發明專利說明書

（本說明書格式、順序，請勿任意更動）

【發明名稱】（中文/英文）

運算控制指標快取/ CALCULATION CONTROL INDICATOR CACHE

【技術領域】

【0001】 本申請案主張申請日爲 2014 年 7 月 2 日之美國專利第 62/020,246 號臨時申請案“Non-Atomic Split-Path Fused Multiply-Accumulate with Rounding cache”與申請日爲 2015 年 6 月 10 日之美國專利第 62/173,808 號臨時申請案“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”之國際優先權。該些優先權案之全文併入本案以供參考。

【0002】 本申請案亦關聯於下列與本申請案同時申請之申請案：VAS.3043，標題爲“Temporally Split Fused Multiply-Accumulate Operation”；VAS.3044，標題爲“Calculation Control Indicator Cache”；VAS.3045，標題爲“Calculation Control Indicator Cache”；VAS.3046，標題爲“Standard Format Intermediate Result”；VAS.3047，標題爲“Split-Path Heuristic for Performing a Fused FMA Operation”；VAS.3048，標題爲“Subdivision of a Fused Compound Arithmetic Operation”；與 VAS.2779，標題爲“Non-Atomic Split-Path Fused

Multiply-Accumulate”。這些申請案之全文併入本案以供參考。

【0003】 本發明係有關於一種執行算術運算之微處理器設計，尤其是融合浮點乘積-相加(FMA)運算之微處理器設計。

【先前技術】

【0004】 在現代電腦設計中，從大約 1990 年起，融合浮點乘積-相加(floating-point multiply-accumulate, FMA)運算就已經成爲一個受到商業矚目與學術關注的領域。融合 FMA 運算是一種算術運算，其形式爲±A*B±C，其中，A、B 與 C 是浮點輸入運算元(operand)（分別是一個被乘數(multiplicand)、一個乘數(multiplier)、與一個相加數(accumulator)），並且在 C 相加至 A 與 B 的乘積前不存在捨入(rounding)運算。±A*B±C 可包含，但不限於，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 設爲零); 與(f)A+C (即 B 設爲 1.0)。

【0005】 在大約 1990 年，此算術運算即以一原子(atomic)或不可分割(inseparable)運算之形式商業實現於 IBM 的精簡指令集(RISC)系統/6000。而後續設計進一步最佳化浮點乘積-相加運算。

【0006】 在其 2004 年的文獻“Floating-Point Multiply-Add-Fused with Reduced Latency”中，Tomas Lang 與 Javier D. Bruguera (“Lang et al.”)提出與最佳化 FMA 設計有關之許多重要課題。這些課題包括，指數差值與相加器移位/對準量之預計算，相加器與相乘陣列之平行對準，必要時使用 2 的補碼相加器(2’s complement accumulator)，和向量與進位向量之條件反轉，在最終相加/捨入模組前對於和向量與進位向量之標準化處理，LZA/LOA 與標準化移位之重疊運算，進位位元、捨入位元、

保護位元與黏(sticky)位元之分別運算，以及在合一之相加/捨入模組中具有 1m 寬度之雙總和加法器的使用（其中，m 是其中一個運算元的尾數(mantissa)寬度）。

【0007】 在其 2005 年的文獻 “Floating-Point Fused Multiply-Add: Reduced Latency for Floating-Point Addition” 中，Tomas Lang 與 Javier D. Bruguera (“Lang et al.”)提出利用分離數據路徑(或雙數據路徑)把對準方向從標準化的情況移開，其中，“近(close)”的數據路徑是用以從{2,1,0,-1}有效減去指數差值（此概念在本案詳細說明中會有更進一步的發展與改進），而“遠(far)”的數據路徑則是用來處理其他情況。Lang 等人還提出在遠的數據路徑使用雙對準平移器來處理相乘陣列之進位儲存輸出，以及在近的數據路徑使用非常有限對準平移的方法。

【0008】 在其 2004 年的文獻 “Multiple Path IEEE Floating-Point Fused Multiply-Add” 中，Peter-Michael Seidel (“Seidel”)提出對於 FMA 設計之不同改進方法，即使用多個平行運算路徑。Seidel 並提出關閉未使用之路徑閘門；從指數差值與有效運算確認多個運算路徑；使用兩個不同 運算路徑，其中一個提供給容易產生塊消去(mass cancellation)之小指數差值使用，另一個則是用來處理其他情況；對於具有效減法之小指數差值的情況，在有效乘積運算中插入相加器數值。

【0009】 在現今隨處可見個人行動運算裝置來提供擴充媒體分發與網路內容存取的情況下，更需要去設計一個之低製作成本、少功耗，但又能以高效能執行指令之 FMA 邏輯電路。

【0010】 執行FMA運算之主要方法涉及合一的乘積相加單元之使用，以執行整個 FMA 運算，包含對結果進行捨入。大部分

學術提案與商業應用都描述一個單塊的(monolithic)，或是不可切割的功能方塊，其具有將兩個數相乘，將未進行捨入之乘積與一第三運算元、加數或相加器進行相加操作、以及對運算結果進行捨入運算。

【0011】 另一種替代方案係使用傳統之乘法單元來執行 A*B 之子運算，然後使用傳統之加法單元來將 A 與 B 的乘積與 C 相加。不過，此傳統分工處理的方法會犧牲運算速度以及在同一個單元內將 C 與 A 和 B 之部分乘積-相加運算所能得到的效能增益。傳統分工處理的方法也涉及兩個捨入運算，因 A 與 B 的乘積會進行捨入運算，而後續 C 與乘積的加總也會進行捨入運算。因此，相較於合一處理的方法，傳統分工處理的方法有時候會產生不同且較不準確的結果。同樣地，因爲重複捨入運算的存在，此傳統分工處理的方法無法執行“融合(fused)” FMA 運算，而未能符合 IEEE 754 技術標準關於浮點運算的要求。

【0012】 因爲 FMA 硬體可同時用於多個運算目的並遵從 IEEE 754 技術標準，在現代的產品中，電腦設計者往往會選擇不可切割的 FMA 利用不可切割的 FMA 執行單元來完全取代傳統分離之乘積與加法功能方塊。然而，此方法會帶來許多不利影響。

【0013】 第一，相較於使用獨立的加法與乘法功能單元，FMA 硬體的設置成本較高，也較複雜。第二，在執行簡單的加法或乘法運算時，相較於使用獨立的加法與乘法功能單元，使用 FMA 硬體的延遲較長，且通常會消耗較多能量。第三，對於超純量(superscalar)電腦設計而言，將乘法與加法的功能合併於單一個功能方塊會減少算術指令可發送埠之數量，而削減電腦針對源碼、機械層級或軟體開發平行處理的能力。

【0014】 前述第三個不利影響可加入更多功能單元來處理，例如一個獨立的加法功能單元，但這會增加設置成本。基本上，一個額外的加法器(舉例)就是爲了在提供不可切割的 FMA 功能時，維持可接受之指令層級平行處理所需付出的代價，這又會增加整個硬體的設置尺寸，並會增加寄生電容與電阻。隨著半導體製造技術朝向小尺寸發展，寄生電容與電阻會對於算術運算之時間延遲，或稱延遲(latency)，有更顯著的影響。此時間延遲有時會被模擬爲因爲“長導線”而造成之延遲。因此，爲了減少實行不可切割的 FMA 運算對於指令層級平行處理影響所額外設置的功能方塊，在考量所需之晶片尺寸、功耗與算術運算之延遲後，所能提供的改善其實相當有限。

【0015】 因此，最佳之提案與實施通常會(但不總是)提供正確的運算結果(對應於 IEEE 之捨入與其他說明)，有時提供較高的指令產出(throughput)，但顯然需要額外的硬體電路而增加設置成本，並且會需要增加功耗來在複雜的 FMA 硬體上執行簡單的乘法與加法運算。

【0016】 現代 FMA 設計所要達成的目標仍未能完全達成。

【發明內容】

【0017】 本發明之一面向係提供一種微處理器，其包括有一指令執行單元，用以操作產生一中間結果向量及與產生的複數個運算控制指標共存；其中之運算控制指標係指明自中間結果向量產生一最終結果之接續運算該如何進行，且至少指標之一部份係自運算與/或中間結果向量產生期間推導而來。微處理器亦包括一設於指令執行單元外之儲存空間用以儲存中間結果向量及該些複

5

數個運算控制指標。

【0018】 依據本發明之一實施例，指令執行單元爲一算術處理單元，其設定爲具有三或更多數量之運算元輸入，且中間結果向量係自一複合算術運算之一第一算術運算之一運用到至少二運算元輸入所產生，其中之複數個運算控制指標係指明在使用一複合算術運算之一第二算術運算子之情形下、複合算術運算之第二算術運算該如何進行。

【0019】 依據本發明之一實施例，複合算術運算爲一接續算術運算。在一更特別之實施例中，第一與第二算術運算子爲基礎算術運算子，乃選自由加、減、乘、除所組成之群組中。在一更加特別之實施例中， 複合算術運算爲一乘積-相加運算，而第一算術運算是爲至少一被乘數運算元與一乘數運算元之相乘，且第二算術運算是爲一相加運算運算元與一被乘數與乘數運算元乘積之相加運算。

【0020】 依據本發明之一實施例，獨立於運算控制指標外之中間結果向量係以少於所需位元代表、以連貫產生複合算術運算之一算術上正確代表。在另一方面，與複數個運算控制指標結合之中間結果向量係提供足夠之資訊以產生複合算術運算之一算術上正確代表。其中，所謂複合算術運算之算術上正確代表的定義係與一代表無明顯區別、可由複合算術運算一非限定精確運算之結果所產生，藉以降低其重要性至一目標資料大小。

【0021】 例如，中間結果向量可爲一未捨入截斷值，乃由第一算術運算一結果之最顯著位元所組成，最不重要位元之運用將導致基本上可產生複合算術運算一正確捨入最終結果資訊之漏失。在一實施例中，最不顯著位元將會被壓縮至一或多個可提供

足夠資訊以自中間結果向量產生一算術上正確捨入結果之運算控制指標中(特別是捨入控制指標)。

【0022】 依據本發明之一實施例，儲存空間包括一通用儲存空間及一運算控制指標儲存空間；當運算控制指標儲存空間僅爲可操作以儲存與下載一運算控制指標之指令所接觸，此二儲存空間係可區別的，因通用儲存空間可爲微處理器一指令集之大部分指令所接觸以進行指令結果之儲存。

【0023】 更甚者，微處理器包括一結果匯流排及一與結果匯流排分離與區別之資料路徑；此結果匯流排將結果由指令執行單元輸送至通用儲存空間，而資料路徑係聯通指令執行單元與運算控制指標儲存空間、以使得自運算控制指標儲存空間進行運算控制指標之下載與儲存。

【0024】 依據本發明之一實施例，運算控制指標係提供關於有多少複合算術運算已完成產生中間結果向量之資訊。在本發明之另一實施例中，運算控制指標係提供關於第一算術運算是否導致不足位與溢位狀況之資訊。

【0025】 在本發明之另一面向中，係在提供執行一算術運算之微處理器中之一方法，此方法包括使用一指令執行單元去產生一中間結果向量與複數個運算控制指標、用以指明自中間結果向量產生一最終結果之接續運算該如何進行。此方法亦包括儲存中間結果向量與設置於指令執行單元外之複數個運算控制指標記憶體。

【0026】 依據本發明之一實施例，此方法更包括自記憶體中下載中間結果向量與複數個運算控制指標、並依據運算控制指標、在中間結果向量上、執行運算，以產生一最終結果。

【0027】 依據本發明之一實施例，算術運算爲一複合或接續算術運算。在本發明之另一實施例中，算術運算爲一相關至少一相乘運算與至少一相加運算之融合運算。在一更特別之實施例中，算術運算係爲一融合浮點乘積-相加運算，其運算元包括一被乘數、一乘數與一相加數，且其中之中間結果向量爲一至少被乘數與乘數之部分乘積之總和。

【0028】 依據本發明之一實施例，此方法更包括將一複合算術運算分路爲一使用一第一算術運算元之第一算術運算、及一使用一第二算術運算元之第二算術運算。運算控制指標可指示第二算術運算該如何進行、可提供關於多少複合算術運算已完成中間結果向量產生資訊、且/或提供關於第一算術運算是否導致不足位與溢位狀況之資訊。

【0029】 依據本發明之一實施例，中間結果向量具有較一原始結果(其可具有 2M 或更多位元)爲少之位元數(如 1M 位元)。因此，當考慮獨立於運算控制指標外時，中間結果向量係以比原需要位元爲少之位元代表、以連貫產生複合算術運算之一算術上正確之代表。然而，與複數個運算控制指標結合後，中間結果向量乃提供足夠之資訊、以產生複合算術運算之一算術上正確之代表。

【0030】 在本發明之另一面向中，係在提供一微處理器，乃包括複數個設定爲產生未捨入結果之指令執行單元、以及複數個用以捨入該未捨入結果之捨入指標。此微處理器更包括一捨入快取，其係可爲一關聯快取，其設定爲儲存複數個捨入指標之指令執行單元外。

【0031】 依據本發明之一實施例，微處理器更包括一與捨入快取不同之通用儲存空間、用以儲存由複數個指令執行單元產生

之未捨入結果。在一更特別之實施例中，此微處理器更包括一捨入位元轉換路徑、以及一與捨入位元轉換路徑不同之結果匯流排，其中之指令執行單元係設定爲輸出未捨入結果至結果匯流排、並輸出捨入位元轉換路徑上的捨入指標至捨入快取。

【0032】 依據本發明之一實施例，複數個指令執行單元中至少之一係設定爲產生一相對於一第一類型一指令之未捨入結果、以及一相對於一第二類型一指令之未捨入結果。在本發明之另一實施例中，微處理器係設定爲用以供應(a)一由一第一指令執行單元產生至一第二指令執行單元之未捨入結果、以及(b)複數個捨入指標中至少之一係自捨入快取傳送至第二指令執行單元。此第二指令執行單元係設定爲、在至少未捨入結果運算元上、執行一數學運算以利用所供應之複數個捨入指標中至少之一產生一最終捨入結果。

【0033】 在本發明之另一面向中，係在提供一微處理器，其係包括一可操作以產生一中間結果向量之第一指令執行單元、以及用以指明自中間結果向量產生一最終結果之接續運算該如何進行之複數個運算控制指標。此微處理器更包括一轉送匯流排，係設於指令執行單元外，並設定爲可用以將中間結果向量與複數個運算控制指標傳送至一第二指令執行單元者。依據本發明之一實施例，第一指令執行單元係設定爲相對一第一類型一指令而產生一未捨入結果、以及相對一第二類型一指令而產生一捨入結果者。

【0034】 在本發明之另一面向中，係提供一微處理器一用以執行一捨入運算之方法。一第一指令執行單元係先產生一未捨入結果；接著，將至少一捨入指標存入一設於第一指令執行單元外之捨入快取；一第二指令執行單元稍後讀取未捨入結果與來自捨

入快取之至少一捨入指標、並自該些輸入與可選用自一或多個其他運算元來產生一最終捨入結果。

【0035】 依據本發明之一實施例，此方法更包括將未捨入結果儲存至一與捨入快取區別之通用儲存空間中之步驟。在一更特別之實施例中，此方法更包括將一捨入指標自第一指令單元、透過一與一結果匯流排分離之資料路徑、轉換至捨入，此快取結果匯流排係將複數個指令執行單元耦接至一通用儲存空間。

【0036】 在本發明之另一面向中，係在提供一微處理器中之方法，用以執行一算術運算。一第一指令執行單元係先產生一中間結果向量、以及可指明自中間結果向量產生一最終結果之接續運算該如何進行之複數個運算控制指標；接著，將中間結果向量與複數個運算控制指標傳送至一第二指令執行單元；該第二指令執行單元接著依據運算控制指標產生最終結果與完成算術運算。

【0037】 依據本發明之一實施例，此算術運算爲一複合算術運算。在一更特別之實施例中，此複合算術運算屬一種融合類型，其僅容許單一之捨入運算、以產生最終結果。在一更加特別之實施例中，此算術運算爲一種融合乘積-相加運算，其中之中間結果向量係爲一此乘積-相加運算一部分之未捨入結果，且運算控制指標包括用以產生乘積-相加運算一最終捨入結果之捨入指標。

【0038】 依據本發明之一實施例，中間結果向量之傳送係藉由一結果匯流排完成，且運算控制字元之傳送則是藉由一與結果匯流排不同之資料路徑完成。

【0039】 此方法與裝置可縮減所需之電路、設置成本與複合算術運算之累增功耗。簡單來說，此裝置與方法係將複合算數運算分成至少二個子運算，由物理上與/或邏輯上各自獨立的硬體單

元來執行。每個硬體單元執行此複合算術運算之一部分。捨入運算或運算控制所需之位元係於此二運算間，儲存於一快取內。這些子運算係於不同時間與位置完成，而必要的資料片段會被整合以完成最終捨入運算。

【0040】 此方法與裝置具有許多值得注意的優點，尤其是針對 FMA 運算。

【0041】 第一，此方法與裝置可辨識 FMA 運算並將其區分爲至少二種類型，其中部分運算係以暫時或物理性分割之方式來執行。

【0042】 第二，此方法與裝置可將來自指令集架構之一不可切割的或一體的 FMA 指令轉譯或轉換爲至少二個子運算。

【0043】 第三，此方法與裝置容許這些子運算以非不可切割的、或暫時物理性分割之方式來執行，例如在非循序超純量電腦處理器裝置內執行。

【0044】 第四，FMA 運算（例如對應於部分第一類型之 FMA 運算或部分第二類型之 FMA 運算）中部分之必須算術運算係於執行第一特定微指令時執行。

【0045】 第五，此方法與裝置係以一新的方式預先計算 FMA 之符號資料。

【0046】 第六，此方法與裝置會儲存中間運算之部分結果，例如儲存於一結果（重命名）暫存器。

【0047】 第七，此方法與裝置會儲存運算結果之其他部分，例如儲存至稱爲捨入快取或運算控制指標快取之其他儲存單元。

【0048】 第八，此方法與裝置會將所收集到的資料，即中間結果，以新的標準化儲存格式來儲存。此外，此方法與裝置可能

會將儲存格式中間結果轉送至後續特殊類型之第二微指令，而非進行儲存。

【0049】 第九，此方法與裝置在需要提供所儲存之資料至後續第二微指令時，會存取捨入快取。

【0050】 第十，因應來自捨入快取之資料，此方法與裝置會選擇性地提供 FMA 加數至第二微指令或使輸入值歸零。

【0051】 第十一，此方法與裝置會在執行一第二(或更多）微指令之過程中，使用此儲存格式中間結果作爲輸入，來執行第一類型或第二類型之 FMA 運算中剩下必須執行之算術 FMA 運算。

【0052】 第十二，此方法與裝置係將經少量調整之先前技術乘法硬體執行單元與加法硬體執行單元結合，如結合所述之捨入快取與資料轉送網路用以迴避捨入快取。

【0053】 第十三，此方法與裝置不會讓分配埠無法於算術運算中使用、或是在電腦利用指令平行處理之能力與投資之硬體成本間妥協。

【0054】 本發明所採用的具體實施例，將藉由以下之實施例及圖式作進一步之說明。

【圖式簡單說明】

【0055】 第一圖係一微處理器之方塊示意圖，此微處理器具有執行單元與一捨入或運算控制指標快取，並使用二個子運算、一個調整後的乘法器、與一個調整後的加法器來執行 FMA 運算；

【0056】 第二圖係一示意圖，將一數字空間區分爲 FMA 運算之五個類型以爲例示(但非限定於此)；

【0057】 第三圖係一功能方塊圖，描述用以執行 FMA 運算之

調整後的乘法器與調整後的加法器內之邏輯元件；

【0058】 第四圖係一功能方塊圖，顯示本發明一實施例之乘法運算單元之路徑確認邏輯與尾數乘法模組，此乘法運算單元係經適當調整以接收 FMA 乘數、被乘數與相加數作爲輸入運算元；

