(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 200947343 A1
(43)公開日： 中華民國 98 (2009) 年 11 月 16 日

(21)申請案號：098105531 (22)申請日： 中華民國 98 (2009) 年 02 月 20 日
(51)Int. Cl.： *G06T1/20 (2006.01)*
(30)優先權：2008/02/22 美國 12/035,667
(71)申請人：高通公司 (美國) QUALCOMM INCORPORATED (US)
美國
(72)發明人：陳琳 CHEN, LIN (CN)
(74)代理人：陳長文
申請實體審查：有 申請專利範圍項數：50 項 圖式數：8 共 64 頁

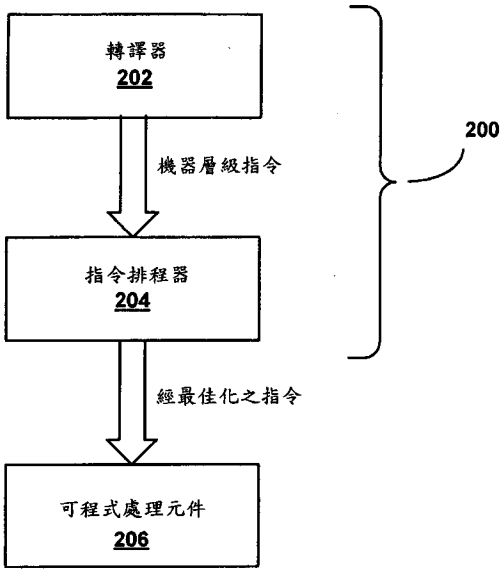
(54)名稱

圖像處理中用於減低指令潛時之系統及方法

SYSTEM AND METHOD FOR INSTRUCTION LATENCY REDUCTION IN GRAPHICS PROCESSING

(57)摘要

本發明揭示一種系統、方法及裝置，其中一編譯器(例如，一著色器編譯器)之一指令排程器基於相依的前導指令與後繼指令之間的一經判定的指令距離而減小指令延時。



200：編譯器
202：轉譯器
204：指令排程器
206：可程式處理元件

(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 200947343 A1
(43)公開日： 中華民國 98 (2009) 年 11 月 16 日

(21)申請案號：098105531 (22)申請日： 中華民國 98 (2009) 年 02 月 20 日
(51)Int. Cl.： *G06T1/20 (2006.01)*
(30)優先權：2008/02/22 美國 12/035,667
(71)申請人：高通公司 (美國) QUALCOMM INCORPORATED (US)
美國
(72)發明人：陳琳 CHEN, LIN (CN)
(74)代理人：陳長文
申請實體審查：有 申請專利範圍項數：50 項 圖式數：8 共 64 頁

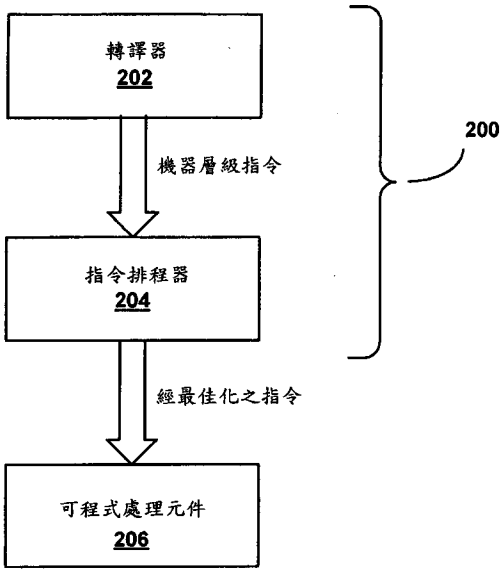
(54)名稱

圖像處理中用於減低指令潛時之系統及方法

SYSTEM AND METHOD FOR INSTRUCTION LATENCY REDUCTION IN GRAPHICS PROCESSING

(57)摘要

本發明揭示一種系統、方法及裝置，其中一編譯器(例如，一著色器編譯器)之一指令排程器基於相依的前導指令與後繼指令之間的一經判定的指令距離而減小指令延時。



200：編譯器
202：轉譯器
204：指令排程器
206：可程式處理元件

六、發明說明：

【發明所屬之技術領域】

本揭示案係關於減小用於圖形處理之程式碼中的指令延時，且更特定言之，係關於減小用於圖形處理中之著色器中的指令延時。

【先前技術】

圖形處理單元(GPU)為用以產生用於在顯示器件上顯示之電腦化圖形的專用圖形再現器件。GPU通常與通用中央處理單元(CPU)一起使用以處理圖形影像資料，例如，三維電腦化圖形影像資料。在該狀況下，GPU可實施許多基元圖形運算以比使用CPU來繪製用於在顯示器件上顯示的影像更快地產生用於在顯示器件上顯示的三維影像。通常，GPU包括以硬體實施某一數目之複雜演算法的硬體。

典型GPU接收影像幾何且使用管線方法來產生可經輸出(例如)以用於在顯示器件上顯示的圖形。典型圖形管線包括平行操作之許多階段，其中來自一階段之輸出可能用於管線中之另一階段處。舉例而言，典型圖形管線包含頂點著色器、基元組合、視埠變換(viewport transformation)、基元設定、柵格化、隱藏基元及像素拒斥(pixel rejection)、屬性設定、屬性內插法及片段著色器階段。

頂點著色器被應用於影像之影像幾何及產生頂點座標及影像幾何內之頂點的屬性。頂點屬性包括(例如)與頂點相關聯之色彩、法線及紋理座標。基元組合基於影像幾何而自頂點形成基元，例如，點、線及三角形基元。可使用變

換(例如，將基元自經正規化之器件空間變換至螢幕空間之視埠變換)而將經形成之基元自一空間變換至另一空間。基元設定可用以判定基元之區域、邊緣係數，及執行遮擋裁切(例如，隱面裁切)及3-D截割操作。

柵格化基於基元內之頂點的XY座標及包括於基元中之像素的數目而將基元轉換成像素。隱藏基元及像素拒斥使用基元及/或像素之z座標以判定及拒斥經判定為隱藏之彼等基元及像素(例如，定位於影像圖框中之一基元或像素後的另一基元或像素、透明基元或像素)。屬性設置判定與基元內之像素相關聯之屬性的屬性梯度，例如，在水平(X)方向或垂直(Y)方向上移動之基元內的第一像素處之屬性值與第二像素處之屬性值之間的差。屬性內插法基於經判定之屬性梯度值而在基元內之像素上內插屬性。將經內插之屬性值發送至片段著色器以用於像素再現。將片段著色器(fragment shader)之結果輸出至後處理區塊及圖框緩衝器以用於將經處理之影像呈現在顯示器件上。

著色器(例如，頂點著色器及片段著色器)通常為計算及控制用於圖形或其他多媒體系統中之基元(例如，頂點或像素)之屬性的電腦程式。舉例而言，通常以諸如高階或低階程式化語言之程式化語言編寫著色器。高階程式化語言可為C++程式化語言及其類似者。組合語言為低階語言之實例。

著色器編譯器充當將以高階或低階語言編寫之著色器程式碼轉譯成機器層級語言的轉譯器。在以高階語言編寫著

色器之狀況下，轉譯器將著色器程式碼自編寫著色器之高階語言轉譯成低階語言且接著將低階著色器程式碼轉譯成機器層級指令。著色器編譯器之指令排程器對著色器之機器指令重新排序以努力加快著色器執行。此外，著色器編譯器藉由插入虛設指令(例如，無操作或NOP)而解決硬體之時間限制以使著色器符合執行著色器之硬體的時序限制。

能夠最佳化著色器之指令同時考慮硬體限制將為有益的。

【發明內容】

本揭示案試圖解決該技術中之缺點及提供一或多種用於最佳化指令之排程以實施圖形處理管線(例如，諸如頂點著色器及/或片段著色器之著色器)之至少一部分的方法、裝置及電腦可讀媒體。

根據一或多項實施例，識別圖形處理指令中之兩個指令之間的相依性，兩個指令中之一者包含前導指令且兩個指令中之另一者包含後繼指令。判定與前導指令與後繼指令之間的相依性相關聯之初始邊緣延時。對應於前導指令及後繼指令之指令距離被判定，及用以將初始邊緣延時減小經判定之指令距離以判定與前導指令與後繼指令之間的相依性相關聯之經減小的邊緣延時。舉例而言，圖形處理指令可實施著色器，諸如頂點著色器或片段著色器。

根據一或多項實施例，藉由將初始邊緣延時減小經判定之指令距離而判定的經減小的邊緣延時係用以對後繼指令

之執行進行排程。根據一或多項實施例，判定在開始後繼指令的執行之前待執行之同步指令(例如，獨立著色器指令或NOP)的數目以使後繼指令與前導指令之執行同步。

根據一或多項實施例，初始邊緣延時為與前導指令相關聯之硬體延時，及/或前導指令與後繼指令之間的相依性包含流相依性，以使得前導指令之目的地為後繼指令之源。

根據一或多項實施例，識別圖形處理指令中之兩個指令之間的相依性，兩個指令中之一者包含前導指令且兩個指令中之另一者包含後繼指令。判定與前導指令與後繼指令之間的相依性相關聯之初始邊緣延時。對應於前導指令及後繼指令之指令距離被判定，及用以將初始邊緣延時減小經判定之指令距離以判定與前導指令與後繼指令之間的相依性相關聯之經減小的邊緣延時。藉由判定對應於前導指令之目的地運算元之後繼指令之每一源運算元的遮罩距離及自經判定之遮罩距離選擇最小遮罩距離作為指令距離來判定指令距離。

根據一或多項實施例，與源運算元相關聯及用於判定指令距離之遮罩距離係藉由以下動作來判定：判定前導指令之目的地運算元的分量遮罩及後繼指令之源運算元的分量遮罩；藉由串連連接目的地運算元之分量遮罩及源運算元之分量遮罩而產生分量串；使用分量串而判定與分量集合中之每一分量相關聯的分量距離；及識別經判定之分量距離中之最小分量距離以作為源運算元之遮罩距離。

根據一或多項實施例，與分量集合中之每一分量相關聯的分量距離係藉由以下動作來判定：檢查分量串以定位該分量串中之分量的第一出現；在分量之第一出現經定位之狀況下，檢查分量串以定位分量串中之分量的第二出現，第二出現在分量串中的第一出現之後；及在分量之第二出現經定位之狀況下，判定分量串中之分量之第一出現與第二出現之間的分量的數目，及將分量之分量距離設定成分量之經判定的數目。

根據一或多項實施例，圖形處理指令實施頂點著色器，分量集合包含X、Y、Z及W分量，且對於分量集合中之X、Y、Z及W分量中的每一者而判定分量距離。根據一或多項實施例，圖形處理指令實施片段著色器，分量集合包含R、G、B及A分量，且對於分量集合中之R、G、B及A分量中的每一者而判定分量距離。

已提供此簡短[發明內容]以使得可快速理解本發明之本質。可參考結合隨附圖式之本發明之較佳實施例的以下[實施方式]而獲得對本發明之更完整的理解。

【實施方式】

本揭示案之以上所提及特徵及目標參考結合隨附圖式所進行之以下描述將變得更顯而易見，其中相同參考數字表示相同元件。

現將參看以上所提及之諸圖而論述本揭示案之特定實施例，其中相同參考數字指代相同組件。

根據一或多項實施例，揭示一種系統、方法及裝置，其

中編譯器(例如，著色器編譯器)之指令排程器基於相依的前導指令與後繼指令之間的經判定的指令距離而減小指令延時。根據本揭示案之一或多項實施例，著色器編譯器包含：指令最佳化器及指令排程器，其解決硬體時序限制及最小化排程長度，例如，指令/執行排程中之指令的數目。雖然本文中關於實施著色器之指令而揭示指令排程最佳化，但應顯而易見，本揭示案之實施例不需限於實施著色器之指令的最佳化。本揭示案之實施例可用以最佳化任何指令、電腦程式、程式碼，或程式或程式碼片段之排程。藉由非限制性實例，本文中所揭示之實施例中的一或多者可與支援由多個分量(例如，兩個、三個或四個分量)組成之原生向量(native vector)之任何程式化語言及使用該程式化語言而定義之任何程式或程式片段一起使用。

圖1為說明例示性計算器件100之方塊圖，計算器件100包括用於根據本揭示案之一或多項實施例而使用的圖形處理單元(GPU)104。計算器件100可包含個人電腦、桌上型電腦、膝上型電腦、工作站、視訊遊戲平台或遊戲機、蜂巢式或衛星無線電電話、陸線電話、網際網路電話、手持式器件(諸如攜帶型視訊遊戲器件或個人數位助理)、個人音樂播放器、伺服器、中間網路器件、主機電腦，或輸出圖形資訊之另一類型之器件。

在圖1之實例中，計算器件100包括中央處理單元(CPU)102、GPU 104及記憶體模組116(例如，一或若干個隨機存取記憶體(RAM)記憶體模組)。CPU 102、GPU 104

及記憶體模組116使用匯流排106而進行通信，匯流排106可包含現已知或以後開發之任何類型之匯流排或器件互連。CPU 102可包含通用或專用微處理器。舉例而言，CPU 102可包含由California, Santa Clara之Intel公司所提供之Core 2處理器或另一類型之微處理器。GPU 104為專用圖形再現器件。GPU 104可經整合至計算器件100之主機板中，可存在於經安裝在計算器件100之主機板中的埠中的圖形卡上，或(例如)可另外經組態以與計算器件100交互操作。

