發明專利說明書

公告本

（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：97102940

※ 申請日期：97.1.25　　※IPC 分類：G06F 9/38 (2006.01)
G06F 9/455 (2006.01)

一、發明名稱：（中文/英文）

平行緒計算之虛擬架構及指令集

VIRTUAL ARCHITECTURE AND INSTRUCTION SET FOR PARALLEL THREAD COMPUTING

二、申 請 人：（共 1 人）

姓名或名稱：（中文/英文）

美商輝達公司

NVIDIA CORPORATION

代表人：（中文/英文）

理查　B　多明尼哥

DOMINGO, RICHARD B.

住居所或營業所位址：（中文/英文）

美國加州聖塔克拉市聖湯瑪斯大道2701號

2701 SAN TOMAS EXPRESSWAY, SANTA CLARA, CALIFORNIA 95050, U.S.A.

國　籍：（中文/英文）

美國　　U.S.A.

三、發明人：(共 7 人)

姓　名：(中文/英文)

1. 約略　R　尼柯爾斯
NICKOLLS, JOHN R.

2. 亨利　P　莫爾頓
MORETON, HENRY P.

3. 拉斯　S　耐蘭德
NYLAND, LARS S.

4. 伊安　A　巴克
BUCK, IAN A.

5. 理察　C　強森
JOHNSON, RICHARD C.

6. 羅伯特　S　格蘭維爾
GLANVILLE, ROBERT S.

7. 傑安特　B　柯西
KOLHE, JAYANT B.

國　籍：(中文/英文)

1. 美國　U.S.A.
2. 美國　U.S.A.
3. 美國　U.S.A.
4. 美國　U.S.A.
5. 美國　U.S.A.
6. 美國　U.S.A.
7. 印度　INDIA

四、聲明事項：

☐ 主張專利法第二十二條第二項☐第一款或☐第二款規定之事實，其事實發生日期為：　年　月　日。

☑ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號　順序註記】

☑ 有主張專利法第二十七條第一項國際優先權：

1. 美國；2007年01月26日；11/627,892

2.

☐ 無主張專利法第二十七條第一項國際優先權：

1.

2.

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號　順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須暫存生物材料者：

國內生物材料　【格式請依：暫存機構、日期、號碼　順序註記】

國外生物材料　【格式請依：暫存國家、機構、日期、號碼　順序註記】

☐ 不須暫存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須暫存。

九、發明說明：

【發明所屬之技術領域】

本發明大體上涉及平行處理，且更特定而言涉及用於平行緒計算的虛擬架構及指令集。

【先前技術】

在平行處理中，多個處理單元(例如，多個處理器晶片或單個晶片內的多個處理核心)同時操作以處理資料。可使用此種系統來解決便於分解成多個部分的問題。一個實例是影像過濾，其中根據輸入影像的某一數目的像素來計算輸出影像的每個像素。每個輸出像素的計算一般獨立於其它所有輸出像素，因此不同的處理單元可平行計算不同的輸出像素。其它許多類型的問題也適於平行分解。總的來說，N向(N-way)平行執行可將此種問題的解決方案加速大約N倍。

另一類問題在平行的執行緒可彼此協調的情況下則適於平行處理。一個實例是快速傅立葉變換(FFT)，其為一種遞歸演算法：其中在每個級，對前一級的輸出執行計算，以便產生新的值，新的值用於對下一級的輸入，如此繼續直到到達輸出級為止。單個執行緒可執行多個級，只要該線緒能可靠地獲得來自前一級的輸出資料即可。如果要在多個緒之間劃分任務，那麼必須提供某種協調機制，以便(例如)線緒不會試圖讀取尚未寫入的輸入資料。(在2005年12月15日申請的共同轉讓、共同待決的美國專利申請案第11/303,780號中描述了這個問題的一個解決方案)。

然而，平行處理系統的程式設計可能較為困難。通常要求程式設計人員知道可用的處理單元的數目及其能力(指令集、資料暫存器數目、互連件等)，以便建立處理單元可實際執行的代碼。雖然機器特定的編譯器可在此方面提供相當大的幫助，但仍有必要在每次將代碼移植(port)到不同處理器時重新編譯代碼。

此外，平行處理架構的各種態樣正在迅速發展。舉例而言，正在陸續開發新的平台架構、指令集及程式設計模型。隨著平行架構的各種態樣(例如，程式設計模型或指令集)從一代改變成下一代，必須也相應地改變應用程式、軟體庫、編譯器及其它軟體及工具。這種不穩定性可能會為平行處理代碼的開發及維護增加相當大的額外開銷。

當需要線緒之間協調時，平行程式設計變得更加困難。程式設計人員必須確定在特定處理器或電腦系統中有何種可用於支援(或仿真)線緒間通信的機制，且必須編寫利用可用機制的代碼。由於不同的電腦系統上的可用及/或最佳機制一般不同，所以這種類型的平行代碼一般不可移植；必須針對其執行於的每種硬體平台重編寫平行代碼。

此外，除了為處理器提供可執行代碼，程式設計人員還必須為"主"處理器提供控制代碼，控制代碼協調各種處理單元的操作，例如向每個處理單元指示要執行哪個程式及處理哪些輸入資料。此控制代碼通常專用於特定的主處理器及處理器間的通信協定，且如果要替換不同的主處理

器則通常必須重編寫控制代碼。

編譯及重新編譯平行處理代碼中的困難會使使用者不願隨著計算技術的發展而升級其系統。因此，將需要使已編譯的平行處理代碼脫離特定的硬體平台，並提供一種作為平行應用程式及工具的目標的穩定的平行處理架構及指令集。

【發明內容】

本發明的實施例提供一種用於平行緒計算的虛擬架構及虛擬指令集。虛擬平行架構定義：一種虛擬處理器，其支援具有不同虛擬緒之間的不同級別的資料共用及協調(例如，同步)的多個虛擬緒的併發執行；以及一種虛擬執行驅動器，其控制虛擬處理器。虛擬處理器的虛擬指令集架構用來定義虛擬緒的行為，且包含與平行緒行為(例如，資料共用及同步)有關的指令。藉由使用虛擬平行平台，程式設計人員能夠開發出其中虛擬緒併發執行以處理資料的應用程式。應用程式可用高度可移植的中間形式儲存及分布，該形式例如為瞄準虛擬平行平台的程式代碼。在安裝時間或執行時間，硬體特定的虛擬指令翻譯器及執行驅動器使中間形式的應用程式代碼適應於其執行於的特定硬體。因此，應用程式更加可移植且更容易開發，因為開發過程獨立於特定處理硬體。

根據本發明的一個態樣，一種用於定義平行處理操作的方法包含提供第一程式代碼，第一程式代碼定義要針對協作的虛擬緒的陣列中的若干虛擬緒中的每一者執行的操作

序列。將第一程式代碼編譯成虛擬緒程式，虛擬緒程式定義要針對陣列中的代表性虛擬緒執行的每線緒指令的序列，且每線緒指令的序列包含至少一個指令，該指令定義代表性虛擬緒與陣列中的一或多個其它虛擬緒之間的協作行為。儲存虛擬緒程式(例如，儲存在記憶體中或碟片上)，且可隨後將虛擬緒程式轉譯成符合目標平台架構的指令序列。

此外，也可提供第二程式代碼以定義協作的虛擬緒的陣列，陣列適合於處理輸入資料集以產生輸出資料集，其中陣列中的每個虛擬緒併發執行虛擬緒程式。有利地將第二程式代碼轉換成虛擬函數庫中的函數調用序列，其中庫包含初始化及致使執行協作的虛擬緒的陣列的虛擬函數。也可儲存這個函數調用序列。接著，可將所儲存的虛擬緒程式及函數調用序列轉譯成可在目標平台架構上執行的程式代碼，其中可執行的程式代碼定義一或多個執行協作的虛擬緒的陣列的平台線緒。可在符合目標平台架構的電腦系統上執行可執行的程式代碼，因而產生輸出資料集，可將輸出資料集儲存在儲存媒體(例如，電腦記憶體、碟片等)中。

如已提到的，虛擬緒程式代碼中的每線緒指令的序列有利地包含至少一個指令，該指令定義代表性虛擬緒與陣列中的一或多個其它虛擬緒之間的協作行為。舉例而言，每線緒指令的序列可包含：在序列中的特定點處暫停執行代表性虛擬緒的操作直到例如其它虛擬緒中的一或多者達到

特定點為止的時間的指令，使代表性虛擬緒將資料儲存在其它虛擬緒中的一或多者可存取的共用記憶體中的指令，使代表性虛擬緒自動讀取及更新儲存在其它虛擬緒中的一或多者可存取的共用記憶體中的資料的指令，等等。

虛擬緒程式也可包含變數定義語句，語句在若干虛擬狀態空間之一中定義變數，其中不同的虛擬狀態空間對應於虛擬緒之間的不同的資料共用模式。在一個實施例中，至少支援每線緒不共用模式及全局共用模式。在其它實施例中，也可支援額外模式，例如虛擬緒的一個陣列內的共用模式及/或複數個虛擬緒陣列之間的共用模式。

根據本發明的另一態樣，一種用於操作目標處理器的方法包含提供輸入程式代碼。輸入程式代碼包含第一部分，該第一部分定義要對虛擬緒陣列中的若干虛擬緒中的每一者執行的操作序列，該陣列適合於處理輸入資料集以產生輸出資料集，且輸入程式代碼還包含第二部分，該第二部分定義虛擬緒的陣列的維度。將輸入程式代碼的第一部分編譯成虛擬緒程式，虛擬緒程式定義要對陣列中的代表性虛擬緒執行的每線緒指令的序列。每線緒指令的序列包含至少一個指令，該指令定義代表性虛擬緒與陣列中的一或多個其它虛擬緒之間的協作行為。將輸入程式代碼的第二部分轉換成對虛擬函數庫的函數調用序列，其中該庫包含初始化及致使執行協作的虛擬緒的陣列的虛擬函數。將虛擬緒程式及函數調用序列轉譯成可在目標平台架構上執行的程式代碼，其中可執行的程式代碼定義執行協作的虛擬

緒的陣列的一或多個真實線緒。在符合目標平台架構的電腦系統上執行可執行程式代碼，因而產生輸出資料集，可將輸出資料集儲存在儲存媒體中。

在一些實施例中，可在兩個或兩個以上維度上定義虛擬緒的陣列。此外，輸入程式代碼的第二部分還可包含定義虛擬緒陣列的柵格的一或多個維度的函數調用，其中柵格中的每個陣列均將被執行。

可使用任何目標平台架構。在一些實施例中，目標平台架構包含主處理器及共處理器。在轉譯期間，可將虛擬緒程式轉譯成可由定義於共處理器上的若干線緒平行執行的程式代碼，同時將函數調用序列轉譯成對在主處理器上執行的用於共處理器的驅動程式的調用序列。在其它實施例中，目標平台架構包含中央處理單元(CPU)。在轉譯期間，將虛擬緒程式及函數調用序列的至少一部分轉譯成目標程式代碼，所述目標程式代碼使用少於虛擬緒數目的數目的CPU線緒來執行虛擬緒陣列。

根據本發明的又一實施例，一種用於操作目標處理器的方法包含獲得虛擬緒程式，虛擬緒程式定義要針對虛擬緒陣列中的若干虛擬緒中的代表性虛擬緒執行的每線緒指令序列，虛擬緒陣列適合於處理輸入資料集以產生輸出資料集。每線緒指令序列包含至少一個指令，該指令定義代表性虛擬緒與陣列中的一或多個其它虛擬緒之間的協作行為。也獲得定義虛擬緒陣列的維度的額外程式代碼。將虛擬緒程式及額外的程式代碼轉譯成可在目標平台架構上執

行的程式代碼，其中可執行的程式代碼定義執行虛擬緒陣列的一或多個平台線緒。在符合目標平台架構的電腦系統上執行可執行的程式代碼，因而產生輸出資料集且將輸出資料集儲存在記憶體中。

在一些實施例中，可藉由接收用高階程式設計語言編寫的源程式代碼並編譯該源程式代碼以產生虛擬緒程式來獲得虛擬緒程式。或者，可從儲存媒體讀取虛擬緒程式或經由網路從遠端電腦系統接收虛擬緒程式。應瞭解，所讀取或接收的虛擬緒代碼可能之前已經從高階語言編譯，或者直接建立為符合虛擬指令集架構的代碼。

以下詳細描述連同附圖一起將提供對本發明的性質及優點的更好瞭解。

【實施方式】

本發明實施例提供一種用於平行緒計算的虛擬架構及指令集。虛擬架構提供支援在不同緒之間具有多層資料共用及協調(例如，同步)的多個緒的併發執行的處理器模型，以及控制模型處理器的虛擬執行驅動器。用於定義處理緒的行為的虛擬指令集包含與平行緒行為有關的指令，例如允許某些線緒間的資料共用的指令及要求不同緒在程式內的由程式設計人員指定的某些點處變得同步的指令。使用虛擬平台，程式設計人員可開發在其中執行併發、協作的緒以處理資料的應用程式。硬體特定的虛擬指令翻譯器及執行驅動器使應用代碼適應其執行於的特定硬體。因此，應用程式更具移植性並更易於開發，因為開發過程獨立於

特定處理硬體。

1. 系統概述

圖1為根據本發明實施例的電腦系統100的方塊圖。電腦系統100包含中央處理單元(CPU)102，及包含記憶體橋105的經由匯流排路徑通信的系統記憶體104。記憶體橋105(其可以是(例如)Northbridge晶片)經由匯流排或其它通信路徑106(例如，HyperTransport鏈路)連接到I/O(輸入/輸出)橋107。I/O橋107(其可以是(例如)Southbridge晶片)從一或多個使用者輸入裝置108(例如，鍵盤、滑鼠)接收使用者輸入，並將輸入經由路徑106及記憶體橋105轉遞到CPU 102。平行處理子系統112經由匯流排或其它通信路徑113(例如，PCI Express或加速圖形埠鏈路)耦接到記憶體橋105；在一個實施例中，平行處理子系統112是圖形子系統，其將像素遞送到顯示器裝置110(例如，習知的基於CRT或LCD的監視器)。系統碟片114也連接到I/O橋107。切換器116提供I/O橋107與例如網路配接器118及各種內插卡120及121等其它組件之間的連接。其它組件(未明確展示)包含USB或其它埠連接、CD驅動器、DVD驅動器等，也可連接到I/O橋107。使圖1中的各個組件互連的通信路徑可使用任何適宜的協定來實施，例如PCI(周邊組件互連)、PCI Express(PCI-E)、AGP(加速圖形埠)、HyperTransport，或任何其它匯流排或點對點通信協定，且不同裝置之間的連接可使用此項技術中已知的不同協定。

平行處理子系統112包含平行處理單元(PPU)122及平行處理(PP)記憶體124，其可(例如)使用例如可程式設計處理器、特殊應用積體電路(ASIC)及記憶體裝置等的一或多個積體電路裝置來實施。PPU 122有利地實施包含一或多個處理核心的高度平行處理器，所述處理核心的每一者能夠併發執行大量(例如，數百個)緒。PPU 122可經程式設計以執行一大批計算，包含線性及非線性資料變換、視訊及/或音訊資料的過濾、模型化(例如，應用物理定律來確定物件的位置、速度及其它屬性)、影像渲染等。PPU 122可將資料從系統記憶體104及/或PP記憶體124轉移到內部記憶體中，處理所述資料，並將結果資料寫回到系統記憶體104及/或PP記憶體124，在其中此類資料可由其它系統組件(包含，例如，CPU 102)存取。在一些實施例中，PPU 122是圖形處理器，其還可經組態以執行與以下操作有關的各種任務：依據由CPU 102及/或系統記憶體104經由記憶體橋105及匯流排113供應的圖形資料產生像素資料、與PP記憶體124(其可用作圖形記憶體，包含(例如)習知圖框緩衝器)交互以儲存及更新像素資料、將像素資料遞送到顯示器裝置110等。在一些實施例中，PP子系統112可包含一個作為圖形處理器操作的PPU 122，及另一用於進行通用計算的PPU 122。PPU可相同或不同，且每一PPU可具有其自身的專用PP記憶體裝置。