【0059】 第五圖係一功能方塊圖，顯示第四圖之乘法運算單元之指數結果產生器與捨入指標產生器，此乘法運算單元係經適當調整以產生一儲存格式中間結果；

【0060】 第六圖係一功能方塊圖，顯示經適當調整以接收一儲存格式中間結果與相加數之加法運算單元之一實施例；

【0061】 第七圖係一功能方塊圖，顯示非不可分割之分路 FMA 運算之一實施例之第一 FMA 子運算中之一路徑確認部分；

【0062】 第八圖係一功能方塊圖，顯示非不可分割之分路 FMA 運算之第一 FMA 子運算中之一乘法與相加部分；

【0063】 第九 A 與九 B 圖係一功能方塊圖，顯示非不可分割之分路 FMA 運算之第一 FMA 子運算中之一儲存格式中間結果產生部分；

【0064】 第十圖係一功能方塊圖，顯示非不可分割分路 FMA 運算之第二 FMA 子運算；以及

【0065】 第十一圖係一示意圖，顯示將融合 FMA 指令轉譯爲第一與第二 FMA 微指令之一實施例。

【實施方式】

【0066】 **微處理器**

【0067】 第一圖係一微處理器之方塊示意圖。此微處理器 10 具有複數個執行單元 45,50,60 用以執行 FMA 運算。此微處理器

10 包含一指令快取 15，一指令轉譯器與/或微碼唯讀記憶體 20，一重命名單元與保留站 25，複數個執行單元（包含一調整後的乘法器 45、一調整後的加法器 50 與其他執行單元 60），一捨入快取 55（或是指運算控制指示器儲存空間），架構暫存器 35 與一重排緩衝器 30（包含重命名暫存器）。其他的功能單元（未圖示）可包含一微碼單元；分支預測器；一記憶體子系統，其包含一快取記憶體階層架構（例如階層一之資料快取、階層二之快取）、記憶體排列緩衝器、與記憶體管理單元；資料預擷取單元；與一匯流排介面單元等等。微處理器 10 具有一非循序執行微架構，可以不依照程式順序發布指令以供執行。進一步來說，架構指令（或微指令）轉譯或轉換出的微指令，可以不依照程式順序發布指令以供執行。微指令之程式順序係相同於轉譯或轉換出這些微指令之相對應架構指令之程式順序。微處理器 10 並具有一超純量微架構，其能在單一個時鐘週期內發布多個指令至執行單元執行。在一種實施例中，微處理器 10 可以說是以相容於 x86 指令集架構之方式提供指令以供執行。

【0068】 指令快取 15 儲存由系統記憶體擷取之架構指令。指令轉譯器與/或微碼唯讀記憶體 20 係將由系統記憶體擷取之架構指令轉譯或轉換爲微處理器 10 之微架構微指令集之微指令。執行單元 45, 50, 60 執行這些微指令。這些由架構指令轉譯或轉換出之微指令可實現架構指令。重命名單元 25 接收微指令並將重排緩衝器之項目依據程式順序分配給微指令、依據所分配之重排緩衝器項目索引更新這些微指令、將各個微指令發送至與將要執行這些微指令之執行單元相關聯的保留站 25、以及爲這些微指令執行暫存器重命名與關聯建立。

【0069】 分類運算

【0070】 在本發明之一實施例中，FMA 運算是依據輸入運算元之指數值的差(由變數 ExpDelta 表示)，以及是否涉及一有效之減法運算來進行分類。第二圖利用一包含有數字線 70 之數字空間 65 來顯示 ExpDelta 的值。在數字線 70 下方的空間顯示此運算會構成一有效減法運算。在數字線 70 上方的空間顯示此運算會構成一有效加法運算（亦即不存在有效減法運算）。

【0071】 指數差，表示爲 ExpDelta，係乘數與被乘數之輸入指數值的加總，減去任何指數偏移值，再減去加數或減數之輸入指數值。在相加值遠大於偏移調整乘積向量(bias-adjusted product vector)時，計算出的 ExpDelta 爲負。而在相加值遠小於偏移調整乘積向量時，計算出的 ExpDelat 爲正。

【0072】 有效減法，由變數 EffSub 表示，係指輸入運算元的符號與所欲執行的運算（例如乘法－加法，或是乘法－減法）之結合，將會有效降低浮點數結果的大小，而非增加。舉例來說，負的被乘數乘上正的乘數（乘積爲負）後，加上一個正的加數，就會有效降低結果的大小，而會表示爲有效減法(EffSub)。

【0073】 如第二圖之數字空間 65 之右側所示，在乘積向量的大小主導運算結果的情況下，相加數會直接用於初始捨入位元或是黏位元之運算。如本文後續將描述的，相加數與乘積尾數之相對對準有利於在計算影響捨入運算之位元前將此二者相加。在第二圖之數字空間 65 內，這些不存在“有效減法”的情況顯示爲“類型 2”運算 80，而存在“有效減法”的情況顯示爲“類型 4”運算 90。

【0074】 如第二圖之數字空間 65 之左側所示，在相加數的大

小主導運算結果，並且相加數之尾數大小係小於或等於所欲產生結果之尾數大小，相加數就不會用於初始捨入位元或是黏位元之運算。在第二圖之數字空間 65 內，這些不存在“有效減法”的情況顯示爲“類型 3”運算 85，而存在“有效減法”的情況顯示爲“類型 5”運算 95。因爲相加數有效對準乘積尾數之左側，所以可獲得在加上相加數之前，先確認黏位元與捨入位元的好處。

【0075】 區分出 ExpDelta 位於第二圖之數字線 70 之右側的情況與 ExpDelta 位於第二圖之數字線 70 之左側的情況會有許多優點。舉例來說，傳統 FMA 利用非常寬的對準移位器－相當於或大於輸入尾數寬度的三倍－來解釋相加數對準於乘數與被乘數之乘積的左側或右側之運算。透過將 FMA 運算區分爲由兩個調整後的執行單元（一個調整後的乘法器 45 與一個調整後的加法器 50）分別執行之兩個子運算，即可利用較小的資料路徑與較小的對準移位器來進行運算。

【0076】 對於數字線 70 之右側的運算，相加數的大小會小於中間乘積向量，此情況有利於在調整後的乘法器 45 內將相加數與乘法器之乘積相加。此運算只需要一個比傳統 FMA 之資料路徑的寬度還要小上大約一個尾數寬度的資料路徑。因爲調整後的乘法器 45 原本就具有一些內部延遲，相加數會有效地對準加總樹/陣列(summation tree/array)，也會簡化標準化與捨入運算。捨入運算將會由調整後的加法器 50 會在第二 FMA 子運算中執行。

【0077】 反之，對於數字線 70 之左側的運算，相加數會是較大的運算元而不會用於捨入運算。因爲相加數不會用於捨入運算（除了以下提到的特殊狀況），因而可以對乘法器的乘積執行一些初始黏收集(initial sticky collection)、將中間結果儲存至記憶體

（例如重排緩衝器與/或快取）、並且可以用調整後的加法器 50 來進行相加數之加法運算。傳統的捨入邏輯可有效處理相加數對捨入運算之選擇造成影響的特殊狀況，若是存在總數溢位，捨入位元會成爲黏位元之其中之一，而總數之最重要位元(least significant bit, LSB)會成爲捨入位元。

【0078】 某些種類的FMA運算－“有效減法”運算中對應於在第二圖之數字空間 65 下半部之子集合－會使一個或多個最重要位元歸零，傳統上係將此稱爲“塊消去(mass cancellation)”。在第二圖中，具有塊消去潛力之運算係表示爲“類型 1”運算 75。此情況需要在捨入運算前執行標準化運算來確認捨入點的位置。標準化向量所涉及之移位運算會產生顯著的時間延遲與/或呼叫前導位預測(leading digit prediction)以供使用。另一方面，不涉及塊消去之 FMA 運算可以省略前導位預測。

【0079】 總之，如第二圖所示，FMA 運算可依據 ExpDelta 與 EffSub 進行分類。第一類之 FMA 運算 75 係定義爲包含 ExpDelta 落於{-2, -1, 0, +1}之範圍且 EffSub 爲真之運算，這些運算包含那些具有位元塊消去潛力的運算。第二類之 FMA 運算 80 係定義爲包含 ExpDelta 大於或等於-1 且 EffSub 爲假之運算。第三類之 FMA 運算 85 係定義爲包含 ExpDelta 小於或等於-2 且 EffSub 爲假之運算。第四類之 FMA 運算 90 係定義爲包含 ExpDelta 大於{+1}且 EffSub 爲真之運算。第五類之 FMA 運算 95 係定義爲包含 ExpDelta 小於{-2}且 EffSub 爲真之運算。值得注意的是，這些分類之定義僅爲例示，FMA 運算當可採不同方式進行分類。舉例來說，在另一實施例中，第二類與第四類可以用同一類表示，相同地，第三類與第五類也可以同一類表示。此外，在其他實施例中，第二圖

之數字線 70 之左部分與右部分之分隔線亦可改變。

【0080】 **融合 FMA 指令執行元件組**

【0081】 第三圖係顯示一般用以執行 FMA 運算之融合 FMA 指令執行元件組 100 之一實施例。此元件組 100 包含兩個物理上與/或邏輯上分開之算術邏輯單元，（在一實施例中，即爲一調整後的乘法器 45 與一調整後的加法器 50）與共享儲存空間 155 與 55 以儲存複數個未經捨入運算之中間結果向量與捨入指標。

【0082】 各個調整後的乘法器 45 與調整後的加法器 50 都是一個指令執行單元，進一步來說，是指令管線 24 內之一個算術處理單元，用以對機器指令（例如複雜指令集(CISC)微架構內一個指定的指令集或是精簡指令集(RISC)微架構內一個指定的指令集）進行解碼，以及從一組共享的高速記憶體讀取運算元並寫入結果。指令執行單元可被理解爲一個邏輯電路的特性集合，用以執行傳送過來之指定機器指令集，而不同於可平行執行（而不僅止於管線化執行）多個機器指令之較大的電路群(如果存在的話)。

【0083】 更特別的是，調整後的乘法器 45 與調整後的加法器 50 係互相分離、不可分割的、能獨立運行的執行單元，能獨立對微碼進行解碼並執行，並提供控制信號至內部的資料路徑。共享高速記憶體可以是一個暫存器檔案或是一組非架構運算暫存器，供微指令交換資料並使其運算結果可爲其他執行單元看見。

【0084】 更特別的是，調整後的乘法器 45 是一個適當的乘法運算單元，在大部分情況下，就像傳統的乘法運算單元，能執行不屬於 FMA 運算之一般乘法微指令。不過，此調整後的乘法器係經適當調整，使能接收 FMA 乘數 105、被乘數 110、與相加數 115 作爲其輸入運算元，並產生一儲存格式中間結果 150，這在後

續段落會有更進一步的描述。相類似地，調整後的加法器 50 是一個適當的加法運算單元，在大部分情況下，就像傳統的加法運算單元，能執行不屬於 FMA 運算之一般相加微指令，例如加或減。不過，此調整後的加法器係經適當調整，使能接收儲存格式中間結果 150 並產生一個正確且經捨入運算之 FMA 結果。

【0085】 調整後的乘法器 45 能夠執行第一階層或部分的融合 FMA 運算（FMA1 子運算）。此調整後的乘法器 45 包含一輸入運算元分析器 140、一乘法加總陣列 120、一最終加法器 125、一標準化移位器、與一前導位元預測與編碼器 135。在執行 FMA1 子運算時，調整後的乘法器 45 會產生並輸出一未經捨入運算之標準化加總結果 145 與複數個捨入位元（或是捨入指標）。相較之下，在執行非融合 FMA 運算時，調整後的乘法器 45 會產生一經捨入運算且相容於 IEEE 標準的結果。

【0086】 捨入位元與未經捨入運算之標準化加總結果 145 之最重要位元係依據一儲存格式進行儲存。在一實施例中，未經捨入運算之標準化加總結果 145 的最重要位元係輸出至一結果匯流排 146，以儲存至一尾數寬度等於目標資料格式之尾數寬度的重命名暫存器 155。捨入位元輸出至一專用捨入位元或運算控制指標資料路徑、或是位於調整後的乘法器外且不同於結果匯流排 146 之連接網路 148，以儲存至一不同於儲存重命名暫存器 155 之儲存單元（例如重排緩衝器 30）的捨入快取 55。這些未經捨入運算之標準化加總結果 145 之最重要位元，連同捨入位元，係包含一儲存格式中間結果 150。

【0087】 因爲重命名暫存器 155 與捨入快取 55 屬於可爲其他執行單元看見之共享記憶體的一部份，物理上與/或邏輯上獨立於

調整後的乘法器 45 之調整後的加法器 50，可以透過運算元匯流排 152 與捨入位元資料路徑 148 接收此儲存格式中間結果 150，並執行一第二（完成）階層或部分的融合 FMA 運算（FMA2 子運算）。此外，在 FMA1 子運算與 FMA2 子運算間亦可執行其他不相關的運算。

【0088】 調整後的加法器 50 提供一運算元乘數 160，使 FMA 狀態下之相加運算元歸零，而在 FMA 狀態下，調整後的乘法器 45 已完成必要的相加運算。調整後的乘法器 50 並具有捨入位元選擇邏輯 175，從調整後的乘法器 45 產生之捨入位元、調整後的加法器 50 內部產生之捨入位元，或是二者的組合，選擇用於捨入模組 180 以產生最終捨入結果之捨入位元。調整後的加法器並具有一近路徑加總電路 165，用以在存在兩個加總運算元之塊消去的情況下對總數進行標準化運算，並具有一遠路徑加總電路 170，用以執行最多只需單一位元移位之加總運算以產生總數。如下所述，FMA2 子運算可完全由遠路徑加總電路 170 進行處理。

【0089】 **調整後的乘法器**

【0090】 第四與五圖係詳細顯示調整後的乘法器 45 之一實施例。第四圖特別顯示調整後的乘法器 45 之路徑確認邏輯 185 與尾數乘法模組 190。第五圖特別顯示調整後的乘法器 45 之指數結果產生器 260 與捨入指標產生器 245。

【0091】 如第四圖所示，路徑確認邏輯 185 包含一輸入解碼器 200、一輸入運算元分析器 140、路徑控制邏輯 215 與一相加數對準與射入邏輯電路 220。尾數乘法模組 190 包含第三圖之乘法加總陣列 120，這在第四圖中係以二個元件顯示，即一乘法陣列 235 與一部分乘積加法器 240。尾數乘法模組 190 並包含一最終加

法器 125、一前導位元預測器與編碼器 135、與標準化移位器 130。

【0092】 如第五圖所示，指數結果產生器 260 包含一 PNExp 產生器 265、一 IRExp 產生器 270、與一不足位/溢位偵測器 275。捨入指標產生器 245 包含一中間符號產生器 280、一結果向量埠 285、一端迴進位指標 290、一黏位元產生器 295 與一捨入位元產生器 300。

【0093】 請參照第四圖，調整後的乘法器 45 透過一個或多個輸入埠 195 接收一輸入微指令以及運算元數值。就 FMA 微指令而言，調整後的乘法器 45 接收一被乘數運算元 A、一乘數運算元 B 與一相加數運算元 C，各個運算元都包含一符號指標或位元、一尾數與一指數。在第四圖與第六圖中，浮點運算元之符號、尾數與指數部分分別以下標 S、M 與 E 表示。舉例來說，A_S、A_M、A_E 分別代表被乘數之符號位元、被乘數之尾數與被乘數之指數。

【0094】 解碼器 200 對輸入微指令進行解碼以產生 FMA 指標 M 與二進位數運算符號指標（或位元）P_S 與 O_S，M 意指接獲 FMA 微指令。在一實施例中，形式爲 A*B+C 之 FMA 微指令會產生二進位數零之一正乘法/向量負乘法符號運算子 P_S 與一加/減運算子 O_S。形式爲-A*B+C 之負乘積-相加微指令會產生一個二進位數一之運算子 P_S 與一個二進位數零之運算子 O_S。形式爲 A*B-C 之乘積-相減微指令會產生一個二進位數零之運算子 P_S 與一個二進位數一之運算子 O_S。形式爲-A*B-C 之向量負乘積-相減微指令會產生二進位數一之運算子 P_S 與運算子 O_S。在其他較爲簡化之實施例中，調整後的乘法器 45 並不直接支援向量負微指令與/或減法微指令，但由微處理器 10 來支援等效的運算，也就是在調用乘積-相加/相減微指令至調整後的乘法器 45 前，視情況額外反轉一個

或多個運算元或符號指標。

【0095】 乘法陣列 235 接收被乘數與乘數之尾數值 A_M 與 B_M 並計算出 A_M 與 B_M 之部分乘積。可以理解的是，當 A_M 或者 B_M 的絕對值爲一或零，乘法陣列 235 所產生之單一個“部分乘積”的值就會是 A_M 與 B_M 之完整乘積。部分乘積係提供至部分乘積加法器 240，其提供複數個項目以接收 A 與 B 之部分乘積以等待將其加總，而至少一個之部分乘積加法器 240 之項目用以接收一相加來源值 C_X。如下所述，在討論完輸入運算元分析器 140 與相加數對準射入邏輯 220 後，會對部分乘積加法器 240 有其他的說明。

【0096】 輸入運算元分析器 140 包含一 ExpDelta 分析子電路 210 與一 EffSub 分析子電路 205。ExpDelta 分析子電路 210 產生 ExpDelta(Exp Δ)值。在一實施例中，ExpDelta 透過將乘數與被乘數之輸入指數值 A_E 與 B_E 加總，減去加數或減數輸入指數值 C_E，並在存在指數偏移值 ExpBias 時減去此指數偏移值(如果任一者存在)。在 A_E、B_E 與 C_E 是以偏移指數(biased exponent)表示時，例如 IEEE754 之規範，被乘數 A 與乘數 B 之乘積的偏移量將會是相加數 C 偏移量的兩倍。而導入 ExpBias 值可對此進行校正。

【0097】 EffSub 分析子電路 205 分析運算元符號指標 A_S、B_S 與 C_S 以及運算元符號指標 P_S 與 O_S。EffSub 分析子電路 205 產生一“EffSub”值，用以表示 FMA 運算是否爲一有效減法運算。舉例來說，若是對 A 與 B 之乘積與 C 執行運算元特定之加法或減法運算（或是在負向量乘法運算子的情況下，此運算結果的負值）所產生之結果 R 的絕對值小於(a) A 與 B 之乘積的絕對值或是(b) C 的絕對值，就會是有效減法。若以數學符號表示，若是($|R|<|A*B|$) $\vee$ ($|R|<|C|$)，其中 R 爲 FMA 運算之結果，此 FMA 運算就會構成

有效減法。雖然以 FMA 運算之結果可以簡化 EffSub 的描述，不過需理解的是，在 EffSub 分析子電路 205 預先確認 EffSub 時，實際上是透過分析符號指標 A_S、B_S、C_S、P_S 與 O_S 來進行，而不去評估 A、B 與 C 之尾數、指數與大小。

【0098】 路徑控制邏輯 215 接收由輸入運算元分析器 140 產生之 ExpDelta 與 EffSub 指標，並藉以產生一路徑控制信號，其數值係以變數 Z 表示。路徑控制邏輯 215 可控制 C 的相加運算是否會連同 A 與 B 之部分乘積一併在調整後的乘法器 45 內執行。在一實施例中，路徑控制邏輯 215 用以產生 Z 之設定條件係顯示於第二圖中。在一實施例中，對於任何調整後的乘法器 45 被選用來執行乘積-相加運算之相加部分運算之情況（例如類型 1, 2 與 4），Z 會是二進位數一，而對於任何 ExpDela 與 EffSub 之其他組合（例如類型 3 與 5），Z 會是二進位數零。

【0099】 另外，路徑控制邏輯 215 也可以透過判斷 C 所具有一大小，相較於 A 與 B 之乘積大小，是否可使 C 對準於加總樹(summation tree)內，而不需將 C 之最重要位元移位至 A 與 B 之部分乘積加總之加總樹所提供之最重要位元的左側來產生 Z。另一個或替代之設定條件可以是，在執行 FMA 運算時是否具有執行塊消去之潛力。再另一個或替代之設定條件可以是，對 A 與 B 之乘積進行 C 之相加運算所產生之捨入前結果 R 所需要的位元數，是否少於將 C 對準 A 與 B 之乘積所需要的位元數。由此可知，路徑控制之條件可因應調整後的乘法器 45 之設計進行改變。

【0100】 相加數對準射入邏輯 220 接收路徑控制邏輯 215 所產生之 Z、ExpDelta 分析子電路 210 所產生之 ExpDelta、一移位常數 SC、與相加數尾數值 C_M。在一實施例中，此相加數對準射

入邏輯 220 亦接收 C_M 之位元反相(bitwise negation)，即 $\overline{C_M}$，與加法/減法相加運算子指標 O_S。在另一實施例中，相加數對準射入邏輯 220 係在加法/減法相加數指標 O_S 顯示調整後的乘法器 45 所接收之微指令爲乘積-相減微指令時，選擇性地額外反轉 C_M 之值。

【0101】 相應於所接收之輸入，相加數對準射入邏輯 220 會產生一數值 C_X 射入部分乘積加法器 240。此射入陣列之 C_X 的寬度是 2m+1，或可理解爲輸入運算元之尾數 A_M、B_M 與 C_M 的兩倍寬度額外加上一個位元。

