(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6200824号
(P6200824)

(45)発行日　平成29年9月20日(2017.9.20)　(24)登録日　平成29年9月1日(2017.9.1)

(51)Int.Cl.		F I		
G06F 15/80	(2006.01)	G06F	15/80	
G06F 9/50	(2006.01)	G06F	9/46	465E
G06F 12/0802	(2016.01)	G06F	12/08	501Z
G06F 12/08	(2016.01)	G06F	12/08	513

請求項の数 4　(全 40 頁)

(21)出願番号　特願2014-23044(P2014-23044)
(22)出願日　平成26年2月10日(2014.2.10)
(65)公開番号　特開2015-149038(P2015-149038A)
(43)公開日　平成27年8月20日(2015.8.20)
審査請求日　平成28年10月3日(2016.10.3)

(73)特許権者　302062931
ルネサスエレクトロニクス株式会社
東京都江東区豊洲三丁目２番２４号
(74)代理人　100103894
弁理士　家入　健
(72)発明者　京　昭倫
神奈川県川崎市中原区下沼部１７５３番地
ルネサスエレクトロニクス株式会社内

審査官　漆原　孝治

最終頁に続く

(54)【発明の名称】演算制御装置及び演算制御方法並びにプログラム、ＯｐｅｎＣＬデバイス

(57)【特許請求の範囲】
【請求項１】
複数の演算素子と、該複数の演算素子に対して設けられた階層の異なる複数のメモリとを有するＯｐｅｎＣＬ（Ｏｐｅｎ　Ｃｏｍｐｕｔｉｎｇ　Ｌａｎｇｕａｇｅ）デバイスの前記複数の演算素子による並列演算を制御する演算制御装置であって、
前記複数のメモリのうちの最下位階層のメモリに１つ以上格納されたデータブロックであって、前記並列演算の演算対象としてそのデータが他の階層のメモリに転送されるリードブロックと、前記並列演算後に前記他の階層のメモリから前記最下位階層のメモリに転送される１つ以上のデータブロックであって、前記１つ以上のリードブロックに対する並列演算の演算結果であるライトブロックとに対して夫々設定された属性群を取得して保持する属性群保持部と、
前記属性群保持部により保持された各前記属性群と、前記ＯｐｅｎＣＬデバイスの構成を示す構成パラメータとに基づいて、夫々の前記リードブロックと前記ライトブロックの転送方式を決定し、決定した転送方式に応じて各前記リードブロックと前記ライトブロックの転送、及び該転送に対応する前記並列演算の制御を行うシナリオ決定部とを有し、
前記属性群は、
前記転送方式を決定するために必要である一方、前記ＯｐｅｎＣＬデバイスの構成に依存しない属性を複数含むものであり、
当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示す割当属性と、

当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示す余白属性と、
当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、前記依存関係がある場合における全ての依存方向を示す依存属性とを有し、
前記ライトブロックの属性群は、該ライトブロックが既に前記他の階層のメモリに存在し、かつ、前記最下位階層のメモリに転送されると仮定して設定されたものであり、
前記シナリオ決定部は、
第１のカーネルに続いて第２のカーネルを前記ＯｐｅｎＣＬデバイスに実行させ、かつ、前記第１のカーネルに対応する並列演算の各前記ライトブロックのうちに、前記第２のカーネルに対応する並列演算の前記リードブロックとして使用される継続ライトブロックが含まれており、かつ、前記第１のカーネルに対して設定された前記継続ライトブロックの前記割当属性と、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記割当属性とが一致する場合において、該継続ライトブロックが前記最下位階層へ転送されず、前記他の階層のメモリを介して前記第２のカーネルの実行に供されることにより、前記第１のカーネルと前記第２のカーネルの実行をパイプライン化するパイプライン化制御を行い、
前記パイプライン化制御に際して、
前記第１のカーネルに対応する各前記リードブロックの夫々に対して、該リードブロックに対して設定された前記余白属性と前記依存属性に、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記余白属性と前記依存属性を夫々論理和加算する、
演算制御装置。
【請求項２】
複数の演算素子と、該複数の演算素子に対して設けられた階層の異なる複数のメモリとを有するＯｐｅｎＣＬ（Ｏｐｅｎ　Ｃｏｍｐｕｔｉｎｇ　Ｌａｎｇｕａｇｅ）デバイスの前記複数の演算素子による並列演算を制御する演算制御方法であって、
前記複数のメモリのうちの最下位階層のメモリに１つ以上格納されたデータブロックであって、前記並列演算の演算対象としてそのデータが他の階層のメモリに転送されるリードブロックと、前記並列演算後に前記他の階層のメモリから前記最下位階層のメモリに転送される１つ以上のデータブロックであって、前記１つ以上のリードブロックに対する並列演算の演算結果であるライトブロックとに対して夫々設定された属性群を取得して保持する第１のステップと、
保持された各前記属性群と、前記ＯｐｅｎＣＬデバイスの構成を示す構成パラメータとに基づいて、夫々の前記リードブロックと前記ライトブロックの転送方式を決定し、決定した転送方式に応じて各前記リードブロックと前記ライトブロックの転送、及び該転送に対応する前記並列演算の制御を行う第２のステップとを有し、
前記属性群は、
前記転送方式を決定するために必要である一方、前記ＯｐｅｎＣＬデバイスの構成に依存しない属性を複数含むものであり、
当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示す割当属性と、
当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示す余白属性と、
当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、前記依存関係がある場合における全ての依存方向を示す依存属性とを有し、
前記ライトブロックの属性群は、該ライトブロックが既に前記他の階層のメモリに存在し、かつ、前記最下位階層のメモリに転送されると仮定して設定されたものであり、
前記第２のステップにおいて、

第１のカーネルに続いて第２のカーネルを前記ＯｐｅｎＣＬデバイスに実行させ、かつ、前記第１のカーネルに対応する並列演算の各前記ライトブロックのうちに、前記第２のカーネルに対応する並列演算の前記リードブロックとして使用される継続ライトブロックが含まれており、かつ、前記第１のカーネルに対して設定された前記継続ライトブロックの前記割当属性と、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記割当属性とが一致する場合において、該継続ライトブロックが前記最下位階層へ転送されず、前記他の階層のメモリを介して前記第２のカーネルの実行に供されることにより、前記第１のカーネルと前記第２のカーネルの実行をパイプライン化するパイプライン化制御を行い、

前記パイプライン化制御に際して、

前記第１のカーネルに対応する各前記リードブロックの夫々に対して、該リードブロックに対して設定された前記余白属性と前記依存属性に、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記余白属性と前記依存属性を夫々論理和加算する、

演算制御方法。

【請求項３】

複数の演算素子と、該複数の演算素子に対して設けられた階層の異なる複数のメモリとを有するＯｐｅｎＣＬ（Ｏｐｅｎ　Ｃｏｍｐｕｔｉｎｇ　Ｌａｎｇｕａｇｅ）デバイスの前記複数の演算素子による並列演算を制御する際に、

前記複数のメモリのうちの最下位階層のメモリに１つ以上格納されたデータブロックであって、前記並列演算の演算対象としてそのデータが他の階層のメモリに転送されるリードブロックと、前記並列演算後に前記他の階層のメモリから前記最下位階層のメモリに転送される１つ以上のデータブロックであって、前記１つ以上のリードブロックに対する並列演算の演算結果であるライトブロックとに対して夫々設定された属性群を取得して保持する第１のステップと、

保持された各前記属性群と、前記ＯｐｅｎＣＬデバイスの構成を示す構成パラメータとに基づいて、夫々の前記リードブロックと前記ライトブロックの転送方式を決定し、決定した転送方式に応じて各前記リードブロックと前記ライトブロックの転送、及び該転送に対応する前記並列演算の制御を行う第２のステップとをコンピュータに実行させ、

前記属性群は、

前記転送方式を決定するために必要である一方、前記ＯｐｅｎＣＬデバイスの構成に依存しない属性を複数含むものであり、

当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示す割当属性と、

当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示す余白属性と、

当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、前記依存関係がある場合における全ての依存方向を示す依存属性とを有し、

前記ライトブロックの属性群は、該ライトブロックが既に前記他の階層のメモリに存在し、かつ、前記最下位階層のメモリに転送されると仮定して設定されたものであり、

前記第２のステップにおいて、

第１のカーネルに続いて第２のカーネルを前記ＯｐｅｎＣＬデバイスに実行させ、かつ、前記第１のカーネルに対応する並列演算の各前記ライトブロックのうちに、前記第２のカーネルに対応する並列演算の前記リードブロックとして使用される継続ライトブロックが含まれており、かつ、前記第１のカーネルに対して設定された前記継続ライトブロックの前記割当属性と、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記割当属性とが一致する場合において、該継続ライトブロックが前記最下位階層へ転送されず、前記他の階層のメモリを介して前記第２のカーネルの実行に供されることにより、前記第１のカーネルと前記第２のカーネルの実行をパイ

プライン化するパイプライン化制御を行い、
前記パイプライン化制御に際して、
前記第１のカーネルに対応する各前記リードブロックの夫々に対して、該リードブロックに対して設定された前記余白属性と前記依存属性に、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記余白属性と前記依存属性を夫々論理和加算する、
プログラム。
【請求項４】
複数の演算素子と、
該複数の演算素子に対して設けられた階層の異なる複数のメモリと、
前記複数の演算素子による並列演算を制御する演算制御部とを備えるＯｐｅｎＣＬデバイスであって、
前記演算制御部は、
前記複数のメモリのうちの最下位階層のメモリに１つ以上格納されたデータブロックであって、前記並列演算の演算対象としてそのデータが他の階層のメモリに転送されるリードブロックと、前記並列演算後に前記他の階層のメモリから前記最下位階層のメモリに転送される１つ以上のデータブロックであって、前記１つ以上のリードブロックに対する並列演算の演算結果であるライトブロックとに対して夫々設定された属性群を取得して保持する属性群保持部と、
前記属性群保持部により保持された各前記属性群と、前記ＯｐｅｎＣＬデバイスの構成を示す構成パラメータとに基づいて、夫々の前記リードブロックと前記ライトブロックの転送方式を決定し、決定した転送方式に応じて各前記リードブロックと前記ライトブロックの転送、及び該転送に対応する前記並列演算の制御を行うシナリオ決定部とを有し、
前記属性群は、
前記転送方式を決定するために必要である一方、前記ＯｐｅｎＣＬデバイスの構成に依存しない属性を複数含むものであり、
当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示す割当属性と、
当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示す余白属性と、
当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、前記依存関係がある場合における全ての依存方向を示す依存属性とを有し、
前記ライトブロックの属性群は、該ライトブロックが既に前記他の階層のメモリに存在し、かつ、前記最下位階層のメモリに転送されると仮定して設定されたものであり、
前記シナリオ決定部は、
第１のカーネルに続いて第２のカーネルを前記ＯｐｅｎＣＬデバイスに実行させ、かつ、前記第１のカーネルに対応する並列演算の各前記ライトブロックのうちに、前記第２のカーネルに対応する並列演算の前記リードブロックとして使用される継続ライトブロックが含まれており、かつ、前記第１のカーネルに対して設定された前記継続ライトブロックの前記割当属性と、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記割当属性とが一致する場合において、該継続ライトブロックが前記最下位階層へ転送されず、前記他の階層のメモリを介して前記第２のカーネルの実行に供されることにより、前記第１のカーネルと前記第２のカーネルの実行をパイプライン化するパイプライン化制御を行い、
前記パイプライン化制御に際して、
前記第１のカーネルに対応する各前記リードブロックの夫々に対して、該リードブロックに対して設定された前記余白属性と前記依存属性に、前記第２のカーネルに対して設定された、前記継続ライトブロックに対応する前記リードブロックの前記余白属性と前記依存属性を夫々論理和加算する、

ＯｐｅｎＣＬデバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は並列プロセッサ、具体的にはＯｐｅｎＣＬデバイスの演算制御技術に関する。
【背景技術】
【０００２】
近年、プロセッサの発熱を抑制する必要性から、プロセッサの動作周波数を上げる代わりに、並列に処理を行うプロセッサ・コア（以下、単に「コア」という）の数を増やすことで性能向上を実現する動きが顕著になっている。複数のコアを有するプロセッサは、マルチコアプロセッサと呼ばれており、マルチコアプロセッサの中の、特にコア数が多いものは、メニコアプロセッサと呼ばれる。本明細書の中で、マルチコアプロセッサとメニコアプロセッサの区別を特にせず、並列に処理を行うコアが複数含まれるプロセッサを概して「並列プロセッサ」という。
【０００３】
並列プロセッサは、アクセラレータとして様々な分野で用いられている。しかし、メーカ、分野などによって種々のアクセラレータが製造され、アクセラレータ用の言語やフレームワークも様々開発されているため、アクセラレータ間でプログラムコードの移植が困難である。
【０００４】
この問題を解決するため、並列プロセッサのための標準的なフレームワークとして、ＯｐｅｎＣＬ（Ｏｐｅｎ　Ｃｏｍｐｕｔｉｎｇ　Ｌａｎｇｕａｇｅ）が定められた（非特許文献１）。ここで、ＯｐｅｎＣＬの概要を説明する。
【０００５】
図１７は、非特許文献１におけるＦｉｇ３．１に対して符号を追加したものであり、典型的なＯｐｅｎＣＬシステムのプラットフォームモデルを示す。
【０００６】
図１７に示すように、ＯｐｅｎＣＬシステム１０は、Ｈｏｓｔ（ホスト）１２、１つ以上のＣｏｍｐｕｔｅ　Ｄｅｖｉｃｅ（以下「ＯｐｅｎＣＬデバイス」という）１４を備える。ＯｐｅｎＣＬデバイス１４は、上述したアクセラレータに該当する。
【０００７】
夫々のＯｐｅｎＣＬデバイス１４は、１つ以上のＣｏｍｐｕｔｅ　Ｕｎｉｔ（以下「ＣＵ」と略す）１６を有し、夫々のＣＵ１６は、１つ以上の演算素子（Ｐｒｏｃｅｓｓｉｎｇ　Ｅｌｅｍｅｎｔ。以下「ＰＥ」と略す）１８を有する。なお、演算素子ＰＥ１８は、前述したコアに該当する。
【０００８】
ＯｐｅｎＣＬのアプリケーションは、ホスト１２側で動作するプログラムコードと、ＯｐｅｎＣＬデバイス１４、すなわちアクセラレータ側で動作するプログラムコードとで構成される。ホスト１２側で動作するプログラムコードは「ホストコード」と呼ばれ、ＯｐｅｎＣＬデバイス１４側で動作するプログラムコードは「カーネル」と呼ばれる。
【０００９】
ホスト１２は、ＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ　Ｐｒｏｇｒａｍ　Ｉｎｔｅｒｆａｃｅ）をコールして演算を指示する。ＯｐｅｎＣＬデバイス１４は、指示された演算を実行する。ホスト１２は、リソースを管理するコンテキストの生成、さらに、ＯｐｅｎＣＬを通じてデバイス動作を調停するためのコマンドキューの生成を行う。「デバイス動作」は、演算を実行すること、メモリを操作すること、同期をとることなどが含まれる。
【００１０】
ＯｐｅｎＣＬにおいて、カーネルは、ｗｏｒｋ－ｉｔｅｍ（以下、略して「アイテム」とも呼ぶ）として、Ｎ次元（１≦Ｎ≦３）のインデックス空間（Ｉｎｄｅｘ　Ｓｐａｃｅ）で実行される。例えば、２次元のインデックス空間として（４，６）が指定されれば、

「４×６」の計２４個のアイテムが実行される。
【００１１】
１アイテムを実行するには、１ＰＥが利用される。したがって、並列に実行されるアイテムの数と、実在するＰＥの数とが同一の場合、カーネルは、４列６行計２４個のＰＥ上で実行されることになる。
【００１２】
なお、並列に実行されるアイテム数に対して実際に存在するＰＥ数が少ない場合には、実際に存在するＰＥ上でアイテムの並列実行を繰り返すことが行われる。例えば、２列３行計６個のＰＥしかないときに、上述した（４，６）のインデックス空間が指定された場合には、該６個のＰＥにより、６個のアイテムの並列実行を４回繰り返す必要がある。
【００１３】
また、ＯｐｅｎＣＬでは、ｗｏｒｋ－ｇｒｏｕｐ（ワークグループ）という概念も導入されている。ワークグループは、同一のＣＵ１６上で実行され、かつ、互いに関連するアイテムの集合である。同一のワークグループ内の各アイテムは、同一のカーネルを実行し、該ＣＵ１６の後述するローカルメモリをシェアする。
【００１４】
各ワークグループは、ユニックなグループＩＤが割り当てられ、各ワークグループ内のアイテムは、該ワークグループ内においてユニックなローカルＩＤが割り当てられる。また、アイテムは、ユニックなグローバルＩＤも割り当てられる。アイテムは、グローバルＩＤ、または、グループＩＤとローカルＩＤの組合せによって識別できる。
【００１５】
ＯｐｅｎＣＬデバイス１４に演算処理を行わせるプロセスは、下記のステップ順でＡＰＩをコールすることにより構成される。
【００１６】
＜ステップ１＞：演算処理の参照用のデータ（以下「参照データ」という）やカーネルをホスト１２からＯｐｅｎＣＬデバイス１４へ転送する。
【００１７】
＜ステップ２＞：「カーネル起動コマンド」によりＯｐｅｎＣＬデバイス１４上でカーネルの実行を開始させる。
【００１８】
＜ステップ３＞：ＯｐｅｎＣＬデバイス１４におけるカーネルの実行完了後、ＯｐｅｎＣＬデバイス１４のメモリ空間から演算処理の結果データをホスト１２側に転送する。
【００１９】
図１８を参照して、メモリ空間を含むＯｐｅｎＣＬデバイス１４の構成を説明する。
図１８は、非特許文献１におけるＦｉｇ３．３に対して符号を追加したものである。前述したように、ＯｐｅｎＣＬデバイス１４は１つ以上のＣＵ１６を備え、夫々のＣＵ１６は１つ以上のＰＥ１８を有する。
【００２０】
上述したステップ２におけるカーネルの実行に際して、ＯｐｅｎＣＬデバイス１４では、４つの異なるメモリへのアクセスが生じ得る。この４つのメモリは、プライベートメモリ２０、ローカルメモリ２２、グローバルメモリ３２、コンスタントメモリ３４である。この４つのメモリについて、まず、図１９を参照して、アイテムとワークグループの視点から説明する。なお、図１９は、非特許文献１におけるＴａｂｌｅ３．１である。
【００２１】
プライベートメモリ２０は、１つのアイテムに対応し、該アイテムの実行にのみ用いられる。１つのアイテムに対応するプライベートメモリ２０に対して定義された変数は、他のアイテムに対して使用できない。
【００２２】
ローカルメモリ２２は、１つのグループに対応し、該グループ内の各アイテムによりシェアすることができる。そのため、ローカルメモリ２２の用途としては、例えば、該グル