CPU 102作為系統100的主處理器操作，其控制及協調其它系統組件的操作。具體而言，CPU 102發布控制PPU 122

的操作的命令。在一些實施例中，CPU 102將針對PPU 122的命令流寫入到命令緩衝器，命令緩衝器可處於系統記憶體104、PP記憶體124或CPU 102及PPU 122兩者均可存取的另一儲存位置中。PPU 122從命令緩衝器讀取命令流，並與CPU 102的操作不同步地執行命令。

將瞭解，本文展示的系統是說明性的，且可能作出變化及修改。可視需要修改連接拓撲，包含橋的數目及組態。舉例而言，在一些實施例中，系統記憶體104直接而不藉由橋連接到CPU 102，且其它裝置經由記憶體橋105及CPU 102與系統記憶體104通信。在其它替代拓撲中，PP子系統112連接到I/O橋107而不連接到記憶體橋105。在另外其它實施例中，I/O橋107及記憶體橋105可能整合到單一晶片中。本文展示的特定組件是任選的；舉例而言，可能支援任何數目的內插卡或周邊裝置。在一些實施例中，排除切換器116，且網路配接器118及內插卡120、121直接連接到I/O橋107。

還可改變PPU 122到系統100的其餘部分的連接。在一些實施例中，PP系統112被實施為可插入到系統100的擴充插槽中的內插卡。在其它實施例中，PPU可與匯流排橋(例如，記憶體橋105或I/O橋107)一起整合在單一晶片上。在另外其它實施例中，PPU 122的一些或所有元件可與CPU 102整合。

PPU可具備任何量的區域PP記憶體，包含無區域記憶體，且可以任何組合使用區域記憶體與系統記憶體。舉例

而言，在統一記憶體結構(UMA)實施例中，PPU 122可以是圖形處理器；在此類實施例中，提供極少或不提供專門的圖形記憶體，且PPU 122將專門地或幾乎專門地使用系統記憶體。在UMA實施例中，PPU可整合到橋晶片中或提供成離散晶片，其中高速鏈路(例如，PCI-E)將PPU連接到橋晶片及系統記憶體。

還將瞭解，可(例如)藉由在單一內插卡上包含多個PPU，藉由將多個內插卡連接到路徑113，及/或藉由將一或多個PPU直接連接到系統主機板，而將任何數目的PPU包含在系統中。多個PPU可平行操作，從而以高於利用單一PPU可能實現的處理量處理資料。

熟習此項技術者還將瞭解，CPU及PPU可整合到單一裝置中，且CPU及PPU可共用例如指令邏輯、緩衝器、高速緩衝記憶體、記憶體、處理引擎等各種資源；或者可針對平行處理及其它操作提供單獨的資源。因此，本文描述為與PPU相關聯的任何或所有電路及/或功能性也可在適當裝備的CPU中實施並由所述適當裝備的CPU執行。

併入有PPU的系統可以多種組態及形狀因素進行實施，包含桌上型電腦、膝上型電腦，或掌上型個人電腦、伺服器、工作站、遊戲機、嵌入式系統等。

熟習此項技術者還將瞭解，本發明的一個優點是相對於特定計算硬體的獨立性增加。因此，將瞭解，本發明實施例可使用任何電腦系統(包含不提供PPU的系統)來實踐。

2. 虛擬程式設計模型概述

在本發明實施例中，需要使用PPU 122或計算系統的其它處理器來使用緒陣列執行通用計算。如本文所使用，"緒陣列"是由對輸入資料集併發執行相同程式以產生輸出資料集的若干(n0)緒組成的群組。緒陣列中的每一緒被分派有唯一緒識別符("緒ID")，其可由緒在其執行期間存取。可定義為一維或多維數值(例如，0到n_0-1)的緒ID控制緒的處理行為的各個態樣。舉例而言，緒ID可用於確定緒將處理輸入資料集的哪一部分，及/或確定緒將產生或寫入輸出資料集的哪一部分。

在一些實施例中，緒陣列是"協作的"緒陣列或CTA。與其它類型的緒陣列一樣，CTA是對輸入資料集併發執行相同程式(本文稱為"CTA程式")以產生輸出資料集的多個緒的群組。在CTA中，緒可藉由以依賴於緒ID的方式彼此共用資料而進行協作。舉例而言，在CTA中，資料可由一個緒產生並由另一緒消耗。在一些實施例中，可在將共用資料的點處將同步指令插入到CTA程式代碼中，以確保在消耗資料的緒試圖訪問資料之前已由產生資料的緒實際產生資料。CTA的緒之間的資料共用(如果存在的話)的程度由CTA程式確定；因此，將瞭解，在使用CTA的特定應用中，CTA的緒可能或可能不實際上彼此共用資料，這取決於CTA程式，且本文中同義地使用術語"CTA"及"緒陣列"。

在一些實施例中，CTA中的緒與同一CTA中的其它緒共用輸入資料及/或中間結果。舉例而言，CTA程式可能包含用於計算共用記憶體中的特定資料將被寫入到的位址的指

令，其中位址是緒ID的函數。每一緒使用其自身的緒ID來計算函數，並寫入到相應位置。有利地定義位址函數使得不同緒寫入到不同位置；只要函數是確定性的，就可預測任何緒寫入到的位置。CTA程式還可包含用於計算共用記憶體中的將從中讀取資料的位址的指令，其中位址是緒ID的函數。藉由定義適宜的函數及提供同步技術，資料可以可預測方式由CTA的一個緒寫入到共用記憶體中的給定位置，並由同一CTA的不同緒從該位置進行讀取。因此，可支援緒之間的任何所需的資料共用模式，且CTA中的任何緒可與同一CTA中的任何其它緒共用資料。

有利地使用CTA(或其它類型的緒陣列)來執行便於資料平行分解(data-parallel decomposition)的計算。如本文所使用，"資料平行分解"包含其中藉由對輸入資料平行執行同一演算法多次以產生輸出資料來對計算問題求解的任何情形；舉例而言，資料平行分解的一個普通實例涉及將同一處理演算法應用於輸入資料集的不同部分以便產生輸出資料集的不同部分。適於資料平行分解的問題的實例包含矩陣代數、任何數目的維度的線性及/或非線性變換(例如，快速傅立葉變換)，及包含任何數目的維度的卷積濾波、多個維度的可分離濾波等各種濾波演算法。在CTA程式中指定待應用於輸入資料集的每一部分的處理演算法，且CTA中的每一緒對輸入資料集的一個部分執行同一CTA程式。CTA程式可使用廣範圍的數學及邏輯運算來實施演算法，且所述程式可包含有條件或分支執行路徑以及直接

及/或間接記憶體存取。

上文引用的申請案第11/303,780號中進一步詳細描述了CTA及其執行。

在一些情形中，定義相關CTA(或更一般來說，緒陣列)的"柵格"也是有用的。如本文所使用，CTA的"柵格"是若干(n1)個CTA的集合，其中所有CTA為相同大小(即，緒數目)並執行相同的CTA程式。柵格內的n1個CTA有利地彼此獨立，意味著柵格中任何CTA的執行不受柵格中的任何其它CTA的執行的影響。如將瞭解，此特徵提供在可用的處理核心之間分配CTA的顯著靈活性。

為了區分柵格內的不同CTA，有利地向柵格的每一CTA分派"CTA識別符"(或CTA ID)。與緒ID一樣，任何唯一識別符(包含但不限於數值識別符)均可用作CTA ID。在一個實施例中，CTA ID只是從0到n_1-1的連續(一維)指數值。在其它實施例中，可使用多維指數標定方案。CTA ID對於CTA的所有緒是共同的，且柵格內給定CTA的緒可使用其CTA ID並結合其緒ID來確定(例如)用於讀取輸入資料的源位置及/或用於寫入輸出資料的目的地位置。以此方式，同一柵格的不同CTA中的緒可對同一資料集併發操作，但在一些實施例中，不支援柵格中的不同CTA之間的資料共用。

定義CTA柵格(例如)在需要使用多個CTA來對單一大問題的不同部分求解的情況下可能是有用的。舉例而言，可能需要執行濾波演算法以產生高清晰度電視(HDTV)影

像。如此項技術中已知，HDTV影像可能包含2百萬個以上的像素。如果每一緒產生一個像素，那麼待執行的緒數目將超過單一CTA中可處理的緒數目(假定為使用習知技術構造的具有合理尺寸及成本的處理平台)。

可藉由在多個CTA之間劃分影像使每一CTA產生輸出像素的不同部分(例如，16x16小片)來管理這種大處理任務。所有CTA執行相同程式，且緒使用CTA ID與緒ID的組合來確定用於讀取輸入資料及寫入輸出資料的位置，使得每一CTA對輸入資料集的正確部分操作並將其輸出資料集部分寫入到正確位置。

應注意，與CTA內的緒(其可共用資料)不同，柵格內的CTA有利地不彼此共用資料或以另外的方式彼此依賴。也就是說，可循序(以任一次序)或併發執行同一柵格的兩個CTA，且仍產生相同結果。因此，處理平台(例如，圖1的系統100)可藉由首先執行一個CTA接著執行下一CTA等等直到已執行柵格的所有CTA為止，來執行CTA柵格並獲得結果。或者，如果有足夠的資源可用，處理平台可藉由平行執行多個CTA來執行相同柵格並獲得相同結果。

在一些實例中，可能需要定義CTA的多個(n2)柵格，其中每一柵格執行資料處理程式或任務的不同部分。舉例而言，資料處理任務可能劃分為若干個"求解步驟"，其中藉由執行CTA柵格來執行每一求解步驟。作為另一實例，資料處理任務可能包含對一連串輸入資料集(例如，連續的視訊資料圖框)執行相同或類似的操作；可針對每一輸入

資料集執行CTA柵格。虛擬程式設計模型有利地支援至少這三個等級的工作定義(即，緒、CTA及CTA柵格)；視需要還可支援額外的等級。

將瞭解，用於對特定問題求解的CTA的大小(緒的數目n_0)、柵格的大小(CTA的數目n_1)及柵格的數目(n_2)將取決於問題的參數及定義問題分解的程式設計人員或自動代理的偏好。因此，在一些實施例中，CTA的大小、柵格的大小及柵格的數目有利地由程式設計人員定義。

受益於CTA方法的問題通常特徵在於，存在可被平行處理的大量資料元素。在一些實例中，資料元素是輸出元素，其每一者是藉由對輸入資料集的不同(可能重疊)部分執行相同演算法而產生的。在其它實例中，資料元素可以是輸入元素，其每一者將使用相同演算法予以處理。

此類問題可始終分解為至少兩個等級並映射到上文描述的緒、CTA及柵格上。舉例而言，每一柵格可能表示複雜資料處理任務中的一個求解步驟的結果。每一柵格有利地劃分為若干"區塊"，其每一者可作為單一CTA被處理。每一區塊有利地含有多個"元素"，即待求解的問題的基本部分(例如，單一輸入資料點或單一輸出資料點)。在CTA內，每一緒處理一或多個元素。

圖2A及2B說明在本發明實施例中使用的虛擬程式設計模型中柵格、CTA與緒之間的關係。圖2A展示若干柵格200，每一柵格由CTA 202的二維(2-D)陣列組成。(本文中，相似物件的多個實例用表示所述物件的參考標號及

(需要時)表示實例的括弧標號來指示。)如圖2B中針對202(0,0)所示，每一CTA 202包含緒(⊖) 204的2-D陣列。對於每一柵格200的每一CTA 202中的每一緒204，可定義I = [i_g, i_c, i_t]的形式的唯一識別符，其中柵格識別符i_g唯一地識別柵格，CTA ID i_c唯一地識別柵格內的CTA，且緒ID i_t唯一地識別CTA內的緒。在此實施例中，可由一維柵格識別符i_g、二維CTA識別符i_c及二維緒識別符i_t來構成識別符I。在其它實施例中，唯一識別符I是整數的三元組，其中$0 \leq i_g < n_2$；$0 \leq i_c < n_1$；且$0 \leq i_t < n_0$。在另外其它實施例中，柵格、CTA及緒識別符中的任一者或全部可能表達為一維整數、2D座標對、3D三元組等。唯一緒識別符I可用於(例如)確定包含針對整個柵格或多個柵格的輸入資料集的陣列內的輸入資料的源位置，及/或確定用於在包含針對整個柵格或多個柵格的輸出資料集的陣列內儲存輸出資料的目標位置。

舉例而言，在HDTV影像的情況下，每一緒204可能對應於輸出影像的像素。CTA 202的大小(緒204的數目)是問題分解中的選擇事項，其僅受對單一CTA 202中的最大緒數目的約束的限制(其反映處理器資源的有限性)。柵格200可對應於HDTV資料的整個圖框，或者多個柵格可映射到單一圖框。

在一些實施例中，問題分解是統一的，意味著所有柵格200具有相同數目及組態的CTA 202，且所有CTA 202具有相同數目及組態的緒204。在其它實施例中，分解可能不

統一。舉例而言，不同柵格可能包含不同數目的CTA，且不同CTA(相同柵格或不同柵格中)可能包含不同數目的緒。

如上定義的CTA可包含數打或甚至數百個併發緒。上面將執行CTA的平行處理系統可能或可能不支援如此大量的併發緒。在一個態樣，本發明藉由允許程式設計人員在不考慮實際硬體能力的情況下使用CTA及CTA柵格的模型來定義處理任務，使程式設計人員脫離於此類硬體限制。舉例而言，程式設計人員可編寫代碼("CTA程式")，其定義待由CTA的單一代表性線緒執行的處理任務；將CTA定義為若干此類緒，每一緒具有唯一識別符；並將柵格定義為若干CTA，每一CTA具有唯一識別符。如下文所描述，此類代碼被自動轉譯為可在特定平台上執行的代碼。舉例而言，如果將CTA定義為包含若干(n_0)個併發緒但目標平台僅支援一個緒，那麼翻譯器可定義執行分派到所有n_0個緒的任務的一個實際緒。如果目標平台支援一個以上但少於n_0個併發緒，那麼可視需要在所述數目的可用緒之間劃分任務。

因此，CTA及柵格的程式設計模型應理解為虛擬模型，即與任何特定實體實現脫離的作為對程式設計人員的概念上的輔助的模型。可在對平行處理具有不同程度的硬體支援的多種目標平台中實現CTA及柵格的虛擬模型。具體而言，本文所使用的術語"CTA緒"是指離散處理任務(可能與一或多個其它處理任務協作)的虛擬模型，且應瞭解，CTA

