(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2014-229173
(P2014-229173A)
(43)公開日　平成26年12月8日(2014.12.8)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/44	(2006.01)	G06F	9/06	620A	5B376

審査請求 未請求　請求項の数 5　ＯＬ　（全 28 頁）

(21)出願番号　特願2013-109741 (P2013-109741)
(22)出願日　平成25年5月24日 (2013.5.24)

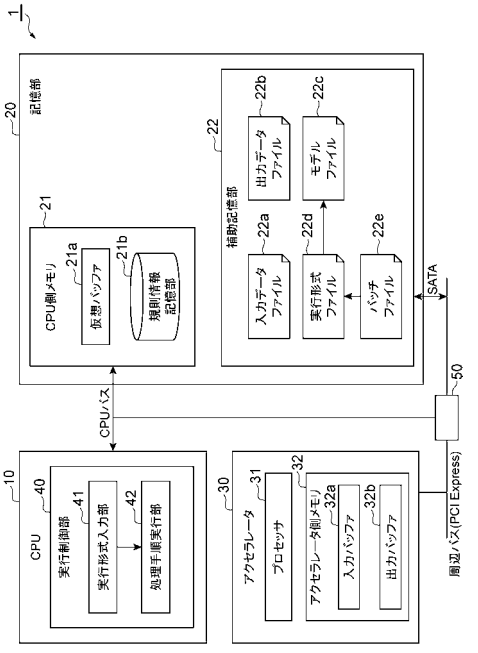
（出願人による申告）平成２４年度、独立行政法人科学技術振興機構、戦略的創造研究推進事業（個人型研究（さきがけ））に関する委託研究、産業技術力強化法第１９条の適用を受ける特許出願

(71)出願人　504171134
国立大学法人　筑波大学
茨城県つくば市天王台一丁目１番１
(74)代理人　100088155
弁理士　長谷川　芳樹
(74)代理人　100113435
弁理士　黒木　義樹
(74)代理人　100124800
弁理士　諏澤　勇司
(74)代理人　100128107
弁理士　深石　賢治
(72)発明者　山際　伸一
茨城県つくば市天王台一丁目１番１　国立大学法人筑波大学内

Ｆターム(参考)　5B376 BC73

(54)【発明の名称】アクセラレータ処理実行装置、及びアクセラレータ処理実行プログラム

(57)【要約】
【課題】アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を向上させる。
【解決手段】アクセラレータ処理実行装置１は、ＣＰＵ１０と、記憶部２０と、アクセラレータプログラムを実行するアクセラレータ３０と、を備え、ＣＰＵ１０は、アクセラレータに関する処理の実行を制御する実行制御部４０を有し、実行制御部４０は、アクセラレータプログラムと、アクセラレータプログラムの出力データ毎にアクセラレータ側メモリ３２に確保される出力バッファ３２ｂと記憶部２０における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力部４１と、アクセラレータに関する処理の手順を決定するために実行形式とは別に予め定められている規則と実行情報とに基づいて決定した処理手順を実行する処理手順実行部４２と、を有する。
【選択図】図１



【特許請求の範囲】
【請求項１】
ＣＰＵと、
前記ＣＰＵに接続され、データを記憶する第１の記憶手段と、
前記ＣＰＵに接続され、前記第１の記憶手段とは異なる、データを記憶する第２の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備え、
前記ＣＰＵは、前記アクセラレータに関する処理の実行を制御する実行制御手段を有し、
前記実行制御手段は、
前記アクセラレータプログラムと、前記アクセラレータプログラムが実行されて出力される出力データ毎に前記第２の記憶手段に確保される出力バッファと前記第１の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、
前記アクセラレータに関する処理の手順を決定するために前記実行形式とは別に予め定められている規則と前記実行情報とに基づいて決定した処理手順を実行する処理手順実行手段と、を有し、
前記処理手順実行手段は、
前記アクセラレータに前記アクセラレータプログラムを送信するプログラム送信処理と、
前記第２の記憶手段に前記出力バッファを確保する出力バッファ確保処理と、
前記プログラム送信処理及び前記出力バッファ確保処理よりも後に、前記アクセラレータに前記アクセラレータプログラムを実行させるプログラム実行処理と、
前記プログラム実行処理により前記出力バッファに出力された前記出力データを、当該出力バッファに対応する前記第１の記憶手段における出力先に送信する出力データ送信処理と、を実行する、
アクセラレータ処理実行装置。
【請求項２】
前記実行情報には、前記アクセラレータプログラムに対する入力データ毎に前記第１の記憶手段における記憶場所と前記第２の記憶手段に確保される入力バッファとを対応付ける入力データ対応情報が含まれ、
前記処理手順実行手段は、
前記プログラム実行処理よりも前に、前記第２の記憶手段に前記入力バッファを確保する入力バッファ確保処理と、
前記入力バッファ確保処理よりも後に、前記記憶場所に記憶されている前記入力データを、当該記憶場所に対応する前記入力バッファに送信する入力データ送信処理と、を更に実行し、
前記処理手順実行手段は、前記入力データ送信処理よりも後に前記プログラム実行処理を実行する、
請求項１記載のアクセラレータ処理実行装置。
【請求項３】
前記入力データ対応情報に含まれる前記第１の記憶手段における記憶場所、又は前記出力データ対応情報に含まれる前記第１の記憶手段における出力先として、メモリ上の共有領域が指定される、
請求項２記載のアクセラレータ処理実行装置。
【請求項４】
前記実行情報には、前記入力バッファ及び前記出力バッファの組み合わせと入替回数とを示すスワップ情報が含まれ、
前記処理手順実行手段は、前記プログラム実行処理において、前記アクセラレータプログラムの１回の実行が完了する毎に前記入力バッファと前記出力バッファとを前記組み合わせに基づいて入れ替えた後に前記アクセラレータプログラムを実行する処理を、前記入

替回数だけ繰り返す、
請求項２又は３記載のアクセラレータ処理実行装置。
【請求項５】
ＣＰＵと、
前記ＣＰＵに接続され、データを記憶する第１の記憶手段と、
前記ＣＰＵに接続され、前記第１の記憶手段とは異なる、データを記憶する第２の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備えるコンピュータに、
前記アクセラレータに関する処理の実行を制御する実行制御手段を実現させるためのアクセラレータ処理実行プログラムであって、
前記実行制御手段は、
前記アクセラレータプログラムと、前記アクセラレータプログラムが実行されて出力される出力データ毎に前記第２の記憶手段に確保される出力バッファと前記第１の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、
前記アクセラレータに関する処理の手順を決定するために前記実行形式とは別に予め定められている規則と前記実行情報とに基づいて決定される処理手順を実行する処理手順実行手段と、を有し、
前記処理手順実行手段は、
前記アクセラレータに前記アクセラレータプログラムを送信するプログラム送信処理と、
前記第２の記憶手段に前記出力バッファを確保する出力バッファ確保処理と、
前記プログラム送信処理及び前記出力バッファ確保処理よりも後に、前記アクセラレータに前記アクセラレータプログラムを実行させるプログラム実行処理と、
前記プログラム実行処理により前記出力バッファに出力された前記出力データを、当該出力バッファに対応する前記第１の記憶手段における出力先に送信する出力データ送信処理と、を実行する、
アクセラレータ処理実行プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、特にＣＰＵに接続されたアクセラレータハードウェアに処理を実行させるアクセラレータ処理実行装置、及びアクセラレータ処理実行プログラムに関する。
【背景技術】
【０００２】
ＧＰＵ（GraphicsProcessing Unit）がもつ潜在的な超並列性を生かした計算技法が2006年くらいから市場に台頭するようになり、ＧＰＵを使ったハイパフォーマンスコンピューティング分野が急速に発達している。最近では、ＧＰＵがＣＰＵ（Central Processing Unit：中央処理装置）の周辺デバイスとして動作するアーキテクチャモデルに基づいて、ＣＰＵで動作するホストプログラム（ＣＰＵ側プログラム）が、ＧＰＵをアクセラレータハードウェアとして扱い、アクセラレータに実行させるアクセラレータ側プログラムをＣＰＵ側からアクセラレータ側に送信（ダウンロード）し、アクセラレータに実行させるプログラミングスタイルが定着している。このアクセラレータ側プログラムは、ＧＰＵハードウェアのアーキテクチャの超並列性を引出すためにデータはストリームで提供されるといった理念に基づき、設計されている。
【０００３】
アクセラレータにアクセラレータ側プログラムを実行させるためのプラットフォームとして、本発明者らが開発したプラットフォームが知られている（下記非特許文献１参照）。
【先行技術文献】

【非特許文献】
【０００４】
【非特許文献１】Shinichi Yamagiwa, leonel Sousa, "Caravela: A NovelStream-Based Distributed Computing Environment," Computer , vol.40, no.5,pp.70-77, May 2007.
【発明の概要】
【発明が解決しようとする課題】
【０００５】
ところで、アクセラレータ側プログラムを作成する場合には、ＣＰＵとアクセラレータとはアーキテクチャが異なるため、これまでのフォンノイマン型のＣＰＵ側プログラムのプログラムコードと、ストリームコンピューティングに従ったアクセラレータ側プログラムのプログラムコードを別々に作成し、異なるコンパイラによってバイナリを２種類作成する必要がある。即ち、ＣＰＵ側で実行させるバイナリとアクセラレータ側で実行させるバイナリとを作成する必要がある。ここで、ＣＰＵ側プログラムには、ＣＰＵとアクセラレータとの間でのデータの受け渡し等に関する詳細な手続きを記述しなければならない。
【０００６】
上記非特許文献１に示したプラットフォームでは、入力データストリームと定数パラメータとを入力して、プログラム（カーネルプログラム）を実行し、出力データストリームを得る、といった流れを規定したｆｌｏｗ－ｍｏｄｅｌを定義することで、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム作成にかかる負担を軽減している。しかしながら、このプラットフォームを用いたとしても、データ入出力（ＣＰＵとアクセラレータとの間でのデータの受け渡し）等について詳細に記述したＣＰＵ側プログラムが、やはり必要となる。
【０００７】
以上述べたように、アクセラレータを用いたアプリケーションを開発するプログラマは、アプリケーション開発の際に、アプリケーションの本来の計算処理（アクセラレータに実行させたい計算処理）とは直接的には関係のないＣＰＵ側プログラム（ＣＰＵとアクセラレータとの間でのデータの受け渡しの制御等を実行するプログラム）を作成する必要がある。このことは、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を低下させる一因となっている。
【０００８】
本発明は、上記の課題に鑑みてなされたものであり、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を向上させることができるアクセラレータ処理実行装置、及びアクセラレータ処理実行プログラムを提供することを目的とする。
【課題を解決するための手段】
【０００９】
本発明に係るアクセラレータ処理実行装置は、ＣＰＵと、ＣＰＵに接続され、データを記憶する第１の記憶手段と、ＣＰＵに接続され、第１の記憶手段とは異なる、データを記憶する第２の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備え、ＣＰＵは、アクセラレータに関する処理の実行を制御する実行制御手段を有し、実行制御手段は、アクセラレータプログラムと、アクセラレータプログラムが実行されて出力される出力データ毎に第２の記憶手段に確保される出力バッファと第１の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、アクセラレータに関する処理の手順を決定するために実行形式とは別に予め定められている規則と実行情報とに基づいて決定した処理手順を実行する処理手順実行手段と、を有し、処理手順実行手段は、アクセラレータにアクセラレータプログラムを送信するプログラム送信処理と、第２の記憶手段に出力バッファを確保する出力バッファ確保処理と、プログラム送信処理及び出力バッファ確保処理よりも後に、アクセラレータにアクセラレータプログラムを実行させるプログラム実行処理と、プログラム実行処理により出力バッファに出力された出力データを、当該出力バッ

ファに対応する第１の記憶手段における出力先に送信する出力データ送信処理と、を実行する。

【００１０】

上記アクセラレータ処理実行装置では、実行制御手段が、アクセラレータプログラムと、出力データのアクセラレータ側での保存先（出力バッファ）とＣＰＵ側での保存先（第１の記憶手段における出力先）とを対応付ける出力データ対応情報を含む実行情報を所定の形式（例えばＸＭＬ形式等）で記述した実行形式を入力する。そして、ＣＰＵが備える実行制御手段は、実行形式及び予め定めた規則に基づいてアクセラレータに関する処理手順を決定し、実行する。これにより、ＣＰＵ側とアクセラレータ側との間の出力データの受け渡しを含む各処理手順が、実行制御手段によって適切に決定及び実行されるため、アクセラレータを用いるアプリケーションを開発するプログラマ（以下、単に「プログラマ」と表記する）は、ＣＰＵ側とアクセラレータ側との間のデータの受け渡し等の処理手順を詳細に規定したプログラム（ＣＰＵ側プログラム）を作成する手間から解放される。即ち、上記アクセラレータ処理実行装置によれば、アクセラレータプログラムをアクセラレータに実行させるためには、上記実行形式を用意するだけでよいため、プログラマのプログラム生産性を向上させることができる。

【００１１】

また、上記アクセラレータ処理実行装置では、実行情報には、アクセラレータプログラムに対する入力データ毎に第１の記憶手段における記憶場所と第２の記憶手段に確保される入力バッファとを対応付ける入力データ対応情報が含まれ、処理手順実行手段は、プログラム実行処理よりも前に、第２の記憶手段に入力バッファを確保する入力バッファ確保処理と、入力バッファ確保処理よりも後に、上記記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファに送信する入力データ送信処理と、を更に実行し、処理手順実行手段は、入力データ送信処理よりも後にプログラム実行処理を実行してもよい。

【００１２】

上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの実行に入力データが必要とされる場合であっても、入力データ対応情報を含む実行形式を入力した実行制御手段によって、ＣＰＵ側とアクセラレータ側との間の入力データの受け渡しに関する処理が適切に決定及び実行される。従って、プログラマは、入力データ対応情報を含む実行形式を用意するだけでよく、ＣＰＵ側プログラムを作成する手間から解放される。

