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

特許第4527571号
(P4527571)

(45)発行日　平成22年8月18日(2010.8.18)　(24)登録日　平成22年6月11日(2010.6.11)

(51)Int.Cl.　F I
*G06F 15/80 (2006.01)*　G06F 15/80
*G06F 9/38 (2006.01)*　G06F 9/38　370C

請求項の数 12　(全 28 頁)

(21)出願番号　特願2005-71320(P2005-71320)
(22)出願日　平成17年3月14日(2005.3.14)
(65)公開番号　特開2006-252440(P2006-252440A)
(43)公開日　平成18年9月21日(2006.9.21)
審査請求日　平成19年7月10日(2007.7.10)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100074099
弁理士　大菅　義之
(74)代理人　100067987
弁理士　久木元　彰
(72)発明者　齋藤　美寿
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　藤沢　久典
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

最終頁に続く

(54)【発明の名称】再構成可能演算処理装置

(57)【特許請求の範囲】
【請求項１】
コンフィギュレーション情報に基づき演算器の処理内容及び演算器間の接続関係が再構成される少なくとも２以上のクラスタから
構成される再構成可能演算処理装置において、
前記２以上のクラスタ間で共有利用される共有演算器を前記クラスタの外部に具備し、
前記共有演算器は、
前記クラスタから入力データと入力ｖａｌｉｄ信号を受け付ける入力手段と、
前記入力手段で前記入力ｖａｌｉｄ信号を受け取ると、前記入力ｖａｌｉｄ信号とともに受け取った前記入力データを演算処理する演算手段と、
前記演算手段の演算処理結果である出力データと、該出力データの出力先の前記クラスタを通知する出力ｖａｌｉｄ信号とを前記クラスタに出力する出力手段と、
を具備し、
前記２以上のクラスタは、
前記共有演算器と接続される第１クラスタと、
前記コンフィギュレーション情報に基づいて設定されるクロスバ又はセレクタを介して前記第１クラスタと接続することにより前記共有演算器から前記入力データ及び前記入力ｖａｌｉｄ信号を受け付ける第２クラスタと
を含む
ことを特徴とする再構成可能演算処理装置。

【請求項２】
前記共有演算器は、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤを生成し、前記ＩＤはどのクラスタから前記入力データと前記入力ｖａｌｉｄ信号を受け付けたかを識別する信号であり、前記ＩＤが前記演算処理結果とともに前記出力手段に伝達され、前記出力手段は、前記ＩＤにしたがって、前記入力データと前記入力ｖａｌｉｄ信号の発行元のクラスタに出力することを特徴とする請求項１に記載の再構成可能演算処理装置。
【請求項３】
前記共有演算器は、パイプライン構成を利用して演算処理をすることを特徴とする請求項１に記載の再構成可能演算処理装置。
【請求項４】
前記パイプラインは、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤとともに内部ｖａｌｉｄを生成し、前記入力データを演算処理した処理データとともに前記ＩＤを、前記内部ｖａｌｉｄを利用して前記パイプラインで転送することを特徴とする請求項３に記載の再構成可能演算処理装置。
【請求項５】
前記共有演算器の構成は、単独演算を行うアプリケーション特化エンジンであることを特徴とする請求項１に記載の再構成可能演算処理装置。
【請求項６】
前記共有演算器の構成は、単独演算を行う複数のアプリケーション特化エンジンを配設した構成であることを特徴とする請求項１に記載の再構成可能演算処理装置。
【請求項７】
前記共有演算器は、前記アプリケーション特化エンジンの切替えを前記クラスタからの前記入力データと前記入力ｖａｌｉｄ信号に基づいて演算処理選択信号を生成し、前記演算処理選択信号により前記共有演算器のアプリケーション特化エンジンの切替えをすることを特徴とする請求項６に記載の再構成可能演算処理装置。
【請求項８】
前記演算処理選択信号は、前記共有演算器の有する前記アプリケーション特化エンジンに対応した演算処理コードからなるテーブルを予め設定し、前記クラスタから前記アプリケーション特化エンジンを選択するために、前記入力データである前記演算処理コードを入力し、前記入力データに対応する前記アプリケーション特化エンジンの切替え制御のための信号を選択して、前記演算処理選択信号を生成することを特徴とする請求項７記載の再構成可能演算処理装置。
【請求項９】
前記共有演算器の入力手段と出力手段は、コンフィギュレーションデータに基づき再構成可能なセレクタを配設していることを特徴とする請求項１に再構成可能演算処理装置。
【請求項１０】
前記共有演算器の入力手段と出力手段は、クロスバスイッチを配設していることを特徴とする請求項１に再構成可能演算処理装置。
【請求項１１】
コンフィギュレーション情報に基づき、再構成される少なくとも１以上のクラスタから構成される再構成可能演算処理装置において、
前記１以上のクラスタ間で共有利用される共有演算器を前記クラスタの外部に具備し、
前記共有演算器は、
前記クラスタから入力データと入力ｖａｌｉｄ信号を受け付ける入力手段と、
前記入力手段で前記入力ｖａｌｉｄ信号を受け取ると、前記ｖａｌｉｄ信号とともに受け取った前記入力データを演算処理する演算手段と、
前記演算手段の演算処理結果である出力データと、該出力データの出力先の前記クラスタを通知する出力ｖａｌｉｄ信号とを前記クラスタに出力する出力手段と、
を具備し、

前記共有演算器は、パイプライン構成を利用して演算処理をし、
前記パイプラインは、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤとともに内部ｖａｌｉｄを生成し、前記入力データを演算処理した処理データとともに前記ＩＤを、前記内部ｖａｌｉｄを利用して前記パイプラインで転送する
ことを特徴とする、再構成可能演算処理装置。
【請求項１２】
クラスタの各々が、
演算器群と、
コンフィギュレーションメモリと、
シーケンサと、
前記クラスタ同士を接続する、クロスバと
を含み、
前記演算器群が、
複数の演算器と、
データメモリと、
データ入力手段と、
データ出力手段と、
前記複数の演算器と前記データメモリと前記データ入力手段と前記データ出力手段との相互接続を行う、演算器間ネットワーク手段と
を含み、前記コンフィギュレーションメモリから供給される前記コンフィギュレーション情報により、前記複数の演算器の処理内容及び前記演算器間ネットワーク手段の構成を変更でき、
前記コンフィギュレーション情報が複数のコンフィギュレーション情報を含み、
前記コンフィギュレーションメモリが前記複数のコンフィギュレーション情報を保持し、
前記シーケンサが、前記複数のコンフィギュレーション情報のうちから選択的にコンフィギュレーション情報を前記演算器群に供給することで、前記クロスバの接続先を変更してコンフィギュレーション状態の管理を行い、
前記クロスバが、
前記演算器群の入出力ポートに接続して、前記演算器群を有するクラスタの外部とのデータ入出力を行い、クラスタ同士のあいだでのデータ転送を行う
ことを特徴とする、請求項１～１１のいずれか一項に記載の再構成可能演算処理装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、複数のＡＬＵ（算術演算器）等で構成されるＡＬＵアレイと、ＡＬＵアレイ内のＡＬＵの命令やＡＬＵ間の接続の制御をコンフィギュレーション情報によって設定する技術に係り、特にはコンフィギュレーション情報の入れ替え等をシーケンサで行う再構成可能演算回路（クラスタ）を複数有する演算装置における再構成可能演算回路間で共有利用される演算器の構成に関する。
【背景技術】
【０００２】
今日、再構成可能演算回路（以下クラスタ）を有する再構成可能演算処理装置は、クラスタを複数配置して演算処理をすることにより演算処理速度の向上、演算装置の小型化などを行う提案がされている。図２０は演算装置の一例を示したものであり、それぞれのクラスタは、例えばクロスバ接続などにより接続され、クラスタ間のデータ転送を可能にしている。
【０００３】
そして、一つのクラスタはＡＬＵアレイ部を有し、ＡＬＵアレイ部には複数の演算器が

設けられている。演算器は通常ＡＬＵや乗算器などで構成されている。
例えば、クラスタは図２１に示すような構成となっている。（図２１は従来の再構成可能演算処理装置内のクラスタの構成を概念的に示す略ブロック図である）
クラスタ１は、演算器群２（ＡＬＵアレイ部）、コンフィギュレーションメモリ３、シーケンサ４から構成されている。

【０００４】

演算器群２は、データ入力部５、データバッファ部６、データバッファ制御部７、演算器間ネットワーク８、データメモリ９、演算器１０から構成されている。
データ入力部５は、外部から入力される入力データを、演算器間ネットワーク８を介してデータメモリ９、各演算器１０などに供給する。例えば、データ入力部５の構成例として、データバッファ部６を具備する構成とし、データバッファ部６は外部より入力される入力データをバッファする／しないの有無をデータバッファ制御部７からの制御信号により選択する。データバッファ制御部７は、コンフィギュレーションメモリ３からコンフィギュレーション情報を受け、その情報に従い、上記制御信号としてデータバッファ部６に制御信号を送り、入力データのバッファの有無を選択する。

【０００５】

演算器間ネットワーク８は、種々の要素（データ入力部５、データメモリ９、演算器１０など）と相互接続されている。また、演算器間ネットワーク８は、外部から供給されるコンフィギュレーションデータ（プログラム：Ｃ言語、ＨＤＬなどで作成されるソース）に基づいて生成されるコンフィギュレーション情報（プログラムをコンパイルして生成されるデータ）に応じて、演算器間ネットワーク８に接続されている種々の要素間のデータ転送を可能とする。データメモリ９は、演算器間ネットワーク８を介しデータを記録する。演算器１０は、コンフィギュレーション情報により、そのコンフィギュレーション情報に関係付けられた機能を果たすように設定され、その設定された演算を実行する。

【０００６】

コンフィギュレーションメモリ３は、コンフィギュレーション情報を格納する外部記憶装置（図示せず：例えばＰＣなど）からコンフィギュレーションメモリ３にコンフィギュレーション情報をロードする。（例えば、ＰＣの通信手段を利用してロードする）そして、コンフィギュレーションメモリ３には、コンフィギュレーションデータロード部（図示しない）、演算器群２を構成する再構成可能な種々の要素のうち主に演算器１０から送信される条件成立信号（例えばチップセレクトのような信号）に基づいてコンフィギュレーション切替条件信号を生成し出力する。例えば、コンフィギュレーション切替条件信号の生成は、上記条件成立信号とコンフィギュレーションメモリ３からのコンフィギュレーションデータに基づいて生成する。

【０００７】

シーケンサ４は、切替条件信号に基づいてコンフィギュレーションメモリ３が次に読み出すべき上記コンフィギュレーション情報のアドレスを生成する。
特許文献１によれば、個々にデータ設定される命令コードに対応してデータ処理を個々に実行するとともに相互の接続関係を切換制御する多数のプロセッサエレメントが行列形状に配列されており、これら多数のプロセッサエレメントの命令コードを状態管理部で順次切り換える。ただし、状態管理部は、相互通信して連携動作する複数からなり、これと同数のエレメント領域に多数のプロセッサエレメントが区分されている。複数のエレメント領域ごとに複数の状態管理部が個々に配置されてプロセッサエレメントに接続されているので、小規模な複数の状態遷移を複数の状態管理部で個別に管理できる。また、大規模な一つの状態遷移を複数の状態管理部で協調して管理することができる。

【０００８】

また、特許文献２によれば、プロセッサエレメントをアレイ状に並べたものをプログラマブルなスイッチで電気的に接続し、演算を主体として行うデータパス部と、状態遷移の制御を行う状態遷移管理部を独立して設け、それぞれを処理目的に応じて特化した構成を実現することで、小型化、高性能化の可能なアレイ型プロセッサが提案されている。

