(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6077018号
(P6077018)
(45)発行日　平成29年2月8日(2017.2.8)　(24)登録日　平成29年1月20日(2017.1.20)

(51)Int.Cl.　　F I
G06F 9/45 (2006.01)　　G06F 9/44 322E
G06F 9/38 (2006.01)　　G06F 9/38 370C

請求項の数 20　(全 33 頁)

(21)出願番号　特願2014-558964(P2014-558964)
(86)(22)出願日　平成25年2月27日(2013.2.27)
(65)公表番号　特表2015-513737(P2015-513737A)
(43)公表日　平成27年5月14日(2015.5.14)
(86)国際出願番号　PCT/US2013/028029
(87)国際公開番号　WO2013/130614
(87)国際公開日　平成25年9月6日(2013.9.6)
審査請求日　平成28年2月4日(2016.2.4)
(31)優先権主張番号　61/603,771
(32)優先日　平成24年2月27日(2012.2.27)
(33)優先権主張国　米国(US)
(31)優先権主張番号　13/777,663
(32)優先日　平成25年2月26日(2013.2.26)
(33)優先権主張国　米国(US)

早期審査対象出願

(73)特許権者　595020643
クゥアルコム・インコーポレイテッド
ＱＵＡＬＣＯＭＭ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100109830
弁理士　福原　淑弘
(74)代理人　100103034
弁理士　野河　信久
(74)代理人　100075672
弁理士　峰　隆司

最終頁に続く

(54)【発明の名称】異種ＣＰＵ－ＧＰＵ計算のための実行モデル

(57)【特許請求の範囲】
【請求項１】
異種計算のための方法であって、
プロセッサを用いて、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信することと、
前記プロセッサを用いて、前記実行モデルの前記パイプライントポロジーがグラフィックス処理ユニット（ＧＰＵ）に実装されるプラットフォームに依存した方法を指示する命令を生成することであって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存する方法は、前記ＧＰＵのプラットフォームに基づき、前記パイプライントポロジーは、消費するカーネルによって消費されるデータを生成する、生成するカーネルを識別し、前記命令は、拡大係数に基づき、前記生成するカーネルによって生成される最大のデータ量は、前記生成するカーネルが受信するデータ量および前記拡大係数に基づく、生成することと、
前記プロセッサを用いて、前記ＧＰＵに前記命令を送信することと、
を備える、方法。
【請求項２】
命令を生成することは、
前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルすることを備える請求項１に記載の方法。
【請求項３】

前記コマンドリストをコンパイルすることは、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存する方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルすることを備える請求項２に記載の方法。
【請求項４】
前記ＧＰＵの前記構成情報は、
前記ＧＰＵ内のプログラマブル計算ユニット数、及び
前記ＧＰＵ内のデータレーン数
のうちの１つ以上を備える請求項３に記載の方法。
【請求項５】
前記コマンドリストをコンパイルすることは、
前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び
前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性
のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルすることを備える請求項２に記載の方法。
【請求項６】
前記パイプライントポロジーを受信することは、
前記パイプライントポロジーを形成するために相互に接続された１つ以上のカーネル及び１つ以上のバッファを示すコマンドリストを受信することを備える請求項１に記載の方法。
【請求項７】
装置であって、
グラフィックス処理ユニット（ＧＰＵ）と、
プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーのインディケーションを受信することと、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成することであって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づき、前記パイプライントポロジーは、消費するカーネルによって消費されるデータを生成する、生成するカーネルを識別し、前記命令は、拡大係数に基づき、前記生成するカーネルによって生成される最大のデータ量は、前記生成するカーネルが受信するデータ量および前記拡大係数に基づく、生成することと、
前記ＧＰＵに前記命令を送信することと
を行うように構成されたプロセッサと、
を備える、装置。
【請求項８】
前記命令を生成するために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルように構成される請求項７に記載の装置。
【請求項９】
前記コマンドリストをコンパイルするために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルするように構成される請求項８に記載の装置。
【請求項１０】
前記ＧＰＵの前記構成情報は、
前記ＧＰＵ内のプログラマブル計算ユニット数、及び

前記ＧＰＵ内のデータレーン数
のうちの１つ以上を備える請求項９に記載の装置。
【請求項１１】
前記コマンドリストをコンパイルするために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び
前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性
のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルするように構成される請求項８に記載の装置。
【請求項１２】
前記パイプライントポロジーを受信するために、前記プロセッサは、
前記パイプライントポロジーを形成するために相互に接続された１つ以上のカーネル及び１つ以上のバッファを示すコマンドリストを受信するように構成される請求項７に記載の装置。
【請求項１３】
前記装置は、
メディアプレーヤー、
セットトップボックス、
無線ハンドセット、
デスクトップコンピュータ
ラップトップコンピュータ
ゲームコンソール
ビデオ会議装置、及び
タブレットコンピューティングデバイス、
のうちの１つを備える請求項７に記載の装置。
【請求項１４】
非一時的なコンピュータによって読み取り可能な記憶媒体であって、
１つ以上のプロセッサによって実行されたときに、
プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信することと、
前記実行モデルの前記パイプライントポロジーがグラフィックス処理ユニット（ＧＰＵ）に実装されるプラットフォームに依存した方法を指示する命令を生成することであって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づき、前記パイプライントポロジーは、消費するカーネルによって消費されるデータを生成する、生成するカーネルを識別し、前記命令は、拡大係数に基づき、前記生成するカーネルによって生成される最大のデータ量は、前記生成するカーネルが受信するデータ量および前記拡大係数に基づく、生成することと、
前記ＧＰＵに前記命令を送信することと
を前記１つ以上のプロセッサに行わせる命令を格納した、非一時的なコンピュータによって読み取り可能な記憶媒体。
【請求項１５】
命令を生成することを前記１つ以上のプロセッサに行わせる前記命令は、
前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備える請求項１４に記載の非一時的なコンピュータによって読み取り可能な記憶媒体。
【請求項１６】
前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる前記命令は、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラット

フォームに依存した方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備える請求項１５に記載の非一時的なコンピュータによって読み取り可能な記憶媒体。

【請求項１７】

前記ＧＰＵの前記構成情報は、

前記ＧＰＵ内のプログラマブル計算ユニット数、及び

前記ＧＰＵ内のデータレーン数

のうちの１つ以上を備える請求項１６に記載の非一時的なコンピュータによって読み取り可能な記憶媒体。

【請求項１８】

前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる前記命令は、

前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び

前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性

のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備える請求項１５に記載の非一時的なコンピュータによって読み取り可能な記憶媒体。

【請求項１９】

装置であって、

グラフィックス処理ユニット（ＧＰＵ）と、

プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信するための手段と、

前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成するための手段であって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵで実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づき、前記パイプライントポロジーは、消費するカーネルによって消費されるデータを生成する、生成するカーネルを識別し、前記命令は、拡大係数に基づき、前記生成するカーネルによって生成される最大のデータ量は、前記生成するカーネルが受信するデータ量および前記拡大係数に基づく、生成するための手段と、

前記ＧＰＵに前記命令を送信するための手段と、

を備えるプロセッサと、

を備える、装置。

【請求項２０】

命令を前記生成するための手段は、

前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルするための手段を備える請求項１９に記載の装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本出願は、ここにおける引用によってその内容全体が組み入れられている米国仮特許出願第６１／６０３，７７１号（出願日：２０１２年２月２７日）の利益を主張するものである。

【０００２】

本開示は、アプリケーションの実行に関するものである。本開示は、より具体的には、様々な処理ユニットにおけるアプリケーションの実行に関するものである。

【背景技術】

【０００３】

グラフィックス処理ユニット（ＧＰＵ）は、グラフィックス処理に加えての目的のために使用されている。例えば、グラフィックスに関連しないアプリケーションは、ＧＰＵの大規模な並列性を利用することによって上昇した速度で実行することができる。この結果、追加のグラフィックスに関連しない処理機能を提供し、汎用ＧＰＵ（ＧＰＧＰＵ）と呼ばれるＧＰＵが得られている。例えば、ＧＰＧＰＵは、１つ以上のシェーダコア（ｓｈａｄｅｒ　ｃｏｒｅ）を含み、シェーダコアは、グラフィックスに関連しないアプリケーションと同様に、グラフィックス関連のアプリケーションも実行するように構成される。

【発明の概要】

【０００４】

概して、本開示は、計算パイプラインを実装するための実行モデルを生成するための技法に関するものである。例えば、多くのデータ処理アルゴリズムは、計算パイプラインとして表すことができ、パイプラインの１つのユニットがデータを受信及び処理し、さらなる処理のためにパイプラインの他のユニットのために処理されたデータを出力する。本開示において説明される技法は、様々なタイプの並列コンピューティングデバイスにおいて効率的に実行することができるような形で計算パイプラインを表すことを考慮することができ、その一例がグラフィックス処理ユニット（ＧＰＵ）である。例えば、それらの技法は、プラットフォームから独立した方法（platform-independent manner）で（例えば、計算パイプラインを実装することになるコンピューティングデバイスに依存しないで）計算パイプラインを表すことができる。

【０００５】

計算パイプラインを表すことで、それらの技法は、利用可能なコンピューティングデバイス、例えば、利用可能なＧＰＵ、さらには中央処理装置（ＣＰＵ）、専用の命令を生成するためにプラットフォームに依存するコンパイルを利用することができる。例えば、計算パイプラインは、プラットフォームから独立した方法で定義することができ、コンパイラは、計算パイプラインが実装されるべきプラットフォーム専用の命令を生成する。

【０００６】

一例では、本開示は、異種計算（heterogeneous computing）のための方法について説明する。その方法は、プロセッサを用いて、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信することを含む。その方法は、プロセッサを用いて、実行モデルのパイプライントポロジーがグラフィックス処理ユニット（ＧＰＵ）に実装されるプラットフォームに依存した方法を指示する命令を生成することも含む。この例では、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法は、ＧＰＵのプラットフォームに基づく。その方法は、プロセッサを用いて、ＧＰＵに命令を送信することも含む。

【０００７】

一例では、本開示は、装置について説明する。その装置は、グラフィックス処理ユニット（ＧＰＵ）と、プロセッサと、を含む。プロセッサは、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーのインディケーション（ｉｎｄｉｃａｔｉｏｎ）を受信するように構成される。プロセッサは、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成するようにも構成される。この例では、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法は、ＧＰＵのプラットフォームに基づく。プロセッサは、ＧＰＵに命令を送信するようにも構成される。

【０００８】

一例では、本開示は、１つ以上のプロセッサによって実行されたときに、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信することを１つ以上のプロセッサに行わせる命令が格納されているコンピュータによって読み取り可能な記憶媒体について説明する。命令は、実行モデルのパイプライントポロジーがグラフィックス処理ユニット（ＧＰＵ）に実装されるプラットフォームに依存した方法を指示する命令を生成することも１つ以上のプロセッサに行わせる。こ

の例では、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法は、ＧＰＵのプラットフォームに基づく。命令は、ＧＰＵに命令を送信することも１つ以上のプロセッサに行わせる。

【０００９】

一例では、本開示は、装置について説明する。その装置は、グラフィックス処理ユニット（ＧＰＵ）と、プロセッサと、を含む。プロセッサは、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信するための手段を含む。プロセッサは、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成するための手段も含む。この例では、実行モデルのパイプライントポロジーがＧＰＵに実装されるプラットフォームに依存した方法は、ＧＰＵのプラットフォームに基づく。プロセッサは、ＧＰＵに命令を送信するための手段も含む。

【００１０】

１つ以上の例の詳細が添付された図面及び以下の説明において示される。これらの説明及び図面から、及び請求項からその他の特徴、目的、及び利点が明確になるであろう。

【図面の簡単な説明】

【００１１】

【図１】実行モデルの例を示した概念図である。

【図２】本開示において説明される１つ以上の例によるデバイスの例を示したブロック図である。

【図３】本開示において説明される１つ以上の例による技法例を示したフローチャートである。

【図４】図２のデバイスをより詳細に例示したブロック図である。

【発明を実施するための形態】

【００１２】

グラフィックス処理ユニット（ＧＰＵ）は、データを並列で素早くかつ効率的に処理するように構成することができる。開発者は、ＧＰＵ上で実行するアプリケーションの形でデータ処理アルゴリズムを開発することができる。例えば、ＧＰＵは、１つ以上のアプリケーションを実行するように構成されるシェーダプロセッサを含むことができる。これらのアプリケーションの例は、シェーダプログラム、例えば、バーテックスシェーダ（ｖｅｒｔｅｘ　ｓｈａｄｅｒ）（頂点シェーダ）、ハルシェーダ（ｈｕｌｌ　ｓｈａｄｅｒ）、フラグメントシェーダ（ｆｒａｇｍｅｎｔ　ｓｈａｄｅｒ）、幾何シェーダ（ｇｅｏｍｅｔｒｙ　ｓｈａｄｅｒ）、及びグラフィックス処理に関連するその他の該アプリケーション、を含む。

【００１３】

さらに、幾つかのアプリケーション開発者は、ＧＰＵの大規模な並列性を利用し、グラフィックスに関連しないアプリケーションをＧＰＵで実行するのが有益であるとみなすであろう。例えば、ＧＰＵによって提供される処理の並列性は、並列行列演算がグラフィックス処理に関連していないときでさえも、それらの行列演算を実行するのに適することができる。グラフィックスに関連しないアプリケーションのその他の例は、並列演算の素早い実行が有益であることができる流体力学又は線形代数に関連する技法を含む。