【００１３】

また、上記アクセラレータ処理実行装置では、入力データ対応情報に含まれる第１の記憶手段における記憶場所、又は出力データ対応情報に含まれる第１の記憶手段における出力先として、メモリ上に確保された共有領域が指定されてもよい。

【００１４】

上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの入力データ又は出力データの受け渡しを、ファイル（例えばＣＳＶファイル等）を介することなく、仮想バッファ（メモリ上に確保された共有領域）を介して実行することができ、データの受け渡しにおいてファイルＩ／Ｏを排除することができる。例えば、複数のアクセラレータプログラムを連続して実行させ、先に実行させたアクセラレータプログラムにより出力された出力データを、後に実行させるアクセラレータプログラムの入力データとして用いたい場合等がある。この場合、先に実行させるアクセラレータプログラムを格納した実行形式に記述する出力データの出力先と、後に実行させるアクセラレータプログラムを格納した実行形式に記述する入力データの記憶場所とを、同一の仮想バッファに指定すれば、２つのアクセラレータプログラム間でのデータの受け渡しは、仮想バッファを介して実現されることとなる。これにより、データの受け渡しにおいてファイルＩ／Ｏを排除して、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。

【００１５】

また、上記アクセラレータ処理実行装置では、実行情報には、入力バッファ及び出力バッファの組み合わせと入替回数とを示すスワップ情報が含まれ、処理手順実行手段は、プログラム実行処理において、アクセラレータプログラムの１回の実行が完了する毎に入力バッファと出力バッファとを上記組み合わせに基づいて入れ替えた後にアクセラレータプログラムを実行する処理を、入替回数だけ繰り返してもよい。

【００１６】

例えば、高速フーリエ変換（ＦＦＴ）のアルゴリズム（いわゆるバタフライ演算）等、アクセラレータプログラムの出力データを入力データとして同一のアクセラレータプログラムを再実行する処理を繰り返したいような場合がある。この場合、上記アクセラレータ処理実行装置によれば、アクセラレータプログラムの１回の実行が完了する毎に、例えばポインタの入替等により入力バッファと出力バッファとを入れ替えてアクセラレータプログラムを再実行することができるため、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。具体的には、出力バッファに出力されたアクセラレータプログラムの出力データを一旦ＣＰＵ側の記憶領域（第１の記憶手段）に移動させてから再度アクセラレータ側に入力データとして入力するといった無駄なデータ受け渡し処理を省略することができる。

【００１７】

ところで、本発明は、上記のようにアクセラレータ処理実行装置の発明として記述できる他に、以下のようにアクセラレータ処理実行プログラムの発明としても記述することができる。これはカテゴリが異なるだけで、実質的に同一の発明であり、同様の作用及び効果を奏する。

【００１８】

即ち、本発明に係るアクセラレータ処理実行プログラムは、ＣＰＵと、ＣＰＵに接続され、データを記憶する第１の記憶手段と、ＣＰＵに接続され、第１の記憶手段とは異なる、データを記憶する第２の記憶手段を有し、アクセラレータプログラムを実行するアクセラレータと、を備えるコンピュータに、アクセラレータに関する処理の実行を制御する実行制御手段を実現させるためのアクセラレータ処理実行プログラムであって、実行制御手段は、アクセラレータプログラムと、アクセラレータプログラムが実行されて出力される出力データ毎に第２の記憶手段に確保される出力バッファと第１の記憶手段における出力先とを対応付ける出力データ対応情報と、を含む実行情報を所定の形式で記述した実行形式を入力する実行形式入力手段と、アクセラレータに関する処理の手順を決定するために実行形式とは別に予め定められている規則と実行情報とに基づいて決定される処理手順を実行する処理手順実行手段と、を有し、処理手順実行手段は、アクセラレータにアクセラレータプログラムを送信するプログラム送信処理と、第２の記憶手段に出力バッファを確保する出力バッファ確保処理と、プログラム送信処理及び出力バッファ確保処理よりも後に、アクセラレータにアクセラレータプログラムを実行させるプログラム実行処理と、プログラム実行処理により出力バッファに出力された出力データを、当該出力バッファに対応する第１の記憶手段における出力先に送信する出力データ送信処理と、を実行する。

【発明の効果】

【００１９】

本発明によれば、アクセラレータを用いるアプリケーションを開発するプログラマのプログラム生産性を向上させることができる。

【図面の簡単な説明】

【００２０】

【図１】本発明の一実施形態に係るアクセラレータ処理実行装置のシステム構成を示すブロック図である。

【図２】ｆｌｏｗ－ｍｏｄｅｌを概念的に示した図である。

【図３】モデルファイルの例（ＸＭＬファイル）を示す図である。

【図４】実行形式ファイルの例（ＸＭＬファイル）を示す図である。

【図５】実行制御部の位置付けを概念的に示した図である。

【図６】スワップ処理を含まない場合におけるアクセラレータ処理実行装置の動作の例を示すフロー図である。
【図７】スワップ処理を含む場合におけるアクセラレータ処理実行装置の動作の例を示すフロー図である。
【図８】（ａ）実行形式ファイルの実行形態の例を示す図及び（ｂ）実行形式ファイルの実行パターンの例を示す図である。
【図９】バッチファイルの例を示す図である。
【図１０】モデルファイル、実行形式ファイル、及びバッチファイルの包含関係を概念的に示す図である。
【図１１】仮想バッファを用いたデータ受け渡しについて説明するための図である。
【図１２】アクセラレータ処理実行装置による計算処理及びバッチファイルの応用例を示す図である。
【図１３】図１２に示した応用例における実行形式ファイル及びモデルファイルを示す図である。
【図１４】本発明の一実施形態に係るアクセラレータ処理実行プログラムの機能構成を示すブロック図である。
【図１５】従来のＣＰＵ側プログラムの例を示す図である。
【発明を実施するための形態】
【００２１】
以下、添付図面を参照しながら本発明の実施形態を詳細に説明する。なお、図面の説明において同一又は同等の要素には同一の符号を付し、重複する説明を省略する。
【００２２】
図１は、本発明の一実施形態に係るアクセラレータ処理実行装置のシステム構成を示すブロック図である。本実施形態に係るアクセラレータ処理実行装置１は、ＣＰＵ（Central Processing Unit：中央処理装置）１０に接続されたアクセラレータ３０による所定の計算処理の実行を制御する機構を備えた装置として構成されている。具体的には、図１に示すように、アクセラレータ処理実行装置１は、ＣＰＵ１０、記憶部２０、及びアクセラレータ３０を備えている。ＣＰＵ１０は、実行制御部４０を備えている。
【００２３】
ＣＰＵ１０は、オペレーティングシステム及びアプリケーションプログラム等を実行するハードウェアである。図１に示すように、ＣＰＵ１０は、例えばＣＰＵのローカルバス（ＣＰＵバス）でＣＰＵ側メモリ２１と接続されている。また、ＣＰＵバスはバスブリッジ５０を介して周辺バス（例えば「PCI Express バス」等）に拡張され、周辺バスはアクセラレータ３０に接続されている。すなわち、ＣＰＵ１０とアクセラレータ３０とは、ＣＰＵバス、バスブリッジ５０、及び周辺バスを介して接続されている。また、補助記憶部２２とバスブリッジ５０とは、ＳＡＴＡ等のストレージ向けインターコネクトで周辺バスからさらにブリッジ（不図示）を介して接続されている。このような接続構成により、ＣＰＵ１０は、記憶部２０（ＣＰＵ側メモリ２１、補助記憶部２２）及びアクセラレータ３０の双方と互いにデータ通信可能とされている。一方、アクセラレータ３０が記憶部２０に直接アクセスする構成とすると、ブリッジによってオーバヘッドが大きくなり、アクセラレータ３０が記憶部２０（補助記憶部２２）を常に監視することによって性能低下を招いてしまう。そのため、通常、アクセラレータ３０に記憶部２０を直接認識させることはせず、記憶部２０とアクセラレータ３０との間におけるデータの受け渡しは、ＣＰＵ１０を介して行われる。
【００２４】
記憶部２０は、データを記憶する記憶手段（第１の記憶手段）であり、上述のようにＣＰＵ１０に接続されている。記憶部２０は、ＲＯＭ（Read Only Memory）及びＲＡＭ（Random Access Memory）等で構成されるＣＰＵ側メモリ２１とハードディスクメモリ等で構成される補助記憶部２２とを有している。ＣＰＵ側メモリ２１には、共有領域（共有メモリ）である仮想バッファ２１ａと、規則情報記憶部２１ｂ（詳細は後述）とを有する。仮

想バッファ２１ａは、例えば「POSIX Shared Memory」等の共有メモリライブラリにより実現される。また、本実施形態では、補助記憶部２２には、入力データファイル２２ａ、出力データファイル２２ｂ、モデルファイル２２ｃ、実行形式ファイル（実行形式）２２ｄ、及びバッチファイル２２ｅ（これらのファイルの詳細は後述）が記憶されるものとする。ここで、モデルファイル２２ｃ、実行形式ファイル２２ｄ、及びバッチファイル２２ｅは、アクセラレータを用いるアプリケーションを開発するプログラマ（以下、単に「プログラマ」と表記する）によって作成又は用意されるものである。

【００２５】

アクセラレータ３０は、上述のようにＣＰＵ１０に接続され、ＣＰＵ１０の計算処理の一部をＣＰＵ１０に代わって実行するハードウェアである。具体的には、アクセラレータ３０は、アクセラレータ３０に所定の計算処理を実行させるためのプログラム（「カーネルプログラム」とも呼ばれる。以下「アクセラレータプログラム」と表記する。）を実行するプロセッサ３１と、データを記憶するアクセラレータ側メモリ（第２の記憶手段）３２とを有している。アクセラレータ側メモリ３２には、アクセラレータプログラムの入力データ及び出力データ等が格納される。以降の説明において、「入力データ」及び「出力データ」と単に表記した場合には、これらはそれぞれ、「アクセラレータプログラムに対する入力データ」及び「アクセラデータプログラムが実行されて出力される出力データ」を指すものとする。

【００２６】

アクセラレータ３０は、例えばメニーコア方式（ストリームコンピューティングとも呼ばれる）と呼ばれる計算方式によりアクセラレータプログラムの並列実行を実現する。この方式では、アクセラレータ３０は、複数のプロセッサ３１を有し、各プロセッサ３１に異なるＩＤ（プロセッサＩＤ）を割り振り、プロセッサＩＤ毎に独立した計算処理をさせることで、計算処理の並列実行を実現する。例えば、ベクトルＡ，Ｂの加算結果をベクトルＣとして保存する処理（Ｃ＝Ａ＋Ｂ）であれば、各プロセッサ３１は、自身のプロセッサＩＤに対応するＩＤについて、Ｃ［ＩＤ］＝Ａ［ＩＤ］＋Ｂ［ＩＤ］という計算を実行する。ここでＸ［ＩＤ］（Ｘ＝Ａ，Ｂ，Ｃ）は、ベクトルＸのＩＤ番目の要素を示している。このように、メニーコア方式のアクセラレータ３０によれば、各ベクトル要素（各ＩＤに対応する要素）のベクトル計算が複数のプロセッサ３１により並列的に計算される。

【００２７】

このようなメニーコア方式のアクセラレータ３０としては、例えばＧＰＵ（Graphics Processing Unit）及びＦＰＧＡ（Field-ProgrammableGate Array）等に実現されたメニーコアフレームワーク等がある。

【００２８】

ここで、上述したように、アクセラレータ３０には記憶部２０を認識させないため、アクセラレータ３０にアクセラレータプログラムを実行させるためには、ＣＰＵ１０が、記憶部２０からアクセラレータ３０に必要なプログラム及びデータを受け渡したり、アクセラレータ３０の処理結果を記憶部２０に記憶させたりといったデータの受け渡し等の実行を制御する必要がある。具体的には、ＣＰＵ１０が以下のような処理手順を実行する必要がある。

【００２９】

即ち、ＣＰＵ１０は、例えば補助記憶部２２等に予め記憶したアクセラレータプログラムを、アクセラレータ３０上で実行可能な形式のバイナリにコンパイルした上でＣＰＵ側メモリ２１を介してアクセラレータ側メモリ３２にダウンロード（送信）し、当該バイナリをプロセッサ３１で実行可能な状態にする。また、アクセラレータプログラムが入力データを必要とする場合には、ＣＰＵ１０は、入力データを記憶する領域である入力バッファ３２ａを入力データ毎にアクセラレータ側メモリ３２上に確保する。ここで、ＣＰＵ１０は、入力バッファ３２ａに対して、当該入力バッファ３２ａがどの入力データに対応するバッファであるかを特定する情報を付与する。そして、ＣＰＵ１０は、例えば補助記憶部２２に予め記憶したＣＳＶファイル等の入力データファイル２２ａに含まれる入力デー

タを、当該入力データに対応付けて用意した入力バッファ３２ａにＣＰＵ側メモリ２１を介してコピーする。また、ＣＰＵ１０は、アクセラレータプログラムによる処理結果（出力データ）を記憶する領域である出力バッファ３２ｂを出力データ毎にアクセラレータ側メモリ３２上に確保する。ここで、ＣＰＵ１０は、出力バッファ３２ｂに対して、当該出力バッファ３２ｂがどの出力データに対応するバッファであるかを特定する情報を付与する。

【００３０】

続いて、ＣＰＵ１０は、アクセラレータ３０に対してアクセラレータプログラムの実行を指示する。そして、アクセラレータ３０によるアクセラレータプログラムの実行が完了し、出力バッファ３２ｂに出力データが出力された後に、ＣＰＵ１０は、当該出力データをＣＰＵ側メモリ２１に読み出す。その後、ＣＰＵ１０は、ＣＰＵ側メモリ２１に読み出された出力データを、例えば補助記憶部２２の出力データファイル２２ｂに書き出す。

