。在本发明的另一实施例中，此方法还包括将未舍入中间结果向量存入可供多个指令执行单元存取的共享储存空间中。

[0031] 如果第一相加运算并未包括运算元C，则一第二相加运算乃以具未舍入中间结果向量的运算元C执行，乘积-相加运算的一最终舍入结果即利用舍入指标产生。

[0032] 本发明的另一方面，所提供微处理器中的方法使用多个算术处理单元于多个运算元上以执行一复合算术运算。该方法包括：通过执行复合算术运算 的至少一第一算术运算，使用一第一算术单元产生一未舍入结果；接着使用此第一算术单元从未舍入结果产生一储存空间格式中间结果，其中，此储存空间格式中间结果包括未舍入结果的多个最重要位(MSB)，但排除未舍入结果的多个最不显著位(LSB)；而储存空间格式中间结果亦包括多

个舍入指标，以使一第二算术单元自储存空间格式中间结果产生一最终舍入结果，使用第二算术单元完成复合算术运算，并从储存空间格式中间结果产生最终舍入结果。

[0033] 此方法与装置可缩减所需的电路、设置成本与复合算术运算的累增功耗。简单来说，此装置与方法是将复合算数运算分成至少二个子运算，由物理上与/或逻辑上各自独立的硬件单元来执行。每个硬件单元执行此复合算术运算的一部分。舍入运算或运算控制所需的位于此二运算间，储存于一快取内。这些子运算于不同时间与位置完成，而必要的数据片段会被整合以完成最终舍入运算。

[0034] 此方法与装置具有许多值得注意的优点，尤其是针对FMA运算。

[0035] 第一，此方法与装置可辨识FMA运算并将其区分为至少二种类型，其中部分运算以暂时或物理性分割的方式执行。

[0036] 第二，此方法与装置可将来自指令集架构(ISA)的一不可分割的(atomic)或一体的(unified)FMA指令转译或转换为至少二个子运算。

[0037] 第三，此方法与装置容许这些子运算以非不可分割的、或暂时物理性分割的方式来执行，例如在非循序超纯量计算机处理器装置内执行。

[0038] 第四，FMA运算(例如对应于部分第一类型的FMA运算或部分第二类型的FMA运算)中部分的必须算术运算于执行第一特定微指令时执行。

[0039] 第五，此方法与装置以一新的方式预先计算FMA的符号数据。

[0040] 第六，此方法与装置会储存中间运算的部分结果，例如储存于一结果(重命名)暂存器。

[0041] 第七，此方法与装置会储存运算结果的其他部分，例如储存至称为舍入快取或运算控制指标快取的其他储存单元。

[0042] 第八，此方法与装置会将所收集到的数据，即中间结果，以新的标准化储存格式来储存。此外，此方法与装置可能会将储存格式中间结果转送至后续特殊类型的第二微指令，而非进行储存。

[0043] 第九，此方法与装置在需要提供所储存的数据至后续第二微指令时，会存取舍入快取。

[0044] 第十，响应来自舍入快取的数据，此方法与装置会选择性地提供FMA加数至第二微指令或使输入值归零。

[0045] 第十一，此方法与装置会在执行一第二(或更多)微指令的过程中，使用此储存格式中间结果作为输入，来执行第一类型或第二类型的FMA运算中剩下必须执行的算术FMA运算。

[0046] 第十二，此方法与装置将经少量调整的现有技术乘法硬件执行单元与加法硬件执行单元结合，如结合所述的舍入快取与数据转送网路用以回避舍入快取。

[0047] 第十三，此方法与装置不会让分配端口无法于算术运算中使用、或是在计算机利用指令平行处理的能力与投资的硬件成本间妥协。

[0048] 本发明所采用的具体实施例，将通过以下的实施例及图式作进一步的说明。

附图说明

[0049] 图1是一微处理器的方块示意图，此微处理器具有执行单元与一舍入或运算控制

指标快取，并使用二个子运算、一个调整后的乘法器与一个调整后的加法器来执行FMA运算。

[0050] 图2是一示意图，将一数字空间区分为FMA运算的五个类型以为例示(但 非限定于此)。

[0051] 图3是一功能方块图，描述用以执行FMA运算的调整后的乘法器与调整后的加法器内的逻辑元件。

[0052] 图4是一功能方块图，显示本发明一实施例的乘法运算单元的路径确认逻辑与尾数乘法模组，此乘法运算单元经适当调整以接收FMA乘数、被乘数与累加数作为输入运算元。

[0053] 图5是一功能方块图，显示图4的乘法运算单元的指数结果产生器与舍入指标产生器，此乘法运算单元经适当调整以产生一储存格式中间结果。

[0054] 图6是一功能方块图，显示经适当调整以接收一储存格式中间结果与累加数的加法运算单元的一实施例。

[0055] 图7是一功能方块图，显示非不可分割的分路FMA运算的一实施例的第一FMA子运算中的一路径确认部分。

[0056] 图8是一功能方块图，显示非不可分割的分路FMA运算的第一FMA子运算中的一乘法与累加部分。

[0057] 图9A及图9B是以一功能方块图，显示非不可分割的分路FMA运算的第一FMA子运算中的一储存格式中间结果产生部分。

[0058] 图10是一功能方块图，显示非不可分割的分路FMA运算的第二FMA子运算。

[0059] 图11是一示意图，显示将融合FMA指令转译为第一与第二FMA微指令的一实施例。

具体实施方式

[0060] 微处理器

[0061] 图1是一微处理器的方块示意图。此微处理器10具有多个执行单元45,50,60用以执行FMA运算。此微处理器10包含一指令快取15，一指令转译 器与/或微码只读存储器20，一重命名单元与保留站25，多个执行单元(包含一调整后的乘法器45、一调整后的加法器50与其他执行单元60)，一舍入快取55(或是指运算控制指示器储存空间)，架构暂存器35与一重排缓冲器30(包含重命名暂存器)。其他的功能单元(未图示)可包含一微码单元；分支预测器；一存储器子系统，其包含一快取存储器阶层架构(例如阶层一的数据快取、阶层二的快取)、存储器排列缓冲器、与存储器管理单元；数据预撷取单元；与一总线接口单元等等。微处理器10具有一非循序执行微架构，可以不依照程序顺序发布指令以供执行。进一步来说，架构指令(或微指令)转译或转换出的微指令，可以不依照程序顺序发布指令以供执行。微指令的程序顺序相同于转译或转换出这些微指令的相对应架构指令的程序顺序。微处理器10并具有一超纯量微架构，其能在单一个时钟周期内发布多个指令至执行单元执行。在一种实施例中，微处理器10可以说是以相容于x86指令集架构的方式提供指令以供执行。

[0062] 指令快取15储存由系统存储器撷取的架构指令。指令转译器与/或微码只读存储器20将由系统存储器撷取的架构指令转译或转换为微处理器10的微架构微指令集的微指令。执行单元45,50,60执行这些微指令。这些由架构指令转译或转换出的微指令可实现架

构指令。重命名单元25接收微指令并将重排缓冲器的项目依据程序顺序分配给微指令、依据所分配的重排缓冲器项目索引更新这些微指令、将各个微指令发送至与将要执行这些微指令的执行单元相关联的保留站25、以及为这些微指令执行暂存器重命名与关联建立。

[0063] 基于类型的分类运算

[0064] 在本发明的一实施例中，FMA运算是依据输入运算元的指数值的差(由变数ExpDelta表示)，以及是否涉及一有效的减法运算来进行分类。图2利用一包含有数字线70的数字空间65来显示ExpDelta的值。在数字线70下方的空间显示此运算会构成一有效减法运算。在数字线70上方的空间显示此运算会 构成一有效加法运算(亦即不存在有效减法运算)。

[0065] 指数差，表示为ExpDelta，是乘数与被乘数的输入指数值的加总，减去任何指数偏移值，再减去加数或减数的输入指数值。在累加值远大于偏移调整乘积向量(bias-adjusted product vector)时，计算出的ExpDelta为负。而在累加值远小于偏移调整乘积向量时，计算出的ExpDelat为正。

[0066] “有效减法”，由变数EffSub表示，是指输入运算元的符号与所欲执行的运算(例如乘法－加法，或是乘法－减法)的结合，将会有效降低浮点数结果的大小，而非增加。举例来说，负的被乘数乘上正的乘数(乘积为负)后，加上一个正的加数，就会有效降低结果的大小，而会表示为有效减法(EffSub)。

[0067] 如图2的数字空间65的右侧所示，在乘积向量的大小主导运算结果的情况下，累加数会直接用于初始舍入位(round bits)或是粘位(sticky bits)的运算。如本文后续将描述的，累加数与乘积尾数的相对对准有利于在计算影响舍入运算的位前将此二者相加。在图2的数字空间65内，这些不存在“有效减法”的情况显示为“类型2”运算80，而存在“有效减法”的情况显示为“类型4”运算90。

[0068] 如图2的数字空间65的左侧所示，在累加数的大小主导运算结果，并且累加数的尾数大小小于或等于所欲产生结果的尾数大小，累加数就不会用于初始舍入位或是粘位的运算。在图2的数字空间65内，这些不存在“有效减法”的情况显示为“类型3”运算85，而存在“有效减法”的情况显示为“类型5”运算95。因为累加数有效对准乘积尾数的左侧，所以可获得在加上累加数之前，先确认粘位与舍入位的好处。

[0069] 区分出ExpDelta位于图2的数字线70的右侧的情况与ExpDelta位于图2的数字线70的左侧的情况会有许多优点。举例来说，传统FMA利用非常宽的对准移位器－相当于或大于输入尾数宽度的三倍－来解释累加数对准于乘数 与被乘数的乘积的左侧或右侧的运算。通过将FMA运算区分为由两个调整后的执行单元(一个调整后的乘法器45与一个调整后的加法器50)分别执行的两个子运算，即可利用较小的数据路径与较小的对准移位器来进行运算。

[0070] 对于数字线70的右侧的运算，累加数的大小会小于中间乘积向量，此情况有利于在调整后的乘法器45内将累加数与乘法器的乘积相加。此运算只需要一个比传统FMA的数据路径的宽度还要小上大约一个尾数宽度的数据路径。因为调整后的乘法器45原本就具有一些内部延迟，累加数会有效地对准加总树/阵列(summation tree/array)，也会简化标准化与舍入运算。舍入运算将会由调整后的加法器50会在第二FMA子运算中执行。

[0071] 反之，对于数字线70的左侧的运算，累加数会是较大的运算元而不会用于舍入运

算。因为累加数不会用于舍入运算(除了以下提到的特殊状况),因而可以对乘法器的乘积执行一些初始粘收集(initial sticky collection)、将中间结果储存至存储器(例如重排缓冲器与/或快取)、并且可以用调整后的加法器50来进行累加数的加法运算。传统的舍入逻辑可有效处理累加数对舍入运算的选择造成影响的特殊状况,若是存在总数溢位,舍入位会成为粘位的其中之一,而总数的最重要位(least significant bit,LSB)会成为舍入位。

[0072] 某些种类的FMA运算－“有效减法”运算中对应于在图2的数字空间65下半部的子集合－会使一个或多个最重要位归零,传统上将此称为“块消去(mass cancellation)”。在图2中,具有块消去潜力的运算表示为“类型1”运算75。此情况需要在舍入运算前执行标准化运算来确认舍入点的位置。标准化向量所涉及的移位运算会产生显著的时间延迟与/或呼叫前导位预测(leading digit prediction)以供使用。另一方面,不涉及块消去的FMA运算可以省略前导位预测。

[0073] 总之,如图2所示,FMA运算可依据ExpDelta与EffSub进行分类。第一类的FMA运算75定义为包含ExpDelta落于{-2,-1,0,+1}的范围且EffSub为真的 运算,这些运算包含那些具有位块消去潜力的运算。第二类的FMA运算80是定义为包含ExpDelta大于或等于-1且EffSub为假的运算。第三类的FMA运算85是定义为包含ExpDelta小于或等于-2且EffSub为假的运算。第四类的FMA运算90是定义为包含ExpDelta大于{+1}且EffSub为真的运算。第五类的FMA运算95是定义为包含ExpDelta小于{-2}且EffSub为真的运算。值得注意的是,这些分类的定义仅为例示,FMA运算当可采不同方式进行分类。举例来说,在另一实施例中,第二类与第四类可以用同一类表示,相同地,第三类与第五类也可以同一类表示。此外,在其他实施例中,图2的数字线70的左部分与右部分的分隔线亦可改变。

[0074] 融合FMA指令执行元件组

[0075] 图3是显示一般用以执行FMA运算的融合FMA指令执行元件组100的一实施例。此元件组100包含两个物理上与/或逻辑上分开的算术逻辑单元,(在一实施例中,即为一调整后的乘法器45与一调整后的加法器50)与共享储存空间155与55以储存多个未经舍入运算的中间结果向量与舍入指标(rounding indicator)。

[0076] 各个调整后的乘法器45与调整后的加法器50都是一个指令执行单元,进一步来说,是指令管线24内的一个算术处理单元,用以对机器指令(例如复杂指令集(CISC)微架构内一个指定的指令集或是精简指令集(RISC)微架构内一个指定的指令集)进行解码,以及从一组共享的高速存储器读取运算元并写入结果。指令执行单元可被理解为一个逻辑电路的特性集合,用以执行传送过来的指定机器指令集,而不同于可平行执行(而不仅止于管线化执行)多个机器指令的较大的电路群(如果存在的话)。

[0077] 更特别的是,调整后的乘法器45与调整后的加法器50互相分离、不可分割的、能独立运行的执行单元,能独立对微码进行解码并执行,并提供控制信号至内部的数据路径。共享高速存储器可以是一个暂存器档案或是一组非 架构运算暂存器,供微指令交换数据并使其运算结果可为其他执行单元看见。

[0078] 更特别的是,调整后的乘法器45是一个适当的乘法运算单元,在大部分情况下,就像传统的乘法运算单元,能执行不属于FMA运算的一般乘法微指令。不过,此调整后的乘法器经适当调整,使能接收FMA乘数105、被乘数110、与累加数115作为其输入运算元,并产生

一储存格式中间结果150，这在后续段落会有更进一步的描述。相类似地，调整后的加法器50是一个适当的加法运算单元，在大部分情况下，就像传统的加法运算单元，能执行不属于FMA运算的一般累加微指令，例如加或减。不过，此调整后的加法器经适当调整，使能接收储存格式中间结果150并产生一个正确且经舍入运算的FMA结果。

[0079] 调整后的乘法器45能够执行第一阶层或部分的融合FMA运算(FMA1子运算)。此调整后的乘法器45包含一输入运算元分析器140、一乘法加总阵列120、一最终加法器125、一标准化移位器、与一前导位预测与编码器135。在执行FMA1子运算时，调整后的乘法器45会产生并输出一未经舍入运算的标准化加总结果145与多个舍入位(或是舍入指标)。相较之下，在执行非融合FMA运算时，调整后的乘法器45会产生一经舍入运算且相容于IEEE标准的结果。

[0080] 舍入位与未经舍入运算的标准化加总结果145的最重要位依据一储存格式进行储存。在一实施例中，未经舍入运算的标准化加总结果145的最重要位输出至一结果总线146，以储存至一尾数宽度等于目标数据格式的尾数宽度的重命名暂存器155。舍入位输出至一专用舍入位或运算控制指标数据路径、或是位于调整后的乘法器外且不同于结果总线146的连接网路148，以储存至一不同于储存重命名暂存器155的储存单元(例如重排缓冲器30)的舍入快取55。这些未经舍入运算的标准化加总结果145的最重要位，连同舍入 位，包含一储存格式中间结果150。

