(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201229898 A1
(43)公開日： 中華民國 101 (2012) 年 07 月 16 日

(21)申請案號：100145974 (22)申請日： 中華民國 100 (2011) 年 12 月 13 日
(51)Int. Cl.： *G06F9/38 (2006.01)*
(30)優先權：2010/12/21 美國 12/974,157
(71)申請人：英特爾公司 (美國) INTEL CORPORATION (US)
美國
(72)發明人：史麥揚斯基 麥克海爾 SMELYANSKIY, MIKHAIL (US)；陳彥廣 CHEN, YEN-KUANG (US)；金大峴 KIM, DAEHYUN (KR)；休吉斯 克里斯多福 J HUGHES, CHRISTOPHER J. (US)；李 維多 W LEE, VICTOR W. (US)
(74)代理人：惲軼群；陳文郎
申請實體審查：無 申請專利範圍項數：20 項 圖式數：5 共 53 頁

(54)名稱

用於利用單指令多資料作衝突檢測之機構

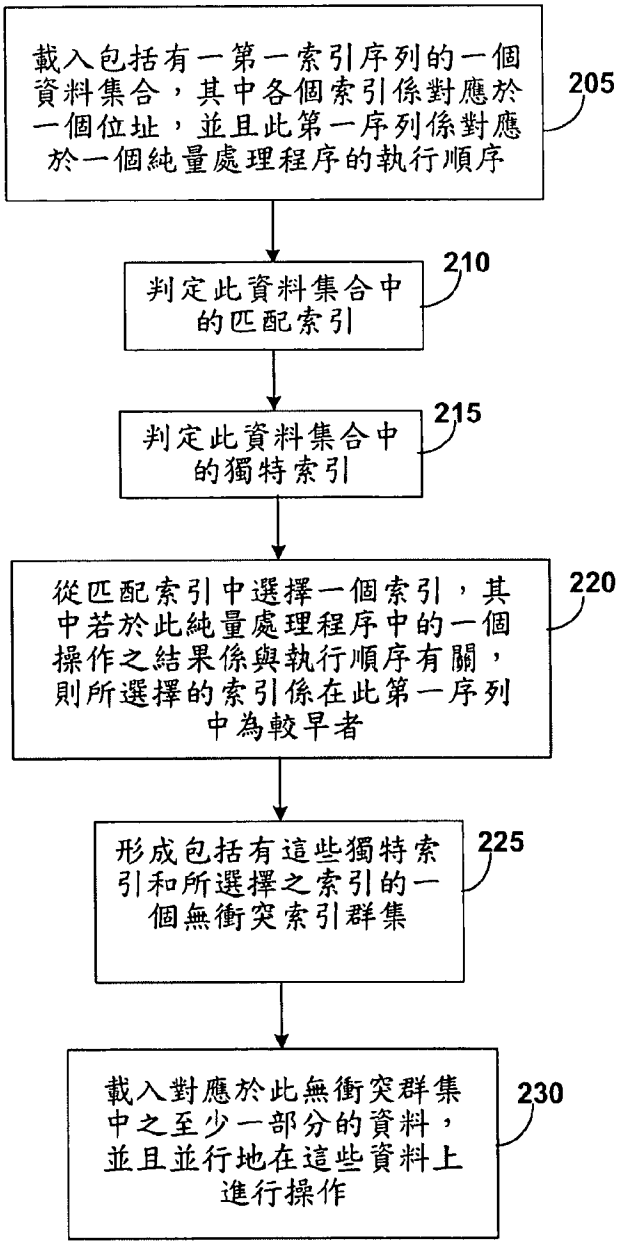
MECHANISM FOR CONFLICT DETECTION USING SIMD

(57)摘要

一種系統和方法係設計來在將純量處理程序轉換成並行處理程序(「SIMD 化」)時檢測衝突。係可針對一個無序單一索引、一個有序單一索引、和/或有序索引對而檢測衝突。可更進一步針對寫後讀關係來檢測衝突。衝突檢測係設計來識別出在一連串迭代中的不可以被並行從事的操作(即，迭代)。

205 ～ 230：操作

200



(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201229898 A1
(43)公開日： 中華民國 101 (2012) 年 07 月 16 日

(21)申請案號：100145974 (22)申請日： 中華民國 100 (2011) 年 12 月 13 日
(51)Int. Cl.： *G06F9/38 (2006.01)*
(30)優先權：2010/12/21 美國 12/974,157
(71)申請人：英特爾公司 (美國) INTEL CORPORATION (US)
美國
(72)發明人：史麥揚斯基 麥克海爾 SMELYANSKIY, MIKHAIL (US)；陳彥廣 CHEN, YEN-KUANG (US)；金大峴 KIM, DAEHYUN (KR)；休吉斯 克里斯多福 J HUGHES, CHRISTOPHER J. (US)；李 維多 W LEE, VICTOR W. (US)
(74)代理人：惲軼群；陳文郎
申請實體審查：無 申請專利範圍項數：20 項 圖式數：5 共 53 頁

(54)名稱

用於利用單指令多資料作衝突檢測之機構

MECHANISM FOR CONFLICT DETECTION USING SIMD

(57)摘要

一種系統和方法係設計來在將純量處理程序轉換成並行處理程序(「SIMD 化」)時檢測衝突。係可針對一個無序單一索引、一個有序單一索引、和/或有序索引對而檢測衝突。可更進一步針對寫後讀關係來檢測衝突。衝突檢測係設計來識別出在一連串迭代中的不可以被並行從事的操作(即，迭代)。

六、發明說明：

【發明所屬之技術領域】

發明領域

本揭露內容係有關在單指令多資料（single-instruction multiple-data, SIMD）中檢測衝突之技術。

【先前技術】

發明背景

許多應用程式具有大量的資料等級平行性，且應能夠從單指令多資料（SIMD）支援中獲益。在SIMD執行中，單一個指令係同時操作在複數個資料元素上。這典型上係藉由延伸各種資源，例如暫存器和算術邏輯單元(arithmetic logic unit, ALU)，的寬度來實施，使這些資源能夠分別保持或是操作在複數個資料元素上。然而，大部分這樣的應用程式一開始便已被設計成一次處理一個指令和一個資料元素的純量處理程序，即，單指令單資料(single-instruction single-data, SISD)。

將純量處理程序轉換成SIMD處理程序（即，「SIMD化」）可提供操作性改良，例如縮短處理時間。然而，在這樣的轉換當中的一項考量，是得在必要時確保純量程式順序有被保留。另一項考量，是得確保在資料被分散到記憶體中時，所導致的記憶體位址之向量僅包括獨特位址（亦即，沒有任何重複的位址）。因此，衝突檢測可有助於這樣的轉換。

【發明內容】

發明概要

依據本發明之一實施例，係特地提出一種用於衝突檢測之方法，其包含下列步驟：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；以及載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

依據本發明之另一實施例，係特地提出一種系統，其包含具有個別地或以組合形式儲存在其上之指令的一或多個儲存媒體，該等指令在被一或多個處理器執行時會致使包含下列之操作：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引

和所選擇的該索引；以及載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

依據本發明之又一實施例，係特地提出一種組配來檢測衝突的設備，該設備包含：一個記憶體；以及一個處理器，其包含一個純量處理單元和一個向量處理單元，該處理器係組配來進行下列步驟：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於該記憶體中的一個位址，並且該第一序列係對應於在使用該純量處理單元時的一個純量處理程序之執行順序；載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於該記憶體中的一個位址，並且該第一序列係對應於在使用該純量處理單元時的一個純量處理程序之執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；及載入對應於該無衝突群集之至少一部分的資料，並利用該向量處理單元並行地在該等資料上進行該操作。

圖式簡單說明

隨著下面的詳細說明之推進，及透過對圖式之參考，所請求之標的之實施例的特徵和優點將會變得明顯可見，在這些圖式中，類似號碼係描述類似部件，並且其中：

第1圖例示出與本揭露內容一致的一個示範系統實施例；

第2圖例示出用於與本揭露內容一致的衝突檢測之示範操作的流程圖；

第3圖例示出用於與本揭露內容一致的單一索引衝突檢測之示範操作的流程圖；

第4圖例示出用於與本揭露內容一致的單一索引衝突檢測之示範操作的流程圖；並且

第5圖例示出用於與本揭露內容一致的有序索引對衝突檢測之示範操作的流程圖。

雖然下面的詳細說明將會參考數個例示性實施例而推進，但對於熟於此技者來說，對於這些實施例的許多替代體、修改體和變異體會是明顯可見的。

【實施方式】

較佳實施例之詳細說明

大體而言，本揭露內容係說明設計來在將純量處理程序轉換至並行處理程序（「SIMD化」）時檢測衝突的系統和方法，例如，以開拓在一些電腦架構中可用的向量處理器。例如，可從本申請案之受讓人處取得的6核Core i7 980處理器包括被設計來執行一個向量指令以處理資料之向量的至少一個向量處理器。將一個純量處理程序轉換成一個向量處理程序典型上會導致較短的並行操作序列。各個並行操作係對應於某個數量的純量操作，其中此數量可係對應於進行這些操作的系統之向量長度。

係可針對一個無序單一索引、一個有序單一索引、和/或數個有序索引對而檢測衝突。可進一步針對寫後讀關係而檢測衝突。衝突檢測係設計來識別在一個迭代序列中之不可並行完成的操作（即，迭代）。

當於本文中使用時，「索引」係對應於被包括在一個陣列中的一個元素之位址。例如，除了別的以外，此陣列可係由一個基底位址指明。此陣列中的一個元素可由索引指向。因而，此元素之位址可係包括基底位址和索引（例如，偏移）。