【０００９】
そして、上記説明したような構成の再構成可能演算処理装置において演算処理を行う際に、場合によって除算処理等の演算負荷の大きい処理も必要になることがある。このような場合、図２２に示す専用のハードウェアアクセラレータを使用して、ＣＰＵやＤＭＡＣ（Ｄｉｒｅｃｔ Ｍｅｍｏｒｙ Ａｃｃｅｓｓ Ｃｏｎｔｒｏｌｌｅｒ）を介在させて演算処理をする方法などが提案されている。
【特許文献１】特開２００４－１３３７８１号公報
【特許文献２】特開２００１－３１２４８１号公報
【発明の開示】
【発明が解決しようとする課題】
【００１０】
しかしながら、図２２に示すような方法で除算処理等の演算処理を行う場合、処理開始やデータの転送を簡単化するためにＣＰＵやＤＭＡＣを介在させなければならない。そのことでインターフェイスは統一されるが、ＣＰＵが介在するため演算処理能力が低下するという問題がある。一方ＣＰＵを介在させない場合を考えると、インターフェイスに統一性をもたせるのが難しくなり、新規にハードウェア設計をする毎にインターフェイスを考えることが必要になり、設計資産を簡単に再利用することが困難である。
【００１１】
そこで、上記クラスタを複数もつ再構成可能演算処理装置に直接除算器等を配置する方法が考えられる。例えばクラスタ内の演算器群２（ＡＬＵアレイ部）に配置すると、処理能力を向上させることができるし、除算器のような汎用演算器ではなく、アプリケーションに特化したハードウェアを演算器群２（ＡＬＵアレイ部）に持てばさらに処理能力を向上させることができる。
【００１２】
しかし、（１）乗算器やＡＬＵに比べると除算器などの汎用演算器は回路規模が大きい、（２）ＡＬＵや乗算器に比べると使用頻度が小さく、そのため再構成可能演算回路内に除算器やアプリケーション特化エンジンを設けると、面積あたりの演算器使用効率が悪くなり、コスト増につながる。また、（３）アプリケーション特化エンジンを設けると、他のアプリケーションでは全く使用しない無駄なものになるため取り除くといった再設計が必要となる。
【００１３】
そのため、除算器などのアプリケーション特化エンジンをクラスタの外部に置き、クラスタ間で共有できれば、面積効率や使用効率を上げられる。また、クラスタの外部にあるので、異なるアプリケーションに対しては、そのアプリケーション特化エンジンを交換すればよい。
【００１４】
しかし、クラスタの外部に置いて、複数のクラスタで共有して使用する場合、以下の問題が発生する。（４）クラスタの外部に、単に配置しても上記説明したように、ＣＰＵなどを介在させなければ、処理の起動ができない。そのため、なんらかの起動方法が必要になる。（５）任意のクラスタからアプリケーション特化エンジンを利用するためのクラスタとの接続方法と制御方法が必要になる。（６）アプリケーション特化エンジンを通常のハードウェアのように、エンジン毎に特有の信号線等を用いた場合、異なるアプリケーションに対して、アプリケーション特化エンジンの交換などはできない。そのため、交換できる仕組みが必要である。（７）クラスタの外部に置いたアプリケーション特化エンジンを複数のクラスタで共有しようとした場合、共通の制御が必要になる。
【００１５】
また、特許文献１によれば、二つのクラスタで共有する、共有リソースなるものがあるが、どちらか一方のクラスタに制御される必要がある。また、特許文献２によれば、クラスタの外部に乗算器を有しているが、その使用方法、動作については開示されていない。
【００１６】

本発明は上記のような実情に鑑みてなされたものであり、アプリケーション特化エンジン等の共有演算器を、クラスタから受け取るデータとＶａｌｉｄ信号とを用いて演算処理し、クラスタ間のネットワークを介すことで、複数のクラスタから共有演算器を利用できる再構成可能演算処理装置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】

【００１７】

請求項１に記載の発明によれば、コンフィギュレーション情報に基づき、再構成される少なくとも１以上のクラスタから構成される再構成可能演算処理装置において、上記１以上のクラスタ間で共有利用される共有演算器を上記クラスタの外部に具備し、上記共有演算器は、上記クラスタから入力データと入力ｖａｌｉｄ信号を受け付ける入力手段と、上記入力手段で上記入力ｖａｌｉｄ信号を受け取ると、上記ｖａｌｉｄ信号とともに受け取った上記入力データを演算処理する演算手段と、上記演算手段の演算処理結果である出力データと、該出力データの出力先の上記クラスタを通知する出力ｖａｌｉｄ信号とを上記クラスタに出力する出力手段と、を具備する構成とする。

【００１８】

複数のクラスタで構成される演算装置に、アプリケーション特化エンジンなどの除算器等の演算器を、クラスタの外部に演算装置の共有演算器として設ける。そのクラスタ間を結ぶネットワークに接続する。このときネットワーク上で遣り取りする信号は、データとデータが有効であることを意味する「ｖａｌｉｄ信号」で構成する。クラスタはｖａｌｉｄ信号を受け取ると演算処理がなされる構成になっておりクラスタから当該共有演算器へはデータとともにｖａｉｌｄ信号が送られる。当該共有演算器ではｖａｌｉｄ信号を受け取ると、Ｖａｉｌｄ信号とともに受取ったデータの処理を開始し、処理完了時にｖａｉｌｄ信号とともに結果データを出力する。

【００１９】

請求項２に記載の発明によれば、前記共有演算器は、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤを生成し、前記出力データの出力先の前記クラスタを通知する構成とする。

【００２０】

請求項３に記載の発明によれば、前記共有演算器は、パイプライン構成を利用して演算処理をする構成とする。

請求項４に記載の発明によれば、前記パイプラインは、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤとともに内部ｖａｌｉｄを生成し、前記入力データを演算処理した処理データとともに前記ＩＤを、前記内部ｖａｌｉｄを利用して前記パイプラインで転送する構成とする。

【００２１】

請求項５に記載の発明によれば、前記共有演算器の構成は、単独演算を行うアプリケーション特化エンジンである構成とする。

請求項６に記載の発明によれば、前記共有演算器の構成は、単独演算を行う複数のアプリケーション特化エンジンを配設した構成とする。

【００２２】

請求項７に記載の発明によれば、前記共有演算器は、前記アプリケーション特化エンジンの切替えを前記クラスタからの前記入力データと前記入力ｖａｌｉｄ信号に基づいて演算処理選択信号を生成し、前記演算処理選択信号により前記共有演算器の前記アプリケーション特化エンジンの切替えをする構成とする。

【００２３】

請求項８に記載の発明によれば、前記演算処理選択信号は、前記共有演算器の有する前記アプリケーション特化エンジンに対応した演算処理コードからなるテーブルを予め設定し、前記クラスタから前記アプリケーションを選択するために、前記入力データである前記演算処理コードを入力し、前記入力データに対応する前記アプリケーションの切替え制御のための信号を選択して、前記演算処理選択信号を生成する構成とする。

【００２４】
請求項９に記載の発明によれば、前記共有演算器の入力手段と出力手段は、コンフィギュレーションデータに基づき再構成可能なセレクタを配設している構成とする。
請求項１０に記載の発明によれば、前記共有演算器の入力手段と出力手段は、クロスバスイッチを配設している構成とする。
【００２５】
上記構成にすることで、共有演算器はｖａｌｉｄ信号で起動でき、利用する側の再構成演算回路からの特殊な制御なしに起動できる。
また、共有演算器はクラスタ間を結ぶネットワーク上に接続することにより、任意のクラスタからデーダ＋ｖａｌｉｄ信号を送るだけで、利用するクラスタから制御借号を送ることなしに利用することができる。
【００２６】
また、共有演算器は、ｖａｌｉｄ信号で起動し、同時に受け取ったデータを処理する。そのためアプリケーション特化エンジンをこのルールで構成することにより、異なるアプリケーション毎に、特殊な信号を設けることが不要になるため、エンジンを交換することが容易である。
【００２７】
さらに、各クラスタからは制御信号なしに、共有演算器を共有でき、パイプライン構成にすることで、各ステージで異なるポートからのデータ処理を並列で実行することができる。
【発明の効果】
【００２８】
本発明によれば、共有演算器をクラスタの外部に置き、クラスタ間で共有することにより、面積効率や使用効率を上げられる。また、クラスタの外部にあるので、異なるアプリケーションに対して、共有演算器のアプリケーション特化エンジンを交換することが容易にでき、設計資産の再利用が効率よくできる。
【発明を実施するための最良の形態】
【００２９】
以下図面に基づいて、本発明の実施形態について詳細を説明する。
（実施例１）
図１は、クラスタ１と共有演算器１１の構成を示した図である。クラスタ間はクラスタ間ネットワークにより接続され、複数のクラスタ１と共有演算器１１は、ポート（ｐｏｒｔ０、ｐｏｒｔ１）により接続される。同図のｐｏｒｔ０、ｐｏｒｔ１は共有演算器１１とクラスタ１を接続するように構成され、例えば入力としてｐｏｒｔ０データ入力（１６ビットバス）、ｐｏｒｔ０ｖａｌｉｄ入力を設置する。また、出力としてｐｏｒｔ０データ出力（１６ビットバス）、ｐｏｒｔ０ｖａｌｉｄ出力を設置する。
【００３０】
図２はクラスタの動作について説明した図である。クラスタＡはデータ＋ｖａｌｉｄ信号によりｖａｌｉｄ駆動で動作する。またクラスタ間はデータとｖａｌｉｄ信号をデータ転送する構成とする。
【００３１】
例えば、クラスタＡに外部からデータとｖａｌｉｄ信号が入力され、クラスタＡでの当該データに対する処理が行われる。その結果がクラスタＡからデータとｖａｌｉｄ信号の形式で出力されクラスタＢに渡される。
【００３２】
さらに、クラスタＢでは受け取ったデータとｖａｌｉｄ信号により当該データによる処理を行い、処理終了後クラスタＢから新たにデータとｖａｌｉｄ信号の形式で出力される。
【００３３】
図３は上記図２の動作をタイムチャートで示した図である。クロックＣＬＫの立下りエ

ッジにより、クラスタＡへの入力データであるクラスタＡ入力データと、クラスタＡへのｖａｌｉｄ信号であるクラスタＡ入力ｖａｌｉｄ信号をクラスタＡ内に取込み内部処理をする。本例ではクラスタＡ内の処理としてクロックＣＬＫ８サイクル分の処理を実行する。ここで、クラスタＢへの入力信号は、クラスタＡ内で処理し８サイクル目でクラスタＡ出力データ（クラスタＢ入力データと同等）と、クラスタＡ出力ｖａｌｉｄ（クラスタＢ入力ｖａｌｉｄと同等）を用意する。そして、クラスタＢに取込みクラスタＢ内で処理を行い、次のクラスタ１または共有演算器１１にクラスタＢ出力データとクラスタＢ出力ｖａｌｉｄを出力する。

【００３４】

図４はクラスタＣと共有演算器Ａが接続された場合の例である。アプリケーション領域にターゲットを絞った場合に、アルゴリズムは多少違っていても共通で使用される複雑な処理を、アプリケーション特化エンジンとして共有演算器１１を外部に設ける。

【００３５】

共有演算器Ａの場合もクラスタ間のときと同様に、データ＋ｖａｌｉｄ信号で駆動とデータ転送をする。すなわちデータ＋ｖａｌｉｄ信号で任意のクラスタＣと遣り取りが可能になる。この駆動方法により共有演算器Ａの中身は任意となり、共有演算器Ａのアプリケーション領域（演算器）に必要なエンジンに置き換えることが可能になる。図４ではクラスタＣへの入力データがデータ０とｖａｌｉｄ０信号によって入力される。入力されたデータ０とｖａｌｉｄ０はクラスタＣ内部で処理され、データ１とｖａｌｉｄ１信号が共有演算Ａの入力として演算し求められる。