耦接至計算器件100之顯示單元124可包含(例如)監視器、電視、投影器件、液晶顯示器、電漿顯示面板、發光二極體(LED)陣列、陰極射線管顯示器、電子紙、表面傳導電子發射顯示器(SED)、雷射電視顯示器、奈米晶體顯示器，或另一類型之顯示單元。在圖1之實例中，顯示單元124可為計算器件100之一部分。舉例而言，顯示單元124可為行動電話之螢幕。或者，顯示單元124可在電腦器件100之外部，且可經由(例如)有線或無線通信連接或其他連接而與計算器件100進行通信。藉由一非限制性實例，顯示單元124可為經由有線或無線連接而連接至個人電腦的電腦監視器或平板顯示器。

可經由CPU 102來執行軟體應用程式110。藉由非限制性實例，軟體應用程式110可包含能夠經由CPU 102執行之任何軟體應用程式，諸如，視訊遊戲、圖形使用者介面引擎、用於工程或藝術應用之電腦輔助設計程式，或使用二

維(2D)或三維(3D)圖形之另一類型之軟體應用程式。

在CPU 102正執行軟體應用程式110時，藉由非限制性實例，軟體應用程式110可調用圖形處理應用程式化介面(API)112之副常式，諸如OpenVG API、OpenGL API、Direct3D API、圖形器件介面(GDI)、Quartz、QuickDraw，或另一類型之2D或3D圖形處理API中之任一或多者。

根據至少一實施例，在軟體應用程式110調用圖形處理API 112之副常式時，圖形處理API 112調用GPU驅動程式114之一或多個副常式，其經由計算器件100上之CPU 102而執行。GPU驅動程式114可包含(例如)在圖形處理API 112與GPU 104之間提供介面之軟體及/或韌體指令的集合。在圖形處理API 112調用GPU驅動程式114之副常式時，GPU驅動程式114編製並發布使GPU 104產生可顯示圖形資訊之命令。根據本文中所揭示之一或多項實施例之著色器編譯器可為GPU驅動程式114之組件，例如，軟體模組。GPU驅動程式114使用著色器編譯器將著色器程式轉譯成機器層級指令及將該等指令傳達至GPU 104。舉例而言，在圖形處理API 112調用GPU驅動程式114之副常式以再現一批次圖形基元時，GPU驅動程式114將處理組態提供給GPU 104，GPU 104使用該處理組態以再現該批次圖形基元。GPU 104再現該批次圖形基元，且(例如)輸出圖形基元之光柵圖形。

由GPU驅動程式114編製之命令可識別GPU 104用以執行該命令之圖形處理組態，該(等)組態可識別待由GPU 104

執行之指令的集合、狀態暫存器值的集合，及GPU 104執行該命令可能需要之其他類型的資訊。

在GPU驅動程式114將該(等)圖形處理組態儲存在記憶體116中的狀況下，GPU驅動程式114可參考對應於由GPU驅動程式114編製之命令中的該(等)圖形處理組態的記憶體模組116中的儲存位置。在GPU 104接收該命令時，GPU 104可自記憶體116擷取在自GPU驅動程式114接收到之該命令中所參考的該(等)圖形處理組態。

根據至少一實施例，GPU 104之命令解碼器126對來自GPU驅動程式114之命令進行解碼，且組態處理元件128中之一或多者以執行命令。藉由非限制性實例，命令解碼器126自記憶體116擷取該(等)圖形處理組態，且將由該(等)圖形處理組態所識別之指令的集合載入至處理元件128中。命令解碼器126亦可經組態以將輸入資料提供至一或多個處理元件128。

根據一或多項實施例，處理元件128實施圖形管線108。根據該等實施例，處理元件128可在平行模式中實施圖形管線108。在平行模式中，處理元件128可平行對資料操作，其中自處理元件128之輸出用作至另一處理元件128之輸入。藉由非限制性實例，處理元件128A對自命令解碼器126所接收到之初始輸入資料的第一集合執行第一圖形操作，且將中間結果之第一集合輸出至處理元件128B。初始輸入資料可包含對應於一或多個頂點之資料，該資料可包含(例如)座標及屬性資料。頂點座標基於(例如)具有X、Y

及Z(寬度、高度及深度)座標及包含透視參數之W座標的四維座標系而識別影像內的位置。頂點屬性可包括(例如)與頂點相關聯之色彩、法線及紋理座標。處理元件128B可對由處理元件128A所輸出之中間結果的第一集合執行另一圖形作，及將中間結果之第二集合輸出至處理元件128中的另一者，等等。當處理元件128B正執行第二圖形操作時，處理元件128A可正對自命令解碼器126所接收到之初始輸入資料的第二集合執行第一圖形操作。

處理元件128可以此方式繼續，直至處理元件128N將像素物件輸出至記憶體模組116中之一或多個緩衝器或將此新像素物件輸出至某一其他目的地。像素物件為描述像素之資料。每一像素物件可指定多個色彩值，且可指定像素之透明度位準。在一些情況下，像素物件可指定在第一色彩格式中之第一色彩及在第二色彩格式中之第二色彩。

根據本揭示案之一或多項實施例，處理元件128中之一者包含：可程式處理元件，其可經組態為執行一或多個頂點著色操作之頂點著色器單元，該等頂點著色操作中之每一者對頂點資料(例如，X、Y、Z及W分量資料)操作。類似地，處理元件128中之同一者或另一者包含：可程式處理元件，其可經組態為執行一或多個片段著色操作之片段著色器，該等片段著色操作中之每一者對像素資料(例如，R、G及B分量資料)操作。

根據本揭示案之一或多項實施例，編譯器產生包括待由可程式處理元件執行以執行著色器操作(例如，頂點著色

器或片段著色器操作)之指令的程式碼。圖2提供對程式碼(例如，著色器程式碼)進行編譯之編譯器的實例。編譯器200包含至少一轉譯器202。轉譯器202將包括以高階程式化語言或組合語言編寫之指令之集合的著色器程式碼轉譯成可由可程式處理元件206辨認之機器層級指令。編譯器200之指令排程器204對用於由可程式處理元件206執行之指令進行排程。

將待排程之包含器件層級或機器可執行指令之群組的輸入清單輸入至指令排程器204。將輸入清單中之不取決於任何其他指令或與任何其他指令衝突的指令自輸入清單移動至就緒清單。就緒清單儲存準備好被排程以用於執行之所有指令。指令在其被排程時自就緒清單移動至現用清單。現用清單儲存當前正被執行之指令。指令在其完成執行時自現用清單移動至結果清單。結果清單儲存已被排程及已完成執行之指令。結果清單包含指令排程器之輸出。

使用識別靜態延時之相依圖來計算控制何時將指令自一清單移動至另一清單的閘控條件(gating condition)。藉由非限制性實例，若一指令I2依賴另一指令I1之結果，或資源衝突存在於兩個指令之間，則相依性存在於兩個指令I1與I2之間。若相依性存在於兩個指令之間且指令I1領先I2，則認為指令I1為前導指令且指令I2為後繼指令。相依圖可用以指定兩個指令之間的相依性。

圖3提供指示相依性存在於前導指令與後繼指令之間的相依圖或其子集的實例。相依圖中之邊緣306可用以表示

相依性。權重308可係與邊緣306相關聯以表示解決相依性所需之指令執行循環的數目或計數。將權重稱作邊緣延時。初始使用輸入清單將邊緣延時計算為靜態延時，例如與前導指令相關聯之硬體延時。動態延時為指令在現用清單中之時間跨度(例如以指令/執行循環量測)。

前導指令P1可具有數目n之後繼指令S1、.....、Sn，其中n可大於或等於零。每一後繼指令S可具有數目n之包括前導指令P1的前導指令P1、.....、Pn。

在後繼指令之動態延時大於或等於其靜態或邊緣延時中之一者時，可自前導指令之後繼清單移除後繼指令，且可自後繼指令之前導清單移除前導指令。隨著時間前進，例如，執行循環出現，且動態延時增大，可移除更多後繼及前導指令。在自後繼清單移除指令之所有後繼指令後，在現用清單上之指令係完成的，且在移除指令之所有前導指令後，輸入清單上之指令已準備好被執行。因此，能夠最小化靜態邊緣延時(例如，加快後繼指令之執行的時序)為有益的。有利地，使用本揭示案之一或多項實施例所識別之邊緣延時的減小改良了排程品質及導致更緊湊的碼。

編寫著色器之程式化語言可包括滿足特殊圖形及多媒體需要之特殊語言結構。特殊著色器程式化語言及通用程式化語言(例如，C++及其類似者)支援類似資料類型、陣列、構造(struct)、語句及函數。雖然著色器程式化語言可能不如通用語言一樣靈活且可能對一些通用特徵具有一些限制，但著色器程式化語言具有在通用語言中不支援之一

些額外特徵。舉例而言，著色器語言可為由多個分量(例如，兩個、三個或四個分量)組成之原生向量提供支援。通用程式化語言通常不具有對該等向量之原生支援。

用於著色器之典型基元包含色彩及頂點，例如，諸如紅色、綠色、藍色(R、G、B)或(X、Y、Z、W)之向量。多分量向量之使用使著色器編譯器比編譯通用語言之編譯器更複雜。本揭示案之實施例使用額外資訊以減小延時，及改良指令排程。根據一或多個該等實施例，分析涉及向量及其屬性之著色器程式化語言的特徵以減小邊緣延時，以使得(例如)可減小等待執行之指令的執行時序，可改良排程，且程式碼可更緊湊。

指令之間存在許多類型之相依性。流程或真實相依性存在於兩個指令I1與I2之間，其中I2使用I1之輸出。可藉由檢查由兩個指令所參考之暫存器號及分量索引而判定真實相依性。藉由非限制性實例，指令I2為指令I1之後繼指令，且將暫存器及由指令I1所輸出之分量(例如，X分量)用作其源運算元中之至少一者。

輸出相依性在兩個指令皆輸出至同一暫存器之狀況下存在於指令I1與I2之間。藉由非限制性實例，指令I1及I2輸出至暫存器R0。控制相依性在由邏輯運算之結果判定執行流程之狀況下(例如，if-else條件)存在於指令I1與I2之間。反相依性在指令I2之輸出使用與指令I1之輸入使用之暫存器相同的暫存器之狀況下存在於指令I1與I2之間。

根據一或多項實施例，可基於與前導指令相關聯之硬體

延時而判定初始邊緣延時。在指令I1及I2相依且相依性為真實相依性之狀況下，可將完整硬體延時用作初始邊緣延時，其可被減小經判定之指令距離。根據一或多個該等實施例，執行額外分析以判定經初始判定為前導指令之硬體延時的邊緣延時是否可被減小經判定之指令距離。本揭示案之實施例可用以在不同於流相依性及固定硬體延時存在之狀況的狀況下設定邊緣延時。舉例而言，根據一或多項實施例，若假相依性存在，諸如反相依性或輸出相依性，則可將邊緣延時設定成一。

通常由執行指令之可程式單元或其他處理單元之硬體設計者識別硬體延時。可提供硬體延時以作為延時表之部分，該延時表識別可由可程式單元執行之指令中之每一者的硬體延時。以下提供用於判定硬體延時之形式化方程式：

HARDWARE_LATENCY(I1, I2)=純量指令I1及I2之由硬體設計者提供的延時。

以下提供以上方程式之實例，其中純量指令I1及I2皆為ADD，且I1為ADD指令且I2為BRANCH指令。

HARDWARE_LATENCY(ADD, ADD)=6

HARDWARE_LATENCY(ADD, BRANCH)=10

以下提供本文中所使用之指令語法的非限制性實例。在此實例中，假設流相依性存在於指令I1與I2之間，以使得n對於指令I1及I2兩者為同一值，且I1定義指令I2中所使用之至少一分量。

I1: (*i) def Rn.c

I2: (*j) use Rn.d

以上指令I1定義具有分量索引c的暫存器Rn，且指令重複(i+1)次。指令I2使用具有分量索引d的暫存器Rn，且指令重複(j+1)次。舉例而言，分量索引可指向X、Y、Z或W中之一或多者。作為此非限制性實例之部分，假設運算元分量索引每當指令中之一者或另一者重複時遞增。以上指令之向量形式可用以判定兩個指令之間的相依性。以下提供由以上純量指令之變換導致的向量形式：

I1: def Rn.mask1

I2: use Rn.mask2

Mask1及mask2識別由指令I1及I2之各別分量索引所指向的組件。由於指令I1及I2皆對暫存器n操作，因此對分量遮罩mask1及mask2之檢查可判定指令I2是否依賴指令I1。分量遮罩可含有一或多個分量。分量之實例包括(但不限於)RGB及XYZW。分量遮罩之實例包括(但不限於)XYZW、YZWX、XY、Z。若mask1及mask2在其共用至少一分量之意義上重疊，則存在相依性。若存在真實或流相依性，則將完整機器延時用作初始邊緣延時。自I1至I2之相依性邊緣的初始邊緣延時為：

初始延時(I1, I2)=硬體延時(I1, I2)　　方程式1

根據一或多項實施例，方程式1識別初始邊緣延時，該初始邊緣延時可減小兩個相依指令之間的經判定之距離(指令距離)，如以下所形式化之方程式2中所說明：

邊緣延時(I1, I2)=硬體延時(I1, I2)-指令距離　　方程式2

根據一或多項實施例，可使用分量遮罩、分量串、分量距離及分量遮罩距離來判定指令距離。分量遮罩為自經壓縮之純量指令之運算元所建構的遮罩。經壓縮之純量指令可藉由遞增分量索引或保持索引而重複自身。以下提供經壓縮之純量指令的實例：