當SIMD化一個純量處理程序時，此處理器要在上頭進行操作的資料可係儲存在一個陣列中，並且此純量處理程序可係設計來操作於此陣列的一個元素上，一次一個元素（SISD，單一個指令，單一個資料）。例如，此純量處理程序可包括一個迴圈，此迴圈中之各遍（即，迭代）都會在一個元素上操作。視處理程序而定，係可在一個元素上操作一次、複數次、或不在其上頭做操作。

例如，一個純量處理程序可包括一個for迴圈，例如，

```
for(i=0; i<N; i++)
    A[Index[i]]++ ;
```

此迴圈係設計成在此迴圈中之各遍對陣列A中的一個元素增量。於此範例中，Index是對應於陣列A中之元素之位址的一個索引陣列。*i*對應於此for迴圈的一個迭代，並且是對陣列Index的一個索引。因此，*i*對應於在陣列Index中的一個元素之位址，並且Index[*i*]對應於在陣列A中的一個元素之位

址。易言之，於陣列Index中的元素是對A陣列的索引，對應於陣列A中之元素之位址。在陣列Index中所含有的這些索引可不為獨特的，這指出在陣列A中的一個元素在此for迴圈中係被取用複數次。依據於本文中所呈現之教示，衝突檢測是設計來檢測在一個元素上操作複數次和/或在此純量處理程序之至少一部分中的排序關係這樣的情況。陣列A的被操作複數次的一個元素在陣列Index中係對應於具有相同數值，亦即，指向陣列A中之相同元素，的複數個元素（索引）。

與本揭露內容一致的一種方法係設計來檢測可能會透過讀取和/或寫入至直到運行時間之前皆未知的記憶體位置而出現的資料關係。這樣的關係可藉由檢測一個資料集合中的匹配索引來檢測。例如，可對在資料關係中可能涉入的所有的索引對作比較。在另一個範例中，係可進行一項操作，此項操作之結果係取決於在此資料集合中是否有匹配索引。在這第二個範例中，操作的結果可接著被用來識別此資料集合中的匹配索引。此種操作之範例包括但不受限於向量混洗操作及分散和/或收集操作。

是否有衝突和衝突之類型係可取決於操作和/或與此（等）操作相關的資料。衝突檢測大體上包括針對VLEN個記憶體位址的一個向量而判定一個位址獨特子集，其中VLEN是向量長度。衝突檢測可係以程式順序進行，或可係無序進行。對於交換性的操作，例如加法，而言，衝突檢測可為無序的，亦即，可以是也可以不是以程式順序進行。易言之，進行這些操作的順序並不會改變結果。因此，衝

突被檢測的順序並不取決於這些操作在純量處理中被進行的順序。對於非交換性的操作，例如除法，而言，衝突檢測可係以程式順序進行。易言之，進行這些操作的順序可能會改變結果。因此，衝突被檢測的順序係對應於操作在純量處理中被進行的順序。

例如，直方圖操作典型上包括在一個資料陣列上之操作。這可係實施在一個純量處理器中，作為在此陣列中之各個元素（或某個元素子集）上進行此處理的一個迴圈。例如，下面的偽碼（範例1）：

```
for(i=0; i<N; i++)
    hist[Index[i]%HIST_SIZE]++ ;
```

係設計成在此for迴圈中之各遍對陣列hist中的一個元素增量達N遍。在陣列hist中的元素係由Index[i]%HIST_SIZE指出，其中%是模數運算子（亦即，Index[i]%HIST_SIZE對應於Index[i]除以HIST_SIZE之餘數）。易言之，在hist中的元素之位址對應於Index[i]%HIST_SIZE。將範例1映對到SIMD係包括判定在索引向量Index[i]中的一個獨特元素（即，位址）子集。對於範例1來說，此獨特元素子集可係以任意順序判定，因為增量操作是交換性的，而且只有一個索引。易言之，陣列hist陣列中之元素被增量的順序並不影響結果。

在另一個範例（範例2）中，下面的偽碼：

```
for(i=0; i<N; i++)
    hist[Index[i]%HIST_SIZE] = 1/( hist[Index[i]%HIST_SIZE] + i) ;
```

係設計成在此for迴圈中之上至N遍的各遍中對陣列hist中的一個元素作更新。在陣列hist中的元素係由Index[i]%HIST_SIZE指出。此更新包括將1除以此陣列中之元素加上索引i。類似於範例1，將範例2映對到SIMD係包括判定在索引向量Index[i]中的一個獨特元素（即，位址）子集。對於範例2來說，此獨特元素子集不可係以任意順序判定，因為除法操作並不是交換性的。易言之，此陣列中之元素被處理的順序可能會影響結果。

在另一個範例（範例3）中，下面的偽碼：

```
for(i=0; i<N; i+=VLEN)
    A[Ind_dst[i]]=A[Ind_src[i]];
```

係設計成從陣列A中讀取由索引Ind_src[i]所指的一個元素，並於由Ind_dst[i]所指的位置將這個元素寫入到陣列A。易言之，向量Ind_dst和Ind_src含有陣列A中之元素的位址。將範例3映對到SIMD係包括判定可被並行處理而不違反資料關係，包括寫後讀、讀後寫、和寫後寫關係，的一個元素子集。此映對可包括檢測有出現在一個索引向量中但不出現在另一個向量中的位址，和/或在這兩個索引向量中為獨特的位址。當關係存在時，係將映對設計成確保這些操作係以程式順序進行，以避免此映對改變結果。例如，一個對A[c]之寫入和一個從A[c]之讀取不可被並行執行，其中這個從A[c]的讀取係被設計成發生在此偽碼之純量（即，連續的）版本的一個較晚迭代中。同樣地，若一個從A[c]之讀取被設計成於連續程序順序中係發生在一個對A[c]之寫入

之前，則至SIMD之映對不應容許在執行此讀取之前執行這個寫入。例如，若純量碼產生這個序列：A[2]=A[3]；A[5]=A[0]；A[0]=A[5] ；A[6]=A[0]，則至SIMD之映對應該要確保A[5]=A[0]賦值係在A[0]=A[5]賦值之前或同時執行。

因此，與本揭露內容一致的一種方法和系統係設計來檢測在純量操作至向量操作的映對中之衝突。此方法可包括判定一個元素向量的一個獨特元素子集。此方法可進一步包括在操作之順序可能會影響結果時保留程式順序。

例如，此方法可包括利用分散（scatter）和/或收集（gather）操作來檢測衝突。許多SIMD架構包括對於分散和/或收集操作的架構性支援。例如，英特爾（INTEL）眾多整合核心（Many Integrated Core）架構包括對於分散和/或收集操作的架構性支援。分散和/或收集操作典型上係使用在向量處理器中，用以從在，例如，一個暫存器，中的可能是稀疏的記憶體位置收集資料到相連記憶體位置，及/或將資料從在，例如，一個暫存器，中的相連記憶體位置分散到在，例如，系統記憶體，中的可能是稀疏的記憶體位置。利用收集和/或分散操作及/或其他常見可用SIMD操作有助於對於無序索引的衝突檢測，而無須額外硬體支援。藉由兩個硬體延伸，如於本文中所說明的，係可將此方法延伸成進行以程式順序檢測衝突的衝突檢測。

係可在任何包括有對於分散和/或收集操作之支援的架構上執行與本揭露內容一致的一種方法。此方法可係執行於包括有對於可規劃向量混洗之支援的一個架構上。在

一些實施例中，至少一些於本文中說明的這些操作係可藉由進行硬體中之操作而被組合和加速。

向量收集和向量分散操作為針對向量的暫存器間接定址的一種形式，其中，收集係涉及有索引（indexed）讀取，而分散係涉及有索引寫入。例如，針對具有為VLEN之向量長度（即，VLEN寬的SIMD）的一個向量架構系統而言，這些向量分散和收集操作可係定義為：

vgather base, Addr, Dst, mask

vscatter base, Addr, Src, mask

其中Addr、Dst和Src係對應於向量，而基底（base）和遮罩（mask）為純量。vgather係設計成從VLEN（不一定是相連的）記憶體位置收集上至VLEN個資料元素，這些記憶體位置的位址可從*base*和*Addr*運算出來（例如，(base[Addr[0]],...,base[Addr[VLEN-1]])），並將這些資料元素相連地儲存在*Dst*（例如，Dst[0],...,Dst[VLEN-1]）中。Vscatter係設計成將相連地儲存在*Src*（例如，Src[0],...,Src[VLEN-1]）中的上至VLEN個資料元素分散到VLEN個記憶體位置，這些記憶體位置的位址可從*base*和*Addr*運算出來，如針對vgather所說明的。

vgather和vscatter可將*mask*用作一個輸入運算元。*mask*可包括VLEN個位元，且*mask*的作動值可係對應於要被分散和/或收集的元素。例如，等於一的一個遮罩位元可係對應於要被收集或分散的一個元素，且等於零的一個遮罩位元可係對應沒有要被收集或分散的一個元素。若在一個分散

操作中有複數個值被分散到同一個記憶體位置，則只有一個能夠成功。

系統架構

第1圖例示出與本揭露內容一致的一個示範系統100實施例。系統100包括一個純量處理單元102、一個向量處理單元104和一個記憶體單元106。系統100可係針對管線化和/或並行處理而組配，如熟於此技者會可瞭解的。系統100可包括耦接至純量處理單元102和記憶體單元106的數個純量暫存器112，以及耦接至向量處理單元104和記憶體單元106的數個向量暫存器114。系統100可進一步包括耦接至記憶體單元106的一或多個快取記憶體122和一個記憶體控制器124。記憶體控制器126可係耦接至一個記憶體126。