【００３６】

共有演算器Ａで内部処理されたデータ１とｖａｌｉｄ１信号は、再びクラスタＣに処理結果を返すためにデータ２とｖａｌｉｄ２信号を演算し求める。そしてクラスタＣはさらに内部処理を行いデータ３とｖａｌｉｄ３信号を出力結果とする。

【００３７】

図５を用いクラスタＣと共有演算器Ａの動作についてのタイムチャートを示す。図３と同様にクラスタＣ入力データ０とクラスタＣ入力ｖａｌｉｄ０信号をクロックＣＬＫの立上りエッジでクラスタＣに取込み、クラスタＣの内部処理をしクロックの８サイクル目までに共有演算器Ａにクラスタ Ｃ入力データ１（共有演算器入力データ１）とクラスタＣ入力ｖａｌｉｄ１信号（共有演算器入力ｖａｌｉｄ１）を入力する。共有演算器Ａは受け取った入力を演算処理しクロックの７サイクル目までに出力結果クラスタＣ入力データ２（共有演算器入力データ２）とクラスタＣ入力ｖａｌｉｄ２信号（共有演算器入力ｖａｌｉｄ２）を算出する。そして、クラスタＣは入力を受取り後半の処理を行い、クロックの６サイクル目までにクラスタＣ入力データ３とクラスタＣ入力ｖａｌｉｄ３信号の算出をする。

【００３８】

ここで、クラスタＡ、Ｂ、Ｃ、共有演算器Ａの内部処理において、当然であるがクロックの立上りエッジでの取込み方法、および各処理に必要クロックの数は特に限定されているものではない。

【００３９】

なお、図２、４ではクラスタ間をクロスバスイッチ経由で接続しているが、これはクラスタ間接続方法の一実現方法であり、データ転送がデータ＋ｖａｌｉｄ信号で、ｖａｌｉｄ駆動ができる接続であれば特に限定するものではない。

【００４０】

次に、共有演算器１１の概略の構成について説明する。共有演算器１１には入力ポートと演算部と出力ポートが設けられている。そして上記説明した図４の構成においては入力ポートと出力ポートは各１ポートによる単純な構成とした。しかし実際にはマルチポートとして使用することが可能である。例えば構成として複数のクラスタ１に対し共有演算器１１を１つ用意する。そして、共有演算器１１にはクラスタ分の入出力ポートを用意してもよい。

【００４１】
また、クラスタ分の入出力ポートを用意しない場合は、共有演算器１１との接続切替えをすることで、共有演算器１１を使用していないときはクラスタ１から外し、必要なときにはクラスタ１を再度接続することも可能である。接続の切替えはクロック単位で設定してもよいし、新規にコンフィギュアブル情報が設定されるまで固定することも可能である。

【００４２】
図６にマルチポート構成された共有演算器の図を示す。共有演算器１１は入力ポート０～Ｎ－１（Ｎ：整数）で処理される。そして演算器の演算結果はマルチポート出力制御を経て出力ポート１～Ｍ－１（Ｍ：整数）から出力される。ここで、Ｍ＝Ｎであってもいいし、Ｍ≠Ｎであってもよい。

【００４３】
ここで、マルチポート入力制御は入力に競合があるときは、例えば番号の小さい入力ポートを優先にし、その他のデータを破棄することで１つの入力ポートを選択する。また、出力ポートに関してもｖａｌｉｄ信号から生成されるポート情報（ｖａｌｉｄ信号と出力ポートの識別信号であるＩＤ）により、１つの出力ポートのみアクティブにする。なお、データは全出力ポートとも同値とすることが好適であるが必ずしも全ポートに出力する必要はない。なお、ポート情報は演算器のレイテンシ分ディレーさせてから入力し、マルチポート出力制御を制御する。

【００４４】
図７の例は２ポートの共有演算器１１の基本構成を示したものである。上記図５で説明した共有演算器１１のマルチポート入力制御は、ｖａｌｉｄ信号の受付およびＩＤ生成部から構成されている。入力であるＰｏｒｔ０側またはＰｏｒｔ１側からのｖａｌｉｄ信号（ｖａｒｉｄＩｎＡ＿０、ｖａｒｉｄＩｎＢ＿０、ｖａｒｉｄＩｎＡ＿１、ｖａｒｉｄＩｎＢ＿１）の入力により内部ｖａｌｉｄ信号を生成し、この内部ｖａｌｉｄ信号を用いて演算処理をパイプラインのように実行する。また当該ｖａｌｉｄ信号の受付およびＩＤ生成部で、どの入力ポートから受けたデータであるかを通知するためにＩＤを生成し、そのＩＤに基づき入力データを受け取ったポートを選択する。

【００４５】
ここで、ｖａｌｉｄ信号の受付およびＩＤ生成部は、同図のソースにあるように演算を実行する。もし、ｖａｌｉｄＩｎＡ＿０とｖａｌｉｄＩｎＢ＿０が有効であればゲート信号によりｄａｔａＩｎＡ＿０とｄａｔａＩｎＢ＿０を演算器に取り入れるか／取り入れないかを選択する。ゲート信号が「１」（取り入れる）であれば、入力データを演算器に取り入れる。それと同時に内部ｖａｌｉｄをＯＮにし、ＩＤを「０」とする。

【００４６】
また、もしｖａｌｉｄＩｎＡ＿１とｖａｌｉｄＩｎＢ＿１が有効であればゲート信号によりｄａｔａＩｎＡ＿１とｄａｔａＩｎＢ＿１を演算器に取り入れるか／取り入れないかを選択する。ゲート信号が「１」（取り入れる）であれば入力データを演算器に取り入れる。それと同時に内部ｖａｌｉｄをＯＮにし、ＩＤを「１」とする。

【００４７】
なお、ゲート信号によってデータを取り入れる構成は、ＡＮＤマスクなどにより、一方を入力データの入力とし、ＡＮＤゲートの他方をゲート信号とするようにしてもよい。また、ｖａｌｉｄによるゲート信号の生成は、必ずしも必要でない。

【００４８】
さらに、ｖａｌｉｄ入力がＯＦＦであれば、入力ポート（ｐｏｒｔ０、１）を選択するセレクタが優先的にどちらかを選択して、ステージ０の演算を行ってもよい。その理由はｖａｌｉｄ入力がＯＦＦの場合には、内部ｖａｌｉｄがＯＦＦになる。そのため中間データ０の値がステージ０の最後のＦＦに書込まれないため、計算をしていないのと同様になる。しかし、このゲート信号を上記説明したような無駄な動作を停止できるため、消費電力の低減には効果がある。

【００４９】
次に、ＩＤは処理データとともにパイプラインで転送され、演算が完了するステージ（本例では演算器ステージ２）と同時にＩＤをデコードし、出力ポートの選択をするためのｖａｌｉｄ信号を生成する。このｖａｌｉｄ信号は出力ポートに対して発行される。
【００５０】
次に、処理データは両出力ポートに発行される。（この例では両方に発行しているが、ＩＤに基づいて選択発行してもよい）
出力されたデータとｖａｌｉｄ信号は入力データを発行したクラスタ１へと送られる。クラスタ１では、ｖａｌｉｄ信号を受け取ると一緒にきたデータの処理を行う。
【００５１】
図６の演算器は、例えば入力ポートから入力された入力データを、図７の演算器（ステージ０～２の組み合わせ回路）のように構成し演算をする。その演算結果を中間データ０～１のＦＦ（２）（フリップ・フロップ）に格納する。また、上記ＩＤも同様にＩＤをＦＦに格納する。そして、内部ｖａｌｉｄ信号により中間データ０の各ＦＦ（２）と、ＩＤのＦＦ（３）と、内部ｖａｌｉｄのＦＦ（１）をイネーブルにし、次の演算器ステージ１に出力する。同様に演算器ステージ１でも演算をして、結果を中間データ１のＦＦ（５）に格納する。また、ＩＤもＩＤ用のＦＦ（６）に格納する。そして、内部ｖａｌｉｄ信号により中間データ１の各ＦＦ（５）と、ＩＤのＦＦ（６）と、内部ｖａｌｉｄのＦＦ（４）をイネーブルにし、次の演算器ステージ２に出力する。
【００５２】
演算器ステージ２では、演算器ステージ１の出力を入力とし、演算器ステージ２の演算処理をし、処理結果データを出力としＦＦに格納する。
また、マルチポート出力制御では、ＩＤデコーダでデコードし、どの出力ポート（ｐｏｒｔ０出力、ｐｏｒｔ１出力）のｖａｌｉｄ信号（ｖａｌｉｄＯｕｔ＿０、ｖａｌｉｄＯｕｔ＿１）を有効にするかを決めＦＦ（８）（９）に格納する。
【００５３】
その後内部ｖａｌｉｄによりＦＦがイネーブルのときに出力ポート（ｐｏｒｔ０出力、ｐｏｒｔ１出力）に出力される。
次に、図８は図７の動作をタイムチャートで示した図である。クロックＣＬＫの立上りエッジで入力データが確定し取込みが行われる。

ＣＬＫ１の期間では、クラスタ１より共有演算器１１にデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿０は「ａ０」（本例ではｈｉｇｈ信号）、ｄａｔａＩｎＡ＿０はデータ（例えば１６ビット幅のデータ「＃」）が転送される。また、同様にｖａｌｉｄＩｎＢ＿０もｈｉｇｈ、ｄａｔａＩｎＢ＿０にもデータが転送される。
【００５４】
また、上記ｖａｌｉｄ信号に基づきｖａｌｉｄ信号の受付およびＩＤ生成部でゲート信号とＩＤが生成される。ＩＤはｐｏｒｔ０側「０」を選択し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ａ０」を選択する。ここで、ゲート信号と「ａ０」がＣＬＫ１立ち上がりエッジより遅れているのは、ｖａｌｉｄ信号の受付およびＩＤ生成部演算処理の演算によるものである。そして、「＃」に基づき演算器ステージ０の演算が実行される。
【００５５】
そして、ｖａｌｉｄＩｎＡ＿０の信号である「ａ０」が、中間データ０用の内部ｖａｌｉｄのＦＦ（１）の入力となる。また中間データ０のＦＦ（２）の入力には、ｄａｔａＩｎＡ＿０のデータである「＃」が演算処理された結果「＃１」が与えらる。また、中間データ０に同期したＩＤのＦＦ（３）への入力として「０」が与えられる。これらのＦＦ（１）、（２）、（３）はＣＬＫ２の立上りエッジで取り込み、確定し、ＣＬＫ２サイクル期間出力する出力する。

ＣＬＫ２の期間では、「＃１」に基づき演算器ステージ１の演算が実行される。

【００５６】
このとき、上記中間データ０に関する各ＦＦに保持されているｖａｌｉｄ信号「ａ０」、ＩＤ「０」、演算器ステージ１の演算結果「＃２」を中間データ１に関するＦＦに転送する。中間データ１用の内部ｖａｌｉｄのＦＦ（４）の入力には「ａ０」が与えられる。また中間データ１のＦＦ（５）の入力には演算器ステージ１の演算結果「＃２」が与えられる。また、中間データ１に同期したＩＤのＦＦ（６）の入力には「０」が与えられる。これらのＦＦ（４）、（５）、（６）はＣＬＫ３の立上りエッジで取り込み、確定し、ＣＬＫ３サイクル期間出力する出力する。

