(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2007-172569
(P2007-172569A)
(43)公開日　平成19年7月5日(2007.7.5)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 15/80	(2006.01)	G06F 15/80		5B046
G06F 17/50	(2006.01)	G06F 17/50	656A	
		G06F 17/50	654M	

審査請求 未請求　請求項の数 10　ＯＬ　（全 38 頁）

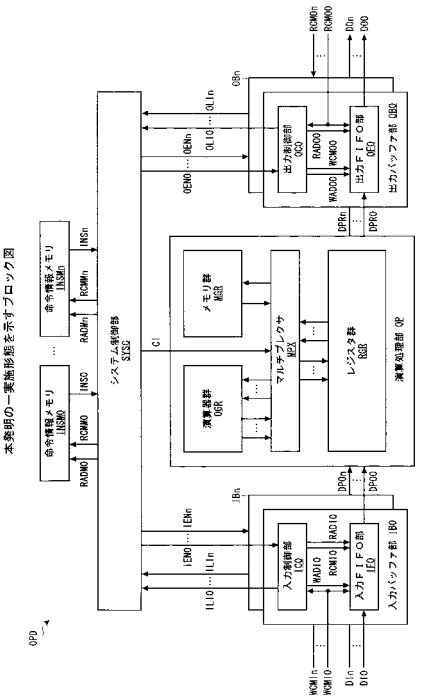
(21)出願番号　特願2006-91025(P2006-91025)
(22)出願日　平成18年3月29日(2006.3.29)
(31)優先権主張番号　特願2005-342158(P2005-342158)
(32)優先日　平成17年11月28日(2005.11.28)
(33)優先権主張国　日本国(JP)

(71)出願人　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
(71)出願人　000237606
富士通デバイス株式会社
東京都品川区西五反田八丁目９番５号
(74)代理人　100072718
弁理士　古谷　史旺
(74)代理人　100116001
弁理士　森　俊秀
(72)発明者　関山　博之
東京都品川区大崎２丁目８番８号　富士通デバイス株式会社内

Ｆターム(参考)　5B046 AA08　BA03

(54)【発明の名称】演算処理装置、演算処理装置の内部構成決定方法および演算処理システム

(57)【要約】
【課題】　演算処理装置のスループットを低下させることなく、演算処理用のリソースの数を最小限に抑える。
【解決手段】　演算処理部は、データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する。制御部は、処理サイクル毎に、複数の命令情報に対して制御処理を所定順序で実施し、制御処理における演算処理部の構成要素の割り当て結果を接続情報として出力する。制御部は、制御処理において、命令情報が示す演算を実行可である場合に、演算に割り当てる演算処理部の構成要素を決定し、演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、出力バッファ部のデータ出力タイミングを満たしたうえで演算を実行すべき処理サイクルを次の処理サイクルに移行させる。
【選択図】　図６



【特許請求の範囲】
【請求項１】
複数のデータ系列に対応して設けられ、対応する外部入力データを格納して処理対象データとして出力する複数の入力バッファ部と、
前記データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する演算処理部と、
前記データ系列に対応して設けられ、対応する処理結果データを格納して外部出力データとして出力する複数の出力バッファ部と、
処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、命令情報が示す演算を実行可である場合に、演算に割り当てる前記演算処理部の構成要素を決定し、前記演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、前記出力バッファ部のデータ出力タイミングを満たしたうえで演算を実行すべき処理サイクルを次の処理サイクルに移行させる制御処理を所定順序で実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する制御部とを備えることを特徴とする演算処理装置。
【請求項２】
請求項１記載の演算処理装置において、
前記各入力バッファ部は、
対応する外部入力データをライト要求に応答して格納し、格納している外部入力データをリード要求に応答して処理対象データとして出力する第１データ蓄積部と、
前記第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を出力する第１優先度情報生成部とを備え、
前記各出力バッファ部は、
対応する処理結果データをライト要求に応答して格納し、格納している処理結果データをリード要求に応答して外部出力データとして出力する第２データ蓄積部と、
前記第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を出力する第２優先度情報生成部とを備え、
前記制御部は、
前記入力バッファ部から供給される複数の第１優先度情報と前記出力バッファ部から供給される複数の第２優先度情報とに応じて前記データ系列の優先順位を決定する優先順位判定部と、
処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、対応するデータ系列の前記優先順位判定部により決定された優先順位が高い順に前記制御処理を実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する構成要素割当部とを備えることを特徴とする演算処理装置。
【請求項３】
請求項２記載の演算処理装置において、
前記第１優先度情報生成部は、
前記第１データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第１データ蓄積部のデータ出力動作に伴ってダウンカウントする第１カウンタと、
前記第１カウンタのカウンタ値を第１閾値と比較して優先度を決定し、決定した優先度を示す第１優先度情報を出力する第１優先度判定部とを備えることを特徴とする演算処理装置。
【請求項４】
請求項２記載の演算処理装置において、
前記第２優先度情報生成部は、
前記第２データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第２データ蓄積部のデータ出力動作に伴ってダウンカウントする第２カウンタと、
前記第２カウンタのカウンタ値を第２閾値と比較して優先度を決定し、決定した優先度