如於本文中所說明的，純量處理單元102係組配來在純量資料上進行純量處理。例如，在處理前和/或處理後，這些純量資料可係儲存在一或多個這些純量暫存器112中。向量處理單元104係組配來在向量資料上進行向量處理。例如，在處理前和/或處理後，這些向量資料可係儲存在一或多個這些向量暫存器114中。

記憶體單元106係組配來管理資料和/或位址在純量暫存器112、向量暫存器114和/或（一或多個）快取122之間的移動。記憶體控制器124係組配來管理資料和/或位址在（一或多個）快取122和記憶體126之間的移動。

向量處理單元104和記憶體單元106可包括混洗邏輯132和或分散/收集邏輯134。如於本文中所說明的，此分散

/收集邏輯係組配來進行分散和/或收集操作。混洗邏輯132可為可規劃的。混洗邏輯132係組配來進行一種向量混洗操作。一種向量混洗操作係設計來排列一個元素集合。例如，若在含有{A, B, C, D}的一個向量暫存器上進行一種向量混洗操作，可能會有{B, D, C, A}的結果。混洗操作可接收複數個輸入集合（例如，二或更多個向量暫存器），並且可從其中任一個輸入中選擇值和/或從兩個輸入都選擇值。在一種可規劃混洗操作中，此排列並不由應用程式或編譯器作硬編碼。易言之，此操作可接收識別出要被輸出之元素的另一個輸入（例如，從記憶體或從一個暫存器）。

因此，與本揭露內容一致的一種系統可係組配來進行如於本文中所說明的任何方法。此系統可進一步被組配成利用可用電路功能，例如分散/收集邏輯，來進行一或多個這些功能。有益地，利用現有的電路功能可在進行這（些）功能上提供效率，且可因而在執行時間負擔上具有相對較小的衝擊。

示範方法論

第2圖例示出用於衝突檢測之示範操作的一個流程圖200。如於本文中所說明的，於此實施例中所例示的這些操作可係由與包括有一個向量處理器的一個系統（例如系統100）相關聯的電路和/或軟體模組進行。

流程可係從操作205開始。操作205可包括載入包括有一第一索引序列的一個資料集合，其中各個索引係對應於一個位址，並且此第一序列係對應於一個純量處理程序的

執行順序。可在操作210中判定此資料集合中的匹配索引。例如，可係藉由在這個資料集合上進行一個分散和一個收集操作來判定匹配索引。可在操作215中判定此資料集合中的獨特索引。係將匹配索引設計成指出一個衝突，且係將獨特索引設計成指出無衝突。

操作220可包括從匹配索引中選擇一個索引。若於此純量處理程序中的一個操作之結果係與執行順序有關（即，有序的），那麼所選擇的索引係設計成在此第一序列中為較早者。可在操作225形成包括有這些獨特索引和所選擇之索引的一個無衝突索引群集。操作230可包括載入對應於此無衝突群集中之至少一部分的資料。操作230可進一步包括並行地在這些資料上進行操作。

操作205、210、215、220、225和230可被重複，直到所有的索引都已經被消耗掉為止。以此方式，可能會干擾從純量處理程序到向量處理程序之變換的衝突可被檢測和處理，而有助於此項變換。

第3圖例示出與本揭露內容一致之用於單一索引衝突檢測之示範操作的一個流程圖300。如於本文中所說明的，於此實施例中所例示的這些操作可係由與包括有一個向量處理器的一個系統（例如系統100）相關聯的電路和/或軟體模組進行。表1包括用於對應於第3圖之流程圖300的無序單一索引衝突檢測之偽碼的一個範例。

表1

```
//初始化一些變數
conflictarray=(int*)malloc(MAX_SIZE);
Vmyid={0,1,2, ...VLEN-1};
for(i=0; i < N; i+=VLEN) {
        Vind=vload &Index[i];
        mainmask=all ones;
        do {
                //第 1 階段：檢測無衝突群集索引子集
                vscatter conflictarray, Vind, Vmyid, mainmask;
                vgather conflictarray, Vind, Vwrittenid;
                successmask = Vwrittenid= =Vmyid ? mainmask : 0;
                //第 2 階段：從列表中移除此群集
                mainmask = (mainmask & (~successmask)) ;
                //第 3 階段：在 Vind 之無衝突元素上作直方圖
                vgather hist, Vind, Vdest, successmask;
                vadd Vdest, Vdest, Vones, successmask;
                vscatter hist, Vind, Vdest, successmask;
        } while(mainmask);
}
```

流程可從操作305開始。操作305可包括初始化，例如，宣告和/或初始化變數。例如，可以MAX_SIZE的大小來配置一個陣列，「conflictarray」。操作305可進一步包括宣告和初始化長度為VLEN的一個向量Vmyid。例如，VLEN可係

對應於一個處理器的一個向量大小。如於本文中所說明的，係可以獨特值（例如，單調地增加整數值的元素，即，從零到VLEN-1）來初始化向量Vmyid。

係可針對一個索引集合的各個索引子集來進行操作310、315、320和325。例如，此索引集合可係對應於與陣列hist的元素之位址對應的索引，如於本文中針對範例1所說明的。此索引子集（即，索引「量塊（chunk）」）可係對應於進行這些操作的這個系統的一個向量之長度（例如，VLEN）。因此，操作310可包括量塊初始化。例如，係可以對應於一個特定索引量塊（子集）的索引Index[i]來載入一個向量Vind，並且可將一個遮罩（mainmask）設為全為一，而指出在這個量塊中之所有的VLEN個元素皆為有效的。這個範例係假設N是VLEN的倍數，以使得對於所有的量塊而言，所有的元素都是有效的。

更進一步地，係假設被組配來執行例示於表1中之示範操作的處理器（例如，處理器100）支援向量遮罩。易言之，當進行被設計成接收N個元素作為輸入的一個向量操作時，係假設此操作亦被設計成接收具有N個位元的一個遮罩。若一個遮罩位元被設定（例如，等於1），則此操作可在N中之對應元素上被操作。否則的話（例如，此遮罩位元為零），在對應元素上，此操作可被跳過（即，不被進行）。

可於操作315檢測在這個索引子集中的一個無衝突群組。例如，向量Vmyid之內容的至少一部分可在由向量Vind所指的位置被分散到陣列conflictarray中。此部份係對應於

遮罩mainmask。易言之，向量Vmyid的對應於mainmask之（一或多個）非零位元的（一或多個）值可被分散到conflictarray中。conflictarray的在由向量Vind所指之位置的值可接著被收集到目標向量Vwrittenid中。Vwrittenid可接著被與Vmyid作比較，元素對元素地（即，Vwrittenid[i]與Vmyid[i]比較），並且若這些元素是相等的，則在遮罩successmask中的一個對應位元可被設成mainmask的一個對應位元之值，否則便設為零。易言之，若Vwrittenid[i]等於Vmyid[i]，那麼Vind[i]是無衝突的，且對應的第i個successmask位元被設為一。否則的話，Vind[i]是與某個其他元素Vind[j]有衝突的（即，為與Vind中之另一個元素相同的值，且可能具有較低優先性），並且successmask的第i個位元被設為0。因此，successmask的非零位元係對應於此索引子集的無衝突群組。

於操作320，可將這個無衝突群組從索引「列表」中移除。例如，可將mainmask位元式地與非（not）successmask作且（and）處理。因此，mainmask之對應於此無衝突群集的位元可被設為零。當無衝突索引被檢測出來並接著在上面被作操作時，連續數遍的操作315、320和325可因而導致更多的mainmask位元被設為零。

操作325可包括在Vind之無衝突元素上的操作。操作325可係並行地在Vind之元素上進行。例如，繼續於本文在範例1中所說明的這個陣列hist，這個無衝突元素之群集可被收集到向量Vdest中，向量Vones可被加到Vdest，並且其

結果可接著依據向量Vind而被分散回到記憶體中。遮罩successmask係被設計成將這些操作限制於在操作315中所識別出的無衝突群集。係可在操作330判定是否已消耗此量塊。若此量塊還沒有被消耗掉，則流程可回到操作315。

可重複操作315、320、325和330，直到已消耗Vind。若此量塊已被消耗，則可在操作335判定此集合是否已被消耗。若此集合還沒有被消耗掉，則流程可回到操作310，並且Index[i]的下一個向量可接著被載入到Vind中。若已消耗此集合，則可結束操作340。可重複操作310、315、320、325、330和335，直到Index[i]之所有的N個元素都已被消耗為止。因此，無序單一索引衝突檢測係可包括有多個巢套迴圈。一個外層迴圈係設計成要消耗向量Index[i]的數個量塊，其中各個量塊係對應於此系統（例如，系統100）的向量長度。一個內層迴圈係設計成要在各個量塊上操作，以檢測和/或識別此量塊之無衝突元素群集。此內層迴圈可接著在所識別之（一或多個）群集上操作並將其移除。這可重複，直到已消耗此量塊為止。

針對第3圖所說明的這些操作可能不是那麼有效率。首先，這些操作是在conflictarray已配置且係大到足以確定分散是在陣列界限內這樣的一個假設上的述詞。確保分散在陣列界線內，係假設儲存在Vind中之資料的最大值為已知的。在一些情況中，要知道Vind的最大值可能會是很困難的，尤其是在資料可能正在動態地改變時。