【0102】 若是 M 爲二進位數零以顯示調整後的乘法器 45 正在執行一般的乘法運算而非 FMA1 子運算，多工器 230 就會將一捨入常數 RC，而非 C_X，射入部分乘積加法器 240，藉此，調整後的乘法器 45 即可以傳統方式產生一捨入後結果。RC 的數值部分是由指令顯示之捨入運算的類型所決定（例如：捨入向上(round half up)、捨入相等(round half to even)、捨入遠離零(round half away from zero)），部分是由輸入運算元之位元尺寸（例如 32 位元與 64 位元）所決定。在一實施例中，部分乘積加法器 240 係利用兩個不同的捨入運算常數來計算出兩個總數，並選擇其中較適當的一個。藉此，調整後的乘法器 45 之 IMant 輸出就會是一般乘法運算之正確捨入後之尾數結果。

【0103】 若是 M 爲二進位數一而 Z 爲二進位數零，即表示不需對 A 與 B 之乘積執行 C 的相加運算，在此情況下，就一實施例而言，此相加數對準射入邏輯 220 會將 C_X 設定爲零，而使多工器 230 將零值射入用以接收 C_X 值之部分乘積加法器 240 陣列。若是 M 爲二進位數一且 Z 爲二進位數一，相加數對準射入邏輯 220 就會將 C_X 右移相等於 ExpDelta 加上一移位常數 SC 的量，以產生

C_X。在一實施例中，移位常數 SC 等於 2，以對應於第二圖，當 C 的相加運算係執行於調整後的乘法器內，圖中數字空間內最大的 ExpDelta 負值。隨後，多工器 230 會將運算結果 C_X 射入部分乘積加法器 240。

【0104】 相加數對準射入邏輯 220 內並結合有一黏收集器 (sticky collector)。相加數 C_X 中任何移位超過部分乘積加法器 240 加總樹之最不重要位元之部分，係保留於 XtraStky 位元供捨入運算之用。因爲會有多達 m 個位元移位超過部分乘積加法器 240 加總樹之最不重要位元，XtraStky 位元係作爲一寬度 m 之額外黏位元陣列，用於黏位元 S 之運算。

【0105】 回到調整後的乘法器 45 之加法邏輯，在部分實施方式中，部分乘積加法器 240 係爲一加總樹，而在一實施方式中，係爲一個或多個進位儲存加法器。此部分乘積加法器 240 係根據所提供之部分乘積加總樹之位元欄的進位儲存向量，對一未經捨入之冗餘代表或總數，依據先前技術之乘法執行單元通常會執行之運算方法，來進行相加運算，這包含在部分乘積之相加運算中，對相加數輸入值選擇性進行額外的位元反相與對準運算。

【0106】 同樣地，部分乘積加法器 240 所執行的算術運算會受到 Z 之數值的影響。若是 Z=1，部分乘積加法器 240 就會對於 A_M 與 B_M 的乘積執行 C_X 的連帶相加運算。若是 Z=0，部分乘積加法器 240 就會對 A_M 與 B_M 的乘積執行一基本相加運算。部分乘積加法器 240 會產生一由一 2m 位元總數向量與一 2m 位元進位向量表示之冗餘二進位數總數，作爲連帶相加運算或是基本相加運算之運算結果。

【0107】 這些進位與總數向量同時轉送至一最終加法器 125

與一前導位預測器與編碼器 135。此最終加法器 125 可爲一進位預看加法器(carry-lookahead adder)或一進位傳播加法器(carry propagate adder)，透過將進位與總數向量轉換成寬度爲 2m+1 之正或負的預標準化未捨入非冗餘總數(prenormalized unrounded nonredundant sum) PNMant 以完成相加運算程序。最終加法器 125 並產生一總數符號位元 SumSign，來顯示 PNMant 爲正數或負數。

【0108】 在最終加法器 125 產生 PNMant 之同個時間週期，前導位預測器與編碼器 135 會同步預測需要被消除以標準化 PNMant 之前導位的數量。相較於傳統對乘積-相加運算分工處理之 FMA 設計中，因其最終加法器 125 執行之最終加法運算係於標準化運算後執行而需同時對進位向量與總數向量執行標準化運算，因而必須等待前導位預測之輸出，故本發明之處理方式相較傳統作法實具有優點。在一實施例中，此前導位預測器與編碼器 135 不是可以適用於正總數的情況，就是可適用於負總數的情況。

【0109】 在一實施例中，前導位預測只在類型 1 之運算執行。如前所述，所選擇之前導位預測的方法不是可以適用於正總數就是可以適用於負總數，爲熟習浮點運算設計領域者所熟知。

【0110】 因爲前導位預測器與編碼器 135 具有最多一個位元的誤差，任何可用於校正此誤差之常用技術均可使用，這些技術可設置於標準化移位器 130 內，或是關聯於標準化移位器 130。一個解決方案係提供邏輯來預測此誤差。另一個解決方案係透過確認 PNMant 之 MSB 是否已經設定，並相應地選擇 PNMant 之一額外移位。

【0111】 標準化移位器 130 從最終加法器 125 接收此未捨入非冗餘總數 PNMant 並產生一原始尾數值 GMant。在 C_X 之相加運

算是由部分乘積加法器 240 執行的情況下，GMant 會是 A_M 和 B_M 乘積與 C_X 之標準化加總的絕對值。而在其他情況下，GMant 會是 A_M 和 B_M 乘積之標準化加總的絕對值。

【0112】 爲了產生 GMant，標準化移位器 130 在 SumSgn 顯示 PNMant 爲負時對 PNMant 執行位元反相之運算。對負的 PNMant 數值執行標準化移位器 130 的位元反相運算，可產生如下所述之儲存格式中間結果 150，並有助於正確的捨入運算。在調整後的乘法器內反轉 PNWant，即可產生一正數提供給調整後的加法器，而不需先知會 PNWant 之數值爲負。此處理方式可使相加運算實施起來就像加總運算並以簡化的方式進行捨入運算。

【0113】 此外，此標準化移位器 130 會將 PNMant 左移一個由 LDP、EffSub 與 Z 的函數所計算出來的量。值得注意的是，即使沒有發生最重要前導位元之消除，仍需要將 PNMant 左移零、一或二個位元位置以產生一有用之標準化儲存格式中間結果 150，以確保後續之捨入運算可正確進行。此標準化運算係由一個左移所構成，以將算術上最重要位元移動至標準化之最左位置，使能表示於如下所述之儲存格式中間結果 150。

【0114】 相較於傳統之 FMA 設計，此實施方式具有三個額外的優點。第一，此實施方式不需在部分乘積加法器 240 內插入額外的進位位元(若因應於 EffSub 對相加數尾數執行 2'補數時有此需求的話)。第二，此實施方式不需提供一大的符號位元偵測器/預測器模組，來檢測並選擇性補足非冗餘部分乘積與相加數加總值之冗餘總數與進位向量表示。第三，此實施方式不需輸入額外的進位位元來確保部分乘積與相加數加總運算中，選擇性補足之總數與進位向量表示之運算正確。

【0115】 關於第五圖之指數結果產生器 260，PNExp 產生器 265 產生一預標準化指數值 PNExp，此數值爲被乘數與乘數指數值 A_E 與 B_E、指數偏移值 ExpBias 與移位常數 SC 之函數。在一實施例中，PNExp 係以算式 $SC+ A_E+B_E$-ExpBias 來計算。

【0116】 IRExp 產生器 270 使 PNExp 遞減，作爲標準化移位器 130 執行之尾數標準化運算，以產生一中間結果指數 IRExp。此數值爲 PNExp 與前導位預測(leading digit prediction, LDP)之函數。隨後，IRExp 係轉送至如下所述之結果向量埠 280。

【0117】 中間符號產生器 280 產生中間結果符號指標 IRSgn，其爲 EffSub、E、A_S、B_S 與 Z 的函數。在一實施例中，IRSgn 在某些情況下係以被乘數符號位元 A_S 與乘數符號位元 B_S 之邏輯互斥或來計算。不過，若是 Z 位元爲二進位數一顯示相加運算已經執行，EffSub 也是二進位數一顯示有效減法，而 E 位元值爲二進位數零顯示沒有等待中的端迴進位(end-around carry)，IRSgn 就會以被乘數符號位元 A_S 與乘數符號位元 B_S 之邏輯互斥或的反相值(XNOR)來計算。換句話說，中間符號通常是 A 與 B 之乘積的符號。當相加數的大小大於 A 與 B 之乘積時，A 與 B 之乘積的符號會反轉，乘積-相加運算會是一有效減法運算，而相加運算之完成不需要使用端迴進位（因爲相加運算爲負）。

【0118】 中間結果符號指標 IRSgn 係用於一可用以確認最終符號位元以供具塊消去可能之 FMA 運算使用之創新方法。不同於傳統之分路 FMA 實施方式，此實施方式不需要符號預測器，也不需要用以預測符號之大量電路。另外，零結果的符號，或是具有符號零輸入之運算結果之符號，容易預先計算，來納入例如一捨入模式輸入。

【0119】 結果向量埠 280 輸出一儲存格式中間結果向量 IRVector，其包含中間結果指數 IRExp、中間結果符號 IRSgn 與中間結果尾數 IRMant。在此儲存格式之一實施例中，IRMant 包含 GMant 之最重要 m 位元，其中 m 爲目標資料類型的寬度。舉例來說，在 IEEE 雙精確度(double precision)運算中，結果向量埠 280 輸出 IRVector 作爲單一個符號位元、十一個指數位元、與 GMant 最重要 53 位元的組合。在儲存格式之另一個實施例中，m 等於 A_M、B_M 與 C_M 之尾數的寬度。在儲存格式之又一個實施例中，m 大於 A_M、B_M 與 C_M 之尾數的寬度。

【0120】 類似於 IEEE 之標準儲存格式，這些尾數位元中之單一個最重要位元係作爲儲存時之一隱含值。IRVector 儲存至一共享記憶體，例如重排緩衝器 30 之重命名暫存器 155，藉此，其他指令執行單元就可存取 IRVector，且/或 IRvector 可透過一結果轉送匯流排 40 轉送至其他指令執行單元。在一實施例中，IRVector 儲存至一重命名暫存器 155。此外，不同於架構暫存器會對重排緩衝器給予一固定不變之任務分派，中間結果向量係在重排緩衝器內給予一不可預期之任務分派。在另一實施例中，IRVector 暫時儲存於將用以儲存 FMA 運算之最終捨入結果之目標暫存器內。

【0121】 現在請參照第五圖之捨入指標產生器 245，不足位/溢位偵測器 275 產生不足位指標 U_I 與溢位指標 O_I，其爲 IRExp 與指數範圍值 ExpMin 與 ExpMax 的函數。指數範圍值係與儲存格式中間結果 150（下面將有進一步討論）之精確度或目標資料類型有關。若是 IRExp 小於可代表此 FMA 運算目標資料類型之指數值的範圍，或是小於可代表之任何中間儲存空間，例如重命名暫存器，之指數值的範圍，U_I 位元就會是二進位數一，否則

U_I 位元就會是二進位數零。相反地，若是 IRExp 大於可代表此 FMA 運算目標資料類型之指數值的範圍，或是大於可代表之任何中間儲存空間，例如重命名暫存器，之指數值的範圍，O_I 位元就會是二進位數一，否則 O_I 位元就會是二進位數零。另外，U 與 O 可經編碼以表示四個可能的指數範圍，其中至少一個編碼會表示不足位，而至少一個編碼會表示溢位。

【0122】 在傳統一般乘法單元之實施方式中，U_I 與 O_I 位元會報告至例外事件控制邏輯。不過，在執行 FMA1 子運算時，調整後的乘法器 45 會輸出 U_I 與 O_I 位元至中間儲存空間以供調整後的加法器 50 執行。

【0123】 端迴進位指標產生器 290 會產生等待中之端迴進位指標 E_I 位元，其爲 Z、EffSub 與 SumSgn 之函數。若是已確認之 Z 位元的數值爲二進位數一，表示部分乘積加法器 240 已經執行 C 的相加運算，已確認之 EffSub 變數顯示此相加運算造成一有效減法，並且 SumSgn 顯示所產生之未捨入非冗餘值 PNMant 爲正，E_I 位元就會是二進位數一。在其他情況下，E_I 就會是二進位數零。

【0124】 結果向量埠 280 儲存 GMant 之最重要位元作爲中間結果向量之中間結果尾數，而黏位元產生器 295 與捨入位元產生器 300 會使剩下較不重要之位元（例如超出中間結果尾數之第 53 個位元之位元）減少至剩下捨入(R_I)與黏(S_I)位元。黏位元產生器 295 係產生黏位元 S_I，其爲 SumSgn、Z、GMant 之最不重要位元、EffSub 與 XtraStky 位元之函數。捨入位元產生器 300 產生捨入位元 R_I，其爲 GMant 之最不重要位元之函數。

【0125】 捨入快取

【0126】 捨入位元埠 305 會輸出各個位元 U_I、O_I、E_I、S_I、

R_1 與 Z，藉此，這些位元就可以被其他執行單元（例如調整後的加法器 50）使用來產生 FMA 運算最終之捨入後結果。爲了方便說明，這些位元在本文中都表示爲捨入位元，即使其中有部分位元會在產生 FMA 運算最終結果的過程中有其他用途，又即使並非所有的位元都用於捨入運算。舉例來說，在某些實施方式中，O1 位元就不會用於捨入運算。這些位元可互換地被指爲運算控制指標。舉例來說，位元 Z 與 E，即指出那些後續運算需要執行；位元 U 與 O，即指出這些運算應如何執行。此外，這些位元可表示爲運算間歇狀態值(calculation intermission state value)，因爲他們提供一壓縮格式(compact format)來表示與選擇性地儲存在調整後的乘法器 45 之 FMA1 子運算與調整後的加法器 50 之 FMA2 子運算間之間歇時間中的運算狀態資訊。

【0127】 這些位元，無論被稱爲捨入位元、運算控制指標、運算狀態指標或其他，連同中間結果向量與相加數值 C，除了運算元數值，還提供後續指令執行單元需要之任何事物，以產生算術上正確之最終結果。換句話說，中間結果向量與捨入位元之結合可提供算術上正確表示 FMA 運算結果所需之任何結果，此運算結果與一具有±A*B±C 形式但變爲目標資料尺寸之無限精確 FMA 運算之運算結果係無從辨別。

【0128】 依據本發明之一實施例，微處理器 10 係用以將捨入位元儲存至捨入快取 55 內，並將捨入位元透過轉送匯流排 40 轉送至其他指令執行單元，此捨入快取 55 亦可稱爲運算控制指標儲存空間。依據本發明之另一實施例，微處理器 10 並不具有捨入快取 55，而僅將捨入位元透過轉送匯流排 40 轉送至其他指令執行單元。依據本發明之又一實施例，微處理器 10 係將捨入位元儲存

至捨入快取 55 內，但不提供轉送匯流排 40 來將捨入位元直接轉送至其他的指令執行單元。

【0129】 指令快取 55 與所儲存之指令位元或運算控制指標係非架構，亦即其非爲終端用戶所能看見。相較之下，架構暫存器或架構指標（例如浮點狀態字(floating point status word)）則是可以被程式人員看見且指定爲架構指令集之一部分的訊號來源。

【0130】 在此所描述之捨入位元僅爲例示，而不同的實施方式會產生不同之捨入位元組。舉例來說，在另一實施例中，調整後的乘法器 45 亦包含一保護位元產生器以產生一保護位元 G_1。在另一實施例中，調整後的乘法器亦對一零結果(zero result)之符號執行一預運算，並將其值儲存於捨入快取。若是調整後的加法器 50 後續運算之結果爲零結果，捨入加法器 50 就會使用此儲存的零結果符號指標來產生最終之帶符號零結果。

【0131】 依據本發明之另一實施例，捨入快取 55 係一位於調整後的乘法器 45 外部之記憶體儲存空間。不過，在另一個不同的實施例中，此捨入快取 55 係結合於調整後的乘法器 55 內。

【0132】 尤其是在一實施例中，此捨入快取 55 係未經結果匯流排而獨自連接至指令執行單元。有鑑於結果匯流排係用以將結果從指令執行單元傳送至一通用儲存空間，捨入快取 55 獨自連接至結果匯流排 55 而未經結果匯流排。此外，運算控制指標儲存空間僅能爲用於儲存或載入運算控制指標之指令所存取。藉此，即可透過輸出指令結果之結果匯流排以外之其他機制來存取捨入快取 55，舉例來說，可透過捨入快取自身的導線。此外，亦可透過指令執行單元之輸入運算元埠外之其他機制來存取捨入快取 55。

【0133】 在一實施例中，捨入快取 55 係一完全關聯(fully

associative)之可存取記憶體，其具有之寫入埠的數量係相當於可平行分派之 FMA1 微指令之最大數量；其具有之讀取埠的數量係相當於可平行分派之 FMA2 微指令之最大數量；而其深度（項目數量）係關聯於指令排程之容量，與 FMA1 微指令分派後而在指令排程分派 FMA2 微指令前所能清空之最長時間週期（以時鐘週期計）。另一實施例係使用較小之捨入快取 55，而微處理器 10 係在捨入快去 55 內無法取得儲存 FMA1 微指令之捨入位元結果的空間時，重新執行 FMA1 微指令。

【0134】 快取之各個項目係儲存快取資料與相關聯之旗標值(tag value)，此旗標值可與用以辨識儲存有儲存格式中間結果向量之重命名暫存器 155 之旗標值相同。在微處理器 10 準備(prepare)/拿取(fetch)供第二使用之運算元時，微處理器 10 係使用重排緩衝器索引(ROB index)來由重命名暫存器 155 取回所儲存之中間資料，與此相同之索引將會提供至捨入快取 55，並提供中間結果 150 之剩下部分（即運算控制指標）。

【0135】 其優點在於，本發明之配置給捨入快取 55 之物理上儲存項目的數量係明顯少於配置給重命名暫存器 155 之項目數量。重命名暫存器 155 之數量係進行中(in flight)之微指令數量與需要使非循序微處理器或設計之執行單元飽和所需之暫存器名稱數量的函數。比較起來，捨入快取 55 所需之項目數量則爲進行中之 FMA 微指令之可能數量的函數。因此，在一未受限之實例中，微處理器之核心可提供六十五個重命名暫存器 155，但只提供八個捨入快取 55 之項目提供給至多八個平行處理之算術運算。

【0136】 本發明之另一實施例係擴充用以儲存中間結果向量之重命名暫存器 155（即擴大重命名暫存器之寬度），以提供額外

的位元供捨入快取 55 使用。此實施例雖非本發明空間利用之最佳者，但亦屬於本發明之範圍內。

【0137】 捨入位元連同中間結果指標 IRVector 係包含儲存格式中間結果 150。此儲存格式係依據一標準資料格式儲存與/或傳送未捨入標準化加總結果 145 之最重要位元（即具有默認值之位元），並且將未捨入標準化加總結果 145 之剩下位元（縮減或未縮減），連同 E_I、Z、U_I 與 O_I 位元，進行儲存與/或傳送，因而相較於先前技術具有顯著的優勢。

【0138】 **調整後的加法器**

【0139】 現在請參照第六圖，調整後的加法器 50 包含一運算元調整器 60、對準與條件邏輯 330、以及一個與一單一位元溢位移位邏輯 345 配對之遠路徑相加模組 340。此運算元調整器 160 並包含一指數產生器 335、一符號產生器 365、一加法器捨入位元產生器 350、捨入位元選擇邏輯 175 與一捨入運算模組 180。

【0140】 值得注意的是，在本發明之一實施例中，調整後的加法器 50 具有一分路設計而使其透過各自獨立之近運算與遠運算來計算結果，此技術爲浮點運算設計之技術人員所習知。近路徑之計算能力需要一個與一多位元標準化移位器（未圖示）配對之近路徑加總模組（未圖示），此等能力未在第六圖中顯現。在一實施例中，運算元 C 與 D 之輸入指數值的差值位於{-1, 0, +1}內而構成有效減法之一般加總運算會被導向至近路徑 16，其他的加法運算則會被導向至遠路徑 170。其優點在於，本發明可使調整後的加法器 50 之所有的 FMA2 子運算都被導向至遠路徑 170。