を示す第２優先度情報を出力する第２優先度判定部とを備えることを特徴とする演算処理装置。

【請求項５】

請求項２記載の演算処理装置において、

前記優先順位判定部は、

前記データ系列に対応して設けられ、対応する入力バッファ部から供給される第１優先度情報、対応する出力バッファ部から供給される第２優先度情報、および優先度定義テーブルを参照して優先度を決定し、決定した優先度を示す第３優先度情報を出力する複数の第３優先度判定部と、

前記第３優先度判定部から供給される複数の第３優先度情報に応じて、前記データ系列の優先順位を決定する調停部とを備えることを特徴とする演算処理装置。

【請求項６】

請求項１記載の演算処理装置において、

前記演算処理部は、

構成要素として設けられる複数のレジスタ、複数の演算器および複数のメモリと、

前記接続情報に応じて、前記レジスタ、前記演算器および前記メモリの接続関係を変更する接続関係変更部とを備えることを特徴とする演算処理装置。

【請求項７】

請求項１記載の演算処理装置において、

前記制御部に供給される命令情報は、処理サイクルあたりの演算の数を示す情報を含むとともに、演算毎に、演算の種類を示す情報と演算対象データを識別するための情報とを含むことを特徴とする演算処理装置。

【請求項８】

複数のデータ系列に対応して設けられ、対応する外部入力データを格納して処理対象データとして出力する複数の入力バッファ部と、

前記データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する演算処理部と、

前記データ系列に対応して設けられ、対応する処理結果データを格納して外部出力データとして出力する複数の出力バッファ部と、

処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、命令情報が示す演算を実行可である場合に、演算に割り当てる前記演算処理部の構成要素を決定し、前記演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、演算を実行すべき処理サイクルを次の処理サイクルに移行させる制御処理を所定順序で実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する制御部とを備える演算処理装置の内部構成を決定する方法であって、

前記演算処理部の構成要素の数をパラメータ化した前記演算処理装置の回路記述を用いて、前記演算処理部の構成要素の数を、前記制御処理で常に演算実行可となる値に設定してシミュレーションを実施する第１工程と、

前記第１工程で実施したシミュレーションの結果から、そのシミュレーションで前記制御部により演算に割り当てられた前記演算処理部の構成要素の数を仮設計値として取得する第２工程と、

前記演算処理部の構成要素の数をパラメータ化した前記演算処理装置の回路記述を用いて、前記演算処理部の構成要素の数を、前記仮設計値より小さい値を初期値として順次減少させながら、前記出力バッファ部のデータ出力タイミングが満たされなくなるまでシミュレーションを繰り返し実施する第３工程と、

前記第３工程で実施したシミュレーションのうち、前記出力バッファ部のデータ出力タイミングが満たされた最後のシミュレーションの結果から、そのシミュレーションで前記制御部により演算に割り当てられた前記演算処理部の構成要素の数を実設計値として取得する第４工程とを含むことを特徴とする演算処理装置の内部構成決定方法。