ＣＬＫ３の期間では、クラスタ１より共有演算器１１の入力ポートにデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿１は「ｂ０」（本例ではｈｉｇｈ信号）、ｄａｔａＩｎＡ＿１はデータ（例えば１６ビット幅のデータを「@」）が転送される。また、同様にｖａｌｉｄＩｎＢ＿１もｈｉｇｈ、ｄａｔａＩｎＢ＿１にもデータが転送される。ｖａｌｉｄ信号の受付およびＩＤ生成部ではゲート信号とＩＤが生成される。ＩＤはｐｏｒｔ１側を選択「１」し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ｂ０」を選択する。そして、「@」に基づき演算器ステージ０の演算が実行される。
【００５７】
そして演算ステージ０の演算処理結果を、ｖａｌｉｄＩｎＡ＿１の信号である「ｂ０」が、中間データ０用の内部ｖａｌｉｄのＦＦ（１）の入力となる。また中間データ０のＦＦ（２）の入力にはｄａｔａＩｎＡ＿１のデータである「＃」が演算処理された結果「＃１」が与えられる。また、中間データ０に同期したＩＤのＦＦ（３）への入力として「１」が与えられる。
【００５８】
また、演算器ステージ２の演算が実行され、ｄａｔａＯｕｔ＿０およびｄａｔａＯｕｔ＿１用のＦＦ（７）の入力として、「＃２」に基づく演算器ステージ２の演算結果である「＃３」が与えられる。
【００５９】
また、マルチポート出力制御は、ＩＤデコーダによりＩＤをデコードし、一定の規則に基づいて符号化されたデータに復号し、どの出力ポートを有効にするか決める。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）への入力として「（ａ０）」が与えられ、ｐｏｒｔ０出力が有効となる。ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）への入力はＬｏｗのままになる。
【００６０】
これらのＦＦ（１）、（２）、（３）、（７）、（８）、（９）は上記の与えられた入力をＣＬＫ４の立上りエッジで取り込み、確定し、ＣＬＫ４サイクル期間出力する。

ＣＬＫ４の期間では、中間データ１用の内部ｖａｌｉｄのＦＦ（４）への入力にはｖａｌｉｄＩｎＡ＿１「ｂ０」が与えられる。また中間データ１のＦＦ（５）への入力には、ｄａｔａＩｎＡ＿１から入力された「@」を演算器ステージ０で演算処理した結果「@１」が与えられる。また中間データ１に同期したＩＤのＦＦ（６）への入力には「１」が与えられる。これらのＦＦ（４）、（５）、（６）は上記の与えられた入力をＣＬＫ５の立上りエッジで取り込み、確定し、ＣＬＫ５サイクル期間出力する。
【００６１】
また、出力ポートには、結果ｖａｌｉｄＯｕｔ＿０が有効なｐｏｒｔ０出力からデータ「＃３」が出力されクラスタ１に転送される。

ＣＬＫ５の期間では、再びクラスタ１より共有演算器１１の入力ポートにデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿０は「ａ１」（ｈｉｇｈ信号）、ｄａｔａＩｎＡ＿０はデータ（例えば１６ビット幅のデータを「$」）が転送される。また、同様にｖａｌｉｄＩｎＢ＿０もｈｉｇｈ、ｄａｔａＩｎＢ＿０にもデータが転送される。ｖａｌｉｄ信号の受付およびＩＤ生成部ではゲート信号とＩＤが生成される。ＩＤはｐｏｒｔ１

側「０」を選択し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ａ１」を選択する。

【００６２】

そしてｖａｌｉｄＩｎＡ＿０の信号である「ａ１」が、中間データ０用の内部ｖａｌｉｄのＦＦ（１）への入力として与えられる。また中間データ０のＦＦ（２）への入力としてｄａｔａＩｎＡ＿０のデータである「＄」が演算処理された結果「＄１」が与えられる。また、中間データ０に同期したＩＤのＦＦ（３）への入力として「０」が与えられる。

【００６３】

このとき、演算器ステージ２の演算が実行され、ｄａｔａＯｕｔ＿０およびｄａｔａＯｕｔ＿１用のＦＦ（７）への入力として、「@２」に基づいて演算器ステージ２の演算結果である「@３」が与えられる。

【００６４】

また、マルチポート出力制御は、ＩＤデコーダによりＩＤをデコードし、一定の規則に基づいて符号化されたデータに復号し、どの出力ポートを有効にするか決める。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）の入力はＬｏｗのままになる。ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）の入力に「（ｂ０）」が与えられ、Ｐｏｒｔ１出力が有効になる。

【００６５】

これらのＦＦ（１）、（２）、（３）、（７）、（８）、（９）は上記の与えられた入力をＣＬＫ６の立上りエッジで取り込み、確定し、ＣＬＫ６サイクル期間出力する。

ＣＬＫ６の期間で、さらにクラスタ１より共有演算器１１の入力ポートにデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿０は「ａ２」（本例ではｈｉｇｈ信号）、ｄａｔａＩｎＡ＿０はデータ（例えば１６ビット幅のデータを「！」）が転送される。また、同様にｖａｌｉｄＩｎＢ＿０もｈｉｇｈ、ｄａｔａＩｎＢ＿０にもデータが転送される。ｖａｌｉｄ信号の受付およびＩＤ生成部ではゲート信号とＩＤが生成される。ＩＤはｐｏｒｔ１側「０」を選択し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ａ２」を選択する。

【００６６】

そしてｖａｌｉｄＩｎＡ＿０の信号である「ａ２」が、中間データ０用の内部ｖａｌｉｄのＦＦ（１）の入力として与えられる。また中間データ０のＦＦ（２）への入力として、ｄａｔａＩｎＡ＿０のデータである「！」の演算処理された結果「！１」が与えられる。また、中間データ０に同期したＩＤのＦＦ（３）の入力には「０」が与えられる。

【００６７】

このとき、演算器ステージ１により「＄１」を演算処理し、演算結果「＄２」を取得する。上記中間データ１に関する各ＦＦにｖａｌｉｄ信号「ａ１」、ＩＤ、「０」演算器ステージ１の演算結果を、中間データ１に関するＦＦに転送する。中間データ１用の内部ｖａｌｉｄのＦＦ（４）への入力として「ａ１」が与えれられる。また、中間データ１のＦＦ（５）への入力として「＄１」が与えられる。中間データ１に同期したＩＤのＦＦ（６）への入力には「０」が与えられる。

【００６８】

また、出力ポートには、ｖａｌｉｄＯｕｔ＿１が有効なｐｏｒｔ１出力からデータ「@３」が出力されクラスタ１に転送される。

これらのＦＦ（１）、（２）、（３）、（４）、（５）、（６）は上記の与えられた入力をＣＬＫ７の立上りエッジで取り込み、確定し、ＣＬＫ７サイクル期間出力する。

ＣＬＫ７の期間で、クラスタ１より共有演算器１１の入力ポートにデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿１は「ｂ１」（本例ではｈｉｇｈ）、ｄａｔａＩｎＡ＿０はデータ（例えば１６ビット幅のデータを「%」）が転送される。また、同様にｖａｌｉｄＩｎＢ＿１もｈｉｇｈ、ｄａｔａＩｎＢ＿１にもデータが転送される。ｖａｌｉｄ信号の受付およびＩＤ生成部ではゲート信号とＩＤが生成される。ＩＤはｐｏｒｔ１側

「１」を選択し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ｂ１」を選択する。
【００６９】
このとき、「％」に基づき演算器ステージ０の演算結果「％１」を取得する。上記中間データ０に関する各ＦＦへの入力にはｖａｌｉｄ信号「ｂ１」、ＩＤ「１」、演算器ステージ０の演算結果「％１」が与えられる。中間データ０用の内部ｖａｌｉｄのＦＦ（１）への入力には「ｂ１」が与えられる。また中間データ０のＦＦ（２）の入力には「％１」が与えられる。中間データ０にＩＤのＦＦ（３）の入力には「０」が与えられる。
【００７０】
さらに、演算器ステージ１により「！１」を演算処理し、演算結果「！２」を取得する。上記中間データ１に関する各ＦＦへの入力としてｖａｌｉｄ信号「ａ２」、ＩＤ「０」、演算器ステージ１による演算結果「！２」が与えられる。中間データ１用の内部ｖａｌｉｄのＦＦ（４）への入力として「ａ２」が与えられる。また中間データ１のＦＦ（５）への入力として「！２」が与えられる。中間データ１に同期したＩＤのＦＦ（６）への入力として「０」が与えられる。
【００７１】
また、演算器ステージ２により「＄２」を演算処理し、演算結果「＄３」を取得する。上記中間データ２に関する各ＦＦへの入力としてｖａｌｉｄ信号「ａ１」、ＩＤ「０」、演算器ステージ２による演算結果「＄３」を保持する。
【００７２】
ｄａｔａＯｕｔ＿０およびｄａｔａＯｕｔ＿１用のＦＦ（７）への入力として、「＄２」の演算結果である「＄３」が与えられる。
マルチポート出力制御は、ＩＤデコーダによりＩＤ「ａ１」をデコードし「(ａ１)」を算出し、どの出力ポートを有効にするか選択する。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）への入力には「(ａ１)」が与えられ、ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）への入力はＬｏｗのままになり、ｖａｌｉｄＯｕｔ＿０が有効になる。
【００７３】
これらのＦＦ（１）、（２）、（３）、（４）、（５）、（６）、（７）、（８）、（９）は上記の与えられた入力をＣＬＫ８の立上りエッジで取り込み、確定し、ＣＬＫ８サイクル期間出力する。

ＣＬＫ８の期間では、演算器ステージ１により「％１」を演算処理し、演算結果「％２」を取得する。上記中間データ１に関する各ＦＦへの入力としてｖａｌｉｄ信号「ｂ１」、ＩＤ「１」、演算器ステージ１の演算結果「％２」が与えられる。中間データ１用の内部ｖａｌｉｄのＦＦ（４）への入力として「ｂ１」が与えられる。また中間データ１のＦＦ（５）への入力として演算器ステージ１の演算結果「％２」が与えられる。中間データ１に同期したＩＤのＦＦ（６）への入力として「１」が与えられる。
【００７４】
さらに、演算器ステージ２により「！２」を演算処理し、演算結果「！３」を取得する。上記中間データ２に関する各ＦＦへの入力としてｖａｌｉｄ信号「ａ２」、演算器ステージ２による演算結果「！３」が与えられる。ｄａｔａＯｕｔ＿０およびｄａｔａＯｕｔ＿１用のＦＦ（７）への入力としてに、「！２」の演算結果である「！３」が保持される。
【００７５】
マルチポート出力制御は、ＩＤデコーダによりＩＤ「ａ２」をデコードし「（ａ１）」を算出し、どの出力ポートを有効にするか選択する。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）への入力には「(ａ２)」が与えられ、ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）への入力はＬｏｗのままになり、ｖａｌｉｄＯｕｔ＿０が有効になる。
【００７６】
また、出力ポートには、ｖａｌｉｄＯｕｔ＿０が有効なｐｏｒｔ０出力からデータ「＄３」が出力されクラスタ１に転送される。

これらのＦＦ（４）、（５）、（６）、（７）、（８）、（９）は上記の与えられた入力をＣＬＫ９の立上りエッジで取り込み、確定し、ＣＬＫ９サイクル期間出力する。

ＣＬＫ９の期間では、演算器ステージ２により「％２」を演算処理し、演算結果「％３」を取得する。上記中間データ２に関する各ＦＦ（結果出力用のＦＦ）への入力として「ｂ１」、「％３」、が与えられる。

【００７７】

マルチポート出力制御は、ＩＤデコーダによりＩＤ「ｂ１」をデコードし「（ｂ１）」を算出し、どの出力ポートを有効にするか選択する。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）への入力はＬｏｗのままになり、ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）への入力は「（ｂ１）」が与えられる、ｖａｌｉｄＯｕｔ＿１が有効になる。