【００１４】

該グラフィックスに関連しないアプリケーションを実行することが可能なＧＰＵは、汎用ＧＰＵ（ＧＰＧＰＵ）であるとみなすことができる。例えば、ＧＰＵがグラフィックスに関連しないアプリケーションを実行中であるときには、ＧＰＵは、ＧＰＧＰＵとして機能している。ほとんどのＧＰＵは、ＧＰＧＰＵとして機能するように構成することができる。

【００１５】

例示を目的として、本開示は、ＧＰＧＰＵとして機能しているＧＰＵに関して技法を説明する。しかしながら、それらの技法は、ＧＰＵがＧＰＧＰＵとして機能している（すな

わち、グラフィックスに関連しないアプリケーションを実行している）事例には限定されず、それらの技法は、ＧＰＵがグラフィックス関連のアプリケーションを実行している事例にも適用することができる。さらに、本開示において説明される技法は、あらゆるタイプの処理ユニット、例えば、中央処理装置（ＣＰＵ）、アクセラレータ、又はその他のカスタムデバイスによって実装することができる。それらの技法は、ＧＰＵに関して説明されが、それらの技法は、その他のタイプの処理ユニットにも拡張可能であることが理解されるべきである。

【００１６】

ＧＰＵ内のシェーダプロセッサは、複数のシェーダコア（これらのコアはグラフィックス関連及びグラフィックスに関連しない、の両方のアプリケーションに関する命令を実行できることを示すためにプログラマブル計算ユニットとも呼ばれる）を含むことができる。プログラマブル計算ユニットの各々は、そのプログラマブル計算ユニットによって実行される命令に関して予約されているローカルメモリ、及びそれらの命令の実行によって生成されたデータ、例えば、命令の実行中に生成される即座の結果、を含むことができる。プログラマブル計算ユニットのローカルメモリは、その他のプログラマブル計算ユニットによってはアクセス不能であることができる。幾つかの例では、ＧＰＵで実行されるべき異なるアプリケーションは、異なるプログラマブル計算ユニットによって実行することができる。

【００１７】

本開示において説明される技法では、グラフィックス関連のアプリケーションはシェーダと呼ばれ、グラフィックスに関連しないアプリケーションは、カーネルと呼ばれる。例えば、シェーダ（すなわち、グラフィックス関連のアプリケーション）の例は、バーテックスシェーダと、フラグメントシェーダと、幾何シェーダと、含み、ただしこれらに限定されない。カーネル（すなわち、グラフィックスに関連しないアプリケーション）の例は、行列演算、流体力学、画像処理動作、映像処理動作、等を行うためのアプリケーションを含む。

【００１８】

さらに、カーネルは、ＧＰＵによって実行されるアプリケーションのみに必ずしも限定する必要はなく、ＧＰＵの固定機能ユニット（ｆｉｘｅｄ－ｆｕｎｃｔｉｏｎ　ｕｎｉｔ）（すなわち、プログラミング不能なユニット）も含む。例示のみを目的として、本開示において説明される技法は、ＧＰＵで実行されるアプリケーションであるカーネルに関して説明される。例えば、それらの技法は、ＧＰＵがＧＰＧＰＵとして機能するためにＧＰＵのシェーダプロセッサで実行するグラフィックスに関連しないアプリケーションに関して説明される。

【００１９】

カーネルは、複数のワークグループ、タスク、又はスレッドを含むことができる（これらはすべて、本開示では同義語として用いられる）。例えば、スレッドは、カーネルのその他のスレッドから独立して実行することができるカーネルの命令の組であることができる。幾つかの例では、カーネルを実行するためには、プログラマブル計算ユニットのうちの１つ以上がカーネルの１つ以上のスレッドを各々実行することができる。例えば、第１のプログラマブル計算ユニットは、カーネルの第１のスレッドを実行することができ、第２のプログラマブル計算ユニットは、同じカーネルの第２のスレッドを実行することができる。幾つかの例では、１つのプログラマブル計算ユニットが１つのカーネルの１つ以上のスレッドを実行することができ、他のプログラマブル計算ユニットは、他のカーネルの１つ以上のスレッドを実行する。幾つかの例では、２つの組み合わせが可能である（すなわち、幾つかのプログラマブル計算ユニットが同じカーネルの異なるスレッドを実行中であり、他方、幾つかのその他のプログラマブル計算ユニットが異なるカーネルのスレッドを実行中である）。

【００２０】

ＧＰＵは、大規模な並列性を処理のために提供する一方で、開発者、例えば、カーネル

の開発者、は、様々なタイプのＧＰＵにおいてパイプライン方式で効率的に実行するカーネルを開発するのは困難であるとみなすであろう。パイプライン方式でカーネルを実行することは、１つのカーネルによって生成されたデータが他のカーネルによって消費されるような形でカーネルを実行することを意味する。他の例として、パイプライン方式でカーネルを実行することは、同じカーネルの他のスレッドによって消費されるべきデータを生成するカーネルのスレッドを実行することを意味する。この開示では、データを生成するスレッドは、生成するスレッド（ｐｒｏｄｕｃｅｒ　ｔｈｒｅａｄ）と呼ぶことができ、データを受信するスレッドは、消費するスレッド（ｃｏｍｓｕｍｅｒ　ｔｈｒｅａｄ）と呼ぶことができる。

【００２１】

幾つかの例では、生成するスレッド及び消費するスレッドは、同じカーネルのスレッドであることができる。幾つかの例では、生成するスレッド及び消費するスレッドは、異なるカーネルのスレッドであることができる。これらの例では、生成するスレッドを含むカーネルは生成するカーネルと呼ぶことができ、消費するスレッドを含むカーネルは消費するカーネルと呼ぶことができる。

【００２２】

一例として、データ処理アルゴリズム、例えば、画像処理又は映像処理、を実装するために、開発者は、複数のカーネルを開発することができ、各カーネルは、全体的アルゴリズムの一部分を実装する。第１のカーネルは、第２のカーネルによる消費のために処理されるべきデータ（例えば、グラフィックスに関連しないデータ）を受信し、データを処理し、及びデータを出力することができる。第２のカーネルは、第３のカーネルによる消費のために第１のカーネルによって出力されたデータを受信し、データをさらに処理し、データを出力し、以下同様である。

【００２３】

この例では、第１、第２、及び第３のカーネルがパイプラインを形成すると考えることができ、第１のカーネル（例えば、生成するカーネル）は、第２のカーネル（例えば、第１のカーネルの観点からの消費するカーネル）によって消費されるべきデータを生成する。第２のカーネルは、第３のカーネルによって消費されるべきデータを生成する。この例では、第２のカーネルは、第３のカーネルの観点からの生成するカーネルであり、第３のカーネルは、消費するカーネルである。この方法により、ＧＰＵは、パイプライン方式で第１、第２、及び第３のカーネルを実行することができる。

【００２４】

幾つかの例では、パイプライン方式でカーネルを実行することは、カーネルを逐次に実行する（例えば、第１のカーネルを実行し、第２のカーネルを実行することによって後続され、第３のカーネルを実行することによって後続され、以下同様である）ことを意味することができる。しかしながら、本開示において説明される技法は、そのようには限定されない。幾つかの例では、パイプライン方式でカーネルを実行することは、並列して（同時に又は時間が重なる状態で）カーネルを実行することを意味することができる。例えば、ＧＰＵは、第２のカーネルが第１のカーネルに関する消費するカーネルであり、第３のカーネルが第２のカーネルに関する消費するカーネルである場合でも第１、第２、及び第３のカーネルのうちの２つ以上を同時に実行することができる。

【００２５】

開発者は、データ処理アルゴリズムを実装するためにパイプライン方式で実行するカーネルを開発することができるが、開発者は、様々なタイプのＧＰＵにまたがってカーネルの最適な実行を保証することはできないであろう。例えば、開発者は、プロセッサで実行する命令を書くことができる。これらの命令は、いつカーネルを実行すべきかをＧＰＵに命令することをプロセッサに行わせることができる。上述されるように、カーネルは、１つ以上の計算ユニットで実行することができる。しかしながら、開発者は、特定のＣＰＵＩで利用可能な計算ユニットの数、より一般的には、特定のＧＰＵの並列処理能力、を知らないことがある。

【００２６】
この場合は、開発者は、ＧＰＵの処理能力を開発者が知らないため、いつカーネルを実行すべきかを予め決定することができないことがある。この結果、開発者は、各々が異なるタイプのＧＰＵ専用の異なる命令を書くことになる。例えば、開発者は、第１のＧＰＵタイプ専用の、プロセッサで実行する命令の第１の組を書くことができる。例えば、第１のＧＰＵタイプが３つの計算ユニットを含む場合は、命令の第１の組は、３つの計算ユニットを有するＧＰＵでカーネルを実行することができる方法を定義することができる。開発者は、第２のＧＰＵタイプ専用の、プロセッサで実行する、命令の第２の組も書くことができる。例えば、第２のＧＰＵタイプが４つの計算ユニットを含む場合は、命令の第２の組は、４つの計算ユニットを有するＧＰＵでカーネルを実行することができる方法を定義することができる。

【００２７】
幾つかの例では、異なるＧＰＵタイプに関する命令を書くのではなく、開発者は、１つのみのタイプのＧＰＵ（例えば、推定される最悪の事態のシナリオでのＧＰＵ）に関する命令を書くことができる。これらの例では、１つのタイプのみのＧＰＵがデータ処理アルゴリズムを効率的に実装することができ、その他のＧＰＵタイプは、データ処理アルゴリズムを効率的に実装することができない。

【００２８】
換言すると、開発者がＧＰＵで効率的な方法で実行することをカーネルに行わせる命令を書くことができるプラットフォームから独立した方法は存在することができない。むしろ、開発者は、その他のＧＰＵタイプで非効率的に実行する（例えば、ＧＰＵタイプに依存しない）一般的命令を書くことができる。他方、開発者は、結果的には非ポータブルな（ｎｏｎ－ｐｏｒｔａｂｌｅ）命令になるプラットフォームに依存する命令を書くことができる。例えば、開発者は、異なるＧＰＵタイプの各々に関して別々の命令を書かなければならないことがあり、それは過度に厄介であろう。

【００２９】
本開示において説明される技法は、プラットフォームから独立した方法で（すなわち、異種の計算のために）データ処理アルゴリズムを実装するためにカーネルを効率的に実行することを考慮する。本開示において説明される技法では、異種計算は、プラットフォームから独立した方法での計算を意味する。より詳細に説明されるように、本開示において説明される技法により、開発者は、データ処理アルゴリズムを実装するためのカーネルに関するパイプライン実行モデルを指定する。

【００３０】
パイプライン実行モデルを指定するために、開発者は、パイプラインのトポロジーを定義することができる。パイプラインのトポロジーは、相互に接続されたカーネル及びバッファを含む実行グラフであるとみなすことができる。例えば、第１のカーネルが第２のカーネルによって消費されるべきデータを生成する場合。開発者は、第１のカーネルがバッファ、例えば、先入れ先出し（ＦＩＦＯ）バッファ、に結合され、バッファが第２のカーネルに結合されるような形で、トポロジーを定義することができる。この例では、実行グラフは、第１のカーネルがバッファにデータを出力し、第２のカーネルがバッファからデータを受信することを示すことができる。

【００３１】
トポロジーを定義することに加えて、開発者は、実行モデルの一部としてトポロジーの特徴を定義することもできる。一例として、開発者は、各カーネルに関する拡大係数（ａｍｐｌｉｆｉｃａｔｉｏｎ　ｆａｃｔｏｒ）を定義することができる。拡大係数は、カーネルが受信する所定の量のデータに関してカーネルが生成する最大のデータ量を示すことができる。例えば、拡大係数がカーネルに関して５であり、カーネルが２つのデータパケットを受信する場合は、カーネルが生成する最大のデータ量は、１０のデータパケットである。

【００３２】

他の例として、開発者は、バッファのサイズを定義することができる。例えば、開発者は、バッファの幅（例えば、バッファの記憶場所内に格納することができるデータの量）及びバッファの長さ（例えば、バッファ内の記憶場所の数）を定義することができる。

【００３３】

この方法により、開発者は、データ処理アルゴリズムに関するプラットフォームから独立した実行モデルを定義することができる。例えば、開発者は、データ処理アルゴリズムが実装される特定のＧＰＵを説明する必要がない。むしろ、各ＧＰＵに関する実行モデルは、同じであることができる。

【００３４】

本開示おいて説明される技法は、開発者がバウンドされた方法で（ｂｏｕｎｄｅｄ　ｍａｎｎｅｒ）実行モデルを定義するのを可能にすることができる。例えば、開発者は、いずれのカーネルが必要であるか、いずれのカーネルが生成するカーネルを形成するか及びいずれのカーネルが消費するカーネルを形成するかを完全に定義することができる。バウンドされた方法で実行モデルを定義することは、静的実行モデル（例えば、実行前に定義されるそれ）であるとみなすことができる。

【００３５】

バウンドされた方法で実行モデルを定義することは、バウンドされない方法で（ｕｎｂｏｕｎｅｄｅｄ　ｍａｎｎｅｒ）実行モデルを定義することと比較して計算効率の利得を可能にすることができる。実行モデルのバウンドされない定義では、開発者は、実行前には、必要になるカーネル数、いずれのカーネルが生成するカーネルになるか及びいずれのカーネルが消費するカーネルになるかを定義することができない（すなわち、カーネル間の相互接続を定義することができない）。この結果、バウンドされた実行モデルと比較して、バウンドされない実行モデルの性能が最適でなくなる可能性がある。

【００３６】

例えば、本開示において説明される技法では、プロセッサは、実行モデルを受信することができ、及び実行モデルをコンパイルしてＧＰＵによって処理することができるオブジェクトコード（すなわち、バイナリコード）にすることができる。コンパイルステップは、プラットフォームに依存するステップであることができる。例えば、プロセッサは、データ処理アルゴリズムが実装されるＧＰＵの処理能力を示す情報で予め構成することができる。一例として、プロセッサは、ＧＰＵ内の計算ユニット数を示す情報で予め構成することができる。