第二，分散到大陣列和/或從大陣列作收集的效率並不

高。在一些架構上，收集/分散的效能是與收集/分散操作所存取的快取行之數量成反比。因此，與較小陣列相較之下，由於所存取之快取行以及快取未命中的較大數量，於此等架構上從大陣列作收集/分散到大陣列中可能會導致顯著的效能降格。在其他架構上，收集/分散的效能是與所存取的快取行之數量無關。與較小陣列相較之下，由於快取未命中增加，於此等架構上從大陣列作收集/分散到大陣列中可能會導致效能降格。

將收集/分散操作限制於存取較小陣列可能會是值得嚮往的。將分散/收集操作限制於較小陣列的一種示範技術是將Vind的值雜湊到一個較小的，例如為HashTableSize大小的，陣列中。此較小陣列之大小可係選擇來對應於單一個快取行。在一個實施例中，一種雜湊技術可係利用來自於各個索引值（Vind）的$\log_2$(HashTableSize)個最低有效位元（least significant bits, LSB）。於此範例中，若HashTableSize是，例如，八，那麼係可利用來自於各個索引值（Vind）的$\log_2$8（也就是3）個LSB來編索引到雜湊表（例如，Vind_hashed）中。在另一個範例中，一種雜湊技術可係使用模數操作。於此範例中，係可使用索引值（Vind）除以HashTableSize的餘數（例如，Vind mod HashTableSize）來編索引到雜湊表中。若HashTableSize等於VLEN，則可係使用一個一般可規劃向量混洗操作，而非收集/分散操作，如果這樣的混洗比收集/分散更快的話。

第4圖例示出與本揭露內容一致之用於單一索引衝突

檢測之示範操作的一個流程圖400。如於本文中所說明的，於此實施例中所例示的這些操作可係由與包括有一個向量處理器的一個系統（例如系統100）相關聯的電路和/或軟體模組進行。表2包括用於對應於第4圖之流程圖400的無序單一索引衝突檢測之偽碼的一個範例。

表2

```
//初始化一些變數
_declspec(align(VLEN)) int conflictarray[HashTableSize];
Vmyid={0,1,2, ...VLEN-1};
for(i=0; i < N; i+=VLEN) {
        Vind=vload &Index[i];
        //將索引雜湊到一個較小資料結構中
        Vind_hashed=Vind % HashTableSize;
        mainmask=all ones;
        do {
            //第1階段：檢測無衝突群集索引子集
            vscatter conflictarray, Vind_hashed, Vmyid, mainmask;
            vgather conflictarray, Vind_hashed, Vwrittenid;
            successmask = Vwrittenid= =Vmyid ? mainmask;
            //第2階段：從列表中移除此群集
            mainmask = (mainmask & (~successmask)) ;
            //第3階段：在Vind之無衝突元素上進行直方圖
            vgather hist, Vind, Vdest, successmask;
```

```
        vadd Vdest, Vdest, Vones, successmask;
        vscatter hist, Vind, Vdest, successmask;
    } while(mainmask);
}
```

描繪於表2和第4圖中的操作分別與描繪於表1和第3圖中的操作類似，除了conflictarray可為HashTableSize大小，以及索引可被雜湊到這個conflictarray中以外。因此，只說明有所差異的那些操作。

流程可從操作405開始。操作405可包括初始化，例如，宣告和/或初始化變數。例如，可將一個陣列，「conflictarray」，宣告為具有HashTableSize的大小。conflictarray可係對齊於一個向量處理器（例如，第1圖中的向量處理器104）的一個向量長度（VLEN）。在一些實施例中，conflictarray可係對齊於被組配來執行於流程圖400中所描繪之操作的處理器（例如，處理器100）的一個快取行大小。操作405可進一步包括宣告和初始化向量Vmyid，如於本文中針對操作305所說明的。

係可針對一個索引集合中之索引的各個子集（「量塊」）進行操作310、410、315、320、325和330。因此，操作310可包括量塊初始化。例如，可以對應於一個特定索引量塊（子集）的索引Index[i]來載入一個向量Vind。可接著在操作410將向量Vind雜湊到向量Vind_hashed中。例如，Vind可係藉由進行Vind模HashTableSize（Vind modulo HashTableSize）

而雜湊，如於本文中所說明的。操作410可進一步包括將一個遮罩（例如，mainmask）設為全為一。可接著利用Vind_hashed（取代Vind）來進行操作315，如於本文中所說明的。可接著進行操作320、325和330，如於本文中所說明的。

可重複操作315、320、325和330，直到已消耗Vind為止。Index[i]的下一個向量可接著被載入到Vind（操作310），並且可重複操作315、320、325和330。可重複操作310、410、315、320、325、330和335，直到Index[i]之所有元素都已被消耗為止。流程可接著在操作340結束。

與流程圖300之操作相比，流程圖400的這些操作係可提供效率增進。例如，conflictarray的大小可係對應於被組配來執行流程圖400之操作的處理器之向量長度。Conflictarray之大小可被設計成對應於此處理器的一個快取行大小。以此方式，流程圖400之操作可被設計來開拓被組配來進行這些操作的處理器之架構。

可識出，雜湊可能會導致錯誤肯定（false-positive）。易言之，雜湊，例如模數操作，可能會對不同索引提供相同結果。這樣的錯誤肯定可能會因而導致平行性降低。

在於本文中所說明的直方圖範例1中，係可以任意順序來檢測衝突。易言之，陣列hist之元素被增量的順序並不會影響結果（增量操作是交換性的）。在於本文中所說明的範例2中，並不能以任意順序來檢測衝突，因為處理陣列中之元素的順序可能會影響結果（即，除法操作是非交換性的）。

因此，在順序可能會影響結果時，對於單一索引的衝突檢測（例如，範例2）係以程式順序進行。

可利用例示於第3圖和/或第4圖中的方法來進行有序單一索引衝突檢測。例如，係可以具有不同特點的分散操作「vscatterp」來使用例示於表1或表2中的偽碼。如於本文中所說明的，若在一個分散操作（例如，vscatter）中有複數個值被分散到同一個記憶體位置，則只有一個會成功。在典型的分散操作中，係假設元素沒有排序，或者是重疊的數個寫入係可以程式順序進行。

在vscatterp中，當複數個值被分散到同一個記憶體位置時（即，具有相同的索引），vscatterp係設計成確保在程式順序中最早的值會最晚被寫入。例如，vscatterp可被設計成最後寫入在程式順序中最早的值。在另一個範例中，vscatterp可被設計成寫入在程式順序中最早的那個，並抑制後續的值。係可以多種方式來實施vscatterp。例如，vscatterp可為一個新的指令。在另一個範例中，vscatterp可係對應於可達到vscatterp結果的一個指令序列。因此，可藉由將以「vscatterp」取代「vscatter」來利用例示於表1和2中的偽碼來進行有序單一索引衝突檢測。那麼，程式順序可被保留。

在例示於第3和4圖中的操作以及表1和表2之示範偽碼中，係可在一個索引集合的一個索引子集（即，「量塊」）上進行衝突檢測操作，一次一個量塊。量塊的大小可係對應於進行這些操作的處理器之向量大小（即，SIMD寬度（例如，

VLEN))。易言之，一個量塊被處理，然後下一個量塊被處理，並以此類推，直到已處理此索引集合為止（利用，例如，雙巢套迴圈）。對索引對進行衝突檢測至少部份係相對地更為複雜，因為可能會有複數個排序限制。

第5圖例示出與本揭露內容一致之用於有序索引對衝突檢測之示範操作的一個流程圖500。如於本文中所說明的，於此實施例中所例示的這些操作可係由與包括有一個向量處理器的一個系統（例如系統100）相關聯的電路和/或軟體模組進行。表3包括對應於流程圖500之偽碼的一個範例。

如針對表1和表2之示範偽碼所說明的，在流程圖300和400中，會處理各個索引子集（「量塊」），直到整個子集都被消耗為止，然後才前進到下一個子集。在流程圖500和表3之範例中，可係處理一個有序元素集合。不若於各個迭代中在一個元素量塊上操作直到這整個量塊被消耗為止，而係在從此子集之開端開始可以的話越多越好的元素上操作，直到檢測出資料相依性為止。例如，若向量長度為VLEN，目前子集從元素j開始，且在元素j+1檢測到一個衝突，則元素j會在目前迭代中被消耗。在下一個迭代中，下一個子集會是從元素j+1開始，並且將會包括元素j+1到j+VLEN。處理可接著前進，直到另一個衝突被檢測出來為止，例如元素j+m，其中m大於一且小於或等於VLEN，可消耗元素j+1到j+m-1，並且接下來的迭代會從j+m開始。可重複此迭代處理程序，直到整個集合都被操作且消耗掉為

止。

以此方式做迭代可簡化對於有序索引對的衝突檢測。在所檢測到的第一個資料相依性處停止各個迭代可容許藉由檢測寫後讀關係來檢測衝突。易言之，若在純量程式中，迭代j寫入一個位置，且迭代j+1從同一位置讀取，則在SIMD版本中，迭代j+1應該要讀取在迭代j被寫入的值，並且因此不能夠與迭代j並行執行。在沒有以此方式迭代的情況下，衝突檢測可能會包括檢測讀後寫和/或寫後寫關係，並且因而會相對地較為複雜。

表3中之範例示出在有序索引對上的衝突檢測，於所檢測出來的第一個寫後讀關係處停止。例如，可係利用一個衝突陣列「conflictarray」來檢測寫後讀相依性。一開始，可對conflictarray的所有元素寫入一個已知常數值（例如CONSTANT）。可接著利用來自於在純量情況中要被寫入的資料陣列的索引（例如Ind_dst[i]），而利用，例如，vscatterp來將獨特數分散到conflictarray中。可對這些索引，Ind_dst[i]，做雜湊以使得能夠利用相對較小的衝突陣列，如於本文中所說明的。可接著利用來自於在純量情況中要被讀取的資料陣列之索引（例如，Ind_src[i]）而從conflictarray收集資料。可對這些索引，Ind_src[i]，做雜湊，如於本文中所說明的。