【００３１】

従来、ＣＰＵに上述した処理手順を実行させるために、プログラマは、上述した処理手順を詳細に規定したプログラム（以下「ＣＰＵ側プログラム」と表記する）と、アクセラレータプログラムとの両方を作成している。

【００３２】

図１５に、従来プログラマが作成しているプログラムの例を示す。図１５に示すプログラムは、ＧＰＵを用いるプログラムを作成するためのプログラミングツールであるＯｐｅｎＣＬを用いて記述されたプログラムである。同図上段の破線で囲まれた記述Ｄ１がＣＰＵ側プログラムに対応する記述であり、同図下段の破線で囲まれた記述Ｄ２がＧＰＵのプログラム（アクセラレータプログラム）に対応する記述である。同図に示すように、ＣＰＵ側プログラムとして、ＧＰＵのプログラムをコンパイルする処理、ＧＰＵのメモリ上に入出力用バッファを用意する処理、入力データを用意する処理、ＧＰＵのプログラムをＧＰＵに実行させる処理、及び、ＧＰＵのプログラムの処理結果をＧＰＵのメモリから読み出す処理等がプログラマによって詳細に記述される。

【００３３】

実行制御部４０は、従来プログラマが作成していたＣＰＵ側プログラムを不要として、アクセラレータ３０に関する処理の実行を制御する実行制御手段である。具体的には、実行制御部４０は、アクセラレータプログラムと、アクセラレータプログラムの入出力データの対応付けの定義を含むデータ構造に基づいて、従来プログラマが作成していたＣＰＵ側プログラムにより規定されていた処理に対応する処理を決定及び実行する。これにより、プログラマは、アクセラレータプログラムと、当該アクセラレータプログラムの入出力データの対応付けの定義を作成するだけでよくなり、従来のＣＰＵ側プログラムを作成する必要がなくなる。

【００３４】

例えば後述するアクセラレータ処理実行プログラムＰ１がＣＰＵ１０に読み込まれることで、ＣＰＵ１０上において実行制御部４０の機能が実現される。本実施形態では、実行制御部４０は、アクセラレータ処理実行装置１上で、例えばシェルプログラムのように動作するものとして説明する。即ち、実行制御部４０は、アクセラレータ処理実行装置１が備えるキーボード等の入力装置（不図示）を介してコマンドラインでのユーザ入力等によって入力された所定のコマンドを受け付け、当該コマンドに対応して予め記憶された処理を実行する。コマンドの詳細については後述する。実行制御部４０は、実行形式入力部４１、及び処理手順実行部４２を有している。

【００３５】

実行形式入力部４１は、アクセラレータ３０に関する処理の実行制御に必要な情報（実行情報）をＸＭＬ形式（所定の形式）で記述した実行形式ファイル（実行形式）２２ｄを入力する実行形式入力手段である。実行形式入力部４１は、例えばコマンドラインでのユーザ入力等により、補助記憶部２２に記憶された実行形式ファイル２２ｄを入力する。実行形式ファイル２２ｄには、モデルファイル２２ｃを特定するモデルファイル特定情報と

、入力データに関する情報と、出力データに関する情報とが含まれている。さらに、モデルファイル２２ｃには、後述するｆｌｏｗ－ｍｏｄｅｌの情報が含まれている。具体的には、モデルファイル２２ｃには、ｆｌｏｗ－ｍｏｄｅｌの情報として、アクセラレータプログラムに関する情報と、入力バッファ３２ａに関する情報と、出力バッファ３２ｂに関する情報とが含まれている。ここで、実行情報とは、実行形式ファイル２２ｄ及びモデルファイル２２ｃに含まれる情報、並びにこれらの情報を組み合わせて得られる情報を含む全ての情報を指す。

【００３６】

図２～図４を用いて、モデルファイル２２ｃ、実行形式ファイル２２ｄ、及びこれらのファイルから得られる実行情報について詳細に説明する。

【００３７】

まず、図２及び図３を用いて、アクセラレータプログラムを実行するための標準化モデルとして本発明者らが開発したｆｌｏｗ－ｍｏｄｅｌと、このｆｌｏｗ－ｍｏｄｅｌの情報を格納したモデルファイル２２ｃについて説明する。図２は、ｆｌｏｗ－ｍｏｄｅｌを概念的に示した図である。図２に示すように、ｆｌｏｗ－ｍｏｄｅｌは、アクセラレータプログラム自体と、当該アクセラレータプログラムに対する入力データ（入力データストリーム及び定数）及び出力データ（出力データストリーム）とからなるデータ構造として定義される。

【００３８】

図３は、モデルファイル２２ｃの例を示す図である。図３に示すように、モデルファイル２２ｃは、ＸＭＬ形式のファイルとして実現される。

【００３９】

モデルファイル２２ｃには、アクセラレータプログラムに関する情報として、プログラムコード（Ｋｅｒｎｅｌ）と、アクセラレータプログラムの言語情報（ＬａｎｇＴｙｐｅ）と、アクセラレータプログラムを解釈して実行するプラットフォーム（開発環境及び実行ランタイム等を含むシステム）であるＯｐｅｎＣＬ及びＣＵＤＡ等のランタイムタイプを示す情報（ＲｕｎｔｉｍｅＴｙｐｅ）と、アクセラレータプログラムの実行における並列度を示すスレッド数及びブロック数等の情報（ＴｏｔａｌＮｕｍＴｈｒｅａｄｓ、ＴｏｔａｌＮｕｍＢｌｏｃｋｓ）とが含まれている。

【００４０】

また、入力バッファ３２ａに関する情報（Ｉｎｐｕｔ）として、入力データの型（ＤａｔａＴｙｐｅ）と、入力データの個数（Ｌｅｎｇｔｈ）と、入力バッファ３２ａとアクセラレータプログラムにおいて定義される入力引数（入力変数）との対応を示す情報とが含まれている。ここで、入力データの型（ＤａｔａＴｙｐｅ）は、整数型（ＩＮＴ）や実数型（ＦＬＯＡＴ）等を示す情報である。入力データの個数（Ｌｅｎｇｔｈ）は、入力データの大きさを示す情報であり、具体的には、入力データの型（ＤａｔａＴｙｐｅ）で示される個々のデータの個数である。入力バッファ３２ａと入力引数との対応を示す情報は、入力バッファ３２ａを特定する情報（Ｉｎｄｅｘ）と入力引数名（Ｎａｍｅ）との組を示す情報である。図３の例では、Ｉｎｄｅｘが「０」の入力バッファ３２ａが、入力引数「ａ」に対応する入力バッファ３２ａとして対応付けられており、Ｉｎｄｅｘが「１」の入力バッファ３２ａが、入力引数「ｂ」に対応する入力バッファ３２ａとして対応付けられている。また、入力引数「ａ」には、１０２４個の整数型（ＩＮＴ）の入力データ（データストリーム）が対応付けられている。

【００４１】

また、出力バッファ３２ｂに関する情報（Ｏｕｔｐｕｔ）として、出力データの個数（Ｌｅｎｇｔｈ）と、出力バッファ３２ｂとアクセラレータプログラムにおいて定義される出力引数（出力変数）との対応を示す情報と、出力データの型（ＤａｔａＴｙｐｅ）とが含まれている。ここで、出力バッファ３２ｂと出力引数との対応を示す情報は、出力バッファ３２ｂを特定する情報（Ｉｎｄｅｘ）と出力引数名（Ｎａｍｅ）との組を示す情報である。図３の例では、Ｉｎｄｅｘが「０」の出力バッファ３２ｂが、出力引数「ｃ」に対

応する出力バッファ３２ｂとして対応付けられている。

【００４２】

なお、アクセラレータプログラムが入力データを必要としない場合、即ちアクセラレータプログラムにおいて入力引数が定義されない場合には、入力バッファ３２ａに関する情報（Ｉｎｐｕｔ）は省略される。また、入力バッファ３２ａに関する情報は、アクセラレータプログラムにおいて定義された入力引数の数、即ちアクセラレータプログラムへの入力データの数だけ定義される。同様に、出力バッファ３２ｂに関する情報は、アクセラレータプログラムにおいて定義された出力引数の数、即ちアクセラレータプログラムからの出力データの数だけ定義される。

【００４３】

このように、モデルファイル２２ｃには、アクセラレータプログラムの入力引数毎に対応する入力バッファ３２ａと、出力引数毎に対応する出力バッファ３２ｂとに関する情報とが含まれる。しかし、モデルファイル２２ｃ（即ちｆｌｏｗ－ｍｏｄｅｌとして定義されるデータ構造）には、入力バッファ３２ａに用意される具体的な入力データの記憶場所（例えば入力データファイル２２ａ及び仮想バッファ２１ａ等）、及び出力バッファ３２ｂに出力された出力データのＣＰＵ１０側における具体的な出力先（例えば出力データファイル２２ｂ及び仮想バッファ２１ａ等）に関する情報が含まれていない。即ち、モデルファイル２２ｃには、アクセラレータ３０に計算処理を実行させるために必要となる具体的な入力データ及び出力データに関する情報が含まれていない。そこで、これらの入力データ及び出力データに関する情報をｆｌｏｗ－ｍｏｄｅｌに関連付けたデータ構造を記述したものが、上述の実行形式ファイル２２ｄである。

【００４４】

図４は、実行形式ファイル２２ｄの例を示す図である。図４に示すように、実行形式ファイル２２ｄは、モデルファイル２２ｃと同様に、ＸＭＬ形式のファイルとして実現される。

【００４５】

実行形式ファイル２２ｄには、モデルファイル特定情報として、アクセラレータ３０に実行させるアクセラレータプログラムを含むモデルファイル２２ｃの場所（例えばファイル名等）を示す情報（ＭｏｄｅｌＦｉｌｅ）が含まれている。

【００４６】

また、入力データに関する情報（Ｉｎｐｕｔ）として、モデルファイル２２ｃに含まれるアクセラレータプログラムの入力引数名（Ｎａｍｅ）と、入力データの記憶場所（例えば入力データファイル２２ａ及び仮想バッファ２１ａ等）を示す情報（ＤａｔａＦｉｌｅ）とが含まれている。図４の例では、ファイル名が「ｓａｍｐｌｅ１＿ａ．ｃｓｖ」の入力データファイル２２ａが、入力引数「ａ」に対応する入力データとして対応付けられ、ファイル名が「ｓａｍｐｌｅ１＿ｂ．ｃｓｖ」の入力データファイル２２ａが、入力引数「ｂ」に対応する入力データの記憶場所として対応付けられている。ここで、モデルファイル２２ｃに含まれるアクセラレータプログラムが入力データを必要としない場合、即ちアクセラレータプログラムにおいて入力引数が定義されない場合には、入力データに関する情報（Ｉｎｐｕｔ）は省略されてもよい。

【００４７】

また、出力データに関する情報（Ｏｕｔｐｕｔ）として、モデルファイル２２ｃに含まれるアクセラレータプログラムの出力引数名（Ｎａｍｅ）と、出力データの出力先（例えば出力データファイル２２ｂ及び仮想バッファ２１ａ等）を示す情報（ＤａｔａＦｉｌｅ）とが含まれている。図４の例では、ファイル名「ｓａｍｐｌｅ１＿ｃ．ｃｓｖ」で示されるファイルが、出力引数「ｃ」に対応する出力データの出力先として対応付けられている。

【００４８】

また、実行形式ファイル２２ｄには、後述するスワップ機能に関するスワップ情報（ＳｗａｐＰａｉｒ）として、スワップペアの入力引数名（Ｉｎｐｕｔ）と、スワップペアの

出力引数名（Ｏｕｔｐｕｔ）と、スワップ回数（入替回数、ＮｕｍＳｗａｐ）とが含まれている。ただし、スワップ機能の実行が不要な場合には、スワップ情報（ＳｗａｐＰａｉｒ）は省略されてもよい。

【００４９】

実行形式入力部４１は、実行形式ファイル２２ｄを入力することにより、当該実行形式ファイル２２ｄに直接記述されている情報（入力データに関する情報、及び出力データに関する情報）とともに、当該実行形式ファイル２２ｄに含まれるモデルファイル特定情報（ＭｏｄｅｌＦｉｌｅ）に示されるモデルファイル２２ｃに記述されている情報（アクセラレータプログラムに関する情報、入力バッファ３２ａに関する情報、及び出力バッファ３２ｂに関する情報）を実行情報として取得する。

【００５０】

さらに、実行形式入力部４１は、実行形式ファイル２２ｄにおける入力データの記憶場所を示す情報（ＤａｔａＦｉｌｅ）と、モデルファイル２２ｃにおける入力バッファ３２ａを特定する情報（Ｉｎｄｅｘ）とを、同一の入力引数（Ｎａｍｅ）で対応付けることにより、入力データの記憶場所と入力バッファ３２ａとの対応付けを把握する。即ち、実行形式入力部４１は、アクセラレータプログラムへの入力データ毎に、記憶部２０における記憶場所とアクセラレータ側メモリ３２に確保される入力バッファ３２ａとを対応付ける情報（入力データ対応情報）を実行情報として取得する。

【００５１】

図３及び図４の例では、実行形式入力部４１は、入力引数「ａ」での対応付けを行うことにより、ファイル名が「ｓａｍｐｌｅ１＿ａ．ｃｓｖ」の入力データファイル２２ａとＩｎｄｅｘが「０」の入力バッファ３２ａとの対応付けを把握する。同様に、実行形式入力部４１は、入力引数「ｂ」での対応付けを行うことにより、ファイル名が「ｓａｍｐｌｅ１＿ｂ．ｃｓｖ」の入力データファイル２２ａとＩｎｄｅｘが「１」の入力バッファ３２ａとの対応付けを把握する。