【請求項９】
請求項８記載の演算処理装置の内部構成決定方法において、
前記演算処理部は、構成要素として設けられる複数のレジスタ、複数の演算器および複数のメモリと、前記接続情報に応じて、前記レジスタ、前記演算器および前記メモリの接続関係を変更する接続関係変更部とを備え、
前記第１および第３工程において、前記演算処理装置の構成要素の数として、前記レジスタの数、前記演算器の数および前記メモリの数をパラメータ化した前記演算処理装置の回路記述を用いてシミュレーションを実施することを特徴とする演算処理装置の内部構成決定方法。
【請求項１０】
複数のデータ系列に対応する複数のメイン演算処理部と、
前記メイン演算処理部に共通して設けられ、前記メイン演算処理部に代わって演算処理を実施するサブ演算処理部とを備え、
前記サブ演算処理部は、請求項１～請求項７のいずれかに記載の演算処理装置により構成されることを特徴とする演算処理システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、演算処理装置、演算処理装置の内部構成決定方法および演算処理システムに関する。
【背景技術】
【０００２】
所望のアプリケーションを実現するＬＳＩ（Large Scale Integration）の手段として、ＡＳＩＣ（Application Specific Integrated Circuit）やプロセッサなどが挙げられる。ＡＳＩＣは、ハードワイヤード構成であるため、プログラマビリティが低いという欠点がある。また、ＡＳＩＣでは、アプリケーションがトランジスタへの展開により実現されるため、アプリケーションの処理を開始してから終了するまでの一連の動作におけるリソースの有効活用という点で効率が悪い。
【０００３】
一方、プロセッサでは、リソースを非共有とするマルチプロセッシングのような分散処理型アーキテクチャを採用する場合、アプリケーションを各々のプロセッサエレメント毎にタスク分解するために、分解能力の高いコンパイラが必要となる。しかしながら、そのようなコンパイラは現存せず、人手によりタスク分解を実施した後に、タスク分解の最適化を図るために膨大な時間を費やして微調整する必要がある。
【０００４】
また、プロセッサでは、リソースを非共有とする分散処理型アーキテクチャを採用する場合、リソースの使用率が高くなり難いという欠点がある。プロセッサ間でのリソースの共有がないため、各々のプロセッサにおいて限られたリソースを使い切らなくてはならない。このため、タスク分解およびタスク分解の最適化が難しいわりには、処理性能および回路面積の面で十分な実力があるとは言い難い。
【０００５】
より詳細に説明すると、マルチプロセッサでは、複数のプロセッサコアがネットワークあるいはバスを介して接続されており、プロセッサコア毎に専用の拡張演算処理部あるいはプログラマブル拡張演算処理部が設けられている。各プロセッサコアは、拡張演算処理部やプログラマブル拡張演算処理部のリソースを用いて演算処理を実施する。しかしながら、演算処理用のリソースがプロセッサコア毎に独立して設けられているため、プロセッサコアに割り当てられたタスクによってリソースが最大限に効率良く使用されるとは限らない。多くの場合は、演算処理が短時間で完了して頻繁にアイドル状態となるプロセッサコアがある一方で、常に演算処理を実施し、場合によっては要求される時間までに演算処理が完了しないプロセッサコアがあるなど、リソースが効率良く使用されていない。ソフ

トウェア設計（機能割り当て）の効率を重視するほど、この傾向が強くなるが、逆に、リソースの使用効率を重視すると、タスク分解やコンパイラの負荷が高まる。そして、このようなコンパイラの開発は難しいとされている。

【０００６】

以上のようなことから、所望のアプリケーションを容易に実現でき、かつリソースを有効に活用できる技術が望まれている。このような要望に応えるために、例えば、アーキテクチャの再構成を可能にしたＬＳＩ（リコンフィギュアブルＬＳＩ）の開発が進んでいる。特許文献１には、電気的に再構成可能なゲートアレイを用いて論理構成を構築する技術が開示されている。特許文献２には、プログラマブル論理チップの内部に入出力回路間の接続または入出力回路および論理・配線ブロック間の接続を可能にするプログラマブル配線領域を設けることで、マルチチップシステムを構成する場合の遅延量の増加やチップ間配線に使用される配線要素の増加を抑制する技術が開示されている。