【００７８】

また、出力ポートには、ｖａｌｉｄＯｕｔ＿０が有効なｐｏｒｔ０出力からデータ「！３」が出力されクラスタ１に転送される。

これらのＦＦ（７）、（８）、（９）は上記の与えられた入力をＣＬＫ１０の立上りエッジで取り込み、確定し、ＣＬＫ１０サイクル期間出力する。

ＣＬＫ１０の期間では、出力ポートには、ｖａｌｉｄＯｕｔ＿１が有効なｐｏｒｔ１出力からデータ「％３」が出力されクラスタ１に転送される。

【００７９】

また、ＣＬＫ１０の期間では、同じＣＬＫ期間に入力ポートから入力データが入力されたときの例を説明する。

クラスタ１より共有演算器１１の入力ポートにデータとｖａｌｉｄ信号が入力され、ｖａｌｉｄＩｎＡ＿０はａ３（本例ではｈｉｇｈ）、ｄａｔａＩｎＡ＿０はデータ（例えば１６ビット幅のデータを＆）が転送される。

【００８０】

さらに、ｖａｌｉｄＩｎＡ＿１はｂ２（判例ではｈｉｇｈ）、ｄａｔａＩｎＡ＿１はデータ（例えば１６ビット幅のデータを＊）が転送される。

この場合、上記説明したｖａｌｉｄ信号の受付およびＩＤ生成部のソースコードに沿ってゲート信号が生成されるので、ｐｏｒｔ０入力側が優先される。ＩＤはｐｏｒｔ０側「０」を選択し、内部ｖａｌｉｄ（マルチポート入力制御直後）は「ａ３」が選択される。その後は上記説明してきた動作と同様各演算器ステージ０～２の演算処理を実行する。

【００８１】

そして、マルチポート出力制御は、ＩＤデコーダによりＩＤ「ａ３」をデコードし「（ａ３）」を算出し、どの出力ポートを有効にするか選択する。ｖａｌｉｄＯｕｔ＿０用のＦＦ（８）に「（ａ３）」が保存され、ｖａｌｉｄＯｕｔ＿１が有効になる。ｖａｌｉｄＯｕｔ＿１用のＦＦ（９）はＬｏｗのままになる。

【００８２】

その後ＣＬＫ１３で、ｖａｌｉｄＯｕｔ＿０により有効な出力ポートであるｐｏｒｔ０出力から、データ「＆３」が出力されクラスタ１に転送される。

図９は共有演算器１１の入力ポートと出力ポートが共に３ある場合の例である。入力ポート数が増えた場合でも、図７で説明した２ポートと同様に、ｖａｌｉｄ受付およびＩＤ生成部で、優先順位を持たせ、各入力ポートにＩＤを与えることで共有演算器１１を実現することが可能である。ここで、ｖａｌｉｄ信号の受付およびＩＤ生成部は、同図のソースにあるように演算を実行する。もし、ｖａｌｉｄＩｎＡ＿０とｖａｌｉｄＩｎＢ＿０であればゲート信号によりｄａｔａＩｎＡ＿０とｄａｔａＩｎＢ＿０を演算器に取り入れるか／取り入れないかを選択する。ゲート信号が「１」であれば、入力データを演算器に取り入れる。それと同時に内部ｖａｌｉｄをＯＮにし、ＩＤを「０」とする。

【００８３】

また、もしｖａｌｉｄＩｎＡ＿１とｖａｌｉｄＩｎＢ＿１であればゲート信号によりｄａｔａＩｎＡ＿１とｄａｔａＩｎＢ＿１を演算器に取り入れるか／取り入れないかを選択する。ゲート信号が「１」であれば入力データを演算器に取り入れる。それと同時に内部ｖａｌｉｄをＯＮにし、ＩＤを「１」とする。

【００８４】

また、もしｖａｌｉｄＩｎＡ＿２とｖａｌｉｄＩｎＢ＿２であればゲート信号によりｄａｔａＩｎＡ＿２とｄａｔａＩｎＢ＿２を演算器に取り入れるか／取り入れないかを選択する。ゲート信号が「１」であれば入力データを演算器に取り入れる。それと同時に内部ｖａｌｉｄをＯＮにし、ＩＤを「２」とする。その後演算を行いマルチポート出力制御でＩＤをデコードし出力先を選択し、選択されたポートから出力データを出力する。

【００８５】

当然であるが、演算器ステージを複数段設けてもよい。

（共有演算器の演算器に除算部を使用した例）

図１０は、共有演算部に使用する除算部の例である。除算をする場合は除数と被除数が必要である。そこでマルチポート入力制御から転送されるデータとｖａｌｉｄ信号を、除数のときはＤａｔａＢとｖａｌｉｄＢ信号、被除数のときはＤａｔａＡとｖａｌｉｄＡ信号を設ける。

【００８６】

同図の制御部１０１でＤａｔａＡおよびＤａｔａＢの２の補数をとり、除算パイプライン１０２（演算器ステージ）を複数段（本例Ｍ段）演算し、商と剰余を求め、出力Ｄａｔａ（剰余）と出力Ｄａｔａ（商）をマルチポート出力制御に転送する。それと同時にｖａｌｉｄ信号を設定し出力ｖａｌｉｄとしてマルチポート出力制御に転送する。

【００８７】

ここで、ｃｌｏｃｋ、ｒｅｓｅｔは全てのＦＦに供給される。また、除算器のＳｉｇｎ選択機能や、ＳＴＡＬＬ機能、強制停止機能、Ｅｒｒｏｒ検出機能を設けてもよい。また、除算器は開平器などで使用することが考えられるため、同図に示す出力除数の信号を設けておいてもよい。

（共有演算器の演算器にＰｏｌａｒ演算器を使用した例）

図１１は共有演算部に使用するＰｏｌａｒ演算器の例で、ｐｏｌａｒ関数を用い複素数を作成するための構成例である。マルチポート入力制御から転送されるデータとｖａｌｉｄ信号を入力する。入力データ０および入力データ１は角度（Ｘ軸から半径ベクトルへの角度をラジアンで表わす：－２π～２π）を入力してデータｖａｌｉｄ０、１信号とともにマルチポート入力制御１１１から入力する。なお、入力データ１は一定量（例えば半径ベクトルの長さでデータ空間単位など）とデータｖａｌｉｄ１信号を設けてもよい。

【００８８】

図１２に示すフローのステップＳ１のように、入力データとｖａｌｉｄ信号を受け付ける。例えば、入力データ０（角度：例えば－２π～２π１６ビット幅のＱ１２フォーマット）とデータｖａｌｉｄ０信号を受け付ける。

【００８９】

次に演算器ステージは、図１２のステップＳ２では象限判定ブロック１１３で象限判定をする。象限判定は１～４象限のどの正眼のデータであるかの判定をする。また、第１象限化＆対象まるめブロック１１２で該データを０～２／πの値に変換した後、下位２ビットのまるめ処理などをする。（例えば１６ビットＱ１２フォーマットを１１ビットＱ１０フォーマットに変換する）

ステップＳ３では、Ｓｉｎ＿ＲＯＭ１１４（角度→ｓｉｎ変換用テーブルなど）およびＣｏｓ＿ＲＯＭ１１５（角度→ｃｏｓ変換用テーブルなど）にあるデータを同時に読み出す。（例ではＲＯＭ内のデータは１３ビットＱ１２フォーマットにしている）

ステップＳ４では、元の象限に変換ブロック１１６、１１７により、－２π～２πの値

に変換する。つまり上記の各ＲＯＭ１１４、１１５から読み出した値を元の象限のデータに変換をする。

【００９０】

ステップ５では、マルチポート出力制御１１８、１１９により、データ入力のあったポートに対してｖａｌｉｄを付加してデータを出力する。出力はＳｉｎ計算結果としてＳｉｎデータ（１６ビットＱ１２フォーマット）、ｓｉｎ＿ｖａｌｉｄ０、ｓｉｎ＿ｖａｌｉｄ１信号（出力先ポートの指定）を出力し、Ｃｏｓ計算結果としてＣｏｓデータ（１６ビットＱ１２フォーマット）、ｃｏｓ＿ｖａｌｉｄ０、ｃｏｓ＿ｖａｌｉｄ１信号（出力先ポートの指定）を出力する。

（共有演算器の演算器にアークタンジェント器を使用した例）

図１３はアークタンジェントの演算についての構成例である。マルチポート入力制御１３１に入力データとして虚数部／実数部を入力する。入力データ０（例えば１３ビットＱ１２フォーマット）とｖａｌｉｄ０信号、入力データ１（例えば１３ビットＱ１２フォーマット）とｖａｌｉｄ１信号を受け取るようにバスを設置する。アークタンジェント演算（演算ステージ）は、例えば入力データを四捨五入ブロック１３２で四捨五入をし、ＲＯＭ１２３内の－２π～２πのアークタンジェント計算した値を保持したテーブルから、入力データに対応した値を選択する。その後、ブロック１３５においてπ／４を加えて演算結果を算出しマルチポート出力制御１３５に出力する。マルチポート出力制御１３５は出力データと、ｖａｌｉｄ０信号またはｖａｌｉｄ１信号を選択した結果をクラスタ１に転送する。

上記説明したように構成することで、共有演算器は、ｖａｌｉｄ信号により起動することで、利用する側のクラスタから特殊な制御信号などによる制御なしに起動できる。また、共有演算器は、クラスタ間を結ぶネットワーク上に接続することにより、任意のクラスタからデータとｖａｌｉｄ信号を送るだけで、ｖａｌｉｄ信号により起動し、同時に受け取ったデータを処理することができる。このため利用するクラスタから特別な制御信号を送ることなしに利用できる。

【００９１】

また、アプリケーション特化エンジンを上記説明したルールで構成することにより、異なるアプリケーション毎に、特殊な信号を設けることが不要になる。そのため、エンジンを交換することが容易になる。さらに各クラスタからは制御信号なしに、共有演算器を共有でき、上記例のようにパイプライン構成にすれば、各ステージで異なるポートからのデータ処理を並列で実行することができる。

【００９２】

なお、当然であるが共有演算器とクラスタからなるブロックを、再構成可能演算処理装置内に複数構築することができる。

（実施例２）

図１４は共有演算器の複数機能化について示した図である。クラスタ１４１から共有演算器１４２に対しデータ入力０とｖａｌｉｄ入力０、データ入力１とｖａｌｉｄ入力１、データ入力２とｖａｌｉｄ入力２をマルチポート入力制御に入力する。

【００９３】

そしてデータ入力０とｖａｌｉｄ入力０、データ入力１とｖａｌｉｄ入力１に入力されたデータに基づいて、演算処理を実行し演算処理結果出力する。

このとき、演算処理部１４３（演算ステージ）は上記説明したようなアプリケーション特化エンジン（除算、ｐｏｌａｒ演算、アークタンジェント演算など）のような単独演算をする場合と、開平器のように除算など単独演算を行う処理部を含んだ処理とが考えられる。このような場合にデコード部１４４を用意し、データ入力２とｖａｌｉｄ入力２を入力することで除算器と開平器の演算の切替えを行うようにする。