緒可能或可能不一對一地映射到目標平台上的緒。

3. 虛擬架構

根據本發明的一個態樣，定義用於執行CTA及CTA柵格的虛擬平行架構。虛擬平行架構是支援執行能夠在所需時間實現彼此的協作行為(例如，共用資料及同步)的大量併發CTA緒的平行處理器及相關聯的記憶體空間的表示。此虛擬平行架構可映射到多種實際處理器及/或處理系統上，包含(例如)圖1的系統100的PPU 122。虛擬架構有利地定義支援不同等級的資料共用及存取類型的若干虛擬記憶體空間，以及識別可由虛擬處理器執行的所有功能的虛擬指令集架構(ISA)。虛擬架構還有利地(例如)藉由定義及啟動CTA或CTA柵格來定義可用於控制CTA執行的虛擬執行驅動器。

圖3是根據本發明實施例的虛擬架構300的方塊圖。虛擬架構300包含帶有虛擬核心308的虛擬處理器302，虛擬核心308經組態以平行執行大量CTA緒。虛擬架構300還包含可由虛擬處理器302存取的全局記憶體304，及供應控制虛擬處理器302的操作的命令的虛擬驅動器320。虛擬驅動器320也可存取全局記憶體304。

虛擬處理器302包含接收及解譯來自虛擬驅動器320的命令的前端306，以及能夠併發執行單一CTA的所有n_0個緒的執行核心308。虛擬核心308包含大量(n_0個或n_0個以上)虛擬處理引擎310；在一個實施例中，每一虛擬處理引擎310執行一個CTA緒。虛擬處理引擎310併發執行其各自CTA

緒，但不一定平行執行。在一個實施例中，虛擬架構300指定虛擬處理引擎310的數目T(例如，384、500、768等)；此數目設定CTA中的緒數目n_0的上限。應瞭解，虛擬架構300的實現可包含少於指定數目T的實體處理引擎，且單一處理引擎可執行若干CTA緒，作為單一"真實"(即，平台支援的)緒或作為多個併發的真實緒。

虛擬處理器302還包含虛擬指令單元312，其保持向虛擬處理引擎310供應針對其各自CTA緒的指令；所述指令由作為虛擬架構300的一部分的虛擬ISA定義。下文描述用於平行緒計算的虛擬ISA的實例。指令單元312在向虛擬處理引擎310供應指令的過程中管理CTA緒同步及CTA緒行為的其它協作性態樣。

執行核心308提供具有不同等級的可存取性的內部資料儲存。特殊暫存器311可由虛擬處理引擎310讀取但不可由其寫入，且用於儲存定義圖2的問題分解模型內每一CTA緒的"位置"的參數。在一個實施例中，特殊暫存器311對於每CTA緒(或每虛擬處理引擎310)包含一個儲存緒ID的暫存器；每一緒ID暫存器僅可由虛擬處理引擎310中的個別虛擬處理引擎310存取。特殊暫存器311還可包含可由所有CTA緒(或由所有虛擬處理引擎310)讀取的額外暫存器，其儲存CTA識別符、CTA維度、CTA所屬的柵格的維度，及CTA所屬的柵格的識別符。在初始化期間響應於經由前端306從虛擬驅動器320接收的命令而寫入特殊暫存器311，且特殊暫存器311在CTA執行期間不改變。

區域虛擬暫存器314由每一CTA緒用作臨時空間(scratch space)；分配每一暫存器專用於一個CTA緒(或一個虛擬處理引擎310)，且區域暫存器314的任一者中的資料僅可由其被分配到的CTA緒存取。共用記憶體316可由所有CTA緒(單一CTA內)存取；共用記憶體316中的任何位置可由同一CTA內的任何CTA緒存取(或可由虛擬核心308內的任何虛擬處理引擎310存取)。參數記憶體318儲存可由任何CTA緒(或任何虛擬處理引擎310)讀取但不可由其寫入的執行時間參數(常數)。在一個實施例中，虛擬驅動器320在引導虛擬處理器302開始執行使用此等參數的CTA之前，將參數提供到參數記憶體318。任何CTA內的任何CTA緒(或虛擬核心308內的任何虛擬處理引擎310)均可藉由記憶體介面322存取全局記憶體304。

在虛擬架構300中，虛擬處理器302在虛擬驅動器320的控制下作為共處理器操作。虛擬架構規範有利地包含虛擬應用程式介面(API)，其識別由虛擬驅動器320認可的函數調用及每一函數調用預期產生的行為。下文描述用於平行緒計算的虛擬API的實例函數調用。

虛擬架構300可在多種硬體平台上實現。在一個實施例中，虛擬架構300在圖1的系統100中實現，其中PPU 122實施虛擬處理器302，且在CPU 102上執行的PPU驅動器程式實施虛擬驅動器320。全局記憶體304可在系統記憶體104及/或PP記憶體124中實施。

在一個實施例中，PPU 122包含一或多個處理核心，其

使用單指令多資料(SIMD)及多線緒技術來支援來自單一指令單元(實施虛擬指令單元312)的大量(例如，384或768個)緒的併發執行。每一核心包含P個(例如，8、16等)平行處理引擎302的陣列，其經組態以從指令單元接收並執行SIMD指令，從而允許平行處理具有多達P個緒的群組。核心為多線緒的，其中每一處理引擎能夠(例如)藉由維持與每一緒相關聯的當前狀態信息使得處理引擎可從一個緒快速切換到另一緒，而併發執行多達某一數目G(例如，24個)的緒群組。因此，核心併發執行各有P個緒的G個SIMD群組，總計為P*G個併發緒。在此實現過程中，只要P*G ≥ n_0，(虛擬)CTA緒與在真實PPU 122上執行的併發緒之間就可存在一對一對應。

特殊暫存器311可藉由以下操作而實施在PPU 122中：向每一處理核心提供P*G輸入項暫存器堆，其中每一輸入項能夠儲存一緒ID；以及提供一組全局可讀取的暫存器來儲存CTA ID、柵格ID及CTA及柵格維度。或者，可使用其它儲存位置來實施特殊暫存器311。

區域暫存器314可實施在PPU 122中，作為在實體上或邏輯上劃分為P個巷道(lane)的區域暫存器堆，其中每一巷道具有某一數目的輸入項(其中每一輸入項可能儲存(例如)32位元字組)。一個巷道被分派到P個處理引擎中的每一者，且不同巷道中的相應輸入項可用執行相同程式以有助於SIMD執行的不同緒的資料填充。巷道的不同部分可分配到G個併發緒群組中的不同緒群組，使得區域暫存器堆中

的給定輸入項僅可由特定緒存取。在一個實施例中，區域暫存器堆內的某些輸入項保留用於儲存緒識別符，從而實施特殊暫存器311中的一者。

共用記憶體316可實施在PPU 122中作為共用暫存器堆或共用晶片上高速緩衝記憶體，其具有允許任何處理引擎對共用記憶體中的任何位置進行讀取或寫入的互連。參數記憶體318可實施在PPU 122中作為實施共用記憶體316的同一共用暫存器堆或共用高速緩衝記憶體內的指定區域，或作為處理引擎具有唯讀(read-only)存取權的單獨共用暫存器堆或晶片上高速緩衝記憶體。在一個實施例中，實施參數記憶體的區域還用於儲存CTA ID及柵格ID以及CTA及柵格維度，從而實施特殊暫存器311的若干部分。

在一個實施例中，在圖1的CPU 102上執行的PPU驅動器程式藉由將命令寫入到記憶體(例如，系統記憶體104)中的推播緩衝器(pushbuffer)(未明確展示)來響應虛擬API函數調用，PPU 122從推播緩衝器處讀取所述命令。命令有利地與狀態參數相關聯，狀態參數例如CTA中緒的數目、待使用CTA處理的輸入資料集在全局記憶體中的位置、待執行的CTA程式在全局記憶體中的位置，及輸出資料將被寫入到的全局記憶體中的位置。響應於命令及狀態參數，PPU 122將狀態參數載入到其核心中的一者中，接著開始啟動緒直到已啟動CTA參數中指定的數目的緒為止。在一個實施例中，PPU 122包含控制邏輯，其在啟動緒時將緒ID依次分派到緒；可將緒ID儲存(例如)在區域暫存器堆內

或專用於此目的的特殊暫存器中的指定位置。

在替代實施例中，在單線緒處理核心中(例如，在一些CPU中)實現虛擬架構300，單線緒處理核心使用少於n_0個實際緒來執行所有CTA緒；虛擬程式設計模型使得與不同CTA緒相關聯的處理任務可(例如)藉由針對一個CTA緒接著針對下一CTA緒等等執行任務(或其一部分)而組合到單一緒中。可利用向量執行、SIMD執行及/或機器中可用的任何其它形式的平行性來平行執行與多個CTA緒相關聯的處理任務，或平行執行與同一CTA緒相關聯的多個處理任務。因此，可使用單一緒、n_0個緒或任何其它數目的緒來實現CTA。如下文所描述，虛擬指令翻譯器有利地將編寫為目標虛擬架構300的代碼轉譯為特定針對目標平台的指令。

將瞭解，本文描述的虛擬架構是說明性的且可能作出變化及修改。舉例而言，在一個替代實施例中，每一虛擬處理引擎可具有儲存分派到其緒的唯一緒ID的專門的緒ID暫存器，而不使用區域虛擬暫存器中的空間用於此目的。

作為另一實例，虛擬架構可指定關於虛擬核心308的內部結構的或多或少的細節。舉例而言，可能指定虛擬核心308包含用於執行P向SIMD群組中的CTA緒的P個多線緒虛擬處理引擎，其中多達G個SIMD群組共存於核心308中使得P*G確定T(CTA中緒的最大數目)。還可指定不同類型的記憶體及共用等級。

虛擬架構可實現於使用硬體及/或軟體元件的任何組合

的多種電腦系統中以定義及控制每一組件。雖然已以舉例的方式描述使用硬體組件的一個實現方案，但應瞭解，本發明涉及使程式設計任務與特定硬體實現脫離。

4. 對虛擬架構進行程式設計

圖4是根據本發明實施例使用虛擬架構300來操作目標處理器或平台440的概念模型400。如模型400所展示，虛擬架構300的存在使經編譯的應用程式及API與目標處理器或平台的硬體實施脫離。

應用程式402定義資料處理應用，其利用上述虛擬程式設計模型(包含單一CTA及/或CTA柵格)。一般來說，應用程式402包含多個態樣。第一，程式定義單一CTA緒的行為。第二，程式定義CTA的維度(以CTA緒的數目計)，且如果將使用柵格，那麼定義柵格的維度(以CTA的數目計)。第三，程式定義待由CTA(或柵格)處理的輸入資料集及輸出資料集將被儲存在的位置。第四，程式定義總體處理行為，包含(例如)何時啟動每一CTA或柵格。程式可包含動態確定CTA或柵格的維度、是否不斷啟動新的CTA或柵格等的額外代碼。

可用例如C/C++、FORTRAN等高階程式設計語言編寫應用程式402。在一個實施例中，C/C++應用程式直接指定一個(虛擬)CTA緒的行為。在另一實施例中，使用資料平行語言(例如，Fortran 90、C*或資料平行C)編寫應用程式，且應用程式指定對陣列及聚合資料結構的資料平行操作；此程式可編譯成指定一個(虛擬)CTA緒的行為的虛擬ISA程

式代碼。為了允許定義CTA緒的行為，可提供語言擴充或函數庫，程式設計人員可經由所述語言擴充或函數庫來指定平行CTA緒行為。舉例而言，可定義特殊符號或變數以對應於緒ID、CTA ID及柵格ID，且可提供函數，程式設計人員可經由所述函數來指示CTA緒應何時與其它CTA緒同步。

當編譯應用程式402時，編譯器408針對應用程式402的定義CTA緒行為的那些部分產生虛擬ISA代碼410。在一個實施例中，以圖3的虛擬架構300的虛擬ISA表達虛擬ISA代碼410。虛擬ISA代碼410是程式代碼，但不一定是可在特定目標平台上執行的形式的代碼。如此，虛擬ISA代碼410可作為任何其它程式代碼被儲存及/或分布。在其它實施例中，可將應用程式完全或部分指定為虛擬ISA代碼410，且可完全或部分避免使用編譯器408。

虛擬指令翻譯器412將虛擬ISA代碼410轉換為目標ISA代碼414。在一些實施例中，目標ISA代碼414是可由目標平台440直接執行的代碼。舉例而言，如由圖4中的虛線框所示，在一個實施例中，目標ISA代碼414可由PPU 122中的指令單元430接收及正確解碼。視目標平台440的特殊性而定，虛擬ISA代碼410可能被轉譯成待由目標平台440上的n_0個緒的每一者執行的每線緒代碼。或者，虛擬ISA代碼410可能被轉譯成將在少於n_0個的緒中執行的程式代碼，其中每一緒包含與CTA緒中的一者以上有關的處理任務。

在一些實施例中，CTA及/或柵格的維度的定義以及定義

輸入資料集及輸出資料集是由虛擬API處理的。應用程式402可包含對於虛擬API函數庫404的調用。在一個實施例中，將虛擬API的規範(包含(例如)函數名稱、輸入、輸出及作用，但不包含實施細節)提供給程式設計人員，且程式設計人員將虛擬API調用直接併入到應用程式402中，藉此直接產生虛擬API代碼406。在另一實施例中，藉由編譯使用某一其它語法來定義CTA及柵格的應用程式402來產生虛擬API代碼406。

部分地藉由提供虛擬執行驅動器416來實現虛擬API代碼406，虛擬執行驅動器416將代碼406的虛擬API命令轉譯為可由目標平台440處理的目標API命令418。舉例而言，如由圖4中的虛線框所示，在一個實施例中，目標API命令418可由PPU驅動器432接收及處理，PPU驅動器432將相應命令傳送到PPU 122前端434。(在此實施例中，虛擬執行驅動器416可以是PPU驅動器432的一態樣或部分。)在另一實施例中，虛擬執行驅動器可能不對應於用於共處理器的驅動器；其可能只是在執行所述虛擬執行驅動器的同一處理器上啟動其它程式或緒的控制程式。

應瞭解，可針對能夠支援CTA執行的任何平台或結構創建虛擬指令翻譯器412及虛擬執行驅動器416。就針對不同平台或結構的虛擬指令翻譯器412可從同一虛擬ISA進行轉譯而言，同一虛擬ISA代碼410可與任何平台或結構一起使用。因此，不需要針對每一可能的平台或結構重新編譯應用程式402。

此外，目標平台440不必包含如圖4所示的PPU及/或PPU驅動器。舉例而言，在一個替代實施例中，目標平台是使用軟體技術仿真大量緒的併發執行的CPU，且目標ISA代碼及目標API命令對應於待由目標CPU執行的程式(或互相通信的程式的群組)中的指令，所述目標CPU可以是(例如)單核心或多核心CPU。

5. 虛擬ISA實例

現在將描述根據本發明實施例的虛擬ISA的實例。如上所述，虛擬ISA有利地對應於上述虛擬程式設計模型(CTA及柵格)。因此，在本實施例中，由編譯器408產生的虛擬ISA代碼410定義要由圖3的虛擬核心308中的虛擬處理引擎310之一執行的單個CAT緒的行為；所述行為可包含與其它CTA緒的協作性交互，例如同步及/或資料共用。

應瞭解，本文中描述的虛擬ISA只用於說明的用途，且本文中描述的特定要素或要素組合並不限制本發明的範圍。在一些實施例中，程式設計人員可用虛擬ISA編寫代碼；在其它實施例中，程式設計人員用另一高階語言(例如，FORTRAN, C, C++)編寫代碼，且編譯器408產生虛擬ISA代碼。程式設計人員也可編寫"混合"代碼，其中代碼中有些部分是用高階語言編寫而其它部分是用虛擬ISA編寫。