(*2) ADD R0.X, (*)R1.Y, R2.X,

其中在*之後的2指示在指令之初始執行之後重複指令兩次，亦即共執行指令三次。與第一運算元R0及第二運算元R1相關聯之*指示對應於該兩個運算元之分量索引將在指令之第一及第二執行之後遞增。在該實例中，初始執行指令一次，且重複兩次。經壓縮之純量指令為接下來三個未經壓縮之純量指令的等效物：

E1:ADD R0.X, R1.Y, R2.X

E2:ADD R0.Y, R1.Z, R2.X

E3:ADD R0.Z, R1.W, R2.X

未經壓縮之指令的第一執行E1分別使用暫存器R0、R1及R2以及X、Y及X分量。未經壓縮之指令的執行E1導致相關聯於R1及R2暫存器的X及Y分量值正被相加，且結果正被儲存於X分量(與暫存器R0相關聯)中。使與R0及R1暫存器相關聯之分量索引遞增，如由與R0及R1暫存器中之每一者相關聯的*所指示。在未經壓縮之指令的第二執行E2中，與R0及R1暫存器相關聯之分量分別為Y及Z分量。未改變與R2暫存器相關聯之分量，亦即，X分量。第二執

行E2使與R1及R2暫存器相關聯之Z及X分量值相加，且結果儲存於與R0暫存器相關聯之Y分量中。在遞增與R0及R1暫存器相關聯之分量索引，以使得與R0及R1暫存器相關聯之分量(分別)為Z及W之後，未經壓縮之指令的第三執行E3使與暫存器R1及R2相關聯之W及X分量值相加，且將結果儲存於與R0暫存器相關聯之Z分量中。

暫存器R0、R1及R2中之每一者的分量遮罩分別為XYZ、YZW及X。為了進一步說明，使用用於未經壓縮之純量指令之每一執行E1至E3中之R0暫存器的分量(亦即，分別為X、Y及Z)來形成R0暫存器的分量遮罩。類似地，由用於指令執行E1至E3中之Y、Z及W分量形成與R1暫存器相關聯的分量遮罩，且藉由將暫存器之X分量用於指令執行E1至E3中之所有三者中而形成與R2暫存器相關聯的分量遮罩。

根據一或多項實施例，與用於向量指令中之拌和遮罩(swizzle mask)不同，自純量指令所建構的在遮罩中具有重複分量之遮罩為無效分量遮罩。舉例而言及根據該等實施例，XXY為有效的拌和遮罩，但並非為有效的分量遮罩。

藉由串連連接分量遮罩(例如，來自前導及後繼指令的兩個分量遮罩)而產生分量串。舉例而言，分量遮罩XYZW及XY之串連連接形成分量串XYZWXY。可將分量距離COMPONENT_DIST(C,S)(其中C表示分量，例如，X、Y、Z或W中之一者，且S表示分量串)定義為分量串S中之分量C之兩次出現之間的分量(非分量C)出現的數目。若分量串

S中之分量C不出現或出現一次，則距離為正無限+INF。

以下提供將分量串XYZWXY用作S及將分量X、Y、Z及W用作C的分量距離計算的實例：

COMPONENT_DIST (X, XYZWXY)=3

COMPONENT_DIST (Y, XYZWXY)=3

COMPONENT_DIST (Z, XYZWXY)=+INF

COMPONENT_DIST (W, XYZWXY)=+INF

在使用分量X之第一分量距離判定中，X出現兩次，且三個分量YZW存在或出現於分量串中之X之兩次出現之間。在使用分量Y之第二分量距離判定中，存在出現於Y之出現之間的三個分量YZW。在涉及Z及W之接下來兩個分量距離判定中，分量串中之此等分量中的兩者之出現僅有一次。因此將分量距離設定成+INF。

可將分量遮罩距離判定為使用藉由串連連接分量遮罩M1及M2而形成之分量串所判定之分量距離中的最小分量距離。以下方程式3提供根據一或多項實施例之分量遮罩距離判定的形式化：

MASK_DIST (M1, M2)=最小(COMP_DIST (X, M), COMP_DIST (Y, M), COMP_DIST(Z, M), COMP_DIST (W, M))，其中M為M1及M2的串連連接。　　方程式3

舉例而言：

由於分量遮罩XYZW及XY之串連連接為XYZWXY且X、Y、Z及W之分量距離分別為3、3、+INF及+INF，因此MASK_DIST (XYZW, XY)=3。

由於分量遮罩YZWX及XY之串連連接為YZWXXY且X、

Y、Z及W之分量距離分別為0、4、+INF及+INF，因此MASK_DIST (YZWX, XY)=0。

可將運算元距離定義為使用與使用同一暫存器之源運算元及目的地運算元(例如，用於前導指令中之目的地運算元及後繼指令中之源運算元)相關聯之遮罩所判定的分量遮罩距離。運算元距離為使用方程式3所判定之遮罩距離。舉例而言，M等於與前導指令中之運算元相關聯之分量遮罩和與後繼指令中之運算元相關聯之分量遮罩的串連連接。

根據一或多項實施例，將指令距離定義為運算元(例如，共用於兩個相依指令之間的每一運算元)之遮罩距離之最小遮罩距離或運算元距離。後繼指令中可存在使用由前導指令定義之同一暫存器的一或多個源運算元。在存在使用同一暫存器之多個源運算元的狀況下，可計算多個遮罩距離，且選擇最小遮罩距離作為兩個指令之間的距離。根據一或多項實施例，兩個相依指令I1及I2之間的指令距離可如下形式化，其中I1為前導指令且I2為後繼指令：

INSTR_DIST (I1, I2)= 最小(MASK_DIST (MD, MS1)、.....、MASK_DIST (MD, MSn))，其中MD為I1之目的地運算元的分量遮罩，MSi為使用I1之目的地之I2的第i個源運算元，且n為使用I1目的地之I2源的總數目。　　方程式4

舉例而言，以下兩個指令之間的指令距離為1。

I1: ADD R1.XYZ, R2.XYZ, R3.XYZ

I2: MUL R4.X, R1.X, R1.Y

I1之輸出/目的地R1.XYZ與I2之第一及第二輸入/源之間的經判定指令距離分別為2及1。因此，可如下表達以上樣本指令中之I1與I2之間的距離：

INSTR_DIST (I1, I2)=1

如以上所論述，本揭示案之實施例將初始邊緣延時減小指令距離。以下方程式5提供根據一或多項實施例所使用之邊緣延時判定的形式化：

EDGE_LATENCY (I1, I2)=最大(1, HARDWARE_LATENCY (I1, I2)-INSTR_DIST (I1, I2))　　方程式5

可程式著色器單元支援許多指令類型，包括(但不限於)ALU、ELU、FLOW及MEM。ALU指令包含算術及邏輯指令，諸如(但不限於)ADD、MUL及MOV。ELU指令包含初等函數指令，諸如(但不限於)EXP、LOG及COS。FLOW指令包含控制流程指令，諸如(但不限於)JUMP及BRANCH。MEM指令包含記憶體定向指令，諸如(但不限於)SAMPLE及LOAD。諸如(但不限於)ALU之一些類型具有確定性硬體延時。諸如(但不限於)MEM之其他指令類型具有非確定性硬體延時。對於非確定性延時，一指令集合通常(例如)使用WAIT運算而支援同步機制。

對於具有相依關係且I1領先I2之任何兩個指令I1及I2，本揭示案之一或多項實施例如下定義邊緣延時：

在諸如(但不限於)反相依性或輸出相依性之假相依性的情況下，EDGE_LATENCY (I1, I2)=1。　　方程式6

若相依性為流相依性且I1具有非確定性延時，則EDGE_LATENCY (I1, I2)=1。　　方程式7

若相依性為流相依性且I1具有固定/確定性延時，則 方程式8
EDGE_LATENCY (I1, I2)= 最大(1, HARDWARE_LATENCY (I1, I2)-INSTR_DIST (I1, I2))。
若相依性為控制相依性而非資料相依性，則 方程式9
EDGE_LATENCY (I1, I2)=HARDWARE_LATENCY (I1, I2)。

以下提供說明以上邊緣延時形式化之使用的實例。為了舉例起見及為了說明之目的，認為ALU類型指令具有為5之固定延時及為2之指令距離，ELU類型指令具有非確定性延時，且FLOW類型指令具有為1之延時。使用例示性延時及指令距離，使用以上形式化而判定以下邊緣延時。

EDGE_LATENCY (ALU, ALU)=3

EDGE_LATENCY (ALU, ELU)=3

EDGE_LATENCY (ALU, FLOW)=3

EDGE_LATENCY (ALU, MEM)=3

在以上實例中，假設兩個指令之間存在流相依性，且因此方程式8適用。在每一實例中，指令I1為具有為5之固定延時及為2之指令距離的ALU類型指令。假設流相依性存在於兩個指令I1與I2之間，將方程式8用以判定邊緣延時。ALU指令之初始邊緣延時對應於為5之ALU類型指令之硬體/固定延時，其被減小為2的兩個指令之間的指令距離，或5-2=3。

包含圖4A及圖4B之圖4說明根據本揭示案之一或多項實施例的由指令排程器模組所執行的邊緣延時判定流程。本揭示案之實施例重複經提供至指令排程器204之輸入清單中之指令I1及I2的每一對的過程，其中指令對中之指令中

的一者(例如，I2)取決於另一者(例如，I1)。

在步驟402處，作出相依性是否存在於兩個指令I1與I2之間的判定。若不存在相依性，則過程對於當前指令對而結束，且可對於下一對指令而執行，直至不再存在待檢查之指令為止。若判定相依性存在於兩個指令之間，則處理在步驟404處繼續以判定相依性是否為流相依性及前導指令(例如，I1)是否具有固定延時。若否，則處理在圖4B之步驟420處繼續以根據方程式6、7或9而設定與兩個相依指令相關聯之邊緣延時，且對於指令對之處理結束。

若在步驟404處判定，流相依性存在於兩個相依指令之間且前導指令具有固定延時，則處理在步驟406處繼續以判定前導及後繼指令的至少一分量遮罩。在步驟408處，使用在步驟406處為前導及後繼指令判定之分量遮罩而判定分量串。處理在步驟410處繼續以使用在步驟408處所判定之分量串而判定指令距離。

圖5說明根據本揭示案之一或多項實施例的指令距離判定流程。簡要地及根據一或多項實施例，藉由檢查前導及後繼指令之共同運算元(例如，暫存器)以識別每一共同運算元的遮罩距離而判定指令距離。在前導指令與後繼指令之間的流相依性的狀況下，後繼指令之源運算元使用前導指令之目的地運算元。接著檢查為共同運算元中之每一者識別的遮罩距離以識別具有最小遮罩距離之共同運算元。將指令距離設定成等於與共同運算元相關聯之遮罩距離中的最小者。

更特定言之，在步驟502處作出是否仍要處理由前導及後繼指令共用之運算元中的任一者之判定。若如此，則處理在步驟504處繼續以識別由前導及後繼指令所共用之下一個運算元，例如，由前導指令用以儲存輸出及由後繼指令用作輸入之暫存器。在步驟506處，判定共同運算元之遮罩距離。

圖6說明用於根據本揭示案之一或多項實施例使用的遮罩距離判定流程。簡要地及根據一或多項實施例，判定：一分量遮罩，其包含與為前導指令中之目的地之暫存器相關聯的分量；及一分量遮罩，其包含與用作後繼指令中之源的同一暫存器相關聯的分量。串連連接經判定之分量遮罩以產生分量串，且使用該分量串來判定分量距離。將經判定之分量距離中的最小者用作遮罩距離。

圖7提供根據本揭示案之一或多項實施例之前導及後繼指令實例之指令距離判定的說明。前導指令I1及後繼指令I2展示於圖7之702處。暫存器R1為指令I1之目的地運算元及指令I2之源運算元。事實上，暫存器R1為指令I2之兩個運算元的源。對於源分量遮罩中之每一者，MS1及MS2，執行圖6之步驟以使得判定兩個遮罩距離，一個遮罩距離用於指令I2中之使用暫存器R1之源運算元中的每一者。如表704中所展示，對應於指令I1之目的地運算元的分量遮罩MD等於XYZ，且對應於指令I2之兩個源運算元MS1及MS2的分量遮罩分別為X及Y。

在步驟602處，關於指令I2之第一源運算元，分量遮罩

分別為MD及MS1，例如，前導指令I1及後繼指令I2之分量遮罩。在步驟604處，MD及MS1經串連連接以產生XYZX。在步驟606及608中比較分量X、Y、Z及W與經串連連接之串XYZW，以判定每一分量之分量距離。表708展示使用步驟606及608所判定之每一分量的分量距離。

在步驟602處，關於指令I2之第二源運算元，分量遮罩分別為MD及MS2，例如，前導指令I1及後繼指令I2之分量遮罩。如表704中所展示，MD等於XYZ且MS2等於Y。在步驟604處，MD及MS2經串連連接以產生XYZY。在步驟606及608中比較分量X、Y、Z及W與經串連連接之串XYZY，以判定每一分量之分量距離。表712展示使用步驟606及608所判定之每一分量的分量距離。

若在步驟606處判定所有分量經處理以判定分量距離，則處理在步驟610處繼續以自經判定之分量距離判定遮罩距離。更特定言之，在步驟610處，識別經判定之分量距離中之最小分量距離，且在步驟612處，在步驟610中經識別之最小分量距離用以設定遮罩距離。