【００９４】
デコード部１４４には図１５に示すような演算処理コード表を用意し、演算処理コードと処理内容を対応させる。演算処理コードが０００であればＮｏ＿Ｏｐｅｒａｔｉｏｎとして何もしない設定とし、００１であればＲｅｓｅｒｖｅｄとする。０１０であれば符号無し除算、０１１であれば符号付き除算、１００であれば開平により平方根を求めるようにする。このように予め用意した演算処理コードをデータ入力２より入力する。
【００９５】
そして、演算処理コードに対応する演算処理選択信号を選択し演算処理部１４３に転送する。演算処理選択信号は共有演算器１４２のアプリケーションの構成を切替えて制御を行う信号である。この演算処理選択信号を受けた演算処理部１４３が、演算処理コード１００を受信すれば共有演算器１４２は開平器となり平方根を演算する演算処理部１４３となる。
【００９６】
ここで、例えば除算器から開平器への切替えは、演算処理選択信号を受信して、その演算処理選択信号の内容に基づいて、回路構成を変更する。そのためには、セレクタ（アプリケーションを切替えることができる構成であればよい）などを用意して演算処理選択信号の内容を共有演算器１４２の構成に反映させる。
【００９７】
なお、デコード部１４４にｖａｌｉｄ信号（例ではｖａｌｉｄ入力２）を入力する際に、有効であること特に指定しなくてもよいし、デコード部１４４にｖａｌｉｄ信号の受け入れポートを設けない構成としてもよい。

（共有演算器の演算器に開平器を使用した例）
図１６に開平器の構成を示す。上記説明したようにマルチポート入力制御より入力データ（被開平数：平方根を求めたい数値）としてＤａｔａＡと、ｖａｌｉｄ信号としてｖａｌｉｄＡを入力する。さらにこのとき図示はしないがデコード部に開平演算処理をする通知をし、演算処理選択信号を生成し開平のアプリケーションに切替える。同図の例では平方根を近似方式により求める例を示している。
【００９８】
近似Ｔａｂｌｅ（ＲＯＭ：近似値格納）１６１内の平方根を求めるための除数データの中から、対応する除数データを選択し除数とする。除算器は入力データＤａｔａＡを被除数として、上記説明した除算器１６２（図１０）と同様の演算をして商と除数（２の補数とする）を算出する。その後ＡＬＵ１６３において商と除数を加え出力Ｄａｔａとともに出力ｖａｌｉｄをマルチポート出力制御に出力する。
【００９９】
このように共有演算器（本例開平器）内に単独で実行できる演算器（本例除算器）を含む場合にデコード部を設けることで共有演算器を小型化することができる。なお、共有演算器にデコード部を接続するには、クラスタを介さずに直接接続する構成としてもよい。

（実施例３）
図１７～１９に共有演算器のクラスタとの接続方法について説明する。図１７はクラスタ１７１のクロスバスイッチに共有演算器１７２とを直接接続する例である。クラスタ１７１間はクロスバスイッチにより接続される。共有演算器１７２の入力とクラスタ１７１はマルチポート入力制御の入力ポートに直接接続する。また、共有演算器１７２の出力とクラスタ１７１はマルチポート出力制御の出力ポートに直接接続する。この接続はコンフィギュレーション情報により切替える必要がない場合に有効である。
【０１００】
図１９は共有演算器１９２にセレクタを設けた例である。セレクタをコンフィギュレーション情報に基づいて変更することで接続方法を選択できるため、クラスタ１９１からのデータとｖａｌｉｄ信号の接続経路を動的に選択することが可能になる。

【０１０１】
また、図１８は共有演算器１８２にクロスバスイッチを設けた例である。クラスタ１８１からのデータとｖａｌｉｄ信号を接続する際の経路を動的に選択することが可能である。
【０１０２】
このように、データ伝送時には必要な接点を選んで開閉することで、接続中はデータの送信側と受信側とを一対一で直結させることができるため、データの衝突や混線などの問題が起こりにくくなる。
また、本発明は、上記実施の形態に限定されるものでなく、本発明の要旨を逸脱しない範囲内で種々の改良、変更が可能である。
【０１０３】
（付記１）　コンフィギュレーション情報に基づき、再構成される少なくとも１以上のクラスタから構成される再構成可能演算処理装置において、
前記１以上のクラスタ間で共有利用される共有演算器を前記クラスタの外部に具備し、
前記共有演算器は、
前記クラスタから入力データと入力ｖａｌｉｄ信号を受け付ける入力手段と、
前記入力手段で前記入力ｖａｌｉｄ信号を受け取ると、前記ｖａｌｉｄ信号とともに受け取った前記入力データを演算処理する演算手段と、
前記演算手段の演算処理結果である出力データと、該出力データの出力先の前記クラスタを通知する出力ｖａｌｉｄ信号とを前記クラスタに出力する出力手段と、
を具備することを特徴とする再構成可能演算処理装置。
（付記２）前記共有演算器は、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤを生成し、前記出力データの出力先の前記クラスタを通知することを特徴とする付記１に記載の再構成可能演算処理装置。
（付記３）前記共有演算器は、パイプライン構成を利用して演算処理をすることを特徴とする付記１に記載の再構成可能演算処理装置。
（付記４）前記パイプラインは、複数の前記クラスタより前記入力データと前記入力ｖａｌｉｄ信号を受け付けたとき、識別信号であるＩＤとともに内部ｖａｌｉｄを生成し、前記入力データを演算処理した処理データとともに前記ＩＤを、前記内部ｖａｌｉｄを利用して前記パイプラインで転送することを特徴とする付記３に記載の再構成可能演算処理装置。
（付記５）前記共有演算器の構成は、単独演算を行うアプリケーション特化エンジンであることを特徴とする付記１に記載の再構成可能演算処理装置。
（付記６）前記共有演算器の構成は、単独演算を行う複数のアプリケーション特化エンジンを配設した構成であることを特徴とする付記１に記載の再構成可能演算処理装置。
（付記７）前記共有演算器は、前記アプリケーション特化エンジンの切替えを前記クラスタからの前記入力データと前記入力ｖａｌｉｄ信号に基づいて演算処理選択信号を生成し、前記演算処理選択信号により前記共有演算器のアプリケーション特化エンジンの切替えをすることを特徴とする付記６に記載の再構成可能演算処理装置。
（付記８）前記演算処理選択信号は、前記共有演算器の有するアプリケーション特化エンジンに対応した演算処理コードからなるテーブルを予め設定し、前記クラスタから前記アプリケーション特化エンジンを選択するために、前記入力データである前記演算処理コードを入力し、前記入力データに対応する前記アプリケーション特化エンジンの切替え制御のための信号を選択して、前記演算処理選択信号を生成することを特徴とする付記７記載の再構成可能演算処理装置。
（付記９）前記共有演算器の入力手段と出力手段は、コンフィギュレーションデータに基づき再構成可能なセレクタを配設していることを特徴とする付記１に再構成可能演算処理装置。
（付記１０）前記共有演算器の入力手段と出力手段は、クロスバスイッチを配設していることを特徴とする付記１に再構成可能演算処理装置。

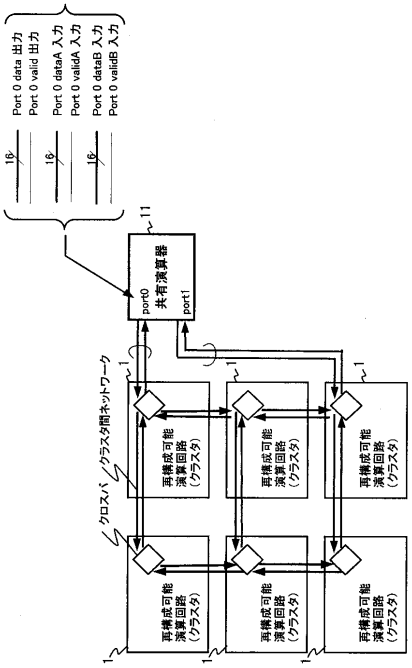
（付記１１）前記共有演算器の入力手段はマルチポートであることを特徴とする付記１に再構成可能演算処理装置。
（付記１２）前記共有演算器の出力手段はマルチポートであることを特徴とする付記１に再構成可能演算処理装置。
【図面の簡単な説明】
【０１０４】
【図１】クラスタ１と共有演算器１１の構成を示した図である。
【図２】クラスタの動作について説明した図である。
【図３】上記図２の動作をタイムチャートで示した図である。
【図４】クラスタＣと共有演算器Ａが接続された場合の例である。
【図５】クラスタＣと共有演算器Ａの動作についてのタイムチャートを示した図である。
【図６】マルチポート構成された共有演算器１１を示した図である。
【図７】共有演算器の基本構成を示す図である。（２ポートの場合）
【図８】図７の動作をタイムチャートで示した図である。
【図９】共有演算器の基本構成を示す図である。（３ポートの場合）
【図１０】共有演算部に使用する除算部の構成例である。
【図１１】共有演算部に使用するＰｏｌａｒ演算器の構成例である、
【図１２】図１１に示すＰｏｌａｒ演算器の動作フローを示す図である。
【図１３】共有演算部に使用するアークタンジェントの演算器の構成例である。
【図１４】共有演算器の複数機能化について示した図である。
【図１５】演算処理コード表である。
【図１６】開平器の構成を示す図である。
【図１７】クラスタのクロスバスイッチに直接接続する例である。
【図１８】共有演算器にクロスバスイッチを設けた構成例を示した図である。
【図１９】共有演算器にセレクタを設けた構成例を示した図である。
【図２０】クラスタの一例を示した図である。
【図２１】従来の再構成可能演算処理装置内のクラスタの構成を概念的に示す略ブロック図である。
【図２２】専用のハードウェアアクセラレータを使用して、ＣＰＵやＤＭＡＣを介在させて演算処理をする方法を示す図である。
【符号の説明】
【０１０５】

| | |
|---|---|
| １ | 再構成可能演算回路（クラスタ） |
| ２ | 演算器群 |
| ３ | コンフィギュレーションメモリ |
| ４ | シーケンサ |
| ５ | データ入力部 |
| ６ | データバッファ部 |
| ７ | データバッファ制御部 |
| ８ | 演算器間ネットワーク |
| ９ | データメモリ |
| １０ | 演算器 |
| １１ | 共有演算器 |
| １０１ | 制御部 |
| １０２ | M段演算ステージ（除算パイプラインの構成） |
| １１１ | マルチポート入力制御 |
| １１２ | 第１象限化＆対象まるめブロック |
| １１３ | 象限判定ブロック |
| １１４ | ＳｉｎＲＯＭ |
| １１５ | ＣｏｓＲＯＭ |

１１６　元の象限に変換ブロック
１１７　元の象限に変換ブロック
１１８　マルチポート出力制御
１１９　マルチポート出力制御
１３１　マルチポート入力制御
１３２　四捨五入ブロック
１３３　ＲＯＭ（アークタンジェント）
１３４　ブロック
１３５　マルチポート出力制御
１４１　クラスタ
１４２　共有演算器
１４３　演算処理部
１４４　デコード部
１６１　近似ｔａｂｌｅ（ＲＯＭ：開平）
１６２　除算器
１６３　ＡＬＵ
１７１　クラスタ
１７２　共有演算器
１８１　クラスタ
１８２　共有演算器
１９１　クラスタ
１９２　共有演算器