5.1 特殊變數

圖5是列舉由實例虛擬ISA定義的"特殊"變數的表格500(本文中用前綴"%"來表示特殊變數)。此等變數與圖2

的程式設計模型有關，其中藉由每個緒204在CTA 202內的位置來識別線緒，而CTA 202又位於某一數目的柵格200中的特定一者內。在一些實施例中，表格500的特殊變數對應於圖3的虛擬架構300中的特殊暫存器311。

在表格500中，假設CTA及柵格各自用三維空間來定義，且不同的柵格在一維空間中循序編號。虛擬ISA預期圖5的特殊變數將在啟動CTA時被初始化，且虛擬ISA代碼可簡單地使用此等變數而無需初始化。以下參看虛擬API論述特殊變數的初始化。

如圖5所示，特殊變數的第一個3向量%ntid=(%ntid.x, % ntid.y, %ntid.z)定義CTA的維度(以緒數目計)。CTA中的所有緒將共用同一%ntid向量。在虛擬架構300中，預期%ntid向量的值將經由虛擬API函數調用提供給虛擬處理器302，所述虛擬API函數調用如下所述地建立CTA的維度。

如圖5所示，特殊變數的第二個3向量%tid=(%tid.x, % tid.y, %tid.z)是指CTA內的給定緒的緒ID。在圖3的虛擬架構300中，預期虛擬處理器302將在啟動CTA的每個緒時分派滿足制約條件0≤ %tid.x < %ntid.x、0 ≤ %tid.y < %ntid.y及0 ≤ %tid.z < %ntid.z的唯一%tid向量。在一個實施例中，%tid向量可定義成使得其可儲存在壓縮的32位元的字組中(例如，對於%tid.x為16個位元，對於%tid.y為10個位元，且對於%tid.z為6個位元)。

如圖5所示，特殊變數的第三個3向量%nctaid=(%nctaid.x, %nctaid.y, %nctaid.z)定義柵格的維度(以CTA

的數目計)。在圖3的虛擬架構300中，預期%nctaid向量的值將經由虛擬API函數調用提供給虛擬處理器302，虛擬API函數調用建立CTA的柵格的維度。

如圖5所示，特殊變數的第四個3向量%ctaid=(%ctaid.x, %ctaid.y, %ctaid.z)是指柵格內的給定CTA的CTA ID。在圖3的虛擬架構300中，預期當啟動CTA時將把滿足CTA的限制條件 $0 \leq$ %ctaid.x $<$ %nctaid.x、$0 \leq$ %ctaid.y $<$ %nctaid.y及 $0 \leq$ %ctaid.z $<$ %nctaid.z的唯一%ctaid向量提供給虛擬處理器302。

特殊變數還包含提供CTA所屬的柵格的柵格識別符的純量%gridid變數。在圖3的虛擬架構300中，預期將把%gridid值提供給虛擬處理器302，以便識別包括當前CTA的柵格。例如當正在使用多個柵格來解決較大問題的不同部分時，有利地在虛擬ISA代碼中使用%gridid值。

5.2 程式定義的變數及虛擬狀態空間

虛擬ISA使得程式設計人員(或編譯器)可定義任意數目的變數以代表正被處理的資料項目。藉由類型及指示如何使用變數以及變數在何種程度上共用的"虛擬狀態空間"來定義變數。使用目標平台中可用的暫存器或其它記憶體結構來實現變數；在許多目標平台中，狀態空間可能會影響對用來實現特定變數的記憶體結構的選擇。

圖6是列舉實例虛擬ISA實施例中支援的變數類型的表格600。支援四個類型：未指定類型的位元、有正負號的整數、無正負號整數及浮點。未指定類型的變數就是單個位

元或具有指定長度的位元的群組。可根據習知格式(例如，IEEE 754標準)來定義有正負號整數及無正負號整數格式以及浮點格式。

在本實施例中，針對每個類型支援多個寬度，其中使用參數<n>來指定寬度；因此，舉例而言，.s16指示16個位元的有正負號的整數，.f32指示32位元的浮點數字等。如表格600所示，有些變數類型限於特定的寬度；舉例而言，浮點變數必須至少為16個位元；且整數類型必須至少為8個位元。預期虛擬ISA的實現支援所有指定寬度；如果處理器的資料路徑及/或暫存器比最寬寬度窄，那麼如此項技術中已知的可使用多個暫存器及處理器循環來處置較寬類型。

應瞭解，本文中使用的資料類型及寬度是說明而不是限制本發明。

圖7是列舉實例虛擬ISA中支援的虛擬狀態空間的表格。定義了九個狀態空間，其對應於不同的共用級別及在圖3的虛擬架構300中的可能的儲存位置。

在緒級別上共用前三個狀態空間，這意味著每個CTA緒將具有單獨的變數實例，且沒有任何CTA緒將可存取任何其它CTA緒的實例。有利地使用虛擬暫存器(.reg)狀態空間來定義要由每個CTA緒執行的計算的運算數、臨時值及/或結果。程式可宣稱任何數目的虛擬暫存器。只能藉由靜態編譯時間名稱而不是計算出來的位址來定址虛擬暫存器。這個狀態空間對應於圖3的虛擬架構300中的區域虛擬暫存

器314。

特殊暫存器(.sreg)狀態空間對應於圖5的預定義的特殊變數，其儲存在虛擬架構300中的特殊暫存器311中。在一些實施例中，虛擬ISA代碼可能不會宣稱.sreg空間中的任何其它變數，而是可使用特殊變數作為對計算的輸入。所有CTA緒均可讀取.sreg狀態空間中的任何變數。對於%tid(或其分量)，每個CTA緒將讀取其唯一的緒識別符；對於.sreg狀態空間中的其它變數，同一CTA中的所有CTA緒將讀取相同值。

每線緒區域記憶體(.local)變數對應於全局記憶體304中在每CTA緒基礎上分配及定址的區域。換句話說，當CTA緒存取.local變數時，其存取其自身的變數實例，且在一個CTA緒中對.local變數作出的改變不會影響其它CTA緒。與.reg及.sreg狀態空間不同，每線緒區域記憶體可使用計算出來的位址來定址。

接下來的兩個狀態空間定義每CTA變數，這意味著每個CTA將具有變數的一個實例，其任何(虛擬)緒均可存取所述實例。任何CTA緒均可讀取或寫入共用(.shared)變數。在一些實施例中，這個狀態空間映射到虛擬架構300(圖3)的虛擬共用記憶體316。在虛擬架構300的實現中，.shared狀態空間可映射到晶片上共用記憶體實施方案(例如，共用暫存器堆或共用高速緩衝記憶體)上，而在其它實現中，.shared狀態空間可映射到晶片外記憶體的分配及定址為任何其它可全局存取記憶體的每CTA區域上。

參數(.param)變數是唯讀的，且可由CTA中的任何(虛擬)緒讀取。這個狀態空間映射到虛擬架構300的參數記憶體318，且可(例如)在晶片上共用參數記憶體或高速緩衝記憶體中或者在可全局存取的晶片外記憶體的分配及定址為其它任何可全局存取記憶體的區域中實現。預期此等變數將響應於來自虛擬驅動器320的驅動器命令來初始化。

常量(.const)狀態空間用來定義可由柵格中的任何CTA中的任何(虛擬)緒讀取(但不可修改)的每柵格常量。在虛擬架構300中，.const狀態空間可映射到全局記憶體中可由CTA緒進行唯讀存取的區域。.const狀態空間可在晶片上共用參數記憶體或高速緩衝記憶體中或在可全局存取晶片外記憶體的分配及定址為任何其它可全局存取記憶體的每柵格區域中實現。與.param狀態空間一樣，預期.const狀態空間中的變數將響應於來自虛擬驅動器320的驅動器命令而被初始化。

其餘三個狀態空間定義"內容"變數，其可供與應用程式相關聯的任何CTA中的任何(虛擬)緒存取。此等狀態空間映射到虛擬架構300中的全局記憶體304。全局(.global)變數可用於一般用途。在一些實施例中，也可定義用於共用紋理(.tex)及表面(.surf)的特定狀態空間。此等可用於(例如)圖形相關應用程式的狀態空間可用來定義並提供對圖形紋理及像素表面資料結構的存取，所述資料結構提供對應於2D(或在一些實施例中為3D)陣列的每個像素的資料值。

在圖4的虛擬ISA代碼410中，藉由指定狀態空間、類型及名稱來宣稱變數。名稱是占位符，且可由程式設計人員或編譯者來選擇。因此，例如：

.reg .b32 vr1;

宣稱虛擬暫存器狀態空間中名為vr1的32個位元的未指定類型變數。虛擬ISA代碼的後續行可引用vr1(例如)作為操作的來源或目的地。

實例虛擬ISA還支援虛擬變數的陣列及向量。舉例而言：

.global .f32 resultArray[1000][1000];

宣稱32位元浮點數字的虛擬可全局存取1000×1000陣列。虛擬指令翻譯器412可將陣列映射到對應於分派的狀態空間的可定址記憶體區域中。

可使用向量前綴.v<m>來定義一個實施例中的向量，其中m是向量的分量的數目。舉例而言：

.reg .v3 .f32 vpos;

宣稱每線緒虛擬暫存器狀態空間中的32位元浮點數字的3分量向量。一旦宣稱了向量，便可使用例如vpos.x, vpos.y, vpos.z等後綴來識別其分量。在一個實施例中，允許m=2、3或4，且可使用例如(.x, .y, .z, .w)、(.0, .1, .2, .3)或(.r, .g, .b, .a)等後綴來識別分量。

由於變數是虛擬的，所以虛擬ISA代碼410可定義或引用任何狀態空間(除了.sreg，其中變數是預定義的)中的任何數目的變數。可能針對虛擬ISA代碼410中的特定狀態空間

定義的變數的數目可能會超過特定硬體實施方案中的相應類型的儲存量。虛擬指令翻譯器412有利地經組態以包含合適的儲存管理指令(例如，在暫存器與晶片外記憶體之間移動資料)，以便使變數在需要時可用。虛擬指令翻譯器412也可能能夠檢測到不再需要臨時變數的情況並允許為其分配的空間供另一變數重新使用；可使用習知的用於分配暫存器的編譯器技術。

此外，雖然實例虛擬ISA定義向量變數類型，但不要求目標平台支援向量變數。虛擬指令翻譯器412可將任何向量變數實施為適當數目(例如，2、3或4)個純量的集合。

5.3. 虛擬指令

圖8A-8H是列舉實例虛擬ISA中定義的虛擬指令的表格。藉由指令的效果來定義指令，所述效果例如是使用一或多個運算數計算出特定結果並將所述結果放置在目的地暫存器中、設置暫存器值等。將大多數虛擬指令分類以識別輸入及/或輸出的格式，且指令執行的各態樣可取決於類型。指令的一般格式是：

name.<type> result, operands;

其中name是指令名稱；.<type>是圖6中列舉的任一類型的占位符；result是儲存結果的變數；且operands是提供為對指令的輸入的一或多個變數。在一個實施例中，虛擬架構300是暫存器到暫存器處理器，且用於除記憶體存取之外的操作的result及operands(圖8F)被要求是在虛擬暫存器狀態空間.reg(或在一些運算數的情況下是特殊暫存器狀態

空間.sreg)中的變數。

預期目標平台實現虛擬ISA中的每個指令。指令可實現為產生指定效果的相應的機器指令(本文中稱為"硬體支援"),或實現為在執行時產生指定效果的機器指令序列(本文中稱為"軟體支援")。特定目標平台的虛擬指令翻譯器412有利地經組態以識別對應於每個虛擬指令的機器指令或機器指令序列。

以下子章節描述圖8A-8H中列舉的各種類別的指令。應瞭解,本文中提供的指令列表是說明性的,且虛擬ISA可包含本文中未明確描述的額外指令,且可不包括本文中描述的一些或所有指令。

5.3.1. 虛擬指令-算術

圖8A是列舉實例虛擬ISA中定義的算術運算的表格800。在此實施例中,虛擬架構只支援暫存器到暫存器算術,且所有算術運算均操縱一或多個虛擬暫存器運算數(圖8A中表示為a、b、c),以產生寫入到虛擬暫存器的結果(d)。因此,算術運算的運算數及目的地始終在虛擬暫存器狀態空間.reg中,除了圖5的特殊暫存器(在特殊暫存器狀態空間.sreg中)可用作運算數以外。

表格800中的算術運算的列表包含四個基本的算術運算:加法(add)、減法(sub)、乘法(mul)及除法(div)。此等運算可對所有整數及浮點資料類型執行,且產生相同類型的結果作為輸入;在一些實施例中,也可將捨入模式的限定符(qualifier)添加到指令中,以允許程式設計人員指定

應當如何對結果進行捨入，以及在整數運算數的情況下是否應當強加飽和極限。

也支援a、b及c運算數的三個複合算術運算：乘加(mad)、熔合乘加(fma)以及絕對差和(sad)。乘加計算axb的乘積(具有括號指示的捨入)，且將結果與c相加。熔合乘加與mad的區別在於，在與c相加之前，不對axb的乘積進行捨入。絕對差和計算絕對值|a-b|，然後與c相加。

餘數(rem)運算只對整數運算數執行，且當將運算數a除以運算數b時計算餘數(a mod b)。絕對值(abs)及求反(neg)是可應用於浮點格式或有正負號的整數格式的運算數a的一元運算。可應用於整數或浮點運算數的最小值(min)及最大值(max)運算將目的地暫存器設置成較小運算數或較大運算數；也可指定對一個或兩個運算數是非正規數(例如，根據IEEE 754標準)的特殊情況的處置。

表格800中的其餘運算只對浮點類型執行。分數(frc)運算返回其輸入的分數部分。正弦(sin)、餘弦(cos)及比率的反正切(atan2)提供對應於三角函數的方便的指令。也支援以2為底的對數(lg2)及取冪(ex2)。也支援倒數(rcp)、平方根(sqrt)及倒數平方根(rsqrt)。

應注意，這個算術運算列表是說明而不是限制本發明。可支援其它運算或運算組合，其中包含任何預期會以足夠的頻率被調用的運算。

在一些實施例中，虛擬ISA還定義向量運算。圖8B是列舉實例虛擬ISA支援的向量運算的表格810。向量運算包含

點積(dot)運算，其計算運算數向量a及b的純量點積d；叉積(cross)運算，其計算運算數向量a及b的向量叉積d；以及量值(mag)運算，其計算運算數向量a的純量長度d。向量歸約(vred)運算藉由以迭代方式對向量運算數a的要素執行指定操作<op>來計算純量結果d。在一個實施例中，對於浮點向量只支援歸約運算add、mul、min及max；對於整數向量，也可支援額外的歸約運算(例如，and、or及xor，如下文所述)。

除了此等運算之外，也可在虛擬ISA中定義例如向量加法、向量定標等其它向量運算(圖8B中未列舉)。

如上所述，虛擬架構300的一些硬體實現可能不支援向量處理。此種實現的虛擬指令翻譯器412有利地適合於產生純量機器指令的適當序列以執行此等運算；熟習此項技術者將能夠確定適當的序列。