ープ内の各アイテムによりシェアされる変数をローカルメモリ２２に割り当てることが挙げられる。

【００２３】

グローバルメモリ３２とコンスタントメモリ３４は、全てのグループの全てのアイテムからアクセスできる。なお、グローバルメモリ３２は、アイテムからリードとライトのいずれからもアクセスできるが、コンスタントメモリ３４は、アイテムからリードアクセスのみができる。以下、グローバルメモリ３２とコンスタントメモリ３４をまとめてデバイスメモリ３０という。

【００２４】

アイテムとＰＥ１８の１対１の対応関係から、上記４つのメモリと、ＣＵ１６及びＰＥ１８との対応関係は、以下のようになる。

【００２５】

プライベートメモリ２０は、ＰＥ１８と１対１で対応し、相対応するＰＥ１８によりのみアクセス可能である。

【００２６】

ローカルメモリ２２は、ＣＵ１６と１対１で対応し、相対応するＣＵ１６内の全てのＰＥ１８によりアクセス可能である。

【００２７】

デバイスメモリ３０は、全てのＣＵ１６内の全てのＰＥ１８、すなわちＯｐｅｎＣＬデバイス１４内の全てのＰＥによりアクセス可能である。

【００２８】

また、ＯｐｅｎＣＬデバイス１４によっては、デバイスメモリ３０のキャッシュメモリとして機能するキャッシュ２４がさらに設けられる場合もある。

【００２９】

このように、ＯｐｅｎＣＬデバイス１４には、階層の異なる複数のメモリが設けられている。これらのメモリは、上位階層にあるほど、ＰＥからのアクセスが高速にできる。デバイスメモリ３０（最下位）、ローカルメモリ２２（中位）、プライベートメモリ２０（最上位）の順で階層が上位になり、同様の順でＰＥからのアクセスもより高速になる。

【００３０】

ＯｐｅｎＣＬデバイス１４の性能を充分に引き出すためには、例えば、利用頻度の高いデータを、なるべくより高速なメモリ空間に移動してから参照するようにするなど、デバイスメモリ３０と、プライベートメモリ２０／ローカルメモリ２２との間でのデータ移動について工夫する必要がある。

【００３１】

制御方式がＯｐｅｎＣＬデバイスと異なる逐次プロセッサの場合においても、グローバルメモリ空間とプライベートメモリ空間との間でのデータ移動が行われる。図２０に示す逐次プロセッサの例を参照して説明する。

【００３２】

図２０に示す逐次プロセッサ２０は、演算素子であるＰＥ５２、プライベートメモリ５４、グローバルメモリ５６、キャッシュ制御機構５８を有する。

【００３３】

図示のように、逐次プロセッサ５０の記憶装置は、プライベートメモリ５４とグローバルメモリ５６に分けられている。プライベートメモリ５４は、物理的にオンチップの小容量メモリであり、グローバルメモリ５６は、物理的にオフチップの大容量のメモリである。

【００３４】

逐次プロセッサ５０では、記憶装置がプライベートメモリ５４とグローバルメモリ５６に分けられているが、プライベートメモリ５４とグローバルメモリ５６の間に設けられたキャッシュ制御機構５８により、プライベートメモリ５４とグローバルメモリ５６間のデータ移動は自動的に行われ、逐次プロセッサ５０のユーザは、１つの大きなメモリ空間し

か見えない。つまり、逐次プロセッサ５０のユーザは、グローバルメモリ５６とプライベートメモリ５４との間でどのようにデータを移動するかを意図しなくても、ＰＥ５２に演算処理を行わせるユーザプログラムを容易に開発できる。

【００３５】

ところで、並列プロセッサ、特に図１８に示すＯｐｅｎＣＬデバイス１４のように多数のコア（ＰＥ）を搭載した場合には、コアの数と同数のプライベートメモリ２０が存在し、さらに、ＣＵ１６の数と同数のローカルメモリ２２が存在する。これらのメモリを全て１つのキャッシュ制御機構で統一的に管理するのは、ハードウェアのコストが高く、一般的には実現困難である。

【００３６】

他方、キャッシュ制御機構が無いと、ＯｐｅｎＣＬシステム１０のユーザ（以下、単に「ユーザ」と呼ぶ）には、複数のメモリ空間が見えてしまう。前述したように、利用頻度の高いデータを、なるべくより高速なメモリ空間（すなわちより上位階層のメモリ空間）に移動してから参照するようにするなど、より良い性能を追求するためには、演算処理に伴う階層の異なるメモリ間でのデータ移動について、ユーザプログラムで明示的に指示するする必要がある。これを正しく実現するためには、ユーザには、上述した各メモリ同士間の速度差、容量差、機能差等に関する知識を持つ必要がある。図２１を参照して具体例を説明する。

【００３７】

図２１は、複数のデータブロック（データブロックＡ～Ｄ）から、データブロックＡ’～Ｂ’を得る演算処理を実行する場合を説明するための図である。なお、図２１において、ホストからデバイスへのカーネル転送の図示を省略している。また、データブロックＡ～Ｄは、上述したステップ１でホスト１２からＯｐｅｎＣＬデバイス１４に転送された参照データであり、グローバルメモリ３２に格納される。データブロックＡ’～Ｂ’は、上述したステップ２にいてデータブロックＡ～Ｄに対して行われた演算の結果であり、グローバルメモリ３２に書き込まれ、後に、上述したステップ３でホスト１２に転送される。

【００３８】

ここで、ステップ２の処理、すなわちカーネルを実行する演算処理を説明する。なお、本明細書において、以下、プライベートメモリについて、複数が有り得る場合には、「プライベートメモリ群」という。

【００３９】

演算処理の性能を追求しなければ、演算において、プライベートメモリ群／ローカルメモリ２２を使わずに、グローバルメモリ３２のみを使用する手法が考えられる。この場合、グローバルメモリ３２と、プライベートメモリ群／ローカルメモリ２２間のデータ転送が無い。

【００４０】

この手法は、制御が単純であるが、性能が良くない。演算処理をより良い性能で行うために、上述したように、演算対象のデータをグローバルメモリ３２からプライベートメモリ群／ローカルメモリ２２に転送してから演算を行い、演算の結果をプライベートメモリ群／ローカルメモリ２２に格納してからグローバルメモリ３２に転送する手法が用いられる。

【００４１】

この手法を用いる場合について、まず、全てのアイテムが同時に並行して実行可能なときの手順（ステップＡ～Ｃ）を説明する。なお、「全てのアイテムが同時に並行して実行可能」とは、ＰＥ数が総アイテム数以上であり、かつ、プライベートメモリ群とローカルメモリの容量は、演算対象の全てのデータを格納可能できることなどを意味し、この場合、演算対象のデータをグローバルメモリ３２からプライベートメモリ群／ローカルメモリ２２への転送、各ＰＥ１８による演算の並列実行、演算結果をプライベートメモリ群／ローカルメモリ２２からグローバルメモリ３２への転送が一度しか行われない。

【００４２】

＜ステップＡ＞：グローバルメモリ３２に格納されたデータブロックＡ～Ｄをプライベートメモリ群／ローカルメモリ２２に転送する。

【００４３】

この転送は、例えば、演算対象のデータのうちの、ＰＥ１８によりのみ使用されるデータをＰＥ１８のプライベートメモリに転送し、複数のＰＥ１８により共有されるデータをローカルメモリ２２に転送することである。

【００４４】

なお、以下において、グローバルメモリ３２からプライベートメモリ群／ローカルメモリ２２へのデータ転送を「リード転送」という。また、データブロックＡ～Ｄのような、リード転送されるデータブロックを「リードブロックＲＢ」という。

【００４５】

＜ステップＢ＞：各ＰＥ１８上で演算処理を実行し、演算処理の結果を該ＰＥ１８がアクセスできるプライベートメモリ／ローカルメモリ２２に格納する。

【００４６】

＜ステップＣ＞：ステップＢの演算処理により得られ、プライベートメモリ群／ローカルメモリ２２に格納されたデータブロックＡ’～Ｂ’をグローバルメモリ３２に転送する。

【００４７】

なお、以下において、プライベートメモリ群／ローカルメモリ２２からグローバルメモリ３２へのデータ転送を「ライト転送」という。また、データブロックＡ’～Ｂ’のような、プライベートメモリ群／ローカルメモリ２２に格納され、ライト転送されるデータブロックを「ライトブロックＷＢ」という。

【００４８】

該３つのステップの全てについて、ユーザにより作成されたカーネルの中で明示的に指定される必要がある。これらの指定は、演算処理の内容や、ＯｐｅｎＣＬデバイス１４の構成（ＰＥ数（＝プライベートメモリ数）や、個々のプライベートメモリの容量、ローカルメモリの容量など）に依存する内容が含まれる。

【００４９】

例えば、演算対象となるリードブロックＲＢが複数あり、かつ全てが一度に一つのワークグループ内のプライベートメモリ群／ローカルメモリ２２に入りきらないため、それぞれのリードブロックＲＢをサブブロックに分割しなければならない場合に、該複数のリードブロックＲＢに対して、ステップＡにおいて、サブブロック間の対応付け方法を指定する必要がある。リードブロックＲＢのサブブロック間の「対応付け方法」は、上記複数のリードブロックＲＢのサブブロック同士で、どのリードブロックＲＢのサブブロック同士を、同一のワークグループ内のプライベートメモリ群、または同一ワークグループ内のローカルメモリ２２に転送するかを意味する。これは、演算処理の内容、またどのように分割すべきかは、ＯｐｅｎＣＬデバイス１４の構成に依存する。

【００５０】

同様に、演算結果として複数のライトブロックＷＢがある場合には、該複数のライトブロックＷＢのそれぞれのサブブロックが、どのようなリードブロックＲＢのサブブロック同士の組合せの下で、演算結果として求められることになるか、という意味での対応付け方法も指定する必要がある。なお、ライトブロックＷＢの各サブブロックの内容とは、すなわち各ワークグループのプライベートメモリ群またはローカルメモリ２２に演算結果として格納されたデータである。そしてライトブロックＷＢをローバルメモリ３２へ転送するとは、該データをグローバルメモリ３２内のライトブロックＷＢの各サブブロック位置に書き込むことを意味する。リードブロックＲＢの対応付け方法と同様に、ライトブロックＷＢの対応付け方法も、演算処理の内容と、ＯｐｅｎＣＬデバイス１４の構成に依存する。

【００５１】

上記のように所要のデータブロック全体がワークグループ内のメモリに入りきらない場合以外、ＰＥの総数が、インデックス空間のサイズより小さいなどのときにおいても、全

てのアイテムが同時に並行して実行することができないため、ＰＥによるアイテムの並列実行を複数回繰り返す必要がある。当然ながら、並列実行の繰り返しに合わせてリード転送とライト転送も繰り返す必要がある。この場合、演算処理の内容とＯｐｅｎＣＬデバイス１４の構成に応じて、データブロックの分割方法、データブロックを分割して得たサブブロック間の対応付け方法を指定する必要がある。

【００５２】

データブロックの「分割方法」は、該データブロックをどのようにしてサブブロックに分割するかを意味する。「サブブロックＳＢ」は、リード転送とライト転送の転送単位である。以下において、リードとライトを区別する必要がある場合については、リードブロックＲＢを分割して得たサブブロックを「サブリードブロックＳＲＢ」といい、ライトブロックＷＢを分割して得たサブブロック「サブライトブロックＳＷＢ」という。

【００５３】

サブブロックＳＢ間の「対応付け方法」とは、異なるリードブロックあるいはライトブロックに夫々含まれるどのサブブロックＳＢ同士を、同時に同一のプライベートメモリ群、または同一のローカルメモリ２２上に存在させるかを意味する。

【００５４】

データブロックの分割方法はＯｐｅｎＣＬデバイス１４の構成に依存し、一方、サブブロックの対応付け方法は演算処理の内容に依存する。分割が必要になると、データブロックを分割しない場合より、さらに指定が複雑である。

【００５５】

図２２は、ＯｐｅｎＣＬデバイス１４に演算処理を行わせるために、ユーザにより指定する必要のある内容をまとめて示す。

【００５６】

図示のように、第１の部分は、リード転送用の指定であり、演算処理の内容とＯｐｅｎＣＬデバイス１４の構成に依存する部分が含まれる。

【００５７】

演算処理の内容とＯｐｅｎＣＬデバイス１４の構成に依存する部分は、例えば、リードブロックＲＢを分割するか否かの指定（例１）や、分割する場合における分割方法の指定（例２）とサブリードブロックＳＲＢ間の対応付け方法の指定（例３）がある。

【００５８】

第２の部分は、リードブロックＲＢまたはサブリードブロックＳＲＢに対する演算処理の指定である。演算処理を指定するための内容であるため、当然演算処理の内容に依存する。さらに、この部分は、リード転送用の指示に合わせる必要があるため、アイテムの並行実行の回数の指定（例４）など、ＯｐｅｎＣＬデバイス１４の構成に依存する内容が含まれる。

【００５９】

第３の部分は、ライト転送用の指定であり、リード転送用の指示に合わせる必要があるため、必然的に、演算処理の内容とＯｐｅｎＣＬデバイス１４の構成に依存する部分（例５）が含まれる。

【００６０】

このように、より良い性能を追求するために、ユーザは、演算処理の内容と、ＯｐｅｎＣＬデバイス１４の構成に合わせてカーネル（ユーザコード）を開発する必要がある。

【００６１】

しかし、ＯｐｅｎＣＬに準拠したデバイスであっても、メーカが異なれば、各メモリ空間の容量、アクセス速度、アクセス遅延、キャッシュ制御の有無などの点において、千差万別である。そのため、あるＯｐｅｎＣＬデバイスにとって、階層の異なるメモリ間のデータ移動が理想的に開発されたユーザコードであっても、他のＯｐｅｎＣＬデバイスや、同一シリーズであるものの世代が違うＯｐｅｎＣＬデバイスなどにとっては、反って性能劣化を引き起こすものになってしまう可能性がある。つまり、ユーザコードの性能上の可搬性が低い。

【００６２】
ある程度の性能可搬性を実現するために、特定のＯｐｅｎＣＬデバイスを対象にユーザコードを開発するのではなく、なるべく、既存の多くの種類のＯｐｅｎＣＬデバイスの構成を念頭に入れつつ、ユーザコードを作成することが考えられる。しかし、そうした作業は、演算処理の設計者からみれば本質的でない作業のために多大な負担に感じる上に、コードの可読性の低下、そして煩雑度の増大等を引き起こす要因にもなる。
【００６３】
特許文献１には、ＯｐｅｎＣＬデバイスに対して、演算処理に伴う階層の異なる複数のメモリ間でのデータ移動について、ユーザコードの開発者の負担を減らすと共に、ユーザコードの可搬性を高める技術を開示している。
【００６４】
該技術では、ユーザコードの開発者は、カーネルの引数として、ＯｐｅｎＣＬデバイスの演算対象及び演算結果の各データブロック（リードブロックとライトブロック）の夫々について複数の属性を含む属性群を設定し、ＯｐｅｎＣＬデバイスの演算制御部は、ホストから転送されたカーネルの引数が示す演算対象及び演算結果の各データブロックの属性群と、ＯｐｅｎＣＬデバイスの構成を示すパラメータとに基づいて、自動的に転送方式を決定すると共に、決定した転送方式でデータの転送と、ＯｐｅｎＣＬデバイスによる演算を制御する。
【００６５】
なお、転送方式は、主に、デバイスメモリと、ローカルメモリ及びプライベートメモリとの間でどのようにしてデータブロックを転送するかに関するものである。
【００６６】
図２３は、特許文献１の技術を適用したＯｐｅｎＣＬシステムにおいて、ユーザがカーネルを開発する際に指定する必要のある内容を示す。図示のように、これらの内容は、属性群の指定とユーザ処理の指定のみであり、いずれも、デバイスの構成に依存しないものである。
【００６７】
図２３に示すように、属性群は、固有属性、演算属性、ポリシ属性のグループに分けられており、各グループの属性は、転送方式を決定するために必要である一方、ＯｐｅｎＣＬデバイスの構成に依存しないものである。例えば、ポリシ属性には、当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示す割当属性や、当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示す余白属性や、当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、依存関係がある場合における全ての依存方向を示す依存属性が含まれる。
【００６８】
なお、ライトブロックの属性群は、該ライトブロックが既にローカルメモリまたはプライベートメモリに存在し、かつ、デバイスメモリに転送されると仮定して設定されたものである。
【００６９】
特許文献１に開示された技術によれば、ユーザコード（カーネル）の開発者は、ＯｐｅｎＣＬデバイスの構成を知らなくても、カーネルが使用するデータブロック毎に属性群さえ指定すれば、高性能なカーネルを可搬性の高い形で実現できる。それは、ＯｐｅｎＣＬデバイスの演算制御部は、カーネルが参照する各データブロックの属性群と、ＯｐｅｎＣＬデバイスの構成とに基づいて、データブロックを該デバイスにとって最適なサイズに適宜に分割して、「リード転送→カーネルによる演算処理→ライト転送」をボディとする反復処理を、分割数に等しい回数だけ繰り返すよう制御可能である。デバイス側の演算制御部の設計については、ＯｐｅｎＣＬデバイスの構成を熟知した専門家、例えば該ＯｐｅｎＣＬデバイスのメーカの開発者により行えばよい。