【００３７】

コンパイルステップでは、プロセッサは、メタスケジューラ（ｍｅｔａ－ｓｃｈｅｄｕｌｅｒ）に関する命令を生成することができる。メタスケジューラは、ＧＰＵで実行するソフトウェアであることができ又はＧＰＵ内のハードウェアであることができる。メタスケジューラに関する命令は、実行モデルが実行される方法を定義することができる。この例では、実行モデルはバウンド（ｂｏｕｎｄ）することができ（例えば、カーネル数及びカーネルの相互接続が知られている）、プロセッサは、ＧＰＵの処理能力を示す情報で予め構成することができるため、コンパイラは、ＧＰＵが実行モデルを実装する方法を最適化するメタースケジューラに関する命令を定義することができる。バウンドされない実行モデルの場合は、カーネル数及びそれらの各々の相互接続は知ることができず、コンパイラは、ＧＰＵでの実行モデルの実行を適切に最適化することができない。

【００３８】

図１は、実行モデルの例を示した概念図である。例えば、図１は、実行モデル１０を例示する。開発者は、データ処理アルゴリズムを実装するための実行モデル１０を定義することができる。例えば、開発者は、画像処理、映像処理、線形代数演算、又は流体力学を計算するためのアルゴリズムを実装するために実行モデル１０を定義することができる。概して、開発者は、グラフィックス処理ユニット（ＧＰＵ）によって提供される大規模な並列計算効率を利用するデータ処理アルゴリズムを実装するために実行モデル１０を定義することができ、グラフィックスに関連しない目的を含む。ＧＰＵがグラフィックスに関

連しないアルゴリズムを実装中である例では、ＧＰＵは、汎用ＧＰＵ（ＧＰＧＰＵ）のように機能しているとみなすことができる。

【００３９】

例示されるように、実行モデル１０は、バッファ１２Ａ乃至１２Ｄと、カーネル１４Ａ乃至１４Ｃと、を含む。幾つかの例では、図１に示されるバッファ及びカーネルよりも多い又は少ないそれらが存在することができる。バッファ１２Ａ乃至１２Ｄの例は、先入れ先出し（ＦＩＦＯ）バッファとリングバッファと含み、ただしこれらに限定されない。

【００４０】

カーネル１４Ａ乃至１４Ｃの例は、実行モデル１０が実装するように定義される全体的データ処理アルゴリズムの少なくとも一部分を実装する、開発者によって開発されたアプリケーションを含む。開発者は、カーネル１４Ａ乃至１４Ｃを開発するためにあらゆるプログラミング言語を利用することができる。

【００４１】

開発者が実行モデル１０を定義することができる様々な方法が存在することができる。一例として、開発者は、コンピューティングデバイス、例えば、デスクトップコンピュータ又はラップトップコンピュータ、において実行モデルを定義することができる。開発者は、グラフィカルユーザインタフェース（ＧＵＩ）を提示するコンピューティングデバイスでアプリケーションを実行することができる。開発者は、図１において例示される方法でバッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃを相互接続するためにＧＵＩを利用することができる。さらに、開発者は、バッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｄの特徴を定義するためにＧＵＩを利用することができる。

【００４２】

他の例として、開発者は、特定のアプリケーション処理インタフェース（ＡＰＩ）によりコマンドを利用して実行モデルを定義することができる。該ＡＰＩの例は、Ｍｉｃｒｏｓｏｆｔ（登録商標）によるＤｉｒｅｃｔＸ（登録商標）、Ｋｈｒｏｎｏｓグループによる ＯｐｅｎＧＬ（登録商標）、及びＫｈｒｏｎｏｓグループによるＯｐｅｎＣＬ（登録商標）を含む。しかしながら、本開示の態様は、ＤｉｒｅｃｔＸ、ＯｐｅｎＧＬ又はＯｐｅｎＣＬ ＡＰＩには限定されず、開発済みの、現在開発中の、又は将来開発予定のその他のタイプのＡＰＩにまで拡張することができる。さらに、本開示において説明される技法は、ＡＰＩにより機能することは要求されない。

【００４３】

例えば、コマンドは、開発者が実行モデルを定義中であることを示すコマンドを含むことができる。コマンドは、バッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃが実行モデル１０に属することを開発者が定義するのを可能にし、及びバッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃが相互接続される方法を定義するコマンドも含むことができる。

【００４４】

いずれの例でも（すなわち、ＧＵＩに基づく又はコマンドに基づく）、開発者が実行モデル１０を定義したコンピューティングデバイスは、実行モデル１０を変換することができ、実行モデル１０のトポロジーを指定するコマンドリストを含む。例えば、例示されるように、カーネル１４Ａは、バッファ１２Ａからデータを受信し、データを処理し、バッファ１２Ｂ及び１２Ｃにデータを格納する。カーネル１４Ｂは、バッファ１２Ｂからデータを受信し、データを処理し、及びバッファ１２Ｄにデータを格納する。カーネル１４Ｃは、バッファ１２Ｄ及び１２Ｃからデータを受信し、データを処理する。

【００４５】

この方法により、バッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃは、計算パイプラインとして構成される。例えば、カーネル１４Ａは、カーネル１４Ｂ及び１４Ｃに関する生成するカーネルである。カーネル１４Ｂは、カーネル１４Ａに関する消費するカーネル及びカーネル１４Ｃに関する生成するカーネルである。カーネル１４Ｃは、カーネル１４Ａ及び１４Ｂの両方に関する消費するカーネルである。

【００４６】
理解を助けるために、図１は、実行モデル１０のパイプライントポロジーを例示するとみなすことができる。例えば、開発者は、実行モデル１０のパイプライントポロジーを定義する実行グラフを定義するとみなすことができる。この実行グラフでは、カーネル１４Ａ乃至１４Ｃは、バッファ１２Ａ乃至１２Ｄと相互接続されるノードであるとみなすことができる。
【００４７】
幾つかの例では、開発者は、異なる実行モデルを相互接続することもできる。例えば、データ処理アルゴリズムに関して１つの実行モデルを定義する代わりに、開発者は、複数の実行モデルを開発することができ、各実行モデルがデータ処理アルゴリズムの一部分を実装する。これらの例では、各々の実行モデル内のカーネルは、全体的データ処理アルゴリズムの一部分の小部分を実装することができる。開発者は、カーネル１４Ａ乃至１４Ｃ及びバッファ１２乃至１２Ｄを相互接続するのと同様の方法で実行モデルを相互接続することができる。例えば、開発者は、バッファ１２Ａを他の実行モデルに相互接続すること及び／又はカーネル１４Ｃを他の実行モデルに相互接続することができる。
【００４８】
複数の実行モデルを定義するのが有益であることができる。より詳細に説明されるように、プロセッサは、実行モデル、例えば、実行モデル１０、をコンパイルしてオブジェクトコードにし、その結果得られたオブジェクトコードを格納することができる。実行モデル１０が複数の実行モデルのうちの１つである例では、プロセッサは、実行モデル１０を再コンパイルする必要がない。換言すると、実行モデルは、全体的なデータ処理アルゴリズムに関するビルディングブロックであるとみなすことができ又はデータ処理アルゴリズムの全体を定義することができる。これで、共通して使用される実行モデルは、実行モデルが使用されるすべての事例に関して再コンパイルする必要がない。
【００４９】
実行モデル１０のトポロジーを定義することに加えて、開発者は、バッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃの特徴を定義することもできる。開発者は、ＧＵＩ又は上述されるコマンドに基づくフォーマットを用いて特徴を定義することができる。開発者は、バッファ１２Ａ乃至１２Ｄ内の記憶場所の数（すなわち、バッファ１２Ａ乃至１２Ｄの長さ）を定義することができる。開発者は、バッファ１２Ａ乃至１２Ｄの各記憶場所内に格納することができるデータの量（すなわち、バッファ１２Ａ乃至１２Ｄの幅）を定義することもできる。
【００５０】
幾つかの例では、開発者は、バッファ１２Ａ乃至１２Ｄの次元を定義することができる。例えば、幾つかの画像処理技法、例えば、畳み込み、は、ピクセルのブロック（例えば、７×７のピクセルのブロック）において生じる。これらの例では、ピクセルをブロック形態で格納するためにバッファ１２Ａ乃至１２Ｄは二次元バッファであることが有益であることができる。例えば、ピクセルのブロックが７×７のピクセルのブロックである場合は、バッファ１２Ａ乃至１２Ｄのうちの１つ以上は、４９の記憶場所を有する線形バッファではなく、７×７の記憶場所を有する状態で（すなわち、二次元バッファとして）構成することができる。
【００５１】
カーネル１４Ａ乃至１４Ｃに関しては、開発者は、一例として、拡大係数を定義することができる。拡大係数は、カーネルが消費する所定のデータ量に関してそのカーネルが生成する最大のデータ量を示すことができる。例えば、カーネル１４Ｂに関する拡大係数が２であり、カーネル１４Ｂが５つのデータパケットをバッファ１２Ｂから受信する場合は、カーネル１４Ｂが生成する最大のデータ量は１０のデータパケットである。他の例として、カーネル１４Ａ乃至１４Ｃのうちの１つ以上に関して、開発者は、カーネルが（例えば、受信されたデータ量から独立して）生成する最大データ量を定義することもできる。
【００５２】

さらに他の例として、開発者は、相対的重要性をカーネル１４Ａ乃至１４Ｃに割り当てることができる。例えば、重要性は、カーネル１４Ａ乃至１４Ｃのうちのいずれが割り込まれないで実行すべきかを示すことができ、カーネル１４Ａ乃至１４Ｃのうちのより重要なそれらは割り込まれないで実行し、カーネル１４Ａ乃至１４Ｃのうちの重要性がより低いそれらは割り込まれないで又は割り込まれた状態で実行することができる（すなわち、その他の実行のために解放するために実行が断続的に休止される）。

【００５３】

カーネル１４Ａ乃至１４Ｃ及びバッファ１２Ａ乃至１２Ｄの特徴は、例示する目的で説明されており、限定するものであるとはみなされるべきでない。開発者が上述されるすべての特徴例を定義する必要はない。例えば、開発者は、バッファ１２Ａ乃至１２Ｄのサイズ（例えば、長さ及び幅）を定義することができ、カーネル１４Ａ乃至１４Ｃの特徴は定義することができない。これらの例では、カーネル１４Ａ乃至１４Ｃが生成又は消費するデータの量は、バッファ１２Ａ乃至１２Ｄのサイズが既に定義されているため重要ではない。他の例として、開発者は、拡大係数又はカーネル１４Ａ乃至１４Ｃが生成する最大データ量を定義することができ、バッファ１２乃至１２Ｄの特徴は定義しない。これらの例では、実行モデル１０をコンパイルするプロセッサは、拡大係数及び／又はカーネル１４Ａ乃至１４Ｃが生成する最大データ量に基づいてバッファ１２Ａ乃至１２Ｄのサイズを決定することができる。さらに、これらの例では、実行モデル１０をコンパイルするプロセッサは、バッファ１２Ａ乃至１２Ｄが一次元（すなわち、線形）バッファ又は多次元バッファのいずれであるべきかを決定することができる。

【００５４】

概して、開発者又はプロセッサは、バッファ１２Ａ乃至１２Ｄが大きくなりすぎないようにする一方で、“行き詰まった”状況を回避するためにカーネル１４Ａ乃至１４Ｃ及びバッファ１２Ａ乃至１２Ｄの特徴を決定することができる。行き詰まった状況は、消費するカーネルが、データを受信すべきバッファ内に格納されていないデータを期待したとき又は生成するカーネルが、消費するカーネルがデータを消費するよりも高速でデータを格納しているためバッファがデータでオーバーフローするときに発生する可能性がある。行き詰まった状況では、カーネルは、“停滞し”（ｈａｎｇ）、カーネルが行き詰まった状況で停滞しないようにするための追加措置を講じないかぎりデータ処理アルゴリズムの実装を停止する。幾つかの例では、行き詰まりが発生したときに停滞を回避するために追加のタスクを実装するようにＧＰＵを構成するよりも、行き詰まりが発生しないようにカーネル１４Ａ乃至１４Ｃ及びバッファ１２Ａ乃至１２Ｄの特徴を定義したほうが良いであろう。

【００５５】

開発者又はプロセッサが行き詰まりを緩和するために相対的に大きいサイズのバッファ１２Ａ乃至１２Ｄを定義することが可能である。しかしながら、バッファ１２Ａ乃至１２Ｄのサイズが不必要に大きい場合は、プロセッサは、必要なメモリスペースよりもはるかに大きいメモリスペースをバッファ１２Ａ乃至１２Ｄのために予約する可能性があり、その結果、非効率的なメモリの使用になってしまうおそれがある。

【００５６】

従って、バッファ１２Ａ乃至１２Ｄ及び／又はカーネル１４Ａ１４Ｄの特徴を定義することによって、開発者は、メモリが効率的に使用される一方で行き詰まりの機会が低下されるような形で実行モデル１０を定義することができる。この方法により、開発者は、完全にバウンドされた静的な実行モデル１０を定義することができる。例えば、開発者は、実行モデル１０を実装するために必要なカーネル及びバッファの数が動的に（すなわち、実装中に）定義されるのではなく、実装前に、実行モデル１０を実装するために必要なカーネル１４Ａ乃至１４Ｃ及びバッファ１２Ａ乃至１２Ｄの数、及びバッファ１２Ａ乃至１２Ｄ及びカーネル１４Ａ乃至１４Ｃの特徴を定義することができる。