5.3.2. 虛擬指令-選擇及設置暫存器

圖8C是列舉實例虛擬ISA中定義的選擇及設置暫存器運算的表格820。此等運算可對任何數值資料類型執行，且基於比較運算的結果設置目的地暫存器。如果c是非零那麼基本選擇(sel)運算選擇運算數a，且如果c是零則選擇運算數b。比較與設置(set)對運算數a及b執行比較運算<cmp>以產生比較結果t，接著基於比較結果t是真(～0)還是假(0)而將目的地暫存器d設置成布林(Boolean)真(～0)或假(0)。一個實施例中允許的比較運算<cmp>包含等於(如果a=b則t為真)、大於(如果a>b則t為真)、小於(如果a<b則t為真)、

大於或等於(如果a≥b則t為真)、小於或等於(如果a≤b則t為真)，以及其它比較，其中包含例如a及/或b是數字值還是未定義的值。

setb運算是對比較與設置的變化形式，其在比較運算<cmp>的結果t與第三運算數c之間執行進一步的布林運算<bop>；布林運算t<bop>c的結果確定是將目的地暫存器d設置成布林真還是布林假。一個實施例中允許的布林運算<bop>包含and、or及xor(見下述圖8C)。setp運算與setb相似，區別只是設置了兩個1位元"斷言"(predicate)目的地暫存器：將目的地暫存器d1設置成t<bop>c的結果，而將目的地暫存器d2設置成(!t)<bop>c的結果。

5.3.3. 虛擬指令-邏輯及位元操縱

圖8D是列舉實例虛擬ISA中定義的邏輯及位元操縱運算的表格830。按位元的布林運算and、or及xor藉由以下方式來執行：對運算數a及b的每個位元執行指定運算，且將暫存器d中的相應位元設置成結果。按位元求反(not)運算反轉運算數a的每個位元，而如果a是零(布林假)則邏輯求反(cnot)運算將目的地暫存器設置成1(布林真)，否則設置成0(布林假)。

藉由左移(shl)及右移(shr)運算來支援移位，所述運算將運算數a中的位元欄位左移或右移運算數b指定的位元數。對於有正負號的格式，右移有利地基於符號位元來填補引導位元；對於無正負號格式，右移用零來填補引導位元。

5.3.4. 虛擬指令-格式轉換

圖8E是列舉實例虛擬ISA中定義的格式轉換運算的表格840。格式轉換(cvt)指令將第一類型<atpye>的運算數a轉換成目標類型中的均等值<dtpye>，且將結果儲存在目的地暫存器d中。圖6中列舉一個實施例中的有效類型；無法將未指定類型值(.b<n>)轉換成整數或浮點類型或者將整數或浮點類型轉換成未指定類型值。格式轉換指令的變化形式使得程式設計人員可指定捨入模式<mode>；也可指定對當表達為目標類型時飽和的數字的處置。

5.3.5. 虛擬指令-資料移動及資料共用

圖8F是列舉實例虛擬ISA中定義的資料移動及資料共用指令的表格850。移動(mov)操作將目的地暫存器d設置成立即運算數a的值，或者如果運算數a是暫存器，則設置成暫存器a的內容。移動操作可限於虛擬暫存器類型的狀態空間，例如圖7中的.reg及.sreg。

載入(ld)指令將來自記憶體中的源位置的值載入到目的地暫存器d中，所述目的地暫存器d在一個實施例中必須在虛擬暫存器(.reg)狀態空間中。.<space>限定符指定源位置的狀態空間，且可限於圖7中的可定址狀態空間，例如.reg及.sreg之外的空間(其中可改為使用移動操作)。由於這個實施例中的虛擬架構300是暫存器到暫存器處理器，所以載入指令有利地用來將變數從可定址的狀態空間轉移到虛擬暫存器.reg狀態空間中，使其可用作運算數。

使用可用各種方式定義的來源參數<src>來識別特定的源位置，以便支援不同的定址模式。舉例而言，在一些實

施例中，來源參數<src>可能是以下中的任何一者：其值儲存在d中的經命名的可定址變數、對保存來源位址的暫存器的引用、對保存將與偏移值(提供為立即運算數)相加的位址的暫存器的引用，或立即絕對位址。

類似地，儲存(st)操作將來源暫存器a中的值儲存到由目的地參數<dst>識別的記憶體位置。一個實施例中的來源暫存器a必須在.reg狀態空間中；目的地必須在可寫入及可定址的狀態空間(例如，圖7中的.local、.global或.shared)中。目的地參數<dst>可用各種方式定義以支援不同的定址模式，這與載入指令中的來源參數<src>相似。例如，可使用儲存指令將操作結果從暫存器轉移到可定址的狀態空間。

在提供紋理及表面狀態空間的實施例中，可使用額外的虛擬指令從紋理記憶體狀態空間進行讀取(tex)及從表面記憶體狀態空間進行讀取(suld)並對其進行寫入(sust)。紋理讀取的運算數(t, x, y)指定紋理識別符(t)及座標(x, y)；同樣，表面讀取或寫入的運算數(s, x, y)指定表面識別符(s)及座標(x, y)。

CTA緒可藉由與其它CTA緒共用資料而與其它CTA緒協作。舉例而言，為了在CTA內共用資料，CTA緒可使用載入與儲存虛擬指令(以及下述原子更新指令atom)以將資料寫入到每CTA虛擬狀態空間及從中讀取資料。因此，一個CTA緒可使用具有合適定義的目的地位址的st.shared指令將資料寫入到.shared狀態空間；相同CTA內的另一CTA緒

可隨後藉由在ld.shared指令中使用相同位址來讀取所述資料。下述同步指令(例如，bar及membar)可用來確保CTA線緒間的資料共用操作的正確序列，例如，產生資料的CTA緒在消耗資料的CTA緒讀取資料之前先寫入資料。類似地，st.global及ld.global指令可用於相同CTA中的CTA緒、相同柵格中的CTA及/或相同應用程式中的不同柵格之間的協作及資料共用。

5.3.6. 虛擬指令-程式控制

圖8G是列舉實例虛擬ISA中提供的程式控制操作的表格860。此等控制操作是熟習此項技術者所熟悉的，且其使得程式設計人員可將程式執行重定向。分支(bra)將程式流重定向到目標位置<target>。在一些實施例中，藉由將字母標籤放置在虛擬ISA代碼中的目標指令之前並使用所述標籤作為分支指令的目標識別符<target>來定義分支目標。舉例而言，在一個實施例中：

label: add.int32 d, vr1, vr2;

將add指令識別為具有標籤label的分支目標。在代碼中的其它位置的指令：

bra label;

將執行重定向到帶有標籤的指令。

Call與返回(ret)指令支援函數及子例行程式調用；fname識別函數或子例行程式。(在一個實施例中，"子例行程式"就是其返回值被忽略的函數。)可使用.func指示(directive)來宣稱函數fname，且也提供定義所述函數的虛擬ISA代

碼。波形括號{}或其它分組符號可用來將定義函數或子例行程式的代碼與其它虛擬ISA代碼隔開。

對於函數來說，可指定參數列表<rv>來識別應當將返回值儲存在何處。對於函數及子例行程式來說，在引數列表<args>中指定輸入引數。當執行call時，儲存下一指令的位址；當執行ret時，進入到所儲存位址的分支。

exit指令終止遭遇其的CTA緒。陷阱指令調用處理器定義的或使用者定義的陷阱例行程式。斷點(brkpt)指令使執行暫停，且可用於(例如)調試用途。無操作(nop)是在執行時沒有影響的指令。例如，其可用來控制可在多久以後執行下一操作。

5.3.7. 虛擬指令-平行緒

圖8H是列舉在根據本發明實施例的實例虛擬ISA中提供的明確平行的虛擬指令的表格870。此等指令支援CTA執行所需要的協作緒行為，例如在CTA緒之間交換資料。

屏障(bar)指令指示：到達其的CTA緒應當在執行任何進一步的指令之前等待，直到其它所有CTA緒(相同CTA中的)也已經到達同一屏障指令時為止。可在CTA程式中使用任何數目的屏障指令。在一個實施例中，屏障指令不需要任何參數(不論使用多少個屏障)，因為必須在所有CTA緒均到達第n個屏障之後，任何緒才可前進到第(n＋1)個屏障，依此類推。

在其它實施例中，可例如藉由指定應當在特定屏障處等待的CTA緒(或特定CTA緒的識別符)的數目來使屏障指令

參數化。

另外其它實施例提供"等待"及"不等待"兩種屏障指令。在等待屏障指令下，CTA緒一直等到其它相關CTA緒也已經到達屏障為止；在不等待指令處，CTA緒指示其已到達，但可在其它CTA緒到達之前繼續。在給定屏障處，一些CTA緒可能等待而其它CTA緒不等待。

在一些實施例中，bar虛擬指令可用來同步CTA緒，所述CTA緒正在使用共用記憶體狀態空間進行協作或共用資料。舉例而言，假設一組CTA緒(其可包含CTA的一些或全部緒)各自在每線緒變數(例如，.fp32虛擬暫存器變數myData)中產生一些資料，然後讀取集合中的另一CTA緒產生的資料。指令序列：

st.shared.fp32 myWriteAddress, myData;

bar;

ld.shared.fp32 myData, myReadAddress;

提供所需的行為，其中myWriteAddress及myReadAddress是對應於.shared狀態空間中的位址的每線緒變數。在每個CTA緒將其產生的資料寫入到共用記憶體之後，其一直等到所有CTA緒已經儲存其資料為止，然後繼續從共用記憶體中讀取資料(其可能已經由不同的CTA緒寫入)。

記憶體屏障(membar)指令指示每個CTA緒應當等待其先前請求的記憶體操作(或至少所有寫入操作)完成。這個指令保證在membar指令之後發生的記憶體存取將看到在其之前的任何寫入操作的結果。membar指令在一個實施例中

使用可選的狀態空間名稱<space>以將其範圍限制於瞄準指定狀態空間的記憶體操作，所述指定狀態空間應當是記憶體狀態空間(例如，不是.reg或.sreg狀態空間)。如果未指定任何狀態空間名稱，那麼CTA緒等待瞄準所有記憶體狀態空間的所有待決操作完成。

原子更新(atom)指令致使對由引用<ref>識別的共用變數a的原子更新(讀取-修改-寫入)。所述共用變數a可處於任何共用狀態空間中，且與其它記憶體引用一樣，可使用各種定址模式。舉例而言，<ref>可為以下中的任一者：命名的可定址變數a、對保存變數a的位址的暫存器的引用、對保存將添加到偏移值(提供為立即操作數)以定位變數a的位址的暫存器的引用，或變數a的立即絕對位址。CTA緒將變數a從共用狀態空間位置載入到目的地暫存器d中，接著使用對運算數a及(取決於操作)第二及第三運算數b、c執行的指定操作<op>來更新變數a，並將結果儲存回由<ref>識別的位置。目的地暫存器d保持a的原先載入的值。原子地執行載入、更新及儲存操作，保證沒有其它任何CTA緒在第一CTA緒執行原子更新時存取變數a。在一個實施例中，變數a限於.global或.shared狀態空間，且可用與上述載入及儲存操作一樣的方式來指定。

在一些實施例中，只有特定的操作可作為原子更新來執行。舉例而言，在一個實施例中，如果a是浮點類型那麼只可指定以下操作<op>：將a與b相加；用a及b的最小值或最大值來替換a；以及三元比較與交換操作，其在a等於b

的情況下用c替換a，且否則使a保持不變。對於整數a，可支援額外操作，例如運算數a與b之間的按位的and、or及xor，以及使運算數a遞增或遞減。也可支援其它原子操作或操作組合。

vote指令在預定義組的CTA線緒間對布林(例如，.bl類型)運算數a執行歸約運算<op>。在一個實施例中，虛擬架構指定CTA緒在SIMD群組中執行，且預定義的群組對應於SIMD群組；在其它實施例中，其它群組的CTA緒可由虛擬架構或程式設計人員來定義。歸約運算<op>要求基於在群組中的CTA線緒間對運算數a的歸約以及.<op>限定符指定的歸約運算將結果值d設置成布林真或布林假。在一個實施例中，所允許的歸約運算是：(1).all，其中如果a對於群組中的所有CTA緒為真則d為真，且否則為假；(2).any，其中如果a對於群組中的任何CTA緒為真則d為真；以及(3).uni，其中如果a對於群組中的所有活動的CTA緒具有相同值(真或假)則d為真。

5.3.8. 虛擬指令-斷言執行

在一些實施例中，虛擬ISA支援任何指令的斷言執行。在斷言執行中，將布林"保護斷言(guard predicate)"值與指令相關聯，且指令只有當在執行的時候保護斷言評估為真時才執行。

在實例虛擬ISA中，保護斷言可為任何1位布林虛擬暫存器變數(本文中表示為P)。藉由將斷言保護@P或不斷言保護@!P放置在指令的操作碼前面來指示斷言執行。例如藉

由將P識別為產生了布林結果的指令(例如，表820(圖8C)中的setp指令)的目的地暫存器來在斷言暫存器中建立一個值。在遇到@P或@!P保護斷言時，虛擬處理器讀取P暫存器。對於@P保護，如果P為真，則執行指令，否則將其跳過；對於@!P保護，如果P為假則執行指令，否則跳過。在遇到斷言指令的每個CTA緒的執行時間評估斷言P；因此，一些CTA緒可能執行斷言指令而其它CTA緒不執行。

在一些實施例中，可將斷言設置為指令執行。舉例而言，表800-870(圖8A-8H)中的特定虛擬指令可能接受指定斷言暫存器作為輸出的參數；此種指令基於指令結果的某種性質來更新指定的斷言暫存器。舉例而言，斷言暫存器可用來指示算術運算的結果是不是特殊數字(例如，零、無窮大或IEEE 754浮點運算中的非數字)等等。

6. 虛擬指令翻譯器。

如參看圖4所述，虛擬指令翻譯器412瞄準特定的平台架構。虛擬指令翻譯器412可實施為(例如)在例如圖1的CPU 102等的處理器上執行的軟體程式，所述虛擬指令翻譯器412接收虛擬ISA代碼410並將其轉譯成目標ISA代碼414，所述目標ISA代碼414可在虛擬指令翻譯器412瞄準的特定平台架構上執行(例如，藉由圖1的PPU 122)。虛擬指令翻譯器412將在虛擬ISA代碼410中宣稱的虛擬變數映射到可用的儲存位置上，其中包含處理器暫存器、晶片上記憶體、晶片外記憶體等。在一些實施例中，虛擬指令翻譯器412將每個虛擬狀態空間映射到特定類型的儲存設備上。

舉例而言，可將.reg狀態空間映射到緒特定的資料暫存器上，將.shared狀態空間映射到處理器的可共用記憶體上，將.global狀態空間映射到分配給應用程式的虛擬記憶體區域，等等。其它映射也是可能的。

將虛擬ISA代碼410中的虛擬指令轉譯成機器指令。在一個實施例中，虛擬指令翻譯器412經組態以將每個虛擬ISA指令映射成相應的機器指令或機器指令序列，這取決於在將執行CTA緒的處理器的指令集中是否存在相應的機器指令。

虛擬指令翻譯器412也將CTA緒映射到目標平台架構中的"實體"緒或處理上。舉例而言，如果目標平台架構支援至少n_0個併發緒，那麼每個CTA緒可映射到一個實體緒上，且虛擬指令翻譯器412可為單個CTA緒產生虛擬指令代碼，並預期目標平台440將為具有n_0個唯一識別符的n_0個緒執行代碼。如果目標平台架構支援少於n_0個緒，那麼虛擬指令翻譯器412可產生虛擬ISA代碼410，所述虛擬ISA代碼410併入有對應於多個CTA緒的指令，並預期這個代碼將對每個CTA執行一次，因而將多個CTA緒映射到單個實體緒或處理。