【００５２】

また、実行形式入力部４１は、実行形式ファイル２２ｄにおける出力データの出力先を示す情報（ＤａｔａＦｉｌｅ）と、モデルファイル２２ｃにおける出力バッファ３２ｂを特定する情報（Ｉｎｄｅｘ）とを、同一の出力引数（Ｎａｍｅ）で対応付けることにより、出力データの出力先と出力バッファ３２ｂとの対応付けを把握する。即ち、実行形式入力部４１は、アクセラレータプログラムからの出力データ毎に、アクセラレータ側メモリ３２に確保される出力バッファ３２ｂと記憶部２０における出力先とを対応付ける情報（出力データ対応情報）を実行情報として取得する。

【００５３】

図３及び図４の例では、実行形式入力部４１は、出力引数「ｃ」での対応付けを行うことにより、ファイル名が「ｓａｍｐｌｅ１＿ｃ．ｃｓｖ」の出力先ファイルとＩｎｄｅｘが「０」の出力バッファ３２ｂとの対応付けを把握する。

【００５４】

実行形式入力部４１は、上述のように取得した実行情報を処理手順実行部４２に出力する。

【００５５】

処理手順実行部４２は、アクセラレータ３０に関する処理の手順を決定するために実行形式ファイル２２ｄとは別に予め定められている規則と、実行形式入力部４１から取得した実行情報とに基づいて決定した処理手順を実行する処理手順実行手段である。具体的には、処理手順実行部４２は、例えば上述の規則を示す規則情報が予め記憶されたＣＰＵ側メモリ２１上の規則情報記憶部２１ｂを参照することで、処理手順を決定する。規則情報記憶部２１ｂは、例えば、アクセラレータ処理実行プログラムＰ１のプログラムコード自体や、当該プログラムから参照可能なファイル等により実現される。この場合、規則情報は、アクセラレータ処理実行プログラムＰ１のプログラムコードにハードコーディングされた情報や、当該プログラムから参照可能なファイル等に記述（記憶）された情報等であ

る。
【００５６】
規則情報は、どの実行情報を用いてどのような処理手順を決定するかを予め定めた規則を示す情報であり、後述するアクセラレータ処理実行プログラムＰ１を作成する管理者等によって作成されるものである。規則情報には、例えば、実行形式入力部４１により入力された実行形式ファイル２２ｄあるいはモデルファイル２２ｃに記述されたタグ要素の読取方法、及び、読み取った各タグ要素の情報に基づいてどのような処理手順をどのような順序（又はタイミング）で実行するかを示す情報等が含まれる。
【００５７】
処理手順実行部４２は、具体的には、上述した実行情報及び規則情報に基づいて、プログラム送信処理、入力バッファ確保処理、入力データ送信処理、出力バッファ確保処理、プログラム実行処理、及び出力データ送信処理を実行する。以下、各処理について詳細に説明する。
【００５８】
プログラム送信処理は、アクセラレータ３０にアクセラレータプログラムを送信する処理である。具体的には、処理手順実行部４２は、実行形式入力部４１によって読み込まれたモデルファイル２２ｃに含まれる言語情報（ＬａｎｇＴｙｐｅ）及びランタイムタイプを示す情報（ＲｕｎｔｉｍｅＴｙｐｅ）に基づいて該当するコンパイラを特定する。このようなコンパイラの特定は、例えば言語情報及びランタイムタイプの組に対して使用されるコンパイラを関連付けた情報を規則情報として規則情報記憶部２１ｂに格納しておくことで実現される。続いて、処理手順実行部４２は、特定したコンパイラで当該モデルファイル２２ｃに含まれるプログラムコード（Ｋｅｒｎｅｌ）をコンパイルしてバイナリ（バイナリカーネル）を取得する。続いて、処理手順実行部４２は、取得したバイナリをアクセラレータ３０にダウンロード（送信）し、プロセッサ３１で当該バイナリを実行可能な状態にする。
【００５９】
入力バッファ確保処理は、プログラム実行処理よりも前に、アクセラレータ側メモリ３２に入力バッファ３２ａを確保する処理である。具体的には、処理手順実行部４２は、モデルファイル２２ｃの入力バッファに関する情報（Ｉｎｐｕｔ）において指定された入力データの個数（Ｌｅｎｇｔｈ）及びデータ型（ＤａｔａＴｙｐｅ）に基づいて必要な領域の大きさを計算し、当該大きさの入力バッファ３２ａを、識別情報（Ｉｎｄｅｘ）を付与した上でアクセラレータ側メモリ３２上に確保する。入力バッファ３２ａとして必要な領域の大きさの計算は、例えば、入力データの型（ＤａｔａＴｙｐｅ）毎のデータ１つあたりに必要な領域の大きさ、及び当該大きさと入力データの個数（Ｌｅｎｇｔｈ）とを掛け合わせるという計算規則を規則情報として規則情報記憶部２１ｂに格納しておくことで実現できる。
【００６０】
また、処理手順実行部４２は、入力バッファ３２ａに対応する入力データの記憶場所がファイル（入力データファイル２２ａ）であるか否かを判定する。例えば、処理手順実行部４２は、入力バッファ３２ａに付与されたＩｎｄｅｘに対応付けられる入力データの記憶場所を入力データ対応情報に基づいて特定し、特定された入力データの記憶場所がファイル名を示すものか否かを判定する。このような判定は、例えば、ファイルであることを示す特定の文字列を、規則情報として規則情報記憶部２１ｂに格納しておき、実行形式ファイル２２ｄに含まれる入力データの記憶場所（ＩｎｐｕｔのＤａｔａＦｉｌｅ）にファイル名を示す特定の文字列（例えば「．ｃｓｖ」等）が含まれているか否かを判定することにより行うことができる。このようにして、入力バッファ３２ａに対応する入力データの記憶場所がファイル（入力データファイル２２ａ）であると判定された場合には、処理手順実行部４２は、入力バッファ３２ａとして確保したサイズと同じサイズのバッファ領域をＣＰＵ側メモリ２１上にも確保する。
【００６１】

入力データ送信処理は、入力バッファ確保処理よりも後に、記憶部２０における記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファ３２ａに送信する処理である。具体的には、処理手順実行部４２は、上述のように、入力データ対応情報に基づいて、入力バッファ３２ａに対応する入力データの記憶場所が入力データファイル２２ａであるか仮想バッファ２１ａであるかを判定する。入力データの記憶場所が入力データファイル２２ａであると判定された場合には、処理手順実行部４２は、当該入力データファイル２２ａを開き、当該入力データファイル２２ａに含まれる入力データを、入力バッファ確保処理においてＣＰＵ側メモリ２１上に確保したバッファ領域に読み出す。さらに、処理手順実行部４２は、このバッファ領域に読み出した入力データを入力バッファ３２ａにコピーする。一方、入力データの記憶場所が仮想バッファ２１ａであると判定された場合には、処理手順実行部４２は、仮想バッファ２１ａから入力データを直接読み込み、読み込んだ入力データを入力バッファ３２ａにコピーする。ここで、どの入力データ（入力データファイル２２ａ又は仮想バッファ２１ａに含まれる入力データ）をどの入力バッファ３２ａにコピーするかについては、入力データ対応情報及び入力バッファ３２ａに付与されたＩｎｄｅｘに基づいて特定される。

【００６２】

処理手順実行部４２は、入力バッファ確保処理及び入力データ送信処理を、モデルファイル２２ｃにおいて定義された全ての入力バッファ３２ａに関して実行する。図３に示した例では、処理手順実行部４２は、入力引数「ａ」に対応する入力バッファ３２ａ及び入力引数「ｂ」に対応する入力バッファ３２ａのそれぞれについて、上述の入力バッファ確保処理及び入力データ送信処理を実行する。

【００６３】

出力バッファ確保処理は、アクセラレータ側メモリ３２に出力バッファ３２ｂを確保する処理である。具体的には、処理手順実行部４２は、モデルファイル２２ｃの出力バッファに関する情報（Ｏｕｔｐｕｔ）において指定された出力データの個数（Ｌｅｎｇｔｈ）及びデータ型（ＤａｔａＴｙｐｅ）に基づいて必要な領域の大きさを計算し、当該大きさの出力バッファ３２ｂを、識別情報（Ｉｎｄｅｘ）を付与した上でアクセラレータ側メモリ３２上に確保する。出力バッファ３２ｂとして必要な領域の大きさの計算は、例えば上述した入力バッファ確保処理と同様の方法により実現できる。

【００６４】

また、処理手順実行部４２は、出力バッファ３２ｂに対応する出力データの出力先がファイル（出力データファイル２２ｂ）であるか否かを判定する。処理手順実行部４２は、出力バッファ３２ｂに付与されたＩｎｄｅｘに対応付けられる出力データの出力先を出力データ対応情報に基づいて特定し、特定された出力データの出力先がファイル名を示すものか否かを判定する。例えば、処理手順実行部４２は、出力データの出力先と同名の仮想バッファ２１ａがＣＰＵ側メモリ２１上に存在するか否かを判定し、仮想バッファ２１ａがＣＰＵ側メモリ２１上に存在しなければ、出力データの出力先がファイル名を示すものと判定することができる。また、このような判定は、例えば、ファイルであることを示す特定の文字列を、規則情報として規則情報記憶部２１ｂに格納しておき、実行形式ファイル２２ｄに含まれる出力データの出力先（ＯｕｔｐｕｔのＤａｔａＦｉｌｅ）にファイル名を示す特定の文字列（例えば「．ｃｓｖ」等）が含まれているか否かを判定することにより行うこともできる。このようにして、出力バッファ３２ｂに対応する出力データの出力先がファイル（出力データファイル２２ｂ）であると判定された場合には、処理手順実行部４２は、出力バッファ３２ｂとして確保したサイズと同じサイズのバッファ領域をＣＰＵ側メモリ２１上にも確保する。

【００６５】

図３に示した例では、出力バッファの定義は、引数「ｃ」に対応して定義された出力バッファのみであるが、モデルファイル２２ｃにおいて複数の出力バッファが定義されている場合には、処理手順実行部４２は、出力バッファ確保処理を、全ての出力バッファ３２ｂに関して実行する。

【００６６】
プログラム実行処理は、プログラム送信処理、入力データ送信処理、及び出力バッファ確保処理よりも後に、アクセラレータ３０にアクセラレータプログラムを実行させる処理である。処理手順実行部４２は、プログラム送信処理において送信したアクセラレータプログラムの実行をアクセラレータ３０に指示する。その後、アクセラレータ３０によってアクセラレータプログラムが実行され、その計算結果が、出力バッファ確保処理で出力引数毎に確保された出力バッファ３２ｂに出力される。
【００６７】
ここで、実行形式ファイル２２ｄにスワップ情報が含まれており、実行形式入力部４１によりスワップ情報が読み込まれている場合には、処理手順実行部４２は、プログラム実行処理において、アクセラレータプログラムの１回の実行が完了する毎に、スワップ情報（ＳｗａｐＰａｉｒ）の入力引数名（Ｉｎｐｕｔ）に対応する入力バッファ３２ａとスワップ情報（ＳｗａｐＰａｉｒ）の出力引数名（Ｏｕｔｐｕｔ）に対応する出力バッファ３２ｂとを入れ替え、アクセラレータプログラムを再実行する。処理手順実行部４２は、入力バッファ３２ａと出力バッファ３２ｂとの入替を、例えばポインタの入替により実際のデータ移動を伴うことなく実行する。処理手順実行部４２は、このような入出力バッファの入替及びアクセラレータプログラムの再実行を入替回数（ＮｕｍＳｗａｐ）だけ繰り返す。
【００６８】
出力データ送信処理は、プログラム実行処理により出力バッファ３２ｂに出力された出力データを、当該出力バッファ３２ｂに対応する記憶部２０における出力先に送信（コピー）する処理である。具体的には、処理手順実行部４２は、上述のように、出力データ対応情報に基づいて、出力バッファ３２ｂに対応する出力データの出力先が出力データファイル２２ｂであるか仮想バッファ２１ａであるかを判定する。出力データの出力先が出力データファイル２２ｂであると判定された場合には、処理手順実行部４２は、出力バッファ３２ｂに格納された出力データを、出力バッファ確保処理においてＣＰＵ側メモリ２１上に確保したバッファ領域にコピーする。さらに、処理手順実行部４２は、このバッファ領域に読み出したデータを出力データファイル２２ｂに書き込む。一方、出力データの出力先が仮想バッファ２１ａであると判定された場合には、処理手順実行部４２は、出力バッファ３２ｂに格納された出力データを仮想バッファ２１ａに書き込む。
【００６９】
図５を用いて、以上説明した実行制御部４０による処理を実現する構成について補足する。図５は、実行制御部４０の位置付けを概念的に示した図である。図５に示すように、実行制御部４０は、実行形式ファイル２２ｄを入力し、実行形式ファイル２２ｄのモデルファイル特定情報により関連付けられたモデルファイル２２ｃに定義されたアクセラレータプログラムに入力データを与え、アクセラレータ３０に計算処理を実行させ、出力データを得る処理を実現している。実行制御部４０は、このような処理を実現するために、ＧＰＵ及びＦＰＧＡ等に実現されたメニーコアフレームワーク等のアクセラレータ３０を制御するソフトウェア（アクセラレータ用ドライバ）、並びにアクセラレータに応じたコンパイラ及びライブラリ等を備えた各種の実行ランタイムシステムを利用している。ここで、実行ランタイムシステムは、例えば上述したＯｐｅｎＣＬ及びＣＵＤＡ等のプラットフォームや、本発明者らが開発したＣａｒａｖｅｌａプラットフォーム等で構成される。
【００７０】
ここで、Ｃａｒａｖｅｌａプラットフォームは、下位層に位置付けられるＯｐｅｎＣＬ及びＣＵＤＡ等のプラットフォームを利用して、ｆｌｏｗ－ｍｏｄｅｌをアクセラレータ３０にマップする（ｆｌｏｗ－ｍｏｄｅｌで定義されたアクセラレータプログラムをアクセラレータ３０上で実行可能にする）処理等を実行するライブラリ（Ｃａｒａｖｅｌａライブラリ）を備えたプラットフォームである。
【００７１】
次に、図６及び図７を用いて、処理手順実行部４２により実行される処理の動作につい