[0081] 因为重命名暂存器155与舍入快取55属于可为其他执行单元看见的共享存储器的一部分，物理上与/或逻辑上独立于调整后的乘法器45的调整后的加法器50，可以通过运算元总线152与舍入位数据路径148接收此储存格式中间结果150，并执行一第二(完成)阶层或部分的融合FMA运算(FMA2子运算)。此外，在FMA1子运算与FMA2子运算间亦可执行其他不相关的运算。

[0082] 调整后的加法器50提供一运算元乘数160，使FMA状态下的累加运算元归零，而在FMA状态下，调整后的乘法器45已完成必要的累加运算。调整后的乘法器50并具有舍入位选择逻辑175，从调整后的乘法器45产生的舍入位、调整后的加法器50内部产生的舍入位，或是二者的组合，选择用于舍入模组180以产生最终舍入结果的舍入位。调整后的加法器并具有一近路径加总电路165，用以在存在两个加总运算元的块消去的情况下对总数进行标准化运算，并具有一远路径加总电路170，用以执行最多只需单一位移位的加总运算以产生总数。如下所述，FMA2子运算可完全由远路径加总电路170进行处理。

[0083] 调整后的乘法器

[0084] 图4和5是详细显示调整后的乘法器45的一实施例。图4特别显示调整后的乘法器45的路径确认逻辑185与尾数乘法模组190。图5特别显示调整后的乘法器45的指数结果产生器260与舍入指标产生器245。

[0085] 如图4所示，路径确认逻辑185包含一输入解码器200、一输入运算元分析器140、路径控制逻辑215与一累加数对准与射入逻辑电路220。尾数乘法模组190包含图3的乘法加总阵列120，这在图4中以二个元件显示，即一乘法阵列235与一部分乘积加法器240。尾数乘法模组190并包含一最终加法器125、一前导位预测器与编码器135、与标准化移位器130。

[0086] 如图5所示，指数结果产生器260包含一PNExp产生器265、一IRExp产生 器270、与一不足位/溢位侦测器275。舍入指标产生器245包含一中间符号产生器280、一结果向量端

口285、一端回进位指标290、一粘位产生器295与一舍入位产生器300。

[0087] 请参照图4，调整后的乘法器45通过一个或多个输入端口195接收一输入微指令以及运算元数值。就FMA微指令而言，调整后的乘法器45接收一被乘数运算元A、一乘数运算元B与一累加数运算元C，各个运算元都包含一符号指标或位、一尾数与一指数。在图4与图6中，浮点运算元的符号、尾数与指数部分分别以下标S、M与E表示。举例来说，A_S、A_M、A_E分别代表被乘数的符号位、被乘数的尾数与被乘数的指数。

[0088] 解码器200对输入微指令进行解码以产生FMA指标M与二进位数运算符号指标(或位)P_S与O_S，M意指接获FMA微指令。在一实施例中，形式为A*B+C的FMA微指令会产生二进位数零的一正乘法/向量负乘法符号运算子P_S与一加/减运算子O_S。形式为-A*B+C的负乘积-累加微指令会产生一个二进位数一的运算子P_S与一个二进位数零的运算子O_S。形式为A*B-C的乘积-相减微指令会产生一个二进位数零的运算子P_S与一个二进位数一的运算子O_S。形式为-A*B-C的向量负乘积-相减微指令会产生二进位数一的运算子P_S与运算子O_S。在其他较为简化的实施例中，调整后的乘法器45并不直接支援向量负微指令与/或减法微指令，但由微处理器10来支援等效的运算，也就是在调用乘积-累加/相减微指令至调整后的乘法器45前，视情况额外反转一个或多个运算元或符号指标。

[0089] 乘法阵列235接收被乘数与乘数的尾数值A_M与B_M并计算出A_M与B_M的部分乘积。可以理解的是，当A_M或者B_M的绝对值为一或零，乘法阵列235所产生的单一个“部分乘积”的值就会是A_M与B_M的完整乘积。部分乘积提供至部分乘积加法器240，其提供多个项目以接收A与B的部分乘积以等待将其加总，而至少一个的部分乘积加法器240的项目用以接收一累加来源值C_X。如 下所述，在讨论完输入运算元分析器140与累加数对准射入逻辑220后，会对部分乘积加法器240有其他的说明。

[0090] 输入运算元分析器140包含一ExpDelta分析子电路210与一EffSub分析子电路205。ExpDelta分析子电路210产生ExpDelta(Exp Δ)值。在一实施例中，ExpDelta通过将乘数与被乘数的输入指数值A_E与B_E加总，减去加数或减数输入指数值C_E，并在存在指数偏移值ExpBias时减去此指数偏移值(如果任一者存在)。在A_E、B_E与C_E是以偏移指数(biased exponent)表示时，例如IEEE754的规范，被乘数A与乘数B的乘积的偏移量将会是累加数C偏移量的两倍。而导入ExpBias值可对此进行校正。

[0091] EffSub分析子电路205分析运算元符号指标A_S、B_S与C_S以及运算元符号指标P_S与O_S。EffSub分析子电路205产生一“EffSub”值，用以表示FMA运算是否为一有效减法运算。举例来说，若是对A与B的乘积与C执行运算元特定的加法或减法运算(或是在负向量乘法运算子的情况下，此运算结果的负值)所产生的结果R的绝对值小于(a)A与B的乘积的绝对值或是(b)C的绝对值，就会是有效减法。若以数学符号表示，若是$(|R|<|A*B|)\vee(|R|<|C|)$，其中R为FMA运算的结果，此FMA运算就会构成有效减法。虽然以FMA运算的结果可以简化EffSub的描述，不过需理解的是，在EffSub分析子电路205预先确认EffSub时，实际上是通过分析符号指标A_S、B_S、C_S、P_S与O_S来进行，而不去评估A、B与C的尾数、指数与大小。

[0092] 路径控制逻辑215接收由输入运算元分析器140产生的ExpDelta与EffSub指标，并借以产生一路径控制信号，其数值以变数Z表示。路径控制逻辑215可控制C的累加运算是否会连同A与B的部分乘积一并在调整后的乘法器45内执行。在一实施例中，路径控制逻辑215用以产生Z的设定条件显示于图2中。在一实施例中，对于任何调整后的乘法器45被选用来

执行乘积-累加运算的累加部分运算的情况合(例如类型1,2与4),Z会是二进位数一,而对于任何ExpDela与EffSub的其他组合(例如类型3与5),Z会是二进位数零。

[0093] 另外,路径控制逻辑215也可以通过判断C所具有一大小,相较于A与B的乘积大小,是否可使C对准于加总树(summation tree)内,而不需将C的最重要位移位至A与B的部分乘积加总的加总树所提供的最重要位的左侧来产生Z。另一个或替代的设定条件可以是,在执行FMA运算时是否具有执行块消去的潜力。再另一个或替代的设定条件可以是,对A与B的乘积进行C的累加运算所产生的舍入前结果R所需要的位数,是否少于将C对准A与B的乘积所需要的位数。由此可知,路径控制的条件可响应调整后的乘法器45的设计进行改变。

[0094] 累加数对准射入逻辑220接收路径控制逻辑215所产生的Z、ExpDelta分析子电路210所产生的ExpDelta、一移位常数SC、与累加数尾数值C_M。在一实施例中,此累加数对准射入逻辑220亦接收C_M的位反相(bitwise negation),即$\overline{C_M}$,与加法/减法累加运算子指标O_S。在另一实施例中,累加数对准射入逻辑220在加法/减法累加数指标O_S显示调整后的乘法器45所接收的微指令为乘积-相减微指令时,选择性地额外反转C_M的值。

[0095] 相应于所接收的输入,累加数对准射入逻辑220会产生一数值C_X射入部分乘积加法器240。此射入阵列的C_X的宽度是2m+1,或可理解为输入运算元的尾数A_M、B_M与C_M的两倍宽度额外加上一个位。

[0096] 若是M为二进位数零以显示调整后的乘法器45正在执行一般的乘法运算而非FMA1子运算,多工器230就会将一舍入常数RC,而非C_X,射入部分乘积加法器240,借此,调整后的乘法器45即可以传统方式产生一舍入后结果。RC的数值部分是由指令显示的舍入运算的类型所决定(例如:舍入向上(round half up)、舍入相等(round half to even)、舍入远离零(round half away from zero)),部分是由输入运算元的位尺寸(例如32位与64位)所决定。在一实施例中,部分乘积加法器240利用两个不同的舍入运算常数来计算出两 个总数,并选择其中较适当的一个。借此,调整后的乘法器45的IMant输出就会是一般乘法运算的正确舍入后的尾数结果。

[0097] 若是M为二进位数一而Z为二进位数零,即表示不需对A与B的乘积执行C的累加运算,在此情况下,就一实施例而言,此累加数对准射入逻辑220会将C_X设定为零,而使多工器230将零值射入用以接收C_X值的部分乘积加法器240阵列。若是M为二进位数一且Z为二进位数一,累加数对准射入逻辑220就会将C_X右移相等于ExpDelta加上一移位常数SC的量,以产生C_X。在一实施例中,移位常数SC等于2,以对应于图2,当C的累加运算执行于调整后的乘法器内,图中数字空间内最大的ExpDelta负值。随后,多工器230会将运算结果C_X射入部分乘积加法器240。

[0098] 累加数对准射入逻辑220内并结合有一粘收集器(sticky collector)。累加数C_X中任何移位超过部分乘积加法器240加总树的最不重要位的部分,保留于XtraStky位供舍入运算之用。因为会有多达m个位移位超过部分乘积加法器240加总树的最不重要位,XtraStky位作为一宽度m的额外粘位阵列,用于粘位S的运算。

[0099] 回到调整后的乘法器45的加法逻辑,在部分实施方式中,部分乘积加法器240为一加总树,而在一实施方式中,为一个或多个进位储存加法器。此部分乘积加法器240根据所提供的部分乘积加总树的位栏的进位储存向量,对一未经舍入的冗余代表或总数,依据现有技术的乘法执行单元通常会执行的运算方法,来进行累加运算,这包含在部分乘积的累

加运算中，对累加数输入值选择性进行额外的位反相与对准运算。

[0100] 同样地，部分乘积加法器240所执行的算术运算会受到Z的数值的影响。若是Z＝1，部分乘积加法器240就会对于A_M与B_M的乘积执行C_X的连带累加运算(joint accumulation)。若是Z＝0，部分乘积加法器240就会对A_M与B_M的乘积执行一基本累加运算。部分乘积加法器240会产生一由一2m位总数向量与一 2m位进位向量表示的冗余二进位数总数，作为连带累加运算或是基本累加运算的运算结果。

[0101] 这些进位与总数向量同时转送至一最终加法器125与一前导位预测器与编码器135。此最终加法器125可为一进位预看加法器(carry-lookahead adder)或一进位传播加法器(carry propagate adder)，通过将进位与总数向量转换成宽度为2m+1的正或负的预标准化未舍入非冗余总数(prenormalized unrounded nonredundant sum)PNMant以完成累加运算程序。最终加法器125并产生一总数符号位SumSign，来显示PNMant为正数或负数。

[0102] 在最终加法器125产生PNMant的同个时间周期，前导位预测器与编码器135会同步预测需要被消除以标准化PNMant的前导位的数量。相较于传统对乘积-累加运算分工处理的FMA设计中，因其最终加法器125执行的最终加法运算是于标准化运算后执行而需同时对进位向量与总数向量执行标准化运算，因而必须等待前导位预测的输出，故本发明的处理方式相较传统作法实具有优点。在一实施例中，此前导位预测器与编码器135不是可以适用于正总数的情况，就是可适用于负总数的情况。

[0103] 在一实施例中，前导位预测只在类型1的运算执行。如前所述，所选择的前导位预测的方法不是可以适用于正总数就是可以适用于负总数，为熟习浮点运算设计领域者所熟知。

[0104] 因为前导位预测器与编码器135具有最多一个位的误差，任何可用于校正此误差的常用技术均可使用，这些技术可设置于标准化移位器130内，或是关联于标准化移位器130。一个解决方案提供逻辑来预测此误差。另一个解决方案通过确认PNMant的MSB是否已经设定，并相应地选择PNMant的一额外移位。

[0105] 标准化移位器130从最终加法器125接收此未舍入非冗余总数PNMant并产生一原始尾数值GMant。在C_X的累加运算是由部分乘积加法器240执行的 情况下，GMant会是A_M和B_M乘积与C_X的标准化加总的绝对值。而在其他情况下，GMant会是A_M和B_M乘积的标准化加总的绝对值。

[0106] 为了产生GMant，标准化移位器130在SumSgn显示PNMant为负时对PNMant执行位反相的运算。对负的PNMant数值执行标准化移位器130的位反相运算，可产生如下所述的储存格式中间结果150，并有助于正确的舍入运算。在调整后的乘法器内反转PNWant，即可产生一正数提供给调整后的加法器，而不需先知会PNWant的数值为负。此处理方式可使累加运算实施起来就像加总运算并以简化的方式进行舍入运算。

[0107] 此外，此标准化移位器130会将PNMant左移一个由LDP、EffSub与Z的函数所计算出来的量。值得注意的是，即使没有发生最重要前导位的消除，仍需要将PNMant左移零、一或二个位位置以产生一有用的标准化储存格式中间结果150，以确保后续的舍入运算可正确进行。此标准化运算由一个左移所构成，以将算术上最重要位移动至标准化的最左位置，使能表示于如下所述的储存格式中间结果150。

[0108] 相较于传统的FMA设计，此实施方式具有三个额外的优点。第一，此实施方式不需

在部分乘积加法器240内插入额外的进位位(若响应于EffSub对累加数尾数执行2'补数时有此需求的话)。第二,此实施方式不需提供一大的符号位侦测器/预测器模组,来检测并选择性补足非冗余部分乘积与累加数加总值的冗余总数与进位向量表示。第三,此实施方式不需输入额外的进位位来确保部分乘积与累加数加总运算中,选择性补足的总数与进位向量表示的运算正确。

[0109] 关于图5的指数结果产生器260,PNExp产生器265产生一预标准化指数值PNExp,此数值为被乘数与乘数指数值A_E与B_E、指数偏移值ExpBias与移位常数SC的函数。在一实施例中,PNExp以算式SC+A_E+B_E-ExpBias来计算。

[0110] IRExp产生器270使PNExp递减,作为标准化移位器130执行的尾数标准 化运算,以产生一中间结果指数IRExp。此数值为PNExp与前导位预测(leading digit prediction, LDP)的函数。随后,IRExp转送至如下所述的结果向量端口280。

[0111] 中间符号产生器280产生中间结果符号指标IRSgn,其为EffSub、E、A_S、B_S与Z的函数。在一实施例中,IRSgn在某些情况下以被乘数符号位A_S与乘数符号位B_S的逻辑互斥或来计算。不过,若是Z位为二进位数一显示累加运算已经执行,EffSub也是二进位数一显示有效减法,而E位值为二进位数零显示没有等待中的端回进位(end-around carry),IRSgn就会以被乘数符号位A_S与乘数符号位B_S的逻辑互斥或的反相值(XNOR)来计算。换句话说,中间符号通常是A与B的乘积的符号。当累加数的大小大于A与B的乘积时,A与B的乘积的符号会反转,乘积-累加运算会是一有效减法运算,而累加运算的完成不需要使用端回进位(因为累加运算为负)。

[0112] 中间结果符号指标IRSgn用于一可用以确认最终符号位以供具块消去可能的FMA运算使用的创新方法。不同于传统的分路FMA实施方式,此实施方式不需要符号预测器,也不需要用以预测符号的大量电路。另外,零结果的符号,或是具有符号零输入的运算结果的符号,容易预先计算,来纳入例如一舍入模式输入。