更特定而言，將與資料共用(例如，存取.shared或.global狀態空間的載入、儲存及原子更新指令)及/或協作緒行為(例如，圖8H中的屏障、原子更新及其它指令)有關的虛擬指令轉譯成機器指令或機器指令序列。針對CTA執行而優化的目標平台架構有利地包含硬體支援的屏障指

令，例如其中用指令單元中的計數器及/或暫存器來計數已經到達屏障指令的緒的數目，並設置旗標以防止在緒在屏障處等待時發布緒的進一步的指令。其它目標結構可能不提供對緒同步的直接硬體支援，在此情況下可使用其它線緒間通信技術(例如，信號量、記憶體中的狀態陣列等)來創建所需的行為。

也將斷言指令轉譯成機器指令。在一些實例中，目標硬體直接支援斷言執行。在其它實例中，可將斷言與有條件的分支指令等一起儲存在(例如)處理器暫存器中，所述有條件的分支指令等用來詢問暫存器並藉由有條件地圍繞斷言指令產生分支而創建所需的執行時間行為。

圖9是使用根據本發明實施例的虛擬指令翻譯器的過程900的流程圖。在步驟902處，程式設計人員用高階語言編寫CTA程式代碼。在一個實施例中，所述CTA程式代碼定義單個CTA緒的所需行為，且可使用緒ID(包含CTA ID及/或柵格ID)作為參數來定義或控制CTA緒的行為的各個態樣。舉例而言，可將要讀取或寫入的共用記憶體位置確定為緒ID的函數，使得相同CTA中的不同CTA緒將對共用記憶體中的不同記憶體位置進行讀取及/或寫入。在一個實施例中，包含CTA程式代碼作為應用程式代碼(例如，圖4的程式代碼402)的一部分。除了定義CTA緒行為之外，應用程式代碼還可定義CTA及/或柵格、設置輸入及輸出資料集等。

在步驟904處，編譯器(例如，圖4的編譯器408)根據高

階語言代碼產生定義單個(虛擬)CTA緒的行為的虛擬ISA代碼。如果代碼包含CTA程式代碼及其它代碼兩者，那麼編譯器408可將CTA程式代碼與其餘代碼分開，以便只使用CTA程式代碼來產生虛擬ISA代碼。可使用習知的用於將用一種語言編寫的程式代碼編譯成另一(虛擬)語言的技術。應注意，由於所產生的代碼是使用虛擬語言，所以編譯器不需要束縛於特定硬體或針對特定硬體而優化。編譯器可優化根據特定的輸入代碼序列而產生的虛擬ISA代碼(例如，更偏好虛擬ISA指令的較短序列)。可將虛擬ISA中的程式代碼儲存在碟片上的記憶體中及/或分布到各種各樣的平台架構，其中包含在實體上不同於圖3的虛擬架構300的結構。虛擬ISA中的代碼是獨立於機器的，並且可在任何有虛擬指令翻譯器可用的目標平台上執行。在替代實施例中，程式設計人員可用虛擬ISA直接編寫CTA程式代碼，或者虛擬ISA代碼可由程式自動產生；如果程式代碼起初創建為虛擬ISA代碼，那麼可省略編譯步驟904。

在步驟906處，虛擬指令翻譯器(例如，圖4的翻譯器412)讀取虛擬ISA代碼，並以目標ISA產生可在目標平台上執行的代碼。與編譯器不同的是，虛擬指令翻譯器瞄準特定的(真實)平台架構且有利地經組態以調適並優化目標ISA代碼以便在結構上實現最好的性能。在目標結構支援至少n_0個緒的一個實施例中，虛擬指令翻譯器產生目標緒程式，目標緒程式可由n_0個緒中的每一者併發執行以實現CTA。在另一實施例中，虛擬指令翻譯器產生目標程式，

所述目標程式使用軟體技術(例如，指令序列)來仿真n_0個併發緒，線緒各自執行對應於虛擬ISA代碼的指令。翻譯器可在程式安裝時、在程式初始化期間或在程式執行期間在及時的基礎上操作。

在步驟908處，目標平台中的處理器(例如，圖1的PPU 122)執行目標ISA代碼以處理資料。在一些實施例中，步驟908可包含將命令及狀態參數提供給處理器，以便控制其行為，如下文進一步描述的。

將明白，過程900是說明性的，且可進行更改及修改。描述為循序的步驟可平行執行，步驟次序可更改，且步驟可被修改或組合。舉例而言，在一些實施例中，程式設計人員可直接使用虛擬ISA來編寫CTA程式代碼，從而無需產生虛擬ISA代碼的編譯器。在其它實施例中，將CTA程式代碼編寫為較大的應用程式的一部分，且所述CTA程式代碼還包含(例如)定義要執行以解決特定問題的CTA及/或CTA柵格的維度的代碼。在一個實施例中，只將代碼中那些闡述CTA程式的部分編譯到虛擬ISA代碼中；可將其它部分編譯到其它(真實或虛擬)指令集中。

在其它實施例中，一個虛擬指令翻譯器可經組態以產生適合於不同目標平台的目標代碼的多個版本。舉例而言，翻譯器可產生高階語言(例如，C)的程式代碼、用於PPU的機器代碼及/或用於使用軟體技術來仿真PPU行為的單核或多核CPU的機器代碼。

7. 虛擬執行驅動器

在一些實施例中，使用虛擬ISA代碼410及虛擬指令翻譯器412來產生要對CTA的每個緒執行的CTA程式代碼。就圖2A-2B的程式設計模型來說，指定CTA程式會為每個CTA緒204定義處理任務。為了完成所述模型，也有必要定義CTA 202的維度、柵格中的CTA的數目、要處理的輸入資料集等。此種信息在本文中稱為"CTA控制信息"。

如圖4所示，在一些實施例中，應用程式402藉由使用對虛擬庫404中的函數的調用來指定CTA控制信息。在一個實施例中，虛擬庫404包含各種函數調用，程式設計人員可經由函數調用來定義CTA或CTA柵格並指示何時應當開始執行。

圖10是列舉實例虛擬庫404中可用的函數的表格1000。第一群組的函數涉及定義CTA。具體來說，initCTA函數是第一個要調用以創建新CTA的函數。這個函數使得程式設計人員可定義CTA的維度(ntid.x, ntid.y, ntid.z)，並向新CTA分派識別符cname。SetCTAProgram函數指定要由CTA cname的每個緒執行的CTA程式；參數pname是對應於所需的CTA程式(例如，用虛擬ISA代碼編寫的程式)的邏輯程式識別符。SetCTAInputArray函數使得程式設計人員可指定全局記憶體中CTA cname將從中讀取輸入資料的源位置(開始位址及大小)，且setCTAOutputArray函數使得程式設計人員可指定全局記憶體中CTA cname將向其寫入輸出資料的目標位置(開始位址及大小)。使用setCTAParams函數來設置CTA cname的執行時間常量參數。程式設計人員向函

數提供參數列表，例如作為(名稱，值)對。

在一個實施例中，setCTAParams函數也可由編譯器408在產生虛擬ISA代碼410時使用。由於setCTAParams函數定義CTA的執行時間參數，所以編譯器408可將這個函數解譯為將每個參數定義為.param狀態空間中的虛擬變數。

表格1000還列舉與定義CTA的柵格有關的函數。InitGrid函數是第一個調用來創建新柵格的函數。這個函數使得程式設計人員可定義柵格的維度(nctaid.x, nctaid.y, nctaid.z)，識別將在柵格上執行的CTA cname，並將識別符gname分派給新定義的柵格。setGridInputArray及setGridOutputArray函數與CTA級函數相似，允許針對柵格中的所有CTA的所有緒定義單個輸入及/或輸出陣列。setGridParams函數用來為柵格gname中的所有CTA設置執行時間常量參數。編譯器408可將這個函數解譯為將每個參數定義為.const狀態空間中的虛擬變數。

launchCTA及launchGrid函數指示應開始指定CTA cname或柵格gname的執行。

虛擬API也可包含其它函數。舉例而言，一些實施例提供可用來協調多個CTA的執行的同步函數。舉例而言，如果第一CTA(或柵格)的輸出將用作第二CTA(或柵格)的輸入，那麼API可包含一個函數(或啟動函數的參數)，可經由所述函數來指示虛擬執行驅動器：在第一CTA(或柵格)的執行完成之前不應啟動第二CTA(或柵格)。

根據本發明的實施例，表格1000中的任何或所有函數調

用均可包含在應用程式中，所述應用程式也定義要執行的CTA程式(或者如果在應用程式中存在多個CTA則要定義多個CTA程式)。在編譯時，將函數調用視為對應用程式介面(API)庫404的調用，因而產生虛擬API代碼406。

使用實施虛擬庫中的每個函數的虛擬執行驅動器418來實現虛擬API代碼。在一個實施例中，虛擬執行驅動器418是在圖1的CPU 102上執行的驅動器程式，驅動器程式控制實現CTA緒的PPU 122。實施圖10的表格1000中的各種函數調用，使得其導致驅動器經由推送緩衝器(pushbuffer)向PPU 122提供命令。在另一實施例中，CPU執行一或多個程式以實現CTA，且虛擬執行驅動器418設置參數並控制CPU對此種程式的執行。

將瞭解，本文中描述的虛擬API是說明性的，且變更及修改是可能的。可支援其它函數或函數組合。此項技術中已知的虛擬API技術可適合於本發明的用途。

進一步的實施例

雖然已經參照特定實施例描述了本發明，但熟習此項技術者將認識到許多修改是可能的。舉例而言，不需要本文中描述的特定虛擬架構、虛擬指令及虛擬API函數；可用其它支援併發、協作緒的虛擬架構、指令及/或函數來代替。此外，上述實施例可引用所有區塊具有相同數目的元件、所有CTA具有相同數目的緒且執行相同CTA程式等的情況。在一些應用程式(例如，其中使用多個依賴性柵格的情況)中，可能需要使不同柵格中的CTA執行不同CTA程

式或具有不同數目及/或大小的柵格。

雖然本文中引用"協作緒陣列"，但應瞭解，一些實施例可使用其中不支援併發緒之間的資料共用的緒陣列；在支援此種資料共用的其它實施例中，針對給定應用程式定義的緒可能或可能不在實際上共用資料。

此外，雖然上述實施例可將緒陣列稱為具有多個緒，但應瞭解，在"退化"的情況下，緒陣列可能只具有一個緒。因此，本發明可應用於在要在具有一或多個單線緒或多線緒核心的CPU上執行的程式中提供可擴展性。藉由使用本文中描述的技術，可用可在任何數目的可用CPU核心上分布緒(例如，使用操作系統功能性)而無需對虛擬ISA代碼的修改或重新編譯的方式編寫程式。

本文中使用術語"虛擬"及"真實"來反映程式設計人員用來描述問題解決方案的概念性程式設計模型與可在上面最終執行程式的實際電腦系統的脫離。"虛擬"程式設計模型及其相關聯的結構使得程式設計人員可對平行的處理任務採用高階視角，且應瞭解，可能存在或可能不存在其組件一一對應地映射到本文中描述的虛擬架構組件的實際計算系統或裝置。有利地將包含虛擬ISA代碼及虛擬API代碼的虛擬代碼實現為用一種語言編寫的可能或可能不與任何實際處理裝置的指令集一一對應的代碼。與所有程式代碼一樣，本文中所指的虛擬代碼可儲存在有形的媒體(例如，記憶體或碟片)中、藉由網路傳輸等。

併入有本發明的各種特徵的電腦程式--包含但不限於虛

擬ISA及/或虛擬API代碼、虛擬指令翻譯器、虛擬驅動器、編譯器、虛擬函數庫等--可編碼在各種電腦可讀媒體上以供儲存及/或傳輸；合適的媒體包含磁碟或磁帶、光學儲存媒體，例如光碟(CD)或DVD(多功能數位光碟)、快閃記憶體等。此種程式也可使用適合經由符合各種協定的有線、光學及/或無線網路(包含網際網路)傳輸的載波信號。用程式代碼編碼的電腦可讀儲存媒體可用相容裝置封裝，或者程式代碼可與其它裝置獨立地提供(例如，經由網際網路下載)。

此外，本文中可將特定動作描述為由"程式設計人員"作出。預期程式設計人員可以是人類、藉由很少或沒有人為介入來產生程式代碼的自動過程、或人類交互與自動或半自動處理的用以產生程式代碼的任何組合。

此外，雖然本文中描述的實施例可能引用了特定目標平台的特徵，但本發明不限於此等平台。實際上，虛擬架構可用硬體及/或軟體組件的任何組合來實現。熟習此項技術者將明白，可預期相同虛擬架構的不同實現在效率及/或輸出量態樣不同；然而，此種差異與本發明無關。

因此，雖然已經參考特定實施例描述了本發明，但應明白，本發明並不意圖涵蓋屬於隨附申請專利範圍內的所有修改及等效物。

【圖式簡單說明】

圖1是根據本發明實施例的電腦系統的方塊圖。

圖2A及2B說明本發明實施例中使用的程式設計模型中

的柵格、緒陣列及緒之間的關係。

圖3是根據本發明實施例的虛擬架構的方塊圖。

圖4是根據本發明實施例使用虛擬架構來操作目標處理器的概念模型。

圖5是列舉由根據本發明實施例的虛擬指令集架構(ISA)定義的特殊變數的表格。

圖6是列舉在根據本發明實施例的虛擬ISA中支援的變數類型的表格。

圖7是列舉在根據本發明實施例的虛擬ISA中支援的虛擬狀態空間的表格。

圖8A-8H是列舉在根據本發明實施例的虛擬ISA中定義的虛擬指令的表格。

圖9是根據本發明實施例的用於使用虛擬指令翻譯器的過程的流程圖。

圖10是列舉根據本發明實施例的在虛擬庫中可供虛擬執行驅動器使用的函數的表格。

【主要元件符號說明】

100 電腦系統

102 CPU

104 系統記憶體

105 記憶體橋

106 通信路徑

107 I/O橋

108 使用者輸入裝置

110	顯示器裝置
112	平行處理子系統
113	路徑
114	系統碟片
116	切換器
118	網路配接器
120	內插卡
121	內插卡
122	平行處理單元
124	平行處理(PP)記憶體
202	CTA
204	緒
300	虛擬架構
302	虛擬處理器
304	全局記憶體
306	前端
308	虛擬核心
310	處理引擎
311	特殊暫存器
312	指令單元
314	區域暫存器
316	共用記憶體
318	參數記憶體
320	虛擬驅動器

322	記憶體介面
400	概念模型
402	應用程式
404	虛擬API函數庫
406	虛擬API代碼
408	編譯器
410	虛擬ISA代碼
412	虛擬指令翻譯器
414	目標ISA代碼
416	虛擬執行驅動器
418	目標API命令
430	指令單元
432	PPU驅動器
434	前端
440	平台
500	表格
600	表格
700	表格
800	表格
810	表格
820	表格
830	表格
840	表格
850	表格

860 表格

870 表格

900 過程

1000 表格

五、中文發明摘要：

一種虛擬架構及指令集支援明確的平行緒計算。該虛擬架構定義：虛擬處理器，其支援具有不同虛擬緒之間的多個級別的資料共用及協調(例如，同步)的多個虛擬緒的併發執行；以及虛擬執行驅動器，其控制該虛擬處理器。該虛擬處理器的虛擬指令集架構用來定義虛擬緒的行為，且包含與平行緒行為(例如，資料共用及同步)有關的指令。藉由使用該虛擬平台，程式設計人員能夠開發出其中虛擬緒併發地執行以處理資料的應用程式；虛擬翻譯器及驅動器以對於程式設計人員顯明的方式來調適應用程式代碼於執行該應用程式代碼的特定硬體。

六、英文發明摘要：

A virtual architecture and instruction set support explicit parallel-thread computing. The virtual architecture defines a virtual processor that supports concurrent execution of multiple virtual threads with multiple levels of data sharing and coordination (e.g., synchronization) between different virtual threads, as well as a virtual execution driver that controls the virtual processor. A virtual instruction set architecture for the virtual processor is used to define behavior of a virtual thread and includes instructions related to parallel thread behavior, e.g., data sharing and synchronization. Using the virtual platform, programmers can develop application programs in which virtual threads execute concurrently to process data; virtual translators and drivers adapt the application code to particular hardware on which it is to execute, transparently to the programmer.