て説明する。なお、各処理の詳細については、既に説明しているため、適宜省略する。

【００７２】

まず、図６を用いて、スワップ処理を含まない場合におけるアクセラレータ処理実行装置の動作について説明する。

【００７３】

まず、実行形式ファイル２２ｄが、実行形式入力部４１に入力される（ステップＳ１０１）。続いて、実行形式入力部４１は、実行形式ファイル２２ｄのモデルファイル特定情報により関連付けられるモデルファイル２２ｃを特定し、当該モデルファイル２２ｃを入力する（ステップＳ１０２）。続いて、実行形式入力部４１は、実行形式ファイル２２ｄ及びモデルファイル２２ｃに基づいて、上述した実行情報を取得する（ステップＳ１０３）。

【００７４】

続いて、処理手順実行部４２は、実行形式入力部４１によって取得された実行情報と規則情報記憶部２１ｂに記憶された規則情報とを入力する（ステップＳ１０４）。そして、処理手順実行部４２は、上述したように、実行情報及び規則情報に基づいて処理手順（各処理の内容、処理順序、及び処理実行のタイミング等）を決定し、決定した処理手順（ステップＳ１０５～Ｓ１１１）を実行する。

【００７５】

処理手順実行部４２は、プログラム送信処理において、モデルファイル２２ｃに含まれるアクセラレータプログラムをコンパイルした上でアクセラレータ３０にダウンロードし、実行可能な状態にする（ステップＳ１０５）。

【００７６】

続いて、処理手順実行部４２は、入力バッファ確保処理において、アクセラレータ側メモリ３２上に入力バッファ３２ａを確保する（ステップＳ１０６）。続いて、処理手順実行部４２は、入力データ送信処理において、記憶部２０における記憶場所に記憶されている入力データを、当該記憶場所に対応する入力バッファ３２ａに送信する（ステップＳ１０７）。ここで、ステップＳ１０６，Ｓ１０７の処理は、処理手順実行部４２により、モデルファイル２２ｃに定義されている全ての入力バッファ３２ａに対して実行される。

【００７７】

続いて、処理手順実行部４２は、出力バッファ確保処理において、アクセラレータ側メモリ３２上に出力バッファ３２ｂを確保する（ステップＳ１０８）。続いて、処理手順実行部４２は、プログラム実行処理において、アクセラレータ３０にアクセラレータプログラムを実行させる（ステップＳ１０９）。続いて、処理手順実行部４２は、出力データ送信処理において、アクセラレータプログラムの実行により出力バッファ３２ｂに出力された出力データを、当該出力バッファ３２ｂに対応する記憶部２０における出力先に送信する（ステップＳ１１０）。

【００７８】

その後、処理手順実行部４２は、後処理として、全ての入力バッファ３２ａ及び全ての出力バッファ３２ｂを解放し、アクセラレータプログラムをアクセラレータ３０上から無効化する処理を実行する（ステップＳ１１１）。

【００７９】

次に、図７を用いて、スワップ処理を含む場合におけるアクセラレータ処理実行装置の動作について説明する。ここでは、図６を用いて説明したスワップ処理を含まない場合の動作と異なる部分を主に説明する。

【００８０】

まず、スワップ情報（ＳｗａｐＰａｉｒ）が含まれた実行形式ファイル２２ｄが、実行形式入力部４１に入力される（ステップＳ２０１）。ここで、実行形式入力部４１により、スワップ情報に含まれる入力引数名（Ｉｎｐｕｔ）に対応する入力バッファ３２ａと出力引数名（Ｏｕｔｐｕｔ）に対応する出力バッファ３２ｂとの全てのスワップペア（組み合わせ）の情報が読み込まれる。続いて、実行形式入力部４１は、実行形式ファイル２２

ｄのモデルファイル特定情報により関連付けられるモデルファイル２２ｃを特定し、当該モデルファイル２２ｃを入力する（ステップＳ２０２）。続いて、実行形式入力部４１は、実行形式ファイル２２ｄ及びモデルファイル２２ｃに基づいて、上述した実行情報を取得する（ステップＳ２０３）。

【００８１】

続いて、処理手順実行部４２は、実行形式入力部４１によって取得された実行情報と規則情報記憶部２１ｂに記憶された規則情報とを入力する（ステップＳ２０４）。そして、処理手順実行部４２は、上述したように、実行情報及び規則情報に基づいて処理手順（各処理の内容、処理順序、及び処理実行のタイミング等）を決定し、決定した処理手順（ステップＳ２０５～Ｓ２１１）を実行する。

【００８２】

処理手順実行部４２は、ステップＳ１０５～Ｓ１０８と同様に、プログラム送信処理、入力バッファ確保処理、入力データ送信処理、及び出力バッファ確保処理を実行する（ステップＳ２０５～Ｓ２０８）。

【００８３】

続いて、処理手順実行部４２は、プログラム実行処理（ステップＳ２０９）として、ステップＳ２０９ａ～Ｓ２０９ｅの処理を実行する。まず、処理手順実行部４２は、スワップ回数（ＮｕｍＳｔｅｐ）をカウントするカウンタＳをスワップペア毎に初期値０で用意する（ステップＳ２０９ａ）。続いて、処理手順実行部４２は、アクセラレータ３０にアクセラレータプログラムを実行させる（ステップＳ２０９ｂ）。アクセラレータプログラムの実行が完了したら、処理手順実行部４２は、各スワップペアのカウンタＳをインクリメントし（ステップＳ２０９ｃ）、カウンタＳの値が実行形式ファイル２２ｄに指定されたスワップ回数（ＮｕｍＳｔｅｐ）に達しているか否かをスワップペア毎に判定する（ステップＳ２０９ｄ）。

【００８４】

ステップＳ２０９ｄにおいてカウンタＳの値がスワップ回数に達していないと判定されたスワップペアに関しては、処理手順実行部４２は、入力バッファ３２ａと出力バッファ３２ｂとを交換する（ステップＳ２０９ｅ）。一方、ステップＳ２０９ｄにおいてカウンタＳの値がスワップ回数に達していると判定されたスワップペアに関してはそれ以降のスワップは行わない。ステップＳ２０９ｄにおいてカウントＳの値がスワップ回数に達していないと判定されるスワップペアが一つでもある場合には、処理手順実行部４２は、上述のスワップ処理（ステップＳ２０９ｅ）を実行した上で、ステップＳ２０９ｂに戻ってアクセラレータ３０にアクセラレータプログラムを再実行させる。一方、ステップＳ２０９ｄにおいて全てのスワップペアのカウンタＳの値がスワップ回数に達していると判定された場合（ステップＳ２０９ｄ：ＹＥＳ）には、処理手順実行部４２は、出力データ送信処理において、アクセラレータプログラムの実行により出力バッファ３２ｂに出力された出力データを、当該出力バッファ３２ｂに対応する記憶部２０における出力先に送信する（ステップＳ２１０）。

【００８５】

その後、処理手順実行部４２は、後処理として、全ての入力バッファ３２ａ及び全ての出力バッファ３２ｂを解放し、アクセラレータプログラムをアクセラレータ３０上から無効化する処理を実行する（ステップＳ２１１）。

【００８６】

次に、図８を用いて実行制御部４０による実行形式ファイル２２ｄの実行形態について説明する。図８（ａ）は、実行制御部４０による実行形式ファイル２２ｄの実行形態の例を示す図である。実行制御部４０の実行形態の仕様は何でもよいが、例えば、実行制御部４０は、上述したように、シェルプログラムのように動作し、ユーザによるコマンドライン入力を受け付けるように実装される。この場合、ユーザは、例えばコマンドラインで実行形式ファイル２２ｄのファイル名（「ｆｌｏｗ－ｍｏｄｅｌ」）を直接指定することで、当該実行形式ファイル２２ｄを実行制御部４０に実行させることができる（図８（ａ）

の（１）参照）。また、ユーザは、例えば実行制御部４０の機能を実行することを示すコマンド（ｃａｒｓｈ）に実行形式ファイル２２ｄのファイル名を引数として与えることで、当該実行形式ファイル２２ｄを実行制御部４０に実行させることができるようにしてもよい（図８（ａ）の（２）参照）。また、実行制御部４０は、ユーザによる直接のコマンド入力だけでなく、コマンド列を羅列したバッチファイル２２ｅ（「ｂａｔｃｈｆｉｌｅ」）の入力を受け付け、バッチファイル２２ｅに記載されたコマンド列を順に実行する仕様としてもよい（図８（ａ）の（３）参照）。

【００８７】

このような実行形態の仕様、即ち実行制御部４０が対応可能な入力方法及びコマンド文字列に関連付けられる処理内容等は、例えば規則情報記憶部２１ｂに予め記憶させておけばよい。また、本実施形態では、実行形式入力部４１が、実行形式ファイル２２ｄ以外のコマンド入力及びバッチ形式での入力を受け付ける機能を備えており、コマンド文字列に対応する処理を実行する機能は、例えば処理手順実行部４２が備えているものとする。

【００８８】

処理手順実行部４２は、例えば、実行形式ファイル２２ｄ又はバッチファイル２２ｅの実行に関しては、プロセス及びスレッド等の独立した実行単位で実行する。このプロセスは、例えばＵＮＩＸ（登録商標）系ＯＳ等のシステムコールの一つである「ｆｏｒｋ」等により実現される。このような実行単位は、後述する実行管理コマンドで制御可能とされている。

【００８９】

処理手順実行部４２が実行可能なコマンドとしては、例えば、実行終了コマンド（ｅｘｉｔ）、ヘルプコマンド（ｈｅｌｐ）、実行管理コマンド（ｐｓ，ｋｉｌｌ）、仮想バッファ管理コマンド（ｖｉｒｔｂｕｆ）、実行同期コマンド（ｓｙｎｃ）、タイマーコマンド（ｔｉｍｅｒ）、繰り返し実行コマンド（ｒｅｐｅａｔ）等が実装されている。

【００９０】

実行終了コマンドは、コマンドラインでのコマンド実行を終了する（例えばコンソール画面を閉じる）機能を実現するコマンドである。ヘルプコマンドは、操作方法を画面上に表示させる機能を実現するコマンドである。実行管理コマンドは、現在実行されている実行単位の状態を画面上に表示したり、現在実行されている実行単位を停止・排除したりする機能を実現するコマンドである。仮想バッファ管理コマンドは、仮想バッファ２１ａの作成、削除、情報表示、データファイル（ＣＳＶファイル）から仮想バッファ２１ａへのデータ読み込み、仮想バッファ２１ａからデータファイル（ＣＳＶファイル）へのデータ書込み、仮想バッファ２１ａの名前変更、２つの仮想バッファ２１ａ間の名前の入替等の操作を実現するコマンドである。実行同期コマンドは、既に実行されている実行単位の実行が全て完了するのを待つ機能を実現するコマンドである。タイマーコマンドは、開始時刻を保存したり、保存された開始時刻からの経過時間を計算し出力したりする機能を実現するコマンドである。繰り返し実行コマンドは、引数として、繰り返し回数及び実行対象ファイル（実行形式ファイル２２ｄ又はバッチファイル２２ｅ）を与えることで、引数として指定された実行対象ファイルを繰り返し回数だけ繰り返す機能を実現するコマンドである。

【００９１】

また、処理手順実行部４２は、実行対象ファイルの実行の種類として、フォアグラウンド実行及びバックグラウンド実行の２種類が実装されている。フォアグラウンド実行とは、実行対象ファイルの実行が完了するまで次のコマンドを実行しない実行方式をいう。バックグラウンド実行とは、実行対象ファイルの実行の完了を待たずに次のコマンドを実行する実行方式をいう。例えば図８（ａ）に示すように、「ｆｌｏｗ－ｍｏｄｅｌ２　＆」と、実行形式ファイル２２ｄのファイル名の後に「＆」がある場合に、実行制御部４０は、当該実行形式ファイル２２ｄをバックグラウンド実行する。

【００９２】

実行制御部４０は、このような２種類の実行方式（フォアグラウンド実行、バックグラ

ウンド実行）が実装されていることにより、複数の実行形式ファイル２２ｄの様々な実行パターン（実行順序）を容易に実現することができる。具体的には、例えば図８（ａ）の下欄に示すような３行のコマンド入力により、図８（ｂ）に示すような複数の実行形式ファイル２２ｄ（「ｆｌｏｗ－ｍｏｄｅｌ１」、「ｆｌｏｗ－ｍｏｄｅｌ２」、「ｆｌｏｗ－ｍｏｄｅｌ３」）の実行パターンが実現される。この場合、「ｆｌｏｗ－ｍｏｄｅｌ１」の出力データの出力先と「ｆｌｏｗ－ｍｏｄｅｌ２」及び「ｆｌｏｗ－ｍｏｄｅｌ３」の入力データの記憶場所とが互いに対応するように、それぞれの実行形式ファイル２２ｄを定義する必要がある。ただし、このように実行形式ファイル２２ｄを定義し、上述のようにコマンド列をバッチ形式に並べて実行させるだけで、図８（ｂ）のような実行パターンを実現することができるという点で、プログラマによるアクセラレータプログラムの実行順序の実装は、アクセラレータプログラムの実行順序だけを考慮したプログラミングにより実現可能であり、従来ＣＰＵ側プログラムを作成した場合と比較して容易となっている。