[0113] 结果向量端口280输出一储存格式中间结果向量IRVector,其包含中间结果指数IRExp、中间结果符号IRSgn与中间结果尾数IRMant。在此储存格式的一实施例中,IRMant包含GMant的最重要m位,其中m为目标数据类型的宽度。举例来说,在IEEE双精确度(double precision)运算中,结果向量端口280输出IRVector作为单一个符号位、十一个指数位、与GMant最重要53位的组合。在储存格式的另一个实施例中,m等于A_M、B_M与C_M的尾数的宽度。在储存格式的又一个实施例中,m大于A_M、B_M与C_M的尾数的宽度。

[0114] 类似于IEEE的标准储存格式,这些尾数位中的单一个最重要位作为储存 时的一隐含值。IRVector储存至一共享存储器,例如重排缓冲器30的重命名暂存器155,借此,其他指令执行单元就可存取IRVector,且/或IRvector可通过一结果转送总线(forwarding bus)40转送至其他指令执行单元。在一实施例中,IRVector储存至一重命名暂存器155。此外,不同于架构暂存器会对重排缓冲器给予一固定不变的任务分派,中间结果是向量在重排缓冲器内给予一不可预期的任务分派。在另一实施例中,IRVector暂时储存于将用以储存FMA运算的最终舍入结果的目标暂存器内。

[0115] 现在请参照图5的舍入指标产生器245,不足位/溢位侦测器275产生不足位指标U_I与溢位指标O_I,其为IRExp与指数范围值ExpMin与ExpMax的函数。指数范围值是与储存格式中间结果150(下面将有进一步讨论)的精确度或目标数据类型有关。若是IRExp小于可代表

此FMA运算目标数据类型的指数值的范围，或是小于可代表的任何中间储存空间，例如重命名暂存器，的指数值的范围，U_1位就会是二进位数一，否则U_1位就会是二进位数零。相反地，若是IRExp大于可代表此FMA运算目标数据类型的指数值的范围，或是大于可代表的任何中间储存空间，例如重命名暂存器，的指数值的范围，O_1位就会是二进位数一，否则O_1位就会是二进位数零。另外，U与O可经编码以表示四个可能的指数范围，其中至少一个编码会表示不足位，而至少一个编码会表示溢位。

[0116] 在传统一般乘法单元的实施方式中，U_1与O_1位会报告至例外事件控制逻辑。不过，在执行FMA1子运算时，调整后的乘法器45会输出U_1与O_1位至中间储存空间以供调整后的加法器50执行。

[0117] 端回进位指标产生器290会产生等待中的端回进位指标E_1位，其为Z、EffSub与SumSgn的函数。若是已确认的Z位的数值为二进位数一，表示部分乘积加法器240已经执行C的累加运算，已确认的EffSub变数显示此累加运算造成一有效减法，并且SumSgn显示所产生的未舍入非冗余值PNMant为正，E_1位就会是二进位数一。在其他情况下，E_1就会是二进位数零。

[0118] 结果向量端口280储存GMant的最重要位作为中间结果向量的中间结果尾数，而粘位产生器295与舍入位产生器300会使剩下较不重要的位(例如超出中间结果尾数的第53个位的位)减少至剩下舍入(R_1)与粘(S_1)位。粘位产生器295产生粘位S_1，其为SumSgn、Z、GMant的最不重要位、EffSub与XtraStky位的函数。舍入位产生器300产生舍入位R_1，其为GMant的最不重要位的函数。

[0119] 舍入快取

[0120] 舍入位端口305会输出各个位U_1、O_1、E_1、S_1、R_1与Z，借此，这些位就可以被其他执行单元(例如调整后的加法器50)使用来产生FMA运算最终的舍入后结果。为了方便说明，这些位在本文中都表示为舍入位，即使其中有部分位会在产生FMA运算最终结果的过程中有其他用途，又即使并非所有的位都用于舍入运算。举例来说，在某些实施方式中，O_1位就不会用于舍入运算。这些位可互换地被指为运算控制指标。举例来说，位Z与E，即指出那些后续运算需要执行；位U与O，即指出这些运算应如何执行。此外，这些位可表示为运算间歇状态值(calculation intermission state value)，因为他们提供一压缩格式(compact format)来表示与选择性地储存在调整后的乘法器45的FMA1子运算与调整后的加法器50的FMA2子运算间的间歇时间中的运算状态信息。

[0121] 这些位，无论被称为舍入位、运算控制指标、运算状态指标或其他，连同中间结果向量与累加数值C，除了运算元数值，还提供后续指令执行单元需要的任何事物，以产生算术上正确的最终结果。换句话说，中间结果向量与舍入位的结合可提供算术上正确表示FMA运算结果所需的任何结果，此运算结果与一具有±A*B±C形式但变为目标数据尺寸的无限精确FMA运算的运算结果无从辨别。

[0122] 依据本发明的一实施例，微处理器10用以将舍入位储存至舍入快取55内，并将舍入位通过转送总线40转送至其他指令执行单元，此舍入快取55亦可称为运算控制指标储存空间。依据本发明的另一实施例，微处理器10并不具有舍入快取55，而仅将舍入位通过转送总线40转送至其他指令执行单元。依据本发明的又一实施例，微处理器10将舍入位储存至舍入快取55内，但不提供转送总线40来将舍入位直接转送至其他的指令执行单元。

[0123] 指令快取55与所储存的指令位或运算控制指标为非架构，亦即其非为终端用户所能看见。相较之下，架构暂存器或架构指标(例如浮点状态字(floating point status word))则是可以被程序人员看见且指定为架构指令集的一部分的信号来源。

[0124] 在此所描述的舍入位仅为例示，而不同的实施方式会产生不同的舍入位组。举例来说，在另一实施例中，调整后的乘法器45亦包含一保护位产生器以产生一保护位G1。在另一实施例中，调整后的乘法器亦对一零结果(zero result)的符号执行一预运算，并将其值储存于舍入快取。若是调整后的加法器50后续运算的结果为零结果，舍入加法器50就会使用此储存的零结果符号指标来产生最终的带符号零结果。

[0125] 依据本发明的另一实施例，舍入快取55是一位于调整后的乘法器45外部的存储器储存空间。不过，在另一个不同的实施例中，此舍入快取55结合于调整后的乘法器55内。

[0126] 尤其是在一实施例中，此舍入快取55未经结果总线而独自连接至指令执行单元。有鉴于结果总线用以将结果从指令执行单元传送至一通用储存空间，舍入快取55独自连接至结果总线55而未经结果总线。此外，运算控制指标储存空间仅能为用于储存或载入运算控制指标的指令所存取。借此，即可通过输出指令结果的结果总线以外的其他机制来存取舍入快取55，举例来说，可通过舍入快取自身的导线。此外，亦可通过指令执行单元的输入运算 元端口外的其他机制来存取舍入快取55。

[0127] 在一实施例中，舍入快取55是一完全关联(fully associative)的可存取存储器，其具有的写入端口的数量相当于可平行分派的FMA1微指令的最大数量；其具有的读取端口的数量相当于可平行分派的FMA2微指令的最大数量；而其深度(项目数量)关联于指令排程的容量，与FMA1微指令分派后而在指令排程分派FMA2微指令前所能清空的最长时间周期(以时钟周期计)。另一实施例使用较小的舍入快取55，而微处理器10在舍入快去55内无法取得储存FMA1微指令的舍入位结果的空间时，重新执行FMA1微指令。

[0128] 快取的各个项目储存快取数据与相关联的旗标值(tag value)，此旗标值可与用以辨识储存有储存格式中间结果向量的重命名暂存器155的旗标值相同。在微处理器10准备(prepare)/拿取(fetch)供第二使用的运算元时，微处理器10使用重排缓冲器索引(ROB index)来由重命名暂存器155取回所储存的中间数据，与此相同的索引将会提供至舍入快取55，并提供中间结果150的剩下部分(即运算控制指标)。

[0129] 其优点在于，本发明的配置给舍入快取55的物理上储存项目的数量明显少于配置给重命名暂存器155的项目数量。重命名暂存器155的数量是进行中(in flight)的微指令数量与需要使非循序微处理器或设计的执行单元饱和所需的暂存器名称数量的函数。比较起来，舍入快取55所需的项目数量则为进行中的FMA微指令的可能数量的函数。因此，在一未受限的实例中，微处理器的核可提供六十五个重命名暂存器155，但只提供八个舍入快取55的项目提供给至多八个平行处理的算术运算。

[0130] 本发明的另一实施例扩充用以储存中间结果向量的重命名暂存器155(即扩大重命名暂存器的宽度)，以提供额外的位供舍入快取55使用。此实施例虽非本发明空间利用的最佳者，但亦属于本发明的范围内。

[0131] 舍入位连同中间结果指标IRVector包含储存格式中间结果150。此储存格 式依据一标准数据格式储存与/或传送未舍入标准化加总结果145的最重要位(即具有默认值的位)，并且将未舍入标准化加总结果145的剩下位(缩减或未缩减)，连同E_1、Z、U_1与O_1位，进行

储存与/或传送，因而相较于现有技术具有显著的优势。

[0132] 调整后的加法器

[0133] 现在请参照图6，调整后的加法器50包含一运算元调整器60、对准与条件逻辑330、以及一个与一单一位溢位移位逻辑345配对的远路径累加模组340。此运算元调整器160并包含一指数产生器335、一符号产生器365、一加法器舍入位产生器350、舍入位选择逻辑175与一舍入运算模组180。

[0134] 值得注意的是，在本发明的一实施例中，调整后的加法器50具有一分路设计而使其通过各自独立的近运算与远运算来计算结果，此技术为浮点运算设计的技术人员所已知。近路径的计算能力需要一个与一多位标准化移位器(未图示)配对的近路径加总模组(未图示)，此等能力未在图6中显现。在一实施例中，运算元C与D的输入指数值的差值位于{-1,0,+1}内而构成有效减法的一般加总运算会被导向至近路径16，其他的加法运算则会被导向至远路径170。其优点在于，本发明可使调整后的加法器50的所有的FMA2子运算都被导向至远路径170。

[0135] 调整后的加法器50提供一个或多个输入端口310以接收一个微指令与两个输入运算元。第一输入运算元D是一被减数或一第一加数。第二输入运算元C是一减数或一第二加数。在浮点运算的实施例中，各个输入运算元包含一输入符号、一指数、与一尾数值，分别以下标S、E与M表示。通过解译微指令，解码器315利用信号QS指出此运算究竟为一加法或是一减法运算。通过解译微指令(或是由微指令指令的一运算元参考)，解码器并可以利用信号M来指出此微指令是否支配一个特定的微操作可使调整后的加法器50执行FMA2子运算。

[0136] 在调整后的加法器50被赋予执行FMA2子元运算的任务时，调整后的加法器50接收由调整后的乘法器45先前执行相对应的FMA1子运算所产生的中间结果向量IRVector。因为中间结果向量IRVector的宽度仅为m个位，调整后的加法器50不需(而在一实施例中，不会)被调整成可接收或执行多于m个位的有效位数。因此，相较于以较宽位数呈现的IRVector，本实施例可以简化内部数据路径、累加模组340与调整后的加法器50的其他电路，并使其运作更有效率。此外，因为涉及块消去的累加运算由调整后的乘法器45完成，在调整后的加法器50的近/块消去路径上不需加入舍入运算逻辑来正确计算出FMA结果。

[0137] 在一实施例中，调整后的加法器50从重命名暂存器155接收IRVector。在另一实施例中，则是从转送总线40接收IRVector。而在图6所示的实施例中，IRVector被接收为运算元D。此调整后的加法器50接收累加数值C作为另一个运算元。

[0138] 若是M显示调整后的加法器50被赋予执行FMA2子元运算的任务，运算元调整器160就会在Z是二进位数一时，将输入运算元的一部分设定为等同于二进位数零，以显示C的累加运算已经由调整后的乘法器45执行。在一实施例中，各个指数、尾数与符号的栏位C_E、C_M与C_S均调整为零。在另一实施例中，只由指数与尾数的栏位C_E、C_M调整为零，至于符号的栏位C_S则维持原样。借此，调整后的加法器50即可将加数D加上一带符号的二进位数零。

[0139] 一个二进位数一的M位并通知调整后的加法器50接收由调整后的乘法器45所产生的舍入位，并将其并入储存格式中间结果150。

[0140] 在其他所有情况中，即Z为二进位数零或是M为二进位数零以显示调整后的加法器50被赋予传统累加运算的任务，运算元调整器160只会对指数与尾数栏位C_E与C_M进行传统浮点加法运算需要的调整。

[0141] 在一实施例中，运算元调整器160包含一对多工器接收Z的值，而在C_M 与零之间以及C_E与零之间进行选择。选定的值在图6中以C_M*与C_E*表示。随后，对准与条件逻辑330将对准与/或调节此选定值C_M*与第一运算尾数DM。

[0142] 接下来，远路径累加模组340将C_M*与DM相加。在一实施例中，此累加模组340为一双加加法器，以提供总数与渐增总数。在一实施例中，此累加模组340利用1'补数(one's complement)来执行有效减法。若是此总数会在尾数栏位产生一位的溢位，溢位移位逻辑345就会条件地将总数移动一个位，以使结果值完成舍入运算的准备。

[0143] 指数产生器335利用选定的指数值C_E*、第一运算元指数DE、与由溢位移位逻辑345产生的移位量来产生一最终指数FExp。

[0144] 符号产生器365依据由第一与第二运算元C_S与D_S、加/减运算子QS与加总结果符号构成的函数来产生一最终符号FSgn。

[0145] 在另一实施例中，运算元调整器160以选择器逻辑取代。当输入解码器显示加法器正在执行FMA2子运算而Z是二进位数一以显示C的累加运算已执行，此选择器逻辑会使第一运算元D直接转送至舍入运算模组180，但使加总逻辑维持在休眠状态(quiescent state)。

[0146] 调整后的加法器50的逻辑产生一组自己的舍入位R_2、S_2、U_2、O_2与E_2。当M显示调整后的加法器50被赋予执行FMA2子运算的任务时，调整后的加法器50也会接收多个由执行FMA1子运算的调整后的乘法器45事先产生的舍入位R_1、S_1、U_1、O_1、Z与E_1。

[0147] 在M为二进位数一的情况下，舍入位选择逻辑175会确认来自调整后的乘法器45的舍入位E_1、R_1与S_1、来自调整后的加法器50的舍入位E_2、R_2与S_2、或是二者的某些混合或组合，将会被加法器的舍入运算模组180用以产生最终的舍入后尾数结果。举例来说，若是所执行的运算并非FMA2子运算(即M＝0)，舍入运算模组180就会使用加法器产生的舍入位E_2、R_2与S_2。另外，若是累加运算由调整后的乘法器45执行(即M＝1且Z＝1)，并且不存在不足位 的情形(即UM＝0)，就由乘法器产生的舍入位E_1、R_1与S_1来提供舍入运算模组180产生最终舍入后结果所需的任何物件。

[0148] 此可变位置的舍入运算模组作为调整后的加法器50的远计算功能的部分来设置。而在一实施例中，此舍入运算模组配合由1'补数有效减法所造成的正差值舍入运算，并且还配合由非有效减法的加总运算所造成的正总数舍入运算。此舍入运算模组180以类似于传统单一加/减单元执行的方式，执行选定的舍入位RX、黏位SX，而在有提供的时候也会执行保护位G_X(未图示)。此舍入运算模组180由传统设计进行修改以接收至少一补充输入，即选定的端回进位位EX，而若是由调整后的乘法器45执行1'补数有效减法，即可显示需要一端回进位校正。使用选定的RX、SX与EX输入，舍入运算模组180可正确地对中间结果向量与带符号零的加总执行舍入运算，以产生正确而符合IEEE标准的结果。此为浮点运算设计的技术领域人员所能理解。