因此，若收集操作收集到不為CONSTANT的一個值，則用於分散的索引集合和用於收集的索引集合有重疊。重疊並不一定會導致衝突。例如，若在純量情況中，迭代j永

遠從索引j讀取並寫入到索引j-1，那麼這些操作係可被並行化。因此，若檢測到重疊，係可進行進一步的處理來判定在分散和收集操作中的此重疊是否會導致違反資料相依性。這個進一步的處理可係在當收集索引對應於比分散索引更晚的純量迭代時（即，這個值應該要在讀取之前先被寫入）進行。

例如，被分散到conflictarray的獨特數可為單調增加的。易言之，這些獨特數的順序係設計成與純量迭代順序匹配。在進行收集操作之後，可藉由判定所收集的第j個元素是否大於所分散的第j個元素，而判定出所元素j是否從較晚的迭代收集了一個值（即，可能會違反資料相依性）。

表3

```
V_hi = {CONSTANT, CONSTANT, ..., CONSTANT}
V_upperbound = {N, N, ..., N};
while(i<N)
{
    //第1階段：檢測無衝突群集索引子集
    mainmask = {i, i+1, ..., i+VLEN-1} < V_upperbound;
    Vind_src= vload &Ind_src[i] % VLEN;
    Vind_dst= vload &Ind_dst[i] % VLEN;
    vstore conflictarray, V_hi
    vscatterp conflictarray, Vind_dst, Vmyid, mainmask;
    vgather conflictarray, Vind_src, Vwrittenid, mainmask;
```

```
        //第2階段：運算遮罩並截斷它
        mainmask = Vwrittenid >= Vmyid ? mainmask;
        vmasktruncateZR(mainmask);
        //第2階段：在無衝突索引上進行實際操作
        vgather A, Vind_src, Vsrc, mainmask;
        vscatter A, Vind_dst, Vsrc, mainmask;
        i+= popcount(mainmask);
}
```

現在，請參考第5圖和表3，流程可從操作505開始。操作505可包括初始化，例如，宣告和初始化變數。例如，一個陣列，「conflictarray」，可被宣告成具有HashTableSize的大小，如於本文中針對第4圖和表2所說明的（例如，HashTableSize可係對應於被組配來進行第5圖之操作的一個處理器之向量長度VLEN）。可宣告一個向量，Vmyid，並可以單調增加的獨特值，例如{0, 1, 2,..., VLEN-1}，來將其初始化。在表3中之偽碼中，係假設已進行初始化。

操作510可包括檢測一個索引子集的一個無衝突群組。雖然此子集可係對應於一個量塊，下一個量塊亦係可與目前量塊重疊，如於本文中所說明的。例如，操作510可包括依據針對一個量塊之迭代號碼的集合（即，i、i+1、…、i+VLEN-1）與被初始化成N的一個向量V_upperbound間之比較來設定一個遮罩「mainmask」。此遮罩係設計來識別出在目前迭代中要在上頭做操作的元素。若此索引子集含有

比對應的向量長度更少的元素（例如，此子集靠近索引集合的末端（即，N-i<VLEN)），則此遮罩可不被設為全為一。

操作510可進一步包括從Ind_src[i]載入和雜湊一第一索引集合到一個向量Vind_src中，以及從Ind_dst[i]載入和雜湊一第二索引集合到一個向量Vind_dst中。在純量情況中，Ind_src[i]係對應於被讀出的索引，且Ind_dst[i]係對應於被寫入的索引。Vind_src和Vind_dst可具有為VLEN的長度。Ind_src[i]和Ind_dst[i]可分別藉由進行Ind_src[i]模VLEN和Ind_dst[i]模VLEN，而分別被雜湊到Vind_src和Vind_dst中。

操作510可進一步包括將一個已知的相對較大值（V_hi）集合儲存到conflictarray中。例如，V_hi可係對應於VLEN。若收集到還沒有被分散到的一個元素，則所收集到的這個元素之值可能是V_hi（例如，VLEN），對應於沒有衝突的一個索引。易言之，V_hi可被選擇成使其值大於Vmyid的任何獨特（例如，單調增加的）值。

操作510可進一步包括利用Vind_dst中之「目標」索引而將Vmyid分散至conflictarray中。例如，可利用vscatterp來分散Vmyid，如於本文中所說明的。操作510可包括利用Vind_src中之「來源」索引而將conflictarray之內容收集到一個向量，writtenid，中。

由於操作510，故若在向量Vmyid中的一個值大於在向量Vwrittenid中的對應值，那麼就可能存在有資料相依性衝突。這係對應於在純量序列中之被讀出的索引和被寫入的索引是相同的且讀取是發生在寫入之後的情況。

操作515可包括運算一個遮罩。例如，此遮罩可係藉由將Vwritteni的第j個值與Vmyid的第j個值做比較來運算，並且若Vwritteni的第j個值大於或等於Vmyid的第j個值，那麼（mainmask之）對應的遮罩位元便被維持。否則的話，mainmask之對應的遮罩位元便被清除。易言之，操作515係設計來在存在有寫後讀關係的時候清除mainmask之對應位元。

操作515可進一步包括截斷遮罩。例如，可利用vmasktruncateZR(mainmask)來截斷遮罩（例如，mainmask）。VmasktruncateZR係設計來識別對應於一第一資料相依性的一個遮罩位元，並且係設計來接著清除在所識別之遮罩位元之後的（即，更具有效性的）所有的遮罩位元。例如，若遮罩＝0b1101，則vmasktruncateZR(mask)＝0b0001。有多種方式可實施vmasktruncateZR。例如，vmasktruncateZR可為一個新的指令。在另一個範例中，vmasktruncateZR可係對應於被設計來達到vmasktruncateZR結果的一個指令序列。

在mainmask上使用vmasktruncateZR係設計來保證讀後寫關係受到尊重。讀後寫相依性只可在當被寫入到ind[j]（ind[i]==ind[j]）的值之遮罩位元為1且j大於i時從ind[i]讀出的遮罩位元時為零的時候被違反。藉由在i截斷遮罩，則對於此迭代，接下來的寫入會被自動失效，也就是不會在讀取之前發生。

易言之，在運算出遮罩之後，mainmask係代表可被容

許同時執行的，亦即，不會破壞寫後讀關係的，一個獨特元素集合。然而，係可能存在有其他類型的衝突，例如，讀後寫和/或寫後寫關係。對於並不會違反寫後讀關係的元素而言，mainmask之位元可為一，但這些元素可能會違反讀後寫和/或寫後寫關係。vmasktruncateZR係設計來避免可能會違反，例如，讀後寫和/或寫後寫關係的操作，而不具體地識別出此等關係。清除遮罩位元會導致對應於被清除之位元的操作係在之後的迭代中被執行。

操作520可包括在無衝突元素上操作。例如，陣列A的由Ind_src[i]所指的和對應於值為一之mainmask位元的元素可被收集到向量Vsrc中。Vsrc的元素可接著被分散到陣列A之由Ind_dst[i]所指的位置中。易言之，收集和分散操作可導致陣列A之由Ind_src[i]所指的值被指派到陣列A之由Ind_dst[i] 所指的位置。

操作520可包括判定實際被處理的元素之數量。例如，可藉由popcount(mainmask)（即，在mainmask中的人口數，或一的數量）來增量索引i，以在此操作中藉由所處理之元素的數量推進索引i。可重複操作510、515和520直到整個索引集合（例如，N個元素）皆已被消耗為止。

因此，流程圖500之操作係設計來（利用，例如，vscatterp）將一個獨特（且以程式順序單調增加的）值集合分散到一個衝突陣列中。這些獨特值可係於，例如，初始化時，被儲存在一個向量暫存器，Vmyid，中。目標索引可係儲存在另一個向量暫存器，Vind_dst，中。衝突陣列中的

值可接著利用儲存在Vind_src中的來源索引而被收集到另一個向量暫存器，Vwrittenid，中。可接著將Vwrittenid與Vmyid做比較。若Vwrittenid[i]大於或等於Vmyid[i]，那麼對於所對應的目標和來源索引對而言，可能要尊重這個寫後讀關係。

例如，一個純量序列可能包括有某個數量的（例如，四個）迭代：

迭代ID	迴圈指令
0	A[2]=A[1]
1	A[3]=A[1]
2	A[0]=A[2]
3	A[1]=A[3]

可接著以獨特的單調增加值來初始化一個向量Vmyid：

Vmyid = {0, 1, 2, 3}。

在這個範例中，Vmyid係對應於迭代ID。用於分散操作（vscatterp）的Vind_dst係對應於在陣列A中的被寫入之索引：

Vind_dst = {2, 3, 0, 1}。

用於收集操作的Vind_src係對應於在陣列A中的被讀出之索引：

Vind_src = {1, 1, 2, 3}。

在這個範例中，各個索引都小於衝突陣列的大小（其如下面所述地為4）VLEN，所以並沒有進行雜湊。

係可以一個常數來初始化conflictarray，例如，

conflictarray = {4, 4, 4, 4}。

假設在一個分散操作（vscatterp conflictarray, Vind_dst, Vmyid, mainmask）之後，mainmask全為一，則Vmyid之元素可在Vind_dst所指之位置被寫入到conflictarray中：

conflictarray = {2, 3, 0, 1}.