【００５７】

開発者は、データ処理アルゴリズムを実装するＧＰＵを含むデバイスに実行モデル１０

を格納することができる。例えば、開発者は、実行モデル１０のパイプライントポロジーを指定するコマンドリストを、ＧＰＵを含むデバイスに格納することができる。幾つかの例では、開発者が実行モデル１０をデバイスに格納するのではなく、デバイスのユーザがデバイスでの格納のために実行モデル１０をダウンロードすることができる。概して、データ処理アルゴリズムを実装するＧＰＵを含むデバイスが実行モデル１０を格納する方法は、本開示で説明される技法の制約にはならない。換言すると、データ処理アルゴリズムが実装されるべきＧＰＵを含むデバイスに実行モデル１０（例えば、実行モデル１０のコマンドのリスト）を格納するためにあらゆる技法を利用することができる。例えば、開発者が実行モデル１０を定義したコンピューティングデバイスがデータ処理アルゴリズムを実装すべきＧＰＵを含む同じコンピューティングデバイスであることさえも可能である。

【００５８】

より詳細に説明されるように、ＧＰＵを含むデバイスは、プロセッサを含むこともできる。プロセッサは、実行モデル１０を受信して実行モデル１０をコンパイルし、ＧＰＵが実行モデル１０によって定義されたデータ処理アルゴリズムを実装するために実行すべきオブジェクトコードにする。本開示において説明される技法により、プロセッサは、ＧＰＵの処理能力を説明する実行モデル１０をコンパイルすることができる。

【００５９】

従って、開発者は、プラットフォームから独立した方法で（すなわち、データ処理アルゴリズムを実装するＧＰＵのタイプを考慮せずに）実行モデル１０のパイプライントポロジーを定義することができる。ＧＰＵを含むデバイスのプロセッサは、ＧＰＵが実行モデル１０を実装すべき方法を定義する実行モデル１０に基づいて命令を生成することができる。例えば、プロセッサは、実行モデル１０をコンパイルし、コンパイルの一部として命令を生成することができる。実行モデル１０のコンパイル中には、プロセッサは、ＧＰＵの処理能力を説明することができる。この方法により、プロセッサは、ＧＰＵでの実装のために実行モデル１０を最適化することができる。これは、開発者が、結果的に非常にポータブルな実行モデル（すなわち、異種計算のために異なるタイプのＧＰＵで効率的に実装することができるモデル）が得られる柔軟で理解しやすい方法で実行モデル１０を開発するのを可能にすることができる。開発者は、ＧＰＵのプラットフォーム専用の又は実装によって定義される挙動に関わる必要がない。

【００６０】

さらに、バウンドされた方法で実行モデル１０を定義することは、実行モデル１０の組み込み式のデバッギングを可能にすることができる。一例として、実行モデル１０を定義することは、上述されるように、行き詰まりの機会を低減させることができる。さらに、バウンドされない実行モデルに関しては、開発者が第２のカーネルによって消費されるデータを出力するように第１のカーネルを定義し、実装中に第１のカーネルを実行させるように不注意に第２のカーネルを定義し、従って、第１のカーネルが第２のカーネルによって生成されたデータを消費するようにする可能性がある。これは、実際には、無限のループを作り出す。バウンドされない実行モデルの場合は、実装まで該状況を決定する方法がない。

【００６１】

しかしながら、バウンドされた実行モデル１０を用いることで、開発者は、該無限ループを作り出すのを簡単に回避することができる。例えば、開発者は、該無限ループを作り出したことをＧＵＩで見ることができる。他の例として、アプリケーションが実行モデル１０のパイプライントポロジーを変換して実行モデル１０のコマンドリスト内に入れたときには、アプリケーションは、該無限ループが存在するかどうかを決定することができる。

【００６２】

図２は、本開示において説明される１つ以上の例によるデバイスの例を示したブロック図である。例えば、図２は、デバイス１６を例示する。デバイス１６の例は、ビデオデバイス、例えば、メディアプレーヤー、セットトップボックス、無線ハンドセット、例えば

、携帯電話、パーソナルデジタルアシスタント（ＰＤＡ）、デスクトップコンピュータ、ラップトップコンピュータ、ゲームプレイコンソール、ビデオ会議装置、タブレットコンピューティングデバイス、等を含み、ただしこれらに限定されない。デバイス１６は、図２において例示されるコンポーネントに加えてのそれらを含むことができる。

【００６３】

例示されるように、デバイス１６は、集積回路（ＩＣ）１８と、グローバルメモリ２０と、を含む。グローバルメモリ２０は、デバイス１６のためのメモリとみなすことができる。例えば、グローバルメモリ２０は、ＩＣ１８の外部に存在することができ及びＩＣ１８及びグローバルメモリ２０は、システムバス３６を介して通信することができる。グローバルメモリ２０は、１つ以上のコンピュータによって読み取り可能な記憶媒体を備えることができる。グローバルメモリ２０の例は、ランダムアクセスメモリ（ＲＡＭ）、又は希望されるプログラムコードを搬送又は格納するために使用することができ及びコンピュータ又はプロセッサによってアクセスすることができるその他のあらゆる媒体を含み、ただしこれらに限定されない。

【００６４】

幾つかの態様では、グローバルメモリ２０は、本開示においてプロセッサ２２及びＧＰＵ２４に起因する機能を実行することをプロセッサ２２及び／又はグラフィックス処理ユニット（ＧＰＵ）２４に行わせる命令を含むことができる。従って、グローバルメモリ２０は、実行されたときに様々な機能を実行することを１つ以上のプロセッサ（例えば、プロセッサ２２及びＧＰＵ２４）に行わせる命令が格納されているコンピュータによって読み取り可能な記憶媒体であることができる。

【００６５】

グローバルメモリ２０は、幾つかの例では、非一時的な記憶媒体であるとみなすことができる。用語“非一時的な”は、記憶媒体が搬送波又は伝搬される信号において具現化されないことを示すことができる。しかしながら、用語“非一時的な”は、グローバルメモリ２０が移動不能である又はその内容が静的であることを意味するとは解釈されるべきでない。一例として、グローバルメモリ２０は、データ１６から取り外して他のデバイスに移動させることができる。他の例として、グローバルメモリ２０に実質的に類似するグローバルメモリは、デバイス１６内に挿入することができる。幾つかの例では、非一時的な記憶媒体は、（例えば、ＲＡＭにおいて）経時で変化することが可能なデータを格納することができる。

【００６６】

ＩＣ１８は、プロセッサ２２と、グラフィックス処理ユニット（ＧＰＵ）２４と、を含む。ＩＣ１８は、追加のコンポーネント、例えば、グローバルメモリ２０と通信するためのインタフェースユニット、グローバルメモリ２０内のメモリを管理するためのユニット、及びその他の処理ユニット、例えば、ディスプレイプロセッサ、を含むことができる。ＩＣ１８は、プロセッサ２２及びＧＰＵ２４を収納又は形成するあらゆるタイプの集積回路であることができる。例えば、ＩＣ１８は、チップパッケージ内の処理チップであるとみなすことができる。

【００６７】

プロセッサ２２及びＧＰＵ２４は、単一のＩＣ１８の一部として例示されているが、本開示の態様はそのようには限定されない。幾つかの例では、プロセッサ２２及びＧＰＵ２４は、異なる集積回路（すなわち、異なるチップパッケージ）に収納することができる。

【００６８】

プロセッサ２２及びＧＰＵ２４の例は、デジタル信号プロセッサ（ＤＳＰ）、汎用マイクロプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルロジックアレイ（ＦＰＧＡ）、又はその他の同等の集積回路又はディスクリート論理回路を含み、ただし、これらに限定されない。幾つかの例では、ＧＰＵ２４は、グラフィックス処理に適する大規模な並列処理能力をＧＰＵ２４に提供する集積回路及び／又はディスクリート論理回路を含む専用ハードウェアであることができる。幾つかの例では、ＧＰＵ２４は

、汎用処理を含むこともでき、汎用の処理タスク（すなわち、グラフィックスに関連しないタスク）を実装するときには汎用ＧＰＵ（ＧＰＧＰＵ）と呼ぶことができる。

【００６９】

プロセッサ２２は、ホストと時々呼ばれ、デバイス１６の中央処理装置（ＣＰＵ）であることができる。プロセッサ２２は、様々なタイプのアプリケーションを実行することができる。アプリケーションの例は、ウェブブラウザ、電子リーダー、電子メールアプリケーション、スプレッドシート、ビデオゲーム、映像再生、音声再生、ワードプロセシング、表示のためのビュー可能なオブジェクトを生成するその他のアプリケーション、又はその他のタイプのアプリケーションを含む。グローバルメモリ２０は、１つ以上のアプリケーションの実行のための命令を格納することができる。

【００７０】

幾つかの例では、プロセッサ２２は、処理タスク、例えば、大規模な並列演算を要求するタスク、をＧＰＵ２４に委託することができる。一例として、グラフィックス処理は、大規模な並列演算を要求し、プロセッサ２２は、該グラフィックス処理タスクをＧＰＵ２４に委託することができる。幾つかの例では、プロセッサ２２は、グラフィックス処理に関連しないタスクをＧＰＵ２４に委託することができる。例えば、データ処理アルゴリズム、例えば、行列演算、画像処理、及び映像処理、は、並列演算を要求し、ＧＰＵ２４は、プロセッサ２２と比較して該演算を実装するのにより適している。

【００７１】

タスクを実装するために、ＧＰＵ２４は、１つ以上のアプリケーションを実行するように構成することができる。例えば、グラフィックスに関連する処理の場合は、ＧＰＵ２４は、バーテックスシェーダ、フラグメントシェーダ、及び幾何シェーダ、等のアプリケーションを実行することができる。グラフィックスに関連しない処理に関しては、ＧＰＵ２４は、該処理（例えば、グラフィックス関連処理又はグラフィックスに関連しない処理）に関して設計されたアプリケーションを実行することができる。いずれの例（例えば、グラフィックス関連処理又はグラフィックスに関連しない処理）に関しても、プロセッサ２２は、以下においてより詳細に説明されるように、１つ以上のアプリケーションを実行するようにＧＰＵ２４に命令することができる。

【００７２】

プロセッサ２２は、特定のアプリケーション処理インタフェース（ＡＰＩ）によりＧＰＵ２４と通信することができる。例えば、プロセッサ２２は、命令、例えば、ＡＰＩを利用して１つ以上のアプリケーションを実行するようにＧＰＵ２４に命令する命令、をＧＰＵ２４に送信することができる。該ＡＰＩの例は、Ｍｉｃｒｏｓｏｆｔ（登録商標）によるＤｉｒｅｃｔＸ（登録商標）、Ｋｈｒｏｎｏｓグループによる ＯｐｅｎＧＬ（登録商標）、及びＫｈｒｏｎｏｓグループによるＯｐｅｎＣＬ（登録商標）を含む。しかしながら、本開示の態様は、ＤｉｒｅｃｔＸ、ＯｐｅｎＧＬ又はＯｐｅｎＣＬ ＡＰＩには限定されず、開発済みの、現在開発中の、又は将来開発予定のその他のタイプのＡＰＩにまで拡張することができる。さらに、本開示において説明される技法は、ＡＰＩにより機能することは要求されず、プロセッサ２２及びＧＰＵ２４は、あらゆる通信技法を利用することができる。

【００７３】

一例として、グラフィックス関連のアプリケーションに関しては、プロセッサ２２は、ＯｐｅｎＧＬ ＡＰＩを用いてＧＰＵと通信することができる。グラフィックスに関連しないアプリケーションに関しては、プロセッサ２２は、ＯｐｅｎＣＬ ＡＰＩを用いてＧＰＵ２４と通信することができる。繰り返すと、本開示において説明される技法は、プロセッサ２２がＯｐｅｎＧＬ及び／又はＯｐｅｎＣＬ ＡＰＩを用いてＧＰＵ２４と通信することは必ずしも要求しない。

【００７４】

ＧＰＵ２４が実行するグラフィックス関連のアプリケーションは、シェーダと呼ぶことができ、ＧＰＵ２４が実行するグラフィックスに関連しないアプリケーションは、カーネ

ルと呼ぶことができる。例えば、グローバルメモリ２０は、シェーダ及びカーネルの命令を格納することができ、プロセッサ１４で実行するコンパイラは、シェーダ及びカーネルの命令をＧＰＵ１６での実行のためのオブジェクトコードに変換することができる。他の例として、グローバルメモリ２０は、ＧＰＵ１６が取り出して実行するシェーダ及びカーネルのオブジェクトコードを格納することができる。

【００７５】

図２において例示されるように、グローバルメモリ２０は、実行モデル（図１）を格納する。例えば、グローバルメモリ２０は、実行モデル１０のパイプライントポロジーを定義するコマンドのリストを格納することができる。上述されるように、開発者は、カーネル１４Ａ乃至１４Ｃを含むために実行モデル１０のパイプライントポロジーを定義しておくことができる。従って、グローバルメモリ２０は、カーネル１４Ａ乃至１４Ｃのソースコードを格納することができる。代替として又はさらに加えて、グローバルメモリ２０は、カーネル１４Ａ乃至１４Ｃの予めコンパイルされたソースコード（すなわち、カーネル１４Ａ乃至１４Ｃのオブジェクトコード）を格納することができる。開発者がより多くの又はより少ないカーネル又は異なるカーネルを含むように実行モデルを定義した場合は、グローバルメモリ２０は、それらのカーネルに関するソースコード及び／又はオブジェクトコードを格納することができる。カーネル１４Ａと１４Ｃとの間の楕円形は、カーネル１４Ｂもグローバルメモリ２０に含まれることを示す。