【００９３】

図９に、実行制御部４０が読み取り実行可能なバッチファイルの例を示す。図９に示すように、バッチファイルは、例えばＸＭＬ形式で記述され、上述したコマンドと、実行形式ファイル２２ｄ又はバッチファイルとが羅列されている。図９に示すコマンド処理の一部について、その処理内容を説明する。

【００９４】

「ｖｉｒｔｂｕｆ　ｃｒｅａｔｅ　ｉｎｔ　１０２４　ｂｕｆ＿ａ」は、ｉｎｔ型データを１０２４個格納可能な「ｂｕｆ＿ａ」という名前の仮想バッファ２１ａを作成（ｃｒｅａｔｅ）する処理を実行するコマンドである。

【００９５】

「ｖｉｒｔｂｕｆ　ｆｉｌｌ　ｓａｍｐｌｅ１＿ａ．ｃｓｖ　ｂｕｆ＿ａ」は、「ｂｕｆ＿ａ」という名前で作成された仮想バッファ２１ａに「ｓａｍｐｌｅ１＿ａ．ｃｓｖ」というファイル名のデータファイルに含まれる入力データを読み込む（ｆｉｌｌ）処理を実行するコマンドである。

【００９６】

「ｓａｍｐｌｅ＿ｖｉｒｔ　＆ａｍｐ；」は、「ｓａｍｐｌｅ＿ｖｉｒｔ」という名前の実行対象ファイル（実行形式ファイル２２ｄ又はバッチファイル）をバックグラウンド実行するコマンドである。なお、ＸＭＬでは、「＆」の文字がそのままの意味（＆）を示すものとして使用できないため、ＸＭＬにおいて「＆」を意味する表記として定められている「＆ａｍｐ；」が記述されている。

【００９７】

「ｓｙｎｃ」は、既に実行されている実行単位の実行が全て完了するのを待つ機能を実現するコマンドである。図９の例では、このコマンドにより、実行中の「ｓａｍｐｌｅ＿ｖｉｒｔ」の実行が完了するのを待つことができる。即ち、「ｓａｍｐｌｅ＿ｖｉｒｔ」の実行が完了してから次のコマンド処理を実行させることができる。

【００９８】

「ｒｅｐｅａｔ　１０　ｓａｍｐｌｅ＿ｖｉｒｔ．ｘｍｌ」は、実行対象ファイル（「ｓａｍｐｌｅ＿ｖｉｒｔ．ｘｍｌ」）を１０回繰り返して実行するコマンドである。

【００９９】

「ｖｉｒｔｂｕｆ　ｄｕｍｐ　ｃ＿ｔｍｐ．ｃｓｖ　ｂｕｆ＿ｃ」は、「ｂｕｆ＿ｃ」という名前の仮想バッファ２１ａに格納された出力データを、「ｃ＿ｔｍｐ．ｃｓｖ」という名前のデータファイルに書き込む（ダンプする）コマンドである。

【０１００】

「ｖｉｒｔｂｕｆ　ｓｗａｐ　ｂｕｆ＿ａ　ｂｕｆ＿ｃ」は、「ｂｕｆ＿ａ」という名前の仮想バッファ２１ａと「ｂｕｆ＿ｃ」という名前の仮想バッファ２１ａとを交換する（例えばポインタを入れ替える）コマンドである。

【０１０１】

「ｖｉｒｔｂｕｆ　ｄｅｌｅｔｅ　ｂｕｆ＿ａ」は、「ｂｕｆ＿ａ」という名前の仮想バッファ２１ａを削除する処理である。

【０１０２】

図１０は、モデルファイル２２ｃ、実行形式ファイル２２ｄ、及びバッチファイル２２ｅの包含関係を示す概念図である。図１０に示すように、モデルファイル２２ｃは実行形式ファイル２２ｄに包含される。また、実行形式ファイル２２ｄはバッチファイル２２ｅに包含され得る。さらに、バッチファイル２２ｅの中に別のバッチファイル２２ｅを包含する入れ子構造とすることも可能である。

【０１０３】

次に、図１１を用いて、仮想バッファを用いたデータ受け渡しについて説明する。図１１は、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ１」の実行形式ファイル２２ｄの実行により得られた出力データを、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ２」の実行形式ファイル２２ｄに対して入力データとして与えることで、２つの実行形式ファイル２２ｄを連続して実行する場合を示している。

【０１０４】

同図左側は、２つの実行形式ファイル２２ｄ間でのデータの受け渡しにＣＳＶファイルを介する場合を示している。このようなデータの受け渡しは、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ１」の実行形式ファイル２２ｄの出力データの出力先（ＯｕｔｐｕｔのＤａｔａＦｉｌｅ）と、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ２」の実行形式ファイル２２ｄの入力データの記憶場所（ＩｎｐｕｔのＤａｔａＦｉｌｅ）とに、同一のＣＳＶファイルを指定することで実現できる。

【０１０５】

一方、同図右側は、２つの実行形式ファイル２２ｄ間でのデータの受け渡しに仮想バッファ２１ａを介する場合を示している。このようなデータの受け渡しは、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ１」の実行形式ファイル２２ｄの出力データの出力先（ＯｕｔｐｕｔのＤａｔａＦｉｌｅ）と、ファイル名が「ｆｌｏｗ－ｍｏｄｅｌ２」の実行形式ファイル２２ｄの入力データの記憶場所（ＩｎｐｕｔのＤａｔａＦｉｌｅ）とに、同一の仮想バッファ２１ａを指定することで実現できる。

【０１０６】

同図右側のように仮想バッファ２１ａを介してデータの受け渡しを行うことで、ファイルからのデータ読み込みや、ファイルへのデータ書込み等のファイルＩ／Ｏを排除することができる。

【０１０７】

次に、図１２及び図１３を用いて、アクセラレータ処理実行装置１による計算処理の応用例について説明する。図１２は、アクセラレータ処理実行装置１による計算処理及びバッチファイル２２ｅの応用例を示す図である。図１３は、図１２に示した応用例における実行形式ファイル２２ｄ及びモデルファイル２２ｃを示す図である。

【０１０８】

図１２左側に示される計算処理は、画像処理における高速フーリエ変換（ＦＦＴ）のアルゴリズム（いわゆるバタフライ演算）である。この計算処理では、実部（Ｒｅａｌ　ｐａｒｔ）と虚部（Ｉｍａｇｉｎａｒｙ　ｐａｒｔ）との役割を交換しながら、図中において「Ｒｅｏｒｄｅｒ」、「１Ｄ　ＦＦＴ」、及び「Ｔｒａｎｓｐｏｓｅ」で示される一連の計算処理を２回繰り返した後に「Ｆｉｌｔｅｒ」で示される処理を実行するものである。ここで、各計算処理の詳細（画像処理アルゴリズムの詳細）についての説明は、本発明の内容から外れるため省略する。

【０１０９】

「Ｒｅｏｒｄｅｒ」、「１Ｄ　ＦＦＴ」、「Ｔｒａｎｓｐｏｓｅ」、及び「Ｆｉｌｔｅｒ」で示される処理は、それぞれ異なるアクセラレータプログラムとして実装され、アクセラレータ３０により実行される処理である。図１２右側に示されるバッチファイル２２ｅ中における「ｒｅｏｒｄｅｒ．ｘｍｌ」、「ｆｆｔ．ｘｍｌ」、「ｔｒａｎｓ．ｘｍｌ

」、及び「ｌｏｗｐａｓｓ．ｘｍｌ」は、上述の各アクセラレータプログラムを含むモデルファイル２２ｃに関連付けられた実行形式ファイル２２ｄを示している。

【０１１０】

図１３の左から１列目は、「ｆｆｔ．ｘｍｌ」で示される実行形式ファイル２２ｄの中身を示している。図１３の左から２列目～４列目は、当該実行形式ファイル２２ｄのモデルファイル特定情報（ＭｏｄｅｌＦｉｌｅ）により特定された「ｆｆｔ＿ｆｌｏｗｍｏｄｅｌ．ｘｍｌ」で示されるモデルファイル２２ｃの中身を示している。

【０１１１】

図１３の例では、アクセラレータプログラム（Ｋｅｒｎｅｌ）における入力引数「ｉｎ＿ｒｅａｌ」，「ｉｎ＿ｉｍａｇ」に対してそれぞれ、Ｉｎｄｅｘが「０」，「１」の入力バッファ３２ａが対応付けられ、「ｂｕｆ＿ｉｎ＿ｒｅａｌ」，「ｂｕｆ＿ｉｎ＿ｉｍａｇ」という名前の仮想バッファ２１ａが対応付けられるように定義されている。同様に、出力引数「ｏｕｔ＿ｒｅａｌ」，「ｏｕｔ＿ｉｍａｇ」に対してそれぞれ、Ｉｎｄｅｘが「０」，「１」の出力バッファ３２ｂが対応付けられ、「ｂｕｆ＿ｏｕｔ＿ｒｅａｌ」，「ｂｕｆ＿ｏｕｔ＿ｉｍａｇ」という名前の仮想バッファ２１ａが対応付けられるように定義されている。

【０１１２】

図１２右側に示されるバッチファイル２２ｅが実行形式入力部４１によって入力され、処理手順実行部４２により実行される処理の概要について説明する。まず、入力データ用の仮想バッファ２１ａ（ｂｕｆ＿ｉｎ＿ｒｅａｌ、ｂｕｆ＿ｉｎ＿ｉｍａｇ）と出力データ用の仮想バッファ２１ａ（ｂｕｆ＿ｏｕｔ＿ｒｅａｌ、ｂｕｆ＿ｏｕｔ＿ｉｍａｇ）をＣＰＵ側メモリ２１上に作成する処理が実行される（図１２の（Ａ）参照）。

【０１１３】

続いて、入力データ用の仮想バッファ２１ａ（ｂｕｆ＿ｉｎ＿ｒｅａｌ、ｂｕｆ＿ｉｎ＿ｉｍａｇ）にそれぞれ、「ｌｅｎｎａ１２８＿ｒｅａｌ．ｃｓｖ」及び「ｌｅｎｎａ１２８＿ｉｍａｇ．ｃｓｖ」という名前の入力データファイル２２ａに含まれる入力データを読み込む処理が実行される（図１２の（Ｂ）参照）。

【０１１４】

続いて、上述の各実行形式ファイル２２ｄを順に実行する処理が実行される（図１２の（Ｃ）参照）。ここで、各実行形式ファイル２２ｄの実行毎に、入力データ用の仮想バッファ２１ａと出力データ用の仮想バッファ２１ａとを交換する処理（ｖｉｒｔｂｕｆ　ｓｗａｐ）が実行される。これにより、例えば「Ｒｅｏｒｄｅｒ」の処理（「ｒｅｏｒｄｅｒ．ｘｍｌ」で示される実行形式ファイル２２ｄを実行する処理）によって得られた出力データを、一旦ファイルに書き出すことなく、次の「１Ｄ　ＦＦＴ」の処理（「ｆｆｔ．ｘｍｌ」で示される実行形式ファイル２２ｄを実行する処理）に対する入力データとして受け渡す処理が実現される。

【０１１５】

続いて、「Ｆｉｌｔｅｒ」の処理（「ｌｏｗｐａｓｓ．ｘｍｌ」で示される実行形式ファイル２２ｄを実行する処理）まで完了した後に、出力データ用の仮想バッファ２１に出力された出力データを、それぞれ対応する出力データファイル２２ｂに書き出す処理が実行される（図１２の（Ｄ）参照）。

【０１１６】

その後、後処理として、全ての仮想バッファ２１ａを削除する処理が実行される（図１２の（Ｅ）参照）。

【０１１７】

次に、図１４を用いて、ＣＰＵと、記憶部（ＣＰＵ側メモリ、補助記憶部）と、アクセラレータとを備えるコンピュータをアクセラレータ処理実行装置１として機能させるためのアクセラレータ処理実行プログラムＰ１について説明する。

【０１１８】

図１４は、アクセラレータ処理実行プログラムＰ１のモジュールを示すブロック図であ

る。図１４に示すように、アクセラレータ処理実行プログラムＰ１は、実行形式入力モジュールＰ４１と、処理手順実行モジュールＰ４２と、を備える。上記の実行形式入力モジュールＰ４１及び処理手順実行モジュールＰ４２が実行されることにより実現される機能は、上述したアクセラレータ処理実行装置１において対応する実行形式入力部４１及び処理手順実行部４２の機能と同様である。

【０１１９】

このように構成されたアクセラレータ処理実行プログラムＰ１は、例えばＣＤ－ＲＯＭ及びＤＶＤ等の記録媒体に記憶され、アクセラレータ処理実行装置１として用いられるコンピュータのＣＰＵにより実行される。具体的には、当該コンピュータは、例えばＣＤ－ＲＯＭドライブ及びＤＶＤドライブ等の記録媒体読取部を備え、当該記録媒体読取部に記録媒体が挿入されると、記録媒体読取部から記録媒体に格納されたアクセラレータ処理実行プログラムＰ１にアクセス可能となる。そして、当該アクセラレータ処理実行プログラムＰ１を当該コンピュータに実行させることによって、当該コンピュータを、本実施形態に係るアクセラレータ処理実行装置１として動作させることが可能となる。

【０１２０】

なお、アクセラレータ処理実行プログラムＰ１は、搬送波に重畳されたコンピュータデータ信号としてネットワークを介して提供されるものであってもよい。この場合、アクセラレータ処理実行装置１として用いられるコンピュータは、モデム等のデータ通信を行う通信制御部によって受信したアクセラレータ処理実行プログラムＰ１をＣＰＵ側メモリ２１に格納することにより、当該アクセラレータ処理実行プログラムＰ１を実行することができる。