【特許文献１】特開２０００－３６７３７号公報

【特許文献２】特開平６－２３１２８３号公報

【発明の開示】

【発明が解決しようとする課題】

【０００７】

ＬＳＩとして構成される演算処理装置において、演算処理部は、高速化および高集積化へ向かう傾向にあるが、外部インタフェース部は、必ずしも演算処理部と同じように高速化されるとは限らない。このため、演算処理部が必要以上に高速化および高集積化されている可能性がある。なぜならば、演算処理部の動作速度（処理速度）は、所望のスループット（単位時間あたりに演算処理装置が処理結果データを外部に出力する回数）を満たす範囲であれば遅くてもよいからである。演算処理部が過剰に高速化および高集積化されている場合、演算処理に使用されないリソース（レジスタや演算器など）が多数存在していることになり、回路面積が無駄に増大していることになる。

【０００８】

本発明は、このような問題に鑑みてなされたものであり、演算処理装置のスループットを低下させることなく、演算処理用のリソースの数を最小限に抑える技術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】

本発明の一形態では、演算処理装置は、複数の入力バッファ部、演算処理部、複数の出力バッファ部および制御部を備えて構成される。入力バッファ部は、複数のデータ系列に対応して設けられる。各入力バッファ部は、対応する外部入力データを格納して処理対象データとして出力する。演算処理部は、データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する。例えば、演算処理部は、複数のレジスタ、複数の演算器、複数のメモリおよび接続関係変更部を備えて構成される。レジスタ、演算器およびメモリは、演算処理部の構成要素として設けられる。接続関係変更部は、接続情報に応じて、レジスタ、演算器およびメモリの接続関係を変更する。

【００１０】

出力バッファ部は、データ系列に対応して設けられる。各出力バッファ部は、対応する処理結果データを格納して外部出力データとして出力する。制御部は、処理サイクル毎に、データ系列に対応する複数の命令情報に対して制御処理を所定順序で実施し、制御処理における演算処理部の構成要素の割り当て結果を接続情報として出力する。制御部は、制御処理において、命令情報が示す演算を実行可である場合に、演算に割り当てる演算処理部の構成要素を決定し、演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、出力バッファ部のデータ出力タイミングを満たしたうえで演算を実行すべき処理サイクルを次の処理サイクルに移行させる。

【００１１】

このような演算処理装置の内部構成（演算処理部の構成要素の数）は、以下に示す第１～第４工程を実施して決定される。まず、演算処理部の構成要素の数をパラメータ化した演算処理装置の回路記述を用いて、演算処理部の構成要素の数を、制御処理で常に演算実行可となる値に設定してシミュレーションを実施する（第１工程）。次に、第１工程で実施したシミュレーションの結果から、そのシミュレーションで制御部により演算に割り当てられた演算処理部の構成要素の数を仮設計値として取得する（第２工程）。

【００１２】

次に、演算処理部の構成要素の数をパラメータ化した演算処理装置の回路記述を用いて、演算処理部の構成要素の数を、仮設計値より小さい値を初期値として順次減少させながら、出力バッファ部のデータ出力タイミングが満たされなくなるまでシミュレーションを繰り返し実施する（第３工程）。そして、第３工程で実施したシミュレーションのうち、出力バッファ部のデータ出力タイミングが満たされた最後のシミュレーションの結果から、そのシミュレーションで制御部により演算に割り当てられた演算処理部の構成要素の数を実設計値として取得する（第４工程）。

【００１３】

例えば、演算処理部の構成要素として、複数のレジスタ、複数の演算器および複数のメモリが設けられる場合には、第１および第３工程において、演算処理装置の構成要素の数として、レジスタの数、演算器の数およびメモリの数をパラメータ化した演算処理装置の回路記述を用いてシミュレーションを実施する。好ましくは、第１および第３工程において、レジスタの数、演算器の数およびメモリの数に加えて、各メモリの容量をパラメータ化した演算処理装置の回路記述を用いてシミュレーションを実施する。

【００１４】

演算処理部の構成要素の数が前述の方法により取得された実設計値である演算処理装置では、ある命令情報に対する制御処理において、演算処理部の構成要素の不足に起因して命令情報が示す演算を実行すべき処理サイクルが現在の処理サイクルから次の処理サイクルに変更されたとしても、出力バッファ部のデータ出力タイミングは必ず満たされる。従って、演算処理装置のスループットを低下させることなく、演算処理部の構成要素の数（レジスタの数、演算器の数およびメモリの数）を最小限に抑えることができる。すなわち、演算処理装置の処理性能を保ちながら回路面積を削減できる。