【００７６】

この例では、プロセッサ２２は、実行モデル１０のパイプライントポロジーをグローバルメモリ２０から取り出すことができる。実行モデル１０のパイプライントポロジーに基づき、プロセッサ２２は、実行モデル１０がバッファ１２Ａ乃至１２Ｄを含むことを決定することができる。この例では、プロセッサ２２は、バッファ１２Ａ乃至１２Ｄに関してグローバルメモリ２０内で記憶場所を予約することができる。例えば、実行モデル１０の一部は、バッファ１２Ａ乃至１２Ｄの特徴、例えば、バッファ１２Ａ乃至１２Ｄのサイズ、を含むことができる。この例では、プロセッサ２２は、バッファ１２Ａ乃至１２Ｄの特徴に基づいてグローバルメモリ２０内の記憶場所を予約することができる。バッファ１２Ａと１２Ｄとの間の楕円形は、バッファ１２Ｂ及び１２Ｃもグローバルメモリ２０に含まれることを示す。

【００７７】

他の例として、実行モデル１０の一部は、カーネル１４Ａ乃至１４Ｃの特徴、例えば、拡大係数及び／又はカーネル１４Ａ乃至１４Ｃが生成する最大データ量、を含むことができる。この例では、プロセッサ２２は、カーネル１４Ａ乃至１４Ｃの特徴に基づいてグローバルメモリ２０内の記憶場所を予約することができる。例えば、拡大係数及び／又はカーネル１４Ａ乃至１４Ｃが生成する最大データ量を示す値に基づいて、プロセッサ２２は、バッファ１２Ａ乃至１２Ｄに関する該当するサイズを決定し、決定されたサイズに基づいてグローバルメモリ２０内の記憶場所を予約することができる。

【００７８】

プロセッサ２２は、バッファ１２Ａ乃至１２Ｄに関するグローバルメモリ２０内の記憶場所を予約するとして説明されるが、本開示の態様はそのようには限定されないことが理解されるべきである。幾つかの例では、ＩＣ１８又はＧＰＵ２４は、実行モデル１０によって定義されるデータ処理アルゴリズムを実装するためにＧＰＵ２４によって使用されるバッファを管理するように構成される管理ユニット（図２には示されない）を含むことができる。これらの例では、プロセッサ２２は、バッファ１２Ａ乃至１２Ｄのサイズに関して管理ユニットを命令することができ、管理ユニットは、バッファ１２Ａ乃至１２Ｄに関してグローバルメモリ内の記憶場所を予約するように構成することができる。

【００７９】

この管理ユニットは、その他の機能、例えば、バッファ１２Ａ乃至１２Ｄ内に格納されたキャッシュバッキング（ｃａｃｈｅ－ｂａｃｋｉｎｇ）データ及び／又はＩＣ１８又はＧＰＵ２４のキャッシュ内のカーネル１４Ａ乃至１４Ｃの命令、を実行するように構成す

ることができる。この管理ユニットは、バッファ１２Ａ乃至１２Ｄの各々の１つに格納されるデータ量を示す情報を格納することもできる。この管理ユニットは、ＧＰＵ２４での実行時にカーネル１４Ａ乃至１４Ｃ間でのデータ転送を管理するように構成することができる。例えば、図１において例示されるように、実行モデル１０のパイプライントポロジーは、カーネル１４Ａがバッファ１２Ｂに出力し、カーネル１４Ｂがバッファ１２Ｂからデータを受信することを示す。管理ユニットは、カーネル１４Ａによって生成されたデータのバッファ１２Ｂ内での格納、及びカーネル１４Ｂによるバッファ１２Ｂからのデータの取り出し、そして幾つかの例では、バッファ１２Ｂに格納されるデータ量の格納を管理するように構成することができる。管理ユニットに関する技法は、"GRAPHICS PROCESSING UNIT BUFFER MANAGEMENT"（グラフィックス処理ユニットバッファ管理）という題名を有し、引用によってその内容全体が組み入れられている同時係属米国特許出願第１３／７４７，９４７号（出願日：２０１３年１月２３日）においても記述される。

【００８０】

管理ユニットの利用は、例示することを目的として提供されるものであり、限定するものであるとはみなされるべきでない。例えば、管理ユニットは、プロセッサ２２以外のユニットがグローバルメモリ２０内のバッファ１２Ａ乃至１２Ｄに関する記憶場所を予約する１つの方法例として説明される。しかしながら、本開示の態様は、そのようには限定されず、プロセッサ２２、又はＧＰＵ２４でさえも、該機能を果たすように構成することができる。例えば、ＧＰＵ２４がバッファ１２Ａ乃至１２Ｄ内にデータを格納するときには、ＧＰＵ２４は、ＧＰＵ２４がバッファ１２Ａ乃至１２Ｄに格納したデータ量を格納するようにも構成することができる。説明を容易にするため、技法は、プロセッサ２２又はＧＰＵ２４が該機能を実行することに関して説明される。

【００８１】

本開示において説明される技法により、プロセッサ２２は、実行モデル１０によって定義されたパイプライントポロジーのインディケーション（ｉｎｄｉｃａｔｉｏｎ）（例えば、一例として、実行モデル１０のコマンドリスト）を受信することができる。プロセッサ２２は、ＧＰＵ２４がパイプライントポロジーを実装する方法を定義する命令を生成することができる。

【００８２】

例えば、例示されるように、プロセッサ２２は、コンパイラ２８を実行することができる。コンパイラ２８は、プロセッサ２２内で形成されていないことを示すために破断線で示される。むしろ、グローバルメモリ２０は、コンパイラ２８のオブジェクトを格納することができ、プロセッサ２２が取り出して実行する。

【００８３】

コンパイラ２８は、ＧＰＵ２４が実行するオブジェクトコード、及びＧＰＵ２４が実行モデル１０を実装する方法を定義する命令を生成するために実行モデル１０（例えば、実行モデル１０のコマンドリスト）をコンパイルするように構成することができる。ＧＰＵ２４が実行モデル１０を実装する方法を定義する命令を生成するためのコンパイルの一部として、コンパイラ２８は、ＧＰＵ２４の処理能力を説明する（ａｃｃｏｕｎｔ　ｆｏｒ）ことができる。

【００８４】

例えば、例示されるように、グローバルメモリ２０は、ＧＰＵ構成３２を格納することができる。ＧＰＵ構成３２は、ＧＰＵ２４の処理能力を定義する又は示す構成情報であることができる。一例として、ＧＰＵ構成３２は、ＧＰＵ２４内のプログラマブル計算ユニット数を示すことができる。上述されるように、カーネルは、ＧＰＵ内の１つ以上のプログラマブル計算ユニットで実行する。

【００８５】

他の例として、ＧＰＵ構成３２は、ＧＰＵ２４が並列でデータを処理することが可能な方法を示すことができる。例えば、ＧＰＵ２４は、単一プログラム多データ（ＳＰＭＤ）プログラミングモデル又は単一命令多データ（ＳＩＭＤ）プログラミングモデルを実装す

るように構成することができる。一例として、ＧＰＵ２４がＳＩＭＤプログラミングモデルに関して構成される場合は、ＧＰＵ構成３２は、ＳＩＭＤプログラミングモデル（例えば、８レーンＳＩＭＤ）を実装するためのＧＰＵ２４内のレーン（ｌａｎｅ）の数を示す構成情報を含むことができる。

【００８６】

ＧＰＵ構成３２は、上述される情報の追加の又は上述される情報と異なるＧＰＵ２４の構成情報を含むことができる。概して、ＧＰＵ構成３２は、ＧＰＵ２４の処理能力を記述する構成情報を含むことができる。

【００８７】

さらに、ＧＰＵ構成３２はグローバルメモリ２０に格納されるとして例示されるが、本開示の態様は、そのようには限定されない。幾つかの例では、ＩＣ１８内のレジスタ又はキャッシュがＧＰＵ構成３２を格納することができる。これらの例の両方において、プロセッサ２２は、グローバルメモリ２０からではなく、レジスタからＧＰＵ構成３２の情報を読み取ることができる。幾つかの例では、プロセッサ２２をＧＰＵ構成３２で予め構成することさえも可能である。

【００８８】

コンパイラ２８は、実行モデル１０をコンパイルするために、実行モデル１０の情報に加えて、ＧＰＵ構成３２の情報を利用することができる。コンパイルの結果は、ＧＰＵ２４が実行するオブジェクトコード、及び、ＧＰＵ２４が実行モデル１０を実装する方法に関する命令であることができる。例えば、オブジェクトコードに加えて、コンパイラ２８の出力は、プロセッサ２２がグローバルメモリ２０に格納するメタースケジューラ命令３４であることができる。メタースケジューラ命令３４は、より詳細に説明されるように、ＧＰＵ２４が実行モデル１０を実装する方法を指示する、メタースケジューラ３０に関する命令であることができる。

【００８９】

例えば、メタースケジューラ命令３４は、ＧＰＵ２４がカーン処理ネットワーク（Ｋａｈｎ　Ｐｒｏｃｅｓｓｉｎｇ　Ｎｅｔｗｏｒｋ）（ＫＰＮ）に類似する実行モデル１０を実装するように指示することができる。例えば、ＫＰＮは、データを含むチャネルを決定し、そのチャネルに関するコンシューマ（ｃｏｎｓｕｍｅｒ）を識別し、データのうちの一部の量に関してコンシューマを実行し、全データが処理されるまでこれらのステップを繰り返す。実行モデル１０のトポロジーは、（ＫＰＮのプロセスに類似する）カーネル及び（ＫＰＮのプロセスに類似する）バッファを定義することができる。この方法により、実行モデル１０は、バッファの各々に関する消費するカーネルを示す。より詳細に説明されるように、実行モデル１０を実装する際には、ＧＰＵ２４は、データを含むバッファ１２Ａ乃至１２Ｄのうちの１つを識別することができ、及び、バッファ１２Ａ乃至１２Ｄのうちの識別されたそれのデータを消費することになる消費するカーネル（例えば、カーネル１４Ａ乃至１４Ｃのうちの１つ）を実行することができる。ＫＰＮに関する説明は、単に例示する目的で及び理解を助けるために提供されたものであることが理解されるべきである。本開示において説明される技法は、ＫＰＮのそれらに限定される又はＫＰＮのそれらと同一であるとはみなされるべきではない。

【００９０】

それらの技法により、メタースケジューラ命令３４は、デバイスをターゲットにしたバイナリ（ｄｅｖｉｃｅ　ｔａｒｇｅｔ　ｂｉｎａｒｙ）であることができる。換言すると、実行モデル１０はプラットフォームから独立する（すなわち、ＧＰＵ専用でない）ことができる一方で、メタースケジューラ命令３４は、プラットフォームに依存する（すなわち、ＧＰＵ２４専用である）。例えば、コンパイラ２８は、ＧＰＵ２４での実行モデル１０の実装を最適化するためにＧＰＵ構成３２からの情報を利用することができる。

【００９１】

一例として、コンパイラ２８は、カーネル１４Ａ乃至１４ＣがＧＰＵ２４のプログラマブル計算ユニットで実行する時間を決定するために情報、例えば、ＧＰＵ２４のプログラ

マブル計算ユニット数、を利用することができる。例えば、上述されるように、実行モデル１０によって定義されるデータ処理アルゴリズムのパイプライン実装は、カーネル１４Ａ乃至１４Ｃのうちの１つ以上の並列の実行（例えば、同時）又はカーネル１４Ａ乃至１４Ｃのうちの１つ以上の順次の実行（例えば、１つずつ）を含むことができる。この例では、コンパイラ２８は、カーネル１４Ａ乃至１４Ｃが、数多くの利用可能なプログラマブル計算ユニットが存在しない場合は順次で実行すべきであること又は数多くの利用可能なプログラマブル計算ユニットが存在する場合は並列で実行すべきであることを指示するメタースケジューラ命令３４を生成することができる。

【００９２】

他の例として、コンパイラ２８は、カーネル１４Ａ乃至１４Ｃの一部のそれらがその他よりも重要であることを示す実行モデル１０からの情報を利用することができる。例えば、カーネル１４Ｂがカーネル１４Ｃよりも重要であると仮定する。この例では、コンパイラ２８は、カーネル１４Ｃが幾つかの割り込み有りで実行することになった場合でもカーネル１４Ｂは割り込みなしで実行するようにするメタースケジューラ命令３４を生成するためにＧＰＵ構成３４からの情報を利用することができる。

【００９３】

例えば、プログラマブル計算ユニットが１つのカーネルのスレッドを実行することから他のカーネルのスレッドを実行することに切り換わり、次に戻ることが可能である。この場合は、プログラマブル計算ユニットが切り換わったカーネルは割り込まれたとみなすことができ、プログラマブル計算ユニットがそのカーネルのスレッドを実行するために切り換わって戻るまで休止状態にあるとみなすことができる。幾つかの例では、コンパイラ２８は、重要なカーネルのスレッドを実行中のプログラマブル計算ユニットが他のカーネルのスレッドの実行に切り換わらないように指示するメタースケジューラ命令３４を生成することができる。

【００９４】

さらに他の例として、コンパイラ２８は、ＧＰＵ２４内のデータレーン（ｄａｔａ　ｌａｎｅ）の数を示すＧＰＵ構成３４からの情報を利用することができる。例えば、ＧＰＵ構成３４は、ＧＰＵ２４が８レーンのＳＩＭＤ　ＧＰＵであることを示すと仮定する。この例では、コンパイラ２８は、消費するカーネルに結合されたバッファ内に少なくとも８つのエントリが存在するまでＧＰＵ２４が消費するカーネルを実行すべきでないことを指示するメタースケジューラ命令３４を生成することができる。例えば、図１において例示されるように、カーネル１４Ｂは、カーネル１４Ａにとっての消費するカーネルであり、バッファ１２Ｂからデータを受信する。ＧＰＵ２４は８レーンＳＩＭＤ　ＧＰＵであると仮定すると、コンパイラ２８は、少なくとも８つのデータ項目がバッファ１２Ｂ内に格納されるまでＧＰＵ２４がカーネル１４Ｂを実行すべきでないことを指示するメタースケジューラ命令３４を生成することができる。