舉例而言及參看圖7及表708，對於MD及MS1所判定之最小分量距離等於2。如方程式706中所展示，將MD及MS1之遮罩距離設定成2。藉由另一非限制性實例，如方程式710中所展示，MD及MS2之遮罩距離等於1，亦即，在表714中所展示之最小分量距離。

再次參看圖5，若在步驟502處判定已處理在前導指令與後繼指令之間所共用之所有運算元，則處理在步驟508處

繼續以判定指令距離。更特定言之，在步驟508處，識別經判定之遮罩距離中的最小遮罩距離。藉由非限制性實例及參考圖7之方程式706及710，對應於後繼指令I2之第二源運算元的遮罩距離比對應於後繼指令I2之第一源運算元的遮罩距離小。在圖5之步驟510處，將在步驟508中所識別之最小遮罩距離用以設定指令距離。在圖7中所展示之實例中，在步驟510處將指令距離設定成1。

再次參看圖4，在步驟412處判定前導指令之硬體延時與經判定之指令距離之間的差異。在步驟414處作出經判定之差異是否超過一之判定。若未超過，則處理在步驟422處繼續以將邊緣延時設定成一，且處理對於當前指令對結束。若在步驟414處判定經判定之差異超過一，則處理在步驟416處繼續以將兩個相依指令之間的邊緣延時設定成前導指令之硬體延時與經判定的指令距離之間的差異。

如圖4之實例流程中所展示，本揭示案之實施例在前導指令與後繼指令之間的相依性並非流相依性或前導指令不具有固定硬體延時之狀況下將邊緣延時設定成等於一。以下提供一些非限制性實例。

在以下實例中，指令I1為具有實例中之非確定性延時之ELU類型指令。由於指令I1具有非確定性延時，因此方程式7用於以下實例中之每一者中，以使得無論初始邊緣延時之值如何，皆將邊緣延時設定成1。

EDGE_LATENCY (ELU, ALU)=1

EDGE_LATENCY (ELU, ELU)=1

EDGE_LATENCY (ELU, FLOW)=1

EDGE_LATENCY (ELU, MEM)=1

在下一實例中，指令I1為假設具有與指令I2之相依性的MEM類型指令。在此實例中及藉由此等假設，使用方程式6，以使得無論初始邊緣延時之值如何，皆將邊緣延時設定成1。

EDGE_LATENCY (MEM, ALU)=1

EDGE_LATENCY (MEM, ELU)=1

EDGE_LATENCY (MEM, FLOW)=1

EDGE_LATENCY (MEM, MEM)=1

在下一實例中，指令I1(FLOW類型指令)具有為1之硬體延時及控制相依性，而非資料相依性。在此實例中，使用方程式9及指令I1之硬體延時，以使得將邊緣延時設定成1，初始邊緣延時。

EDGE_LATENCY (FLOW, ALU)=1

EDGE_LATENCY (FLOW, ELU)=1

EDGE_LATENCY (FLOW, FLOW)=1

EDGE_LATENCY (FLOW, MEM)=1

根據一或多項實施例，在不存在可插入之有用指令(例如，實施著色器之一部分的獨立指令)之狀況下，指令排程器204插入許多NOP或WAIT指令以使指令執行同步。以下提供使用以上所論述之例示性初始邊緣延時、指令距離及相依性而排程之著色器碼的實例。在涉及以上所論述之ALU指令的實例中，將為5之初始邊緣延時減小為2之指令

距離，以使得使用方程式8連同為ALU類型指令之第一指令I1及為ELU、FLOW及MEM類型指令中之一者的第二指令I2而判定所得邊緣延時等於3。為3之邊緣延時指示在執行第二ALU指令之前將存在三個執行循環。並非對於為5之硬體延時所需的4個NOP，在以下情形1至4中，將兩個NOP插入於兩個指令I1與I2之間。第一指令佔據一執行循環，兩個NOP指令佔據最後兩個執行循環，以使得可適應為3之邊緣延時。此情形造成減小2個NOP(其減小執行時間)，造成執行較少指令，等等。

情形1：ALU指令取決於ALU指令：

ALU

NOP

NOP

ALU

情形2：ELU指令取決於ALU指令：

ALU

NOP

NOP

ELU

情形3：FLOW指令取決於ALU指令：

ALU

NOP

NOP

FLOW

情形4：MEM指令取決於ALU指令：

ALU

NOP

NOP

MEM

情形5至8涉及作為前導指令之ELU指令。在該實例中，ELU指令具有非確定性延時，該非確定性延時根據方程式7而導致邊緣延時經設定成1。由於該延時為非確定性的，例如，不存在要在執行後繼指令之前執行之可判定數目的執行循環或NOP，因此後繼指令等待ALU執行之執行的完成。

情形5：ALU指令取決於ELU指令：

ELU

(等待)ALU

情形6：ELU指令取決於ELU指令：

ELU

(等待)ELU

情形7：FLOW指令取決於ELU指令：

ELU

(等待)FLOW

情形8：MEM指令取決於ELU指令：

ELU

(等待)MEM

在情形9至12中，將前導指令與後繼指令之間的相依性

視作假相依性，該假相依性造成根據方程式7而將邊緣延時設定成1。由於前導指令與後繼指令之間的相依性為假相依性，因此後繼指令等待前導MEM指令之執行的完成。

情形9：ALU指令取決於MEM指令：

MEM

(等待)ALU

情形10：ELU指令取決於MEM指令：

MEM

(等待)ELU

情形11：FLOW指令取決於MEM指令：

MEM

(等待)FLOW

情形12：MEM指令取決於MEM指令：

MEM

(等待)MEM

在情形13至16之實例中，將該相依性視作控制相依性，此造成根據方程式9而將邊緣延時設定成1。

情形13：ALU指令取決於FLOW指令：

FLOW

ALU

情形14：ELU指令取決於FLOW指令：

FLOW

ELU

情形15：FLOW指令取決於FLOW指令：

FLOW

FLOW

情形16：MEM指令取決於FLOW指令：

FLOW

MEM

本揭示案之實施例減小初始邊緣延時及移除不必要的NOP。舉例而言，在情形1至4中，將等於5之初始邊緣延時減小2至3。替代使用在初始邊緣延時之狀況下將需要的4個NOP，僅使用2個NOP，從而移除兩個不必要的NOP。典型著色器為計算密集的且包括大量情形1至4。因此，可使用本揭示案之實施例而達成執行資源的顯著節省。

根據本揭示案之一或多項實施例，指令排程器比較邊緣延時與動態延時，以便移除後繼指令及前導指令。較小的邊緣延時造成對後繼指令及前導指令的較早移除。可移除之後繼指令及前導指令越多，可視作為獨立的及因此準備好被排程之指令越多。準備好被排程之指令越多，越不需要將NOP插入至程式碼以補償硬體延時。需要插入至經排程之程式碼中之NOP之數目的減小造成經排程及執行之程式碼的高效率。在程式碼實施著色器之狀況下，本揭示案之實施例可用以(例如)：識別邊緣延時之減小；最佳化指令排程器之排程輸出；及最佳化著色器之執行。以上所論述及以下所重製之實例用以提供說明性實例：

I1: ADD R1.XYZ, R2.XYZ, R3.XYZ

I2: MUL R4.X, R1.X, R1.Y

在ADD與MUL之間的硬體延時為3之狀況下，在不使用本揭示案之實施例的情況下，經排程之程式碼將係如下：

I1: ADD R1.XYZ, R2.XYZ, R3.XYZ

NOP

NOP

I2: MUL R4.X, R1.X, R1.Y

如以上所論述，使用至少一實施例而判定之指令距離為1。藉由使用本揭示案之一或多項實施例，可判定邊緣延時為硬體延時減去指令距離。因此，替代以上經排程之程式碼實例中所需之兩個NOP，僅一個NOP用於以下經排程之程式碼實例中。如以下指令排程器輸出中所說明，將NOP之數目簡約至一個NOP，而非以上情形中之兩個NOP：

I1: ADD R1.XYZ, R2.XYZ, R3.XYZ

NOP

I2: MUL R4.X, R1.X, R1.Y

NOP之數目的減小為將指令距離應用於初始邊緣延時(例如，硬體延時)之結果。如以上所論述，根據一或多項實施例對於此等兩個指令所判定之指令距離等於1。根據一或多項該實施例，使前導指令之硬體延時減小經判定之指令距離，以得出兩個指令之間的經減小之邊緣延時。作為經減小之邊緣延時的結果，NOP之數目可自二減小至一。邊緣延時對應於兩個執行循環，其中I1之執行對應於該兩個循環中之一者且NOP對應於第二執行循環。有利

地，本揭示案之實施例減小經判定為不必要之NOP的數目，以使得指令排程器可僅輸出對達成兩個相依指令之間的同步為必需之彼等NOP。

包含圖8A至8F之圖8提供實施結合本揭示案之一或多項實施例所論述之著色器之指令的實例。圖8A及圖8B提供實施至指令排程器之著色器輸入之指令集合的實例。圖8C及圖8D提供在無與本揭示案之一或多項實施例相關聯之處理的益處之情況下的指令排程器的輸出的實例。圖8E及圖8F提供受益於與本揭示案之一或多項實施例相關聯之處理之指令排程器的輸出的實例。

在圖8中所展示之實例中，使用根據本揭示案之一或多項實施例之邊緣延時減小的指令排程器將NOP的數目自圖8C及圖8D中所展示的輸出中的48個有效地減小至圖8E及圖8F中所展示的輸出中的40個，自指令排程器之著色器指令輸出減少了8個不必要的NOP。

可結合指令37、44、53及60來觀看NOP之數目之減小的實例。在每一狀況下，NOP之數目自3減小至1。為了說明使用指令37，在不使用邊緣延時減小之情況下，(如圖8C及圖8D中所展示)指令排程器之程式碼片段輸出係如下：

```
36(154) type3: (rpt1/syn)fmac r18.z, (r)r16.y, (r)r15.y
37(219) type0: (rpt2)nop
38(157) type3: (lock)fmul dummy, r18.x, r18.x
39(158) type3: (rpt1)fmac r20.x, (r)r18.y, (r)r18.y
```

在程式碼片段812中，指令39之兩源運算元皆使用指令

36之結果，亦即，r18.z。在fmac之硬體延時為4之狀況下，在不使用本揭示案之一或多項實施例之情況下，將為4之初始邊緣延時用以判定在指令37處將插入3個NOP，以便確保滿足為4之延時要求。指令37執行第一NOP及接著兩次重複以產生三個執行循環。在執行指令36的情況下，在執行指令38之前出現四個執行循環。

相比之下，藉由本揭示案之一或多項實施例，減小邊緣延時，使得僅使用一個NOP。如下重製來自圖8E之程式碼822：

36(154) type3: (rpt1/syn)fmac r18.z, (r)r16.y, (r)r15.y

37(219) type0: nop

38(157) type3: (lock)fmul dummy, r18.x, r18.x

39(158) type3: (rpt1)fmac r20.x, (r)r18.y, (r)r18.y

指令36及39為fmac，或浮點乘法及累積。指令39取決於指令36，由於其分量遮罩(Z)及(YZ)共用至少一分量。在應用一或多項實施例時，判定作為指令I1之指令36與作為指令I2(其取決於I1及繼指令I1之後)之指令39之間的指令距離。除了具有為1之分量距離的Z之外，分量距離(C, S)對於分量中之每一者為+INF。指令距離為分量距離中之最小者，或1。因為MASK_DIST(Z, YZ)為1，所以判定指令距離為1。因為fmac不遞增分量索引，所以fmac之目的地遮罩為Z而非ZW。在38處具有一個有用指令之情況下，僅需要一個NOP。

在一或多個例示性實施例中，所描述之功能可以硬體、

軟體，及/或韌體，或其任何組合來實施。若以硬體實施，則該等功能可實施於一或多個處理器、微控制器、數位信號處理器(DSP)、特殊應用積體電路(ASIC)、場可程式化閘陣列(FPGA)或其類似者中。該等組件可駐留於通信系統、資料寫入及/或讀取系統，或其他系統內。若以軟體實施，則可將該等功能作為一或多個指令或程式碼而儲存於電腦可讀媒體上或經由電腦可讀媒體來傳輸。電腦可讀媒體包括有形電腦儲存媒體與通信媒體(包括促進電腦程式自一處轉移至另一處之任何媒體)。儲存媒體可為可由電腦存取的任何可用媒體。藉由實例且非限制，該電腦可讀媒體可包含RAM、快閃記憶體、唯讀記憶體(ROM)、電可擦可程式唯讀記憶體(EEPROM)、緊密光碟-唯讀記憶體(CD-ROM)或其他光學碟片儲存器、磁碟儲存器或其他磁性儲存器，或可用於以指令或資料結構之形式儲存所要程式碼構件且可由電腦存取的任何其他媒體。亦可將術語「電腦可讀媒體」定義為有形電腦程式產品。如本文中所使用之磁碟及光碟包括緊密光碟(CD)、雷射光碟、光碟、數位化通用光碟(DVD)、軟性磁碟及藍光光碟，其中「磁碟」通常以磁性方式重製資料，而「光碟」藉由雷射以光學方式重製資料。上文中之組合亦應包括於電腦可讀媒體之範疇內。