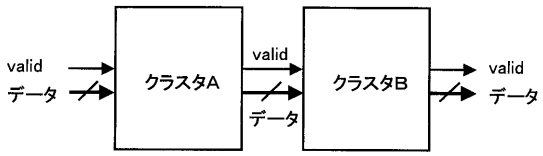
【図１】

クラスタ1と共有演算器11の構成を示した図



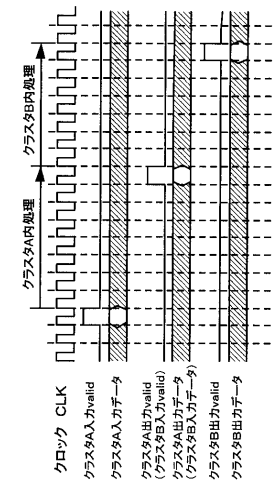

【図２】

クラスタの動作について説明した図



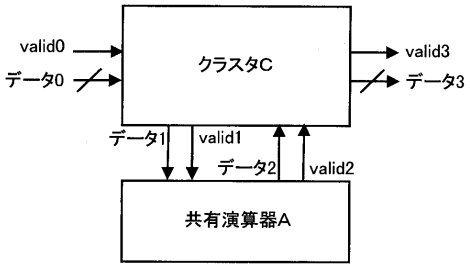
【図３】

上記図2の動作をタイムチャートで示した図



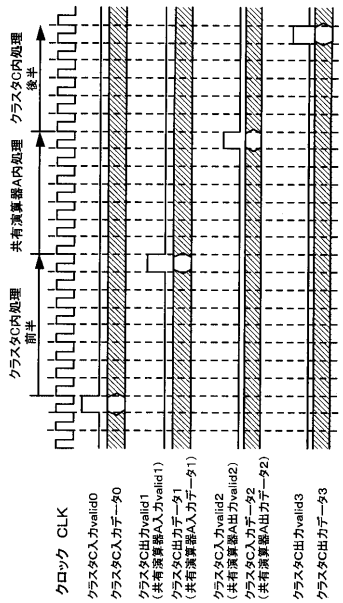

【図４】

クラスタCと共有演算器Aが接続された場合の例



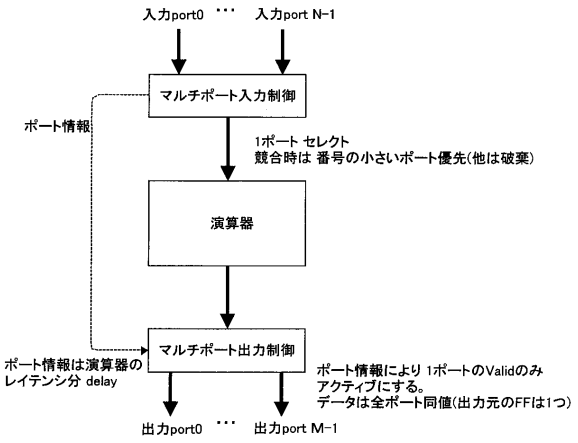

【図５】

クラスタCと共有演算器Aの動作についての
タイムチャートを示した図



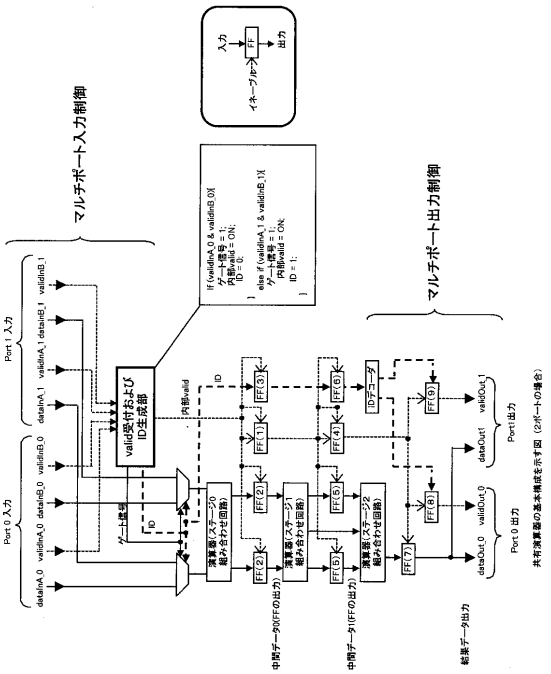

【図６】

マルチポート構成された共有演算器を示した図



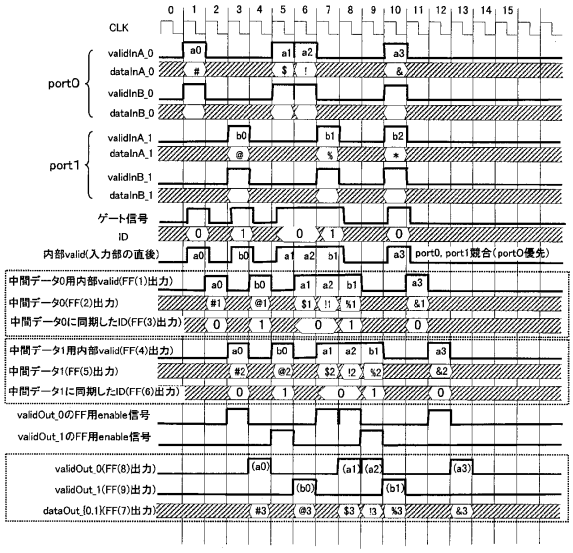
【図７】

共有演算器の基本構成を示す図
（2ポートの場合）



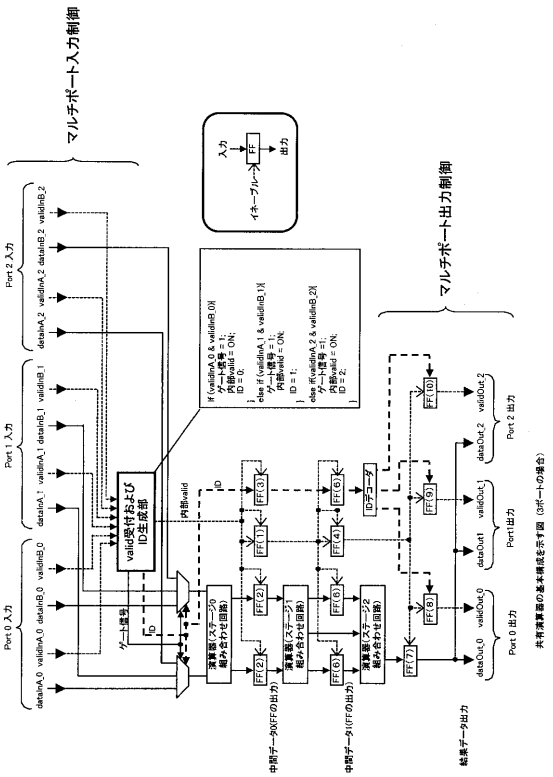

【図８】

図7の動作をタイムチャートで示した図



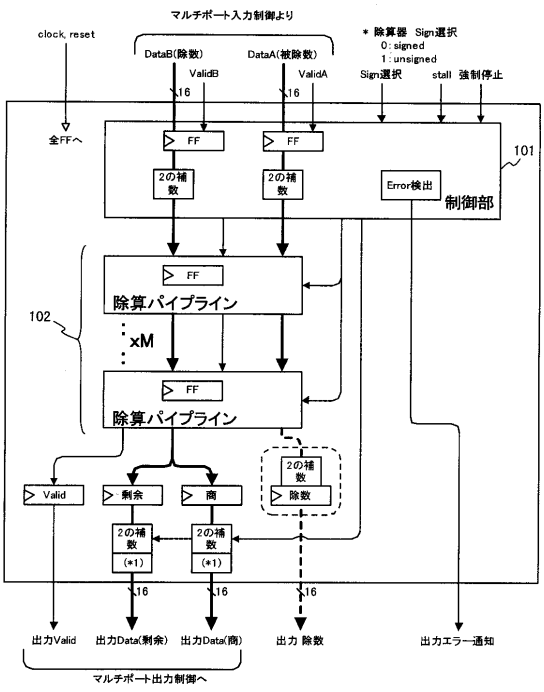

【図９】

共有演算器の基本構成を示す図
（3ポートの場合）



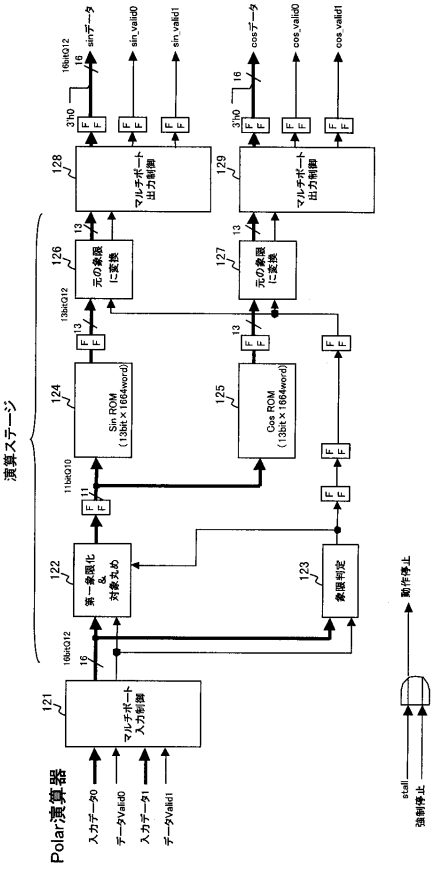

【図１０】

共有演算部に使用する除算部の構成例



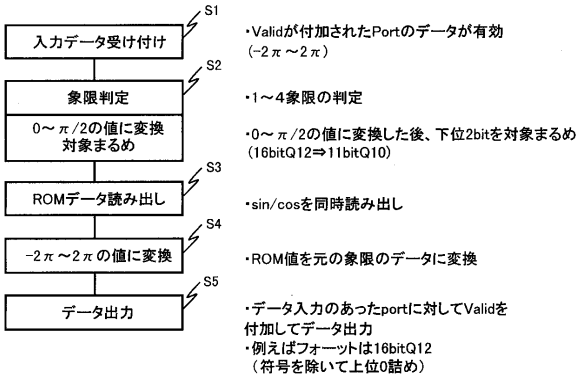
【図１１】

共有演算部に使用するPolar演算器の構成例



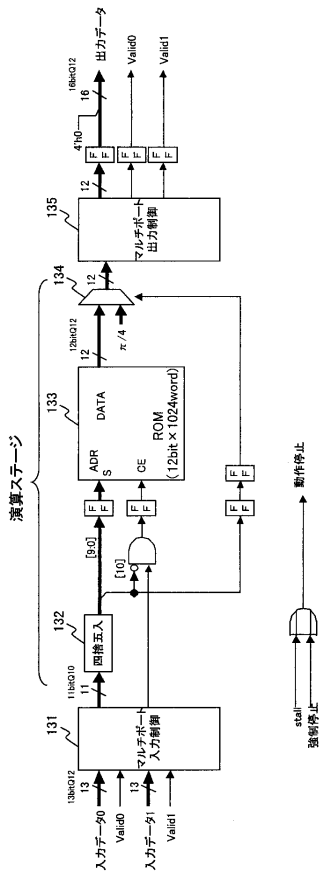

【図１２】

図11に示すPolar演算器の動作フローを示す図



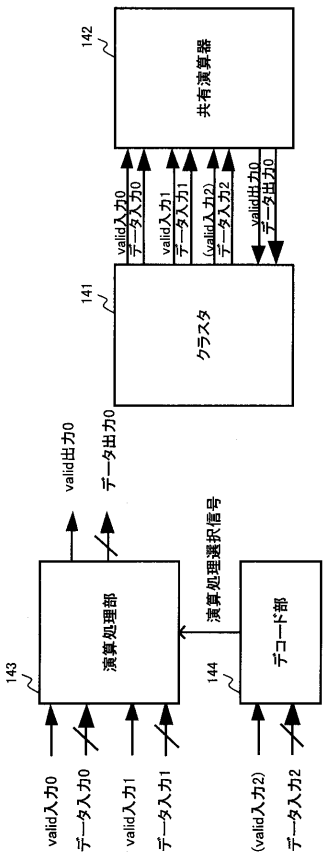

【図１３】

共有演算部に使用するアークタンジェントの演算器の構成例




【図１４】

共有演算器の複数機能化について示した図



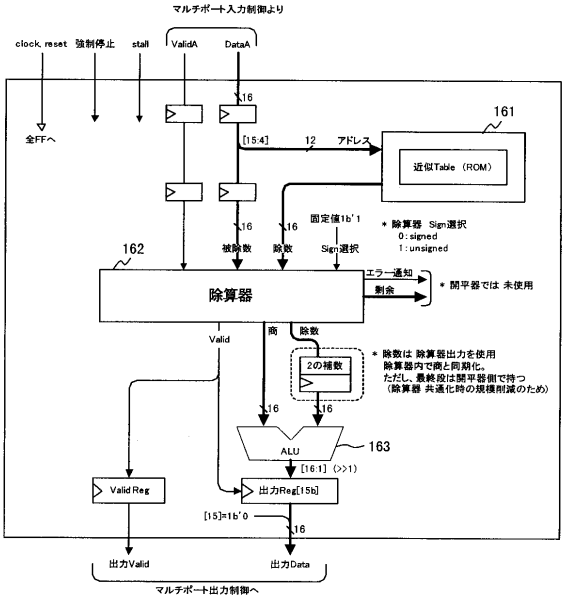
【図１５】

演算処理コード表

| 演算処理コード | 処理内容 |
|---|---|
| 000 | No Operation |
| 001 | Reserved |
| 010 | 符号無し除算 |
| 011 | 符号付除算 |
| 100 | 開平 |