十、申請專利範圍：

1. 一種用於定義一平行處理操作的方法，該方法包括：

提供第一程式代碼，該第一程式代碼定義待針對協作的虛擬緒的一陣列中的複數個虛擬緒中的每一者執行的操作序列；

將該第一程式代碼編譯成一虛擬緒程式，該虛擬緒程式定義待針對該複數個虛擬緒中的一代表性虛擬緒執行的每線緒指令的一序列，該每線緒指令之序列包含至少一個指令，該指令定義該代表性虛擬緒與該複數個虛擬緒中的一或多個其它虛擬緒之間的協作行為；以及

儲存該虛擬緒程式。

2. 如請求項1的方法，其進一步包括：

將該儲存的虛擬緒程式轉譯成符合一目標平台架構的一指令序列。

3. 如請求項1的方法，其進一步包括：

提供第二程式代碼，該第二程式代碼定義協作的虛擬緒的一陣列，該陣列經調適以處理一輸入資料集以產生一輸出資料集，其中該陣列中的每個虛擬緒併發地執行該虛擬緒程式；

將該第二程式代碼轉換成一虛擬函數庫中的一函數調用序列，該庫包含初始化並致使執行協作的虛擬緒的該陣列的虛擬函數；以及

儲存該函數調用序列。

4. 如請求項3的方法，其進一步包括：

將該儲存的虛擬緒程式及該函數調用序列轉譯成可在一目標平台架構上執行的程式代碼，該可執行的程式代碼定義執行該協作的虛擬緒陣列的一或多個平台線緒。

5. 如請求項4的方法，其進一步包括：

在符合該目標平台架構的一電腦系統上執行該可執行的程式代碼，因而產生該輸出資料集；以及

將該輸出資料集儲存在一儲存媒體中。

6. 如請求項1的方法，其中該每線緒指令序列包含用以在該序列中的一特定點處暫停對該代表性虛擬緒的操作的執行，直到該等其它虛擬緒中的一或多者到達該特定點時為止的指令。

7. 如請求項1的方法，其中該每線緒指令序列包含一用以使該代表性虛擬緒將資料儲存在該等其它虛擬緒中的一或多者可存取的一共用記憶體中的指令。

8. 如請求項1的方法，其中該每線緒指令序列包含一用以使該代表性虛擬緒自動讀取及更新儲存在該等其它虛擬緒中的一或多者可存取的一共用記憶體中的資料的指令。

9. 如請求項1的方法，其中該虛擬緒程式包含一變數定義語句，該變數定義語句定義複數個虛擬狀態空間之一者中的一變數，其中該複數個虛擬狀態空間中的不同狀態空間對應於該等虛擬緒之間的資料共用的不同模式。

10. 如請求項9的方法，其中該等資料共用模式包含一每線緒不共用模式及一全局共用模式。

11. 如請求項9的方法，其中該等資料共用模式包含一每線

緒不共用模式、虛擬緒的一個陣列內的一共用模式，以及一全局共用模式。

12. 如請求項9的方法，其中該等資料共用模式包含一每線緒不共用模式、虛擬緒的一個陣列內的一共用模式、虛擬緒的複數個陣列之間的一共用模式，以及一全局共用模式。

13. 一種用於操作一目標處理器的方法，該方法包括：

提供輸入程式代碼，其包含一第一部分，該第一部分定義待對虛擬緒的一陣列中的複數個虛擬緒中的每一者執行的經調適以處理一輸入資料集以產生一輸出資料集的一操作序列，

該輸入程式代碼進一步包含一第二部分，該第二部分定義該虛擬緒陣列的一維度；

將該輸入程式代碼的該第一部分編譯成一虛擬緒程式，該虛擬緒程式定義待針對該複數個虛擬緒中的一代表性虛擬緒執行的一每線緒指令序列，該每線緒指令序列包含至少一個指令，該指令定義該代表性虛擬緒與該複數個虛擬緒中的一或多個其它虛擬緒之間的一協作行為；

將該輸入程式代碼的該第二部分轉換成對一虛擬函數庫的一函數調用序列，該庫包含虛擬函數，該等虛擬函數初始化及致使執行該協作的虛擬緒陣列；

將該虛擬緒程式及該函數調用序列轉譯成可在一目標平台架構上執行的程式代碼，該可執行的程式代碼定義

執行該協作的虛擬緒陣列的一或多個真實緒；

在符合該目標平台架構的一電腦系統上執行該可執行的程式代碼，因而產生該輸出資料集；以及

將該輸出資料集儲存在一儲存媒體中。

14. 如請求項13的方法，其中該輸入程式代碼的該第二部分包含定義該虛擬緒陣列的兩個或兩個以上維度的程式代碼。

15. 如請求項14的方法，其中該輸入程式代碼的該第二部分進一步包含：

一函數調用，其定義虛擬緒陣列的一柵格的一或多個維度，其中該柵格中的每一陣列均將被執行。

16. 如請求項13的方法，其中該目標平台架構包含一主處理器及一共處理器，且其中該翻譯的動作包含：

將該虛擬緒程式轉譯成可由該共處理器上定義的複數個緒平行執行的程式代碼；以及

將該函數調用序列轉譯成對該共處理器的一驅動器程式的調用序列，其中該驅動器程式在該主處理器上執行。

17. 如請求項13的方法，其中該目標平台架構包含一中央處理單元(CPU)，且其中該轉譯的動作包含：

將該虛擬緒程式及該函數調用序列的至少一部分轉譯成目標程式代碼，該目標程式代碼使用少於該虛擬緒數目的一數目之CPU線緒執行該虛擬緒陣列。

18. 一種用於操作一目標處理器的方法，該方法包括：

七、指定代表圖：

(一)本案指定代表圖為：第（4）圖。

(二)本代表圖之元件符號簡單說明：

122	平行處理單元
400	概念模型
402	應用程式
404	虛擬API函數庫
406	虛擬API代碼
408	編譯器
410	虛擬ISA代碼
412	虛擬指令翻譯器
414	目標ISA代碼
416	虛擬執行驅動器
418	目標API命令
430	指令單元
432	PPU驅動器
434	前端
440	平台

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

(無)

獲得虛擬緒程式，該虛擬緒程式定義待對一虛擬緒陣列中的複數個虛擬緒中的一代表性虛擬緒執行的每線緒指令的一序列，該陣列經調適以處理一輸入資料集以產生一輸出資料集；

該每線緒指令序列包含至少一個指令，該至少一個指令定義該代表性虛擬緒與該複數個虛擬緒中的一或多個其它虛擬緒之間的一協作行為；

獲得定義該虛擬緒陣列的維度的額外程式代碼；

將該虛擬緒程式及該額外程式代碼轉譯成可在一目標平台架構上執行的程式代碼，該可執行的程式代碼定義執行該虛擬緒陣列的一或多個平台線緒；及

在符合該目標平台架構的一電腦系統上執行該可執行的程式代碼，因而產生該輸出資料集並將該輸出資料集儲存在一記憶體中。

19. 如請求項18的方法，其中該獲得該虛擬緒程式的動作包含：

接收用一高階程式設計語言編寫的源程式代碼；以及

編譯該源程式代碼以產生該虛擬緒程式。

20. 如請求項18的方法，其中該獲得該虛擬緒程式的動作包含：

從一儲存媒體中讀取該虛擬緒程式。

21. 如請求項18的方法，其中該獲得該虛擬緒程式的動作包含：

經由一網路從一遠端電腦系統接收該虛擬緒程式。

100年12月12日修(更)正替換頁

十一、圖式：

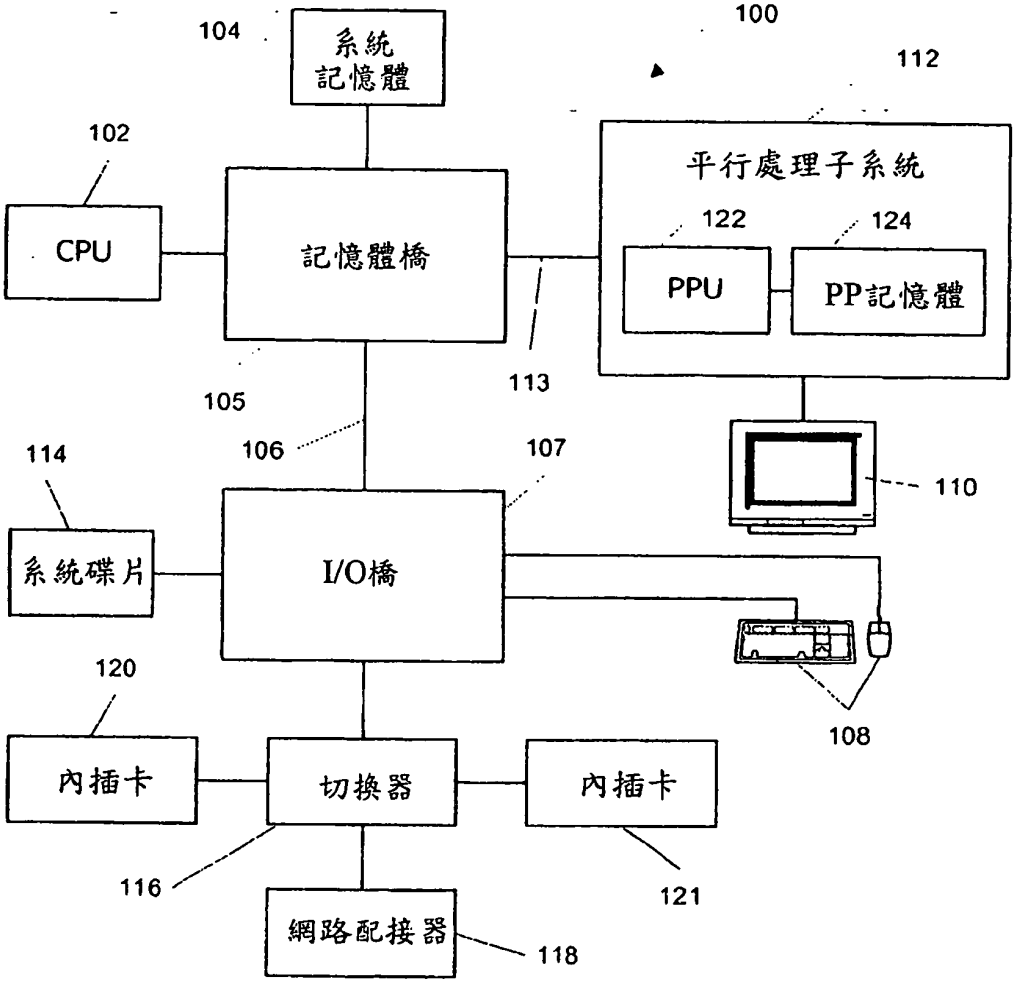

圖1

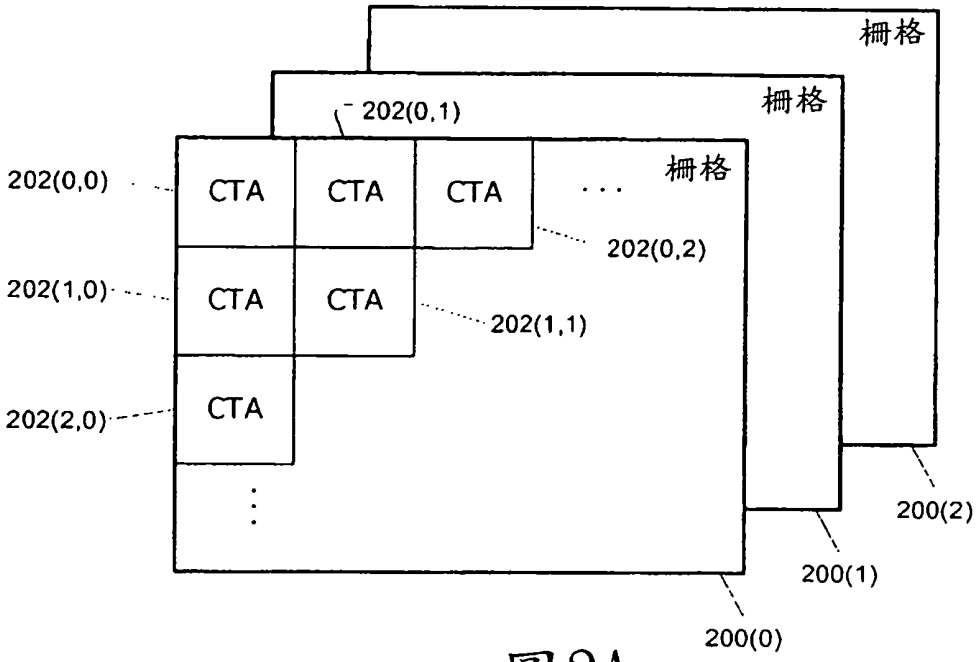
柵格
柵格
柵格
202(0,1)
202(0,0)
CTA
CTA
CTA
202(0,2)
202(1,0)
CTA
CTA
202(1,1)
202(2,0)
CTA
200(2)
200(1)
200(0)

圖2A

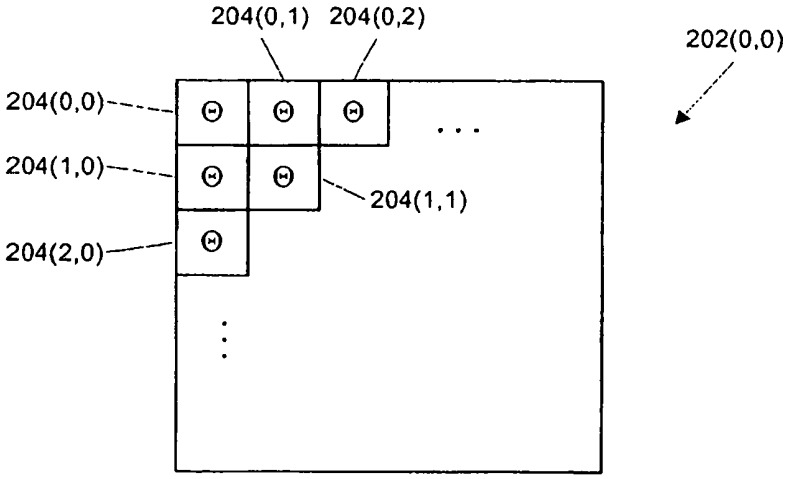
204(0,1)
204(0,2)
202(0,0)
204(0,0)
204(1,0)
204(2,0)
204(1,1)