【0141】 調整後的加法器 50 提供一個或多個輸入埠 310 以接收一個微指令與兩個輸入運算元。第一輸入運算元 D 係一被減數

或一第一加數。第二輸入運算元 C 係一減數或一第二加數。在浮點運算之實施例中，各個輸入運算元包含一輸入符號、一指數、與一尾數值，分別以下標 S、E 與 M 表示。透過解譯微指令，解碼器 315 係利用信號 QS 指出此運算究竟爲一加法或是一減法運算。透過解譯微指令（或是由微指令指令之一運算元參考），解碼器並可以利用信號 M 來指出此微指令是否支配一個特定的微操作可使調整後的加法器 50 執行 FMA2 子運算。

【0142】 在調整後的加法器 50 被賦予執行 FMA2 子元運算之任務時，調整後的加法器 50 係接收由調整後的乘法器 45 先前執行相對應之 FMA1 子運算所產生之中間結果向量 IRVector。因爲中間結果向量 IRVector 之寬度僅爲 m 個位元，調整後的加法器 50 不需（而在一實施例中，係不會）被調整成可接收或執行多於 m 個位元之有效位數。因此，相較於以較寬位元數呈現之 IRVector，本實施例可以簡化內部資料路徑、相加模組 340 與調整後的加法器 50 之其他電路，並使其運作更有效率。此外，因爲涉及塊消去之相加運算係由調整後的乘法器 45 完成，在調整後的加法器 50 之近/塊消去路徑上不需加入捨入運算邏輯來正確計算出 FMA 結果。

【0143】 在一實施例中，調整後的加法器 50 從重命名暫存器 155 接收 IRVector。在另一實施例中，則是從轉送匯流排 40 接收 IRVector。而在第六圖所示之實施例中，IRVector 係被接收爲運算元 D。此調整後的加法器 50 接收相加數值 C 作爲另一個運算元。

【0144】 若是 M 顯示調整後的加法器 50 被賦予執行 FMA2 子元運算之任務，運算元調整器 160 就會在 Z 是二進位數一時，將輸入運算元之一部份設定爲等同於二進位數零，以顯示 C 之相

加運算已經由調整後的乘法器 45 執行。在一實施例中，各個指數、尾數與符號之欄位 C_E、C_M 與 C_S 均調整爲零。在另一實施例中，只由指數與尾數的欄位 C_E、C_M 調整爲零，至於符號的欄位 C_S 則維持原樣。藉此，調整後的加法器 50 即可將加數 D 加上一帶符號的二進位數零。

【0145】 一個二進位數一之 M 位元並通知調整後的加法器 50 接收由調整後的乘法器 45 所產生之捨入位元，並將其併入儲存格式中間結果 150。

【0146】 在其他所有情況中，即 Z 爲二進位數零或是 M 爲二進位數零以顯示調整後的加法器 50 被賦予傳統相加運算之任務，運算元調整器 160 只會對指數與尾數欄位 C_E 與 C_M 進行傳統浮點加法運算需要之調整。

【0147】 在一實施例中，運算元調整器 160 包含一對多工器接收 Z 的值，而在 C_M 與零之間以及 C_E 與零之間進行選擇。選定之值係在第六圖中以 C_M*與 C_E*表示。隨後，對準與條件邏輯 330 將對準與/或調節此選定值 C_M*與第一運算尾數 D_M。

【0148】 接下來，遠路徑相加模組 340 將 C_M*與 D_M 相加。在一實施例中，此相加模組 340 係一雙加加法器，以提供總數與漸增總數。在一實施例中，此相加模組 340 係利用 1'補數(one' s complement)來執行有效減法。若是此總數會在尾數欄位產生一位元之溢位，溢位移位邏輯 345 就會條件地將總數移動一個位元，以使結果值完成捨入運算之準備。

【0149】 指數產生器 335 利用選定的指數值 C_E*、第一運算元指數 D_E、與由溢位移位邏輯 345 產生之移位量來產生一最終指數 FExp。

【0150】 符號產生器 365 依據由第一與第二運算元 C_S 與 D_S、加/減運算子 Q_S 與加總結果符號構成之函數來產生一最終符號 FSgn。

【0151】 在另一實施例中，運算元調整器 160 係以選擇器邏輯取代。當輸入解碼器顯示加法器正在執行 FMA2 子運算而 Z 是二進位數一以顯示 C 之相加運算已執行，此選擇器邏輯會使第一運算元 D 直接轉送至捨入運算模組 180，但使加總邏輯維持在休眠狀態(quiescent state)。

【0152】 調整後的加法器 50 之邏輯係產生一組自己的捨入位元 R_2、S_2、U_2、O_2 與 E_2。當 M 顯示調整後的加法器 50 被賦予執行 FMA2 子運算之任務時，調整後的加法器 50 也會接收複數個由執行 FMA1 子運算之調整後的乘法器 45 事先產生之捨入位元 R_1、S_1、U_1、O_1、Z 與 E_1。

【0153】 在 M 爲二進位數一的情況下，捨入位元選擇邏輯 175 會確認來自調整後的乘法器 45 之捨入位元 E_1、R_1 與 S_1、來自調整後的加法器 50 之捨入位元 E_2、R_2 與 S_2、或是二者之某些混合或組合，將會被加法器之捨入運算模組 180 用以產生最終之捨入後尾數結果。舉例來說，若是所執行之運算並非 FMA2 子運算(即 M=0)，捨入運算模組 180 就會使用加法器產生之捨入位元 E_2、R_2 與 S_2。另外，若是相加運算係由調整後的乘法器 45 執行（即 M=1 且 Z=1)，並且不存在不足位的情形（即 U_M=0)，就由乘法器產生之捨入位元 E_1、R_1 與 S1 來提供捨入運算模組 180 產生最終捨入後結果所需之任何物件。

【0154】 此可變位置之捨入運算模組係作爲調整後的加法器 50 之遠計算功能之部分來設置。而在一實施例中，此捨入運算模

組配合由 1'補數有效減法所造成之正差值捨入運算，並且還配合由非有效減法之加總運算所造成之正總數捨入運算。此捨入運算模組 180 以類似於傳統單一加/減單元執行之方式，執行選定之捨入位元 R_X、黏位元 S_X，而在有提供的時候也會執行保護位元 G_X（未圖示）。此捨入運算模組 180 係由傳統設計進行修改以接收至少一補充輸入，即選定之端迴進位位元 E_X，而若是由調整後的乘法器 45 執行 1'補數有效減法，即可顯示需要一端迴進位校正。使用選定之 R_X、S_X 與 E_X 輸入，捨入運算模組 180 可正確地對中間結果向量與帶符號零之加總執行捨入運算，以產生正確而符合 IEEE 標準之結果。此為浮點運算設計之技術領域者所能理解。

【0155】 如前述，調整後的加法器 50 需要近路徑 165 來執行某些類型之傳統相加運算，但不需要近路徑 165 來執行本文所述之 FMA 運算。因此，在執行本文所描述之 FMA 運算類型時，近路徑邏輯 165 會在 FMA 運算之過程中維持休眠狀態以降低耗能。

【0156】 **第一與第二 FMA 子運算**

【0157】 第七至十圖顯示本發明利用一第一 FMA 子運算(FMA1)與一第二 FMA 子運算(FMA2)執行一不可分割分路乘積-相加運算之方法之一實施例。其中，FMA2 子運算並非臨時性相接於 FMA1 子運算，也不是物理性上相接於 FMA2 子運算。

【0158】 第七圖顯示 FMA1 子運算之一路徑確認部分。在步驟 408 中，FMA1 子運算確認 EffSub 變數。當 EffSub 為二進位數一，即顯示是否相加數運算元與乘數運算元之乘積的相加運算會構成一有效減法。在步驟 411 中，FMA1 子運算選擇性地對相加數運算元執行位元反相。在步驟 414 中，FMA1 子運算計算 ExpDelta。ExpDelta 等於乘數與被乘數之指數之加總減去相加數

之指數與一指數偏移。ExpDelta 不只確認乘數尾數與相加數尾數之相對對準以供加法運算之用，也會配合 EffSub 變數來確認相加數運算元之相加運算是否將由 FMA1 子運算執行。

【0159】 在步驟 417 中，FMA1 子運算確認路徑控制信號 Z。當數值爲二進位數一，即表示有一個相加數運算元之加總運算，會利用調整後的乘法器 45 電路，在 FMA1 子運算中執行。在一實施例中，FMA1 子運算在 ExpDelta 大於或等於負一時，將二進位數一指派給 Z，而在 ExpSub 爲一且 ExpDelta 爲負二時，也會將二進位數一指派給 Z。其他的實施方式會以不同方式分割 ExpDelat 與 EffSub 的數字空間。

【0160】 第八圖係 FMA1 子運算之乘法與條件相加部分之方塊示意圖。在步驟 420 中，FMA1 子運算選擇一相加路徑供相加運算元使用。若是 Z 爲二進位數零，如步驟 426 所示，FMA1 子運算就會計算乘數運算元之部分乘積之加總，而不加上相加數運算元。另外，若是 Z 爲二進位數一，如步驟 423 所示，FMA1 子運算會調整選擇性互補相加數尾數之對準值，其調整量爲 ExpDelta 值之函數。就一實施例而言，此調整量等於 ExpDelta 加上一移位常數。

【0161】 在步驟 426/429 中，FMA1 子運算執行一第一相加運算，其爲(a)乘數和被乘數運算元之部分乘積（即步驟 426），或是(b)相加數運算元與乘數和被乘數運算元之部分乘積（即步驟 429）之一者。在步驟 432 中，FMA1 子運算條件執行一前導位預測，來預測總數之最重要前導位所需要之消去。前導位預測之運算係限於類型 1 之 FMA 運算 75，並且會與步驟 429 中部分之加總運算平行執行。另外，前導位預測邏輯可連接至步驟 426 或步驟 429

以處理其運算結果。

【0162】 經過步驟 426 或 429，以及步驟 432 之執行後，FMA1 子運算會產生一未經捨入、非冗餘之標準化加總結果 145（如步驟 435 所示）。接下來，FMA1 子運算會產生一儲存格式中間結果 150（如步驟 438 所示）。一旦儲存格式中間結果 150 被儲存或分派至轉送匯流排 40，FMA1 子運算就會終止，並釋放執行 FMA1 子運算之資源（例如作爲調整後的乘法器 45 之指令執行單元）給其他與 FMA 運算無關的運算使用。所屬技術領域者當可理解，此技術亦可同樣適用於可同時透過連續階段執行多個運算之管線乘法器。

【0163】 第 9A 與 9B 圖係詳細說明產生儲存格式中間結果 150 之程序。在步驟 441 中，FMA1 子運算確認是否因相加數之相加運算構成有效減法，而存在待定之端迴進位校正。若是 Z 與 EffSub 均爲二進位數一（即類型 1 之 FMA 運算 75 或是類型 4 之 FMA 運算 90），並且由步驟 435 所產生之未經捨入非冗餘之結果爲正，FMA1 子運算就會將二進位數一指派給變數 E_1。

【0164】 在步驟 444 中，FMA1 子運算透過對尾數執行位元反相之運算，而產生一原始尾數結果(germinal mantissa result, GMant)。若運算結果爲負，就透過移位，將尾數標準化爲一標準儲存格式。

【0165】 在步驟 447 中，FMA1 子運算產生一中間結果符號(IRSgn)。若是 E 爲二進位數零，而 Z 與 EffSub 均爲二進位數一，IRSgn 就會是被乘數與乘數之符號位元之反邏輯互斥或結果。否則，IRSgn 就會是被乘數與乘數之符號位元之邏輯互斥或結果。

【0166】 在步驟 453 中，FMA1 子運算將 SC 加上乘數與被乘

數之指數值加總，再減去 ExpBias，以產生 PNExp。

【0167】 在步驟 456 中，FMA1 子運算透過減少 PNExp 來處理 PNMant 之標準化運算，藉以產生中間結果指數值(IRExp)。

【0168】 在步驟 459 中，FMA1 子運算確認中間不足位(U_I)與中間溢位(O_I)位元。

【0169】 在步驟 462 中，FMA1 子運算由原始尾數(GMant)之最重要位元產生一中間結果尾數(IRMant)。

【0170】 在步驟 465 中，FMA1 子運算將構成中間結果向量 IRVector 之 IRSgn、IRMant 與 IRExp，儲存至儲存空間，例如一重命名暫存器。

【0171】 在步驟 468 中，FMA1 子運算會將 GMant 之 LSBs 與部分乘積加法器 240 之移出位元(Xtrastky)縮減至捨入位元(R_I)與黏位元(S_I)。而在另一實施例中，還包含一保護位元(G_I)。

【0172】 在步驟 471 中，FMA1 子運算將 R_I、S_I、E_I、Z、U_I 與 O_I 位元紀錄於捨入快取 55，而在提供有 G_I 位元時，也會一併記錄於捨入快取 55。

【0173】 第十圖係一方塊圖顯示本發明不可分割分路 FMA 運算之一第二 FMA 子運算。

【0174】 在步驟 474 中，FMA2 子運算接收先前儲存在例如重命名暫存器之儲存空間內的中間結果向量 IRVector。另外，FMA2 子運算亦可由轉送匯流排接收 IRVector。

【0175】 在步驟 477 中，FMA2 子運算接收先前儲存在例如捨入快取 55 之儲存空間內的捨入位元。另外，FMA2 子運算亦可由轉送匯流排接收捨入位元。

【0176】 在步驟 480 中，FMA2 子運算接收相加數輸入值。

【0177】 在決定步驟 483 中，FMA2 子運算檢視步驟 474 所接收之 Z 位元。若是 Z 位元爲二進位數一（或真）表示相加數之加總運算已經由 FMA1 子運算執行，此流程就會前進至步驟 486。否則就會前進至步驟 489。

【0178】 在步驟 486 中，FMA2 子運算將相加數輸入值之指數與尾數欄位調整爲零。在一實施例中，FMA2 子運算並不調整輸入相加數之符號位元。隨後在步驟 492 中，此 FMA2 子運算會將計算中間結果指數與一帶符號零運算元之總數。接下來前進至步驟 494。

【0179】 在步驟 489 中，FMA2 子運算計算中間結果指數與相加數之總數。接下來前進至步驟 494。

【0180】 在步驟 494 中，FMA2 子運算利用 FMA1 子運算產生之 Z、U_1、O_1 位元以及 FMA2 子運算產生之 U_2 與 O_2 位元，從捨入位元 E_1、E_2、R_1、R_2、S_1 與 S_2 中，選擇會用來對總數之尾數進行正確捨入運算的捨入位元。

【0181】 在步驟 496 中，FMA2 子運算利用選定的捨入位元來對總數進行正確的捨入運算。此 FMA2 子運算在執行尾數捨入程序之同時，選擇性地使 IRExp 遞增（步驟 498）。如此，FMA2 子運算即可產生一最終經捨入之結果。

【0182】 第七至十圖中所描述之部分步驟可不需圖示順序執行。此外，第七至十圖中所描述之部分步驟亦可平行執行。

【0183】 **運算類型之應用**

【0184】 此章節說明前述應用於第二圖之五種不同運算“類型”之各種變數數值間之功能性關聯，尤其著重於 PNMant 之運算、符號與標準化處理以及與各個資料類型相關之 EffSub、

ExpDelta、Z、E 與 IntSgn 之數值。

【0185】 **第一類型(First Type)**

【0186】 如第二圖所示，類型 1 之 FMA 運算 75 之特點在於，此運算涉及一有效減法(因此，EffSub=1)，而調整後的乘法器 45 被選定以執行 C 之相加運算(因此，Z=1)，而 C 的大小相當接近 A 和 B 之乘積(即-2<=ExpDelta<=1)而會造成塊消去之產生。

【0187】 因爲相加運算將會於調整後的乘法器 45 內執行而造成有效減法（即 EffSub=1 且 Z=1），相加數對準與射入模組 220 會在將相加數運算元尾數值 C_M 射入部分乘積加法器 240 前，對相加數運算元尾數值 C_M 造成與/或選擇一位元反相運算。相對於部分乘積，此相加數對準與射入模組 220 會利用 ExpDelta，來對準部分乘積加法器 240 內之相加數尾數。

【0188】 隨後，對於未經捨入之非冗餘數值 145（即 PNMant）的完整加總運算係依照傳統乘法執行單元通常採用之方法來執行，此執行包含部分乘積之加總運算內，額外被選擇性位元求與相對準之相加數輸入值。因而 PNMant 即可以 1'補數之形式，表示乘數與被乘數尾數值以及相加數尾數值間之算數差。

【0189】 PNMant 可爲正或負。若是 PNMane 爲正就需要端迴進位，而等待中之端迴進位指標 E_1 就會被指派爲二進位數一。若是 PNMant 爲負就不需要端迴進位，而 E_1 就會被指派爲二進位數零。E_1 被指派之值不僅爲 PNMant 之函數，亦爲 Z 與 EffSub 之值的函數，而在類型 1 之運算 75 中，Z 與 EffSub 會是二進位數一。

【0190】 在部分乘積與相加數輸入加總運算之同時，也會執行前導位預測來對任何最重要前導位需要之任何消去進行預測。如前述，於本發明一實施例中，此運算係由一與最終加法器 125

並行之電路，於執行 PNMant 之加總運算時執行。

【0191】 如浮點運算設計之技術領域者所能理解，即使沒有發生前導位之相減消去，可能還是需要對 PNMant 執行一個零、一或二位元位置之標準化運算，利用 SC 至 PNExp 來使其對準於所需之儲存格式，以供本發明描述與使用之中間結果 150 之用。若存在塊消去，顯然就會需要更多的移位運算。同樣地，若是 PNMant 爲負，就會對此數值執行位元反相之運算。對 PNMant 執行選擇性標準化與位元反相運算係用以產生原始尾數值 GMant，其最重要 m 位元係變成中間結果尾數 IRMant。

【0192】 中間結果符號 IRSgn 不是被乘數符號位元 A_S 與乘數符號位元 B_S 之邏輯 XOR 運算結果，就是其邏輯 NXOR 運算結果，端視 E_I 之數值而定。若是 E_I 爲二進位數一，IRSgn 會被計算爲被乘數符號位元與乘數符號位元之邏輯 XOR 運算結果。若是 E_I 爲二進位數零，IRSgn 就會被計算爲被乘數符號位元與乘數符號位元之邏輯 XOR 運算結果。

【0193】 回到 FMA2 運算，調整後的加法器 50 會接收包含路徑控制訊號 Z 在內之已儲存或轉送之捨入位元，因爲 Z 爲 1，中間結果向量 IRVector 需要執行捨入運算，而會對最終之乘積-相加結果產生些微調整。在一實施例中，調整後的加法器 50 係將中間結果向量 IRVector 與一零運算元（或是在另一實施例中，一帶符號二進位數零之運算元）加總，而非與所提供之第二運算元，即相加數 C，做加總運算。

【0194】 在最後的執行步驟中，調整後的加法器 50 會在加總運算與完成捨入運算前對接收到的 IRExp 進行調整，以涵蓋較大的數字範圍，例如可涵蓋 FMA 運算之目標資料類型之不足位與

溢位指數範圍。在接收到的數值 Z=1 位元時，調整後的加法器 50 會以採取一個使 IRExp 遞增、而大部分均爲傳統方式之方法，利用所接收到的 R、S、U、O 與 E 位元來對 IRVector 執行捨入運算。

【0195】 **第二類型(Second Type)**

【0196】 如第二圖所示，類型 2 之 FMA 運算 80 之特點在於，此運算不涉及有效減法（因此，EffSub=0），調整後的乘法器 45 係被選定以執行 C 之相加運算（因此，Z=1），而相較於 A 和 B 之乘積，C 的大小相當小。

【0197】 因爲此運算不會造成有效減法（即 EffSub=0），相加數對準與射入邏輯 220 就不會在將相加數運算元尾數值 C_M 射入部分乘積加法器 240 前，對相加數運算元尾數值 C_M 造成與/或選擇一位元反相運算。

【0198】 相加數對準與射入模組 220 利用 ExpDelta 來使相加數尾數對準部分乘積，以將相加數尾數射入部分乘積加法器 240。

【0199】 在此將不會產生負的 PNMant。此外，所產生之正的 PNMant 並非 1'補數之減法運算的結果，因而不需端迴進位校正。因此，等待中之端迴進位指標 E_1 就會被指派爲二進位數零。

【0200】 因爲並非一有效減法運算，此運算中將不會產生前導位之減法塊消去，因而不需執行前導位預測來預測此等消去。反之，前導位預測則可用以依據 SC 至 PNExp 之貢獻程度，來預測所需之 0、1、或 2 位元位置之標準化運算。

【0201】 如同浮點運算設計之技術領域者所能理解，A 和 B 乘積與 C 的加總會產生一具有算術顯著性或比重、較乘數與被乘數之乘積大於一數位位置的溢位情形，因此需要對 PNMant 執行零、一或二位元位置之標準化運算，來使數值對準用於本發明所