【先行技術文献】
【特許文献】
【００７０】
【特許文献１】特開２０１３－０２５５４７号公報
【非特許文献】
【００７１】
【非特許文献１】Ｔｈｅ　ＯｐｅｎＣＬ　Ｓｐｅｃｉｃａｔｉｏｎ，Ｖｅｒ：１．０，Ｄｏｃｕｍｅｎｔ　Ｒｅｖｉｓｉｏｎ：４３，Ｋｈｒｏｎｏｓ　ＯｐｅｎＣＬ　Ｗｏｒｋｉｎｇ　Ｇｒｏｕｐ（２００９）
【発明の概要】
【発明が解決しようとする課題】
【００７２】
ここで、連続して実行される２つのカーネルの間に関連性がある場合を考える。例えば、第１のカーネルＸは、先に実行され、その演算結果のデータブロックが、後に実行される第２のカーネルＹの入力データとして使用されるとする。なお、各データブロックは、サブブロックに分割して転送されるとする。
【００７３】
特許文献１のＯｐｅｎＣＬシステムの場合、下記の流れになると考える。
＜ステップＸ１＞
ホーストは、第１のカーネルＸをデバイスに転送する。
該第１のカーネルＸの引数には、第１のカーネルＸによる演算処理に使用される各データブロック（リードブロック）毎に、上述した属性群が含まれている。
【００７４】
＜ステップＸ２＞
デバイスの演算制御部は、第１のカーネルＸの引数に基づいて、グローバルメモリに格納された各リードブロック、及び演算結果となるライトブロックの分割方式を含む転送方式を決定する。そして、各リードブロックをサブリードブロックに分割して、ローカルメモリまたはプライベートメモリへ転送すると共に、ＰＥに演算を行わせる。
【００７５】
＜ステップＸ３＞
デバイスの各ＰＥは、演算制御部の制御に従って演算を行い、サブライトブロックを得る。これらのサブライトブロックは、ローカルメモリまたはプライベートメモリに格納された後、演算制御部の制御に従って、ＰＥからグローバルメモリに転送される。
【００７６】
＜ステップ４＞
全ての処理が終わるまで、演算制御部の制御の元で、ステップＸ２～ステップＸ３の処理が繰り返される。最後に、第１のカーネルＸの演算結果となるデータブロック（ライトブロック）が得られ、グローバルメモリに格納される。
【００７７】
第１のカーネルＸの処理が完了すると、下記の流れで第２のカーネルＹの処理が行われる。
【００７８】
＜ステップＹ１＞
ホーストは、第２のカーネルＹをデバイスに転送する。
該第２のカーネルＹの引数には、第２のカーネルＹによる演算処理に使用される各データブロック（リードブロック）毎に、上述した属性群が含まれている。
【００７９】
第１のカーネルＸのライトブロックのうちのあるライトブロックが、第２のカーネルＹの演算処理に使用される場合に、第２のカーネルＹの引数には、該ライトブロックに対応する、第２のカーネルＹのリードブロックについても、属性群が指定されている。以下に

おいて、先に実行されるカーネルのライトブロックであって、後に実行されるカーネルのリードブロックになるデータブロックを「継続ライトブロック」といい、後に実行されるカーネルのリードブロックであって、先に実行されるカーネルのライトブロックになるデータブロックを「継続リードブロック」という。

【００８０】

＜ステップＹ２＞

デバイスの演算制御部は、第２のカーネルＹの引数に基づいて、グローバルメモリに格納された各リードブロック（継続リードブロックを含む）、及び演算結果となるライトブロックの分割方式を含む転送方式を決定する。そして、各リードブロックをサブリードブロックに分割して、ローカルメモリまたはプライベートメモリへ転送すると共に、ＰＥに演算を行わせる。

【００８１】

＜ステップＹ３＞

デバイスの各ＰＥは、演算制御部の制御に従って演算を行い、サブライトブロックを得る。これらのサブライトブロックは、ローカルメモリまたはプライベートメモリに格納された後、演算制御部の制御に従って、ＰＥからグローバルメモリに転送される。

【００８２】

＜ステップ４＞

全ての処理が終わるまで、演算制御部の制御の元で、ステップＹ２～ステップＹ３の処理が繰り返される。最後に、第２のカーネルＹの演算結果となるデータブロック（ライトブロック）が得られ、グローバルメモリに格納される。

【００８３】

上述した流れから分かるように、継続ライトブロックがある場合、該継続ライトブロックのサブライトブロックのグローバルメモリへの転送（ライト転送）と、該継続ライトブロックに対応する継続リードブロックのサブリードブロックのローカルメモリ／プライベートモリへの転送（リード転送）が行わっている。

【００８４】

継続ライトブロックのサブライトブロックが、既にプライベートメモリ／ローカルメモリにリード転送されたサブリードブロックとして次のカーネルに使用可能であれば、継続ライトブロックをサブリードブロックのローカルメモリ／プライベートモリからグローバルメモリへ転送せず、そのまま次のカーネルに供するようにすれば、ライト転送もリード転送も削減でき、ＯｐｅｎＣＬデバイスの処理効率の向上を図ることができる。

【００８５】

その他の課題と新規な特徴は、本明細書の記述および添付図面から明らかになるであろう。

【課題を解決するための手段】

【００８６】

一実施の形態は、ＯｐｅｎＣＬデバイスの複数の演算素子による並列演算を制御する演算制御方法であり、第１のステップと第２のステップを有する。

【００８７】

第１のステップは、ＯｐｅｎＣＬデバイスの最下位階層のメモリ（例えばグローバルメモリ）に１つ以上格納されたリードブロックと、他の階層のメモリ（例えばプライベートメモリやローカルメモリ）から最下位階層のメモリに転送される１つ以上のライトブロックとに対して夫々設定された属性群を取得して保持するステップである。

【００８８】

第２のステップは、第１のステップにより保持された各前記属性群と、前記ＯｐｅｎＣＬデバイスの構成を示す構成パラメータとに基づいて、夫々の前記リードブロックと前記ライトブロックの転送方式を決定し、決定した転送方式に応じて各前記リードブロックと前記ライトブロックの転送、及び該転送に対応する前記並列演算の制御を行うステップである。

【００８９】
リードブロックは、並列演算の演算対象として最下位階層のメモリから他の階層のメモリに転送されるデータブロックである。ライトブロックは、並列演算後に他の階層のメモリから最下位階層のメモリに転送されるデータブロックであり、前記１つ以上のリードブロックに対する並列演算の演算結果である。
【００９０】
前記属性群は、前記転送方式を決定するために必要である一方、前記ＯｐｅｎＣＬデバイスの構成に依存しない属性を複数含むものであり、例えば割当属性、余白属性、依存属性を有する。
【００９１】
割当属性は、当該データブロックを複数のサブブロックに分割して転送するか否か、及び、分割して転送する場合における分割方式を示すものである。
【００９２】
余白属性は、当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックと共に転送される、該サブブロックの周辺のデータのサイズを示すものである。
【００９３】
依存属性は、当該データブロックを複数のサブブロックに分割して転送する場合における、該サブブロックがその周辺の他のサブブロックと依存関係が有るか否か、及び、前記依存関係がある場合における全ての依存方向を示す依存属性を有する。
【００９４】
なお、ライトブロックの属性群は、該ライトブロックが既に前記他の階層のメモリに存在し、かつ、前記最下位階層のメモリに転送されると仮定して設定されたものである。
【００９５】
第２のステップは、第１のカーネルに続いて第２のカーネルをＯｐｅｎＣＬデバイスに実行させ、かつ、第１のカーネルに対応する並列演算の各ライトブロックのうちに、第２のカーネルに対応する並列演算のリードブロックとして使用される継続ライトブロックが含まれており、かつ、第１のカーネルに対して設定された継続ライトブロックの割当属性と、第２のカーネルに対して設定された、継続ライトブロックに対応するリードブロック（継続リードブロック）の割当属性とが一致する場合において、該継続ライトブロックが最下位階層へ転送されず、他の階層のメモリを介して第２のカーネルの実行に供されることにより、前記第１のカーネルと前記第２のカーネルの実行をパイプライン化するパイプライン化制御を行うステップをさらに含む。
【００９６】
該パイプライン化制御ステップにおいて、第１のカーネルに対応する各リードブロックの夫々に対して、該リードブロックに対して設定された余白属性と依存属性に、第２のカーネルに対して設定された、継続ライトブロックに対応する継続リードブロックの余白属性と依存属性を夫々論理和加算する。
【００９７】
なお、上記実施の形態の方法を装置やシステムに置き換えて表現したもの、該方法を実行するＯｐｅｎＣＬデバイス、該方法をコンピュータに実行せしめるプログラムなども、実施の態様としては有効である。
【発明の効果】
【００９８】
前記一実施の形態によれば、ＯｐｅｎＣＬデバイスの処理効率を向上させることができる。
【図面の簡単な説明】
【００９９】
【図１】一実施の形態にかかるＯｐｅｎＣＬシステムを示す図である。
【図２】単独カーネルの場合を説明するための具体例を示す図である。

【図３】リード属性を説明するための図である。
【図４】余白属性を説明するための図である。
【図５】放送属性を説明するための図である。
【図６】割当属性を説明するための図である（その１）
【図７】割当属性を説明するための図である（その２）。
【図８】依存属性を説明するための図である。
【図９】図２に示す各データブロックに対して設定された属性群の例を示す図である。
【図１０】図１に示すＯｐｅｎＣＬシステムにおける演算ユニットの構成を示すパラメータの例である。
【図１１】階層内での対応付けと階層間での対応付けを示す図である。
【図１２】図２に示す例に対してシナリオ決定部が決定した内容の一部を示す図である。
【図１３】関連性のある２つのカーネルを連続して実行する際に行われるパイプライン化制御を説明するための図である（その１）。
【図１４】関連性のある２つのカーネルを連続して実行する際に行われるパイプライン化制御を説明するための図である（その２）。
【図１５】パイプライン化制御なしとパイプライン化制御ありの場合における各データブロックの転送態様を比較するための図である。
【図１６】パイプライン化制御なしとパイプライン化制御ありの場合におけるカーネルの実行態様を比較するための図である。
【図１７】ＯｐｅｎＣＬシステムのプラットフォームモデルを示す図である。
【図１８】ＯｐｅｎＣＬデバイスの構成を示す図である。
【図１９】ＯｐｅｎＣＬデバイスにおける各種メモリを説明するための図である。
【図２０】逐次プロセッサの例を示す図である。
【図２１】特許文献１に開示された技術が解決しようとする課題を説明するための図である（その１）。
【図２２】特許文献１に開示された技術が解決しようとする課題を説明するための図である（その２）。
【図２３】特許文献１のＯｐｅｎＣＬシステムにおいて、ユーザにより指定する必要のある内容を示す図である。

【発明を実施するための形態】

【０１００】

説明の明確化のため、以下の記載及び図面は、適宜、省略、及び簡略化がなされている。また、様々な処理を行う機能ブロックとして図面に記載される各要素は、ハードウェア的には、ＣＰＵ、メモリ、その他の回路で構成することができ、ソフトウェア的には、メモリにロードされたプログラムなどによって実現される。したがって、これらの機能ブロックがハードウェアのみ、ソフトウェアのみ、またはそれらの組合せによっていろいろな形で実現できることは当業者には理解されるところであり、いずれかに限定されるものではない。なお、各図面において、同一の要素には同一の符号が付されており、必要に応じて重複説明は省略されている。

【０１０１】

また、上述したプログラムは、様々なタイプの非一時的なコンピュータ可読媒体（ｎｏｎ－ｔｒａｎｓｉｔｏｒｙ　ｃｏｍｐｕｔｅｒ　ｒｅａｄａｂｌｅ　ｍｅｄｉｕｍ）を用いて格納され、コンピュータに供給することができる。非一時的なコンピュータ可読媒体は、様々なタイプの実体のある記録媒体（ｔａｎｇｉｂｌｅ　ｓｔｏｒａｇｅ　ｍｅｄｉｕｍ）を含む。非一時的なコンピュータ可読媒体の例は、磁気記録媒体（例えばフレキシブルディスク、磁気テープ、ハードディスクドライブ）、光磁気記録媒体（例えば光磁気ディスク）、ＣＤ－ＲＯＭ（Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）ＣＤ－Ｒ、ＣＤ－Ｒ／Ｗ、半導体メモリ（例えば、マスクＲＯＭ、ＰＲＯＭ（Ｐｒｏｇｒａｍｍａｂｌｅ　ＲＯＭ）、ＥＰＲＯＭ（Ｅｒａｓａｂｌｅ　ＰＲＯＭ）、フラッシュＲＯＭ、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ））を含む。また、プログラムは、様々なタイプの

一時的なコンピュータ可読媒体（ｔｒａｎｓｉｔｏｒｙ　ｃｏｍｐｕｔｅｒ　ｒｅａｄａｂｌｅ　ｍｅｄｉｕｍ）によってコンピュータに供給されてもよい。一時的なコンピュータ可読媒体の例は、電気信号、光信号、及び電磁波を含む。一時的なコンピュータ可読媒体は、電線及び光ファイバ等の有線通信路、又は無線通信路を介して、プログラムをコンピュータに供給できる。

【０１０２】

図１は、一実施の形態にかかるＯｐｅｎＣＬシステム１００を示す。ＯｐｅｎＣＬシステム１００は、ＯｐｅｎＣＬホスト（以下、単に「ホスト」という）１１０、ＯｐｅｎＣＬデバイス（以下、単に「デバイス」という）１２０を備える。

【０１０３】

デバイス１２０は、演算制御部２００、演算ユニット１４０を備える。演算制御部２００は、属性群格納部２１０、カーネル間情報格納部２２０、シナリオ決定部２３０を有する。演算ユニット１４０は、デバイスメモリ１５０、複数のコンピュートユニット（ＣＵ）１６０、ＣＵ１６０毎に設けられたローカルメモリ１７０を有する。

【０１０４】

デバイスメモリ１５０は、グローバルメモリ１５２、コンスタントメモリ１５４を有する。また、夫々のＣＵ１６０は、複数のＰＥ１６２と、ＰＥ１６２毎に設けられたプライベートメモリ１６４を有する。

【０１０５】

なお、グローバルメモリ１５２、コンスタントメモリ１５４、ＣＵ１６０、ＰＥ１６２、プライベートメモリ１６４、ローカルメモリ１７０は、通常のＯｐｅｎＣＬデバイスにおける同一名称のものと同様であり、ここで詳細な説明を省略する。

【０１０６】

ホスト１１０は、デバイス１２０に演算を行わせる際に、ユーザにより引数の指定がなされた、該演算に対応するカーネルをデバイス１２０に転送する。

【０１０７】

デバイス１２０において、カーネル間情報格納部２２０は、連続して実行される複数のカーネル間のデータブロックの依存関係を示す情報（「カーネル間データブロック依存情報」という）を格納している。この情報は、ユーザにより直接カーネル間情報格納部２２０に入力されるようにしてもよいし、上記複数のカーネルのうちの１つ（例えば最も先に実行されるカーネル）の引数に含まれるようにして、該カーネルと共にホスト１１０からデバイス１２０に転送され、カーネル間情報格納部２２０に格納されるようにしてもよい。

【０１０８】

上記カーネル間データブロック依存情報は、例えば、先に実行されるカーネルのライトブロックのうちに、後に実行されるカーネルのリードブロックとして使用されるものかあるか否か、及びある場合における、継続ライトブロックと継続リードブロックとの対応関係を示すものである。

【０１０９】

なお、単独カーネルの場合には、カーネル間データブロック依存情報は、「依存なし」を示すことになる。単独カーネルは、他のカーネルと関連性が無く、その演算対象となるリードブロックが、他のカーネルの演算結果となるライトブロックでもなければ、そのライトブロックが他のカーネルのリードブロックとしても使用されないカーネルを意味する。

【０１１０】

演算制御部２００は、ホスト１１０からのカーネルが示す演算内容、属性群格納部２１０に格納された各属性群と、カーネル間情報格納部２２０に格納されたカーネル間データブロック依存情報とにに基づいて、演算ユニット１４０を制御する。演算ユニット１４０は、演算制御部２００の制御に従って演算を行い、演算結果を得る。演算ユニット１４０が行う演算は、演算対象のデータをデバイスメモリ１５０からプライベートメモリ１６４

／ローカルメモリ１７０への転送（リード転送）と、演算結果のデータをデバイスメモリ１５０への転送（ライト転送）が伴う。

【０１１１】

通常、演算対象のデータと演算結果のデータのいずれもグローバルメモリ１５２に格納されるため、以下において、「リード転送」と「ライト転送」がグローバルメモリ１５２と、プライベートメモリ１６４／ローカルメモリ１７０との間の転送であるとして説明をする。演算対象のデータがコンスタントメモリ１５４にも格納される場合には、「リード転送」がコンスタントメモリ１５４からプライベートメモリ１６４／ローカルメモリ１７０への転送も含むことを理解されたい。