[0149] 如前述，调整后的加法器50需要近路径165来执行某些类型的传统累加运算，但不需要近路径165来执行本文所述的FMA运算。因此，在执行本文所描述的FMA运算类型时，近路径逻辑165会在FMA运算的过程中维持休眠状态以降低耗能。

[0150] 第一与第二FMA子运算

[0151] 图7至10显示本发明利用一第一FMA子运算(FMA1)与一第二FMA子运算(FMA2)执行一不可分割分路乘积-累加运算的方法的一实施例。其中，FMA2子运算并非临时性相接于

FMA1子运算，也不是物理性上相接于FMA2子运算。

[0152] 图7显示FMA1子运算的一路径确认部分。在步骤408中，FMA1子运算确认EffSub变数。当EffSub为二进位数一，即显示是否累加数运算元与乘数运算元的乘积的累加运算会构成一有效减法。在步骤411中，FMA1子运算选择性地对累加数运算元执行位反相的运算。在步骤414中，FMA1子运算计算 ExpDelta。ExpDelta等于乘数与被乘数的指数的加总减去累加数的指数与一指数偏移。ExpDelta不只确认乘数尾数与累加数尾数的相对对准以供加法运算之用，也会配合EffSub变数来确认累加数运算元的累加运算是否将由FMA1子运算执行。

[0153] 在步骤417中，FMA1子运算确认路径控制信号Z。当数值为二进位数一，即表示有一个累加数运算元的加总运算，会利用调整后的乘法器45电路，在FMA1子运算中执行。在一实施例中，FMA1子运算在ExpDelta大于或等于负一时，将二进位数一指派给Z，而在ExpSub为一且ExpDelta为负二时，也会将二进位数一指派给Z。其他的实施方式会以不同方式分割ExpDelat与EffSub的数字空间。

[0154] 图8是FMA1子运算的乘法与条件累加部分的方块示意图。在步骤420中，FMA1子运算选择一累加路径供累加运算元使用。若是Z为二进位数零，如步骤426所示，FMA1子运算就会计算乘数运算元的部分乘积的加总，而不加上累加数运算元。另外，若是Z为二进位数一，如步骤423所示，FMA1子运算会调整选择性互补累加数尾数的对准值，其调整量为ExpDelta值的函数。就一实施例而言，此调整量等于ExpDelta加上一移位常数。

[0155] 在步骤426/429中，FMA1子运算执行一第一累加运算，其为(a)乘数和被乘数运算元的部分乘积(即步骤426)，或是(b)累加数运算元与乘数和被乘数运算元的部分乘积(即步骤429)。在步骤432中，FMA1子运算条件执行一前导位预测，来预测总数的最重要前导位所需要的消去。前导位预测的运算限于类型1的FMA运算75，并且会与步骤429中部分的加总运算平行执行。另外，前导位预测逻辑可连接至步骤426或步骤429以处理其运算结果。

[0156] 经过步骤426或429，以及步骤432的执行后，FMA1子运算会产生一未经舍入、非冗余的标准化加总结果145(如步骤435所示)。接下来，FMA1子运算会产生一储存格式中间结果150(如步骤438所示)。一旦储存格式中间结 果150被储存或分派至转送总线40，FMA1子运算就会终止，并释放执行FMA1子运算的资源(例如作为调整后的乘法器45的指令执行单元)给其他与FMA运算无关的运算使用。所属技术领域人员当可理解，此技术亦可同样适用于可同时通过连续阶段执行多个运算的管线乘法器。

[0157] 图9A与9B详细说明产生储存格式中间结果150的程序。在步骤441中，FMA1子运算确认是否因累加数的累加运算构成有效减法，而存在待定的端回进位校正。若是Z与EffSub均为二进位数一(即类型1的FMA运算75或是类型4的FMA运算90)，并且由步骤435所产生的未经舍入非冗余的结果为正，FMA1子运算就会将二进位数一指派给变数E_1。

[0158] 在步骤444中，FMA1子运算通过对尾数执行位反相的运算，而产生一原始尾数结果(germinal mantissa result，GMant)。若运算结果为负，就通过移位，将尾数标准化为一标准储存格式。

[0159] 在步骤447中，FMA1子运算产生一中间结果符号位(IRSgn)。若是E为二进位数零，而Z与EffSub均为二进位数一，IRSgn就会是被乘数与乘数的符号位的反逻辑互斥或结果。否则，IRSgn就会是被乘数与乘数的符号位的逻辑互斥或结果。

[0160] 在步骤453中，FMA1子运算将SC加上乘数与被乘数的指数值加总，再减去ExpBias，以产生PNExp。

[0161] 在步骤456中，FMA1子运算通过减少PNExp来处理PNMant的标准化运算，借以产生中间结果指数值(IRExp)。

[0162] 在步骤459中，FMA1子运算确认中间不足位(U_1)与中间溢位(O_1)位。

[0163] 在步骤462中，FMA1子运算由原始尾数(GMant)的最重要位产生一中间结果尾数(IRMant)。

[0164] 在步骤465中，FMA1子运算将构成中间结果向量IRVector的IRSgn、IRMant与IRExp，储存至储存空间，例如一重命名暂存器。

[0165] 在步骤468中，FMA1子运算会将GMant的LSBs与部分乘积加法器240的移出位(Xtrastky)缩减至舍入位(R_1)与粘位(S_1)。而在另一实施例中，还包含一保护位(G1)。

[0166] 在步骤471中，FMA1子运算将R_1、S_1、E_1、Z、U_1与O_1位纪录于舍入快取55，而在提供有G1位时，也会一并记录于舍入快取55。

[0167] 图10是一方块图显示本发明不可分割分路FMA运算的一第二FMA子运算。

[0168] 在步骤474中，FMA2子运算接收先前储存在例如重命名暂存器的储存空间内的中间结果向量IRVector。另外，FMA2子运算亦可由转送总线接收IRVector。

[0169] 在步骤477中，FMA2子运算接收先前储存在例如舍入快取55的储存空间内的舍入位。另外，FMA2子运算亦可由转送总线接收舍入位。

[0170] 在步骤480中，FMA2子运算接收累加数输入值。

[0171] 在决定步骤483中，FMA2子运算检视步骤474所接收的Z位。若是Z位为二进位数一(或真)表示累加数的加总运算已经由FMA1子运算执行，此流程就会前进至步骤486。否则就会前进至步骤489。

[0172] 在步骤486中，FMA2子运算将累加数输入值的指数与尾数栏位调整为零。在一实施例中，FMA2子运算并不调整输入累加数的符号位。随后在步骤492中，此FMA2子运算会将计算中间结果指数与一带符号零运算元的总数。接下来前进至步骤494。

[0173] 在步骤489中，FMA2子运算计算中间结果指数与累加数的总数。接下来前进至步骤494。

[0174] 在步骤494中，FMA2子运算利用FMA1子运算产生的Z、U_1、O_1位以及FMA2子运算产生的U_2与O_2位，从舍入位E_1、E_2、R_1、R_2、S_1与S_2中，选择会用来对总数的尾数进行正确舍入运算的舍入位。

[0175] 在步骤496中，FMA2子运算利用选定的舍入位来对总数进行正确的舍入运算。此FMA2子运算在执行尾数舍入程序的同时，选择性地使IRExp递增(步骤498)。如此，FMA2子运算即可产生一最终经舍入的结果。

[0176] 图7至10中所描述的部分步骤可不需图示顺序执行。此外，第七至十图中所描述的部分步骤亦可平行执行。

[0177] 运算类型的应用

[0178] 此章节说明前述应用于图2的五种不同运算“类型”的各种变数数值间的功能性关联，尤其着重于PNMant的运算、符号与标准化处理以及与各个数据类型相关的EffSub、ExpDelta、Z、E与IntSgn的数值。

[0179] 第一类型

[0180] 如图2所示，类型1的FMA运算75的特点在于，此运算涉及一有效减法(因此，EffSub＝1)，而调整后的乘法器45被选定以执行C的累加运算(因此，Z＝1)，而C的大小相当接近A和B的乘积(即-2<＝ExpDelta<＝1)而会造成块消去的产生。

[0181] 因为累加运算将会于调整后的乘法器45内执行而造成有效减法(即EffSub＝1且Z＝1)，累加数对准与射入模组220会在将累加数运算元尾数值C_M射入部分乘积加法器240前，对累加数运算元尾数值C_M造成与/或选择一位反相运算。相对于部分乘积，此累加数对准与射入模组220会利用ExpDelta，来对准部分乘积加法器240内的累加数尾数。

[0182] 随后，对于未经舍入的非冗余数值145(即PNMant)的完整加总运算依照传统乘法执行单元通常采用的方法来执行，此执行包含部分乘积的加总运算内，额外被选择性位求与相对准的累加数输入值。因而PNMant即可以1'补数的形式，表示乘数与被乘数尾数值以及累加数尾数值间的算数差。

[0183] PNMant可为正或负。若是PNMane为正就需要端回进位，而等待中的端回进位指标E_1就会被指派为二进位数一。若是PNMant为负就不需要端回进 位，而E_1就会被指派为二进位数零。E_1被指派的值不仅为PNMant的函数，亦为Z与EffSub的值的函数，而在类型1的运算75中，Z与EffSub会是二进位数一。

[0184] 在部分乘积与累加数输入加总运算的同时，也会执行前导位预测来对任何最重要前导位需要的任何消去进行预测。如前述，于本发明一实施例中，此运算由一与最终加法器125并行的电路，于执行PNMant的加总运算时执行。

[0185] 如浮点运算设计的技术领域人员所能理解，即使没有发生前导位的相减消去，可能还是需要对PNMant执行一个零、一或二位位置的标准化运算，利用SC至PNExp来使其对准于所需的储存格式，以供本发明描述与使用的中间结果150之用。若存在块消去，显然就会需要更多的移位运算。同样地，若是PNMant为负，就会对此数值执行位反相的运算。对PNMant执行选择性标准化与位反相运算用以产生原始尾数值GMant，其最重要m位变成中间结果尾数IRMant。

[0186] 中间结果符号IRSgn不是被乘数符号位A_S与乘数符号位B_S的逻辑XOR运算结果，就是其逻辑NXOR运算结果，端视E_1的数值而定。若是E_1为二进位数一，IRSgn会被计算为被乘数符号位与乘数符号位的逻辑XOR运算结果。若是E_1为二进位数零，IRSgn就会被计算为被乘数符号位与乘数符号位的逻辑XOR运算结果。

[0187] 回到FMA2运算，调整后的加法器50会接收包含路径控制信号Z在内的已储存或转送的舍入位，因为Z为1，中间结果向量IRVector需要执行舍入运算，而会对最终的乘积-累加结果产生些微调整。在一实施例中，调整后的加法器50将中间结果向量IRVector与一零运算元(或是在另一实施例中，一带符号二进位数零的运算元)加总，而非与所提供的第二运算元，即累加数C，做加总运算。

[0188] 在最后的执行步骤中，调整后的加法器50会在加总运算与完成舍入运算前对接收到的IRExp进行调整，以涵盖较大的数字范围，例如可涵盖FMA运算的目标数据类型的不足位与溢位指数范围。在接收到的数值Z＝1位时，调整后的加法器50会以采取一个使IRExp递增、而大部分均为传统方式的方法，利用所接收到的R、S、U、O与E位来对IRVector执行舍入运算。

[0189] 第二类型

[0190] 如图2所示，类型2的FMA运算80的特点在于，此运算不涉及有效减法(因此，EffSub＝0)，调整后的乘法器45被选定以执行C的累加运算(因此，Z＝1)，而相较于A和B的乘积，C的大小相当小。

[0191] 因为此运算不会造成有效减法(即EffSub＝0)，累加数对准与射入逻辑220就不会在将累加数运算元尾数值C_M射入部分乘积加法器240前，对累加数运算元尾数值C_M造成与/或选择一位反相运算。

[0192] 累加数对准与射入模组220利用ExpDelta来使累加数尾数对准部分乘积，以将累加数尾数射入部分乘积加法器240。

[0193] 在此将不会产生负的PNMant。此外，所产生的正的PNMant并非1'补数的减法运算的结果，因而不需端回进位校正。因此，等待中的端回进位指标E_1就会被指派为二进位数零。

[0194] 因为并非一有效减法运算，此运算中将不会产生前导位的减法块消去，因而不需执行前导位预测来预测此等消去。反之，前导位预测则可用以依据SC至PNExp的贡献程度，来预测所需的0、1、或2位位置的标准化运算。

[0195] 如同浮点运算设计的技术领域人员所能理解，A和B乘积与C的加总会产生一具有算术显著性或比重、较乘数与被乘数的乘积大于一数位位置的溢位情形，因此需要对PNMant执行零、一或二位位置的标准化运算，来使数值对准用于本发明所描述与利用的中间结果的储存格式。此标准化运算会产生原始尾数值GMant，而其最重要m位会成为中间结果尾数IRMant。

[0196] 预标准化指数PNExp通过将输入的乘数与被乘数指数值相加，再减去任何指数偏移值，最后再加上SC＝2，此数值于Z＝1的情况合下依据最负的ExpDelta而定。如图2对于类型2的运算的描述可知，C的大小并不会明显大于A和B的乘积，因此，所造成的总数会等于或大于所输入的累加数。

[0197] 因为此运算并非有效减法(即EffSub＝0)，中间结果符号IRSgn视为被乘数符号位A_S与乘数符号位B_S的XOR逻辑运算结果。

[0198] 回到FMA2运算，调整后的加法器50会接收包含路径控制信号Z的已储存或转送的舍入位。因为Z是二进位数一，中间结果向量IRVector只需要一些最后处理，主要就是舍入运算，即可产生最终的乘积-累加结果。在一实施例中，此调整后的加法器50将中间结果向量IRVector与一零运算元(在另一实施例中，可为一带符号二进位数零运算元)做加总，而非与所提供的第二运算元，即累加数C，做加总运算。

[0199] 在最后的执行步骤中，调整后的加法器50会对接收到的IRExp进行调整，以涵盖较大的数字范围，例如可涵盖FMA运算的目标数据类型的不足位与溢位指数范围。此调整后的加法器50会以采取一个使IRExp递增、而大部分均为传统方式的方法，来对IRVector执行舍入运算以产生最终的正确结果。

[0200] 第三类型

[0201] 如图2所示，类型3的FMA运算85的特点在于，此运算不涉及有效减法(因此，EffSub＝0)，调整后的加法器50被选定以执行C的累加运算(因此，Z＝0)，而相较于A和B的乘积，C的大小相当大。

[0202] 因此，EffSub为二进位数零。此外，路径控制信号Z亦为二进位数零，以显示累加数运算元的加总运算尚未执行。因为Z与EffSub均为二进位数零，等待中的端回进位指标E_1会被指派为二进位数零。

[0203] 因为Z是二进位数零，累加数对准与射入逻辑220不会将乘数单元部分乘积加总数内的累加数输入的尾数进行对准。不过，累加数对准与射入逻辑220 会使此对准输入的算术值为零。

[0204] 随后，对于部分乘积与未经舍入的非冗余数值145的完整加总运算依照传统乘法执行单元通常采用的方法来执行，此方法不包含输入累加数的尾数值。因为此FMA运算类并非有效减法(即EffSub＝0)，此加总运算将不会产生正的PNMant，并由SumSgn表示。此外，此正值的PNMant并非1'补数减法运算的结果，因而不需要使用端回进位校正。