【０１２１】

以上説明したように、本実施形態に係るアクセラレータ処理実行装置１では、実行形式入力部４１が実行形式ファイル２２ｄを入力して実行情報を取得し、処理手順実行部４２が当該実行情報及び予め定めた規則（規則情報記憶部２１ｂに記憶された規則情報）に基づいてアクセラレータに関する処理手順を決定し、実行する。このように、ＣＰＵ１０側とアクセラレータ３０側との間の入力データ及び出力データの受け渡しを含む各処理手順が、実行制御部４０によって適切に決定及び実行されるため、プログラマは、ＣＰＵ１０側とアクセラレータ３０側との間のデータの受け渡し等の処理手順を詳細に規定したＣＰＵ側プログラムを作成する手間から解放される。即ち、上記アクセラレータ処理実行装置１によれば、アクセラレータプログラムをアクセラレータ３０に実行させるために、プログラマは、実行形式ファイル２２ｄ及びアクセラレータプログラムを含むモデルファイル２２ｃを用意するだけでよいため、プログラマのプログラム生産性を向上させることができる。

【０１２２】

特に、複数のアクセラレータプログラムの実行パターン（図８（ｂ）に示したような直列実行又は並列（同時）実行）を変更したり、この変更に伴い入出力データを変更したりする場合には、従来のＣＰＵ側プログラムでは、多くの修正が必要となる。これに対し、本実施形態に係るアクセラレータ処理実行装置１によれば、バッチファイル２２ｅにおいて実行形式ファイル２２ｄの実行順序、及び実行形式ファイル２２ｄの入出力データに関する定義（Ｉｎｐｕｔ及びＯｕｔｐｕｔのＤａｔａＦｉｌｅ）を変更するだけでよく、プログラマの負担が軽減される。

【０１２３】

以上から、プログラマは、モデルファイル２２ｃにアクセラレータプログラムを実装し、実行形式ファイル２２ｄ及びその他の実行可能なコマンドを羅列したバッチファイル２２ｅを作成することで、複数のアクセラレータプログラムの直列実行及び並列実行を含めて、アクセラレータプログラムの実行順序の設計及び実装を容易に行うことができる。

【０１２４】

また、プログラマがＣＰＵ側プログラムを作成する必要がなくなるということは、裏を返せば、プログラマに対して、ＣＰＵ１０の機能の実行及び制御の自由度が高く、ＣＰＵ

側メモリ２１及びアクセラレータ側メモリ３２等を直接操作すること等が可能なＣＰＵ側プログラムの実行を許可しないことを意味する。これにより、データ破壊及び機密情報の書き換え等を防止し、ハードウェアの安全性を高めることができる。

【０１２５】

また、ＣＰＵ側メモリ２１等の記憶領域を自由にアクセスしたり、ＣＰＵ１０のハードウェアをアクセスしたり、ＣＰＵ１０のプログラムを書き換えたりすることは、アクセラレータプログラムからはできない。従って、いわゆるトロイの木馬型のコンピュータウィルス等を本発明で扱う実行方式（モデルファイル２２ｃ、実行形式ファイル２２ｄ、バッチファイル２２ｅ等を定義する方式）では作成することができない。

【０１２６】

また、上記アクセラレータ処理実行装置１によれば、アクセラレータプログラムの入力データ又は出力データの受け渡しを、ファイル（例えばＣＳＶファイル等）を介することなく、仮想バッファ２１ａを介して実行することができる。従って、データの受け渡しにおいてファイルＩ／Ｏを排除して、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。

【０１２７】

また、上記アクセラレータ処理実行装置１によれば、アクセラレータプログラムの１回の実行が完了する毎に、例えばポインタの入替等により入力バッファ３２ａと出力バッファ３２ｂとを入れ替えてアクセラレータプログラムを再実行することができるため、一連のアクセラレータプログラムによる処理が完了するまでの時間を短縮することができる。具体的には、出力バッファ３２ｂに出力されたアクセラレータプログラムの出力データを一旦ＣＰＵ１０側の記憶部２０に移動させてから再度アクセラレータ３０側に入力データとして入力するといった無駄なデータ受け渡し処理を省略することができる。

【０１２８】

以上、本発明に係る実施形態について詳細に説明した。しかし、本発明は、上記実施形態に限定されるものではない。本発明は、その要旨を逸脱しない範囲において様々な変形が可能である。

【０１２９】

本実施形態では、処理手順実行部４２が処理手順を決定するための実行情報が、実行形式ファイル２２ｄ及びモデルファイル２２ｃに含まれるものとして説明したが、実行情報は、このような形態に限定されない。例えば、実行形式ファイル２２ｄとモデルファイル２２ｃとは同一のファイルとして構成されていてもよいし、３つ以上のファイルに情報が分散されていてもよい。また、これらのファイルのファイル形式はＸＭＬ形式以外の形式であってもよい。また、本実施形態では、モデルファイル２２ｃとして本発明者らが開発したｆｌｏｗ－ｍｏｄｅｌを用いる例を説明したが、必ずしもｆｌｏｗ－ｍｏｄｅｌを用いなくてもよい。即ち、実行形式入力部４１に入力させるものは、本実施形態で説明した実行情報に相当する情報を抽出可能なものであれば何でもよい。

【０１３０】

また、入力データファイル２２ａ、出力データファイル２２ｂ、モデルファイル２２ｃ、及び実行形式ファイル２２ｄは、必ずしも補助記憶部２２に記憶されなくともよく、例えば、ネットワークを介してアクセラレータ処理実行装置と互いに通信可能なリモートコンピュータ等が備える記憶装置に記憶されてもよい。

【０１３１】

また、モデルファイル２２ｃ、実行形式ファイル２２ｄ、及びバッチファイル２２ｅは、例えばＲＳＡ等により暗号化してもよい。この場合における実行形式ファイル２２ｄの実行の一例について以下に示す。例えば、暗号化のパブリックキー（復号のためのキー）は、実行制御部４０（実行形式入力部４１）がアクセスできる場所（例えばディレクトリ等）に保存され、暗号化のプライベートキー（暗号化に用いられたキー）は、プログラマによって与えられる構成とする。この場合、実行制御部４０は、コマンドライン等により実行形式ファイル２２ｄを入力したら、所定の方法でプログラマに対してプライベートキ

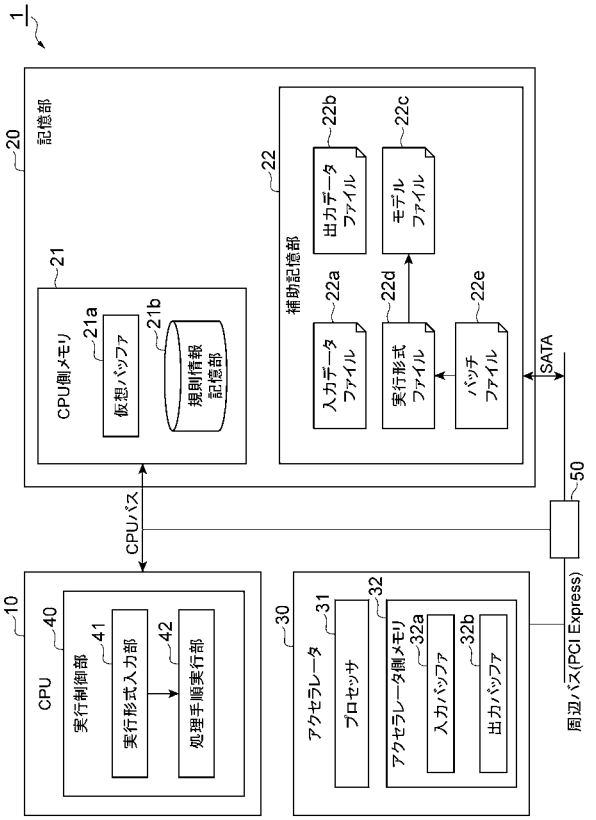
ーを要求する。そして、プログラマから所定の方法でプライベートキーを受け取り、当該プライベートキーで暗号化された上記ファイルを展開し、その内容から実行情報を取得し、当該実行情報に基づいて処理手順を決定及び実行する。このような実行方式によれば、実行制御部４０は、プログラマによって入力されたファイルの安全性（プログラマによって作成された真正のファイルであること）を確認した上で、上記ファイルを実行することができるため、アクセラレータプログラムの実行におけるセキュリティが向上する。また、モデルファイル２２ｃ、実行形式ファイル２２ｄ、及びバッチファイル２２ｅ等に記述された機密情報の漏えいを防止することができる。

【符号の説明】

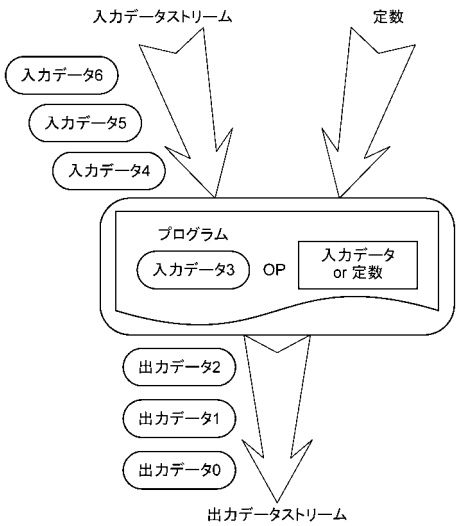
【０１３２】

１…アクセラレータ処理実行装置、１０…ＣＰＵ、２０…記憶部（第１の記憶手段）、２１…ＣＰＵ側メモリ、２１ａ…仮想バッファ、２１ｂ…規則情報記憶部、２２…補助記憶部、２２ａ…入力データファイル、２２ｂ…出力データファイル、２２ｃ…モデルファイル、２２ｄ…実行形式、２２ｅ…バッチファイル、３０…アクセラレータ、３１…プロセッサ、３２…アクセラレータ側メモリ、３２ａ…入力バッファ、３２ｂ…出力バッファ、４０…実行制御部、４１…実行形式入力部、４２…処理手順実行部、５０…バスブリッジ、Ｐ１…アクセラレータ処理実行プログラム、Ｐ４１…実行形式入力モジュール、Ｐ４２…処理手順実行モジュール。

【図１】



【図２】



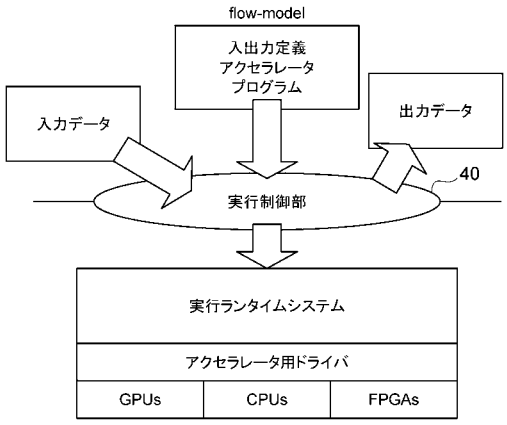
【図３】

```
<?xml version="1.0" encoding="ASCII"?>
<FlowModel xmlns:xsi="http://www.w3.org/2001/
                XMLSchema-instance" xmlns:xsd="http://www.w3.org/
                                              2001/XMLSchema">
<Input>
  <Name>a</Name>
  <DataType>INT</DataType>
  <Length>1024</Length>
  <Index>0</Index>
</Input>
<Input>
  <Name>b</Name>
  <DataType>INT</DataType>
  <Length>1024</Length>
  <Index>1</Index>
</Input>

<Output>
  <Name>c</Name>
  <DataType>INT</DataType>
  <Length>1024</Length>
  <Index>0</Index>
</Output>


<Kernel>
  __kernel void test(__global int *a, __global int *b,__global int *c

  {
  int id;

  id = get_global_id(0);
  //c[id] = + a[id] -100 + b[id]  ;
  c[id] = a[id] + b[id] + 100;
  return;
  }
</Kernel>

<KernelName>test</KernelName>
<LangType>SHADERLANG_OPENCL1.0</LangType>
<RuntimeType>RUNTIME_OPENCLGPU</RuntimeType>
<Binary />
<TotalNumThreads>1024</TotalNumThreads>
<TotalNumBlocks>128</TotalNumBlocks>
<SwapCounter>0</SwapCounter>
</FlowModel>
```

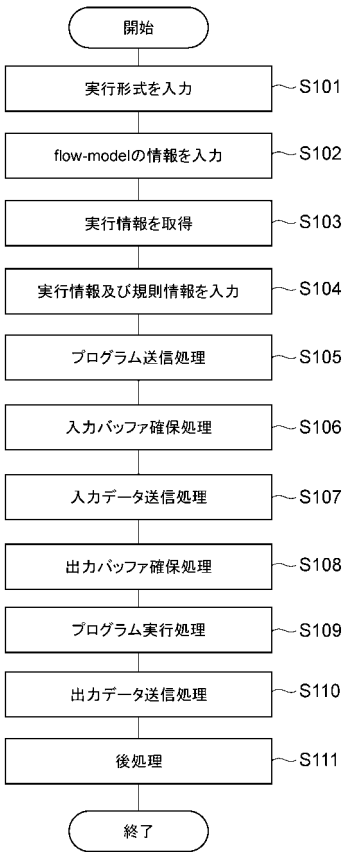
【図４】

```
<?xml version="1.0" encoding="ASCII"?>
<CarshEx>
  <ModelFile>sample1.xml</ModelFile>
  <Input>
    <Name>a</Name>
    <DataFile>sample1_a.csv</DataFile>
  </Input>
  <Input>
    <Name>b</Name>
    <DataFile>sample1_b.csv</DataFile>
  </Input>
  <Output>
    <Name>c</Name>
    <DataFile>sample1_c.csv</DataFile>
  </Output>
  <SwapPair>
    <Input>a</Input>
    <Output>c</Output>
    <NumSwap>1000</NumSwap>
  </SwapPair>
</CarshEx>
```