具體而言，在這個範例中的此分散操作產生

對於i=0,1,...,3，conflictarray[Vind_dst[i]]=Vmyid[i]。

易言之，Vmyid的第一個元素（即，0）可被寫入到在conflictarray中之由Vind_dst的第一個元素（即，2）所指的位置，Vmyid的第二個元素(即，1)可被寫入到在conflictarray中之由Vind_dst的二個元素（即，3）所指的位置，以此類推。

在一個收集操作（vgather conflictarray, Vind_src, Vwrittenid, mainmask）之後，可從Vind_src 所指之位置讀出conflictarray的元素並將其寫入到Vwrittenid中：

Vwrittenid = {3, 3, 0, 1}。

具體而言，在這個範例中的此收集操作產生

對於i=0,1,...,3，Vwrittenid[i]=conflictarray[Vind_src[i]]。

易言之，conflictarray的第二個元素(即，3，對應於Vind_src[0]＝1）可被寫入到Vwrittenid的第一個元素，conflictarray的第二個元素（即，3，對應於Vind_src[1]＝1）可被寫入到Vwrittenid的第二個元素，以此類推。

可接著藉由運算Vwrittenid = {3, 3, 0, 1}和Vmyid = {0, 1, 2, 3}來運算一個遮罩，以產生：

mainmask = {1, 1, 0, 0}。

若在向量Vmyid中的一個值大於在Vwrittenid中的對應值（即，Vmyid[i] > Vwrittenid[i]），那麼便可能存在有一個寫後讀資料相依性。在Vmyid中的值係對應於純量序列中之迭代號碼（迭代ID）。在Vwrittenid中的值係對應於純量序列中之迭代號碼，其中「被寫入」索引與對應的「被讀出」索引是一樣的。例如，對於迭代0（Vmyid[0]=0）而言，被讀出索引為1（Ind_src[0]=1），而對於迭代3（Vwrittenid[0]=3）而言，寫入索引為1（Ind_dst[3]=1）。同樣地，對於迭代2（Vmyid[2]=2）而言，被讀出索引為2（Ind_src[2]=2），而對於迭代0而言，寫入索引為2（Ind_dst[0]=2）。

因此，對應於Vwrittenid >= Vmyid的各個遮罩位元為一，且對應於Vwrittenid < Vmyid的各個遮罩位元為零。例如，對於Vwrittenid和Vmyid的最前面兩個位元而言，遮罩位元是一，且對於Vwrittenid和Vmyid的最後面兩個位元而言，遮罩位元是零。mainmask因而指出在最前面的兩個迴圈迭代（即，迭代ID 0和1）之間並沒有衝突，所以這兩個迭代可並行進行。mainmask進一步指出在最後面的兩個迴圈迭代（即，迭代ID 2和3）與最前面的兩個迴圈迭代中之一者或另一者或二者之間可能有衝突，因此不能與最前面的兩個迴圈迭代並行執行。

藉由審視這個相當簡單的示範純量序列，可識出，在對應於迭代ID 0和1的操作中，陣列A的位置1被讀取，且陣列A的位置2和3被寫入。這可能會很明顯，即，對於這兩個迭

代ID而言，並沒有任何的讀後寫衝突、寫後讀衝突或寫後寫衝突。因此，對應於迭代ID 0和1的操作係可被並行進行（即，可被「SIMD化」）。

繼續這個範例，對應於迭代ID 2和3的操作，除了其他的以外，係包括有對於陣列A之位置2和3的讀取以及對陣列A之位置1的寫入。因此，對於位置2的讀取不可與寫入到位置2（迭代ID 0）並行執行，並且對於位置3的讀取不可與寫入到位置3（迭代ID 1）並行執行，亦即，在迭代ID 0與迭代ID 2及迭代ID 1與迭代ID 3之間存在有寫後讀關係。此外，對位置1的寫入不可與從位置1之讀取並行進行，亦即，在迭代ID 3與迭代ID 0和/或迭代ID 1之間存在有讀後寫關係。

因此，利用與本揭露內容（例如，第5圖和，例如，表3）的一種方法並產生一個為{1, 1, 0, 0}的mainmask，指出對應於迭代ID 0和1之操作係可被並行進行，而對應於迭代ID 2和3的操作不可與對應於迭代ID 0和1的操作並行進行。這符合藉由審視這個相當簡單的範例所得到的結論。雖然這個範例並不包括雜湊，但流程圖500的操作和表3之範例係可包括雜湊，如於本文中所說明的。

大體而言，業已說明被組配來在將純量處理程序轉換成並行處理程序時檢測衝突的系統。業已說明被設計來在無序單一索引、有序單一索引和/或有序索引對中檢測衝突的方法。這（些）方法可包括被設計成確保在複數個值被分散到同一個位置時在程式順序中最早的值會最後被寫入的一個

分散操作之變體，即，vscatterp。這（些）方法可包括被設計成在發現寫後讀關係時對一個迭代截斷衝突檢測以避免在這個迭代中之其他讀取和/或寫入衝突的一個遮罩截斷操作，即，VmasktruncateZR(mask)。在一些實施例中，這（些）方法係被設計成由包括有分散和收集功能的一個系統執行。此等分散和收集功能可接著被開拓來有效地檢測衝突，而無須評估所有的可能索引組合。

於本文中已說明一種示範系統。可能會有修改體。例如，記憶體126可包含一或多個下面的記憶體類型：半導體韌體記憶體、可規劃記憶體、非依電性記憶體、唯讀記憶體、可電氣式規劃記憶體、隨機存取記憶體、快取記憶體、磁碟記憶體、和/或光碟記憶體。額外地，或，或者是，記憶體126可包含其他和/或之後所研發的電腦可讀記憶體類型。

於本文中所說明的這些方法之實施例可係實施在包括有一或多個儲存媒體的系統中，這一或多個儲存媒體個別地或組合式地具有儲存在內之指令，這些指令在由一或多個處理器執行時會進行那些方法。於此，處理器可包括，例如，一個純量處理單元（例如，第1圖之純量處理單元102）、一個向量處理單元（例如，第1圖之向量處理單元104）、和/或可規劃電路。因此，係意欲要使依據於本文中所說明的這些方法的操作可被散佈在多個實體裝置中。同時，亦意欲要使這些方法之操作係可被個別地或以子組合的方式進行，如熟於此技者會可瞭解的。因此，並非在各個流程圖中的所有這些操作都需要被執行，並且本揭露內容係明顯意欲要賦能

此等操作之所有的子組合，如熟於此技者會可瞭解的。

儲存媒體可包括任何類型的有形媒體，例如，任何類型的碟片，包括軟碟、光碟、唯獨光碟記憶體（compact disk read-only memorie, CD-ROM）、可複寫光碟（compact disk rewritable, CD-RW）和磁光碟，半導體裝置，例如唯讀記憶體（read-only memory, ROM）、隨機存取記憶體（random access memory, RAM）（例如動態和靜態RAM）、可拭除可規劃唯讀記憶體（erasable programmable read-only memory, EPROM）、可電氣式拭除可規劃唯讀記憶體（electrically erasable programmable read-only memory, EEPROM）、快閃記憶體、磁卡或光學卡，或任何類型的適於儲存電子指令之媒體。

於本文中之任何實施例中使用時，「電路」可包含，單獨地或是以組合形式，例如，儲存由可規劃電路所執行之指令的硬體電路、可規劃電路、狀態機電路、和/或韌體。

依據一個面向，係揭露有一種用於衝突檢測之方法。此方法可包括下列步驟：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果與該執行順序有關，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨

特索引和所選擇的該索引；以及載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

依據另一個面向，係揭露有一種系統，其包含具有個別地或以組合形式儲存在內之指令的一或多個儲存媒體，該等指令在被一或多個處理器執行時會致使包含下列之操作：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果與該執行順序有關，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；以及載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

依據又一個面向，係揭露有一種組配來檢測衝突的設備。該設備可包括一個記憶體；以及一個處理器，該處理器包含一個純量處理單元和一個向量處理單元。該處理器係組配來進行下列步驟：載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於該記憶體中的一個位址，並且該第一序列係對應於在使用該純量處理單元時的一個純量處理程序之執行順序；判定在該資料集合中的匹配索引；判定在該資料集合中的獨特索

引；從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果與該執行順序有關，則所選擇的該索引係在該第一序列中為較早者；形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；以及載入對應於該無衝突群集之至少一部分的資料，並利用該向量處理單元並行地在該等資料上進行該操作。

已於本文中所運用的詞語和表達係用作說明用語，而非限制用語，並且並沒有意圖要藉由使用這樣的詞語和表達來排除所示出和所說明的特徵（或其中之部分）之任何等效體，並且應認同，係可能會有落在申請專利範圍之範疇內的許多修改體。因此，本案申請專利範圍係意欲要涵蓋所有這樣的等效體。

【圖式簡單說明】

第1圖例示出與本揭露內容一致的一個示範系統實施例；

第2圖例示出用於與本揭露內容一致的衝突檢測之示範操作的流程圖；

第3圖例示出用於與本揭露內容一致的單一索引衝突檢測之示範操作的流程圖；

第4圖例示出用於與本揭露內容一致的單一索引衝突檢測之示範操作的流程圖；並且

第5圖例示出用於與本揭露內容一致的有序索引對衝突檢測之示範操作的流程圖。

【主要元素符號說明】

100...系統/處理器

102...純量處理單元

104...向量處理單元/向量處理器

106...記憶體單元

112...純量暫存器

114...向量暫存器

122...快取記憶體

124...記憶體控制器

126...記憶體

132...混洗邏輯

134...分散/收集邏輯

200、300、400、500...流程圖

205～230、305～340、401～410、505～520...操作

201229898

發明專利說明書

（本說明書格式、順序，請勿任意更動，※記號部分請勿填寫）

※申請案號：100145914

※申請日：100.12.13　　※IPC 分類：G06F 9/38(2006.01)

一、發明名稱：(中文/英文)

用於利用單指令多資料作衝突檢測之機構
MECHANISM FOR CONFLICT DETECTION USING SIMD

二、中文發明摘要：

一種系統和方法係設計來在將純量處理程序轉換成並行處理程序（「SIMD化」）時檢測衝突。係可針對一個無序單一索引、一個有序單一索引、和/或有序索引對而檢測衝突。可更進一步針對寫後讀關係來檢測衝突。衝突檢測係設計來識別出在一連串迭代中的不可以被並行從事的操作（即，迭代）。

三、英文發明摘要：

A system and method are configured to detect conflicts when converting scalar processes to parallel processes ("SIMDifying"). Conflicts may be detected for an unordered single index, an ordered single index and/or ordered pairs of indices. Conflicts may be further detected for read-after-write dependencies. Conflict detection is configured to identify operations (i.e., iterations) in a sequence of iterations that may not be done in parallel.