描述與利用之中間結果之儲存格式。此標準化運算會產生原始尾數值 GMant，而其最重要 m 位元會成爲中間結果尾數 IRMant。

【0202】 預標準化指數 PNExp 係透過將輸入之乘數與被乘數指數值相加，再減去任何指數偏移值，最後再加上 SC=2，此數值係於 Z=1 之情況下依據最負的 ExpDelta 而定。如第二圖對於類型 2 之運算之描述可知，C 的大小並不會明顯大於 A 和 B 的乘積，因此，所造成之總數會等於或大於所輸入之相加數。

【0203】 因爲此運算並非有效減法（即 EffSub=0），中間結果符號 IRSgn 係視爲被乘數符號位元 A_S 與乘數符號位元 B_S 之 XOR 邏輯運算結果。

【0204】 回到 FMA2 運算，調整後的加法器 50 會接收包含路徑控制信號 Z 之已儲存或轉送之捨入位元。因爲 Z 是二進位數一，中間結果向量 IRVector 只需要一些最後處理，主要就是捨入運算，即可產生最終之乘積-相加結果。在一實施例中，此調整後的加法器 50 係將中間結果向量 IRVector 與一零運算元（在另一實施例中，可爲一帶符號二進位數零運算元）做加總，而非與所提供之第二運算元，即相加數 C，做加總運算。

【0205】 在最後的執行步驟中，調整後的加法器 50 會對接收到的 IRExp 進行調整，以涵蓋較大的數字範圍，例如可涵蓋 FMA 運算之目標資料類型之不足位與溢位指數範圍。此調整後的加法器 50 會以採取一個使 IRExp 遞增、而大部分均爲傳統方式之方法，來對 IRVector 執行捨入運算以產生最終之正確結果。

【0206】 **第三類型(Third Type)**

【0207】 如第二圖所示，類型 3 之 FMA 運算 85 之特點在於，此運算不涉及有效減法（因此，EffSub=0），調整後的加法器 50

被選定以執行 C 之相加運算（因此，Z=0），而相較於 A 和 B 之乘積，C 的大小相當大。

【0208】 因此，EffSub 爲二進位數零。此外，路徑控制信號 Z 亦爲二進位數零，以顯示相加數運算元之加總運算尚未執行。因爲 Z 與 EffSub 均爲二進位數零，等待中之端迴進位指標 E_1 會被指派爲二進位數零。

【0209】 因爲 Z 是二進位數零，相加數對準與射入邏輯 220 不會將乘數單元部分乘積加總數內之相加數輸入的尾數進行對準。不過，相加數對準與射入邏輯 220 會使此對準輸入之算術值爲零。

【0210】 隨後，對於部分乘積與未經捨入之非冗餘數值 145 之完整加總運算係依照傳統乘法執行單元通常採用之方法來執行，此方法不包含輸入相加數之尾數值。因爲此 FMA 運算類並非有效減法(即 EffSub=0)，此加總運算將不會產生正的 PNMant，並由 SumSgn 表示。此外，此正值的 PNMant 並非 1'補數減法運算之結果，因而不需要使用端迴進位校正。

【0211】 因爲此運算並非一有效減法運算，此運算中將不會發生前導位之減法塊消去，因而不需執行前導位預測來預測此等消去。

【0212】 A 和 B 的乘積會在乘數與被乘數之尾數乘積產生一位數位置之算術溢位。因而需要對此正的未經捨入非冗餘值執行零或一位元位置之標準化運算，來使此數值對準本發明所描述或使用之中間結果格式。此標準化運算會產生原始尾數值 GMant，而其最重要 m 位元會成爲中間結果尾數 IRMant。

【0213】 因爲事先確認之路徑控制訊號 Z 爲二進位數零，顯

示相加運算尚未被執行，此中間結果符號 IRSgn 就會被視爲被乘數符號位元 AS 與乘數符號位元 BS 之 XOR 邏輯運算結果。

【0214】 回到 FMA2 運算，調整後的加法器 50 接收包含 Z 在內之已儲存或轉送之捨入位元。因爲 Z 爲二進位數零，調整後的加法器 50 會使中間結果向量，即第一運算元，與相加數 C，即第二運算元，做加總運算。

【0215】 在執行此相加運算之前，此調整後的加法器 50 可調整 IRExp 以涵蓋較大的數字範圍，例如可涵蓋 FMA 運算之目標資料類型之不足位與溢位指數範圍。因爲此運算是由相加數數值主導運算結果之類型 3 之運算 85，IRExp 將會比相加數輸入指數值來的小。

【0216】 此運算有利於使調整後的加法器 50 之遠路徑加總運算之二個運算元生效。在遠路徑加總運算中，具有較小指數值之運算元的尾數係於對準程序中向右移位。任何如此移位超過所需捨入位元的尾數位元就會被用於捨入運算。因爲相加數會主導運算結果，因而可以不用提供位元給捨入運算，而能簡化必要的捨入運算。

【0217】 調整後的加法器 50 會使用由調整後的加法器 50 所執行之部分運算所產生之 G_2（如果有的話）、R_2、S_2 與 E_2（具有二進位數值零）捨入位元，連同 R_1、S_1 與 E_1，來對中間結果與相加數輸入值的總數進行捨入運算，以產生 FMA 運算之最終經捨入之正確結果，此爲熟習浮點運算設計領域技術者所能理解。

【0218】 **第四類型(Fourth Type)**

【0219】 如第二圖所示，類型 4 之 FMA 運算 90 之特點在於，此運算涉及有效減法（因此，EffSub=1），調整後的乘法器 45 被

選定以執行 C 之相加運算（因此，Z=1），而相較於 A 和 B 之乘積，C 的大小相當小。

【0220】 因爲相加運算將會於調整後的乘法器 45 內執行而造成有效減法（即 EffSub=1 且 Z=1），相加數對準與射入模組 220 會在將相加數運算元尾數值 C_M 射入部分乘積加法器 240 前，對相加數運算元尾數值 C_M 造成與/或選擇一位元反相運算。相對於部分乘積，此相加數對準與射入模組 220 會利用 ExpDelta，來對準部分乘積加法器 240 內之相加數尾數。

【0221】 因爲 A 和 B 之乘積之大小明顯大於 C，此運算中將不會發生前導位之減法塊消去，因而不需執行前導位預測來預測此等消去。

【0222】 此外，加總程序會產生正的 PNMant。因此，待定之端迴進位指標 E_I 會被指派爲二進位數一，之後會通知調整後的加法器 50 還需要對中間結果尾數執行端迴進位校正。

【0223】 如同浮點運算設計領域具有合理技術者所能理解，PNMant 可能需要依據所提供之 SC 至 PNExp，執行零、一或二位元位置之移位或標準化運算，使其對準於本發明所述或使用之中間結果所需的儲存格式。隨後，此標準化運算會選擇性地執行於此未經捨入之非冗餘值，以產生原始尾數值 GMant，其最重要著 m 位元會成爲中間結果尾數 IRMant。

【0224】 因爲類型 4 之運算 90 涉及 C 之加總運算(即 Z=1)，而此加總運算會構成有效減法(即 EffSub=1)，在需要端迴進位(即 E_I 爲 1)之情況下產生正的 PNMant，中間結果符號 IRSgn 會被視爲被乘數符號位元 A_S 與乘數符號位元 B_S 之 XOR 邏輯運算結果。

【0225】 回到 FMA2 運算，調整後的加法器 50 接收包含路徑

控制信號 Z 在內之已儲存或轉送之捨入位元。因爲 Z 爲 1，中間結果向量 IRVector 只需一些最後處理，主要就是捨入運算，即可產生最終之乘積-相加結果。在一實施例中，此調整後的加法器 50 係將中間結果向量 IRVector 與一零運算元（在另一實施例中，可爲一帶符號二進位數零運算元）做加總，而非與所提供之第二運算元，即相加數 C，做加總運算。

【0226】 在執行零（或二進位數帶符號零）之相加運算前，調整後的加法器會調整 IRExp 以涵蓋較大的數字範圍，例如可涵蓋 FMA 運算之目標資料類型之不足位與溢位指數範圍。

【0227】 因應儲存格式中間結果 150 內所接收之 E 位元二進位數值，此運算需要依據第一微指令執行過程中執行之 1'補數有效減法產生端迴進位校正。因此，此 E 位元係連同儲存格式中間結果 150 之 G_I（如果有的話）、R_I 與 S_I 位元，做爲調整後的加法器 50 執行單元之調整後的捨入邏輯的補充輸入。

【0228】 隨後，此調整後的捨入邏輯利用 G_I（如果存在的話）、R_I、S_I 與 E_I 補充輸入來計算中間結果向量與帶符號零之加總的正確捨入，以對類型 4 之 FMA 運算產生正確的運算結果。而此技術當爲熟習浮點運算設計領域之技術者所能理解。

【0229】 **第五類型(Fifth Type)**

【0230】 如第二圖所示，類型 5 之 FMA 運算之特點在於，此運算涉及有效減法（因此，EffSub=1），調整後的加法器 50 被選定以執行 C 之相加運算（因此，Z=0），而相較於 A 和 B 之乘積，C 的大小相當大。

【0231】 因爲相加運算不會於調整後的乘法器 45 內執行，相加數對準與射入邏輯 220 不會對準部分乘積加總數內之 C_X，或是

使此對準輸入之算術值爲零。此調整後的乘法器 45 係依據先前技術之乘法執行單元之常用方法來執行部分乘積與 PNMant 之完整加總運算。

【0232】 因爲已執行 C 之加總運算，此運算中將不會發生前導位之減法塊消去，因而不需執行前導位預測來預測此等消去。此外，雖然會產生正的 PNMant，此數值並非 1'補數減法之運算結果。因此，此運算不需要端迴進位校正，而 E_1 會被指派爲二進位數零。

【0233】 如同熟習浮點運算設計領域之技術者所能理解，PNMant 可能需要依據所提供之 SC 至 PNExp，執行零、一或二位元位置之移位或標準化運算，使其對準於中間結果 150 所需的儲存格式。此標準化運算會產生原始尾數值 GMant，其最重要 m 位元會成爲中間結果尾數 IRMant。

【0234】 因爲類型 5 之運算不涉及 C 之加總運算（即 Z=0），中間結果符號 IRSgn 會被視爲被乘數符號位元 A_S 與乘數符號位元 B_S 之 XOR 邏輯運算結果。

【0235】 回到 FMA2 運算，調整後的加法器 50 接收包含 Z 在內之已儲存或轉送之捨入位元。因爲 Z 爲 0，中間結果向量 IRVector 需要與相加數 C 加總以產生最終之乘積-相加運算結果。

【0236】 因爲類型 5 之運算係由相加數值主導運算結果，IRExp 會小於相加數輸入指數值，此運算有利於使調整後的加法器 50 之遠路徑加總運算之二個運算元生效。在遠路徑加總運算中，具有較小指數值之運算元的尾數係於對準程序中向右移位，任何如此移位超過所需捨入位元的尾數位元就會被用於捨入運算。因爲相加數會主導運算結果，因而可以不用提供位元給捨入

運算，而能簡化必要的捨入運算。

【0237】 因爲由儲存格式中間結果 150 接收之待執行端迴進位指標 E_1 是二進位數零，因此，FMA1 運算沒有留下任何待執行之端迴進位校正。是以，E_1 位元係連同儲存格式中間結果 150 之 R_1 與 S_1 位元，以及 G_1 位元（如果存在的話），做爲調整後的加法器 50 執行單元之調整後的捨入位元的補充輸入。

【0238】 然而，由調整後的加法器 50 所執行之相加運算可能會另外導致 1'補數有效減法之產生。如此，調整後的捨入邏輯即可產生包含端迴進位在內之捨入位元，來計算出中間結果向量與相加器輸入值之加總的正確捨入值，以產生 FMA 運算之正確結果。而此技術當爲熟習浮點運算設計領域之技術者所能理解。

【0239】 **特定微指令**

【0240】 在本發明之另一個面向中，轉譯器與/或微碼唯讀記憶體(ROM)20 係用以將 FMA 指令轉譯或轉換爲第一與第二特定微指令，分別由乘法與加法單元執行。舉例來說，第一（或更多）特定微指令可以在一類似於將傳統乘法單元進行最小限度之調整，以符合所述目的之乘法執行單元內執行。而第二（或更多）特定微指令可以在一類似於將傳統家法單元進行最小限度之調整，以符合所述目的之加法執行單元內執行。

【0241】 第十一圖顯示本發明將融合 FMA 指令 535 透過 FMA 指令轉譯或轉換，產生第一與第二特定微指令 553 與 571 之一實施例。就一未受限之範例而言，此融合 FMA 指令 535 包含一指令操作碼欄 538、一目標欄 541、一第一運算元（被乘數）欄 544、一第二運算元（乘數）欄 547、與一第三運算元（相加數）欄 550。

【0242】 依據操作碼欄 538 之指示，此 FMA 運算可爲一乘-

加(multiply-add)、一乘-減(multiply-subtract)、一負乘-加(negative multiply-add)、或一負乘-減(negative multiply-subtract)指令。如同存在許多不同類型之 FMA 指令 535，也會存在許多類型之第一特定微指令 553，如乘-加、乘-減、負乘-加、或負乘-減微指令。這些類型之特徵，如果有的話，會反映在相關之微指令 553 之操作碼欄 556 中。

【0243】 第一特定微指令 553 會引導第一至第五類型之 FMA 運算所需之部分算術運算之實行。依據類型之不同，第一特定微指令 553 所執行之特殊運算亦有差異。第一特定微指令 553 係分配給第一執行單元，例如前文所述之調整後的乘法器 45。

【0244】 第二特定微指令 571 會引導第一至第五類型之 FMA 運算所需之剩餘算術運算之實行。依據類型之不同，第二特定微指令 571 所執行之特殊運算亦有差異。在本實施例中，第二特定微指令 571 係分配給第二執行單元，例如前文所述之調整後的加法器 50。爲了方便浮點乘-加融合運算或浮點乘-減融合運算之實行，此第二特定微指令 571 可具有一子類型，例如加或減。

【0245】 尤其是，此第一特定微指令 553 會將第一、第二與第三輸入運算元 544、547 與 550，分別特定爲被乘數運算元 A、乘數運算元 B 與相加數運算元 C。此第一特定微指令還可以特定一目標欄 559，指向暫時暫存器。又或者，此目標暫存器 559 可爲隱含者。

【0246】 第一特定微指令 553 會引導 FMA1 子運算執行，即 A 和 B 之部分乘積與在某些條件下與 C 之加總，以產生未經捨入之儲存格式中間結果 150。此第一特定微指令 553 也會引導變數 EffSub 與 ExpDelta 之確認動作，藉以對一組預先確定之 EffSub

與 ExpDelta 值，產生指派給 Z 位元之數值二進位數一。此數值會依序控制許多相關程序。

【0247】 二進位數一之 Z 位元指出此對相加數位元之加總運算將會於第一運算中執行，而不須由第二微指令來執行。隨後，此 Z 位元之指定與 ExpDelta 會用於使部分乘積加法器 240 內被選擇性互補相加數尾數進行對準，此部分乘積加法器 240 係經適當調整使能接收此額外事項。

【0248】 第一特定微指令 553 並引導一完整加總運算對此未經捨入之非冗餘值(PNMant)執行。此完整加總運算係依據傳統乘法執行單元之通常方法，但在部分乘積之加總運算中，納入額外之選擇性位元反相、已對準的相加數輸入值 C_M 或 $\overline{C_M}$。若是 PNum 爲負，此條件就會由信號 SumSgn 表示。

【0249】 第一特定微指令 553 並引導 PNMant 之移位與位元反相運算，以產生一原始尾數值(GMant)，接下來會執行 GMant 之縮減以產生儲存格式中間結果 150 之中間結果尾數(IMant)。因此，此中間結果尾數 IMant 會是此 EffSub 指定之運算所產生之 1’補數算術差之標準化絕對值，但不執行任何端迴進位校正。

【0250】 第一特定微指令 553 並引導中間結果指數值之運算。首先依據 Z 爲二進位數一時，ExpDelta 之最負值，產生一預標準化指數值(PNExp)，其數值係等於被乘數指數 A_E 與乘數指數 B_E 之加總、減去指數偏移值 ExpBias、再加上一移位常數 SC。然後，從 PNExp 之數值減去由標準化移位器 130 執行之尾數標準化運算所佔據的量，以產生中間結果指數值(IRExp)。

【0251】 第一特定微指令 553 並引導中間結果符號 IRSgn 之運算。此中間結果符號 IRSgn，連同中間結果尾數 IRMant 與中間

結果指數 IRExp，構成儲存格式中間結果 150 之向量 IRVector。

【0252】 第一特定微指令 553 並會產生包含 Z 在內之許多捨入位元。GMant 未納入中間結果尾數之最不重要位元會被縮減至由捨入(R)與黏(S)位元表示，而在一實施例中，還包含保護(G)位元。若是部分乘積加法器 240 已經將 C 與 A 和 B 之乘積相加，並且此運算係一有效減法而產生一正的 PNMant 值，端迴進位位元 E 就會配指派爲二進位數一，以顯示需要執行端迴進位。此第一特定微指令並會使中間不足位(U)與中間溢位(O)位元被確認。

【0253】 最後，在一實施例中，第一特定微指令 553 會使儲存格式中間結果 150 向量 IRVector 儲存於記憶體，在另一實施例會使其轉送，而在又一實施例，則會使其同時儲存與轉送。相同地，在一實施例中，第一特定微指令 553 會使捨入位元儲存於記憶體，在另一實施例中會使其轉送，而在又一實施例中，則會使其同時儲存與轉送。如此可使負責執行第一特定微指令之執行單元，在執行第一 FMA 微指令後而在執行第二微指令前，執行其他與 FMA 運算無關之運算。

【0254】 第二特定微指令 571 提供一操作碼 574 並分別特定第一與第二輸入加法器運算元 580 與 583。第二特定微指令 571 會促使 FMA2 運算執行。若是 C 之相加並非由第一特定微指令 553 造成，此運算還包含 C 與中間結果尾數之條件相加。第二特定微指令 571 並會產生 FMA 運算之最終捨入後結果。

【0255】 第一相加數運算元 580 係將第一特定微指令 553 所產生之乘積作爲其數值，第二相加數運算元 583 係將第一特定微指令所指定之相同相加數值作爲其數值。在一實施例中，第二特定微指令 571 之源運算元欄 580 係指向與第一特定微指令 553 之

目標欄 599 相同之暫存器。第二特定微指令 571 並特定一目標暫存器 577，在一實施例中，此目標暫存器與 FMA 指令 535 之目標欄 541 係爲同一個暫存器。

【0256】 結論

【0257】 雖然此實施方式係在有效減法時提供 1'補數之相加運算，熟習算術或浮點運算設計領域之技術者當可理解，在另一實施方式中，在有效減法時利用本發明之方法來使用 2'補數之相加運算。

【0258】 本發明具有許多優點。本發明相容於 IEEE，並提供其他實施方式明顯無法提供之校正給所需之 FMA 算術結果，尤其是因應 IEEE 捨入運算之要求。

【0259】 本發明透過維持可分別取用之乘法器與加法器單元，最大化供指令分配之各自獨立算術功能單元之可用性，而讓電腦處理器在一定實施成本下，可以更充分地開發指令平行處理。換言之，本發明讓最少的硬體達到最大可能性之同步利用，使其能以最快速度完成預期中最頻繁的運算，此處理方式可改善算數運算結果之產出。透過將所需之特定類型的第一與第二（或更多）微指令進行分配，並使其以暫時性與/或物理性各自獨立之方式分別執行，因此，雖然此等用於 FMA 之第一微指令會被分配至乘法功能單元執行，第二或其他更多無關的微指令則可同時被分配至一個或多個加法功能單元。

【0260】 同樣地，雖然此等用於 FMA 之第二微指令會配分配至功能方塊單元，任何需要乘法功能之其他無關的微指令則可同時被分配至乘法功能單元。

【0261】 如此，依據所需之整體效能與所需之系統指令平行

處理能力，所提供之乘法與加法功能單元的數量可更具彈性。因而相較於一個整體的 FMA 硬體，可減少每個功能單元之設置成本。電腦系統重排微指令之能力也會提升，而能降低成本與功耗。

【0262】 相較於其他設計，本發明不需要使用大型或特殊目的之硬體即可減少指令延遲。其他 FMA 硬體實施之設計都需要使用大型與複雜的電路功能，如先行標準化運算(anticipatory normalization)、先行加法運算、先行符號運算、與複雜之捨入電路。這些複雜元件常常會成爲最終設計中關鍵的耗時路徑、會在運算過程中消耗額外的能量、並在物理上會佔據電路設置空間。