【０１１２】

＜単独カーネルの場合＞

分かりやすいように、まず、図２に示す具体例を参照しながら、単独カーネルを実行する場合について説明する。

【０１１３】

この場合、カーネル間情報格納部２２０には、「依存なし」を示すカーネル間データブロック依存情報が格納されることになる、ＯｐｅｎＣＬシステム１００における処理は、特許文献１に開示されたＯｐｅｎＣＬシステムの処理と同様である。

【０１１４】

なお、以下の説明における用語「次元」は、前述したインデックス空間の次元に対応する。ＯｐｅｎＣＬでは、インデックス空間は、３次元まで指定することができるが、説明上の便宜のため、２次元を例にする。

【０１１５】

図２に示すように、ユーザが実現したい処理は、式（１）に示す処理である。

【数１】

$$R_{(x,\,y)} = \sum \left[P_i * Q_{(x,\,(y+i))} \right] \quad (i = 0 \sim 8) \qquad (1)$$

【０１１６】

演算対象のデータは、リードブロックＰとリードブロックＱに含まれる各データである。演算結果のデータは、ライトブロックＲに含まれる各データである。図２に示す例では、ライトブロックは、ライトブロックＲの１つのみである。

【０１１７】

式（１）に示す処理を実現するためのカーネルは、当然ながら、ユーザにより作成される。カーネルの作成は、デバイス１２０の各ＰＥ１６２により実行されるプログラムコード（以下、「ユーザ指定処理」という）の作成処理と、引数の指定処理を含む。

【０１１８】

本実施の形態のＯｐｅｎＣＬシステム１００では、ＯｐｅｎＣＬで定められた各引数以外に、さらに、８つの属性を含む「属性群」がサポートされている。これらの８つの属性は、「サイズ属性」、「リード属性」、「ライト属性」、「余白属性」、「放送属性」、「割当属性」、「階層属性」、「依存属性」であり、ユーザにより、カーネルの引数として、全てのリードブロックとライトブロックに対して指定される。なお、属性の指定に際して、必ずしも８つの属性の全てを指定する必要があるとは限らない。

【０１１９】

上記属性群に含まれる８つの属性は、グローバルメモリ１５２と、プライベートメモリ１６４及び／またはローカルメモリ１７０との間で転送される方式（転送方式）を決定するために参照する必要があるパラメータである。以下、特別な説明が無い限り、「転送」は、グローバルメモリ１５２と、プライベートメモリ１６４及び／またはローカルメモリ１７０との間の転送を意味する。

【０１２０】

「転送方式」は、下記の内容が含まれる。
（１）転送方式１：分割の有無
該転送方式１は、「分割無し」方式と「分割有り」方式のいずれかである。
【０１２１】
リードブロックについて、「分割無し」とは、該リードブロックの全てのデータを一度のリード転送により転送することを意味し、「分割有り」とは、該リードブロックを複数のサブリードブロックＳＲＢに分割し、一度のリード転送により一つのワークグループへ１つのサブリードブロックＳＲＢのみを転送することを意味する。
【０１２２】
ライトブロックについて、「分割無し」とは、該ライトブロックの全てのデータ（演算結果）を一度のライト転送によりグローバルメモリ１５２に転送することを意味し、「分割有り」とは、該ライトブロックを複数のサブライトブロックＳＷＢに分割し、一度のライト転送により一つのワークグループから１つのサブライトブロックＳＷＢのみを転送することを意味する。
【０１２３】
（２）転送方式２：分割方式
該転送方式２は、該データブロックを複数のサブブロックに分割する場合に、どのように分割するかを意味する。
【０１２４】
（３）転送方式３：分配方式（サブリードブロックＳＲＢの対応付け方法）
該転送方式３は、リードブロックを対象として、上記転送方式１が分割有りの場合に指定される。
【０１２５】
すなわち分配方式は、リードブロックが複数あるときには、同じ回のリード転送により転送される、各リードブロックからの１つずつのサブリードブロックＳＲＢをどのようにしてプライベートメモリ１６４／ローカルメモリ１７０に分配するかを意味する。リードブロックが１つしか無い場合には、分割方式は、該サブリードブロックＳＲＢをプライベートメモリ１６４／ローカルメモリ１７０に均等に分配する方式になる。
【０１２６】
（４）転送方式４：統合方式（サブライトブロックＳＷＢの対応付け方法）
該転送方式４は、ライトブロックを対象として、上記転送方式１が分割有りの場合に指定される。
【０１２７】
なお、「統合する」とは、グローバルメモリ１５２における、該ライトブロックの個々のサブライトブロックＳＷＢに割り当てられた領域に書き込むことを意味する。
【０１２８】
統合方式は、ライトブロックが複数あるときには、同じ回のライト転送により転送される、プライベートメモリ群／ローカルメモリ１７０に格納されている複数のサブライトブロックＳＷＢを、どのようにして夫々のライトブロックに統合するかを意味する。ライトブロックが１つしか無い場合には、統合方式は、プライベートメモリ群／ローカルメモリ１７０に格納された該ライトブロックの各サブライトブロックＳＷＢのデータを、上記１つのライトブロックに統合する方式になる。
【０１２９】
上記した「転送方式」は、従来では、ユーザコードにより指定される必要がある。演算処理の内容、及びデバイス１２０（具体的には演算ユニット１４０）の構成に依存するため、ユーザコードが煩雑であり、可搬性を維持するのは極めて困難と言ってよい。
【０１３０】
対して、本実施の形態のＯｐｅｎＣＬシステム１００は、演算処理の内容とＯｐｅｎＣＬデバイスの構成の双方を考慮しつつその都度、個々のユーザが時間を費やして場当たり的に転送方式を決定してきた処理を、ＯｐｅｎＣＬデバイスの構成に依存する処理と依存

しない処理とに分け、さらに、ＯｐｅｎＣＬデバイスに依存する処理については、ＯｐｅｎＣＬデバイスに依存しない処理により指定したパラメータと、ＯｐｅｎＣＬデバイスの構成を示すパラメータとに基づいて自動的に決定するようにしたものである。

【０１３１】

具体的には、本実施の形態のＯｐｅｎＣＬシステム１００では、ＯｐｅｎＣＬデバイスに依存しない処理は、データブロックに対する上記属性群の指定に該当する。これらの属性群は、転送方式を決定する上で必要なパラメータであるものの、ＯｐｅｎＣＬデバイスの構成に依存しないものである。以下、詳細に説明する。なお、以下において、各属性が「転送方式を決定する上で必要であるものの、ＯｐｅｎＣＬデバイスの構成に依存しないもの」であることを繰り返せず、このこと以外の要素についてのみを説明する。

【０１３２】

上記８つの属性は、さらに、「固有属性」、「演算属性」、「ポリシ属性」の３種類に分けられる。図２に示す各データブロックの例を参照しながら説明する。

【０１３３】

「固有属性」は、演算処理の内容及びユーザの意志に関係なく、当該データブロックが持つ固有の属性である。本実施の形態において、「固有属性」は、下記の「サイズ属性」である。

【０１３４】

＜サイズ属性＞

この属性は、データブロックのサイズを示すパラメータであり、例えば、次元毎のワード数、及びワード毎のバイト数やビット数である。この属性は、全てのデータブロックに対して必ず指定される。

【０１３５】

そのため、図２に示すリードブロックＰ、リードブロックＱ、及びライトブロックＲのサイズ属性は、下記のようになる。

【０１３６】

リードブロックＰは、２次元のデータブロックであり、Ｘ方向サイズＬｘとＹ方向サイズＬｙが、共に３ワードである。なお、リードブロックＰは、１ワードのビット数が、８ビットである。そのため、サイズ属性として、リードブロックＰに対して、「Ｘ方向サイズＬｘ：３、Ｙ方向サイズＬｙ：３、ビット数／ワード：８」が指定される。

【０１３７】

同様に、リードブロックＱに対して、「「Ｘ方向サイズＬｘ：６４０、Ｙ方向サイズＬｙ：４８０、ビット数／ワード：１６」が指定される。

【０１３８】

また、ライトブロックＲに対して、「「Ｘ方向サイズＬｘ：６４０、Ｙ方向サイズＬｙ：４８０、ビット数／ワード：３２」が指定される。

【０１３９】

「演算属性」は、ユーザの意志には関係しないものの、演算処理の内容に関係する属性である。本実施の形態において、「演算属性」は、下記の「リード属性」、「ライト属性」、を含む。演算属性は、各データブロックが１つ以上のサブブロックに分割される仮定の元で指定される。ライトブロックについては、さらに、リードブロックが既にプライベートメモリ群／ローカルメモリ上に存在する仮定が加えられる。なお、データブロックが１つのサブブロックに分割されることは、分割されないことである。

【０１４０】

＜リード属性＞

この属性は、まず、該データブロックが演算対象のデータ（つまり、リード転送されるデータ）であるか否か、及び演算対象のデータである場合の転送順位を示す。転送順位は、該データブロックの各サブブロックを、どのような順位で転送するかを指定するパラメータである。

【０１４１】

ライトブロックは、リード転送されないので、リード転送されないことを示す「ＮＯＮＥ」が指定される。リードブロックは、リード転送されるデータブロックであるので、「リード転送される」として、転送順位が指定される。

【０１４２】

本実施の形態のＯｐｅｎＣＬシステム１００において、リード属性として設定される転送順位は、「ＴＯＰ　ＬＥＦＴ」、「ＢＯＴＴＯＭ　ＲＩＧＨＴ」、「ランダム」が指定可能である。なお、「ランダム」が指定された場合に限り、別途、転送順位を示す情報を格納した領域へのポインタが指定される。

【０１４３】

「ＴＯＰ　ＬＥＦＴ」は、左上端のサブブロックから転送することを示し、「ＢＯＴＴＯＭ　ＲＩＧＨＴ」は、右下端のサブブロックから転送することを示す。図３は、リード属性として「「ＴＯＰ　ＬＥＦＴ」が指定された場合のサブブロックの転送順位を示す。

【０１４４】

図３に示すように、この場合、グローバルメモリ１５２に格納されたデータブロック（リードブロック）は、転送順位が、左上端のサブブロック１、サブブロック１の右隣のサブブロック２、サブブロック３の右隣のサブブロック３、・・・となっている。

【０１４５】

図２に示す各データブロックのリード属性を説明する。

式（１）に示す演算処理の内容に基づいて、リードブロックＰとリードブロックＱは、リード転送されるデータであるため、リード属性として、転送順位が「ＴＯＰ　ＬＥＦＴ」に指定される。

【０１４６】

一方、ライトブロックＲは、リード転送されるデータではないため、リード属性が「ＮＯＮＥ」に指定される。

【０１４７】

＜ライト属性＞

この属性は、まず、該データブロックが演算結果のデータ（つまり、ライト転送されるデータ）であるか否か、及び演算結果のデータである場合の転送順位を示す。転送順位は、各サブライトブロックＳＷＢを、どのような順位で転送するかを指定するパラメータである。

【０１４８】

リードブロックは、ライト転送されないので、ライト転送されないことを示す「ＮＯＮＥ」が指定される。ライトブロックは、ライト転送されるデータブロックであるので、「ライト転送される」として、転送順位が指定される。

【０１４９】

本実施の形態のＯｐｅｎＣＬシステム１００において、ライト属性として設定される転送順位は、「ＴＯＰ　ＬＥＦＴ」、「ＢＯＴＴＯＭ　ＲＩＧＨＴ」、「ランダム」が指定可能である。なお、「ランダム」が指定された場合に限り、別途、転送順位を示す情報を格納した領域へのポインタが指定される。

【０１５０】

リードブロックは、ライト転送されないので、ライト転送されないことを示す「ＮＯＮＥ」が指定される。そのため、図２に示すリードブロックＰとリードブロックＱは、ライト属性として「ＮＯＮＥ」が指定される。

【０１５１】

ライトブロックは、ライト転送されるデータブロックであるので、「ライト転送される」として、上記転送順位が指定される。この転送順位は、該データブロックの各サブブロックを、どのような順位で転送するかを指定するパラメータである。

【０１５２】

本実施の形態のＯｐｅｎＣＬシステム１００において、ライト属性として設定される転送順位は、リード属性として設定される転送順位と同様に、「ＴＯＰ　ＬＥＦＴ」と、「

ＢＯＴＴＯＭ　ＲＩＧＨＴ」と、「ランダム」が指定可能である。「ランダム」が指定された場合に限り、別途、転送順位を示す情報を格納した領域へのポインタが指定される。

【０１５３】

ライト属性として設定される転送順位の夫々のパラメータの意義は、リード属性として設定される転送順位の相対応するパラメータと同一であるので、ここで詳細な説明を省略する。

【０１５４】

図２に示す各データブロックのライト属性を説明する。

式（１）に示す演算処理の内容に基づいて、ライトブロックＲは、ライト転送されるデータであるため、ライト属性として、転送順位が「ＴＯＰ　ＬＥＦＴ」に指定される。

【０１５５】

一方、リードブロックＰとリードブロックＱは、ライト転送されるデータではないため、ライト属性が「ＮＯＮＥ」に指定される。

【０１５６】

「ポリシ属性」は、演算処理の内容と共に、ユーザがどのように転送及び演算処理を実行したいかの意志に関係する属性である。本実施の形態のＯｐｅｎＣＬシステム１００において、「ポリシ属性」は、「余白属性」、「放送属性」、「割当属性」、「階層属性」、「依存属性」を含む。ポリシ属性も、各データブロックが１つ以上のサブブロックに分割される仮定の元で指定される。ライトブロックについては、さらに、ライトブロックが既にプライベートメモリ２０／ローカルメモリ２２上に存在する仮定が加えられる。なお、データブロックが１つのサブブロックに分割されることは、分割されないことである。

【０１５７】

＜余白属性＞

この属性は、リードブロックを対象とするパラメータであり、サブリードブロックＳＲＢ内のデータと共に転送される、該サブリードブロックＳＲＢの境界と隣接する該サブリードブロックＳＲＢ外のデータの量を示すパラメータである。また、余白属性は、次元毎に指定される。余白属性のパラメータの単位は、ワードである。

【０１５８】

なお、ライトブロックについては、余白属性を指定できない、または指定したとしても無視されるようになっている。

【０１５９】

図４は、余白属性として、Ｘ方向に「１」、Ｙ方向に「２」が指定された場合に該データブロック（リードブロック）の各サブリードブロックＳＲＢの転送時の転送範囲を示す。この場合、該サブリードブロックＳＲＢの転送時の転送範囲は、該サブリードブロックＳＲＢ内のデータに加え、該サブリードブロックＳＲＢの左右両側の境界に隣接する１列ずつのデータと、上下両端の境界に隣接する２行ずつのデータが含まれる。なお、図４では、データブロックの上端に位置するサブリードブロックＳＲＢを例としており、該サブリードブロックＳＲＢの上端の境界に隣接するデータが無いため、該サブリードブロックＳＲＢの転送時の転送範囲には、上端の境界に隣接するデータが含まれない。

【０１６０】

後に詳細に説明するが、本実施の形態のＯｐｅｎＣＬシステム１００では、各リードブロックについて、１度のリード転送時に1つのＣＵ１６０に対して１つのサブリードブロックＳＲＢが転送され、１度のリード転送により該ＣＵ１６０のプライベートメモリ／ローカルメモリに転送された各サブブロックサブリードブロックＳＲＢを対象として該ＣＵ１６０により演算が行われる。そして、演算の結果は、夫々のライトブロックの１つのサブブロックとして該ＣＵ１６０からグローバルメモリに転送される。

【０１６１】

例えば、２次元画像に対して３×３のコンボリューション演算を行う場合、注目画素と上下左右の４つの方向において、該注目画素に隣接する１画素が必要である。サブリードブロックＳＲＢ内のデータのみが転送されるのでは、該サブリードブロックＳＲＢの最も

外側に位置する各画素に対する演算ができない。そのため、この場合、余白属性として、Ｘ方向とＹ方向に共に「１」を指定する必要がある。

【０１６２】

図２に示す各データブロックの余白属性を説明する。なお、余白属性の指定は、他の属性の指定と関係するため、余白属性のみでは説明しにくい。そのため、ここでは、図２に示す各データブロックに対して指定された余白属性の値のみを示し、それらの意義の詳細については後述する。

【０１６３】

リードブロックＰとリードブロックＱは、余白属性の指定対象であるが、リードブロックＰに対しては、Ｘ方向とＹ方向のいずれについても余白が「０」に設定される。また、リードブロックＱに対しては、Ｘ方向の余白が「０」、Ｙ方向の余白が「９」に指定される。

【０１６４】

従って、リードブロックＰの転送時には、サブリードブロックＳＲＢ内のデータのみが転送される。一方、リードブロックＱの転送時には、サブリードブロックＳＲＢ内のデータに加え、該サブリードブロックＳＲＢと下端で隣接する９行のデータも転送される。

【０１６５】

ライトブロックＲは、ライト転送されるデータであるため、余白属性の指定対象ではない。すなわち、ライトブロックＲの転送時には、当該サブライトブロックＳＷＢ内のデータのみが転送される。

【０１６６】

＜放送属性＞

この属性は、各リードブロックについてはサブリードブロックＳＲＢの転送先、各ライトブロックについてはサブライトブロックＳＷＢの転送元が、プライベートメモリとローカルメモリのいずれになるかを指定するパラメータであり、「ＯＮ」と「ＯＦＦ」のいずれである。例えば、図５に示すように、放送属性の「ＯＮ」は、上記転送先または転送元としてローカルメモリを指定し、放送属性の「ＯＦＦ」は、上記転送先または転送元としてプライベートメモリを指定する。