七、申請專利範圍：

1. 一種用於衝突檢測之方法，其包含下列步驟：

載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；

判定在該資料集合中的匹配索引；

判定在該資料集合中的獨特索引；

從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所選擇的該索引係在該第一序列中為較早者；

形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；以及

載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

2. 如申請專利範圍第1項之方法，其中判定在該資料集合中的匹配索引之步驟包含分散以及收集。

3. 如申請專利範圍第2項之方法，其中分散係設計成會確保在該純量處理程序之該執行順序中為最早的一個索引最後被寫入。

4. 如申請專利範圍第2項之方法，其進一步包含下列步驟：

載入具有數個獨特值的一個向量，其中該等獨特值係設計成要被分散，並且其中該向量的長度係對應於被組配來進行該操作的一個向量處理器之向量長度。

5. 如申請專利範圍第1項之方法，其中該資料集合包含由

索引組成之該第一序列的經雜湊值，雜湊之步驟係組配成利用所具有之大小比一最大第一索引小的一個陣列來有助於判定該等匹配索引。

6. 如申請專利範圍第1項之方法，其中判定在該資料集合中的該等匹配索引之步驟係以該純量處理程序之該執行順序進行。

7. 如申請專利範圍第1項之方法，其中該資料集合包含由索引組成之一第二序列，並且該第二序列係對應於該純量處理程序的該執行順序。

8. 如申請專利範圍第7項之方法，其進一步包含下列步驟：

檢測在由索引組成之該第一序列與該第二序列之間的寫後讀關係(read-after-write dependencies)。

9. 如申請專利範圍第8項之方法，其中該部分係至少部份基於所檢測出的一個寫後讀關係而被判定。

10. 一種系統，其包含具有個別地或以組合形式儲存在其上之指令的一或多個儲存媒體，該等指令在被一或多個處理器執行時會致使包含下列之操作：

載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於一個位址，並且該第一序列係對應於一個純量處理程序的執行順序；

判定在該資料集合中的匹配索引；

判定在該資料集合中的獨特索引；

從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所

選擇的該索引係在該第一序列中為較早者；

形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；以及

載入對應於該無衝突群集之至少一部分的資料，並且並行地在該等資料上進行該操作。

11. 如申請專利範圍第10項之系統，其中判定在該資料集合中的匹配索引之操作包含分散以及收集。

12. 如申請專利範圍第11項之系統，其中分散係設計成會確保在該純量處理程序之該執行順序中為最早的一個索引最後被寫入。

13. 如申請專利範圍第10項之系統，其中該資料集合包含由索引組成之一第二序列，並且該第二序列係對應於該純量處理程序的該執行順序。

14. 如申請專利範圍第13項之系統，其中該等指令在被一或多個該等處理器執行時會致使包含下列之操作：

檢測在由索引組成之該第一序列與該第二序列之間的寫後讀關係。

15. 一種組配來檢測衝突的設備，該設備包含：

一個記憶體；以及

一個處理器，其包含一個純量處理單元和一個向量處理單元，該處理器係組配來進行下列步驟：

載入一個資料集合，該資料集合包含由索引組成之一第一序列，其中各個索引係對應於該記憶體中的一個位址，並且該第一序列係對應於在使用該

純量處理單元時的一個純量處理程序之執行順序；

判定在該資料集合中的匹配索引；

判定在該資料集合中的獨特索引；

從該等匹配索引中選擇一個索引，其中若在該純量處理程序中的一個操作之結果取決於該執行順序，則所選擇的該索引係在該第一序列中為較早者；

形成由索引組成的一個無衝突群集，該無衝突群集包括該等獨特索引和所選擇的該索引；及

載入對應於該無衝突群集之至少一部分的資料，並利用該向量處理單元並行地在該等資料上進行該操作。

16. 如申請專利範圍第15項之設備，其中該處理器包含分散和收集邏輯，並且判定匹配索引之步驟包含分散以及收集。

17. 如申請專利範圍第16項之設備，其中分散係設計成會確保在該純量處理程序之該執行順序中為最早的一個索引最後被寫入。

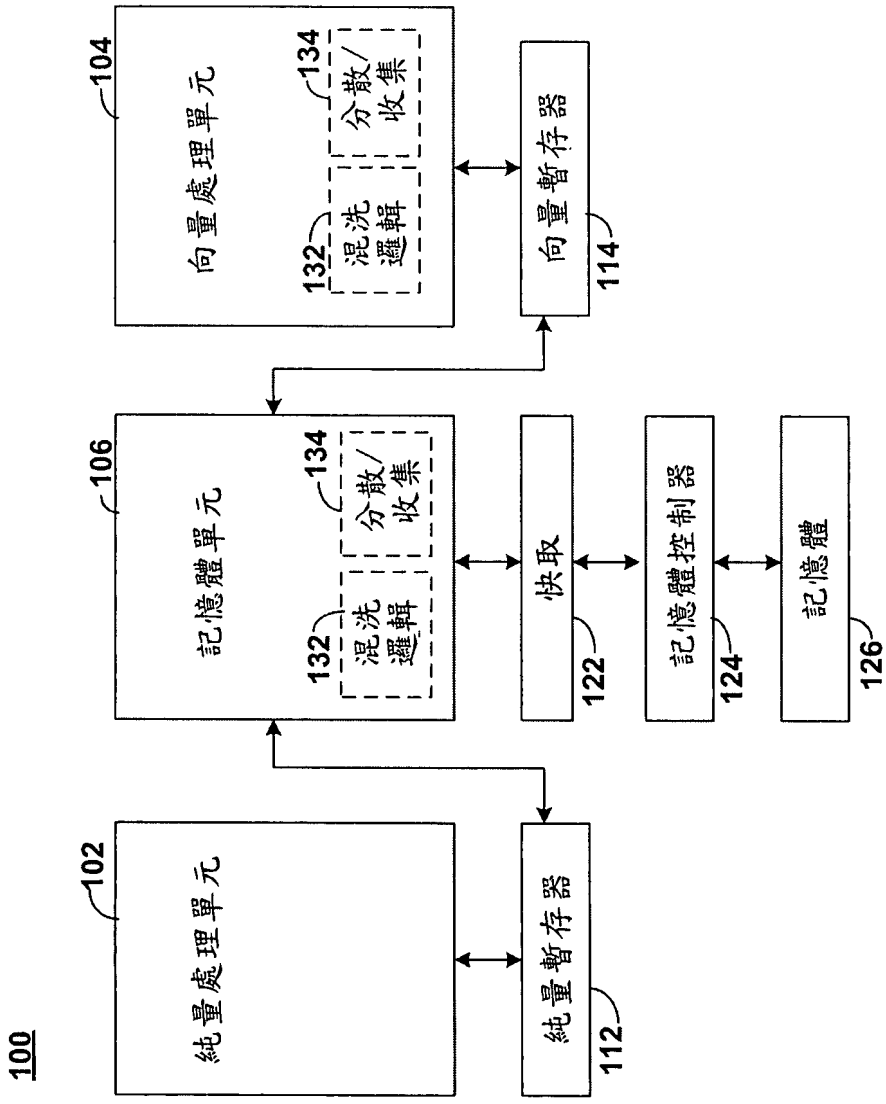
18. 如申請專利範圍第15項之設備，其中該資料集合包含由索引組成之該第一序列的經雜湊值，雜湊之步驟係組配成利用所具有之大小比一最大第一索引小的一個陣列來有助於判定該等匹配索引。