【0263】 本發明不需在大型的FMA硬體內設置特殊旁路電路或模態以減少如先前技術所提供之簡單加法或乘法指令之延遲。

【0264】 本發明之其他實施方式，可在特殊類型之第一微指令的執行過程，執行或多或少之算術運算，可在特殊類型之第二微指令的執行過程，執行或多或少之算術運算。亦即配置給這些微指令之運算可爲不同。如此，其他這些實施方式可對任一/任何需要之獨立運算單元進行或多或少之調整。如此，這些其他實施方式可儲存或多或少之中間結果於捨入快取內，並可將或多或少之中間結果轉送至第二爲指令。

【0265】 其他實施方式中，所述之捨入快取可爲可定址暫存器位元、內容可存取記憶體、佇列儲存空間、或對映功能。

【0266】 其他實施方式可提供多個獨立的硬體或執行單元來執行第一微指令，與/或提供多個獨立的硬體或執行單元來執行第二微指令。同樣地，如果有利的話，這些實施方式也可以提供多個捨入快取給不同之源碼指令流或資料流、或是多核電腦處理器之各種實施方式。

【0267】 此實施方式係用於超純量、非循序指令分配，不過其他實施方式亦可用於依序指令分配之情形，例如，透過從指令快取移除以及提供至資料轉送網路之過程，可將指令從所提供之乘法運算單元分配至獨立的加法運算單元。本發明對於 FMA 運算分類之例示，以及本發明提及之所需最少量硬體調整，在依序指令分配之應用中，亦有其優點。雖然本說明書係將 FMA 運算區分爲五個類型，不過其他劃分方式，無論是較少、較多、與/或使用不同類型，均屬本發明之範疇。

【0268】 此外，雖然本說明書中已描述不同之調整後的乘法與加法單元以執行 FMA 運算，在本發明之一實施例中，亦可使用一乘積-相加單元因應第一乘積-相加指令來執行第一乘積-相加子運算，並將其結果係儲存至外部儲存空間，而此乘積-相加單元會再因應第二乘積-相加指令來執行第二乘積-相加子運算。

【0269】 本發明可用於 FAM 運算之單指令流多數據(SIMD)實施方式，此等方式有時會被稱爲向量指令類型或向量 FMA 運算。並且此方式會有多個調整後的乘法器之實例與多個調整後的加法器之實例。在一實施例中，本發明使用單一捨入快取來配合一個 SIMD 應用之需求。而在另一實施例中，則可使用多個捨入快取來配合多個 SIMD 應用之需求。

【0270】 雖然本發明係關於需要一乘法運算連同後續加法或相加運算之浮點融合乘法加法運算，不過，本發明所提供之方法亦可用於其他實施方式，尤其是針對某些中間結果使用快取之技術、針對需要兩個以上連鎖(chained)的算術運算之計算、針對不同算術運算、或以不同順序執行這些算術運算。舉例來說，某些時候可能需要將這些方法應用於其他算術運算之混合（即，涉及

兩個以上之算術操作子或三個以上之運算元的算術運算），例如連鎖的乘法-乘法-加法運算或乘法-加法-加法運算，以提升算術準確性或提升運算產出。此外，從其他面向來看，本發明亦可應用於整數算術運算。舉例來說，捨入至特定位元位置之整數運算可區分爲第一與第二子運算，其中第一子運算產生一未經捨入之中間結果，而第二子運算由此未經捨入之中間結果產生一經捨入之最終結果。依此，在需要時，其他實施方式可將不同之狀態位元記錄於快取機制內。

【0271】 本說明書關於捨入位元與其他內部位元之描述係爲利於說明本發明之精神，本發明當可用於其他類型之指標，包含捨入運算相關或運算控制相關之變數的不同編碼表現形式。此外，針對說明書中將變數描述爲具有“二進位數一”（也就是邏輯一）的許多段落，此描述係包含具有“二進位數零”（也就是邏輯零）之變數的等效布林實施方式，並包含這些變數之其他表示方式。相同地，針對說明書中將變數描述爲具有“二進位數零”（也就是邏輯零）的許多段落，此描述係包含具有“二進位數一”（也就是邏輯一）之變數的等效布林實施方式，並包含這些變數之其他表示方式。此外，依據本說明書當可理解，在本文中，相加係包含加法加總與加法差之情形。

【0272】 此外，依據本說明書當可理解，在本文中，指令係包含架構指令與可由架構指令轉譯或轉換產生之微指令。同樣地，指令執行單元並非僅限於微處理器直接執行架構指令而未預先將其轉譯爲微指令之情形。由於微指令亦屬指令之一種類型，因此，指令執行單元也會包含微處理器先將指令集架構指令轉譯或轉換爲微指令，而指令執行單元總是只執行微指令之情形。

【0273】 在本說明書中，尾數與有效位數之用語可互用。其他如原始結果與中間結果之用語，係爲區分在 FMA 運算之不同階段所產生之結果與表示方式。此外，在本說明書中，儲存格式中間結果之用語係包含一中間結果向量（意指一數的量）與複數個運算控制變數。這些用語不應被僵化或狹隘地解釋，而應依據申請人所表現出來的目的探究其實質，而理解這些用語依據前後文之不同可能指涉不同的事件。

【0274】 第一與第三至六圖中的功能方塊係可描述爲模組、電路、子電路、邏輯、與其他數位邏輯與微處理器設計之技術領域中常用於指稱由線路、電晶體與/或其他執行一個或多個功能之物理結構所構成之數位邏輯用語。本發明亦可涵蓋將說明書所描述功能以不同於說明書所描述之方式分派之其他替代實施方式。

【0275】 下列參考文獻係納入本案以說明FMA設計之相關概念與並使本案發明能爲讀者理解。

【0276】 參考文獻：

【0277】 Hokenek, Montoye, Cook, "Second-Generation RISC Floating Point with MultiplyAdd Fused", IEEE Journal Of Solid-State Circuits, Vol 25, No 5, Oct 1990.

【0278】 Lang, Bruguera, "Floating-Point Multiply-Add-Fused with Reduced Latency", IEEE Trans On Computers, Vol 53, No 8, Aug 2004.

【0279】 Bruguera, Lang, "Floating-Point Fused Multiply-Add: Reduced Latency for FloatingPoint Addition", Pub TBD - Exact Title Important.

【0280】 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.

【0281】 Galal, Horowitz, "Energy-Efficient Floating-Point Unit Design", IEEE Trans On Computers Vol 60, No 7, July 2011.

【0282】 Srinivasan, Bhudiya, Ramanarayanan, Babu, Jacob, Mathew, Krishnamurthy, Erraguntla, "Split-path Fused Floating Point Multiply Accumulate (FPMAC)", 2013 Symp on Computer Arithmetic (paper).

【0283】 Srinivasan, Bhudiya, Ramanarayanan, Babu, Jacob, Mathew, Krishnamurthy, Erraguntla, "Split-path Fused Floating Point Multiply Accumulate (FPMAC)", 2014 Symp on Computer Arithmetic, Austin TX, (slides from www.arithsymposium.org).

【0284】 Srinivasan, Bhudiya, Ramanarayanan, Babu, Jacob, Mathew, Krishnamurthy, Erraguntla, United States Patent 8,577,948 (B2), Nov 5, 2013.

【0285】 Quach, Flynn, "Suggestions For Implementing A Fast IEEE Multiply-Add-Fused Instruction", (Stanford) Technical Report CSL-TR-91-483 July, 1991.

【0286】 Seidel, "Multiple Path IEEE Floating-Point Fused Multiply-Add", IEEE 2004.

【0287】 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.

【0288】 Paidimarri, Cevrero, Brisk, Ienne, "FPGA

Implementation of a Single-Precision Floating-Point Multiply-Accumulator with Single-Cycle Accumulation", Pub TBD.

【0289】 Henry, Elliott, Parks, "X87 Fused Multiply-Add Instruction", United States Patent 7,917,568 (B2), Mar 29, 2011.

【0290】 Walaa Abd El Aziz Ibrahim, "Binary Floating Point Fused Multiply Add Unit", Thesis Submitted to Cairo University, Giza, Egypt, 2012 (retr from Google).

【0291】 Quinell, "Floating-Point Fused Multiply-Add Architectures", Dissertation Presented to Univ Texas at Austin, May 2007, (retr from Google).

【0292】 Author Unknown, "AMD Athlon Processor Floating Point Capability", AMD White Paper Aug 28, 2000.

【0293】 Cornea, Harrison, Tang, "Intel Itanium Floating-Point Architecture" Pub TBD.

【0294】 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.

【0295】 Wait, "IBM PowerPC 440 FPU with complex-arithmetic extensions", IBM Jour Res & Dev Vol 49 No 2/3 March, May 2005.

【0296】 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 2/3 March, May 2005.

【0297】 藉由以上具體實施例之詳述，係希望能更加清楚描述本發明之特徵與精神，而並非以上述所揭露的具體實施例來對

本發明之範疇加以限制。相反地，其目的是希望能涵蓋各種改變及具相等性的安排於本發明所欲申請之專利範圍的範疇內。

【符號說明】

【0298】

2	微指令
15	指令快取
20	指令轉譯器
24	指令管線
25	重命名單元與保留站
30	重排緩衝器
35	架構暫存器
40	轉送匯流排
45	調整後的乘法單元
50	調整後的加法單元
55	捨入快取
60	其他執行單元
105	乘數運算元
110	被乘數運算元
115	相加數運算元
120	乘法加總陣列
125	最終加法器
130	標準化移位器
135	前導位預測與編碼器
140	輸入運算元分析器

145	未經捨入之標準化加總結果
146	結果匯流排
148	資料路徑
152	運算元匯流排
155	重命名暫存器
160	運算元調整器
165	近路徑
170	遠路徑
175	捨入位元選擇邏輯
180	捨入模組
195	輸入埠
200	解碼器
215	路徑控制邏輯
220	相加數對準與射入邏輯
235	乘法陣列
240	部分乘積加法器
265	PNExp 產生器
270	IRExp 產生器
275	不足位/溢位偵測器
280	中間符號產生器
285	結果向量埠
290	端迴進位指標產生器
295	黏位元產生器
300	捨入位元產生器
305	捨入位元埠

310	輸入埠
315	解碼器
330	對準與條件邏輯
335	指數產生器
340	相加模組
345	溢位移位邏輯
350	加法器捨入位元產生器
365	符號產生器
538	指令操作碼
544、562	被乘數
547、565	乘數
541、577	目標欄
550、568、583	相加數
556、574	微指令操作碼
559	暫時暫存器
580	乘積

I650652

發明摘要

※ 申請案號：104121548

※ 申請日： 104/07/02 ※IPC 分類： G06F 17/16 (2006.01)
G06F 7/485 (2006.01)
G06F 7/487 (2006.01)

【發明名稱】(中文/英文)

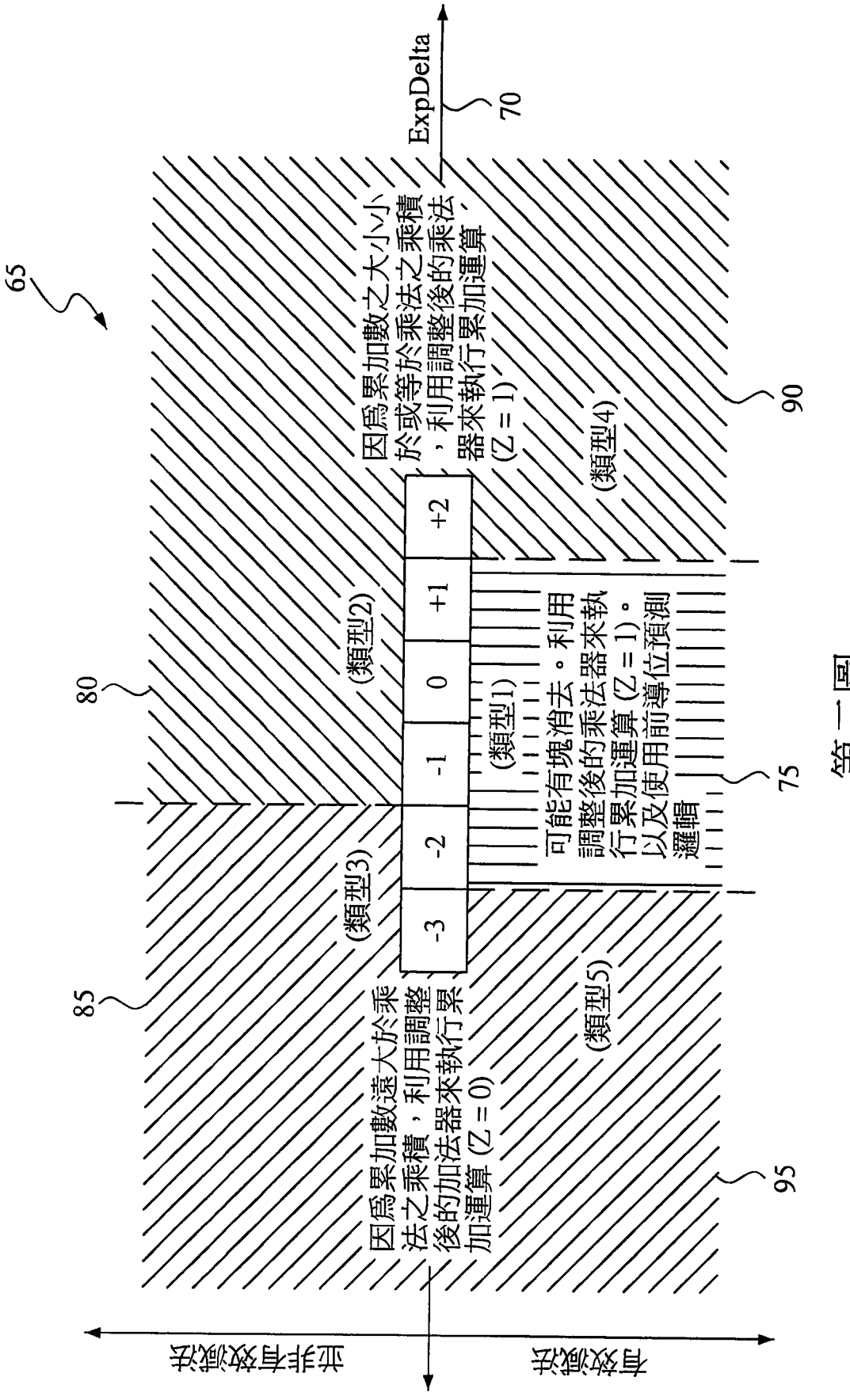
運算控制指標快取/ CALCULATION CONTROL INDICATOR CACHE

【中文】

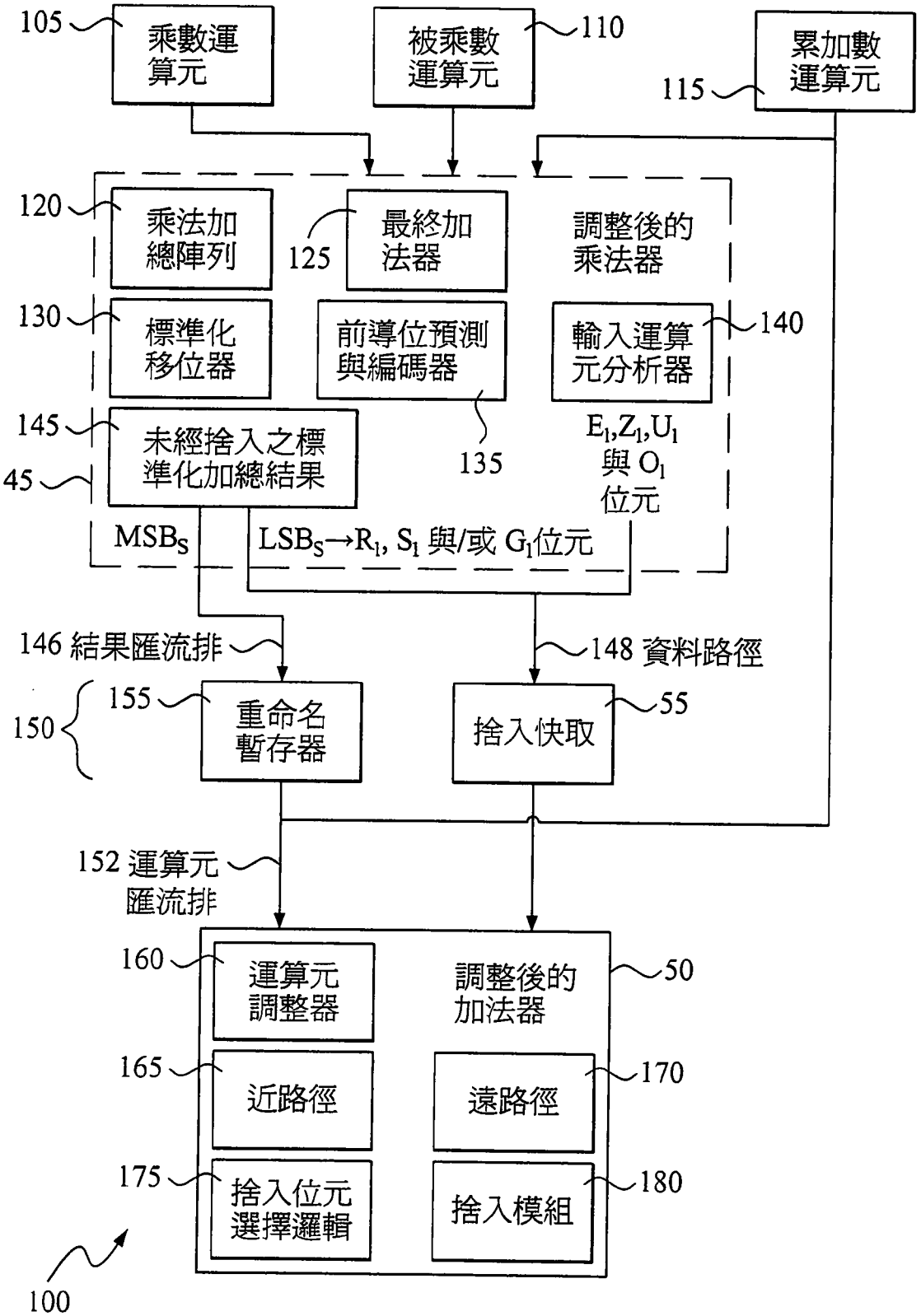
一種算術運算係使用一第一指令執行單元去產生一中間結果向量與複數個運算控制指標；其中，複數個運算控制指標係指明自中間結果向量產生一最終結果之接續運算該如何進行；中間結果向量與複數個運算控制指標係儲存在一設於該指令執行單元外之記憶體中，且隨後由一第二指令執行單元讀取、已完成算術運算。

【英文】

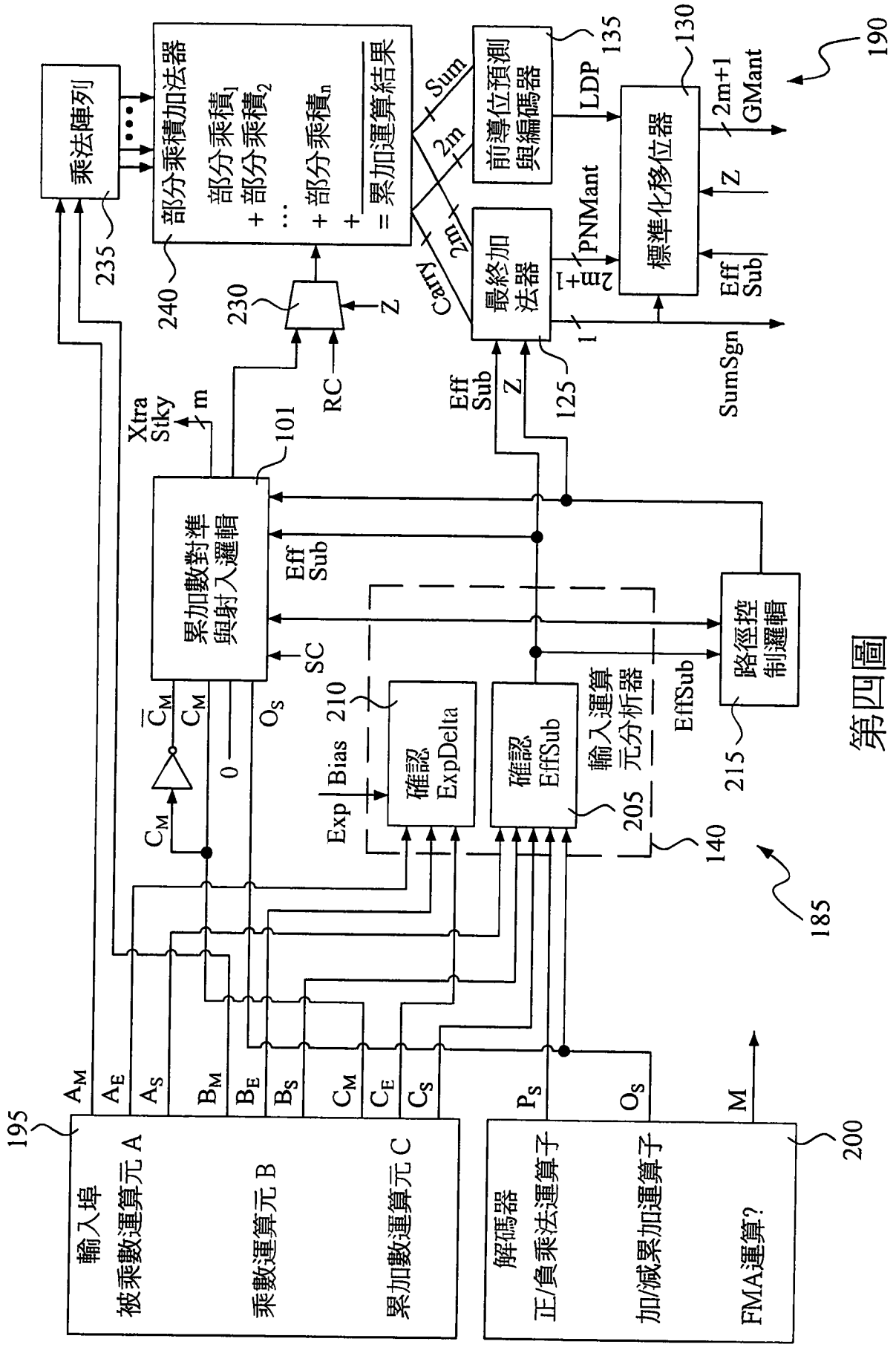
An arithmetic operation is performed using a first instruction execution unit to generate an intermediate result vector and a plurality of calculation control indicators that indicate how subsequent calculations to generate a final result from the intermediate result vector should proceed. The intermediate result vector and the plurality of calculation control indicators are stored in memory external to the instruction execution unit, and later read by a second instruction execution unit to complete the arithmetic operation.