[0205] 因为此运算并非一有效减法运算，此运算中将不会发生前导位的减法块消去，因而不需执行前导位预测来预测此等消去。

[0206] A和B的乘积会在乘数与被乘数的尾数乘积产生一位数位置的算术溢位。因而需要对此正的未经舍入非冗余值执行零或一位位置的标准化运算，来使此数值对准本发明所描述或使用的中间结果格式。此标准化运算会产生原始尾数值GMant，而其最重要m位会成为中间结果尾数IRMant。

[0207] 因为事先确认的路径控制信号Z为二进位数零，显示累加运算尚未被执行，此中间结果符号IRSgn就会被视为被乘数符号位A_S与乘数符号位B_S的XOR逻辑运算结果。

[0208] 回到FMA2运算，调整后的加法器50接收包含Z在内的已储存或转送的舍入位。因为Z为二进位数零，调整后的加法器50会使中间结果向量，即第一运算元，与累加数C，即第二运算元，做加总运算。

[0209] 在执行此累加运算之前，此调整后的加法器50可调整IRExp以涵盖较大的数字范围，例如可涵盖FMA运算的目标数据类型的不足位与溢位指数范围。因为此运算是由累加数数值主导运算结果的类型3的运算85，IRExp将会比累加数输入指数值来的小。

[0210] 此运算有利于使调整后的加法器50的远路径加总运算的二个运算元生效。在远路径加总运算中，具有较小指数值的运算元的尾数于对准程序中向右移位。任何如此移位超过所需舍入位的尾数位就会被用于舍入运算。因为 累加数会主导运算结果，因而可以不用提供位给舍入运算，而能简化必要的舍入运算。

[0211] 调整后的加法器50会使用由调整后的加法器50所执行的部分运算所产生的G2(如果有的话)、R_2、S_2与E_2(具有二进位数值零)舍入位，连同R_1、S_1与E_1，来对中间结果与累加数输入值的总数进行舍入运算，以产生FMA运算的最终经舍入的正确结果，此为熟习浮点运算设计领域技术人员所能理解。

[0212] 第四类型

[0213] 如图2所示，类型4的FMA运算90的特点在于，此运算涉及有效减法(因此，EffSub＝1)，调整后的乘法器45被选定以执行C的累加运算(因此，Z＝1)，而相较于A和B的乘积，C的大小相当小。

[0214] 因为累加运算将会于调整后的乘法器45内执行而造成有效减法(即EffSub＝1且Z＝1)，累加数对准与射入模组220会在将累加数运算元尾数值C_M射入部分乘积加法器240前，对累加数运算元尾数值C_M造成与/或选择一位反相运算。相对于部分乘积，此累加数对

准与射入模组220会利用ExpDelta，来对准部分乘积加法器240内的累加数尾数。

[0215] 因为A和B的乘积的大小明显大于C，此运算中将不会发生前导位的减法块消去，因而不需执行前导位预测来预测此等消去。

[0216] 此外，加总程序会产生正的PNMant。因此，待定的端回进位指标E_1会被指派为二进位数一，之后会通知调整后的加法器50还需要对中间结果尾数执行端回进位校正。

[0217] 如同浮点运算设计领域具有合理技术人员所能理解，PNMant可能需要依据所提供的SC至PNExp，执行零、一或二位位置的移位或标准化运算，使其对准于本发明所述或使用的中间结果所需的储存格式。随后，此标准化运算会选择性地执行于此未经舍入的非冗余值，以产生原始尾数值GMant，其 最重要着m位会成为中间结果尾数IRMant。

[0218] 因为类型4的运算90涉及C的加总运算(即Z＝1)，而此加总运算会构成有效减法(即EffSub＝1)，在需要端回进位(即E_1为1)的情况合下产生正的PNMant，中间结果符号IRSgn会被视为被乘数符号位A_S与乘数符号位B_S的XOR逻辑运算结果。

[0219] 回到FMA2运算，调整后的加法器50接收包含路径控制信号Z在内的已储存或转送的舍入位。因为Z为1，中间结果向量IRVector只需一些最后处理，主要就是舍入运算，即可产生最终的乘积-累加结果。在一实施例中，此调整后的加法器50将中间结果向量IRVector与一零运算元(在另一实施例中，可为一带符号二进位数零运算元)做加总，而非与所提供的第二运算元，即累加数C，做加总运算。

[0220] 在执行零(或二进位数带符号零)的累加运算前，调整后的加法器会调整IRExp以涵盖较大的数字范围，例如可涵盖FMA运算的目标数据类型的不足位与溢位指数范围。

[0221] 响应储存格式中间结果150内所接收的E位二进位数值，此运算需要依据第一微指令执行过程中执行的1'补数有效减法产生端回进位校正。因此，此E位连同储存格式中间结果150的G1(如果有的话)、R_1与S_1位，做为调整后的加法器50执行单元的调整后的舍入逻辑的补充输入。

[0222] 随后，此调整后的舍入逻辑利用G1(如果存在的话)、R_1、S_1与E_1补充输入来计算中间结果向量与带符号零的加总的正确舍入，以对类型4的FMA运算产生正确的运算结果。而此技术当为熟习浮点运算设计领域的技术人员所能理解。

[0223] 第五类型

[0224] 如图2所示，类型5的FMA运算的特点在于，此运算涉及有效减法(因此，EffSub＝1)，调整后的加法器50被选定以执行C的累加运算(因此，Z＝0)，而 相较于A和B的乘积，C的大小相当大。

[0225] 因为累加运算不会于调整后的乘法器45内执行，累加数对准与射入逻辑220不会对准部分乘积加总数内的C_X，或是使此对准输入的算术值为零。此调整后的乘法器45依据现有技术的乘法执行单元的常用方法来执行部分乘积与PNMant的完整加总运算。

[0226] 因为已执行C的加总运算，此运算中将不会发生前导位的减法块消去，因而不需执行前导位预测来预测此等消去。此外，虽然会产生正的PNMant，此数值并非1'补数减法的运算结果。因此，此运算不需要端回进位校正，而E_1会被指派为二进位数零。

[0227] 如同熟习浮点运算设计领域的技术人员所能理解，PNMant可能需要依据所提供的SC至PNExp，执行零、一或二位位置的移位或标准化运算，使其对准于中间结果150所需的储存格式。此标准化运算会产生原始尾数值GMant，其最重要m位会成为中间结果尾数IRMant。

[0228] 因为类型5的运算不涉及C的加总运算(即Z＝0),中间结果符号IRSgn会被视为被乘数符号位A_S与乘数符号位B_S的XOR逻辑运算结果。

[0229] 回到FMA2运算,调整后的加法器50接收包含Z在内的已储存或转送的舍入位。因为Z为0,中间结果向量IRVector需要与累加数C加总以产生最终的乘积-累加运算结果。

[0230] 因为类型5的运算由累加数值主导运算结果,IRExp会小于累加数输入指数值,此运算有利于使调整后的加法器50的远路径加总运算的二个运算元生效。在远路径加总运算中,具有较小指数值的运算元的尾数于对准程序中向右移位,任何如此移位超过所需舍入位的尾数位就会被用于舍入运算。因为累加数会主导运算结果,因而可以不用提供位给舍入运算,而能简化必要的舍入运算。

[0231] 因为由储存格式中间结果150接收的待执行端回进位指标E_1是二进位数 零,因此,FMA1运算没有留下任何待执行的端回进位校正。是以,E_1位连同储存格式中间结果150的R_1与S_1位,以及G1位(如果存在的话),做为调整后的加法器50执行单元的调整后的舍入位的补充输入。

[0232] 然而,由调整后的加法器50所执行的累加运算可能会另外导致1'补数有效减法的产生。如此,调整后的舍入逻辑即可产生包含端回进位在内的舍入位,来计算出中间结果向量与累加器输入值的加总的正确舍入值,以产生FMA运算的正确结果。而此技术当为熟习浮点运算设计领域的技术人员所能理解。

[0233] 特定微指令

[0234] 在本发明的另一个面向中,转译器与/或微码只读存储器(ROM)20用以将FMA指令转译或转换为第一与第二特定微指令,分别由乘法与加法单元执行。举例来说,第一(或更多)特定微指令可以在一类似于将传统乘法单元进行最小限度的调整,以符合所述目的的乘法执行单元内执行。而第二(或更多)特定微指令可以在一类似于将传统家法单元进行最小限度的调整,以符合所述目的的加法执行单元内执行。

[0235] 图11显示本发明将融合FMA指令535通过FMA指令转译或转换,产生第一与第二特定微指令553与571的一实施例。就一未受限的范例而言,此融合FMA指令535包含一指令操作码栏538、一目标栏541、一第一运算元(被乘数)栏544、一第二运算元(乘数)栏547、与一第三运算元(累加数)栏550。

[0236] 依据操作码栏538的指示,此FMA运算可为一乘-加(multiply-add)、一乘-减(multiply-subtract)、一负乘-加(negative multiply-add)、或一负乘-减(negative multiply-subtract)指令。如同存在许多不同类型的FMA指令535,也会存在许多类型的第一特定微指令553,如乘-加、乘-减、负乘-加、或负乘-减微指令。这些类型的特征,如果有的话,会反映在相关的微指令553的操作码栏556中。

[0237] 第一特定微指令553会引导第一至第五类型的FMA运算所需的部分算术 运算的实行。依据类型的不同,第一特定微指令553所执行的特殊运算亦有差异。第一特定微指令553分配给第一执行单元,例如前文所述的调整后的乘法器45。

[0238] 第二特定微指令571会引导第一至第五类型的FMA运算所需的剩余算术运算的实行。依据类型的不同,第二特定微指令571所执行的特殊运算亦有差异。在本实施例中,第二特定微指令571分配给第二执行单元,例如前文所述的调整后的加法器50。为了方便浮点乘-加融合运算或浮点乘-减融合运算的实行,此第二特定微指令571可具有一子类型,例如

加或减。

[0239] 尤其是，此第一特定微指令553会将第一、第二与第三输入运算元544、547与550，分别特定为被乘数运算元A、乘数运算元B与累加数运算元C。此第一特定微指令还可以特定一目标栏559，指向暂时暂存器。又或者，此目标暂存器559可为隐含者。

[0240] 第一特定微指令553会引导FMA1子运算执行，即A和B的部分乘积与在某些条件下与C的加总，以产生未经舍入的储存格式中间结果150。此第一特定微指令553也会引导变数EffSub与ExpDelta的确认动作，借以对一组预先确定的EffSub与ExpDelta值，产生指派给Z位的数值二进位数一。此数值会依序控制许多相关程序。

[0241] 二进位数一的Z位指出此对累加数位的加总运算将会于第一运算中执行，而不须由第二微指令来执行。随后，此Z位的指定与ExpDelta会用于使部分乘积加法器240内被选择性互补累加数尾数进行对准，此部分乘积加法器240经适当调整使能接收此额外事项。

[0242] 第一特定微指令553并引导一完整加总运算对此未经舍入的非冗余值(PNMant)执行。此完整加总运算依据传统乘法执行单元的通常方法，但在部分乘积的加总运算中，纳入额外的选择性位反相、已对准的累加数输入值C_M或$\overline{C_M}$。若是PNum为负，此条件就会由信号SumSgn表示。

[0243] 第一特定微指令553并引导PNMant的移位与位反相运算，以产生一原始尾数值(GMant)，接下来会执行GMant的缩减以产生储存格式中间结果150的中间结果尾数(IMant)。因此，此中间结果尾数IMant会是此EffSub指定的运算所产生的1'补数算术差的标准化绝对值，但不执行任何端回进位校正。

[0244] 第一特定微指令553并引导中间结果指数值的运算。首先依据Z为二进位数一时，ExpDelta的最负值，产生一预标准化指数值(PNExp)，其数值等于被乘数指数A_E与乘数指数B_E的加总、减去指数偏移值ExpBias、再加上一移位常数SC。然后，从PNExp的数值减去由标准化移位器130执行的尾数标准化运算所占据的量，以产生中间结果指数值(IRExp)。

[0245] 第一特定微指令553并引导中间结果符号IRSgn的运算。此中间结果符号IRSgn，连同中间结果尾数IRMant与中间结果指数IRExp，构成储存格式中间结果150的向量IRVector。

[0246] 第一特定微指令553并会产生包含Z在内的许多舍入位。GMant未纳入中间结果尾数的最不重要位会被缩减至由舍入(R)与粘(S)位表示，而在一实施例中，还包含保护(G)位。若是部分乘积加法器240已经将C与A和B的乘积累加，并且此运算是产生一正的PNMant值的一有效减法而，端回进位(end-around-carry bit)E就会被指派为二进位数一，以显示需要执行端回进位。此第一特定微指令并会使中间不足位(U)与中间溢位(O)位被确认。

[0247] 最后，在一实施例中，第一特定微指令553会使储存格式中间结果150向量IRVector储存于存储器，在另一实施例会使其转送，而在又一实施例，则会使其同时储存与转送。相同地，在一实施例中，第一特定微指令553会使舍入位储存于存储器，在另一实施例中会使其转送，而在又一实施例中，则会使其同时储存与转送。如此可使负责执行第一特定微指令的执行单元，在执行第一FMA微指令后而在执行第二微指令前，执行其他与FMA运算无关的运算。

[0248] 第二特定微指令571提供一操作码574并分别特定第一与第二输入加法器运算元580与583。第二特定微指令571会促使FMA2运算执行。若是C的累加并非由第一特定微指令

553造成，此运算还包含C与中间结果尾数的条件累加。第二特定微指令571并会产生FMA运算的最终舍入后结果。

[0249] 第一累加数运算元580将第一特定微指令553所产生的乘积作为其数值，第二累加数运算元583将第一特定微指令所指定的相同累加数值作为其数值。在一实施例中，第二特定微指令571的源运算元栏580指向与第一特定微指令553的目标栏599相同的暂存器。第二特定微指令571并特定一目标暂存器577，在一实施例中，此目标暂存器与FMA指令535的目标栏541为同一个暂存器。

[0250] 结论

[0251] 虽然此实施方式在有效减法时提供1'补数的累加运算，熟习算术或浮点运算设计领域的技术人员当可理解，在另一实施方式中，在有效减法时利用本发明的方法来使用2'补数的累加运算。

[0252] 本发明具有许多优点。本发明相容于IEEE，并提供其他实施方式明显无法提供的校正给所需的FMA算术结果，尤其是响应IEEE舍入运算的要求。

[0253] 本发明通过维持可分别取用的乘法器与加法器单元，最大化供指令分配的各自独立算术功能单元的可用性，而让计算机处理器在一定实施成本下，可以更充分地开发指令平行处理。换言之，本发明让最少的硬件达到最大可能性的同步利用，使其能以最快速度完成预期中最频繁的运算，此处理方式可改善算数运算结果的产出。通过将所需的特定类型的第一与第二(或更多)微指令进行分配，并使其以暂时性与/或物理性各自独立的方式分别执行，因此，虽然此等用于FMA的第一微指令会被分配至乘法功能单元执行，第二或其他更多无关的微指令则可同时被分配至一个或多个加法功能单元。

[0254] 同样地，虽然此等用于FMA的第二微指令会配分配至功能方块单元，任 何需要乘法功能的其他无关的微指令则可同时被分配至乘法功能单元。

[0255] 如此，依据所需的整体效能与所需的系统指令平行处理能力，所提供的乘法与加法功能单元的数量可更具弹性。因而相较于一个整体的FMA硬件，可减少每个功能单元的设置成本。计算机系统重排微指令的能力也会提升，而能降低成本与功耗。

[0256] 相较于其他设计，本发明不需要使用大型或特殊目的的硬件即可减少指令延迟。其他FMA硬件实施的设计都需要使用大型与复杂的电路功能，如先行标准化运算(anticipatory normalization)、先行加法运算、先行符号运算、与复杂的舍入电路。这些复杂元件常常会成为最终设计中关键的耗时路径、会在运算过程中消耗额外的能量、并在物理上会占据电路设置空间。