【００９５】

追加の例として、コンパイラ２８は、バッファ１２Ａ乃至１２Ｄのサイズを説明することができる。例えば、コンパイラ２８は、行き詰まりの変化が最小限になるようにカーネル１４Ａ乃至１４Ｃをいつ実行すべきかを決定するためにバッファ１２Ａ乃至１２Ｄのサイズに関する情報を利用することができる。この例では、コンパイラ２８は、行き詰まりが発生しないようにカーネル１４Ａ乃至１４Ｃが実行する順序を指示するメタースケジューラ命令３４を生成することができる。

【００９６】

幾つかの例では、コンパイラ２８は、ＧＰＵ２４に実行モデル１０を実装する際に誤りが存在するかどうかを決定するように構成することができる。例えば、コンパイラ２８は、実行モデル１０の一部であるが、カーネル１４Ａ乃至１４Ｄのいずれの１つにも結合されていないバッファ１２Ａ乃至１２Ｄが存在するかどうかを決定するように構成することができる。他の例として、コンパイラ２８は、カーネル１４Ａ乃至１４Ｄのうちのいずれか１つが存在していないバッファにアクセスしようとしているか、又はバッファ内のアウ

トオブバウンド（ｏｕｔ－ｏｆ－ｂｏｕｎｄｓ）記憶場所にアクセスしようとしているかを決定するように構成することができる。コンパイラ２８がコンパイル時に実行モデル１０の機能を検証するため、コンパイラ２８は、実行モデル１０の機能を検証することをＧＰＵ２４に行わせる命令をメタースケジューラ命令３４に含める必要がない。

【００９７】

プロセッサ２２が、コンパイラ２８を介して、メタースケジューラ命令３４を生成した後は、プロセッサ２２は、メタースケジューラ命令３４を実行のために取り出すようにＧＰＵ２４に命令することができる。例示されるように、ＧＰＵ２４は、メタースケジューラ３０を含む。メタースケジューラ３０は、ＧＰＵ２４内のハードウェア、ＧＰＵ内のハードウェアで実行するファームウェア、又はＧＰＵ２４内のハードウェアで実行するソフトウェアであることができる。メタースケジューラ３０は、メタースケジューラ命令３４の命令を実行するように構成することができる。

【００９８】

本開示において説明される技法では、メタースケジューラ３０は、ＧＰＵ２４のいずれのプログラマブル計算ユニットがカーネル１４Ａ乃至１４Ｃのいずれのスレッドを何時に実行すべきかを決定するように構成することができる。換言すると、メタースケジューラ３０は、実行モデル１０のパイプラインドポロジーによって定義されたデータ処理アルゴリズムを実装することをＧＰＵ２４に行わせるためにＧＰＵ２４でのカーネル１４Ａ乃至１４Ｃの実行をスケジューリングするように構成することができる。本開示において説明される技法により、メタースケジューラ３０は、メタースケジューラ命令３４に基づいてＧＰＵ２４でカーネル１４Ａ乃至１４Ｃを実行するスケジュールを決定することができる。

【００９９】

例えば、メタースケジューラ命令３４は、カーネル１４Ａ乃至１４Ｃのうちの１つ以上を並列で又は順次実行すべきかを指示することができる。この例では、メタースケジューラ３０は、並列又は順次の実行を達成するためにいずれのプログラマブル計算ユニットがカーネル１４Ａ乃至１４Ｃのスレッドを実行すべきかを決定することができる。他の例として、メタースケジューラ命令３４は、カーネル１４Ａ乃至１４Ｃの重要性を示すことができる。この例では、メタースケジューラ３０は、重要なカーネルを実行するプログラマブル計算ユニットがカーネルを実行中に割り込まれないようにいずれのプログラマブル計算ユニットがカーネル１４Ａ乃至１４Ｃのスレッドを実行すべきかを決定することができる。他の例として、メタースケジューラ命令３４は、ＧＰＵ２４のＳＩＭＤ又はＳＰＭＤ能力に基づいてカーネル１４Ａ乃至１４Ｃのうちの１つがいつ実行すべきかを指示することができる。この例では、メタースケジューラ３０は、メタースケジューラ命令３４の命令に基づいて、いずれのプログラマブル計算ユニットが何時にスレッドを実行するかを決定することができる。メタースケジューラ３０は、行き詰まりを回避するためにカーネル１４Ａ乃至１４Ｃがいつ実行すべきかのタイミングを指示するメタースケジューラ命令３４内の命令を利用することもできる。

【０１００】

繰り返すと、コンパイラ２８は、メタースケジューラ命令３４を生成する際に、ＧＰＵ構成３２内の情報に基づいて、ＧＰＵ２４の計算能力を考慮に入れておくことができる。従って、本開示において説明される技法は、メタースケジューラ３０がメタースケジューラ３４によって示された方法でカーネル１４Ａ乃至１４Ｃを実行するためにプログラマブル計算ユニットを適切に割り当てることができることをある程度のレベルで保証する。例えば、上述されるように、コンパイラ２８は、メタースケジューラ命令３４を生成するために、幾つかの説明例として、幾つかの要因、例えば、ＧＰＵ２４内でのプログラマブル計算ユニットの数、バッファ１２Ａ乃至１２Ｄのサイズ、ＧＰＵ２４のＳＩＭＤ又はＳＰＭＤ能力、又はカーネル１４Ａ乃至１４Ｃの重要性、を説明することができる。ＧＰＵ２４のメタースケジューラ３０がカーネル１４Ａ乃至１４Ｃがどのように実行されるべきかを決定するためにメタースケジューラ命令３４を利用するときには、カーネル１４Ａ乃至

１４ＣはＧＰＵ２４で効率的に実行することをある程度保証することができる。
【０１０１】
この方法により、プロセッサ２２は、実行モデル１０のパイプライントポロジーを示すインディケーションを受信することができ、実行モデル１０は、プラットフォームから独立した形で定義される。プロセッサ２２は、コンパイラ２８を実行することができ、それは、実行モデル１０のパイプライントポロジーがどのようにしてＧＰＵ２４上に実装されるべきかをプラットフォーム専用の方法で定義するメタースケジューラ命令３４を生成するためのＧＰＵ２４の処理能力を説明する。ＧＰＵ２４のメタースケジューラ３０は、メタースケジューラ命令３４を受信し、メタースケジューラ命令３４の命令に基づいて、いずれのプログラマブル計算ユニットが何時にカーネル１４Ａ乃至１４Ｃのスレッドを実行すべきかどうかを決定することができる。
【０１０２】
メタースケジューラ命令３４は、各プラットフォーム専用であるため、メタースケジューラ命令３４は、ＧＰＵ２４での実行モデル１０のパイプライントポロジーの最適な実装になるようにカーネル１４Ａ乃至１４Ｃが実行されるべき方法を定義することができる。例えば、メタースケジューラ命令３４がＧＰＵ２４と異なるタイプのＧＰＵによって使用される場合は、この異なるタイプのＧＰＵは、メタースケジューラ命令３４がＧＰＵ２４のプラットフォーム専用であるため。実行モデル１０のパイプライントポロジーを効率的に実装することができない。
【０１０３】
幾つかの例では、メタースケジューラ３０は、実行ポリシーを実装するように構成することができる。例えば、すべての例においてコンパイラ２８が、カーネル１４Ａ乃至１４Ｃがいつ実行されるべきかを正確に定義する必要がないことがある。むしろ、コンパイラ２８が生成するメタースケジューラ命令３４は、カーネル１４Ａ乃至１４ＣをＧＰＵ２４で実行すべきことを指示することができ及びいずれのカーネル１４Ａ乃至１４Ｃが生成するカーネルであり、いずれが消費するカーネルであるかを示すことができる。
【０１０４】
これらの例では、メタースケジューラ３０は、カーネル１４Ａ乃至１４Ｃがいつ実行すべきかを示す実行ポリシーを実装するように構成することができる。実行ポリシーの一例は、バッファ１２Ａ乃至１２Ｄのうちのいずれがデータを格納するかをメタースケジューラ３０が決定し、データを格納するバッファ１２Ａ乃至１２Ｄからデータを受信するカーネル１４Ａ乃至１４Ｃのうちの１つ以上を実行する。例えば、メタースケジューラ３０は、バッファ１２Ａ乃至１２Ｄをラウンドロビン方式で検査することができ、及び、データを格納するバッファ１２Ａ乃至１２Ｄからデータを消費する全カーネルを実行することができる。
【０１０５】
実行ポリシーの他の例として、メタースケジューラ３０は、カーネル１４Ａ乃至１４Ｃの重要性に基づいていずれのバッファ１２Ａ乃至１２Ｄがデータを格納するかを決定することができる。カーネル１４Ａ乃至１４Ｃの重要性は、カーネル１４Ａ乃至１４Ｃの優先度によって定義することができる。メタースケジューラ３０は、最初に、カーネル１４Ａ乃至１４Ｄの最高の優先度を有するカーネルがデータを受信するバッファ１２Ａ乃至１２Ｄのバッファを検査することができる。そのバッファがデータを格納している場合は、メタースケジューラ３０は、最高の優先度を有するカーネルを実行することができる。次に、メタースケジューラ３０は、カーネル１４Ａ乃至１４Ｄの次に最高の優先度を有するカーネルがデータを受信するバッファ１２Ａ乃至１２Ｄのバッファを検査することができる。そのバッファがデータを格納している場合は、メタースケジューラ３０は、次に最高の優先度を有するカーネルを実行することができ、以下同様である。
【０１０６】
実行ポリシーのさらに他の例として、メタースケジューラ３０は、バッファ１２Ａ乃至１２Ｄのちのいずれが最も多くの量のデータを格納するかを決定することができる。メタ

ースケジューラ３０は、最も多くの量のデータを格納するバッファ１２Ａ乃至１２Ｄのバッファからデータを受信するカーネル１４Ａ乃至１４Ｃのカーネルを実行することができる。一例として、ＣＰＵ２４がバッファ１２Ａ乃至１２Ｄのうちの１つにデータを書き込むときには、ＧＰＵ２４は、ＧＰＵ２４がデータを書き込んだバッファ１２Ａ乃至１２Ｄのうちの１つに格納されるデータの量を示す情報を格納することができる。これらの例では、メタースケジューラ３０は、ＧＰＵ２４が格納し、バッファ１２Ａ乃至１２Ｄ内のデータの量を示す情報に基づいてバッファ１２Ａ乃至１２Ｄのうちのいずれの１つが最も多くの量のデータを格納するかを決定するように構成することができる。

【０１０７】

しかしながら、メタースケジューラ３０を実行ポリシーを実装するように予め構成する必要がない。むしろ、コンパイラ２８が、コンパイラ２８によって生成されるメタースケジューラ命令３４の命令の一部としてメタースケジューラ３０の実行ポリシーを決定することが可能である。

【０１０８】

さらに、開発者がメタースケジューラ３０の実行ポリシーを定義することが可能である。例えば、開発者は、実行モデル１０の一部としてメタースケジューラ３０の実行ポリシーを定義することができ、コンパイラ２８は、開発者によって定義された実行ポリシーに関してメタースケジューラ３０に命令するメタースケジューラ命令３４の命令を生成するためにこの開発者によって定義された実行ポリシーを利用することができる。

【０１０９】

しかしながら、開発者が実行ポリシーを定義しないほうが適切である場合がある。例えば、開発者が実行ポリシーを定義した場合は、実行ポリシーは、プラットフォームから独立した方法では適切に機能しないことがある。すべてのＧＰＵタイプに関して適切に機能し、決定するアプリケーションにおいて同じ機能上の結果（例えば、同じアプリケーションに関する異なるＧＰＵタイプにわたって同じ結果）を生み出す実行ポリシーを開発者が開発するのは困難であろう。

【０１１０】

概して、開発者は、カーネル１４Ａ乃至１４Ｃが実行モデル１０によって定義されたデータ処理アルゴリズムを実装するために適切に実行するかぎりは実行ポリシーには特別の関心を有さないであろう。従って、開発者が実行ポリシーを定義することができないかどうか、及びコンパイラ２８が実行ポリシーを決定することは関係ないであろう。

【０１１１】

上述される例においては、コンパイラ２８は、ＧＰＵで実行されるオブジェクトコードを生成するために及びメタースケジューラ命令３４を生成するために実行モデル１０をコンパイルした。幾つかの例では、コンパイラ２８は、ＧＰＵ２４で実行されるオブジェクトコードをグローバルメモリ２０に格納することができる。実行モデル１０が複数の実行モデルのうちの１つの実行モデルである場合は、コンパイラ２８は、実行モデル１０を再コンパイルする必要がない。換言すると、複数の実行モデルを結合することによって、及び幾つかの例では、グローバルメモリ２０に格納された実行モデルのオブジェクトコードを結合することによって、大きなデータ処理アルゴリズムを生成することが可能である。例えば、データ処理アルゴリズムは、グローバルメモリ２０に格納された実行モデルのオブジェクトコードから生成することができる。データ処理アルゴリズムの該生成は、ＧＰＵ２４がＦＰＧＡ又は埋め込まれたデバイスである例にとって有用であることができる。