雖然已依據目前視作最實用及最佳實施例而描述該等裝置及方法，但應理解，本揭示案不需限於所揭示之實施例。意欲涵蓋包括於申請專利範圍之精神及範疇內之各種

修改及類似排列，申請專利範圍之範疇應符合最寬泛之解釋以使得包含所有該等修改及類似結構。本揭示案包括以下申請專利範圍中之任何及所有實施例。

【圖式簡單說明】

圖1為說明用於根據本揭示案之一或多項實施例使用之例示性器件的方塊圖。

圖2提供根據本揭示案之一或多項實施例之著色器編譯器的例示性方塊圖。

圖3提供用於根據本揭示案之一或多項實施例使用之相依圖的實例。

圖4，包含圖4A及圖4B，說明根據本揭示案之一或多項實施例的由指令排程器模組所執行的邊緣延時判定流程。

圖5說明根據本揭示案之一或多項實施例的指令距離判定流程。

圖6說明根據本揭示案之一或多項實施例的遮罩距離判定流程。

圖7提供根據本揭示案之一或多項實施例之前導及後繼指令實例之指令距離判定的說明。

圖8，包含圖8A至8F，提供實施結合本揭示案之一或多項實施例所論述之著色器之指令的實例。

【主要元件符號說明】

100	計算器件/電腦器件
102	中央處理單元(CPU)
104	圖形處理單元(GPU)

106	匯流排
108	圖形管線
110	軟體應用程式
112	圖形處理應用程式時間介面
114	GPU驅動程式
116	記憶體模組/記憶體
124	顯示單元
126	命令解碼器
128A	處理元件
128B	處理元件
128N	處理元件
200	編譯器
202	轉譯器
204	指令排程器
206	可程式處理元件
302	前導指令
304	後繼指令
306	邊緣
308	權重

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：98105531

※ 申請日：98、2、20　　※IPC 分類：~~G06T~~ G06T 1/20 (2006.01)

一、發明名稱：（中文/英文）

圖形處理中用於減小指令延時之系統及方法

SYSTEM AND METHOD FOR INSTRUCTION LATENCY REDUCTION IN GRAPHICS PROCESSING

二、中文發明摘要：

本發明揭示一種系統、方法及裝置，其中一編譯器(例如，一著色器編譯器)之一指令排程器基於相依的前導指令與後繼指令之間的一經判定的指令距離而減小指令延時。

三、英文發明摘要：

A system, method and apparatus are disclosed, in which an instruction scheduler of a compiler, e.g., a shader compiler, reduces instruction latency based on a determined instruction distance between a dependent predecessor and successor instructions.

七、申請專利範圍：

1. 一種方法，其包含：

識別圖形處理指令中之兩個指令之間的一相依性，該兩個指令中之一者包含一前導指令且該兩個指令中之另一者包含一後繼指令；

判定與該前導指令及該後繼指令之間的該相依性相關聯之一初始邊緣延時；

判定對應於該前導指令及該後繼指令之一指令距離；及

將該初始邊緣延時減小該經判定之指令距離以判定與該前導指令與該後繼指令之間的該相依性相關聯之一經減小的邊緣延時。

2. 如請求項1之方法，其進一步包含：

使用該經減小的邊緣延時而對該後繼指令之執行進行排程。

3. 如請求項1之方法，其進一步包含：

判定在開始該後繼指令的執行之前待執行之同步指令的一數目以使該前導指令及該後繼指令之執行同步，使用該經減小的邊緣延時判定同步指令的該數目，每一同步指令為一獨立指令或一NOP。

4. 如請求項1之方法，其中該等圖形處理指令實施一著色器。

5. 如請求項4之方法，其中該著色器為一頂點著色器。

6. 如請求項4之方法，其中該著色器為一片段著色器。

7. 如請求項1之方法，其中該初始邊緣延時為與該前導指

令相關聯之一硬體延時。

8. 如請求項1之方法，其中該相依性包含一流相依性，以使得該前導指令之一目的地為該後繼指令之一源。

9. 如請求項1之方法，其中判定一指令距離進一步包含：

判定對應於該前導指令之一目的地運算元之該後繼指令之每一源運算元的一遮罩距離；

自該等經判定之遮罩距離選擇一最小遮罩距離作為該指令距離。

10. 如請求項9之方法，其中判定該後繼指令之每一源運算元的一遮罩距離進一步包含：

判定該前導指令之該目的地運算元的一分量遮罩及該後繼指令之該源運算元的一分量遮罩；

藉由串連連接該目的地運算元之分量遮罩及該源運算元之分量遮罩而產生一分量串；

使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離；及

識別該等經判定之分量距離中之一最小分量距離以作為該源運算元之該遮罩距離。

11. 如請求項10之方法，其中使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離進一步包含：

對於該分量集合中之每一分量：

檢查該分量串以定位該分量串中之該分量的一第一出現；

在該分量之一第一出現經定位之一狀況下，檢查該

分量串以定位該分量串中之該分量的一第二出現，該第二出現在該分量串中的該第一出現之後；

在該分量之一第二出現經定位之一狀況下：

判定該分量串中之該分量之該第一出現與該第二出現之間的分量的一數目；及

將該分量之該分量距離設定成分量之該經判定的數目。

12. 如請求項11之方法，其中該等圖形處理指令實施一頂點著色器，且該分量集合包含X、Y、Z及W分量。

13. 如請求項11之方法，其中該等圖形處理指令實施一片段著色器，且該分量集合包含R、G、B及A分量。

14. 一種裝置，其包含：

至少一可程式處理單元，其經組態以實施一圖形管線之至少一部分；及

一指令排程器，其經組態以使用一經減小的邊緣延時而對用於由該至少一可程式處理單元執行的指令進行排程，該指令排程器經組態以：

識別圖形處理指令中之兩個指令之間的一相依性，該兩個指令中之一者包含一前導指令且該兩個指令中之另一者包含一後繼指令；

判定與該前導指令與該後繼指令之間的該相依性相關聯之一初始邊緣延時；

判定對應於該前導指令及該後繼指令之一指令距離；及

將該初始邊緣延時減小該經判定之指令距離以判定與該前導指令與該後繼指令之間的該相依性相關聯之一經減小的邊緣延時。

15. 如請求項14之裝置，其中該指令排程器進一步經組態以：

判定在該至少一可程式處理單元開始該後繼指令的執行之前待執行之同步指令的一數目以使該前導指令及該後繼指令之執行同步，使用該經減小的邊緣延時判定同步指令的該數目，每一同步指令為一獨立指令或一NOP。

16. 如請求項14之裝置，其中該至少一可程式處理單元基於該等圖形處理指令而實施一著色器。

17. 如請求項16之裝置，其中該著色器為一頂點著色器。

18. 如請求項16之裝置，其中該著色器為一片段著色器。

19. 如請求項14之裝置，其中該初始邊緣延時為與由該至少一可程式處理單元對該前導指令之執行相關聯的一硬體延時。

20. 如請求項14之裝置，其中該相依性包含一流相依性，以使得該前導指令之一目的地為該後繼指令之一源。

21. 如請求項14之裝置，其中經組態以判定一指令距離之該指令排程器進一步經組態以：

判定對應於該前導指令之一目的地運算元之該後繼指令之每一源運算元的一遮罩距離；

自該等經判定之遮罩距離選擇一最小遮罩距離作為該

指令距離。

22. 如請求項21之裝置，其中經組態以判定該後繼指令之每一源運算元的一遮罩距離之該指令排程器進一步經組態以：

判定該前導指令之該目的地運算元的一分量遮罩及該後繼指令之該源運算元的一分量遮罩；

藉由串連連接該目的地運算元之分量遮罩及該源運算元之分量遮罩而產生一分量串；

使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離；及

識別該等經判定之分量距離中之一最小分量距離以作為該源運算元之該遮罩距離。

23. 如請求項22之裝置，其中經組態以使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離之該指令排程器進一步經組態以：

對於該分量集合中之每一分量：

檢查該分量串以定位該分量串中之該分量的一第一出現；

在該分量之一第一出現經定位之一狀況下，檢查該分量串以定位該分量串中之該分量的一第二出現，該第二出現在該分量串中的該第一出現之後；

在該分量之一第二出現經定位之一狀況下：

判定該分量串中之該分量之該第一出現與該第二出現之間的分量的一數目；及

將該分量之該分量距離設定成分量之該經判定的數目。

24. 如請求項23之裝置，其中該至少一可程式處理單元使用該等圖形處理指令來實施一頂點著色器，且該分量集合包含X、Y、Z及W分量。

25. 如請求項23之裝置，其中該至少一可程式處理單元使用該等圖形處理指令來實施一片段著色器，且該分量集合包含R、G、B及A分量。

26. 一種其中儲存有電腦可執行程式碼之電腦可讀記憶體媒體，該程式碼包含進行以下動作之碼：

識別圖形處理指令中之兩個指令之間的一相依性，該兩個指令中之一者包含一前導指令且該兩個指令中之另一者包含一後繼指令；

判定與該前導指令與該後繼指令之間的該相依性相關聯之一初始邊緣延時；

判定對應於該前導指令及該後繼指令之一指令距離；及

將該初始邊緣延時減小該經判定之指令距離以判定與該前導指令與該後繼指令之間的該相依性相關聯之一經減小的邊緣延時。

27. 如請求項26之媒體，該程式碼進一步包含進行以下動作之碼：

使用該經減小的邊緣延時而對該後繼指令之執行進行排程。

28. 如請求項26之媒體，該程式碼進一步包含進行以下動作

之碼：

判定在開始該後繼指令的執行之前待執行之同步指令的一數目以使該前導指令及該後繼指令之執行同步，使用該經減小的邊緣延時判定同步指令的該數目，每一同步指令為一獨立指令或一NOP。

29. 如請求項26之媒體，其中該等圖形處理指令實施一著色器。

30. 如請求項29之媒體，其中該著色器為一頂點著色器。

31. 如請求項29之媒體，其中該著色器為一片段著色器。

32. 如請求項26之媒體，其中該初始邊緣延時為與該前導指令相關聯之一硬體延時。

33. 如請求項26之媒體，其中該相依性包含一流相依性，以使得該前導指令之一目的地為該後繼指令之一源。

34. 如請求項26之媒體，其中經組態以判定一指令距離之該程式碼進一步包含進行以下動作之碼：

判定對應於該前導指令之一目的地運算元之該後繼指令之每一源運算元的一遮罩距離；

自該等經判定之遮罩距離選擇一最小遮罩距離作為該指令距離。

35. 如請求項34之媒體，其中經組態以判定該後繼指令之每一源運算元的一遮罩距離之該程式碼進一步包含進行以下動作之碼：

判定該前導指令之該目的地運算元的一分量遮罩及該後繼指令之該源運算元的一分量遮罩；

藉由串連連接該目的地運算元之分量遮罩及該源運算元之分量遮罩而產生一分量串；

使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離；及

識別該等經判定之分量距離中之一最小分量距離以作為該源運算元之該遮罩距離。

36. 如請求項35之媒體，其中經組態以使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離之該程式碼進一步包含進行以下動作之碼：

對於該分量集合中之每一分量：

檢查該分量串以定位該分量串中之該分量的一第一出現；

在該分量之一第一出現經定位之一狀況下，檢查該分量串以定位該分量串中之該分量的一第二出現，該第二出現在該分量串中的該第一出現之後；

在該分量之一第二出現經定位之一狀況下：

判定該分量串中之該分量之該第一出現與該第二出現之間的分量的一數目；及

將該分量之該分量距離設定成分量之該經判定的數目。

37. 如請求項36之媒體，其中該等圖形處理指令實施一頂點著色器且該分量集合包含X、Y、Z及W分量。

38. 如請求項36之媒體，其中該等圖形處理指令實施一片段著色器且該分量集合包含R、G、B及A分量。

39. 一種裝置，其包含：

至少一可程式處理單元，其經組態以實施一圖形管線之至少一部分；及

一指令排程構件，其經組態以使用一經減小的邊緣延時而對用於由該至少一可程式處理單元執行的指令進行排程，該指令排程構件包含：

用於識別圖形處理指令中之兩個指令之間的一相依性的構件，該兩個指令中之一者包含一前導指令且該兩個指令中之另一者包含一後繼指令；

用於判定與該前導指令與該後繼指令之間的該相依性相關聯之一初始邊緣延時的構件；

用於判定對應於該前導指令及該後繼指令之一指令距離的構件；及

用於將該初始邊緣延時減小該經判定之指令距離以判定與該前導指令及該後繼指令之間的該相依性相關聯之一經減小的邊緣延時的構件。

40. 如請求項39之裝置，該指令排程構件進一步包含：

用於判定在開始該後繼指令的執行之前待執行之同步指令的一數目以使該前導指令及該後繼指令之執行同步的構件，使用該經減小的邊緣延時判定同步指令的該數目，每一同步指令為一獨立指令或一NOP。