[0257] 本发明不需在大型的FMA硬件内设置特殊旁路电路或模态以减少如现有技术所提供的简单加法或乘法指令的延迟。

[0258] 本发明的其他实施方式，可在特殊类型的第一微指令的执行过程，执行或多或少的算术运算，可在特殊类型的第二微指令的执行过程，执行或多或少的算术运算。亦即配置给这些微指令的运算可为不同。如此，其他这些实施方式可对任一/任何需要的独立运算单元进行或多或少的调整。如此，这些其他实施方式可储存或多或少的中间结果于舍入快取内，并可将或多或少的中间结果转送至第二为指令。

[0259] 其他实施方式中，所述的舍入快取可为可定址暂存器位、内容可存取存储器、伫列储存空间、或对映功能。

[0260] 其他实施方式可提供多个独立的硬件或执行单元来执行第一微指令，与/或提供多个独立的硬件或执行单元来执行第二微指令。同样地，如果有利的话，这些实施方式也可以提供多个舍入快取给不同的源码指令流或数据流、或是多核计算机处理器的各种实施方式。

[0261] 此实施方式用于超纯量、非循序指令分配，不过其他实施方式亦可用于 依序指令分配的情形，例如，通过从指令快取移除以及提供至数据转送网路的过程，可将指令从所提供的乘法运算单元分配至独立的加法运算单元。本发明对于FMA运算分类的例示，以及本发明提及的所需最少量硬件调整，在依序指令分配的应用中，亦有其优点。虽然本说明书将FMA运算区分为五个类型，不过其他划分方式，无论是较少、较多、与/或使用不同类型，均属本发明的范畴。

[0262] 此外，虽然本说明书中已描述不同的调整后的乘法与加法单元以执行FMA运算，在本发明的一实施例中，亦可使用一乘积-累加单元响应第一乘积-累加指令来执行第一乘积-累加子运算，并将其结果储存至外部储存空间，而此乘积-累加单元会再响应第二乘积-累加指令来执行第二乘积-累加子运算。

[0263] 本发明可用于FAM运算的单指令流多数据(SIMD)实施方式，此等方式有时会被称为向量指令类型或向量FMA运算。并且此方式会有多个调整后的乘法器的实例与多个调整后的加法器的实例。在一实施例中，本发明使用单一舍入快取来配合一个SIMD应用的需求。而在另一实施例中，则可使用多个舍入快取来配合多个SIMD应用的需求。

[0264] 虽然本发明关于需要一乘法运算连同后续加法或累加运算的浮点融合乘法加法运算，不过，本发明所提供的方法亦可用于其他实施方式，尤其是针对某些中间结果使用快取的技术、针对需要两个以上连锁(chained)的算术运算的计算、针对不同算术运算、或以不同顺序执行这些算术运算。举例来说，某些时候可能需要将这些方法应用于其他算术运算的混合(即，涉及两个以上的算术操作子或三个以上的运算元的算术运算)，例如连锁的乘法-乘法-加法运算或乘法-加法-加法运算，以提升算术准确性或提升运算产出。此外，从其他面向来看，本发明亦可应用于整数算术运算。举例来说，舍入至特定位位置的整数运算可区分为第一与第二子运算，其中第一子运算产生一 未经舍入的中间结果，而第二子运算由此未经舍入的中间结果产生一经舍入的最终结果。依此，在需要时，其他实施方式可将不同的状态位记录于快取机制内。

[0265] 本说明书关于舍入位与其他内部位的描述为利于说明本发明的精神，本发明当可用于其他类型的指标，包含舍入运算相关或运算控制相关的变数的不同编码表现形式。此外，针对说明书中将变数描述为具有“二进位数一"(也就是逻辑一)的许多段落，此描述包含具有“二进位数零”(也就是逻辑零)的变数的等效布林实施方式，并包含这些变数的其他表示方式。相同地，针对说明书中将变数描述为具有“二进位数零"(也就是逻辑零)的许多段落，此描述包含具有“二进位数一”(也就是逻辑一)的变数的等效布林实施方式，并包含这些变数的其他表示方式。此外，依据本说明书当可理解，在本文中，累加包含加法加总与加法差的情形。

[0266] 此外，依据本说明书当可理解，在本文中，指令包含架构指令与可由架构指令转译或转换产生的微指令。同样地，指令执行单元并非仅限于微处理器直接执行架构指令而未预先将其转译为微指令的情形。由于微指令亦属指令的一种类型，因此，指令执行单元也会

包含微处理器先将指令集架构指令转译或转换为微指令，而指令执行单元总是只执行微指令的情形。

[0267] 在本说明书中，尾数与有效位数的用语可互用。其他如原始结果与中间结果的用语，为区分在FMA运算的不同阶段所产生的结果与表示方式。此外，在本说明书中，储存格式中间结果的用语包含一中间结果向量(意指一数的量)与多个运算控制变数。这些用语不应被僵化或狭隘地解释，而应依据申请人所表现出来的目的探究其实质，而理解这些用语依据前后文的不同可能指涉不同的事件。

[0268] 图1、3-6中的功能方块可描述为模组、电路、子电路、逻辑、与其他数位逻辑与微处理器设计的技术领域中常用于指称由线路、电晶体与/或其他执 行一个或多个功能的物理结构所构成的数字逻辑用语。本发明亦可涵盖将说明书所描述功能以不同于说明书所描述的方式分派的其他替代实施方式。

[0269] 下列参考文献纳入本申请以说明FMA设计的相关概念与并使本申请发明能为读者理解。

[0270] 参考文献：

[0271] Hokenek,Montoye,Cook,"Second-Generation RISC Floating Point with MultiplyAdd Fused",IEEE Journal Of Solid-State Circuits,Vol 25,No 5,Oct1990.

[0272] Lang,Bruguera,"Floating-Point Multiply-Add-Fused with Reduced Latency",IEEE Trans On Computers,Vol 53,No 8,Aug 2004.

[0273] Bruguera,Lang,"Floating-Point Fused Multiply-Add:Reduced Latency for FloatingPoint Addition",Pub TBD-Exact Title Important.

[0274] Vangal,Hoskote,Borkar,Alvanpour,"A 6.2-GFlops Floating-Point MultiplyAccumulator With Conditional Normalization",IEEE Jour.Of Solid-State Circuits,Vol 41,No 10,Oct 2006.

[0275] Galal,Horowitz,"Energy-Efficient Floating-Point Unit Design",IEEE Trans On Computers Vol 60,No 7,July 2011.

[0276] Srinivasan,Bhudiya,Ramanarayanan,Babu,Jacob,Mathew,Krishnamurthy,Erraguntla,"Split-path Fused Floating Point Multiply Accumulate(FPMAC)",2013Symp on Computer Arithmetic(paper).

[0277] Srinivasan,Bhudiya,Ramanarayanan,Babu,Jacob,Mathew,Krishnamurthy,Erraguntla,"Split-path Fused Floating Point Multiply Accumulate(FPMAC)",2014Symp on Computer Arithmetic,Austin TX,(slides from www.arithsymposium.org).

[0278] Srinivasan,Bhudiya,Ramanarayanan,Babu,Jacob,Mathew,Krishnamurthy,Erraguntla,United States Patent 8,577,948(B2),Nov 5,2013.

[0279] Quach,Flynn,"Suggestions For Implementing A Fast IEEE Multiply-Add-Fused Instruction",(Stanford)Technical Report CSL-TR-91-483July,1991.

[0280] Seidel,"Multiple Path IEEE Floating-Point Fused Multiply-Add",IEEE2004.

[0281] Huang,Shen,Dai,Wang,"A New Architecture For Multiple-Precision

FloatingPoint Multiply-Add Fused Unit Design",Pub TBD,Nat'l University of Defense Tech,China(after)2006.

[0282] Paidimarri,Cevrero,Brisk,Ienne,"FPGA Implementation of a Single-Precision Floating-Point Multiply-Accumulator with Single-Cycle Accumulation",Pub TBD.

[0283] Henry,Elliott,Parks,"X87 Fused Multiply-Add Instruction",United States Patent 7,917,568(B2),Mar 29,2011.

[0284] Walaa Abd El Aziz Ibrahim,"Binary Floating Point Fused Multiply Add Unit",Thesis Submitted to Cairo University,Giza,Egypt,2012(retr from Google).

[0285] Quinell,"Floating-Point Fused Multiply-Add Architectures", Dissertation Presented to Univ Texas at Austin,May 2007,(retr from Google).

[0286] Author Unknown,"AMD Athlon Processor Floating Point Capability",AMD White Paper Aug 28,2000.

[0287] Cornea,Harrison,Tang,"Intel Itanium Floating-Point Architecture"Pub TBD.

[0288] Gerwig,Wetter,Schwarz,Haess,Krygowski,Fleischer,Kroener,"The IBM eServer z990 floating-point unit",IBM Jour Res&Dev Vol 48 No 3/4 May,July 2004.

[0289] Wait,"IBM PowerPC 440 FPU with complex-arithmetic extensions",IBM Jour Res&Dev Vol 49 No 2/3 March,May 2005.

[0290] Chatterjee,Bachega,et al,"Design and exploitation of a high-performance SIMD floating-point unit for Blue Gene/L",IBM Jour Res&Dev,Vol 49 No 2/3March,May 2005。

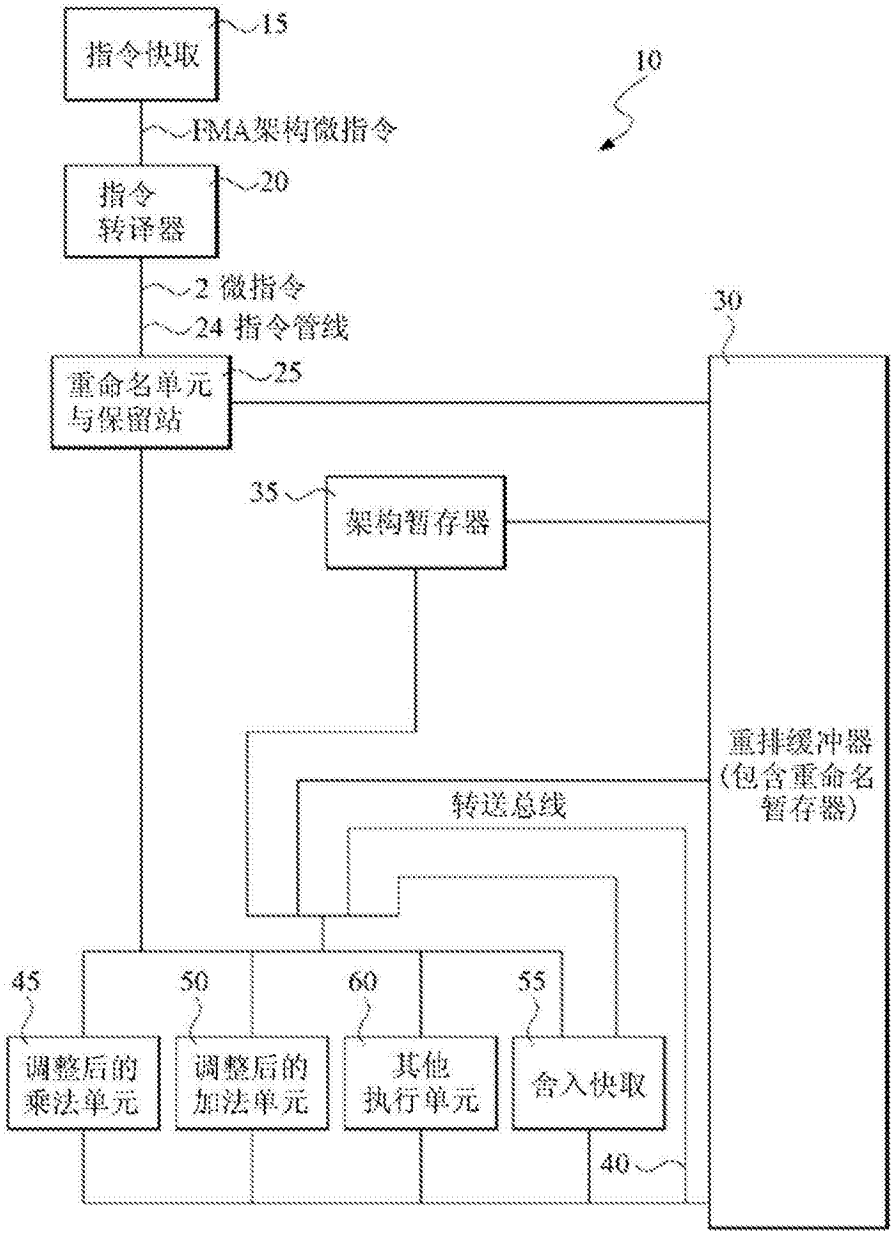

指令快取
15
10
FMA架构微指令
指令
转译器
20
2 微指令
24 指令管线
30
重命名单元
与保留站
25
35
架构暂存器
重排缓冲器
(包含重命名
暂存器)
转送总线
45
50
60
55
调整后的
乘法单元
调整后的
加法单元
其他
执行单元
舍入快取
40

图1

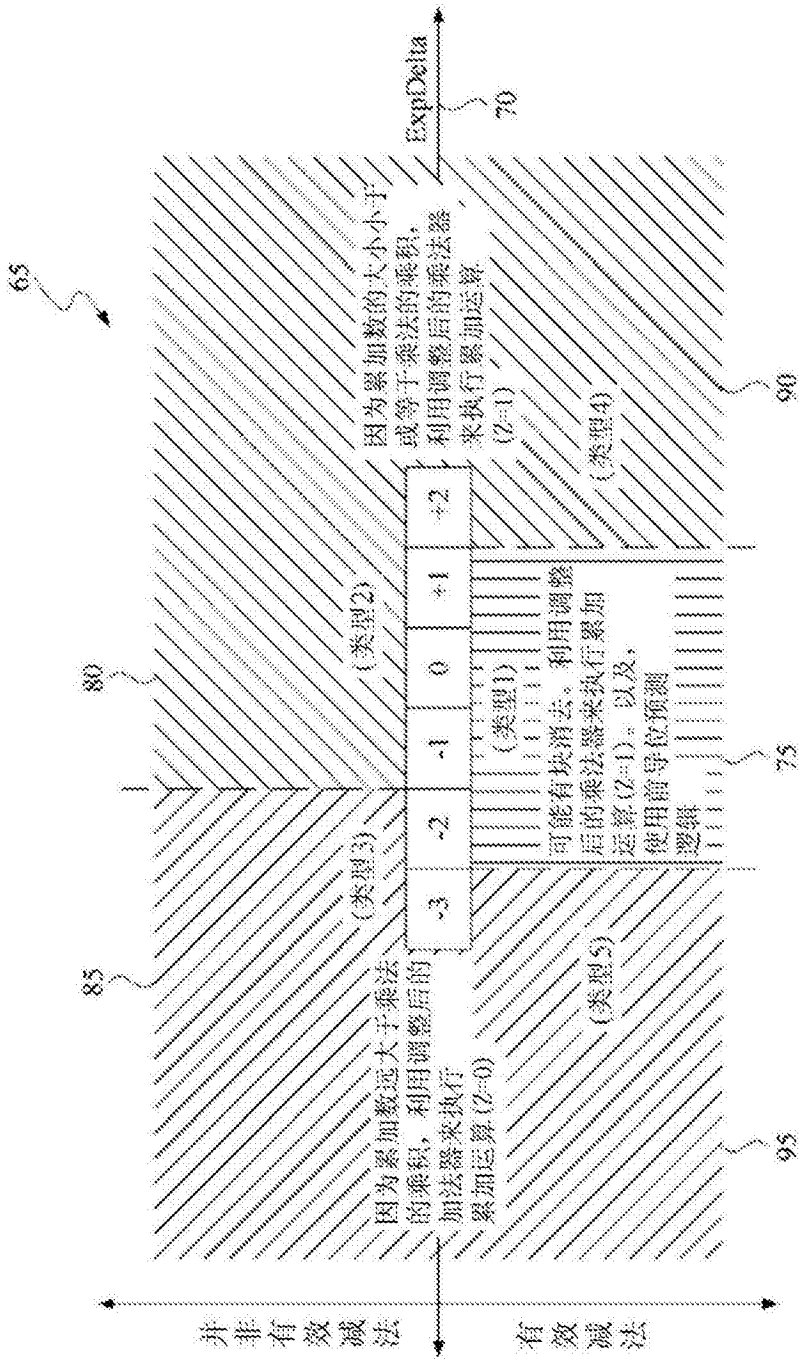

ExpDelta
70
65
90
80
85
75
95
-3
-2
-1
0
+1
+2
(类型1)
(类型2)
(类型3)
(类型4)
(类型5)
因为累加数的大小小于或等于乘法的乘积，利用调整后的乘法器来执行累加运算(Z=1)
因为累加数远大于乘法的乘积，利用调整后的加法器来执行累加运算(Z=0)
可能有块溢出，利用调整后的乘法器来执行累加运算(Z=1)，以及，使用前导位预测逻辑
非有效减法
有效减法