【０１６７】

図２に示す各データブロックの放送属性を説明する。なお、放送属性の指定も、他の属性と関係するため、放送属性のみでは説明しにくい。そのため、ここでは、図２に示す各データブロックに対して指定された放送属性の値のみを示し、それらの意義の詳細については後述する。

【０１６８】

リードブロックＰは、放送属性が「ＯＮ」に指定される。そのため、リードブロックＰの転送時に、各サブリードブロックＳＲＢは、ローカルメモリ１７０に転送される。

【０１６９】

リードブロックＱとライトブロックＲは、放送属性が「ＯＦＦ」に指定される。そのため、リードブロックＱの転送時には、各サブリードブロックＳＲＢは、プライベートメモリ１６４に転送される。また、ライトブロックＲのサブライトブロックＳＷＢは、プライベートメモリ群からグローバルメモリ１５２に転送される。

【０１７０】

＜割当属性＞

この属性は、サブリードブロックＳＲＢとサブライトブロックＳＷＢをどのようにしてＣＵ１６０のプライベートメモリ群／ローカルメモリに割り当てる割当方式を示すパラメータである。

【０１７１】

本実施の形態のＯｐｅｎＣＬシステム１００では、「縦優先」と「横優先」の２つの割当方式がサポートされる。

【０１７２】

割当属性を詳細に説明する前に、まず、ワークグループサイズＷＧｓと、１つのアイテムに割り当てるデータ量を説明する。

【０１７３】

放送属性が「ＯＦＦ」のデータブロックに対応する「ワークグループサイズＷＧｓ」は、１つのワークグループ内のアイテム数により表されるデータ量であり、例えばアイテム数がＮであれば、ワークグループサイズＷＧｓはＮワードになる。一方、放送属性が「ＯＮ」のデータブロックに対応するワークグループサイズＷＧｓは、常に１と見なされる。以降、放送属性が「ＯＦＦ」のデータブロックに対応する「ワークグループサイズＷＧｓ」を単にワークグループサイズＷＧｓと呼ぶ。このワークグループサイズＷＧｓは、後にデバイス１２０における演算制御部２００のシナリオ決定部２３０により決定されるが、ユーザが、カーネルの作成時に、その最大値と最小値のいずれをとるかを指定することができる。

【０１７４】

また、１つのアイテムに割り当てるデータ量も、後にシナリオ決定部２３０により決定される。ワークグループサイズＷＧｓが最小値をとる場合とは、１アイテムが１ＰＥに対応する場合である。それに対し、ＰＥのプライベートメモリをＭ（Ｍ：２以上の整数）分割してＭアイテムを１ＰＥに対応させることで、ワークグループサイズＷＧｓを最大Ｍ倍大きく指定できるＯｐｅｎＣＬデバイスの場合、最大値は最小値のＭ倍となる。一方、最大値のワークグループサイズＷＧｓを利用する場合、１ＰＥ当たりのプライベートメモリは最小値のワークグループサイズＷＧｓを利用する場合のＭ分の１となるだけである。以下、説明を簡単にするため、シナリオ決定部２３０は、常に、ワークグループサイズＷＧｓとしてその最小値を採用するものとし、その結果、１アイテムが１ＰＥに対応するので、以降、アイテムという表現の代わりにＰＥという表現を用いる。

【０１７５】

「縦優先」は、サブブロックのＸ方向のサイズ（ＳＢｓｘ）がワークグループサイズＷＧｓであり、Ｙ方向のサイズＳＢｓｙが１つのアイテムに割り当てるデータ量、すなわち１ＰＥに割り当てるデータ量であるようにデータブロックを割り当てる方式である。

【０１７６】

割当属性として「縦優先」が指定されたデータブロック内の各サブブロックは、１サブブロックが１ワークグループに対応し、かつ、該サブブロック内の１列のデータが該ワークグループ内の１つのＰＥ１６２のプライベートメモリ１６４に対応するように、グローバルメモリ１５２とプライベートメモリ１６４間で転送される。

【０１７７】

図６は、割当属性として「縦優先」が指定されたリードブロックの場合の例を示す。この場合、夫々のサブリードブロックＳＲＢは、該サブリードブロックＳＲＢ内の全てのデータが同一のワークグループ内のＰＥ１６２のプライベートメモリ１６４に格納され、かつ、該サブブロック内の同列のデータが同一のＰＥ１６２のプライベートメモリ１６４に格納されるように、グローバルメモリ１５２からプライベートメモリ１６４に転送される。

【０１７８】

「横優先」は、サブブロックのＸ方向のサイズＳＢｓｘがワークグループサイズＷＧｓの整数倍になるようにデータブロックを分割する方式である。なお、サブブロックのＹ方向のサイズＳＢｓｙは、後にシナリオ決定部２３０により決定される。

【０１７９】

割当属性として「横優先」が指定されたデータブロック内の各サブブロックは、１サブブロックが１つのワークグループに対応し、該サブブロック内の全てのデータが、同一のワークグループに含まれるＰＥ１６２のプライベートメモリ１６４に転送される。さらに、サブブロックの各行は、該行をワークグループサイズＷＧｓの量毎に区切って得た整数個の区切りブロックのデータが、区切りブロック毎に、該ワークグループ内のＷＧｓ個のＰＥ１６２のプライベートメモリ１６４に均等に分配される。

【０１８０】
図７は、割当属性として「横優先」が指定されたリードブロックの場合の例を示す。なお、該例では、サブライトブロックＳＷＢのＹ方向のサイズＳＢｓｙが１ワードとされている。つまり、該リードブロックは、サブリードブロックＳＲＢのＸ方向のサイズＳＢｓｘがワークグループサイズＷＧｓの整数倍であり、行数が１であるように分割される。
【０１８１】
図示のように、この場合、該リードブロックの夫々のサブリードブロックＳＲＢサは、該サブリードブロックＳＲＢ内の全てのデータが同一のワークグループ内のＰＥ１６２のプライベートメモリ１６４に格納されるように転送される。さらに、該サブリードブロックＳＲＢを行方向にワークグループサイズＷＧｓの量毎に区切って得た整数個の区切りブロックのデータが、区切りブロック毎に、該ワークグループ内のＷＧｓ個のＰＥ１６２のプライベートメモリ１６４に均等に分配されるように転送される。例えば、データ１から、データａの前のデータまでの複数のデータは、１つの区切りブロックを構成しており、同一のワークグループの複数のＰＥ１６２のプライベートメモリ１６４に夫々格納されるように転送される。また、データａから、データｊの前のデータまでの複数のデータも、１つの区切りブロックを構成しており、ワークグループの複数のＰＥ１６２のプライベートメモリ１６４に夫々格納されるように転送される。
【０１８２】
図２に示す各データブロックの割当属性を説明する。なお、割当属性の指定も、他の属性と関係するため、割当属性のみでは説明しにくい。そのため、ここでは、図２に示す各データブロックに対して指定された割当属性の値のみを示し、それらの意義の詳細については後述する。
【０１８３】
リードブロックＰは、割当属性として「縦優先」が指定されたとする。
リードブロックＱとライトブロックＲも、割当属性が「縦優先」に指定されたとする。
【０１８４】
＜階層属性＞
この属性は、１以上の自然数で指定される階層数である。同一の階層数が指定された複数のリードブロックは、１度のリードあるいはライト転送により、夫々１つのサブブロックＳＢが転送される。
【０１８５】
図２に示す各データブロックの階層属性を説明する。なお、階層属性の指定も、他の属性と関係するため、階層属性のみでは説明しにくい。そのため、ここでは、図２に示す各データブロックに対して指定された階層属性の値のみを示し、それらの意義の詳細については後述する。
【０１８６】
リードブロックＰとリードブロックＱは、階層属性が「２」と「１」に夫々指定される。また、ライトブロックＲの階層属性が「１」に指定される。
【０１８７】
＜依存属性＞
この属性は、当該データブロックに対して、サブブロックと、該サブブロックと隣接する８つの他のサブブロックとのデータ依存関係を示すパラメータであり、「依存有り」、「依存無し」を指定可能である。また、「依存有り」については、さらに、３種類の依存関係を指定可能である。
【０１８８】
「依存有り」は、Ｒ１型、Ｒ２型、Ｒ３型の３種類がある。図８を参照して説明する。なお、図８において、点線により囲まれる枠は、サブブロックＳＢを示し、数字は、転送順位を示す。
【０１８９】
図８は、割当属性とリード属性が「縦優先」と「ＴＯＰ　ＬＥＦＴ」に夫々指定されて

いる場合のＲ１型～Ｒ３型依存関係を示す。

【０１９０】

図示のように、この場合、「Ｒ１型依存」は、注目サブブロックＳＢ視点から見て上のサブブロックＳＢと依存関係を有する場合に指定される。「Ｒ２型依存」は、注目サブブロックＳＢから見て上、斜め左上、左のサブブロックＳＢと依存関係を有する場合に指定される。また、「Ｒ３型依存」は、注目サブブロックＳＢから見て、隣接する８つのサブブロックＳＢのうちの、下、斜め右下の２つを除いた６つとデータ依存関係を有する場合に指定される。

【０１９１】

図２に例示したデータブロックの依存属性を説明する。なお、依存属性の指定も、他の属性と関係するため、依存属性のみでは説明しにくい。そのため、ここでは、図２に示す各データブロックに対して指定された依存属性の値のみを示し、それらの意義の詳細については後述する。

【０１９２】

該例において、リードブロックＰ、リードブロックＱ、ライトブロックＲのいずれも、依存属性が「ＮＯＮＥ」に指定される。

【０１９３】

以上、本実施の形態のＯｐｅｎＣＬシステム１００でサポートされる８つの属性を説明した。これらの８つの属性は、いずれもデバイス１２０における演算ユニット１４０の構成に依存しない。後の説明時に分かりやすいように、図２に示す３つのデータブロックに対して夫々設定された属性群を図９にまとめて示す。

【０１９４】

本実施の形態のＯｐｅｎＣＬシステム１００において、ホスト１１０からデバイス１２０に転送したカーネルの引数には、演算対象と演算結果の夫々のデータブロックに対して指定した上記の属性群が含まれる。デバイス１２０の演算制御部２００における属性群格納部２１０は、これらの各属性群を格納して、シナリオ決定部２３０に供する。

【０１９５】

シナリオ決定部２３０は、属性群格納部２１０に格納された各属性群に基づいて、後述する転送シナリオ（以下、単に「シナリオ」ともいう）を決定すると共に、決定されたシナリオに基づいて、演算ユニット１４０による演算、及び演算に伴うデータの転送を制御する。なお、シナリオ決定部２３０は、上記制御に際して、指示セットを演算ユニット１４０に送信することを繰り返す。指示セットは、リード転送指示Ｒ、演算実行指示Ｓ、ライト転送指示Ｗを含む。また、指示セットの１回の送信は、リード転送指示Ｒ、演算実行指示Ｓ、ライト転送指示Ｗの順にこれらの指示を送信することを意味する。

【０１９６】

演算ユニット１４０において、演算制御部２００からのリード転送指示Ｒに応じて、リード転送が行われる。

【０１９７】

次いで、演算制御部２００からの演算実行指示Ｓに応じて、演算ユニット１４０のＰＥ１６２により、リード転送指示Ｒに応じたリード転送でプライベートメモリ１６４／ローカルメモリ１７０に格納されたデータに対して演算処理が行われる。演算結果となる各々のデータは、夫々のＰＥ１６２の対応するプライベートメモリ１６４／ローカルメモリ１７０に格納される。

【０１９８】

そして、演算制御部２００からのライト転送指示Ｗに応じて、プライベートメモリ１６４／ローカルメモリ１７０に格納されているデータ（演算結果）は、ライト転送によりグローバルメモリ１５２へ転送される。

【０１９９】

ここで、シナリオ決定部２３０によるシナリオの決定処理を説明する。

シナリオ決定部２３０は、属性群格納部２１０に格納された各属性群と、演算ユニット

１４０の構成を示すパラメータとに基づいて、シナリオを決定する。このシナリオは、転送方式に該当する。

【０２００】

シナリオ決定部２３０の動作を説明する。説明に当たって、演算処理及びデータブロックについては、図２に示す例を使用する。なお、図２に示す各データブロックは、図９に示すように属性群が設定され、属性群格納部２１０に格納されたとする。また、演算ユニット１４０の構成を示すパラメータの例を、図１０に示す。

【０２０１】

また、上記において各属性を説明する際に、分かりやすいように、１アイテムが１ＰＥに対応するとした。以下の説明においては、１ＰＥが１以上のアイテムに対応可能であるとする。

【０２０２】

シナリオ決定部２３０は、下記の規則に従って転送方式の決定と、決定した転送方式に応じた演算処理の制御を行う。

【０２０３】

＜規則１＞

シナリオ決定部２３０は、まず、全てのデータブロックに対して、共通のワークグループサイズＷＧｓを設定すると共に、階層属性が同一である複数のデータブロックが共通の分割数でサブブロックに分割されるように、データブロックの分割サイズ、反復回数を決定する。

【０２０４】

「分割サイズ」とは、サブブロックのサイズを意味し、「分割数」は、１つのデータブロックを分割して得たサブブロックの数を意味する。１となる分割数は、分割しないことを意味する。また、「反復回数」は、該データブロックの全てのサブブロックの転送に必要な転送回数を意味する。

【０２０５】

データブロックのサイズ、分割サイズ、分割数、ワークグループサイズＷＧｓ、反復回数などは、下記の式（２）～（５）に示す関係を有する。

【０２０６】

データブロックのサイズ＝Ｘ方向サイズＬｘ×Ｙ方向サイズＬｙ　　（２）

分割サイズ

＝サブブロックのＸ方向のサイズＳＢｓｘ×Ｙ方向のサイズＳＢｓｙ　　（３）

分割数＝データブロックのサイズ÷（分割サイズ×ワークグループサイズＷＧｓ）（４）

反復回数

＝｛分割数×（Ｘ方向サイズＬｘ÷ワークグループサイズＷＧｓ）／ＷＧ数　　（５）

【０２０７】

シナリオ決定部２３０は、放送属性が「ＯＦＦ」であるデータブロックのうちの、リード属性が「ＮＯＮＥ」でないデータブロック（リードブロック）またはライト属性が「ＮＯＮＥ」でないデータブロック（ライトブロック）のそれぞれに対して、分割サイズと、余白属性が指定されている場合の余白分の総量（余白属性が指定されていない場合には「０」）との和が、個々のアイテムのプライベートメモリの利用可能容量の合計値を上回らない、かつ、同一の階層属性を持つデータブロックが同一の分割数に分割される制約を満たすように、ワークグループサイズＷＧｓと分割サイズ（正確にはサブブロックのＹ方向サイズＳＢｓｙ）を決定する。なお、上記制約を満たす前提の下、ターゲットＯｐｅｎＣＬデバイスの推奨ワークグループサイズＷＧｓがあれば、該ワークグループサイズＷＧｓを採用し、ワークグループサイズＷＧｓの上限と下限が定められていれば、上限と下限により決められる範囲内にワークグループサイズＷＧｓを決定する。

【０２０８】

＜規則２＞

階層属性が同一であるリードブロックの相対応するサブリードブロックＳＲＢを含むサ

ブリードブロックＳＲＢ群毎を、同時にプライベートメモリまたはローカルメモリ空間へ転送すると共に、ユーザ指定処理を起動する。なお、転送されるサブリードブロックＳＲＢに対して余白属性により余白が指定される場合に、該余白分のデータも転送する。

【０２０９】

＜規則３＞

階層属性が異なるデータブロックの分割数の掛け算となる回数だけ、相対応するサブリードブロックＳＲＢのサブリードブロックＳＲＢ群をプライベートメモリまたはローカルメモリ区間に転送した後、ユーザ指定処理を起動する。

【０２１０】

例えば、階層属性が「１」であるリードブロックの分割数をＮ、階層属性が「２」であるリードブロックの分割数をＭとすると、シナリオ決定部２３０は、ユーザ指定処理をＮ×Ｍ回だけ呼び出すように動作する。また、各回の呼出しに先立ち、サブリードブロックＳＲＢの１種類の組合せをプライベートメモリまたはローカルメモリ区間に転送する。

【０２１１】

シナリオ決定部２３０は、サブブロックの対応付け方法（上述した転送方式における分配方式及び統合方式）を決定してから転送を行う。図１２に示すように、同一の階層のデータブロック同士のサブブロック間と、異なる階層のデータブロック同士のサブブロック間とでは対応付け方法が異なる。

＜規則４＞

ユーザ指定処理の各回の起動後に、演算処理の結果となるサブライトブロックＳＷＢをグローバルメモリ区間に転送する。サブライトブロックＳＷＢの転送は、ユーザ指定処理の起動後に行われ、転送方向が「プライベートメモリ及び／またはローカルメモリ空間からグローバルメモリ空間へ」である点を除き、サブリードブロックＳＲＢのときと同様である。

【０２１２】

図１１は、図９に示すリードブロックＰ、リードブロックＱ、ライトブロックＲに対して指定された属性群と、図１０に示す演算ユニット１４０の構成を示すパラメータとに基づいて、シナリオ決定部２３０により決定されたワークグループサイズＷＧｓ、縦分割サイズ（サブブロックのＹ方向のサイズＳＢｓｙ）、反復回数の例を示す。