【００１５】

また、前述のような構成の演算処理装置は、複数のデータ系列に対応する複数のメイン演算処理部と、メイン演算処理部に共通して設けられ、メイン演算処理部に代わって演算処理を実施するサブ演算処理部とを有する演算処理システムにおいて、サブ演算処理部として用いられる。このような演算処理システムとしては、マルチプロセッサが挙げられる。

マルチプロセッサにおいて、ネットワークあるいはバスを介して接続される複数のプロセッサコアが複数のメイン演算処理部に対応し、複数のプロセッサコアにより共有される共有演算処理部がサブ演算処理部に対応する。

【００１６】

このような構成のマルチプロセッサでは、各プロセッサコアにより実現される機能の要求性能を満たしつつ、演算処理用のリソースの数を最小限に抑えることができる。このため、マルチプロセッサの全体で高効率なリソース使用を実現できるとともに、マルチプロセッサの小面積化を図ることができる。そして、マルチプロセッサによるアプリケーションの実現を前提にしたソフトウェア設計の面では、従来、個々のプロセッサコアに対してソフトウェア設計時に実施されていたリソースの使用効率の最適化が共有演算処理部（本発明の演算処理装置）により実施されるため、ソフトウェア設計が容易になるという大きな効果が得られる。このように、本発明の演算処理装置をマルチプロセッサにおける共有演算処理部に用いることで、システム設計における総合的な最適化を実現できる。

【００１７】

本発明の前記一形態における好ましい例では、各入力バッファ部は、第１データ蓄積部

および第１優先度情報生成部を備えて構成される。第１データ蓄積部は、対応する外部入力データをライト要求に応答して格納し、格納している外部入力データをリード要求に応答して処理対象データとして出力する。第１優先度情報生成部は、第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を出力する。各出力バッファ部は、第２データ蓄積部および第２優先度情報生成部を備えて構成される。第２データ蓄積部は、対応する処理結果データをライト要求に応答して格納し、格納している処理結果データをリード要求に応答して外部出力データとして出力する。第２優先度情報生成部は、第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を出力する。制御部は、優先順位判定部および構成要素割当部を備えて構成される。優先順位判定部は、入力バッファ部から供給される複数の第１優先度情報と出力バッファ部から供給される複数の第２優先度情報とに応じてデータ系列の優先順位を決定する。構成要素割当部は、処理サイクル毎に、データ系列に対応する複数の命令情報に対して、対応するデータ系列の優先順位判定部により決定された優先順位が高い順に制御処理を実施し、制御処理における演算処理部の構成要素の割り当て結果を接続情報として出力する。

【００１８】

このため、入力バッファ部の第１データ蓄積部のデータ蓄積量が増加して、第１データ蓄積部からデータを出力する必要性が高くなっているデータ系列、あるいは出力バッファ部の第２データ蓄積部のデータ蓄積量が減少して、第２データ蓄積部にデータを格納する必要性が高くなっているデータ系列に対する演算処理を優先して実施させることができる。従って、各データ系列に対する演算処理をより安定して実施させることができる。

【００１９】

本発明の前記一形態における好ましい例では、第１優先度情報生成部は、第１カウンタおよび第１優先度判定部を備えて構成される。第１カウンタは、第１データ蓄積部のデータ格納動作に伴ってアップカウントし、第１データ蓄積部のデータ出力動作に伴ってダウンカウントする。第１優先度判定部は、第１カウンタのカウンタ値を第１閾値と比較して優先度を決定し、決定した優先度を示す第１優先度情報を出力する。第１カウンタのカウンタ値は、第１データ蓄積部のデータ蓄積量に対応しているため、第１優先度判定部は、第１カウンタのカウンタ値を第１閾値と比較するだけで、第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を生成できる。このため、第１優先度情報生成部を簡易な回路で構成できる。