图2

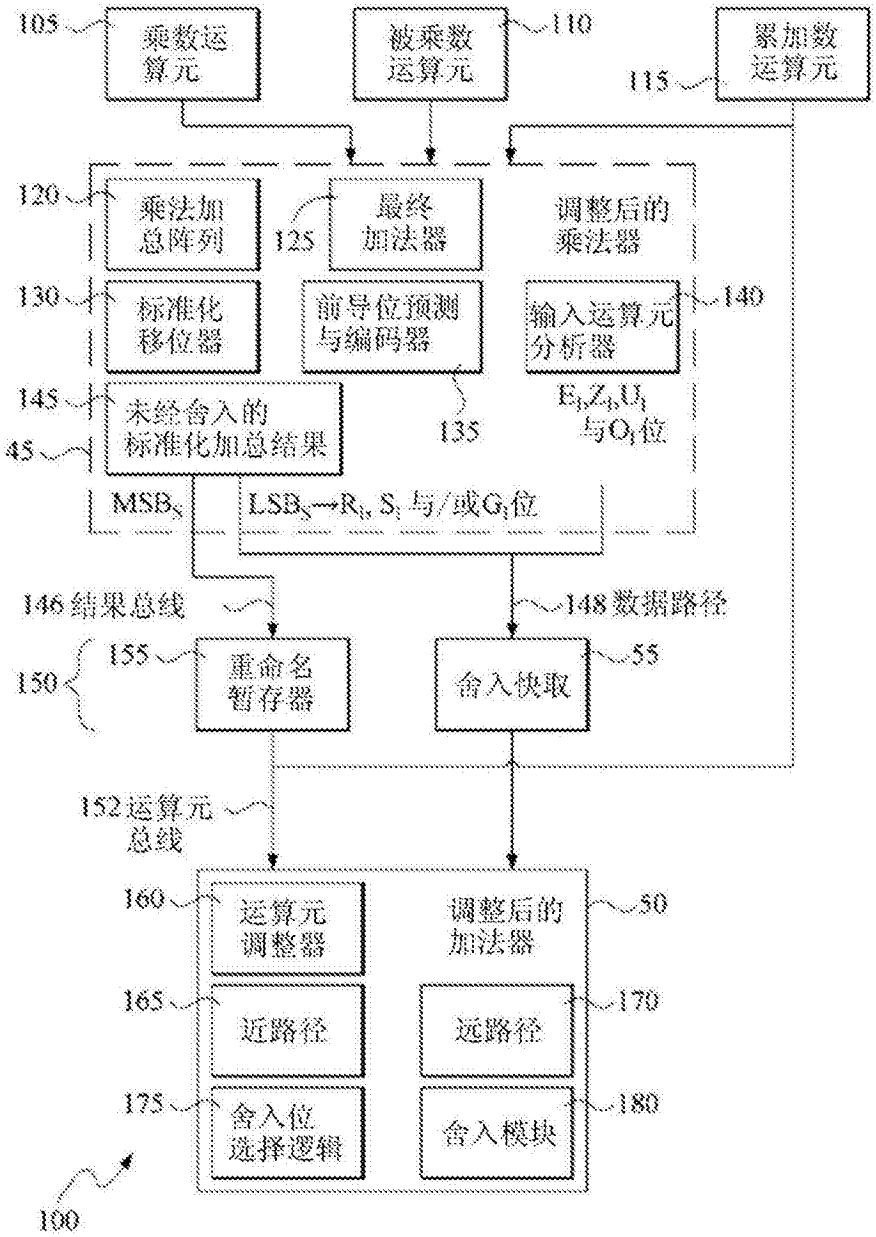

105
乘数运算元
被乘数运算元
110
累加数运算元
115
120
乘法加总阵列
125
最终加法器
调整后的乘法器
130
标准化移位器
前导位预测与编码器
输入运算元分析器
140
145
未经舍入的标准化加总结果
45
135
E_i, Z_i, U_i 与 O_i 位
MSB_S
$LSB_S \rightarrow R_i, S_i$ 与/或 G_i 位
146 结果总线
148 数据路径
150
155
重命名暂存器
舍入快取
55
152 运算元总线
160
运算元调整器
调整后的加法器
50
165
近路径
远路径
170
175
舍入位选择逻辑
舍入模块
180
100

图3

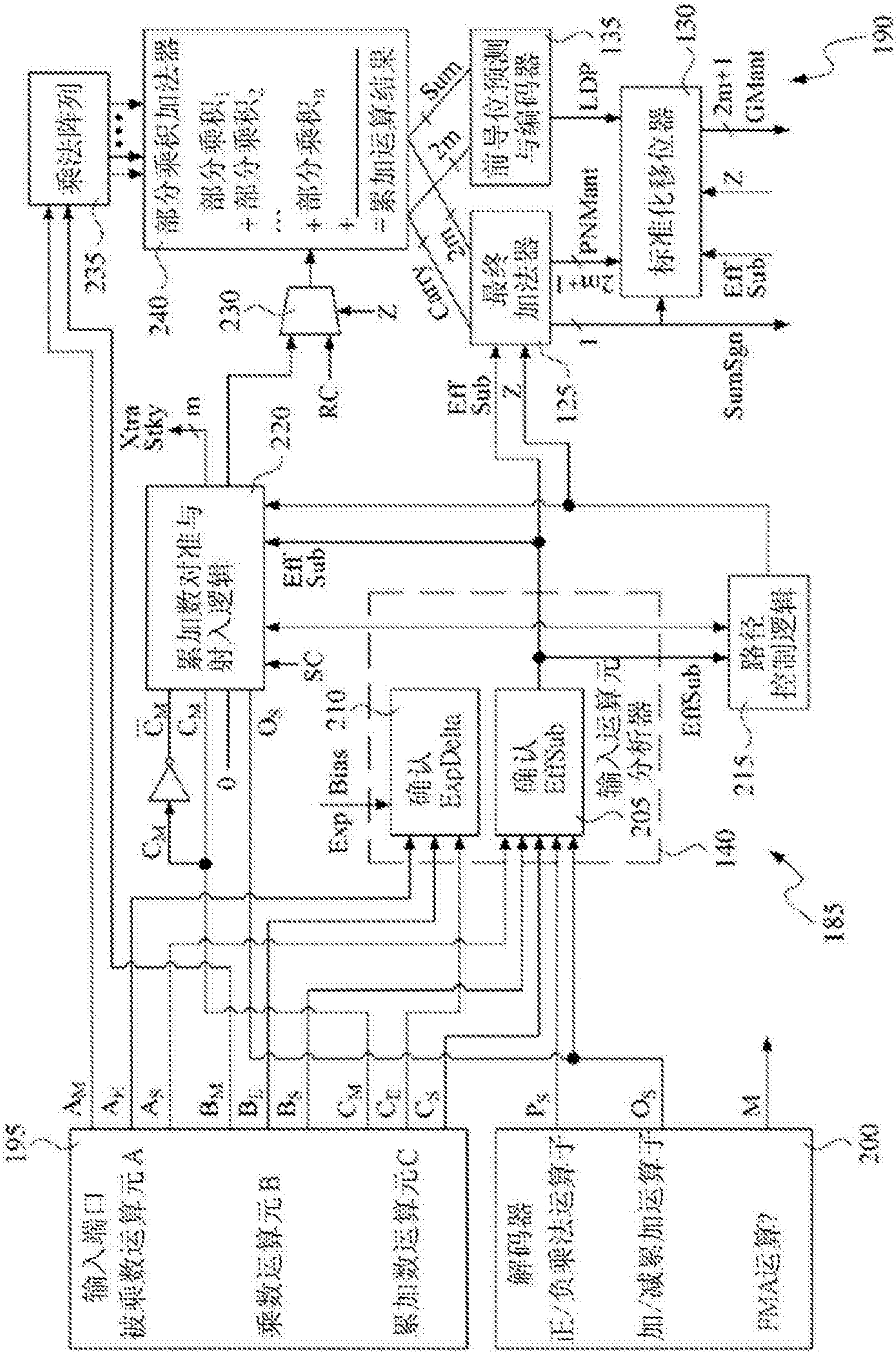

输入端口
被乘数运算元A
乘数运算元B
累加数运算元C
解码器
正/负乘法运算子
加/减累加运算子
FMA运算?
乘法阵列
部分乘积加法器
累加数对准与射入逻辑
确认ExpDelta
确认EffSub
输入运算元分析器
路径控制逻辑
最终加法器
前导位预测与编码器
标准化移位器
195
200
205
210
215
220
230
235
240
125
130
135
140
185
190

图4

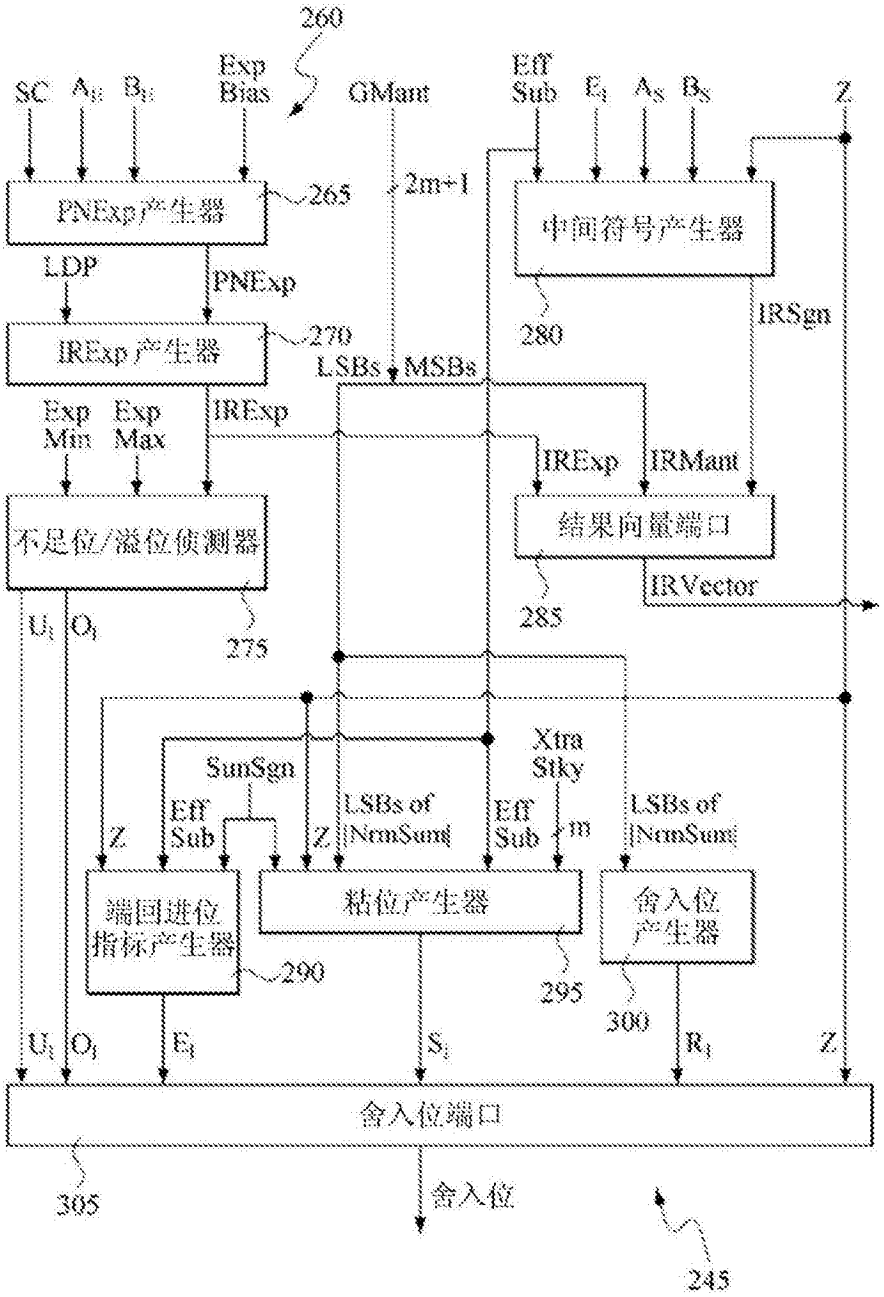

260
SC
A_S
B_S
Exp Bias
GMant
Eff Sub
E_S
A_S
B_S
Z
PNExp产生器
265
2m+1
中间符号产生器
LDP
PNExp
280
IRSgn
IRExp产生器
270
LSBs
MSBs
Exp Min
Exp Max
IRExp
IRExp
IRMant
不足位/溢位侦测器
结果向量端口
IRVector
285
U_I
O_I
275
Xtra Stky
SunSgn
Eff Sub
Z
LSBs of |NrmSum|
Z
Eff Sub
m
LSBs of |NrmSum|
端回进位指标产生器
290
粘位产生器
295
舍入位产生器
300
U_I
O_I
E_I
S_I
R_I
Z
舍入位端口
305
舍入位
245