【０１１２】

本開示において説明される技法では、メタースケジューラ３０に関してメタースケジューラ命令３４を取り出すようにＧＰＵ２４に命令することに加えて、プロセッサ２２は、ＧＰＵ２４がカーネル１４Ａ乃至１４Ｃを実行するために必要な追加情報をＧＰＵ２４に提供することができる。例えば、カーネル１４Ａ乃至１４Ｃは、バッファ１２Ａ乃至１２Ｄからのデータに加えて、機能するために追加情報（例えば、引数）を要求することができる。プロセッサ２２は、該追加情報をＧＰＵ２４に提供することができる。

【０１１３】
プロセッサ２２は、実行モデル１０を実装するようにＧＰＵ２４に命令することができる。ＧＰＵ２４は、メタースケジューラ命令３４の命令及びプロセッサ２２がコンパイラ２８によるコンパイルプロセスの一部として生成したオブジェクトコードに基づいて実行モデル１０を実装することができる。幾つかの例では、ＧＰＵ２４は、プロセッサ２２との同期化なしで実行モデル１０を実装することができる。
【０１１４】
さらに、幾つかの例では、コンパイラ２８及びメタースケジューラ３０をデバッグモードで構成することが可能である。例えば、開発者が実行モデル１０を開発した後に、開発者は、リリース前にＧＰＵでの実行モデルの実装を試験することを希望することができる。試験に関しては、開発者は、デバイス、例えば、デバイス１６、に実行モデル１０をロードし、ＧＰＵ２４で実行モデル１０を試験することができる。試験の一環として、開発者は、デバッグモードを利用することができる。デバッグモードでは、コンパイラ２８は、バッファ１２Ａ乃至１２Ｄの記憶場所の範囲を単一の記憶場所にまで狭める（例えば、ＮＤｒａｎｇｅサイズを最小に小さくする）メタースケジューラ命令３４を生成することができる。メタースケジューラ命令３４は、カーネル１４Ａ乃至１４Ｃのうちの１つのカーネルのみが一度に実行するように指示することもできる。
【０１１５】
デバッグモードでは、開発者は、データがバッファ１２Ａ乃至１２Ｄ内に格納されている方法、及びカーネル１４Ａ乃至１４Ｃの各々の１つがＧＰＵ２４で実行されている方法を追跡することができる。これは、開発者がカーネル１４Ａ乃至１４Ｃ内の問題又は実行モデル１０内の問題に対処するのを可能にすることができる。
【０１１６】
上述されるように、コンパイラ２８は、メタースケジューラ命令３４を生成することができる。以下は、実行モデル１０に関するメタースケジューラ命令３４の擬似コード例である。幾つかの例では、メタースケジューラ命令３４を生成する能力を開発者に与えるのではなく、コンパイラ２８がメタースケジューラ命令３４を生成するのが有益であることができる。例えば、開発者がメタースケジューラ命令３４を生成した場合は、実行モデル１０はポータブルになることができず、混乱が生じ、さらにデバッグが困難なユーザエラーが生じる可能性がある。
【０１１７】
以下の擬似コードでは、Ｆ１は、バッファ１２Ａを指し示し、Ｆ２は、バッファ１２Ｂを指し示し、Ｆ３は、バッファ１２Ｃを指し示し、Ｆ４は、バッファ１２Ｄを指し示す。Ｋ１は、カーネル１４Ａを指し示し、Ｋ２は、カーネル１４Ｂを指し示し、Ｋ３は、カーネル１４Ｃを指し示す。

【数１】

```
while (true)
{
        //Node 1
        if (K1.F1.size( ) > 0)
        {
                maxSafeNdRangeSizeK1 = min(K1.F1.size( ),
                        ((K1.F3.maxSize( ) – K1.F3.size( ))/K1.F3.ampFactor( )),
                        ((K1.F2.maxSize( ) – K1.F2.size ( ))/K1.F2.ampFactor( )));
                enqueue (K1, maxSafeNdRangeSizeK1);
        }
        //Node 2
        …
        //Node 3
        …
        If (F1.size( ) + F2.size( ) + F3.size( ) + F4.size ( ) == 0)
                && (K3 is done))
        {
                Exit //Finished execution of graph
        }
}
```

【０１１８】
図３は、本開示において説明される１つ以上の例による技法例を示したフローチャートである。例えば、図３は、異種計算（例えば、プラットフォームから独立した形での計算）に関する技法を例示する。例示を容易にするために、図２が参照される。
【０１１９】
図３において例示されるように、プロセッサ２２は、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデル１０のパイプライントポロジーのインディケーションを受信することができる（３８）。例えば、実行モデル１０のパイプライントポロジーのインディケーションは、実行モデル１０の開発者によって作成されたコマンドリストであることができる。データ処理アルゴリズムのプラットフォームから独立したに関する定義は、実行モデル１０がＧＰＵの特定のプラットフォームに基づいて設計されていない（例えば、データ処理アルゴリズムを実装するＧＰＵのタイプから独立している）ことを意味する。
【０１２０】
プロセッサ２２のコンパイラ２８は、メタースケジューラ命令３４を生成するためにパイプライントポロジーを指定するコマンドリストをコンパイルすることができる（４０）。メタースケジューラ命令３４は、ＧＰＵ２４が実行モデル１０のパイプライントポロジ

ーを実装するプラットフォームに依存した方法を指示することができる。ＧＰＵ２４がパイプラインドポロジーを実装するプラットフォームに依存した方法は、メタースケジューラ命令が（例えば、ＧＰＵ２４のＧＰＵタイプに基づいて）ＧＰＵ構成３２によって示されるＧＰＵ２４の特定のプラットフォームに基づくことを意味する。プロセッサ２２は、実行モデル１０のパイプラインドポロジーを実装するようにＧＰＵ２４に命令するための命令を送信することができる（４２）。

【０１２１】

プロセッサ２２が、コンパイラ２８を介して、メタースケジューラ命令３４を生成することができる様々な方法が存在する。一例として、コンパイラ２８は、実行モデル１０のパイプラインドポロジーがＧＰＵ２４に実装されるプラットフォームに依存する方法を定義する命令を生成するためにＧＰＵ２４の構成情報に少なくとも基づいてコマンドリストをコンパイルすることができる。ＧＰＵ構成３２は、ＧＰＵ２４の構成情報を提供することができる。例えば、構成情報は、ＧＰＵ２４内のプログラマブル計算ユニット数を含むことができる。構成情報は、ＧＰＵ２４内のデータレーン数（すなわち、ＧＰＵ２４のＳＩＭＤ又はＳＰＭＤ能力）を含むことができる。

【０１２２】

幾つかの例では、コンパイラ２８は、実行モデル１０で提供された情報に基づいてコマンドリストをコンパイルすることができる。例えば、コンパイラ２８は、メタースケジューラ命令３４を生成するために実行モデル１０のパイプラインドポロジーで識別されたバッファ（例えば、バッファ１２Ａ乃至１２Ｄ）のサイズに基づいてコマンドリストをコンパイルすることができる。他の例として、コンパイラ２８は、メタースケジューラ命令３４を生成するために実行モデル１０のパイプラインドポロジーで識別されたカーネル（例えば、カーネル１４Ａ乃至１４Ｃ）の重要性に基づいてコマンドリストをコンパイルすることができる。

【０１２３】

幾つかの要因、例えば、ＧＰＵ２４におけるプログラマブル計算ユニット数、ＧＰＵ２４におけるデータレーン数、バッファ１２Ａ乃至１２Ｄのサイズ、及びカーネル１４Ａ乃至１４Ｃの重要性、等を利用するコンパイラ２８は、例示を目的として提供されるものであり、限定するとはみなされるべきでないことが理解されるべきである。さらに、コンパイラ２８は、要因を単独で又はあらゆる組み合わせで利用することができる。例えば、コンパイラ２８は、メタースケジューラ命令３４を生成する際にこれらの要因のうちの１つのみを利用する必要はない。むしろ、コンパイラ２８は、メタースケジューラ命令３４を生成するためにこれらの要因のうちの１つ、これらの要因のうちの１つ以上、及びこれらの要因のあらゆる組み合わせを利用することができる。

【０１２４】

図４は、図２のデバイスをさらに詳細に例示したブロック図である。例えば、図４は、デバイス１６をさらに例示する。デバイス１６の例は、無線デバイス、携帯電話、パーソナルデジタルアシスタント（ＰＤＡ）、ビデオディスプレイを含むビデオゲームコンソール、モバイルビデオ会議装置、ラップトップコンピュータ、デスクトップコンピュータ、テレビセットトップボックス、ダフレットコンピューティングデバイス、電子書籍リーダー、等を含み、ただしこれらには限定されない。デバイス１６は、プロセッサ２２と、ＧＰＵ２４と、グローバルメモリ２０と、ディスプレイ４４と、ユーザインタフェース４６と、トランシーバモジュール４８と、を含むことができる。プロセッサ２２及びＧＰＵ２４は、図４において例示されるように、共通のＩＣ１８内に収納することができ、又は別々に収納することができる。さらに、例示されるように、プロセッサ２２は、メタースケジューラ命令３４を生成するためにコンパイラ２８を実行することができ、ＧＰＵ２４は、メタースケジューラ命令３４の命令を実装するように構成されたメタースケジューラ３０を含む。

【０１２５】

デバイス１６は、明確化のために図４には示されていない追加のモジュール又はユニッ

トを含むことができる。例えば、デバイス１６は、デバイス１６がモバイル無線電話である例において電話通信を有効にするためのスピーカーとマイクとを含むことができ、これらはいずれも図４には示されていない。さらに、デバイス１６内に示される様々なモジュール及びユニットは、デバイス１６のすべての例において必要であるわけではない。例えば、ユーザインタフェース４６及びディスプレイ４４は、デバイス１６がデスクトップコンピュータである例ではデバイス１６の外部に存在することができる。他の例として、ユーザインタフェース４６は、ディスプレイ４４がモバイルデバイスのタッチ感応式又はプレゼンス感応式（ｐｒｅｓｅｎｃｅ－ｓｅｎｓｉｔｉｖｅ）のディスプレイである例ではディスプレイ４４の一部であることができる。

【０１２６】

グローバルメモリ２０、プロセッサ２２、ＧＰＵ２４、コンパイラ２８、及びメタースケジューラ３０は、グローバルメモリ２０、プロセッサ２２、ＧＰＵ２４、コンパイラ２８、及びメタースケジューラ３０に類似するものであり、図４に関してはこれ以上は説明されない。ユーザインタフェース４６の例は、トラックボールと、マウスと、キーボードと、その他のタイプの入力デバイスとを含み、ただしこれらに限定されない。ユーザインタフェース４６は、タッチ画面であることもでき、ディスプレイ４４の一部として組み入れることができる。トランシーバモジュール４８は、デバイス１６と他のデバイス又はネットワークとの間の無線又は有線通信を可能にするための回路を含むことができる。トランシーバモジュール４８は、変調器と、復調器と、増幅器と、有線又は無線通信のためのその他の該回路と、を含むことができる。ディスプレイ４４は、液晶ディスプレイ（ＬＣＤ）、陰極線管（ＣＲＴ）ディスプレイ、プラズマディスプレイ、タッチ感応式ディスプレイ、プレゼンス感応式ディスプレイ、又は他のタイプのディスプレイデバイスを備えることができる。

【０１２７】

１つ以上の例では、説明される機能は、ハードウェア、ソフトウェア、ファームウェア、又はそれらのあらゆる組み合わせにおいて実装することができる。ソフトウェアにおいて実装される場合は、それらの機能は、コンピュータによって読み取り可能な媒体において１つ以上の命令又はコードとして格納又は送信すること及びハードウェアに基づく処理ユニットによって実行することができる。コンピュータによって読み取り可能な媒体は、コンピュータによって読み取り可能な記憶媒体を含むことができ、それは、有形な媒体、例えば、データ記憶媒体、又は、例えば、通信プロトコルにより、１つの場所から他へのコンピュータプログラムの転送を容易にするあらゆる媒体を含む通信媒体、に対応する。このように、コンピュータによって読み取り可能な媒体は、概して、（１）非一時的である有形なコンピュータによって読み取り可能な記憶媒体又は（２）通信媒体、例えば、信号又は搬送波、に対応することができる。データ記憶媒体は、本開示において説明される技法の実装のために命令、コード及び／又はデータ構造を取り出すために１つ以上のコンピュータ又は１つ以上のプロセッサによってアクセスすることができるあらゆる利用可能な媒体であることができる。コンピュータプログラム製品は、コンピュータによって読み取り可能な媒体を含むことができる。

【０１２８】

一例により、及び制限することなしに、該コンピュータによって読み取り可能な記憶媒体は、希望されるプログラムコードを命令又はデータ構造の形態で格納するために使用することができ及びコンピュータによってアクセス可能であるＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、ＣＤ－ＲＯＭ又はその他の光学ディスク記憶装置、磁気ディスク記憶装置、又はその他の磁気記憶デバイス、フラッシュメモリ、又はその他のいずれかの媒体を備えることができる。さらに、どのような接続も、コンピュータによって読み取り可能な媒体であると適切に呼ばれる。例えば、命令が、同軸ケーブル、光ファイバケーブル、より対線、デジタル加入者ライン（ＤＳＬ）、又は無線技術、例えば、赤外線、無線、及びマイクロ波、を用いてウェブサイト、サーバ、又はその他の遠隔ソースから送信される場合は、該同軸ケーブル、光ファイバケーブル、より対線、ＤＳＬ、又は無線技術、例えば赤外線、無線