【０２１３】

シナリオ決定部２３０は、まず、演算ユニット１４０のワークグループサイズＷＧｓ（３２）を仮決定する。階層属性が「１」である各データブロックの分割数を４とすると、リードブロックＱによるプライベートメモリ占有量は、アイテム毎に、「１２９×２Ｂ」の０．２５８ＫＢとなる。なお、１２９は、「リードブロックＱのＹ方向サイズＬｙ（４８０）／分割数（４）」（縦分割サイズ）に、余白属性により指定されたＹ方向の９を加算して求められた値である。同様に、ライトブロックＲによるプライベートメモリ占有量は、ワークアイテムＷＩ毎に、「（４８０／４）×４Ｂ」の０．７３８ＫＢに求められる。

【０２１４】

リードブロックＱとライトブロックＲによるアイテム毎のプライベートメモリ占有量の和が、演算ユニット１４０の構成を示すパラメータ（図１０）におけるアイテム毎のプライベートメモリ容量（１ＫＢ）より小さいので、規則１が満たされているとして、リードブロックＱとライトブロックＲのワークグループサイズＷＧｓ、分割サイズが決まる。また、反復回数は、上記式（６）に従って「５」と算出される。

【０２１５】

また、階層属性が「２」のリードブロックＰについては、放送属性が「ＯＮ」であるため、ローカルメモリへ割り当てられる。また、そのサイズ（３×３＝９）は、演算ユニット１４０のワークグループＷＧ毎のローカルメモリ容量４ＫＢより小さいので、分割無し（反復回数：１）でローカルメモリに転送される。

【０２１６】

最後に、依存属性が「依存有り」のデータブロックが存在する場合のシナリオ決定部２３０の動作をまとめる。以下の説明が分かりやすいように、ここで、各データブロックを夫々の分割サイズに分割した上で対応付けされたサブブロック群のことを、「サブブロック集合」と呼ぶ。

【０２１７】

シナリオ決定部２３０は、サブブロック集合間でサブブロック同士が依存関係にある場合は、依存元のサブブロック集合に対する処理が行われてから、依存先のサブブロック集合を処理対象とするように、反復の順序を制御する。ここで、サブブロック集合Ｍ０内のある一つのサブブロックＭＸが、サブブロック集合Ｍ１内のある一つのサブブロックＭＹと依存関係にあり、かつＭＸが定義元であるとすると、サブブロック集合Ｍ０とＭ１は依存関係にあると呼び、Ｍ０がＭ１の依存元、Ｍ１がＭ０の依存先と呼ぶ。

【０２１８】

このようにして、ＯｐｅｎＣＬシステム１００は、単独カーネルについての処理を行う。なお、単独カーネルが複数続いた場合には、ＯｐｅｎＣＬシステム１００は、上述した処理をカーネル毎に繰り返す。

【０２１９】

＜関連性のある２つのカーネルの場合＞

この２つのカーネルを夫々第１のカーネルＸと第２のカーネルＹとする。該２つのカーネルに関連性があるとは、先に実行される第１のカーネルＸの演算結果となるライトブロックのうちに継続ライトブロックがあり、該継続ライトブロックが、後続の第２のカーネルＹの演算対象となるリードブロックとして使用されることを意味する。すなわち、第２のカーネルＹのリードブロックのうちに、継続リードブロックが含まれている。

【０２２０】

図１３は、この２つのカーネルのリードブロックとライトブロックを示す。これらのデータブロックのうちに、データブロックＲは継続ライトブロックであり、データブロックＳは継続ライトブロックＲに対応する継続リードブロックである。継続ライトブロックＲと継続リードブロックＳは、同様のデータブロックであるが、第１のカーネルＸのライトブロックと第２のカーネルＹのリードブロックとして異なる属性群が設定されているため、分かりやすいように、別々のデータブロックとして図示すると共に、異なる符号を付与している。

【０２２１】

演算制御部２００の属性群格納部２１０には、図１３に示す各データブロックの夫々の属性群が格納されている。これらの属性群は、単独カーネルの場合に説明した属性群と同様である。

【０２２２】

また、演算制御部２００のカーネル間情報格納部２２０には、カーネル間データブロック依存情報として、第１のカーネルＸの後に第２のカーネルＹが実行されること、第１のカーネルＸのライトブロックＲが、第２のカーネルＹのリードブロックＳとして使用されることを示す情報が格納される。

【０２２３】

例として、第１のカーネルＸが、図２に示す例の演算を行うためのカーネルであるとする。そのため、第１のカーネルＸのリードブロックは、データブロックＰ及びＱであり、第１のカーネルＸのライトブロックは、データブロックＲであり、該データブロックＲは、後続の第２のカーネルＹに使用される継続ライトブロックである。

【０２２４】

また、独立カーネルのときに説明した通りに、リードブロックＰの属性群のうちに、割当属性が「縦優先」であり、余白属性が「Ｘ：０，Ｙ：０」（余白なし）であり、依存属性が「ＮＯＮＥ」（サブリードブロック間に依存性なし）である。

【０２２５】

また、第２のカーネルＹが、第１のカーネルＸのライトブロックＲと同様のデータブロ

ックであるデータブロックＳを演算して、データブロックＵとＶを得るものとする。そのため、第２のカーネルＹのリードブロックは、データブロックＳであり、該データブロックＳは、継続リードブロックである。また、第２のカーネルＹのライトブロックは、データブロックＵ及びＶである。

【０２２６】

図示の例では、継続リードブロックＳの属性群のうちに、、割当属性が「縦優先」であり、余白属性が「Ｘ：１，Ｙ：２」であり、依存属性が「Ｒ２型」（左のサブブロックと、左上のサブブロックと、上のサブブロックに依存する）である。

【０２２７】

すなわち、継続リードブロックＳをサブブロックに分割して転送する際に、該サブブロックと共に、指定されたその左隣の１列のデータと右隣の１列のデータ、及び上隣の２行のデータと下隣の２行のデータも転送するよう、余白属性により規定されている。また、継続リードブロックＳの各サブブロックの転送及び演算の順序については、ターゲットのサブブロックの処理の前に、その依存元の各サブブロック（ここでは、ターゲットのサブブロックの左と、左上と、上との３つのサブブロック）の処理が行われるよう、依存属性により規定されている。

【０２２８】

第２のカーネルＹのライトブロックＵとＶについても属性群が設定されており、それらは、第２のカーネルＹの演算内容などに応じてユーザにより設定されたものであり、ここで詳細な説明と図示を省略する。

【０２２９】

本実施の形態において、演算制御部２００のシナリオ決定部２３０は、継続ライトブロックＲの割当属性と、継続リードブロックＳの割当属性とに基づいて、パイプライン化制御を行うか否かを決定する。具体的には、継続ライトブロックＲと継続リードブロックＳの割当属性が同一であれば、パイプライン化制御を行うと決定し、継続ライトブロックＲと継続リードブロックＳの割当属性が異なれば、パイプライン化制御を行わないと決定する。

【０２３０】

「パイプライン化制御」とは、継続ライトブロックＲがグローバルメモリ１５２に転送されず、プライベートメモリ１６４またはローカルメモリ１７０を介して第２のカーネルＹの実行に供されることにより、第１のカーネルＸと第２のカーネルＹの実行をパイプライン化することである。

【０２３１】

パイプライン化制御を行わないと決定した場合に、シナリオ決定部２３０は、第１のカーネルＸと第２のカーネルＹを夫々単独カーネルとして、まず第１のカーネルＸを実行させ、ライトブロックＲの全てのデータが得られてグローバルメモリ１５２に格納された後に、該ライトブロックＲを第２のカーネルＹのリードブロックＳとして、第２のカーネルＹを実行させる。

【０２３２】

パイプライン化制御を行うと決定した場合に、シナリオ決定部２３０は、パイプライン化制御を実現するために、先に実行される第１のカーネルＸの各データブロックの余白属性と依存属性を変更する。

【０２３３】

具体的には、シナリオ決定部２３０は、第１のカーネルＸのリードブロックＰに対して、その余白属性と依存属性に、継続リードブロックＳの余白属性と依存属性を夫々論理和加算して新たな余白属性と依存属性とする。

【０２３４】

また、リードブロックＱに対しても同様に、シナリオ決定部２３０は、その余白属性と依存属性に、継続リードブロックＳの余白属性と依存属性を夫々論理和加算して新たな余白属性と依存属性とする。

【０２３５】
図１４は、シナリオ決定部２３０により変更後のリードブロックＰとリードブロックＱの属性群を示す。図示のように、変更後、リードブロックＰは、余白属性が元の「Ｘ：０，Ｙ０」から「Ｘ：１，Ｙ：２」になり、依存属性が元の「ＮＯＮＥ」から「Ｒ２型」になっている。
【０２３６】
また、リードブロックＱも、上記変更後、余白属性が元の「Ｘ：０，Ｙ９」から「Ｘ：１，Ｙ：１１」になり、依存属性が元の「ＮＯＮＥ」から「Ｒ２型」になっている。
【０２３７】
そして、シナリオ決定部２３０は、第１のカーネルＸの各ライトブロックについては余白属性と依存属性が変更された属性群を使用し、他の各データブロックについては元の属性群を使用して、各データブロックの転送方式を決定すると共に、決定した転送方式に応じて各リードブロックとライトブロックの転送、及び該転送に対応する並列演算の制御を行う。この際、継続ライトブロックＲの各サブブロックをグローバルメモリ１５２に転送せず、プライベートメモリ１６４またはローカルメモリ１７０を介して第２のカーネルＹの実行に供する点と、第１のカーネルＸと第２のカーネルＹの実行を並列させる点とを除き、単独カーネルの場合と同様である。
【０２３８】
図１５の上部は、第１のカーネルＸと第２のカーネルＹの実行に際して、パイプライン化制御が行われない場合における各データブロックの転送態様を示し、図１５の下部は、第１のカーネルＸと第２のカーネルＹの実行に際して、パイプライン化制御が行われた場合における各データブロックの転送態様を示す。図中実線矢印は、グローバルメモリ１５２を介したデータの受渡しを意味し、点線矢印は、プライベートメモリ１６４またはローカルメモリ１７０を介したデータの受渡しを意味する。
【０２３９】
図１５に示すように、パイプライン化制御が行われない場合において、第１のカーネルＸの演算結果となるライトブロックＲ（継続ライトブロック）は、グローバルメモリ１５２に格納された後に、第２のカーネルＹのリードブロックＳ（継続リードブロック）としてグローバルメモリ１５２からプライベートメモリ１６４またはローカルメモリ１７０に転送され、第２のカーネルＹの実行に供する。
【０２４０】
一方、パイプライン化制御が行われた場合には、継続ライトブロックＲは、グローバルメモリ１５２に転送されず、プライベートメモリ１６４またはローカルメモリ１７０を介して第２のカーネルＹの実行に供されるようになっている。
【０２４１】
なお、プライベートメモリ１６４またはローカルメモリ１７０を介したデータの受渡しについては、例えば、該データを生成する処理と、該データを用いて演算を行う処理とを同一のＰＥに割り当て、該ＰＥは、前の演算により生成してプライベートメモリ１６４またはローカルメモリ１７０に格納したデータをそのまま読み出して後の演算に使えばよい。
【０２４２】
また、ＯｐｅｎＣＬデバイスの各ＰＥがリング状に接続され、各ＰＥが該リング上の隣接するＰＥとデータの受渡しが可能である場合には、前の演算を行ったＰＥは、次の演算を行うＰＥとの間の各ＰＥを介して、前の演算で生成したデータを渡せばよい。
【０２４３】
図１６の上部は、パイプライン化制御が行われない場合における第１のカーネルＸと第２のカーネルＹの実行態様を示し、図１６の下部は、パイプライン化制御が行われた場合における第１のカーネルＸと第２のカーネルＹの実行態様を示す。
【０２４４】
図１６において、斜線で塗り潰された枠は、第１のカーネルＸのためのリード転送の期

間を示し、点で塗り潰された枠は、第１のカーネルＸのためのライト転送の期間を示す。また、縦線で塗り潰された枠は、第２のカーネルＹのためのリード転送の期間を示し、網状に塗り潰された枠は、第２のカーネルＹのためのライト転送の期間を示す。

【０２４５】

また、白い枠は、第１のカーネルＸの実行期間を示し、波線で塗り潰された枠は、第２のカーネルＹの実行期間を示す。

【０２４６】

図１６に示すように、パイプライン化制御が行われない場合において、第１のカーネルＸと第２のカーネルＹは、夫々独立カーネルとして実行され、実行される度に、サブリードブロックのリード転送（グローバルメモリからプライベートメモリ／ローカルメモリへの転送）と、サブライトブロックのライト転送（プライベートメモリ／ローカルメモリからグローバルメモリへの転送）が生じている。

【０２４７】

一方、パイプライン化制御が行われた場合には、第１のカーネルＸの各回の実行により生成されたサブライトブロックのライト転送が行われず、第２のカーネルＹの各回の実行の演算対象となるサブリードブロック（第１のカーネルＸにより生成されたサブライトブロック）のリード転送が行われない。

【０２４８】

さらに、第２のカーネルＹの実行の演算対象となるサブリードブロックのリード転送が行われないため、第２のカーネルＹの実行に並行して、第１のカーネルＸの次の実行のためのリード転送を行うことができる。

【０２４９】

このように、本実施の形態のＯｐｅｎＣＬシステム１００におけるデバイス１２０の演算制御部２００は、第１のカーネルＸと第２のカーネルＹを連続して実行させる際に、第１のカーネルＸの各ライトブロックのうちに、第２のカーネルＹリードブロックとして使用される継続ライトブロックが含まれており、かつ、上記継続ライトブロックの割当属性と、第２のカーネルＹの、該継続ライトブロックに対応するリードブロック（継続リードブロック）割当属性とが一致する場合において、継続ライトブロックのサブブロックがそのまま継続リードブロックのサブブロックとして用いることができるため、パイプライン化制御を行うことにより、第１のカーネルＸと第２のカーネルＹの実行のパイプライン化を実現している。

【０２５０】

具体的には、シナリオ決定部２３０は、シナリオ決定部２３０は、パイプライン化制御に際して、まず、第１のカーネルＸの各リードブロックの余白属性に、第２のカーネルＹの継続リードブロックの余白属性を論理和加算して、第１のカーネルＸの各リードブロックの余白属性を変更すると共に、第１のカーネルＸの各リードブロックの依存属性に、第２のカーネルＹの継続リードブロックの依存属性を論理和加算することにより、第１のカーネルＸの各リードブロックの属性群を変更する。

【０２５１】

そして、第１のカーネルＸと第２のカーネルＹの各データブロックの属性群（第１のカーネルＸのリードブロックについては、変更後の属性群）に基づいて、各データブロックの転送方式を決定し、決定した転送方式に応じて各データブロックの転送、及び該転送に対応する並列演算の制御を行う。この際、継続ライトブロックの各サブブロックをグローバルメモリに転送せず、プライベートメモリまたはローカルメモリを介して第２のカーネルＹの実行に供すると共に、第１のカーネルＸと第２のカーネルＹの実行を並列させることにより、第１のカーネルＸと第２のカーネルＹのパイプライン化を実現する。

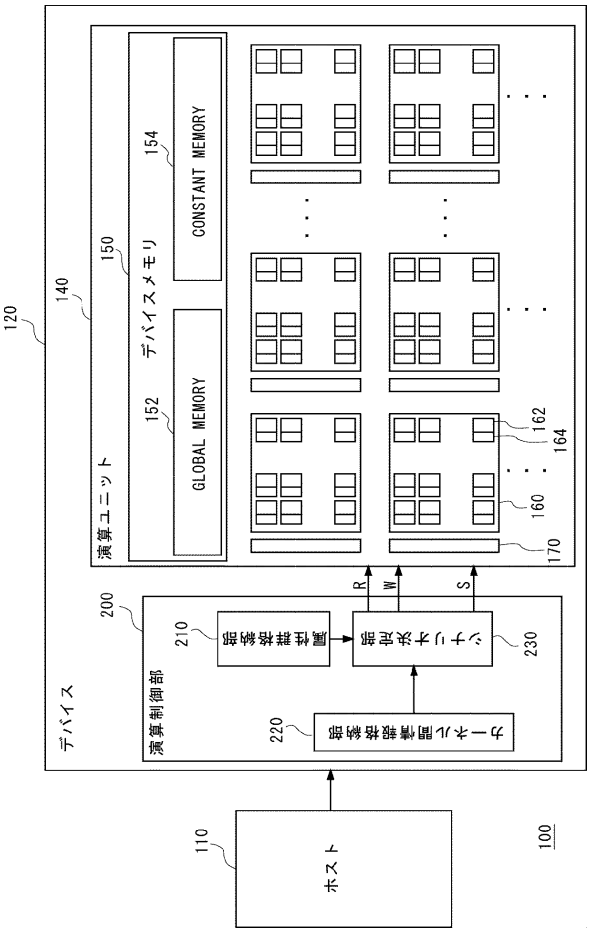
【０２５２】

図１６に示すように、シナリオ決定部２３０のパイプライン化制御の結果、第１のカーネルＸと第２のカーネルＹの実行完了までの時間は、パイプライン化制御が行われない場合より「Ｔ」の分短縮されており、ＯｐｅｎＣＬデバイスの処理が高速になっている。