图5

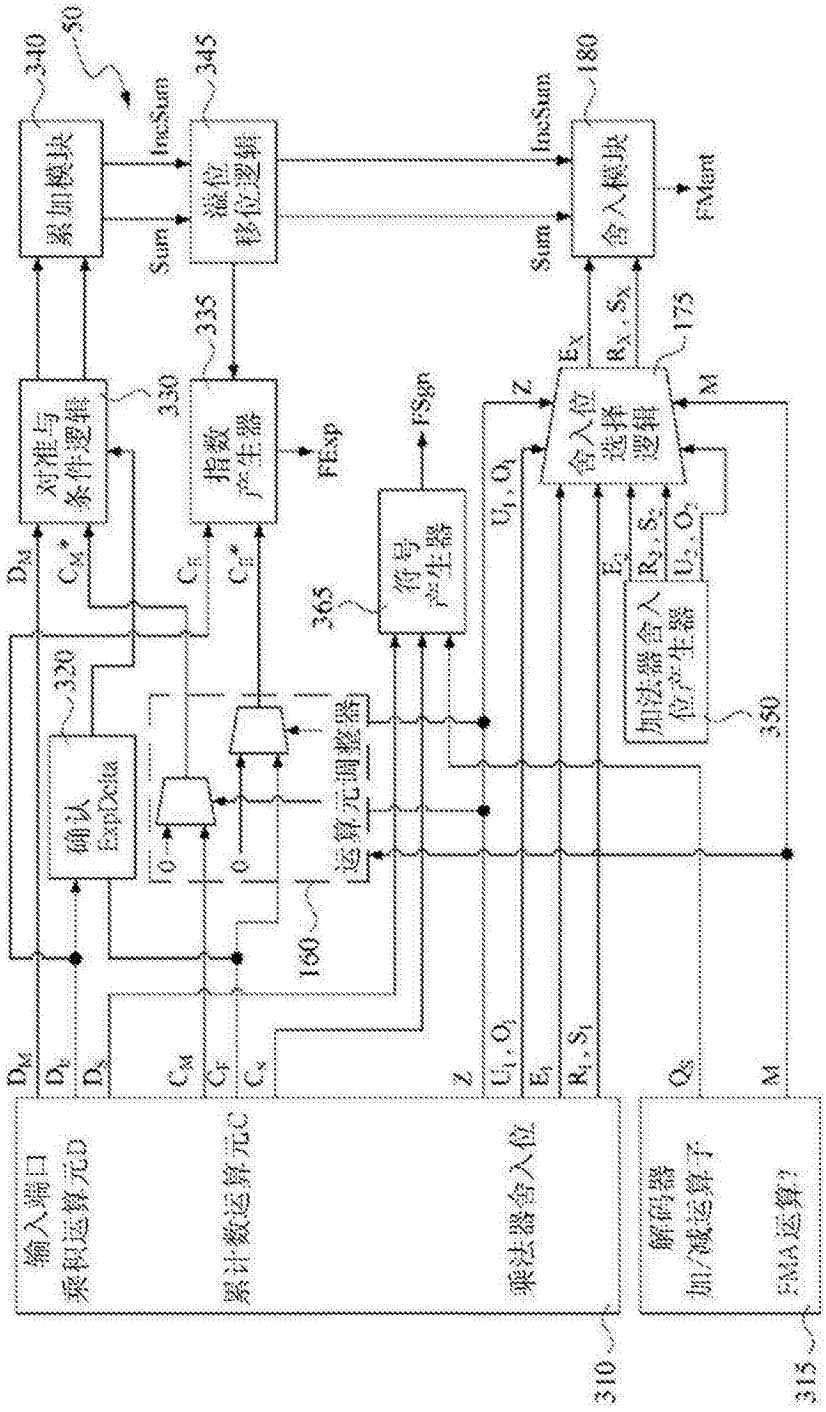

340
50
345
180
累加模块
InxSum
Sum
溢位
移位逻辑
舍入模块
FMant
335
330
175
对准与
条件逻辑
指数
产生器
FExp
FSgn
365
符号
产生器
舍入位
选择
逻辑
320
确认
ExpDelta
加法器舍入
位产生器
350
运算元调整器
160
输入端口
乘积运算元D
累计数运算元C
乘法器舍入位
解码器
加/减运算子
FMA运算?
310
315

图6

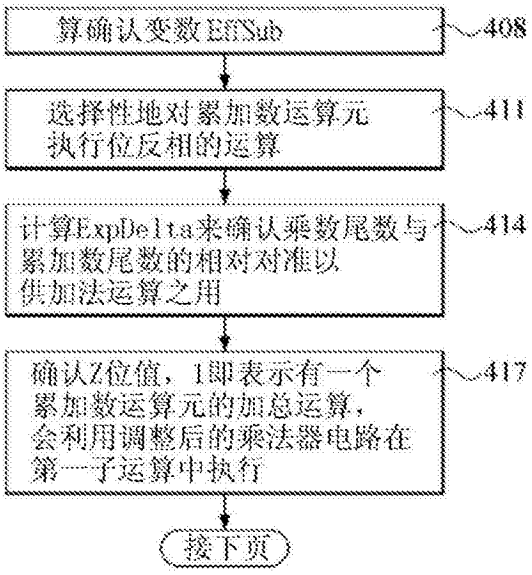

算确认变数EffSub
408
选择性地对累加数运算元执行位反相的运算
411
计算ExpDelta来确认乘数尾数与累加数尾数的相对对准以供加法运算之用
414
确认Z位值，1即表示有一个累加数运算元的加总运算，会利用调整后的乘法器电路在第一子运算中执行
417
接下页

图7

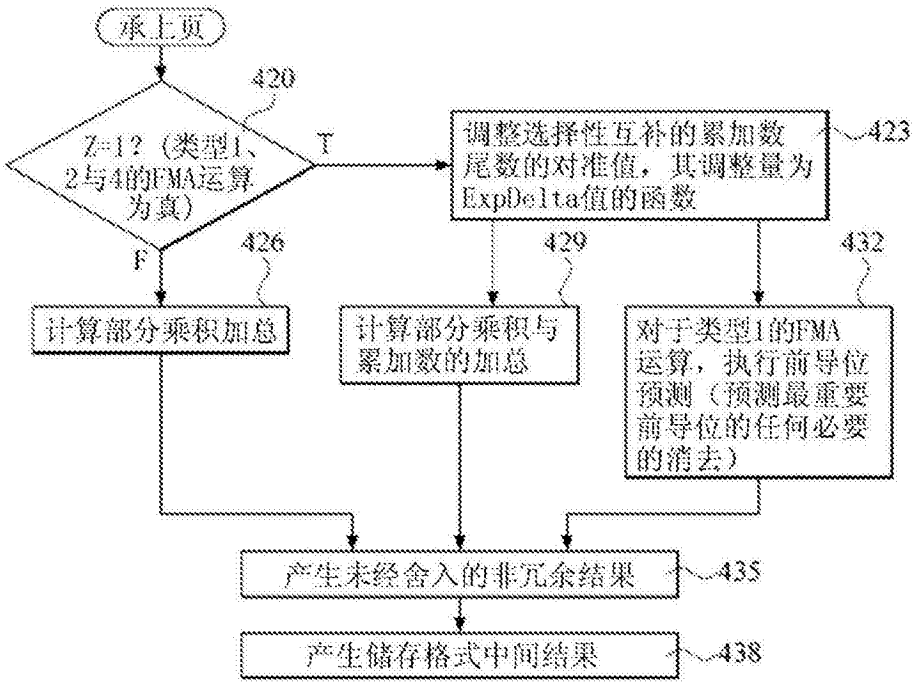

承上页
420
Z=1？（类型1、2与4的FMA运算为真）
T
F
调整选择性互补的累加数尾数的对准值，其调整量为ExpDelta值的函数
423
426
429
432
计算部分乘积加总
计算部分乘积与累加数的加总
对于类型1的FMA运算，执行前导位预测（预测最重要前导位的任何必要的消去）
产生未经舍入的非冗余结果
435
产生储存格式中间结果
438

图8

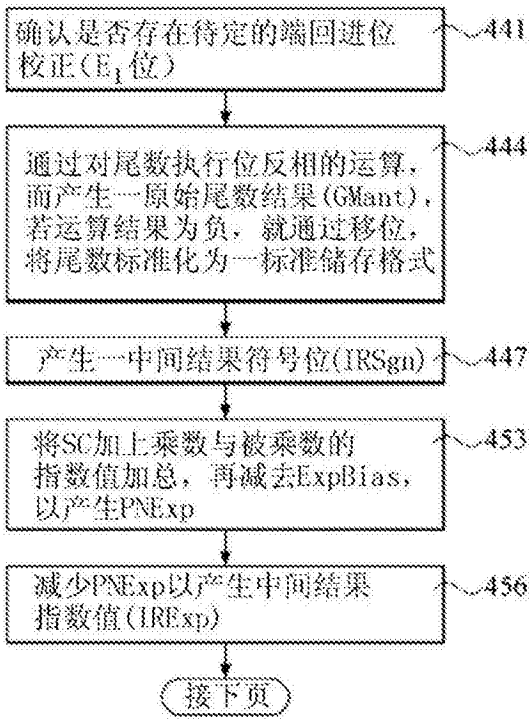

确认是否存在待定的端回进位校正(E_1位)
441
通过对尾数执行位反相的运算，而产生一原始尾数结果(GMant)，若运算结果为负，就通过移位，将尾数标准化为一标准储存格式
444
产生一中间结果符号位(IRSgn)
447
将SC加上乘数与被乘数的指数值加总，再减去ExpBias，以产生PNExp
453
减少PNExp以产生中间结果指数值(IRExp)
456
接下页

图9A

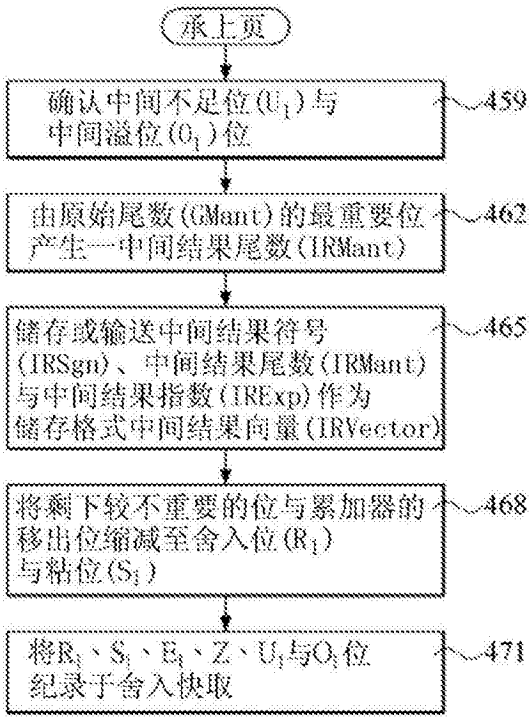

承上页
确认中间不足位(U_1)与中间溢位(O_1)位
459
由原始尾数(GMant)的最重要位产生一中间结果尾数(IRMant)
462
储存或输送中间结果符号(IRSgn)、中间结果尾数(IRMant)与中间结果指数(IRExp)作为储存格式中间结果向量(IRVector)
465
将剩下较不重要的位与累加器的移出位缩减至舍入位(R_1)与粘位(S_1)
468
将E_1、S_1、E_1、Z、U_1与O_1位纪录于舍入快取
471

图9B

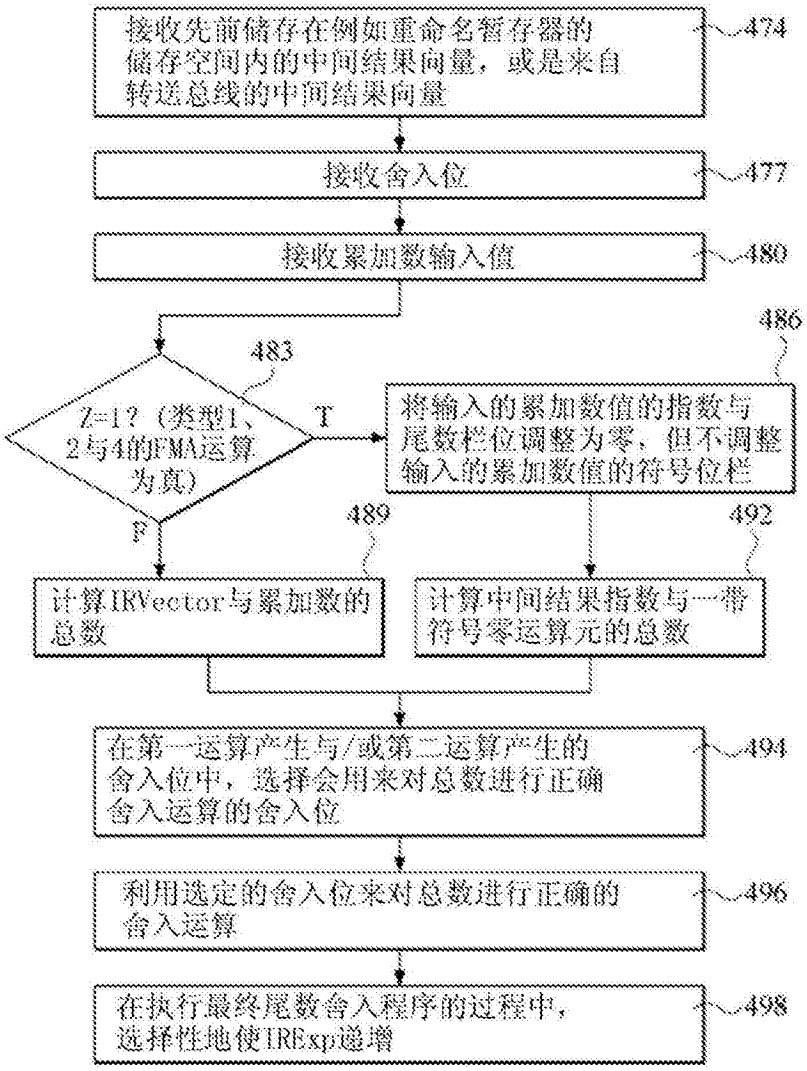

接收先前储存在例如重命名暂存器的储存空间内的中间结果向量，或是来自转送总线的中间结果向量
474
接收舍入位
477
接收累加数输入值
480
483
Z=1？（类型1、2与4的FMA运算为真）
T
F
486
将输入的累加数值的指数与尾数栏位调整为零，但不调整输入的累加数值的符号位栏
489
计算IRVector与累加数的总数
492
计算中间结果指数与一带符号零运算元的总数
在第一运算产生与/或第二运算产生的舍入位中，选择会用来对总数进行正确舍入运算的舍入位
494
利用选定的舍入位来对总数进行正确的舍入运算
496
在执行最终尾数舍入程序的过程中，选择性地使IRExp递增
498

图10

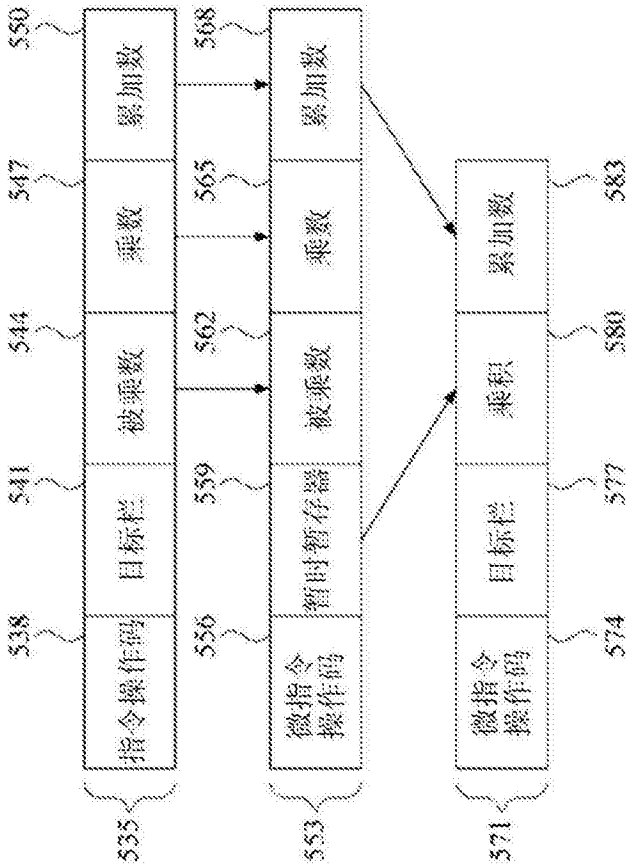

550
547
544
541
538
指令操作码
目标栏
被乘数
乘数
累加数
535
568
565
562
559
556
微指令操作码
暂时暂存器
被乘数
乘数
累加数
553
583
580
577
574
微指令操作码
目标栏
乘积
累加数
571

图11