65
ExpDelta
70
80
85
90
95
75
-3
-2
-1
0
+1
+2
(類型1)
(類型2)
(類型3)
(類型4)
(類型5)
因爲累加數之大小小於或等於乘法之乘積，利用調整後的乘法器來執行累加運算 (Z = 1)
因爲累加數遠大於乘法之乘積，利用調整後的加法器來執行累加運算 (Z = 0)
可能有塊消去。利用調整後的乘法器來執行累加運算 (Z = 1)。以及使用前導位預測邏輯
非常有效率
有效率
非常有效率

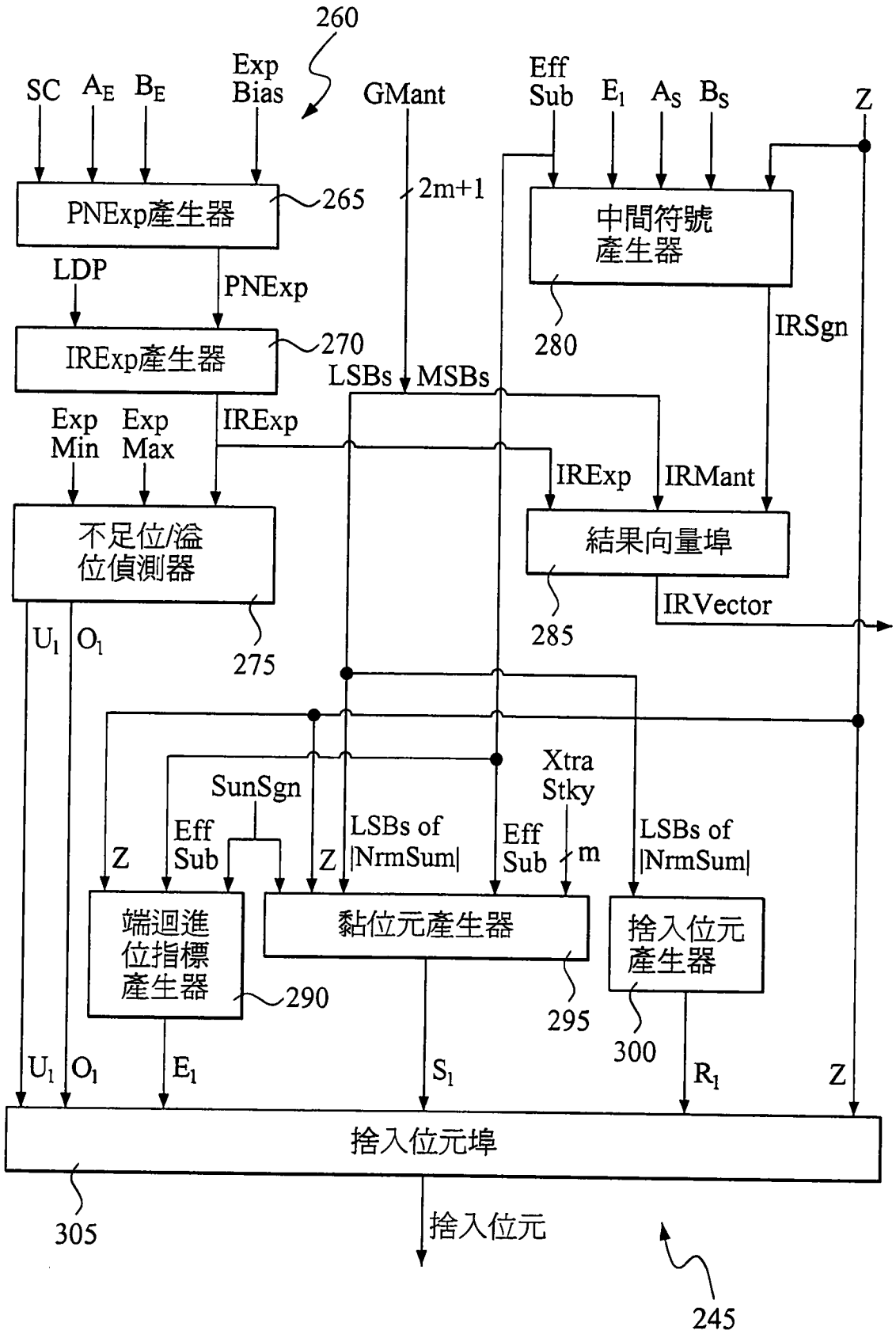
第二圖


105
乘數運算元
被乘數運算元
110
累加數運算元
115
120
乘法加總陣列
125
最終加法器
調整後的乘法器
130
標準化移位器
前導位預測與編碼器
輸入運算元分析器
140
145
未經捨入之標準化加總結果
45
135
E_l, Z_l, U_l 與 O_l 位元
MSB_S
$LSB_S \rightarrow R_l$, S_l 與/或 G_l位元
146 結果匯流排
148 資料路徑
150
155
重命名暫存器
捨入快取
55
152 運算元匯流排
160
運算元調整器
調整後的加法器
50
165
近路徑
遠路徑
170
175
捨入位元選擇邏輯
捨入模組
180
100

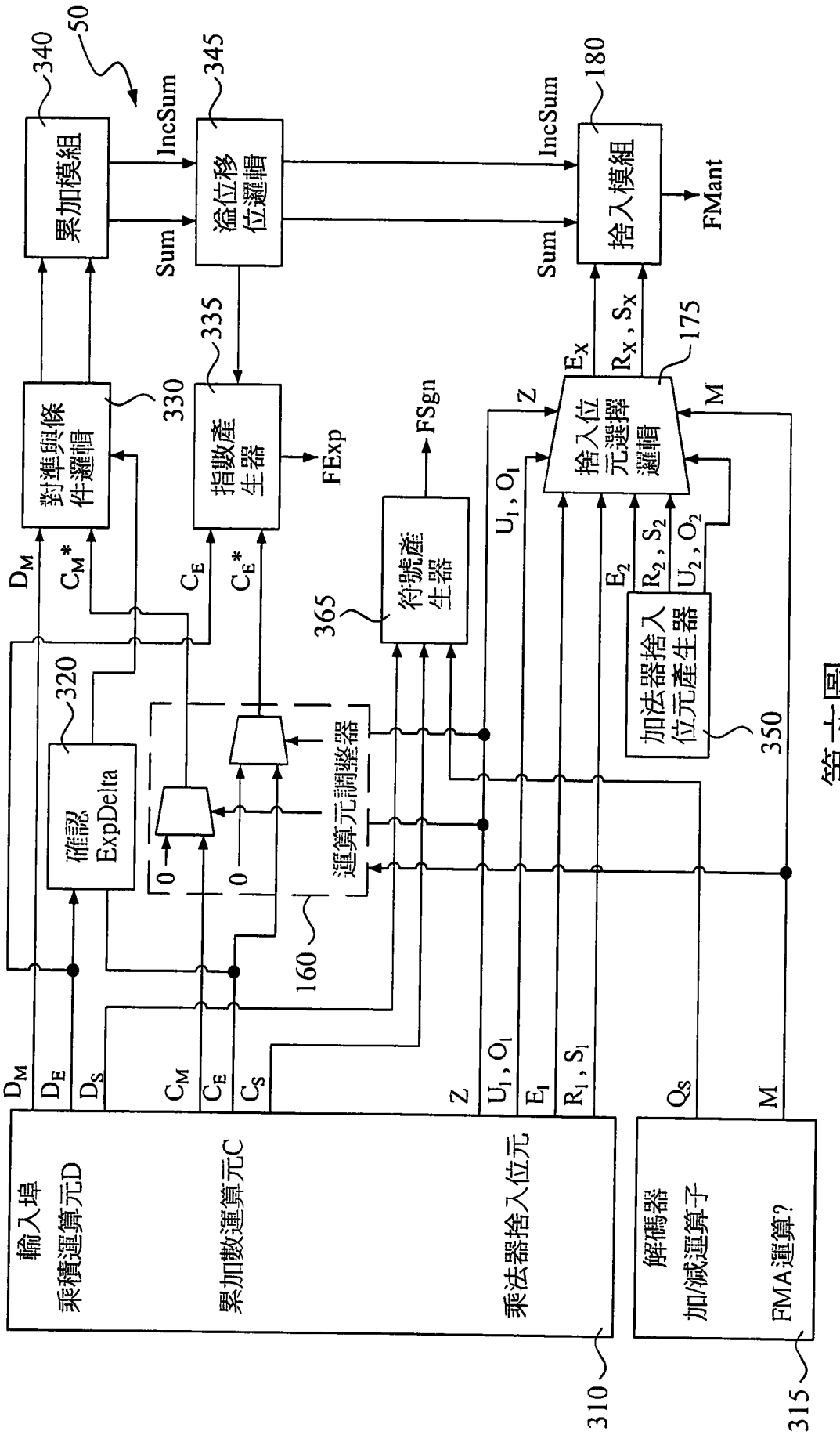
第三圖


輸入埠
被乘數運算元 A
乘數運算元 B
累加數運算元 C
解碼器
正/負乘法運算子
加/減累加運算子
FMA運算?
乘法陣列
部分乘積加法器
部分乘積1
+ 部分乘積2
...
+ 部分乘積n
+
= 累加運算結果
累加數對準與射入邏輯
確認 ExpDelta
確認 EffSub
輸入運算元分析器
路徑控制邏輯
前導位預測與編碼器
最終加法器
標準化移位器
Sum
Carry
PNMant
LDP
Z
Eff Sub
SumSgn
2m+1 GMant
Xtra Sticky
RC
Exp
Bias
SC
EffSub
195
200
185
140
215
210
205
235
240
230
101
125
135
130
190

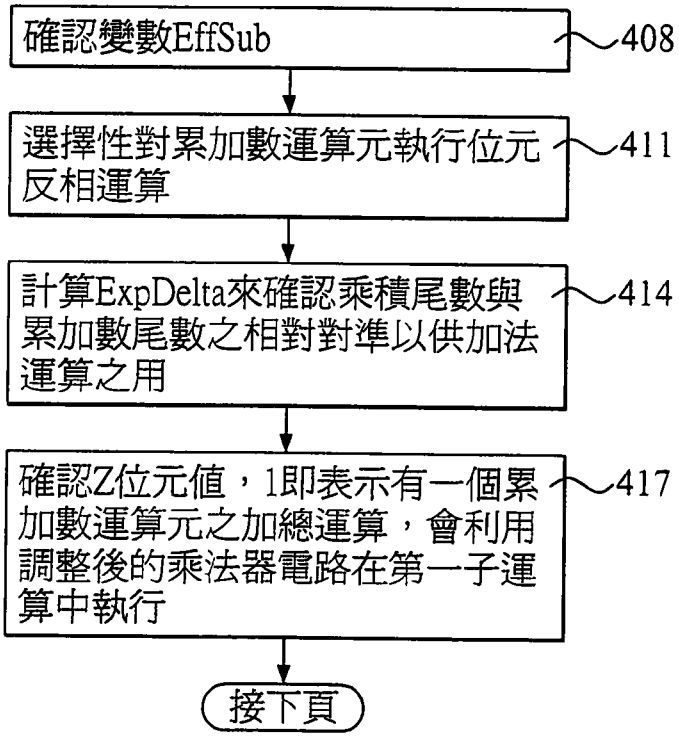
第四圖


260
SC
A_E
B_E
Exp Bias
GMant
Eff Sub
E_I
A_S
B_S
Z
PNExp產生器
265
2m+1
中間符號產生器
LDP
PNExp
IRExp產生器
270
280
IRSgn
LSBs
MSBs
Exp Min
Exp Max
IRExp
IRExp
IRMant
不足位/溢位偵測器
結果向量埠
285
IRVector
U_I
O_I
275
SunSgn
Xtra Stky
Eff Sub
Z
LSBs of |NrmSum|
Z
Eff Sub
m
LSBs of |NrmSum|
端迴進位指標產生器
290
黏位元產生器
295
捨入位元產生器
300
U_I
O_I
E_I
S_I
R_I
Z
捨入位元埠
305
捨入位元
245

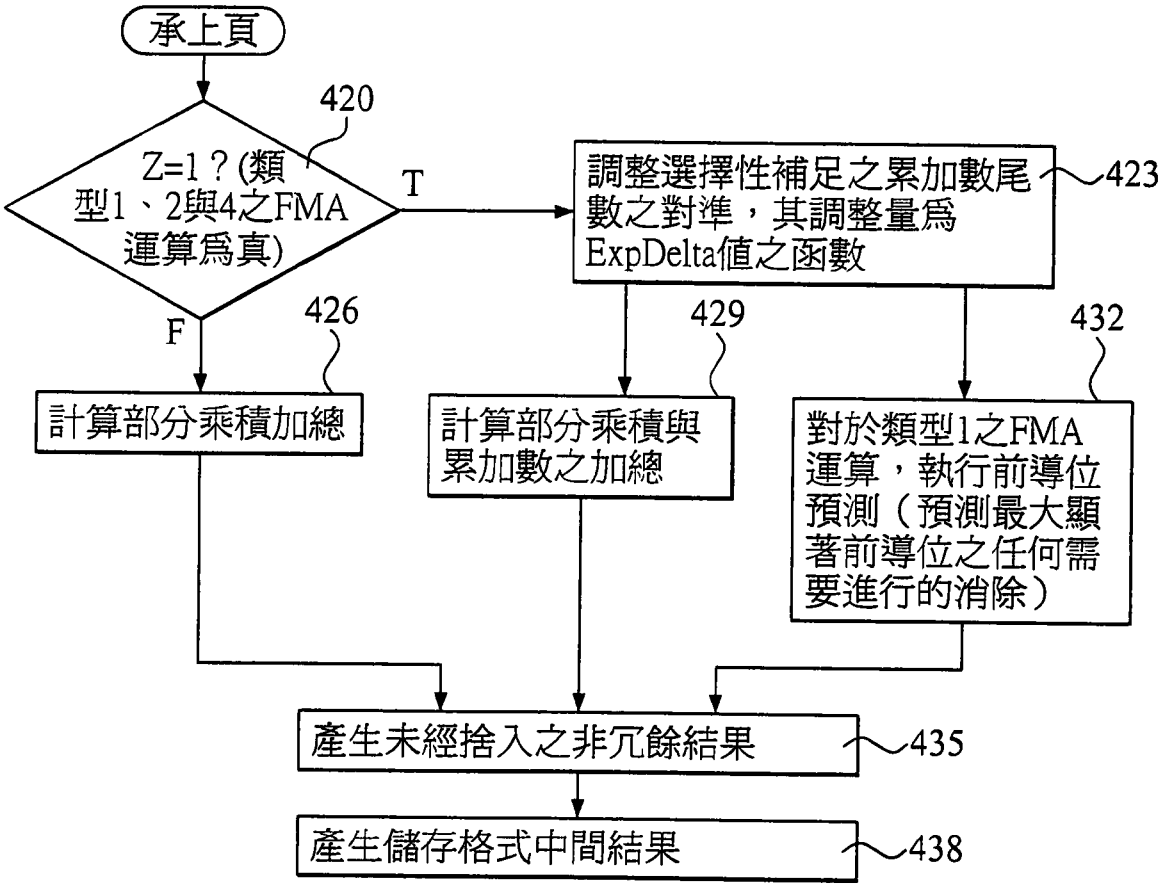
第五圖


輸入埠
乘積運算元D
累加數運算元C
乘法器捨入位元
解碼器
加/減運算子
FMA運算?
310
315
D_M
D_E
D_S
C_M
C_E
C_S
Z
U_1, O_1
E_1
R_1, S_1
Q_S
M
確認 ExpDelta
320
160
運算元調整器
0
0
365
符號產生器
FSgn
對準與條件邏輯
330
C_M^*
C_E^*
指數產生器
335
FExp
累加模組
340
50
IncSum
Sum
溢位移位邏輯
345
捨入位元選擇邏輯
E_X
R_X, S_X
175
M
加法器捨入位元產生器
E_2
R_2, S_2
U_2, O_2
350
捨入模組
180
IncSum
Sum
FMant

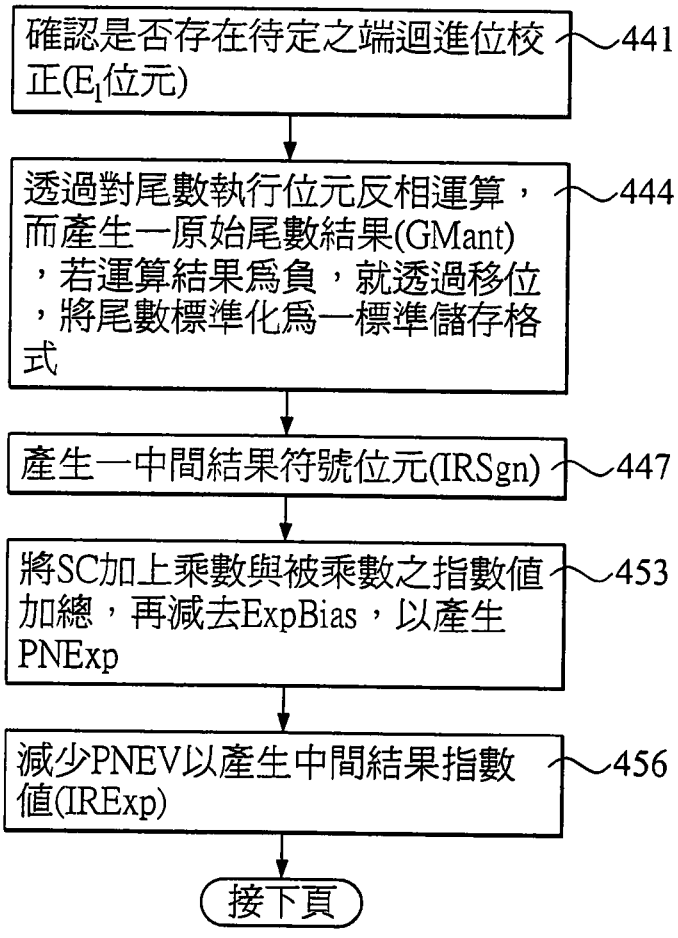
第六圖


確認變數EffSub
408
選擇性對累加數運算元執行位元反相運算
411
計算ExpDelta來確認乘積尾數與累加數尾數之相對對準以供加法運算之用
414
確認Z位元值，1即表示有一個累加數運算元之加總運算，會利用調整後的乘法器電路在第一子運算中執行
417
接下頁