圖2B

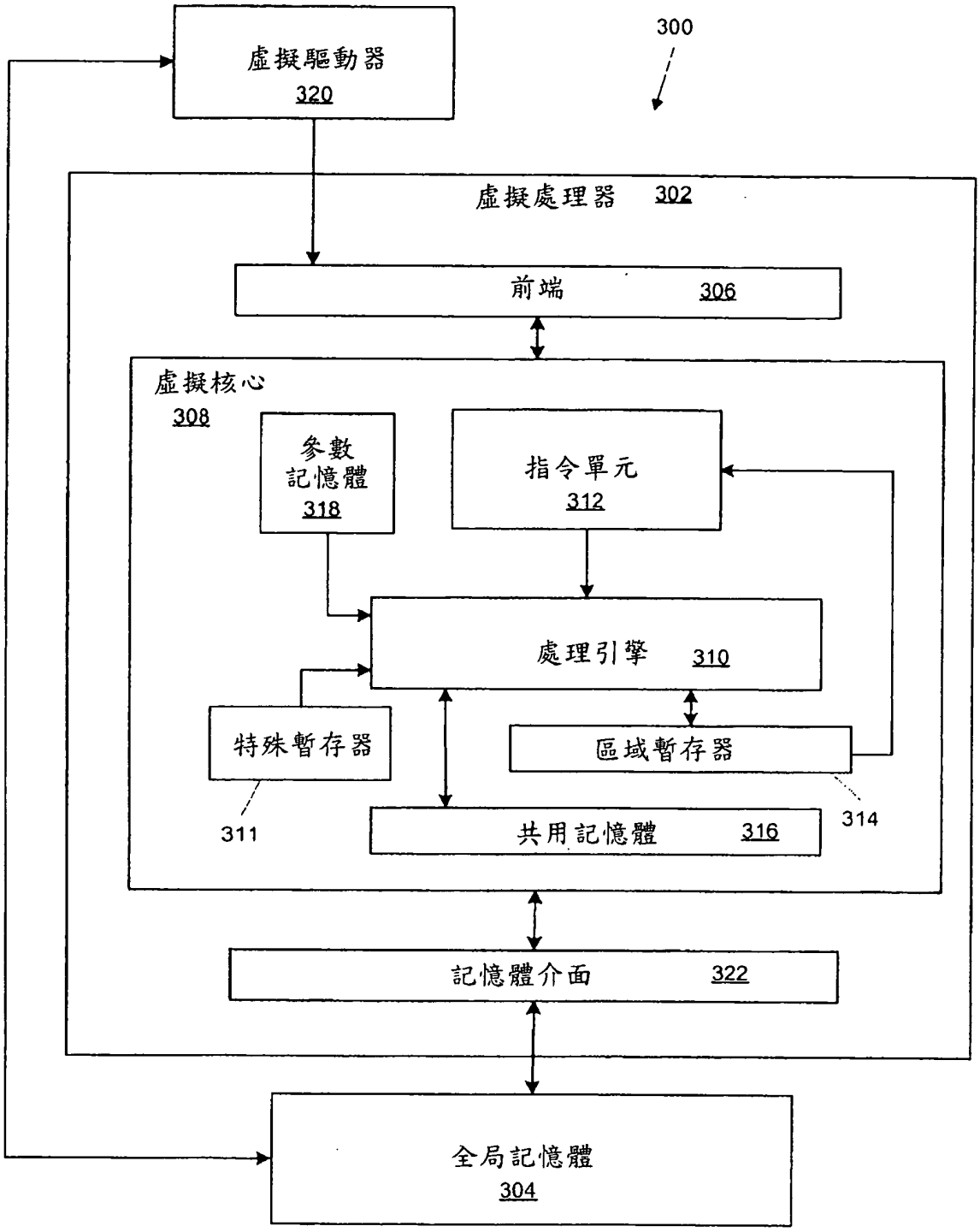
300
虛擬驅動器
320
虛擬處理器 302
前端 306
虛擬核心
308
參數
記憶體
318
指令單元
312
處理引擎 310
特殊暫存器
311
區域暫存器
314
共用記憶體 316
記憶體介面 322
全局記憶體
304

圖3

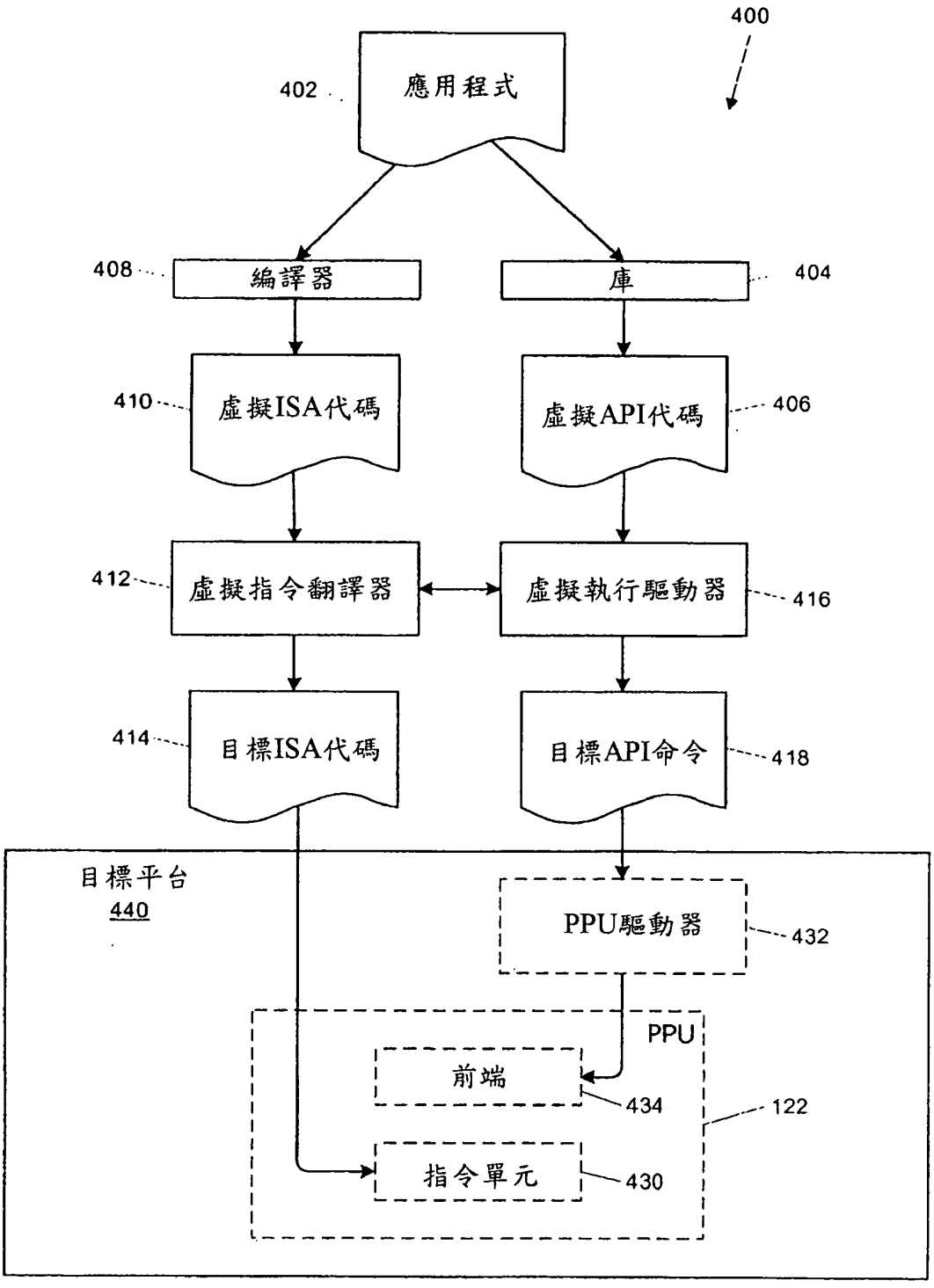
400
402
應用程式
408
編譯器
庫
404
410
虛擬ISA代碼
虛擬API代碼
406
412
虛擬指令翻譯器
虛擬執行驅動器
416
414
目標ISA代碼
目標API命令
418
目標平台
440
PPU驅動器
432
PPU
前端
434
122
指令單元
430

圖4

500

名稱	說明
%ntid.x, %ntid.y, %ntid.z	CTA維度
%tid.x, %tid.y, %tid.z	CTA內的緒ID
%nctaid.x, %nctaid.y, %nctaid.z	柵格維度
%ctaid.x, %ctaid.y, %ctaid.z	柵格內的CTA ID
%gridid	柵格ID

圖5

600

類型	說明	允許的 <n>
.b<n>	未指定類型變數	1, 8, 16, 32, 64
.s<n>	有正負號的整數	8, 16, 32, 64
.u<n>	無正負號整數	8, 16, 32, 64
.f<n>	浮點數	16, 32, 64

圖6

700

名稱	共用	說明	映射到
.reg	緒	運算數暫存器	區域暫存器
.sreg	緒	特殊暫存器（見圖5）	區域暫存器
.local	緒	每線緒不共用記憶體	全局記憶體
.shared	CTA	共用記憶體、讀取/寫入存取	共用記憶體
.param	CTA	可進行唯讀存取的共用值	參數或共用記憶體
.const	柵格	可進行唯讀存取的共用值	參數記憶體
.global	內容	在CTA間共用的值	全局記憶體
.tex	內容	在CTA間共用的紋理	全局記憶體
.surf	內容	在CTA間共用的表面	全局記憶體

圖7

800

指令	效果
add.<type> d, a, b	d = a+b
sub.<type> d, a, b	d = a-b
mul.<type> d, a, b	d = a*b
div.<type> d, a, b	d = a/b
mad.<type> d, a, b, c	d = [a*b] + c
fma.<type> d, a, b, c	d = a*b + c
sad.<type> d, a, b, c	d = |a-b| + c
rem.<type> d, a, b	d = a mod b
abs.<type> d, a	d = |a|
neg.<type> d, a	d = -a
min.<type> d, a, b	d = min(a,b)
max.<type> d, a, b	d = max(a,b)
frc.<type> d, a	d = a - floor(a)
sin.<type> d, a	d = sin a
cos.<type> d, a	d = cos a
atan2.<type> d, a, b	d = $\tan^{-1}$ (a/b)
lg2.<type> d, a	d = $\log_2$ a
ex2.<type> d, a	d = 2^a
rcp.<type> d, a	d = 1/a
sqrt.<type> d, a	d = sqrt(a)
rsqrt.<type> d, a	d = 1/sqrt(a)

圖8A

810

指令	效果
dot.<type> d, **a**, **b**	d = sum(a[i]*b[i])
cross.<type> **d**, **a**, **b**	**d** = **a** X **b**
mag.<type> d, a	d = sqrt(sum(a[i]*a[i]))

vred.<op>.<type> d, a	d = a[0];
	對於i＝1到長度{
	d=<op>(d, a[i])
	}

圖8B

820

指令	效果
sel.<type> d, a, b, c	d = c ? a : b
set.<cmp>.<type> d, a, b	t = a <cmp> b; d = t ? ~0 : 0
setb.<cmp>.<bop>.<type> d, a, b, c	t = a <cmp> b; d = (t <bop> c) ? ~0 : 0
setp.<cmp>.<bop>.<type> d1 | d2, a, b, c	t = a <cmp> b; d1 = (t <bop> c) d2 = (!t <bop> c)

圖8C

830

指令	效果
and.<type> d, a, b	d = a & b
or.<type> d, a, b	d = a | b
xor.<type> d, a, b	d = a ^ b
not.<type> d, a	d = ~a
cnot.<type> d, a	d = !a
shl.<type> d, a, b	d = a << b
shr.<type> d, a, b	d = a >> b

圖8D

840

指令	效果
cvt.<dtype>.<atype> d, a	d = (dtype) a
cvt.<mode>.<dtype>.<atype> d, a	d=(dtype) a, 每<mode>捨入

圖8E

指令	效果
mov.<type> d, a	d = a
ld.<space>.<type> d, <src>	向暫存器d中載入藉由<src>識別的資料
st.<space>.<type> <dst>, a	將暫存器a中的資料儲存到藉由<dst>識別的位置
tex d, [t, x, y]	d = 紋理 (t,x,y)
suld d, [s, x, y]	d = 表面 (s,x,y)
sust [s, x, y], a	表面 (s,x,y) = a

圖8F

850

指令	效果
bra <target>	去往 <target>
call <rv> fname <args>	函數/子例行程式調用
ret	從函數/子例行程式返回
exit	終止緒
trap	調用處理器陷阱例行程式
brkpt	暫停執行
nop	無操作

860

圖8G

指令	效果
bar	在屏障處等待CTA的其它所有緒
Membar <space>	等待記憶體寫入完成
atom.<space>.<op>.<type> d, <ref>, b, c	以原子方式讀取、更新及寫回共用記憶體變數
vote.<op> d, a	基於在線緒間a是真還是假，為群組中的每個緒設定d

870

圖8H

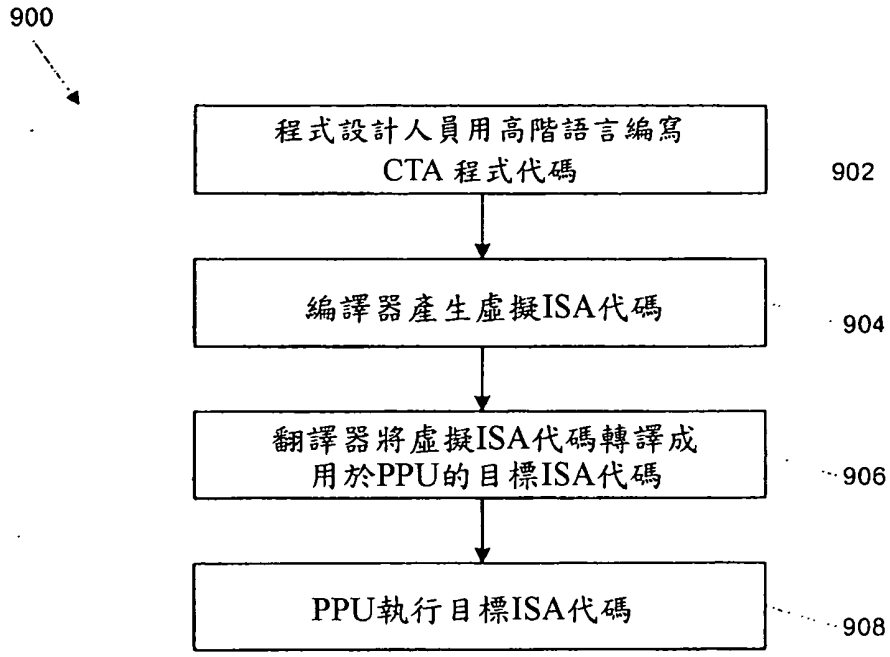

圖9

1000

函數	參數
initCTA	cname, ntid.x, ntid.y, ntid.z
setCTAProgram	cname, pname
setCTAInputArray	cname, baseAddr, size
setCTAOutputArray	cname, baseAddr, size
setCTAParams	cname, np,<list>
initGrid	gname, cname, nctaid.x, nctaid.y, nctaid.z
setGridInputArray	gname, baseAddr, size
setGridOutputArray	gname, baseAddr, size
setGridParams	gname, np,<list>
launchCTA	cname
launchGrid	gname

圖10