41. 如請求項39之裝置，其中該至少一可程式處理單元基於該等圖形處理指令而實施一著色器。

42. 如請求項41之裝置，其中該著色器為一頂點著色器。

43. 如請求項41之裝置，其中該著色器為一片段著色器。

44. 如請求項39之裝置，其中該初始邊緣延時為與由該至少一可程式處理單元對該前導指令之執行相關聯的一硬體延時。

45. 如請求項39之裝置，其中該相依性包含一流相依性，以使得該前導指令之一目的地為該後繼指令之一源。

46. 如請求項39之裝置，其中該用於判定一指令距離之構件進一步包含：

用於判定對應於該前導指令之一目的地運算元之該後繼指令之每一源運算元的一遮罩距離的構件；

用於自該等經判定之遮罩距離選擇一最小遮罩距離作為該指令距離的構件。

47. 如請求項46之裝置，其中該用於判定該後繼指令之每一源運算元的一遮罩距離的構件進一步包含：

用於判定該前導指令之該目的地運算元的一分量遮罩及該後繼指令之該源運算元的一分量遮罩的構件；

用於藉由串連連接該目的地運算元之分量遮罩及該源運算元之分量遮罩而產生一分量串的構件；

用於使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離的構件；及

用於識別該等經判定之分量距離中之一最小分量距離以作為該源運算元之該遮罩距離的構件。

48. 如請求項47之裝置，其中該用於使用該分量串而判定與一分量集合中之每一分量相關聯的一分量距離的構件進

一步包含：

對於該分量集合中之每一分量：

用於檢查該分量串以定位該分量串中之該分量的一第一出現的構件；

在該分量之一第一出現經定位之一狀況下，用於檢查該分量串以定位該分量串中之該分量的一第二出現的構件，該第二出現在該分量串中的該第一出現之後；

在該分量之一第二出現經定位之一狀況下：

用於判定該分量串中之該分量之該第一出現與該第二出現之間的分量的一數目的構件；及

用於將該分量之該分量距離設定成分量之該經判定的數目的構件。

49. 如請求項48之裝置，其中該至少一可程式處理單元使用來自該等圖形處理指令之指令來實施一頂點著色器，且該分量集合包含X、Y、Z及W分量。

50. 如請求項48之裝置，其中該至少一可程式處理單元使用來自該等圖形處理指令之指令來實施一片段著色器，且該分量集合包含R、G、B及A分量。

八、圖式：

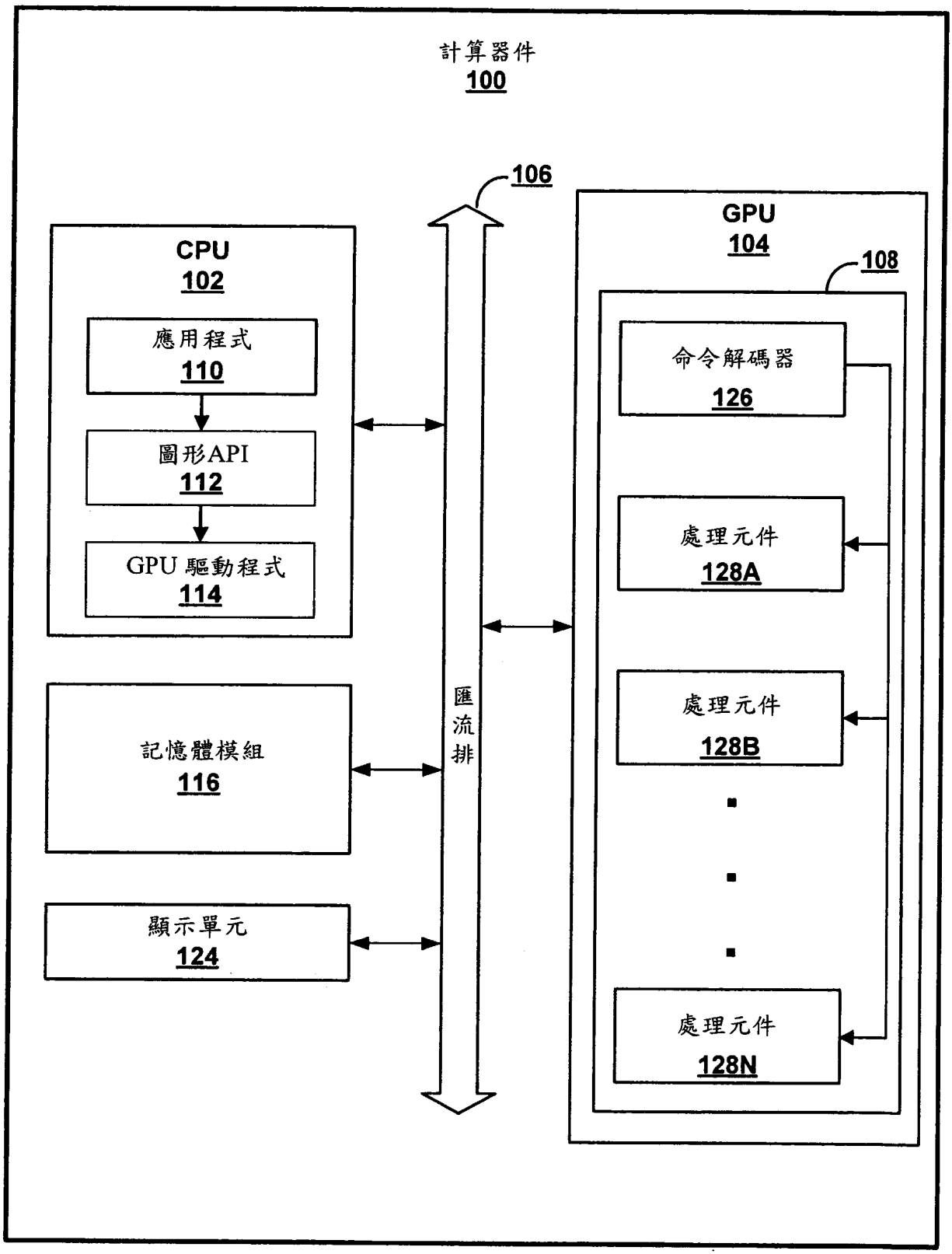

計算器件
100
106
CPU
102
應用程式
110
圖形API
112
GPU 驅動程式
114
記憶體模組
116
顯示單元
124
匯流排
GPU
104
108
命令解碼器
126
處理元件
128A
處理元件
128B
處理元件
128N

圖1

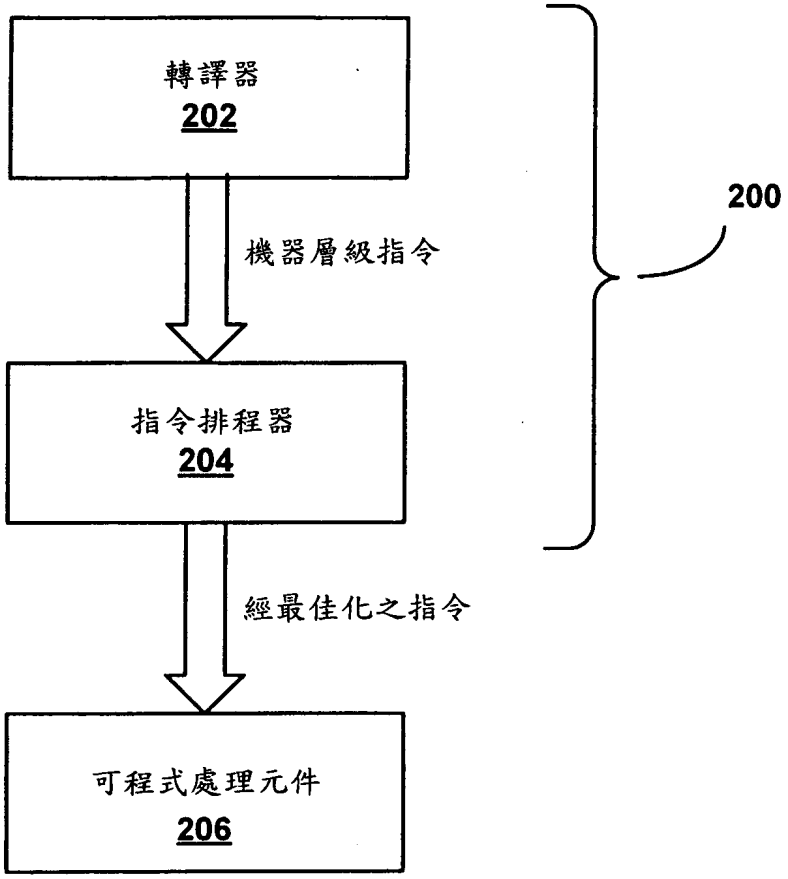

轉譯器
202
機器層級指令
200
指令排程器
204
經最佳化之指令
可程式處理元件
206

圖2

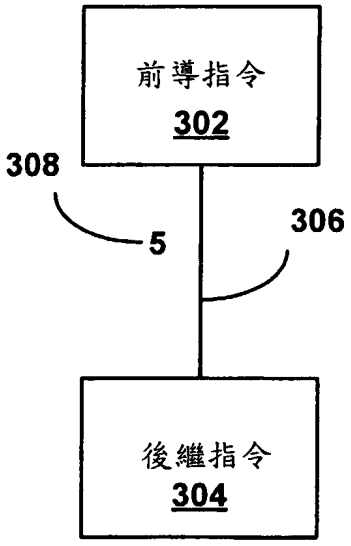

前導指令
302
308
5
306
後繼指令
304

圖3

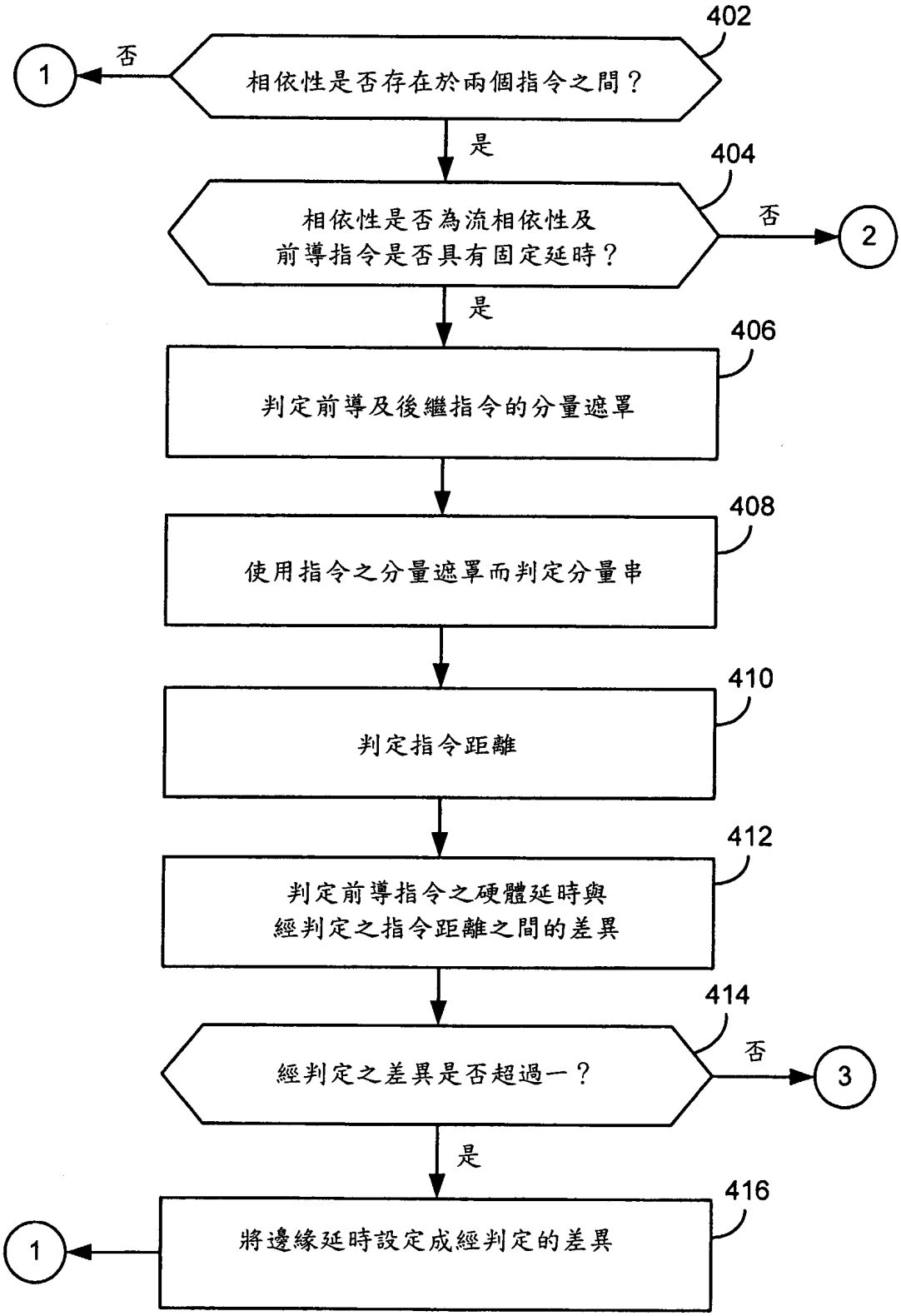

402
1
否
相依性是否存在於兩個指令之間？
是
404
相依性是否為流相依性及
前導指令是否具有固定延時？
否
2
是
406
判定前導及後繼指令的分量遮罩
408
使用指令之分量遮罩而判定分量串
410
判定指令距離
412
判定前導指令之硬體延時與
經判定之指令距離之間的差異
414
經判定之差異是否超過一？
否
3
是
416
1
將邊緣延時設定成經判定的差異