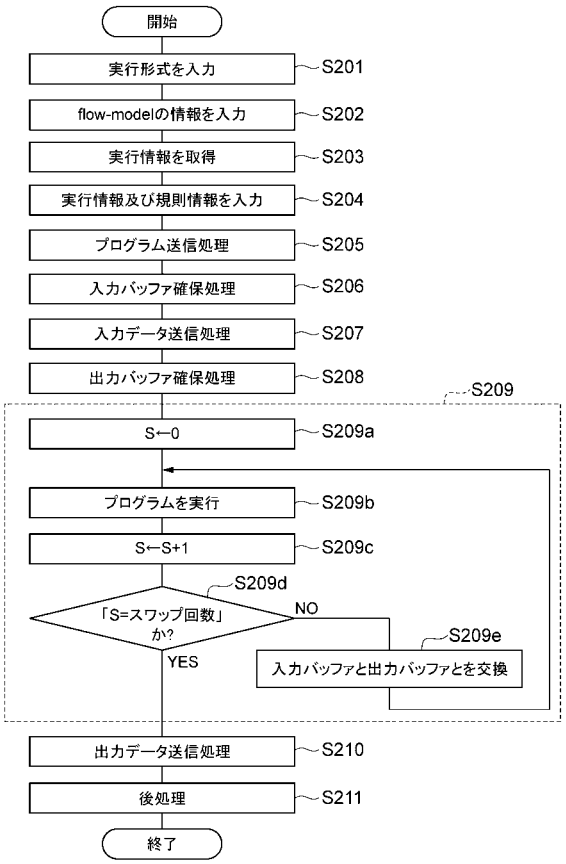
【図５】



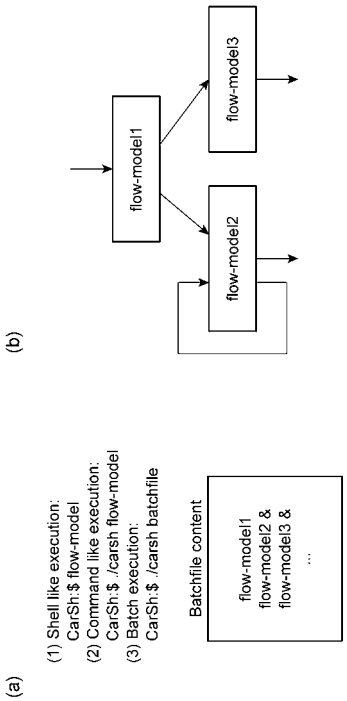
【図６】



【図７】



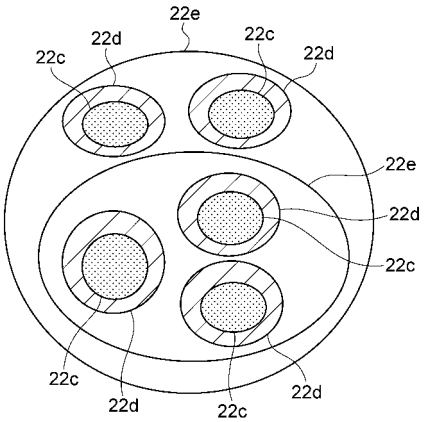
【図８】



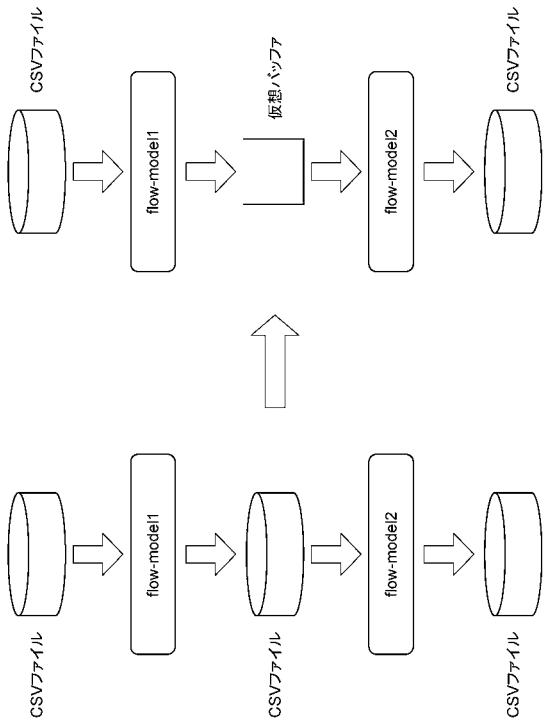
【図９】

```
<?xml version="1.0" encoding="ASCII"?>
<CarshBat>
virtbuf create int 1024 buf_a
virtbuf create int 1024 buf_b
virtbuf fill sample1_a.csv buf_a
virtbuf fill sample1_b.csv buf_b
virtbuf create int 1024 buf_c
debug off
timer start
sample_virt &amp;amp;
sync
timer stop
timer start
repeat 10 sample_virt.xml
sync
timer stop
virtbuf list
virtbuf dump c_tmp.csv buf_c
virtbuf list
virtbuf swap buf_a buf_c
virtbuf list
virtbuf delete buf_a
virtbuf list
virtbuf delete buf_b
virtbuf list
virtbuf delete buf_c
virtbuf list
exit
</CarshBat>
```

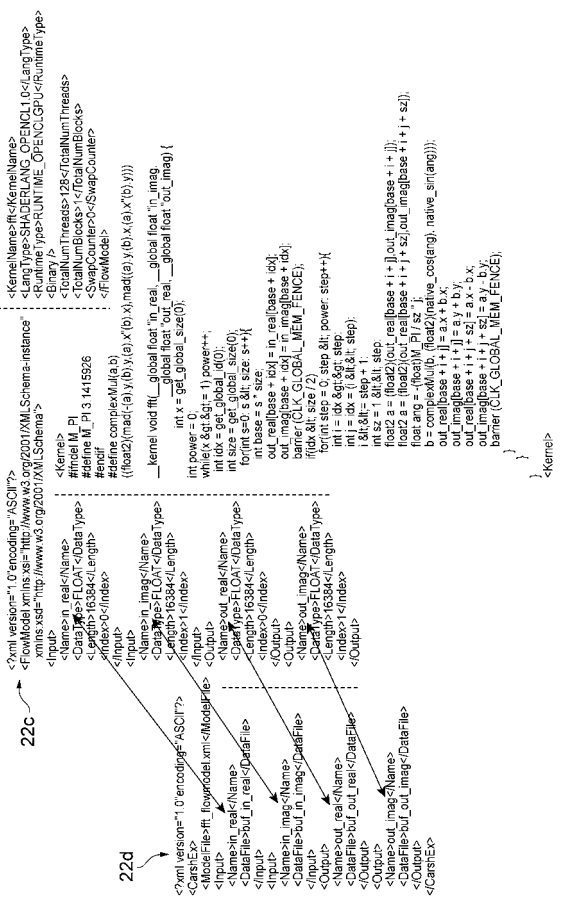
【図１０】



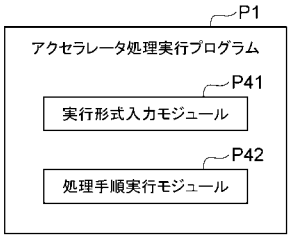
【図１１】



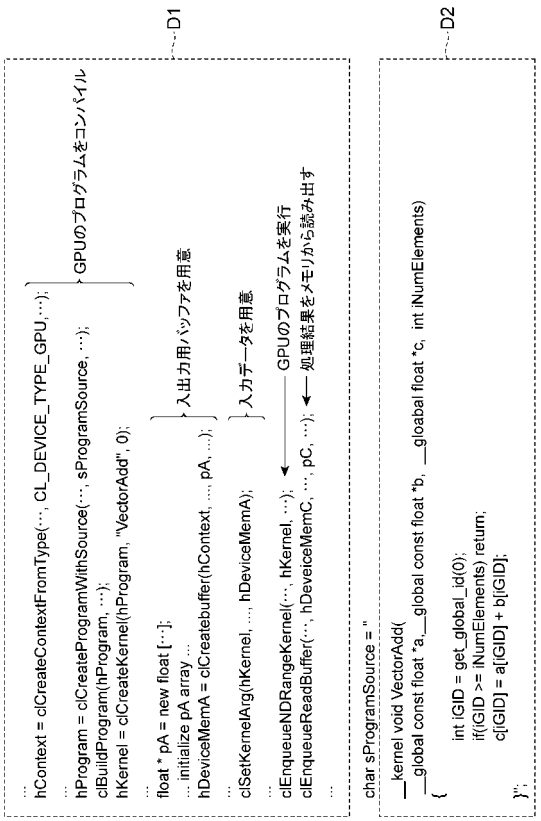
【図１３】



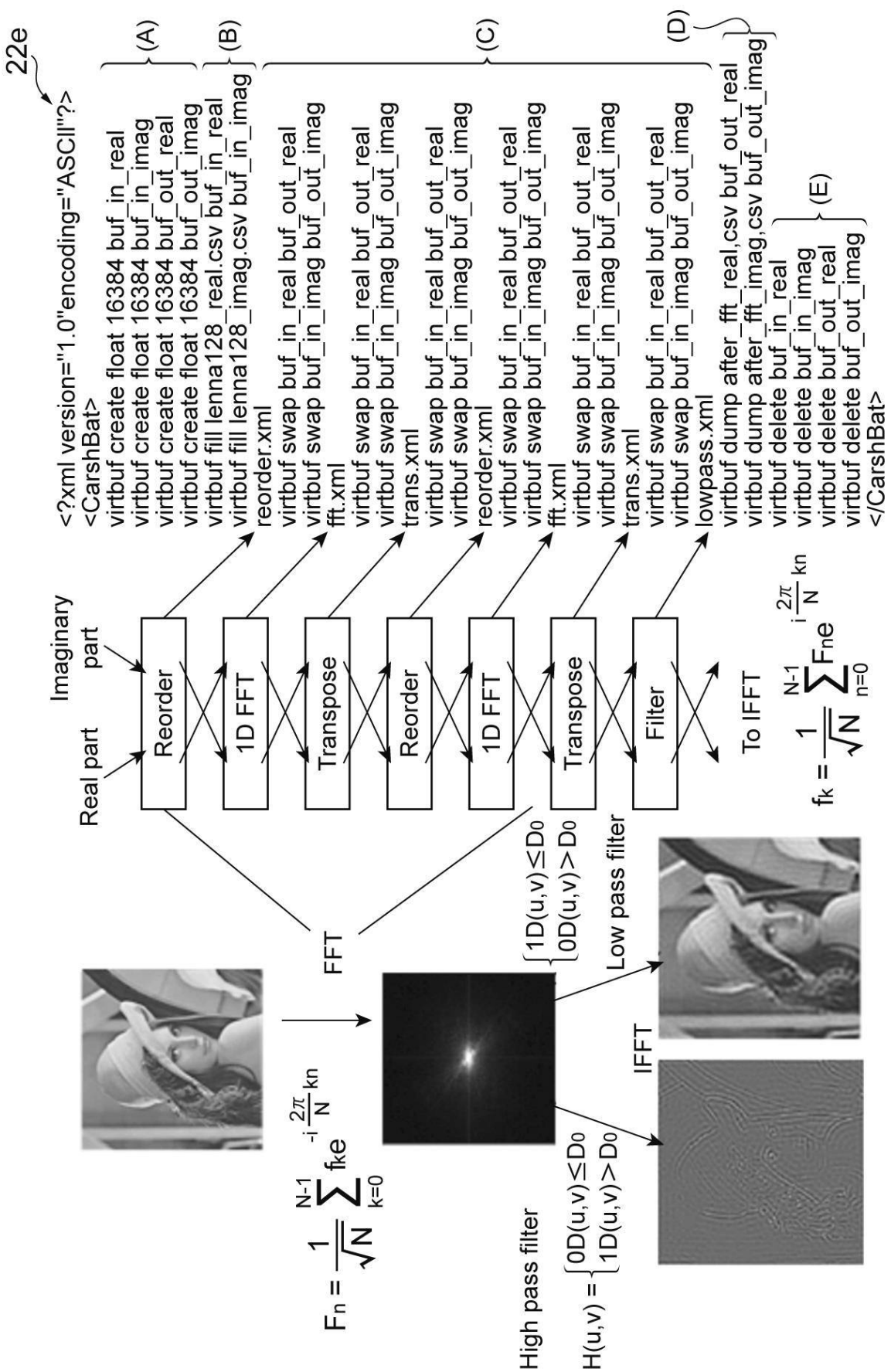
【図１４】



【図１５】



【図１２】


22e
<?xml version="1.0"encoding="ASCII"?>
<CarshBat>
virtbuf create float 16384 buf_in_real
virtbuf create float 16384 buf_in_imag
virtbuf create float 16384 buf_out_real
virtbuf create float 16384 buf_out_imag
virtbuf fill lenna128_real.csv buf_in_real
virtbuf fill lenna128_imag.csv buf_in_imag
reorder.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
fft.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
trans.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
reorder.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
fft.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
trans.xml
virtbuf swap buf_in_real buf_out_real
virtbuf swap buf_in_imag buf_out_imag
lowpass.xml
virtbuf dump after_fft_real,csv buf_out_real
virtbuf dump after_fft_imag,csv buf_out_imag
virtbuf delete buf_in_real
virtbuf delete buf_in_imag
virtbuf delete buf_out_real
virtbuf delete buf_out_imag
</CarshBat>
(A)
(B)
(C)
(D)
(E)
Imaginary part
Real part
Reorder
1D FFT
Transpose
Reorder
1D FFT
Transpose
Filter
To IFFT
FFT
IFFT
Low pass filter
High pass filter
$F_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-i\frac{2\pi}{N}kn}$
$f_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} F_n e^{i\frac{2\pi}{N}kn}$
$\begin{cases} 1D(u,v) \le D_0 \\ 0D(u,v) > D_0 \end{cases}$
$H(u,v) = \begin{cases} 0D(u,v) \le D_0 \\ 1D(u,v) > D_0 \end{cases}$