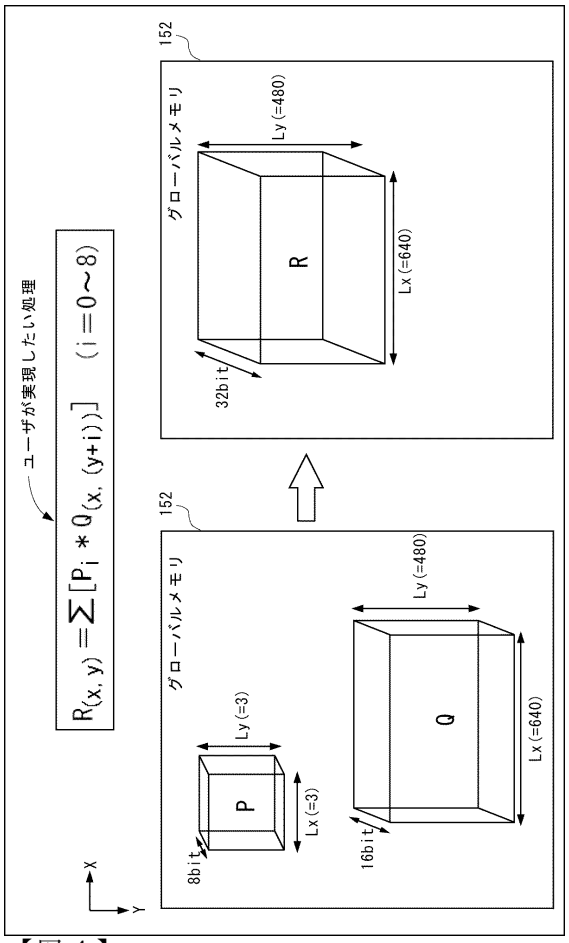
【０２５３】
なお、第１のカーネルＸと第２のカーネルＹの実行をパイプライン化するために、第１のカーネルＸの各リードブロックの余白属性に、第２のカーネルＹの継続リードブロックの余白属性を論理和加算しているため、第１のカーネルＸのサブリードブロックのサイズと、該サブリードブロックと共に転送されるデータのサイズとの総和が、プライベートメモリまたはローカルメモリの容量を超える可能性がある。
【０２５４】
これを回避するために、シナリオ決定部２３０は、第１のカーネルＸの継続ライトブロックの割当属性と、第２のカーネルＹの継続リードブロックの割当属性とが一致し、かつ、変更後の余白属性に基づいて決定した第１のカーネルＸのサブリードブロックの転送時のサイズが予め設定された閾値以下であることを条件に、パイプライン化制御を行うと決定するようにすればよい。なお、この閾値は、プライベートメモリ及び／またはローカルメモリのサイズに応じて予め定めればよい。
【０２５５】
また、分かりやすいように、上記において、関連性のある２つのカーネルのパイプライン化について説明したが、同様の原理で、３以上の関連性のあるカーネルのパイプライン化もできる。
【０２５６】
すなわち、連続して実行されるＮ個（Ｎ：２以上の正数）のカーネルがあり、実行順が２番目以降の各カーネルは、そのリードブロックに、直前に実行されるカーネルのライトブロックが含まれているときに、シナリオ決定部２３０は、各Ｋ番目のカーネル（１≦Ｋ≦Ｎ－１）の継続ライトブロックの割当属性と、該継続ライトブロックに対応する、（Ｋ＋１）番目のカーネルの継続リードブロックの割当属性とが一致する場合にのみ、パイプライン化制御を行うと決定する。
【０２５７】
パイプライン化制御を行うと決定した場合に、シナリオ決定部２３０は、１番目のカーネルの各リードブロックの余白属性に、２番目以降の各カーネルの継続リードブロックの余白属性を論理和加算すると共に、１番目のカーネルの各リードブロックの依存属性に、２番目以降の各カーネルの継続リードブロックの依存属性を論理和加算する。
【０２５８】
その後、シナリオ決定部２３０は、各データブロックの転送方式の決定、及び決定した転送方式に応じて各データブロックの転送、及び該転送に対応する並列演算の制御に際して、継続ライトブロックの各サブブロックをグローバルメモリに転送せず、プライベートメモリまたはローカルメモリを介して次のカーネルの実行に供すると共に、各カーネルの実行を並列させることにより、これらのカーネルのパイプライン化を実現する。
【０２５９】
パイプライン化制御の対象となるカーネルの数が多いほど、１番目のカーネルのサブリードブロックの転送時のサイズが大きくなる。１番目のカーネルのサブリードブロックの転送時のサイズがプライベートメモリまたはローカルメモリの容量を超えることを避けるために、シナリオ決定部２３０は、上記Ｎ個のカーネルを、連続して実行される１つ以上のカーネルにより構成された複数のグループに分け、グループ毎にパイプライン化制御を行うようにすればよい。グループ分けに際しては、変更後の余白属性に基づいて決定した、該グループ内に最も先に実行されるカーネルのサブリードブロックの転送時のサイズが予め設定された閾値以下になるようにすればよい。
【０２６０】
以上、本発明者によってなされた発明を実施の形態に基づき具体的に説明したが、本発明は既に述べた実施の形態に限定されるものではなく、その要旨を逸脱しない範囲において種々の変更が可能であることはいうまでもない。
【符号の説明】
【０２６１】

１０　ＯｐｅｎＣＬシステム
１２　ホスト
１４　ＯｐｅｎＣＬデバイス
１６　ＣＵ
１８　ＰＥ
２０　プライベートメモリ
２２　ローカルメモリ
２４　キャッシュ
３０　デバイスメモリ
３２　グローバルメモリ
３４　コンスタントメモリ
５０　逐次プロセッサ
５２　ＰＥ
５４　プライベートメモリ
５６　グローバルメモリ
５８　キャッシュ制御機構
１００　ＯｐｅｎＣＬシステム
１１０　ホスト
１２０　デバイス
１４０　演算ユニット
１５０　デバイスメモリ
１５２　グローバルメモリ
１５４　コンスタントメモリ
１６０　ＣＵ
１６２　ＰＥ
１６４　プライベートメモリ
１７０　ローカルメモリ
２００　演算制御部
２１０　属性群格納部
２２０　カーネル間情報格納部
２３０　シナリオ決定部
Ｌｘ　データブロックのＸ方向サイズ
Ｌｙ　データブロックのＹ方向サイズ
ＳＢｓｘ　サブブロックのＸ方向サイズ
ＳＢｓｙ　サブブロックのＹ方向サイズ
ＳＲＢ　サブリードブロック
ＳＷＢ　サブライトブロック
ＷＧ　ワークグループ
ＷＧｓ　ワークグループサイズ
ＷＩ　ワークアイテム
Ｒ　リード転送指示
Ｗ　ライト転送指示
Ｓ　演算実行指示

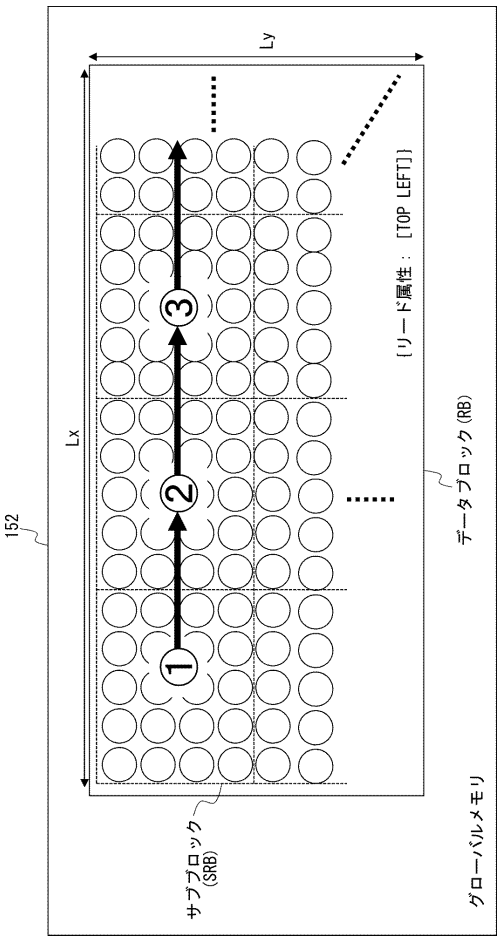
【図１】


100
ホスト
110
デバイス
演算制御部
200
属性群格納部
210
シェーダ実行部
230
カーネル情報格納部
220
R
W
S
演算ユニット
140
デバイスメモリ
150
GLOBAL MEMORY
152
CONSTANT MEMORY
154
160
162
164
170

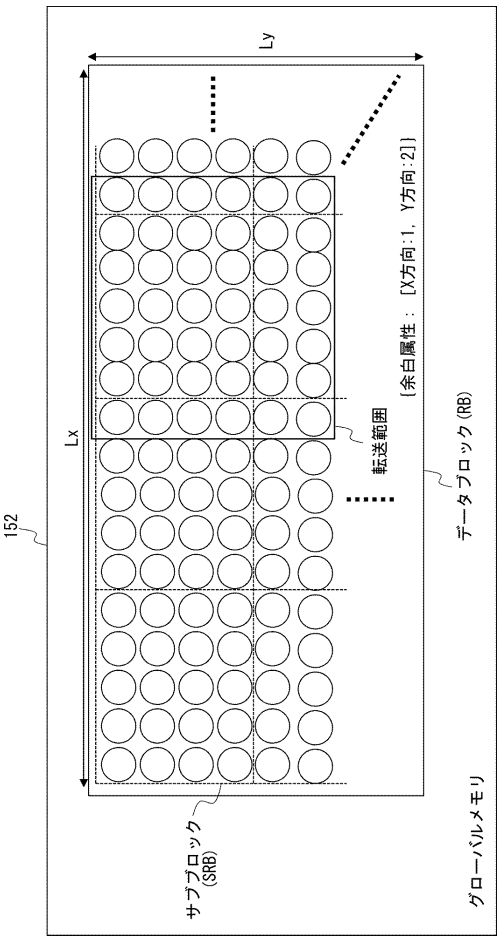
【図２】


ユーザが実現したい処理
$R_{(x,y)} = \sum [P_i * Q_{(x,(y+i))}] \quad (i=0\sim8)$
グローバルメモリ
152
Ly (=3)
Lx (=3)
8bit
P
Ly (=480)
Lx (=640)
16bit
Q
32bit
R
X
Y

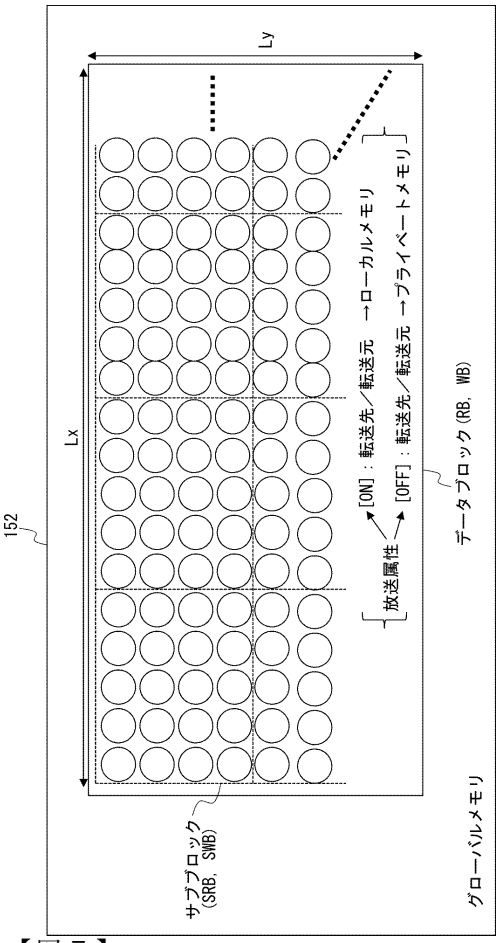
【図３】


152
Lx
Ly
①
②
③
[リード属性： [TOP LEFT]]
データブロック(RB)
サブブロック
(SRB)
グローバルメモリ

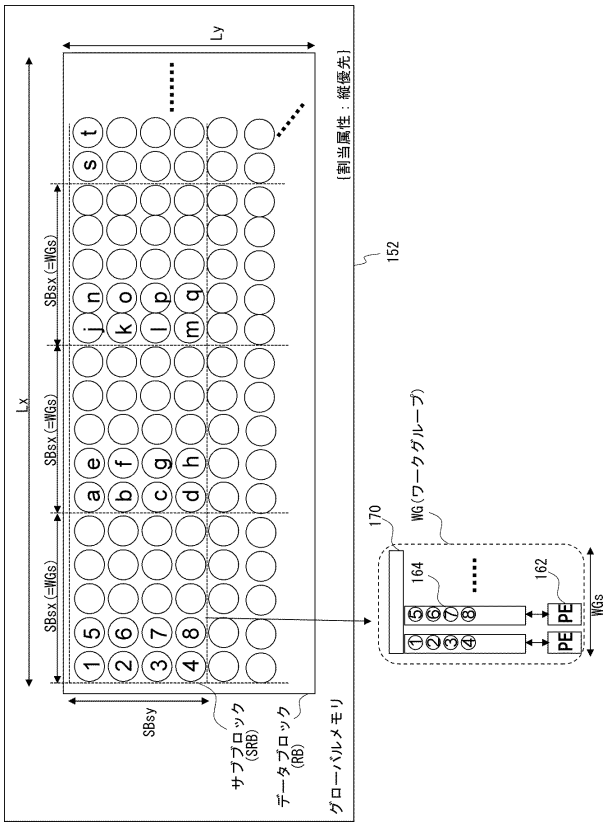
【図４】


152
Lx
Ly
転送範囲
[余白属性： [X方向:1, Y方向:2]]
データブロック(RB)
サブブロック
(SRB)
グローバルメモリ

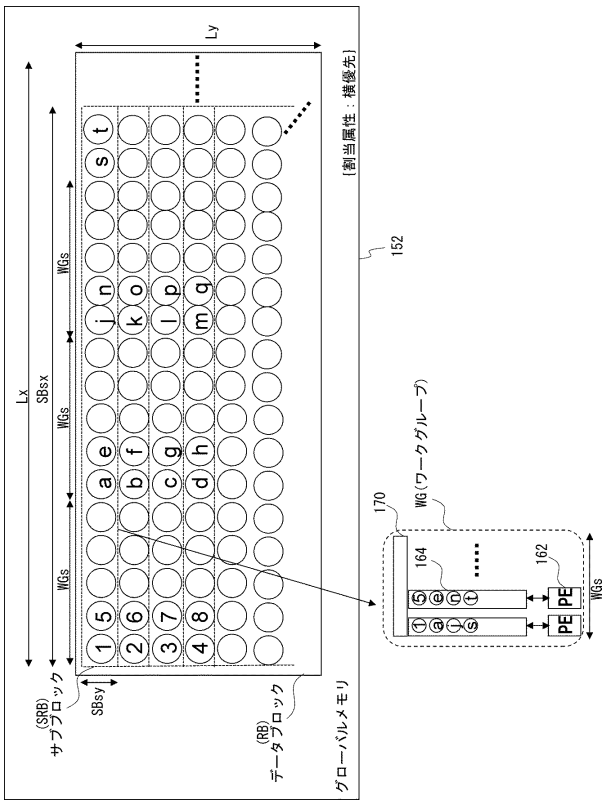
【図５】


152
Lx
Ly
[ON]：転送先／転送元　→ローカルメモリ
放送属性
[OFF]：転送先／転送元 →プライベートメモリ
データブロック(RB, WB)
サブブロック(SRB, SWB)
グローバルメモリ

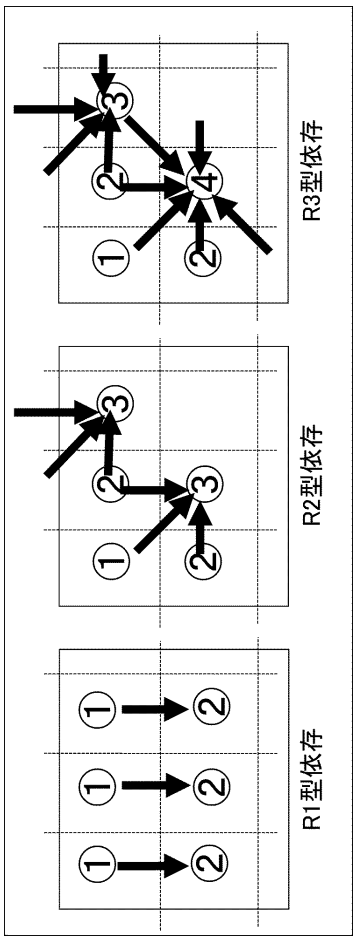
【図６】


Lx
Ly
SBsx(=WGs)
SBsy
サブブロック(SRB)
データブロック(RB)
グローバルメモリ
[割当属性：縦優先]
152
170
WG(ワークグループ)
164
162
PE
WGs

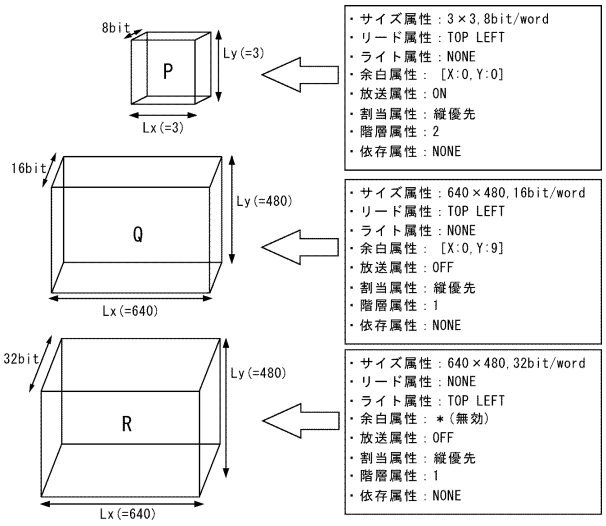
【図７】


Lx
Ly
SBsx
WGs
SBsy
サブブロック(SRB)
データブロック(RB)
グローバルメモリ
[割当属性：横優先]
152
170
WG(ワークグループ)
164
162
PE