圖4A

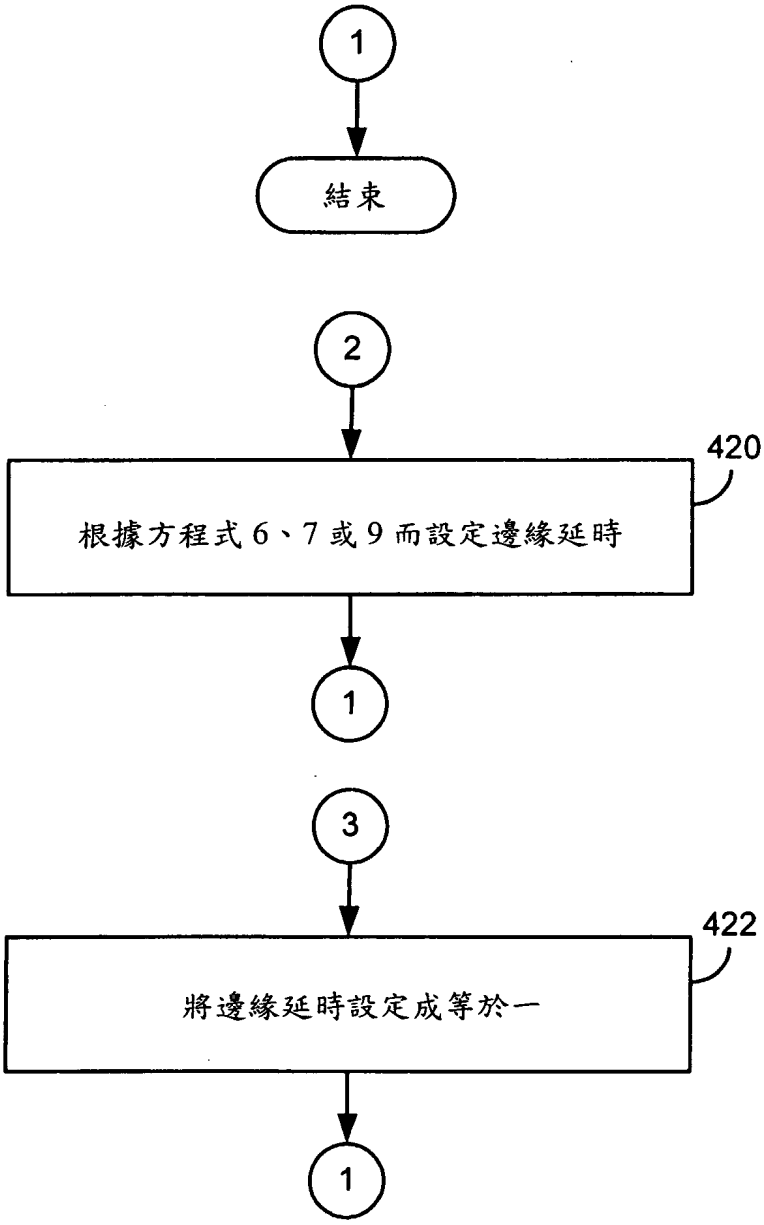


圖4B

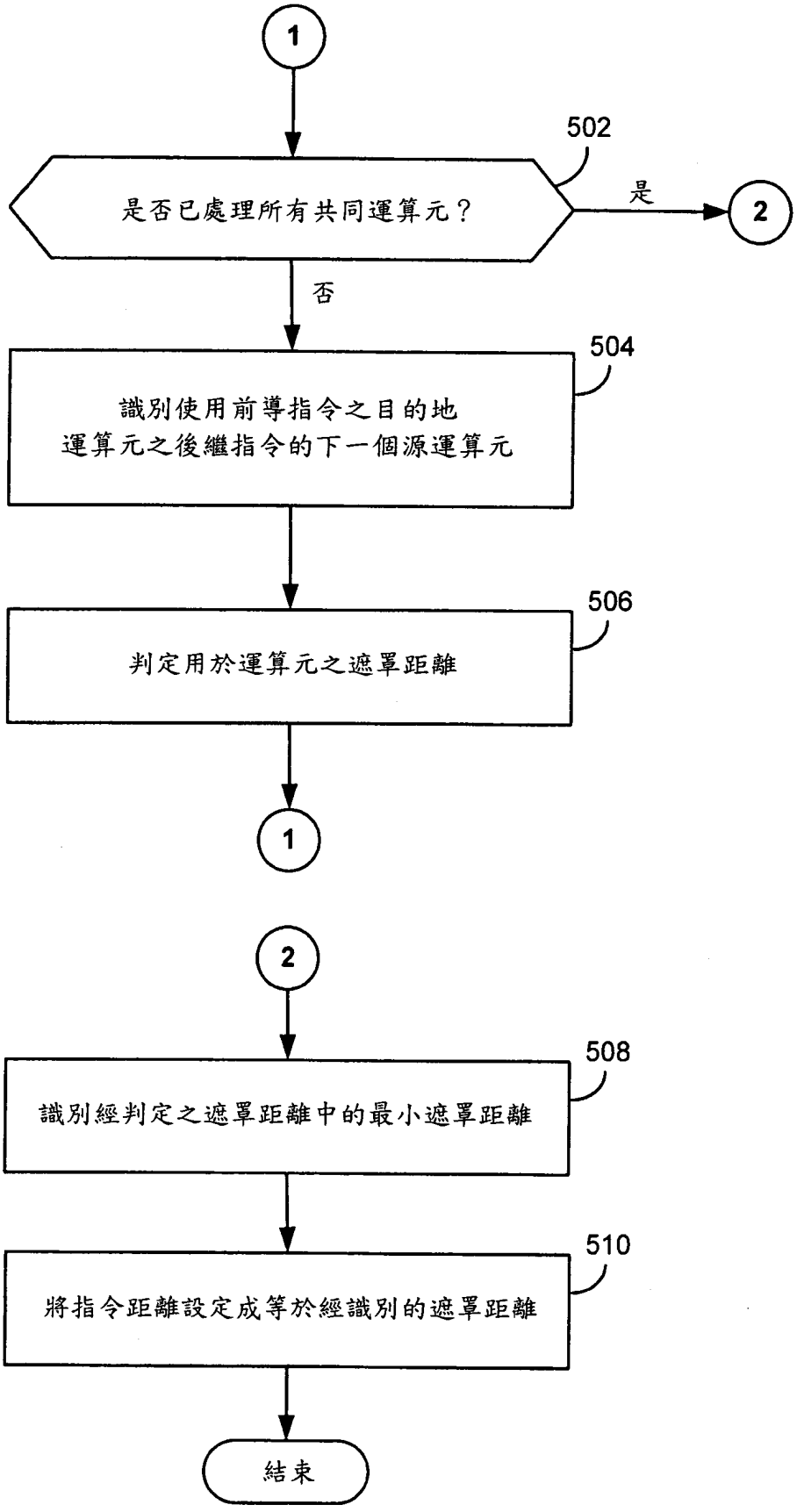

1
502
是否已處理所有共同運算元？
是
2
否
504
識別使用前導指令之目的地
運算元之後繼指令的下一個源運算元
506
判定用於運算元之遮罩距離
1
2
508
識別經判定之遮罩距離中的最小遮罩距離
510
將指令距離設定成等於經識別的遮罩距離
結束

圖5

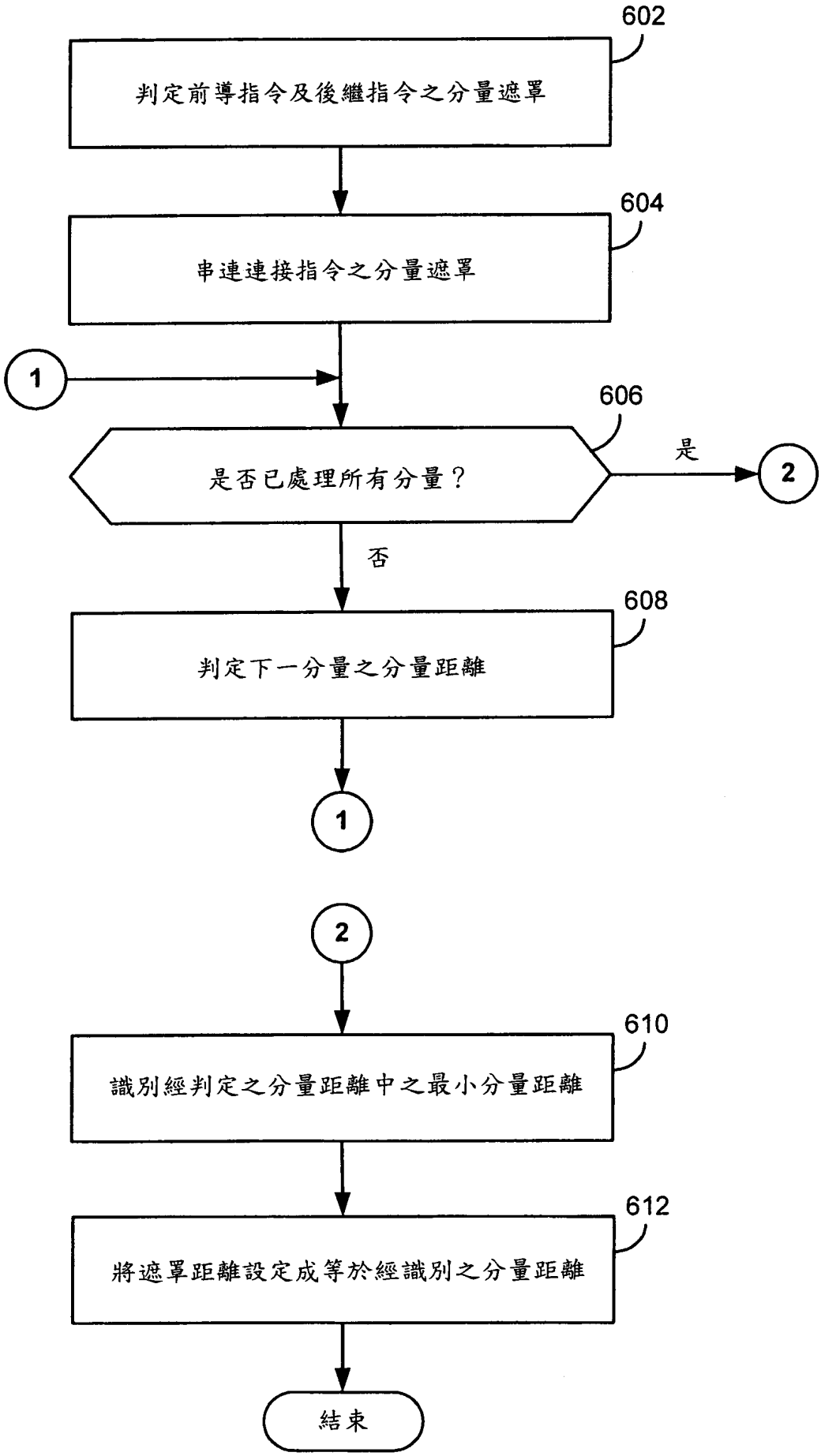

602
判定前導指令及後繼指令之分量遮罩
604
串連連接指令之分量遮罩
1
606
是否已處理所有分量？
是
2
否
608
判定下一分量之分量距離
1
2
610
識別經判定之分量距離中之最小分量距離
612
將遮罩距離設定成等於經識別之分量距離
結束

圖6

I1: ADD R1:XYZ, R2:XYZ, R3:XYZ
I2: MUL R4.X, R1.X, R1.Y

702

指令	運算元	遮罩
I1	R1	MD: XYZ
I2	R1	MS1: X
I2	R1	MS2: Y

704

706

遮罩距離(MD, MS1) = 2:

分量	M	分量距離
X	XYZX	2
Y	XYZX	+INF
Z	XYZX	+INF
W	XYZX	+INF

708

710

遮罩距離(MD, MS2) = 1:

分量	M	分量距離
X	XYZY	+INF
Y	XYZY	1
Z	XYZY	+INF
W	XYZY	+INF

712

指令距離(I1, I2) = 遮罩距離(MD, MS1)及
遮罩距離(MD, MS2) = 1中之較小者

714

圖7

```
0(91) type2: rcp r9.w, r8.w
1(93) type3: (rpt2)fmul r10.x, (r)r4.x, r9.w
2(95) type3: (rpt2)fmul r11.x, (r)r5.x, r9.w
3(97) type3: (rpt2)fmul r12.x, (r)r6.x, r9.w
4(99) type3: (rpt2)fmul r13.x, (r)r7.x, r9.w
5(102) type3: (lock)fmul dummy, r11.x, r11.x
6(103) type3: (rpt1)fmac r9.x, (r)r11.y, (r)r11.y
7(105) type2: rsq r9.x, r9.x
8(107) type3: (rpt2)fmul r9.x, r9.x, (r)r11.x
9(110) type3: (lock)fmul dummy, r12.x, r12.x
10(111) type3: (rpt1)fmac r14.x, (r)r12.y, (r)r12.y
11(113) type2: rsq r14.x, r14.x
12(115) type3: (rpt2)fmul r14.x, r14.x, (r)r12.x
13(118) type3: (lock)fmul dummy, r10.x, r10.x
14(119) type3: (rpt1)fmac r15.x, (r)r10.y, (r)r10.y
15(121) type2: rsq r15.x, r15.x
16(123) type3: (rpt2)fmul r15.x, r15.x, (r)r10.x
17(125) type4: sample r16.XYZW, r3.xyyy, m1, s1
18(132) type4: fmad r17.z, c1.y, r16.y, -c1.x
19(134) type4: fmad r17.y, c1.y, r16.y, -c1.x
20(136) type4: fmad r17.x, c1.y, r16.x, -c1.x
21(138) type2: (rpt2)fmov r16.x, (r)r17.x
22(140) type3: (lock)fmul dummy, r16.x, r16.x
23(141) type3: fmac r16.z, r16.y, r16.y
24(143) type3: fadd r16.z, -c1.x, r16.z
25(145) type2: sqrt r16.z, -r16.z
26(147) type3: (lock)fmul dummy, r16.x, r9.x
27(148) type3: (rpt1)fmac r18.x, (r)r16.y, (r)r9.y
28(150) type3: (lock)fmul dummy, r16.x, r14.x
29(151) type3: (rpt1)fmac r18.y, (r)r16.y, (r)r14.y
30(153) type3: (lock)fmul dummy, r16.x, r15.x
31(154) type3: (rpt1)fmac r18.z, (r)r16.y, (r)r15.y
32(157) type3: (lock)fmul dummy, r18.x, r18.x
33(158) type3: (rpt1)fmac r20.x, (r)r18.y, (r)r18.y
34(160) type2: rsq r20.x, r20.x
35(162) type3: (rpt2)fmul r18.x, r20.x, (r)r18.x
36(165) type3: (lock)fmul dummy, r13.x, r13.x
37(166) type3: (rpt1)fmac r21.x, (r)r13.y, (r)r13.y
38(168) type2: rsq r21.x, r21.x
39(170) type3: (rpt2)fmul r21.x, r21.x, (r)r13.x
```