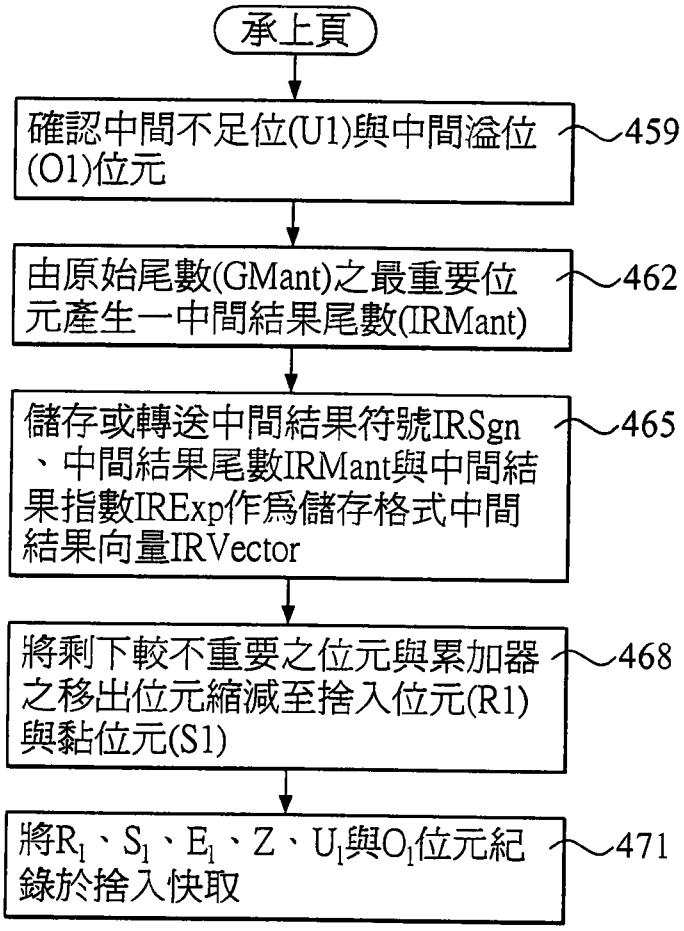
第七圖


承上頁
420
Z=1？(類型1、2與4之FMA運算爲真)
T
F
調整選擇性補足之累加數尾數之對準，其調整量爲ExpDelta值之函數
423
426
計算部分乘積加總
429
計算部分乘積與累加數之加總
432
對於類型1之FMA運算，執行前導位預測（預測最大顯著前導位之任何需要進行的消除）
產生未經捨入之非冗餘結果
435
產生儲存格式中間結果
438

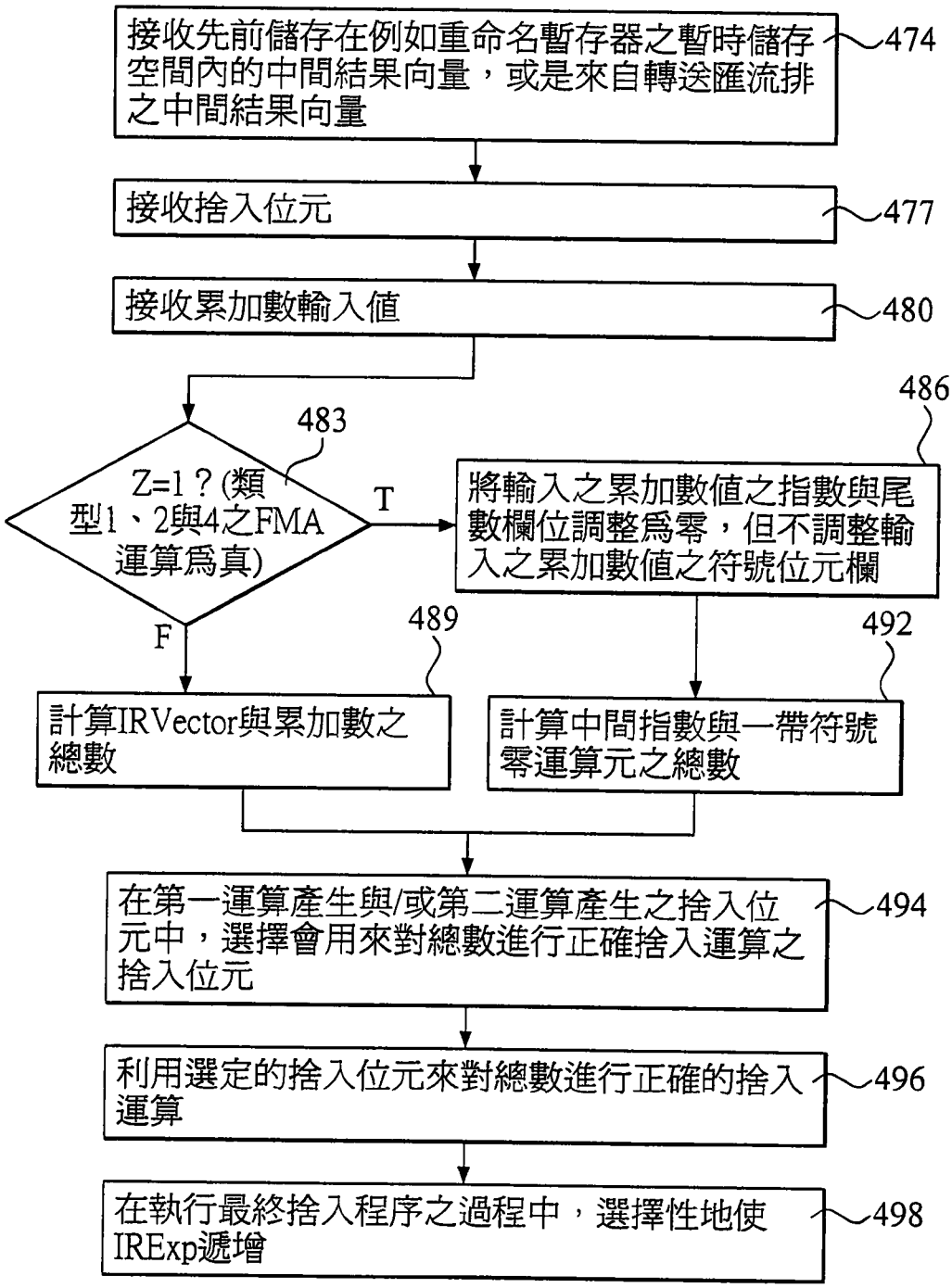
第八圖


確認是否存在待定之端迴進位校正(E_1位元)
441
透過對尾數執行位元反相運算，而產生一原始尾數結果(GMant)，若運算結果爲負，就透過移位，將尾數標準化爲一標準儲存格式
444
產生一中間結果符號位元(IRSgn)
447
將SC加上乘數與被乘數之指數值加總，再減去ExpBias，以產生PNExp
453
減少PNEV以產生中間結果指數值(IRExp)
456
接下頁

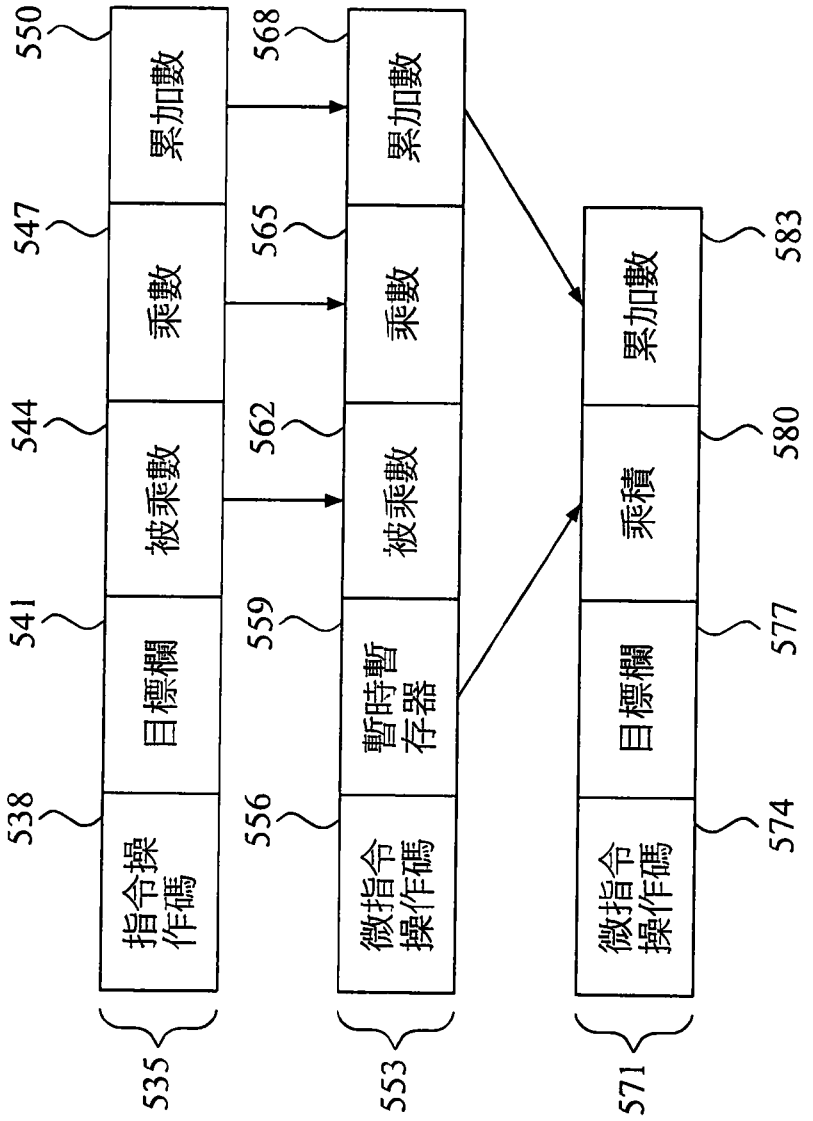
第九A圖


承上頁
確認中間不足位(U1)與中間溢位(O1)位元
459
由原始尾數(GMant)之最重要位元產生一中間結果尾數(IRMant)
462
儲存或轉送中間結果符號IRSgn、中間結果尾數IRMant與中間結果指數IRExp作爲儲存格式中間結果向量IRVector
465
將剩下較不重要之位元與累加器之移出位元縮減至捨入位元(R1)與黏位元(S1)
468
將R_1、S_1、E_1、Z、U_1與O_1位元紀錄於捨入快取
471

第九B圖


接收先前儲存在例如重命名暫存器之暫時儲存空間內的中間結果向量，或是來自轉送匯流排之中間結果向量
474
接收捨入位元
477
接收累加數輸入值
480
483
Z=1？(類型1、2與4之FMA運算爲真)
T
F
486
將輸入之累加數值之指數與尾數欄位調整爲零，但不調整輸入之累加數值之符號位元欄
489
計算IRVector與累加數之總數
492
計算中間指數與一帶符號零運算元之總數
在第一運算產生與/或第二運算產生之捨入位元中，選擇會用來對總數進行正確捨入運算之捨入位元
494
利用選定的捨入位元來對總數進行正確的捨入運算
496
在執行最終捨入程序之過程中，選擇性地使IRExp遞增
498

第十圖


指令操作碼
目標欄
被乘數
乘數
累加數
微指令操作碼
暫時暫存器
被乘數
乘數
累加數
微指令操作碼
目標欄
乘積
累加數
535
538
541
544
547
550
553
556
559
562
565
568
571
574
577
580
583

第十一圖

【代表圖】

【本案指定代表圖】：第一圖。

【本代表圖之符號簡單說明】：

2	微指令
10	微處理器
15	指令快取
20	指令轉譯器
24	指令管線
25	重命名單元與保留站
30	重排緩衝器
35	架構暫存器
40	轉送匯流排
45	調整後的乘法單元
50	調整後的加法單元
55	捨入快取
60	其他執行單元

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

申請專利範圍

1. 一種微處理器中之方法，用以執行一算術運算，該方法係包括：
使用一指令執行單元產生一中間結果向量與複數個運算控制指標，該複數個運算控制指標係指明自該中間結果向量產生一最終結果之接續運算該如何進行；以及
將該中間結果向量與該複數個運算控制指標儲存至一設於該指令執行單元外之記憶體中。

2. 如申請專利範圍第 1 項所述之方法，更包括將該中間結果向量與該複數個運算控制指標自該記憶體下載；並於該中間結果向量上，依據該運算控制指標執行運算，以產生一最終結果。

3. 如申請專利範圍第 1 項所述之方法，其中，該算術運算為一融合浮點乘積-相加運算，其運算元包括一被乘數、一乘數及一相加數，且其中該中間結果向量係至少為該被乘數與該乘數之部分乘積之總和。

4. 如申請專利範圍第 1 項所述之方法，其中，該算術運算為一牽涉至少一乘法與至少一相加運算之融合運算。

5. 如申請專利範圍第 1 項所述之方法，其中，該算術運算為一牽涉二或更多算數運算子之複合算術運算。

6. 如申請專利範圍第 5 項所述之方法，更包括將該複合算術運算

分路為一使用一第一算術運算元之第一算術運算、以及一使用一第二算術運算元之第二算術運算。

7. 如申請專利範圍第 6 項所述之方法，其中，該運算控制指標係指明該第二算術運算該如何進行。

8. 如申請專利範圍第 5 項所述之方法，其中，該複合算術運算為一接續算術運算。

9. 如申請專利範圍第 5 項所述之方法，其中，該運算控制指標係提供關於多少該複合算術運算已完成產生該中間結果向量之資訊。

10. 如申請專利範圍第 5 項所述之方法，其中，該運算控制指標係提供關於該第一算術運算是否會導致不足位與溢位狀況之資訊。

11. 如申請專利範圍第 5 項所述之方法，更包括將該中間結果向量以一儲存空間格式儲存，當考慮獨立於該運算控制指標外時，該儲存空間格式提供較所需為少之位元數，以連貫產生該複合算術運算一結果之一代表，其中之該複合算術運算係與該結果無明顯差異，該結果係可以該複合算術運算之一非限定精確運算產生，可大幅縮減目標資料大小；但，其中與該複數個運算控制指標連結之該儲存空間格式係可提供充足之資訊，以連貫產生該複合算術運算之一結果，該結果係可以該複合算術運算

之一非限定精確運算產生，可大幅縮減目標資料大小。

12. 一種微處理器中之方法，用以執行一捨入運算，該方法係包括：複數個指令執行單元之一第一指令執行單元產生一未捨入結果；

將至少一捨入指標儲存入一設於該複數個指令執行單元外之捨入快取；

自該捨入快取提供該未捨入結果與該至少一 捨入指標至該複數個指令執行單元之一第二指令執行單元中；以及

該第二指令執行單元利用至少該未捨入結果與該至少一捨入指標產生一最終捨入結果。

13. 如申請專利範圍第 12 項所述之方法，更包括將該未捨入結果存入一與該捨入快取不同之通用儲存空間。

14. 如申請專利範圍第 12 項所述之方法，更包括將該一捨入指標透過一資料路徑，自該第一指令單元轉換至該捨入快取，該資料路徑係與一結果匯流排分離，並將該複數個指令執行單元耦接至一通用儲存空間。

15. 一種微處理器中之方法，用以執行一算術運算，該方法係包括：

使用一第一指令執行單元產生一中間結果向量；

共存地產生複數個運算控制指標，該複數個運算控制指標係指明自該中間結果向量產生一最終結果之接續運算該如何進行；

將該中間結果向量與該複數個運算控制指標傳送至一第二指令執行單元；以及

使用該第二指令執行單元產生該最終結果，並依據該運算控制指標完成該算術運算。

16. 如申請專利範圍第 15 項所述之方法，其中，該算術運算為一複合算術運算。

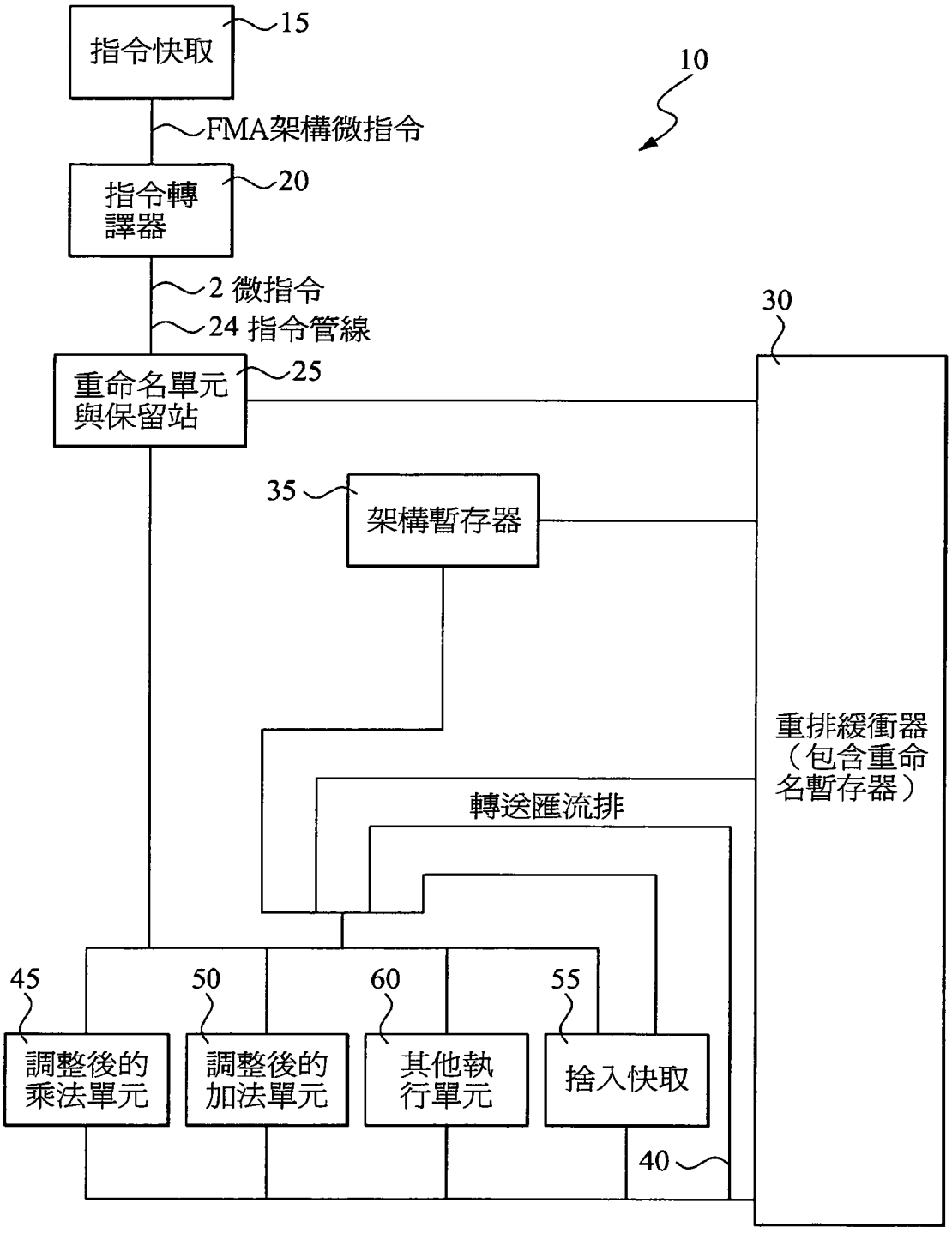
17. 如申請專利範圍第 16 項所述之方法，其中，該複合算術運算係屬一種形態，該形態只允許單一之捨入運算產生該最終 結果。

18. 如申請專利範圍第 15 項所述之方法，其中，該算術運算為一融合乘積-相加運算。

19. 如申請專利範圍第 18 項所述之方法，其中，該中間結果向量為一該乘積-相加運算一部分之未捨入結果，且該運算控制指標係包括 用以產生該乘積-相加運算一最終捨入結果之捨入指標。

20. 如申請專利範圍第 15 項所述之方法，其中，該中間結果向量之傳送完成係藉由一結果匯流排，而該運算控制字元之傳送完成則是藉由一與該結果匯流排不同之資料路徑。

圖式


指令快取
15
10
FMA架構微指令
指令轉譯器
20
2 微指令
24 指令管線
30
重命名單元與保留站
25
35
架構暫存器
重排緩衝器（包含重命名暫存器）
轉送匯流排
45
50
60
55
調整後的乘法單元
調整後的加法單元
其他執行單元
捨入快取
40

第一圖