【図８】


R1型依存
R2型依存
R3型依存

【図９】

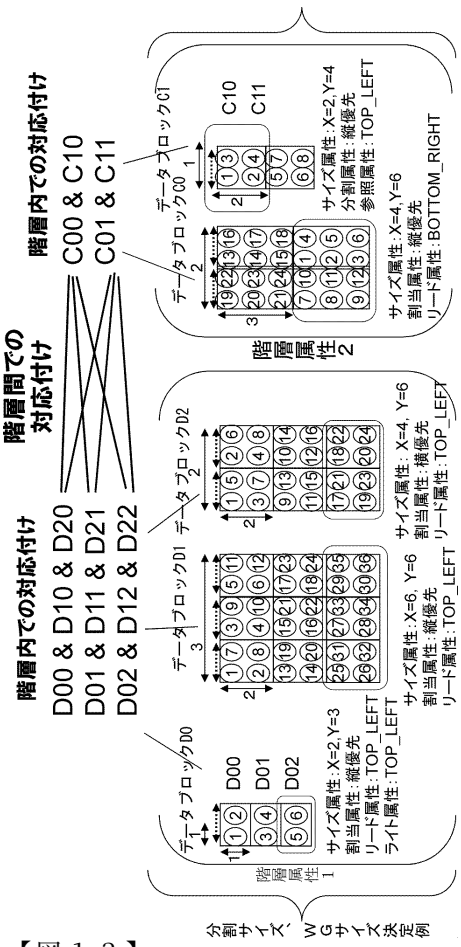


【図１０】

演算ユニットの構成を示すパラメータの例

- ワークアイテム(WI)毎のプライベートメモリ容量：1KB
- ワークグループ(WG)毎のローカルメモリ容量：4KB
- WGサイズ(WGs)の最大値：32
- WG数の最大値：16

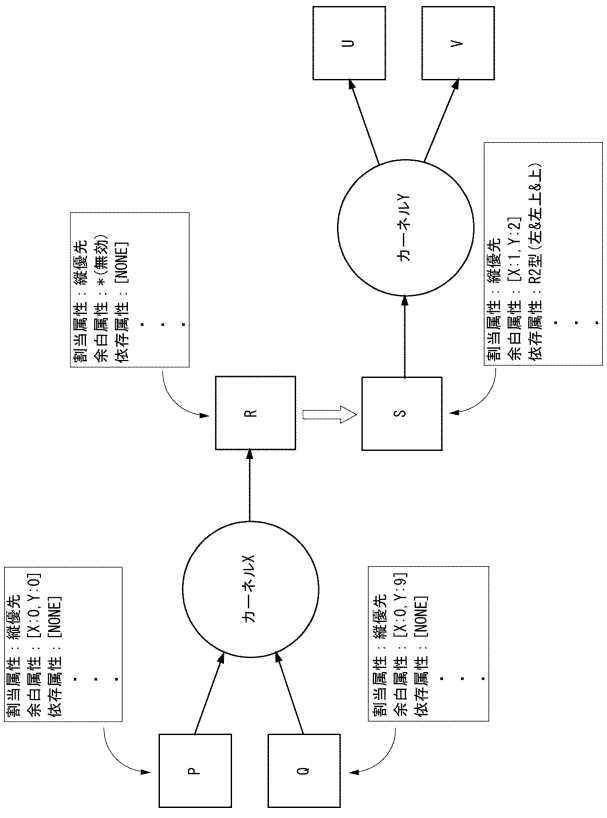
【図１１】



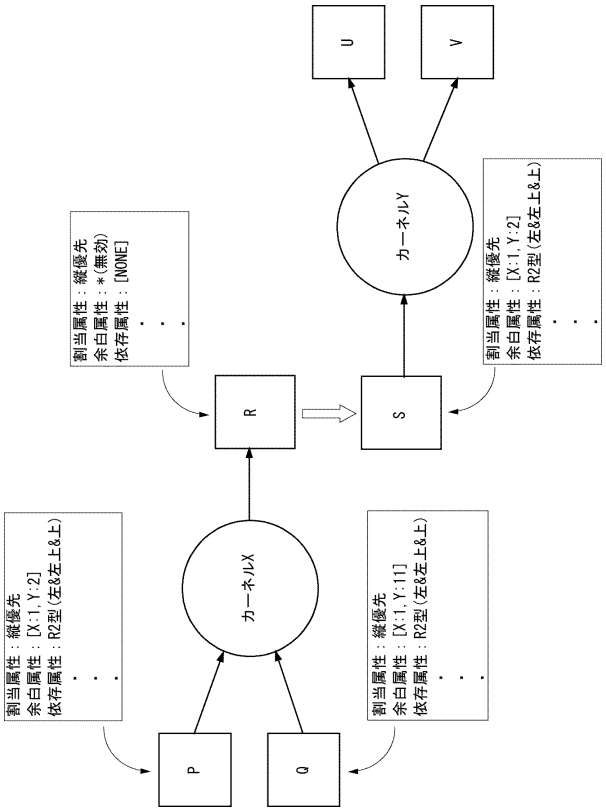
【図１２】

・ワークグループサイズWGs＝32
＜階層１＞
・分割数が4である場合の各データブロックのWI毎のプライベートメモリ占有量(サブブロックのY方向サイズ Sbsy)
・リードブロックQ：WI毎129×2B (=0.258KB/WI)
・ライトブロックR：WI毎120×4B (=0.480KB/WI)
・反復回数：5(=**分割数**4×(640÷WG**サイズ**32)÷WG**数**16)
＜階層２＞
・分割数：1
・リードブロックP
「分割無し」でローカルメモリへ
・反復回数：1

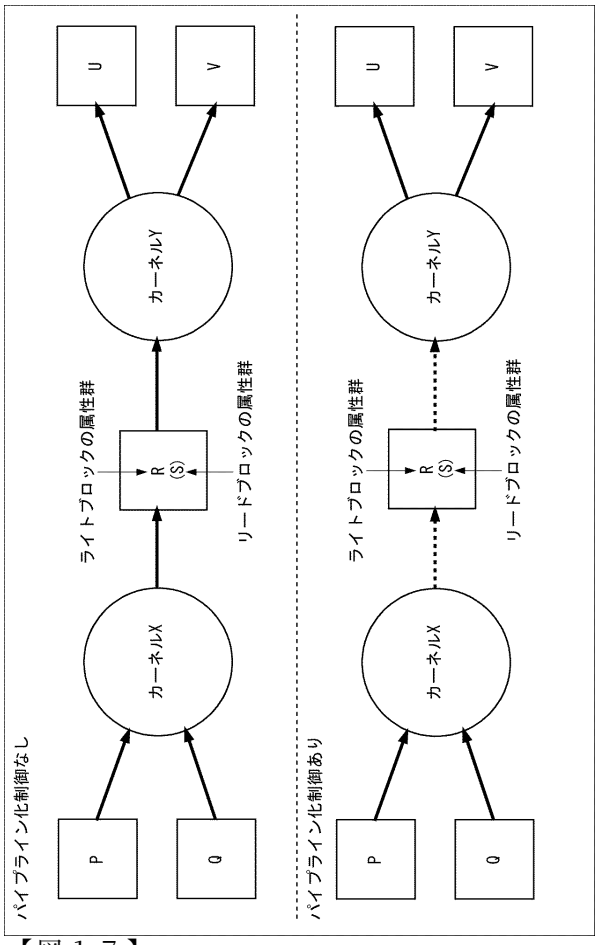
【図１３】



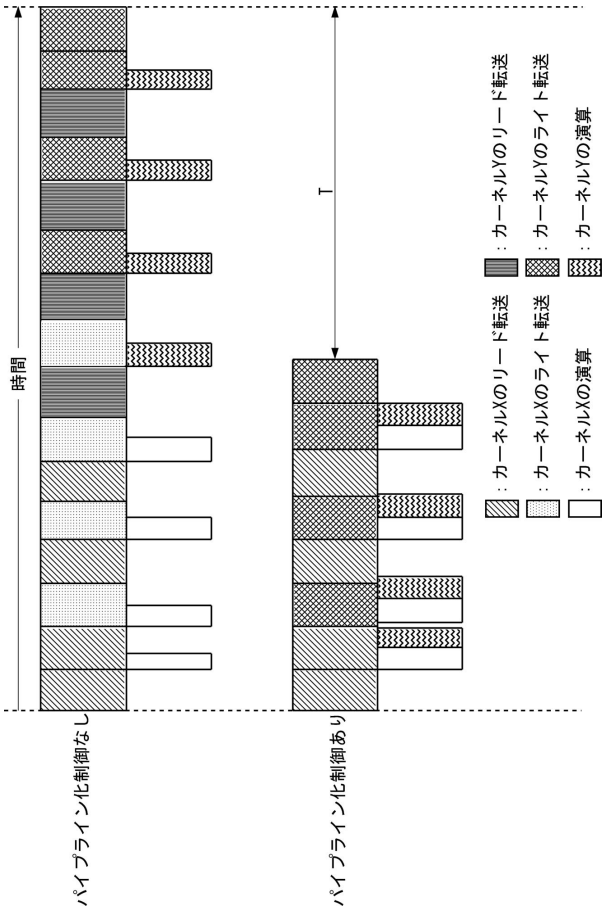
【図１４】



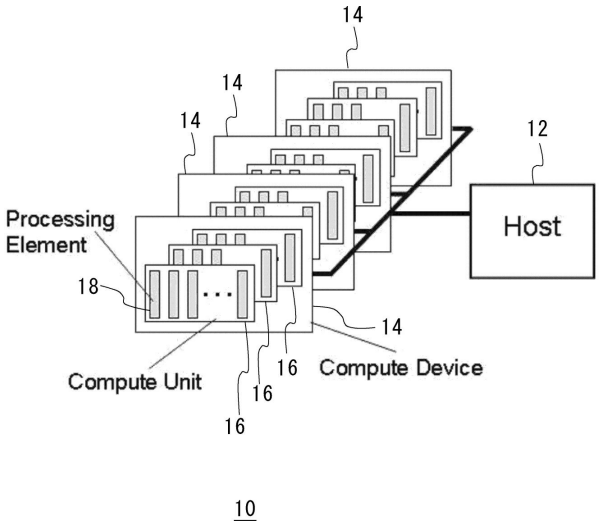
【図１５】



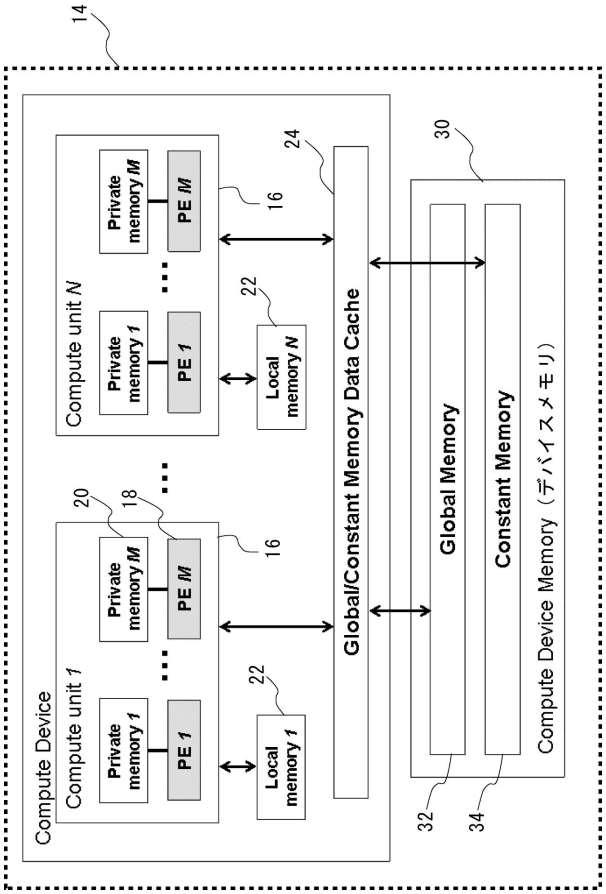
【図１６】



【図１７】



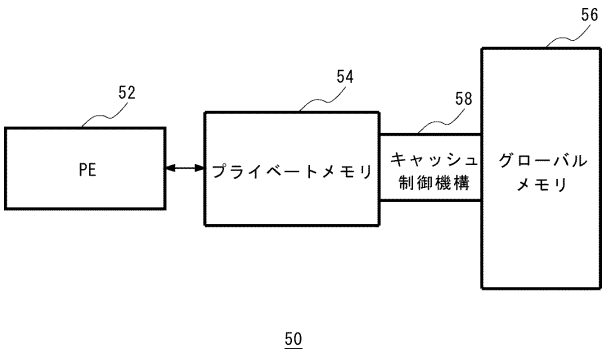
【図１８】



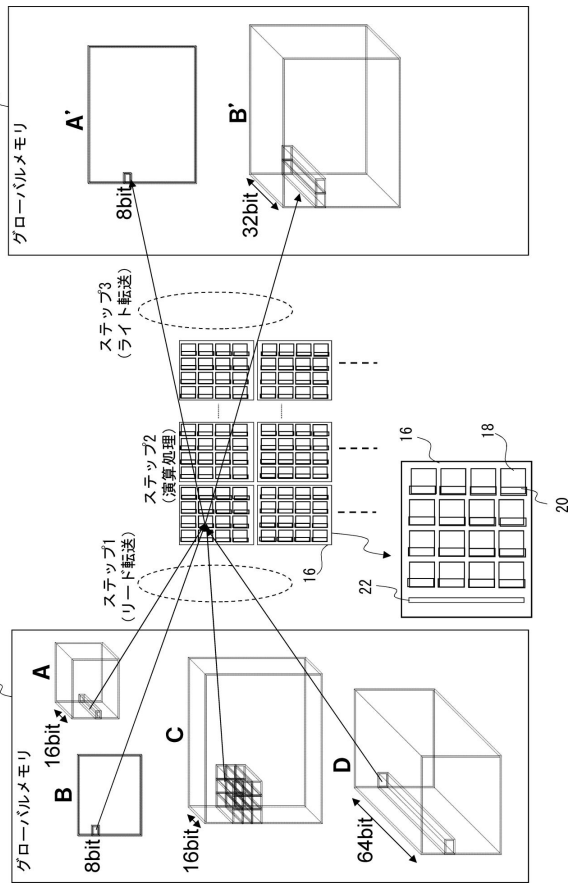
【図１９】

		Global	Constant	Local	Private
Host		Dynamic allocation Read / Write access	Dynamic allocation Read / Write access	Dynamic allocation No access	No allocation No access
Kernel		No allocation Read / Write access	Static allocation Read-only access	Static allocation Read / Write access	Static allocation Read / Write access

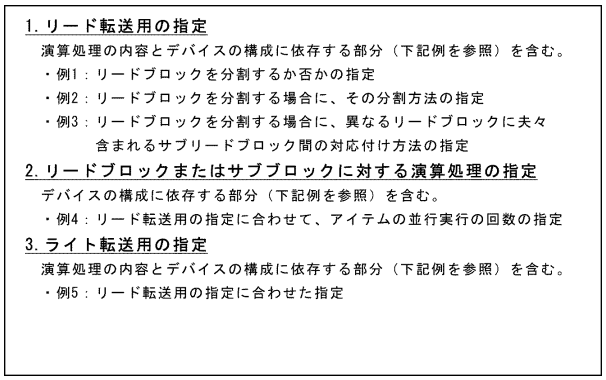
【図２０】



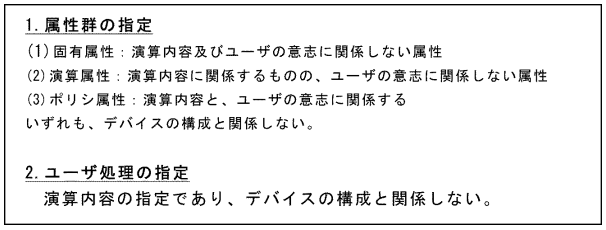
【図２１】



【図２２】


1.リード転送用の指定
演算処理の内容とデバイスの構成に依存する部分（下記例を参照）を含む。
・例1：リードブロックを分割するか否かの指定
・例2：リードブロックを分割する場合に、その分割方法の指定
・例3：リードブロックを分割する場合に、異なるリードブロックに夫々
含まれるサブリードブロック間の対応付け方法の指定
2.リードブロックまたはサブブロックに対する演算処理の指定
デバイスの構成に依存する部分（下記例を参照）を含む。
・例4：リード転送用の指定に合わせて、アイテムの並行実行の回数の指定
3.ライト転送用の指定
演算処理の内容とデバイスの構成に依存する部分（下記例を参照）を含む。
・例5：リード転送用の指定に合わせた指定

【図２３】


1.属性群の指定
(1)固有属性：演算内容及びユーザの意志に関係しない属性
(2)演算属性：演算内容に関係するものの、ユーザの意志に関係しない属性
(3)ポリシ属性：演算内容と、ユーザの意志に関係する
いずれも、デバイスの構成と関係しない。
2.ユーザ処理の指定
演算内容の指定であり、デバイスの構成と関係しない。

フロントページの続き

(56)参考文献 特開２０１３－２５５４７（ＪＰ，Ａ）
特開２００８－２１７１３４（ＪＰ，Ａ）
佐藤 裕幸，グラフィックス処理用プロセッサＧＰＵによるＳＡＲ画像再生処理の高速化，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，２０１０年 4月１５日，第110巻,第８号，pp.7-12

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １５／８０
Ｇ０６Ｆ ９／５０
Ｇ０６Ｆ １２／０８
Ｇ０６Ｆ １２／０８０２