、及びマイクロ波、は、媒体の定義の中に含まれる。しかしながら、コンピュータによって読み取り可能な記憶媒体およびデータ記憶媒体は、接続、搬送波、信号、又はその他の一時的な媒体は含まず、代わりに、非一時的な、有形の記憶媒体を対象とすることが理解されるべきである。ここにおいて用いられるときのディスク（ｄｉｓｋ及びｄｉｓｃ）は、コンパクトディスク（ＣＤ）（ｄｉｓｃ）と、レーザディスク（ｄｉｓｃ）と、光ディスク（ｄｉｓｃ）と、デジタルバーサタイルディスク（ＤＶＤ）（ｄｉｓｃ）と、フロッピー（登録商標）ディスク（ｄｉｓｋ）と、Ｂｌｕ－ｒａｙ（登録商標）ディスク（ｄｉｓｃ）と、を含み、ここで、ｄｉｓｋは、通常は磁気的にデータを複製し、ｄｉｓｃは、レーザを用いて光学的にデータを複製する。上記の組み合わせも、コンピュータによって読み取り可能な媒体の適用範囲内に含められるべきである。

【０１２９】

命令は、１つ以上のプロセッサ、例えば、１つ以上のデジタル信号プロセッサ（ＤＳＰ）、汎用マイクロプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルロジックアレイ（ＦＰＧＡ）又はその他の同等の集積回路又はディスクリート論理回路によって実行することができる。従って、ここにおいて用いられる場合の用語“プロセッサ”は、上記の構造又はここにおいて説明される技法の実装に適するあらゆるその他の構造のうちのいずれかを意味することができる。さらに、幾つかの態様では、ここにおいて説明される機能は、符号化および復号のために構成された専用のハードウェア及び／又はソフトウェアモジュール内において提供されること、又は組み合わされたコーデック内に組み入れることができる。さらに、技法は、１つ以上の回路又は論理素子内に完全に実装することが可能である。

【０１３０】

本開示の技法は、無線ハンドセット、集積回路（ＩＣ）又は一組のＩＣ（例えば、チップセット）を含む非常に様々なデバイス又は装置内に実装することができる。本開示では、開示される技法を実施するように構成されたデバイスの機能上の態様を強調するために様々なコンポーネント、モジュール、又はユニットが説明されるが、異なるハードウェアユニットによる実現は必ずしも要求しない。むしろ、上述されるように、様々なユニットは、適切なソフトウェア及び／又はファームウェアと関係させて、ハードウェアユニット内において結合させること又は上述されるように１つ以上のプロセッサを含む相互運用的なハードウェアユニットの集合によって提供することができる。

【０１３１】

様々な例が説明されている。これらの及びその他の例は、以下の請求項の範囲内である。

以下に、本願出願の当初の特許請求の範囲に記載された発明を付記する。

［Ｃ１］

異種計算のための方法であって、

プロセッサを用いて、プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信することと、

前記プロセッサを用いて、前記実行モデルの前記パイプライントポロジーがグラフィックス処理ユニット（ＧＰＵ）に実装されるプラットフォームに依存する方法を指示する命令を生成することであって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存する方法は、前記ＧＰＵのプラットフォームに基づくことと、

前記プロセッサを用いて、前記ＧＰＵに前記命令を送信することと、を備える、方法。

［Ｃ２］

命令を生成することは、前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルすることを備えるＣ１に記載の方法。

［Ｃ３］

前記コマンドリストをコンパイルすることは、

前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフ

ォームに依存する方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルすることを備えるＣ２に記載の方法。
［Ｃ４］
前記ＧＰＵの前記構成情報は、
前記ＧＰＵ内のプログラマブル計算ユニット数、及び
前記ＧＰＵ内のデータレーン数
のうちの１つ以上を備えるＣ３に記載の方法。
［Ｃ５］
前記コマンドリストをコンパイルすることは、
前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び
前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性
のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルすることを備えるＣ２に記載の方法。
［Ｃ６］
前記パイプライントポロジーを受信することは、
前記パイプライントポロジーを形成するために相互に接続された１つ以上のカーネル及び１つ以上のバッファを示すコマンドリストを受信することを備えるＣ１に記載の方法。
［Ｃ７］
装置であって、
グラフィックス処理ユニット（ＧＰＵ）と、
プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーのインディケーションを受信し、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成し、及び
前記ＧＰＵに前記命令を送信するように構成されたプロセッサと、を備え、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づく、装置。
［Ｃ８］
前記命令を生成するために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルように構成されるＣ７に記載の装置。
［Ｃ９］
前記コマンドリストをコンパイルするために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルするように構成されるＣ８に記載の装置。
［Ｃ１０］
前記ＧＰＵの前記構成情報は、
前記ＧＰＵ内のプログラマブル計算ユニット数、及び
前記ＧＰＵ内のデータレーン数
のうちの１つ以上を備えるＣ９に記載の装置。
［Ｃ１１］
前記コマンドリストをコンパイルするために、前記プロセッサは、
前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び
前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性
のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルするように構成されるＣ８に記載の装置。

［Ｃ１２］
前記パイプライントポロジーを受信するために、前記プロセッサは、
前記パイプライントポロジーを形成するために相互に接続された１つ以上のカーネル及び１つ以上のバッファを示すコマンドリストを受信するように構成されるＣ７に記載の装置。
［Ｃ１３］
前記装置は、
メディアプレーヤー、
セットトップボックス、
無線ハンドセット、
デスクトップコンピュータ
ラップトップコンピュータ
ゲームコンソール
ビデオ会議装置、及び
タブレットコンピューティングデバイス、
のうちの１つを備えるＣ７に記載の装置。
［Ｃ１４］
コンピュータによって読み取り可能な記憶媒体であって、
１つ以上のプロセッサによって実行されたときに、
プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーのインディケーションを受信し、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成し、及び
前記ＧＰＵに前記命令を送信することを前記１つ以上のプロセッサに行わせる命令が格納されており、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づく、コンピュータによって読み取り可能な記憶媒体。
［Ｃ１５］
命令を生成することを前記１つ以上のプロセッサに行わせる前記命令は、
前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備えるＣ１４に記載のコンピュータによって読み取り可能な記憶媒体。
［Ｃ１６］
前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる前記命令は、
前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装される前記プラットフォームに依存した方法を定義する前記命令を生成するために前記ＧＰＵの構成情報に少なくとも基づいて前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備えるＣ１５に記載のコンピュータによって読み取り可能な記憶媒体。
［Ｃ１７］
前記ＧＰＵの前記構成情報は、
前記ＧＰＵ内のプログラマブル計算ユニット数、及び
前記ＧＰＵ内のデータレーン数
のうちの１つ以上を備えるＣ１６に記載のコンピュータによって読み取り可能な記憶媒体。
［Ｃ１８］
前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる前記命令は、
前記実行モデルの前記パイプライントポロジーにおいて識別されたバッファのサイズ、及び

前記実行モデルの前記パイプライントポロジーにおいて識別されたカーネルの重要性のうちの１つ以上に少なくとも基づいて前記コマンドリストをコンパイルすることを前記１つ以上のプロセッサに行わせる命令を備えるＣ１５に記載のコンピュータによって読み取り可能な記憶媒体。

［Ｃ１９］

装置であって、

グラフィックス処理ユニット（ＧＰＵ）と、

プラットフォームから独立した方法でデータ処理アルゴリズムを定義する実行モデルのパイプライントポロジーを受信するための手段と、

前記実行モデルの前記パイプライントポロジーが前記ＧＰＵに実装されるプラットフォームに依存した方法を指示する命令を生成するための手段であって、前記実行モデルの前記パイプライントポロジーが前記ＧＰＵで実装される前記プラットフォームに依存した方法は、前記ＧＰＵのプラットフォームに基づく手段と、

前記ＧＰＵに前記命令を送信するための手段と、を備える、装置。

［Ｃ２０］

命令を生成するための前記手段は、

前記命令を生成するために前記実行モデルの前記パイプライントポロジーを指定するコマンドリストをコンパイルするための手段を備えるＣ１９に記載の装置。

【図１】

図1

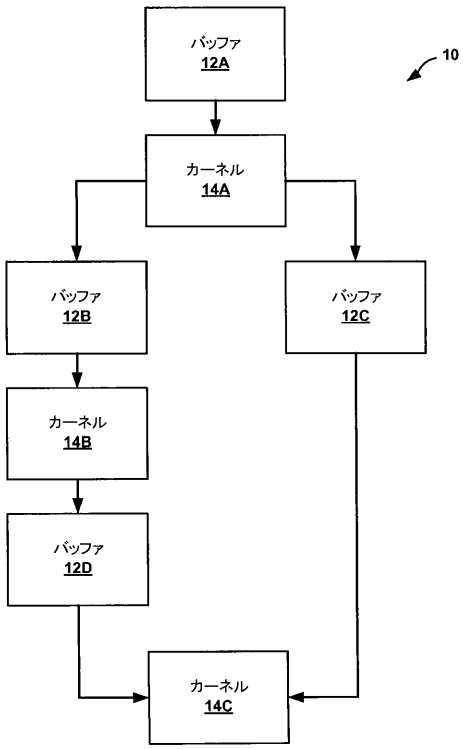

FIG. 1

【図２】

図2

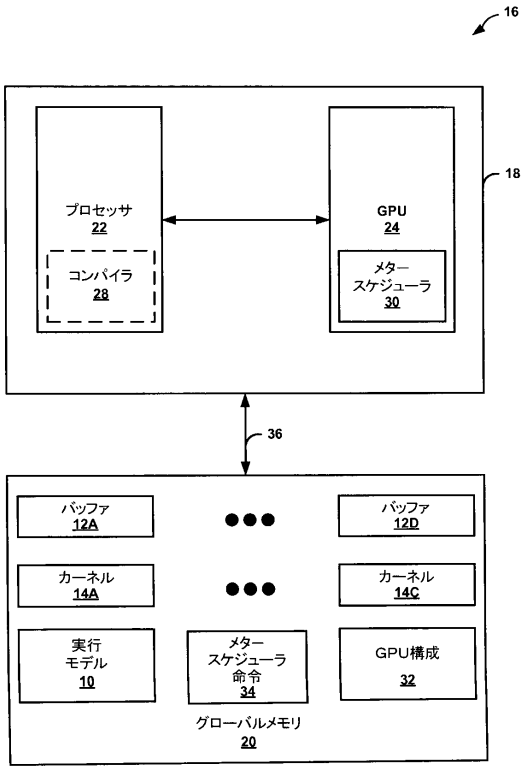

FIG. 2

【図３】

図３

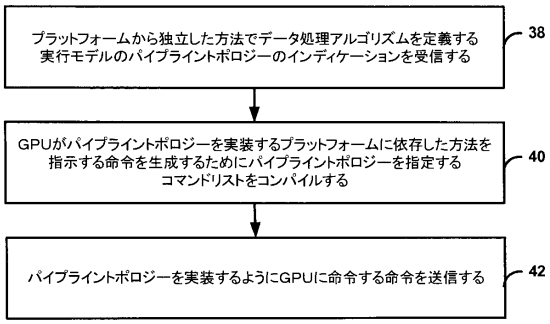

FIG. 3

【図４】

図４

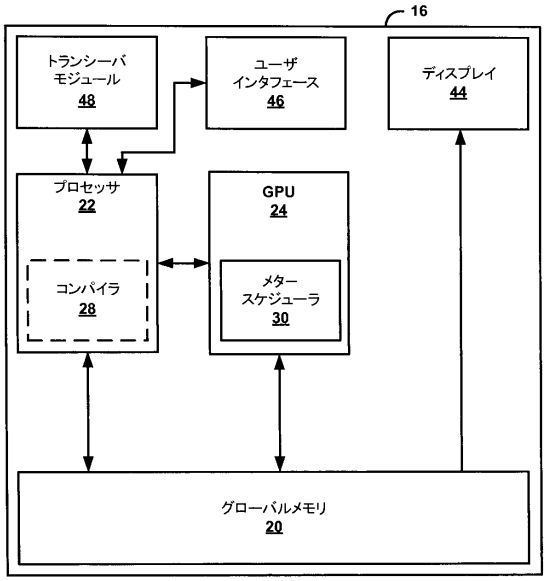

FIG. 4

フロントページの続き

(74)代理人 100153051
弁理士 河野 直樹
(74)代理人 100140176
弁理士 砂川 克
(74)代理人 100158805
弁理士 井関 守三
(74)代理人 100179062
弁理士 井上 正
(74)代理人 100124394
弁理士 佐藤 立志
(74)代理人 100112807
弁理士 岡田 貴志
(74)代理人 100111073
弁理士 堀内 美保子
(72)発明者 ボウルド、アレクセイ・ブイ.
アメリカ合衆国、カリフォルニア州 ９２１２１、サン・ディエゴ、モアハウス・ドライブ ５７７５
(72)発明者 トーゼブスキー、ウィリアム・エフ.
アメリカ合衆国、カリフォルニア州 ９２１２１、サン・ディエゴ、モアハウス・ドライブ ５７７５

審査官 坂庭 剛史

(56)参考文献 特開２０１１－２０４２０９（ＪＰ，Ａ）
特開２０１１－１７５６２４（ＪＰ，Ａ）
特開２００４－１７１２３４（ＪＰ，Ａ）
Andrei Hagiescu et al., Automated architecture-aware mapping of streaming application onto GPUs, 2011 IEEE International Parallel & Distributed Processing Symposium, 米国, IEEE, ２０１１年 5月１６日, pp.467-478, ISBN:978-1-61284-372-8
道浦 悌、大野和彦、佐々木敬泰、近藤利夫, ＧＰＧＰＵにおけるデータ転送自動化コンパイラの設計, 情報処理学会研究報告 ２０１１（平成２３）年度▲２▼［ＣＤ－ＲＯＭ］, 日本, 一般社団法人情報処理学会, ２０１１年 8月１５日, Vol.2011-HPC-130, No.17, ｐｐ. １～９, ISSN 1884-0930

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／４５
Ｇ０６Ｆ ９／３８