19. 如申請專利範圍第15項之設備，其中該資料集合包含由索引組成之一第二序列，並且該第二序列係對應於該純

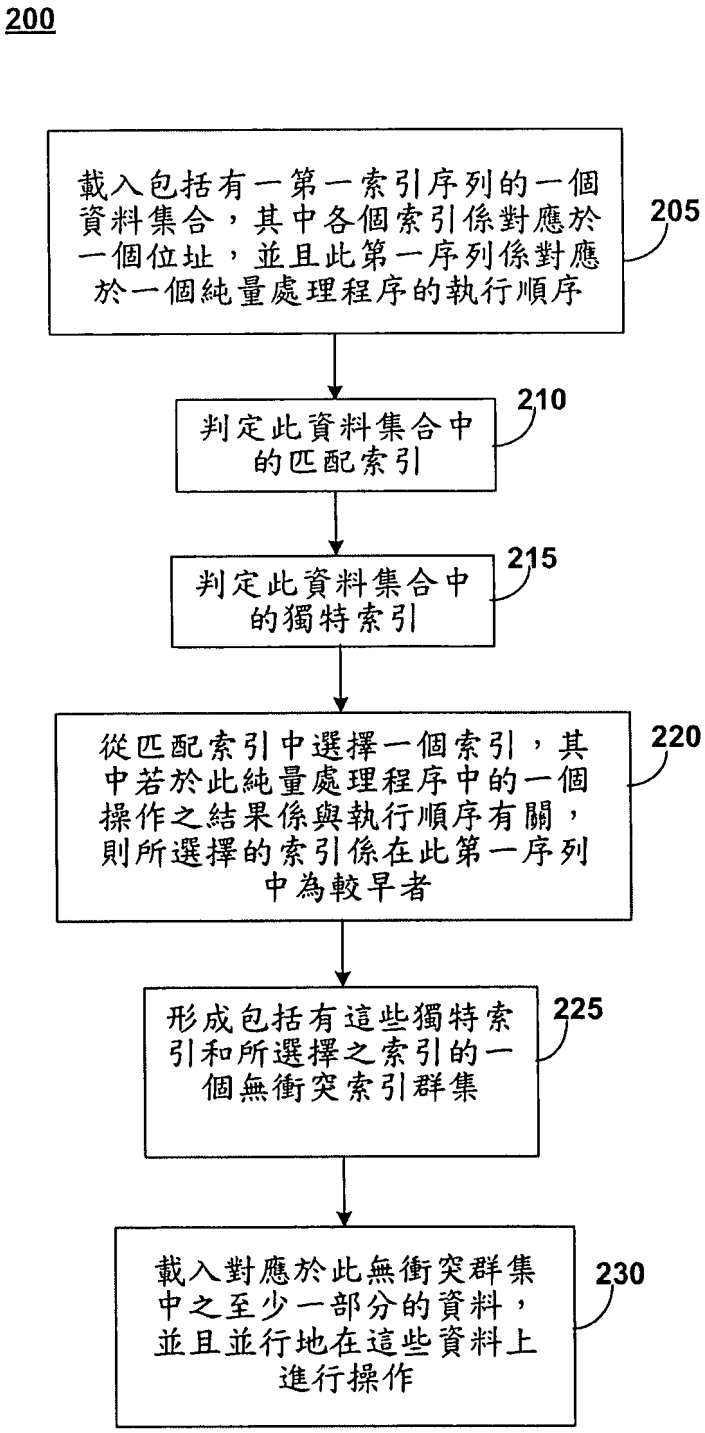
量處理程序的該執行順序。

20. 如申請專利範圍第19項之設備，其中該處理器進一步係組配來進行下列步驟：

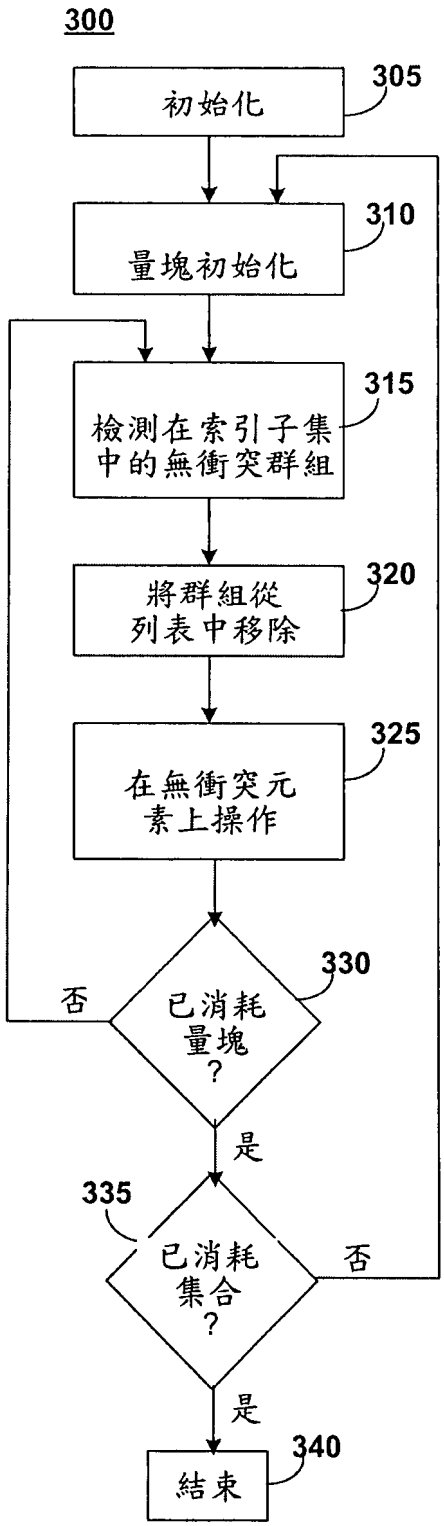
檢測在由索引組成之該第一序列與該第二序列之間的寫後讀關係。


100
102
純量處理單元
112
純量暫存器
106
記憶體單元
132
混洗邏輯
134
分散/收集
122
快取
124
記憶體控制器
126
記憶體
104
向量處理單元
132
混洗邏輯
134
分散/收集
114
向量暫存器

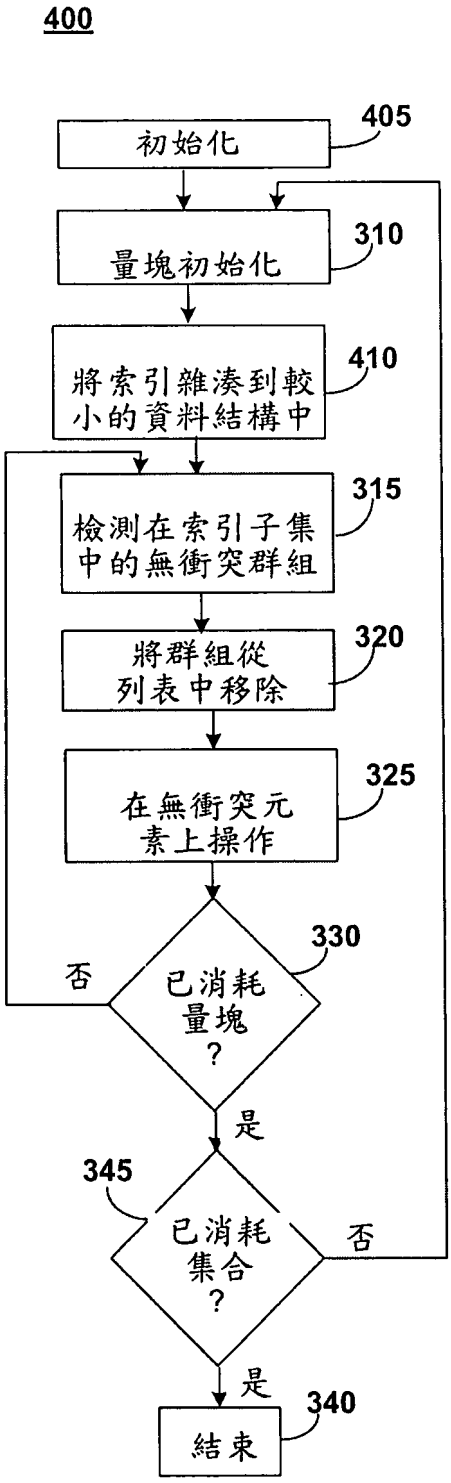
第1圖


200
載入包括有一第一索引序列的一個資料集合，其中各個索引係對應於一個位址，並且此第一序列係對應於一個純量處理程序的執行順序
205
判定此資料集合中的匹配索引
210
判定此資料集合中的獨特索引
215
從匹配索引中選擇一個索引，其中若於此純量處理程序中的一個操作之結果係與執行順序有關，則所選擇的索引係在此第一序列中為較早者
220
形成包括有這些獨特索引和所選擇之索引的一個無衝突索引群集
225
載入對應於此無衝突群集中之至少一部分的資料，並且並行地在這些資料上進行操作
230

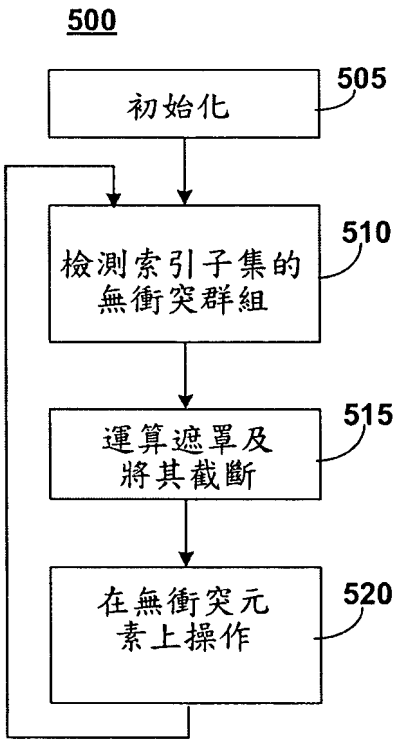
第2圖


300
初始化
305
量塊初始化
310
檢測在索引子集中的無衝突群組
315
將群組從列表中移除
320
在無衝突元素上操作
325
已消耗量塊？
330
否
是
已消耗集合？
335
否
是
結束
340

第3圖


400
初始化
405
量塊初始化
310
將索引雜湊到較小的資料結構中
410
檢測在索引子集中的無衝突群組
315
將群組從列表中移除
320
在無衝突元素上操作
325
已消耗量塊？
330
否
是
已消耗集合？
345
否
是
結束
340

第4圖



第5圖

四、指定代表圖：

(一)本案指定代表圖為：第（ 2 ）圖。

(二)本代表圖之元素符號簡單說明：

205～230...操作

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：