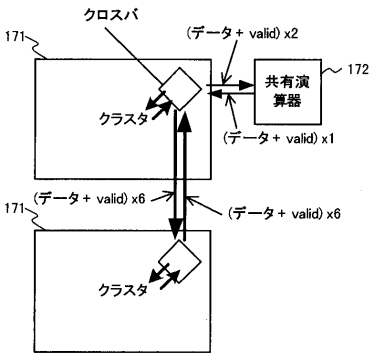
【図１６】

開平器の構成を示す図




【図１７】

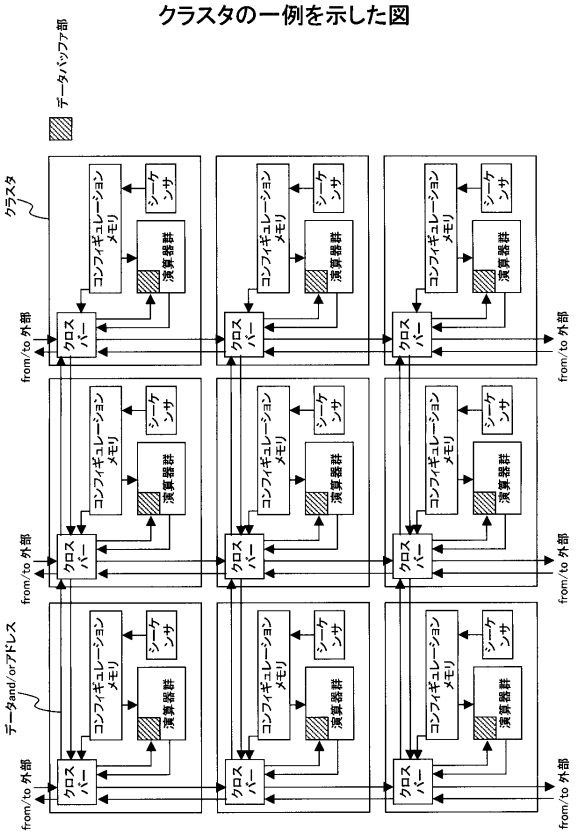
クラスタのクロスバスイッチに直接接続する例




クラスタのクロスバに直接接続する例

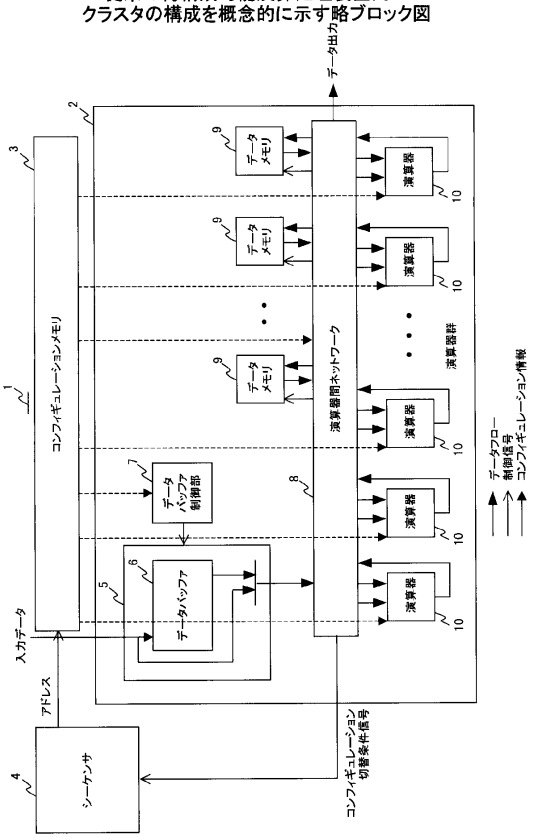
【図２０】

クラスタの一例を示した図



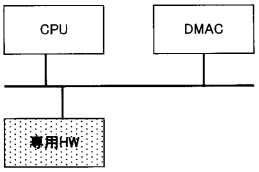
【図２１】

従来の再構成可能演算処理装置内の
クラスタの構成を概念的に示す略ブロック図



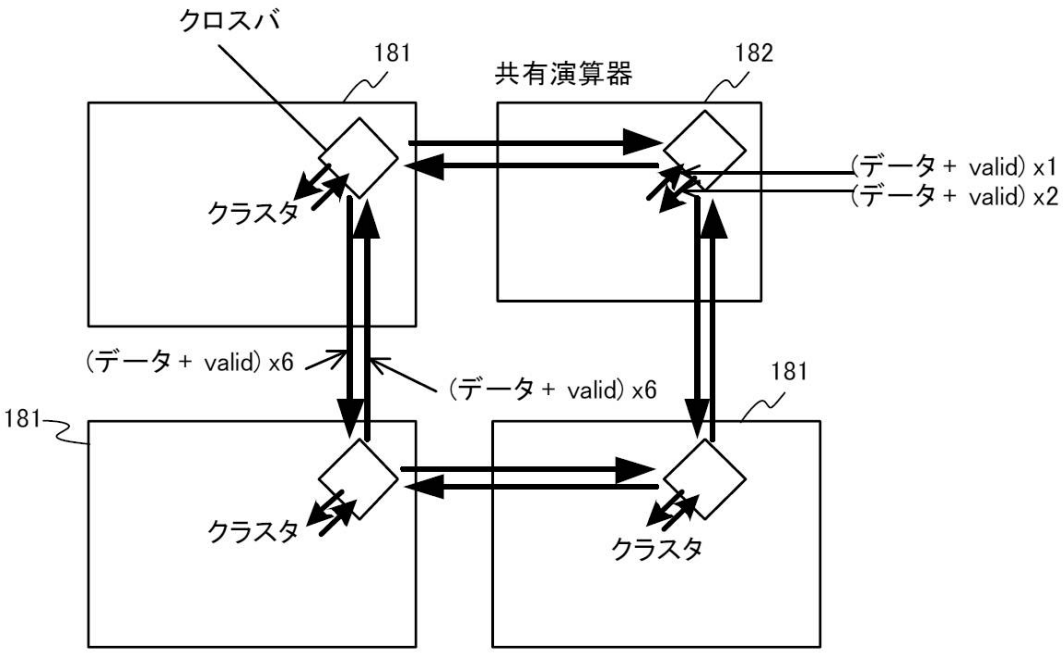

【図２２】

専用のハードウェアアクセラレータを使用して、
CPUやDMACを介在させて演算処理をする方法を示す図



【図１８】

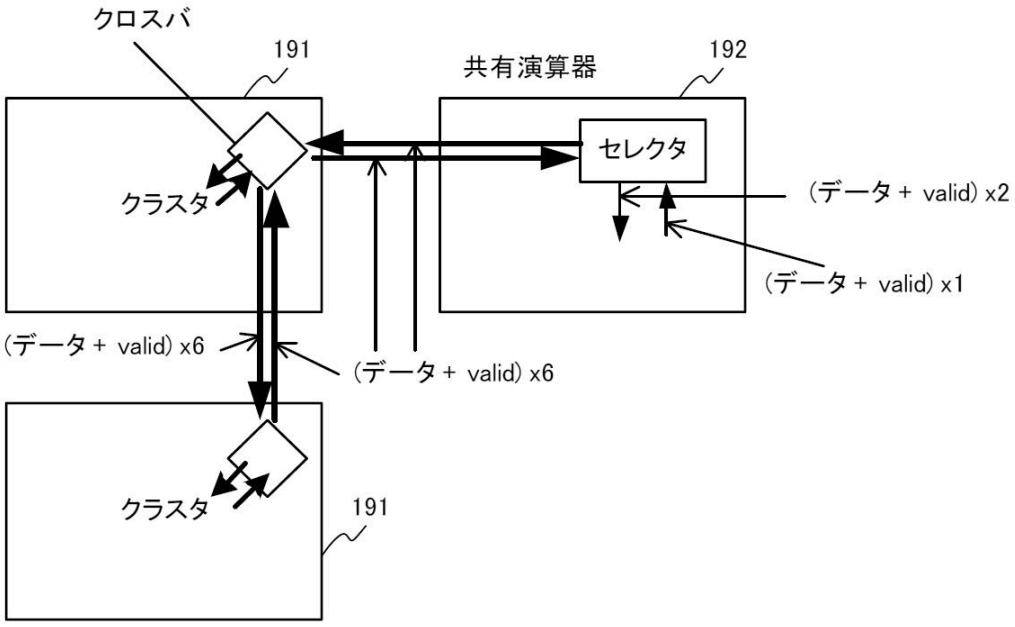
共有演算器にクロスバスイッチを設けた構成例を示した図




共有演算器もクロスバを持つ場合

【図１９】

共有演算器にセレクタを設けた構成例を示した図




共有演算器がセレクタを持っている場合

フロントページの続き

審査官　▲高▼橋　正▲徳▼

(56)参考文献　特開昭５４－０６１８５１（ＪＰ，Ａ）
国際公開第２００４／０２３２９０（WO，Ａ１）
特開平０６－３４８４９２（ＪＰ，Ａ）
特表２００４－５１１０４２（ＪＰ，Ａ）
特開２００２－０７３３３１（ＪＰ，Ａ）
特開昭５９－０１６０７２（ＪＰ，Ａ）
特開平０２－００５１７３（ＪＰ，Ａ）
特開昭６３－２９１１５５（ＪＰ，Ａ）
特開２０００－２０１０６６（ＪＰ，Ａ）
国際公開第０３／０３６５０７（WO，Ａ１）
米国特許第０６７４５３１８（ＵＳ，Ｂ１）
国際公開第２００４／０４２５６０（WO，Ａ１）
特開２００４－１３３７８１（ＪＰ，Ａ）
H. Ito et al, Dynamically reconfigurable logic LSI-PCA-1, 2001 SYMPOSIUM ON VLSI CIRCUITS DIGEST OF TECHNICAL PAPERS, 日本, 2001 SYMPOSIUM ON VLSI CIRCUITS, TOKYO : JSAP, JP, ２００１年　6月１４日, p.103-106, ＵＲＬ, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.6946&rep=rep1&type=pdf
古田浩一郎　外, 柔軟かつ高速な再構成を実現した動的再構成ロジックＬＳＩ, 電子情報通信学会技術研究報告. ＳＤＭ, シリコン材料・デバイス, 日本, 社団法人電子情報通信学会, １９９９年　6月, pp.1-8, ＵＲＬ, http://ci.nii.ac.jp/lognavi?name=nels&lang=jp&type=pdf&id=ART0003740804

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　１５／８０,
Ｇ０６Ｆ　９／３８