圖8A

```
40(172) type3: (lock)fmul dummy, r21.x, r18.x
41(173) type3: (rpt1)fmac r22.x, (r)r21.y, (r)r18.y
42(175) type3: fmul r22.x, c2.x, r22.x
43(177) type4: (rpt2)fmad r22.x, r22.x, (r)r18.x, -(r)r21.x
44(180) type3: (lock)fmul dummy, r22.x, r22.x
45(181) type3: (rpt1)fmac r24.x, (r)r22.y, (r)r22.y
46(183) type2: rsq r24.x, r24.x
47(185) type3: (rpt2)fmul r22.x, r24.x, (r)r22.x
48(187) type3: (lock)fmul dummy, c3.x, r18.x
49(188) type3: (rpt1)fmac (sat)r25.x, (r)c3.y, (r)r18.y
50(190) type4: fmad r25.x, c4.x, r25.x, c4.y
51(192) type3: (lock)fmul dummy, c3.x, r22.x
52(193) type3: (rpt1)fmac (sat)r25.y, (r)c3.y, (r)r22.y
53(196) type2: log r26.x, r25.y
54(197) type3: fmul r26.x, c6.x, r26.x
55(198) type2: exp r25.y, r26.x
56(200) type3: fmul r25.y, c0.w, r25.y
57(202) type4: sample r27.XYZW, r2.xyyy, m0, s0
58(204) type4: (rpt2)fmad r27.x, (r)r27.x, r25.x, r25.y
59(209) type2: sqrt r0.z, (r)r27.z
60(206) type2: (rpt1)sqrt r0.x, (r)r27.x
61(208) type0: end
```

圖8B

```
0(125) type4: (syn)sample r16.XYZW, r3.xyyy, m1, s1
1(202) type4: (syn)sample r27.XYZW, r2.xyyy, m0, s0
2(91) type2: (syn)rcp r9.w, r8.w
3(132) type4: (syn)fmad r17.z, c1.y, r16.y, -c1.x
4(134) type4: (syn)fmad r17.y, c1.y, r16.y, -c1.x
5(136) type4: (syn)fmad r17.x, c1.y, r16.x, -c1.x
6(93) type3: (rpt2/syn)fmul r10.x, (r)r4.x, r9.w
7(95) type3: (rpt2/syn)fmul r11.x, (r)r5.x, r9.w
8(97) type3: (rpt2/syn)fmul r12.x, (r)r6.x, r9.w
9(99) type3: (rpt2/syn)fmul r13.x, (r)r7.x, r9.w
10(138) type2: (rpt2)fmov r16.x, (r)r17.x
11(118) type3: (lock)fmul dummy, r10.x, r10.x
12(119) type3: (rpt1)fmac r15.x, (r)r10.y, (r)r10.y
13(102) type3: (lock)fmul dummy, r11.x, r11.x
14(103) type3: (rpt1)fmac r9.x, (r)r11.y, (r)r11.y
15(110) type3: (lock)fmul dummy, r12.x, r12.x
16(111) type3: (rpt1)fmac r14.x, (r)r12.y, (r)r12.y
17(121) type2: rsq r15.x, r15.x
18(165) type3: (lock)fmul dummy, r13.x, r13.x
19(166) type3: (rpt1)fmac r21.x, (r)r13.y, (r)r13.y
20(105) type2: rsq r9.x, r9.x
21(140) type3: (lock)fmul dummy, r16.x, r16.x
22(141) type3: fmac r16.z, r16.y, r16.y
23(123) type3: (rpt2/syn)fmul r15.x, r15.x, (r)r10.x
24(113) type2: rsq r14.x, r14.x
25(107) type3: (rpt2/syn)fmul r9.x, r9.x, (r)r11.x
26(168) type2: rsq r21.x, r21.x
27(143) type3: fadd r16.z, -c1.x, r16.z
28(115) type3: (rpt2/syn)fmul r14.x, r14.x, (r)r12.x
29(170) type3: (rpt2/syn)fmul r21.x, r21.x, (r)r13.x
30(145) type2: sqrt r16.z, -r16.z
31(147) type3: (lock)fmul dummy, r16.x, r9.x
32(148) type3: (rpt1/syn)fmac r18.x, (r)r16.y, (r)r9.y
33(150) type3: (lock)fmul dummy, r16.x, r14.x
34(151) type3: (rpt1/syn)fmac r18.y, (r)r16.y, (r)r14.y
35(153) type3: (lock)fmul dummy, r16.x, r15.x
36(154) type3: (rpt1/syn)fmac r18.z, (r)r16.y, (r)r15.y
37(219) type0: (rpt2)nop
38(157) type3: (lock)fmul dummy, r18.x, r18.x
39(158) type3: (rpt1)fmac r20.x, (r)r18.y, (r)r18.y
40(222) type0: (rpt1)nop
41(224) type0: (rpt3)nop
42(160) type2: rsq r20.x, r20.x
43(162) type3: (rpt2/syn)fmul r18.x, r20.x, (r)r18.x
44(228) type0: (rpt2)nop
45(172) type3: (lock)fmul dummy, r21.x, r18.x
46(173) type3: (rpt1)fmac r22.x, (r)r21.y, (r)r18.y
47(187) type3: (lock)fmul dummy, c3.x, r18.x
48(188) type3: (rpt1)fmac (sat)r25.x, (r)c3.y, (r)r18.y
49(175) type3: fmul r22.x, c2.x, r22.x
```

812

圖 8C

```
50(231) type0: (rpt1)nop
51(190) type4: fmad r25.x, c4.x, r25.x, c4.y
52(177) type4: (rpt2)fmad r22.x, r22.x, (r)r18.x, -(r)r21.x
53(233) type0: (rpt2)nop
54(180) type3: (lock)fmul dummy, r22.x, r22.x
55(181) type3: (rpt1)fmac r24.x, (r)r22.y, (r)r22.y
56(236) type0: (rpt1)nop
57(238) type0: (rpt3)nop
58(183) type2: rsq r24.x, r24.x
59(185) type3: (rpt2/syn)fmul r22.x, r24.x, (r)r22.x
60(242) type0: (rpt2)nop
61(192) type3: (lock)fmul dummy, c3.x, r22.x
62(193) type3: (rpt1)fmac (sat)r25.y, (r)c3.y, (r)r22.y
63(245) type0: (rpt1)nop
64(247) type0: (rpt3)nop
65(196) type2: log r26.x, r25.y
66(197) type3: (syn)fmul r26.x, c6.x, r26.x
67(251) type0: (rpt1)nop
68(253) type0: (rpt3)nop
69(198) type2: exp r25.y, r26.x
70(200) type3: (syn)fmul r25.y, c0.w, r25.y
71(257) type0: (rpt2)nop
72(204) type4: (rpt2/syn)fmad r27.x, (r)r27.x, r25.x, r25.y
73(260) type0: (rpt1)nop
74(262) type0: (rpt3)nop
75(209) type2: sqrt r0.z, (r)r27.z
76(206) type2: (rpt1)sqrt r0.x, (r)r27.x
77(266) type0: (syn)nop
78(208) type0: end
```

圖8D

```
 0(125) type4: (syn)sample r16.XYZW, r3.xyyy, m1, s1
 1(202) type4: (syn)sample r27.XYZW, r2.xyyy, m0, s0
 2(91) type2: (syn)rcp r9.w, r8.w
 3(132) type4: (syn)fmad r17.z, c1.y, r16.y, -c1.x
 4(134) type4: (syn)fmad r17.y, c1.y, r16.y, -c1.x
 5(136) type4: (syn)fmad r17.x, c1.y, r16.x, -c1.x
 6(93) type3: (rpt2/syn)fmul r10.x, (r)r4.x, r9.w
 7(95) type3: (rpt2/syn)fmul r11.x, (r)r5.x, r9.w
 8(97) type3: (rpt2/syn)fmul r12.x, (r)r6.x, r9.w
 9(99) type3: (rpt2/syn)fmul r13.x, (r)r7.x, r9.w
10(138) type2: (rpt2)fmov r16.x, (r)r17.x
11(118) type3: (lock)fmul dummy, r10.x, r10.x
12(119) type3: (rpt1)fmac r15.x, (r)r10.y, (r)r10.y
13(102) type3: (lock)fmul dummy, r11.x, r11.x
14(103) type3: (rpt1)fmac r9.x, (r)r11.y, (r)r11.y
15(110) type3: (lock)fmul dummy, r12.x, r12.x
16(111) type3: (rpt1)fmac r14.x, (r)r12.y, (r)r12.y
17(121) type2: rsq r15.x, r15.x
18(165) type3: (lock)fmul dummy, r13.x, r13.x
19(166) type3: (rpt1)fmac r21.x, (r)r13.y, (r)r13.y
20(105) type2: rsq r9.x, r9.x
21(140) type3: (lock)fmul dummy, r16.x, r16.x
22(141) type3: fmac r16.z, r16.y, r16.y
23(123) type3: (rpt2/syn)fmul r15.x, r15.x, (r)r10.x
24(113) type2: rsq r14.x, r14.x
25(107) type3: (rpt2/syn)fmul r9.x, r9.x, (r)r11.x
26(168) type2: rsq r21.x, r21.x
27(143) type3: fadd r16.z, -c1.x, r16.z
28(115) type3: (rpt2/syn)fmul r14.x, r14.x, (r)r12.x
29(170) type3: (rpt2/syn)fmul r21.x, r21.x, (r)r13.x
30(145) type2: sqrt r16.z, -r16.z
31(147) type3: (lock)fmul dummy, r16.x, r9.x
32(148) type3: (rpt1/syn)fmac r18.x, (r)r16.y, (r)r9.y
33(150) type3: (lock)fmul dummy, r16.x, r14.x
34(151) type3: (rpt1/syn)fmac r18.y, (r)r16.y, (r)r14.y
35(153) type3: (lock)fmul dummy, r16.x, r15.x
36(154) type3: (rpt1/syn)fmac r18.z, (r)r16.y, (r)r15.y
37(219) type0: nop
38(157) type3: (lock)fmul dummy, r18.x, r18.x
39(158) type3: (rpt1)fmac r20.x, (r)r18.y, (r)r18.y
40(220) type0: (rpt1)nop
41(222) type0: (rpt3)nop
42(160) type2: rsq r20.x, r20.x
43(162) type3: (rpt2/syn)fmul r18.x, r20.x, (r)r18.x
44(226) type0: nop
45(172) type3: (lock)fmul dummy, r21.x, r18.x
46(173) type3: (rpt1)fmac r22.x, (r)r21.y, (r)r18.y
47(187) type3: (lock)fmul dummy, c3.x, r18.x
48(188) type3: (rpt1)fmac (sat)r25.x, (r)c3.y, (r)r18.y
49(175) type3: fmul r22.x, c2.x, r22.x
```

822

圖8E

```
50(227) type0: (rpt1)nop
51(190) type4: fmad r25.x, c4.x, r25.x, c4.y
52(177) type4: (rpt2)fmad r22.x, r22.x, (r)r18.x, -(r)r21.x
53(229) type0: nop
54(180) type3: (lock)fmul dummy, r22.x, r22.x
55(181) type3: (rpt1)fmac r24.x, (r)r22.y, (r)r22.y
56(230) type0: (rpt1)nop
57(232) type0: (rpt3)nop
58(183) type2: rsq r24.x, r24.x
59(185) type3: (rpt2/syn)fmul r22.x, r24.x, (r)r22.x
60(236) type0: nop
61(192) type3: (lock)fmul dummy, c3.x, r22.x
62(193) type3: (rpt1)fmac (sat)r25.y, (r)c3.y, (r)r22.y
63(237) type0: (rpt1)nop
64(239) type0: (rpt3)nop
65(196) type2: log r26.x, r25.y
66(197) type3: (syn)fmul r26.x, c6.x, r26.x
67(243) type0: (rpt1)nop
68(245) type0: (rpt3)nop
69(198) type2: exp r25.y, r26.x
70(200) type3: (syn)fmul r25.y, c0.w, r25.y
71(249) type0: (rpt2)nop
72(204) type4: (rpt2/syn)fmad r27.x, (r)r27.x, r25.x, r25.y
73(252) type0: (rpt3)nop
74(206) type2: (rpt1)sqrt r0.x, (r)r27.x
75(256) type0: (rpt1)nop
76(209) type2: sqrt r0.z, (r)r27.z
77(258) type0: (syn)nop
78(208) type0: end
```

圖8F

四、指定代表圖：

(一)本案指定代表圖為：第（ 2 ）圖。

(二)本代表圖之元件符號簡單說明：

200　　編譯器

202　　轉譯器

204　　指令排程器

206　　可程式處理元件

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

（無）