【００２０】

本発明の前記一形態における好ましい例では、第２優先度情報生成部は、第２カウンタおよび第２優先度判定部を備えて構成される。第２カウンタは、第２データ蓄積部のデータ格納動作に伴ってアップカウントし、第２データ蓄積部のデータ出力動作に伴ってダウンカウントする。第２優先度判定部は、第２カウンタのカウンタ値を第２閾値と比較して優先度を決定し、決定した優先度を示す第２優先度情報を出力する。第２カウンタのカウンタ値は、第２データ蓄積部のデータ蓄積量に対応しているため、第２優先度判定部は、第２カウンタのカウンタ値を第２閾値と比較するだけで、第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を生成できる。このため、第２優先度情報生成部を簡易な回路で構成できる。

【００２１】

本発明の前記一形態における好ましい例では、優先順位判定部は、複数の第３優先度判定部および調停部を備えて構成される。第３優先度判定部は、データ系列に対応して設けられる。各第３優先度判定部は、対応する入力バッファ部から供給される第１優先度情報、対応する出力バッファ部から供給される第２優先度情報、および優先度定義テーブルを参照して優先度を決定し、決定した優先度を示す第３優先度情報を出力する。調停部は、第３優先度判定部から供給される複数の第３優先度情報に応じて、データ系列の優先順位を決定する。第３優先度判定部により、各データ系列の優先度が入力側の優先度と出力側の優先度とに基づいて決定され、調停部により、データ系列の優先順位が各データ系列の総合的な優先度に基づいて決定されるため、優先順位判定部を容易に構成できる。

【００２２】
本発明の前記一形態における好ましい例では、制御部に供給される命令情報は、処理サイクルあたりの演算の数を示す情報を含むとともに、演算毎に、演算の種類を示す情報と演算対象データを識別するための情報とを含む。演算に割り当てられる演算処理部の構成要素を一意に示す情報を命令情報に含ませる必要はなく、命令情報が示す演算に割り当てられる演算処理部の構成要素はハードウェア上で制御部により決定される。従って、命令情報を生成するコンパイラの負荷を軽減できる。
【発明の効果】
【００２３】
本発明によれば、演算処理装置のスループットを低下させることなく、演算処理部の構成要素（演算処理用のリソース）を最小限に抑えることができ、演算処理装置の小規模化に大きく寄与できる。
【発明を実施するための最良の形態】
【００２４】
本発明の実施形態を説明する前に、図面を用いて本発明の基本概念を説明する。
図１は、演算処理装置の概要を示している。ＬＳＩとして構成される演算処理装置において、演算処理部は、高速化および高集積化へ向かう傾向にあるが、入力ＩＦ部（入力インタフェース部）および出力ＩＦ部（出力インタフェース部）は、必ずしも演算処理部と同じように高速化されるとは限らない。このため、例えば、演算処理部の動作周波数が３００ＭＨｚであり、入力ＩＦ部および出力ＩＦ部の動作周波数が５０ＭＨｚである場合、演算処理部が過剰に高速化および高集積化されている可能性がある。従って、演算処理部におけるＲＥＧ（レジスタ）、ＡＬＵあるいはＭＵＬ（乗算器）などの演算処理用のリソースをより有効に活用する必要性が高まっている。演算処理用のリソースの有効活用を実現するためには、所定のハードウェア構成に対する最適なアルゴリズム（演算処理プログラム）、あるいは所定のアルゴリズムに対する最適なハードウェア構成を実現する必要がある。本発明は、所定のアルゴリズムに対する最適なハードウェア構成を実現することを目的としてなされたものである。
【００２５】
図２は、本発明で用いる命令情報の生成工程を示している。例えば、図２（ａ）の演算処理プログラムを図１のような構成の演算処理装置で実現する場合について説明する。まず、図２（ａ）のプログラム（演算処理）について、演算処理部の演算器（ＡＬＵやＭＵＬなど）を考慮しながら、演算処理を複数の演算要素に分解し、演算要素を互いに異なる処理サイクルに対応づけることで、図２（ｂ）のプログラムを生成する。なお、演算処理部の演算器は、２入力１出力の演算器（２つの演算対象データに対する演算を実施して１つの演算結果データを出力する演算器）であるものとする。
【００２６】
次に、図２（ｂ）のプログラムについて、演算の並列化を実現するために、各演算要素をデータ依存関係が破綻しない範囲で可能な限り前の処理サイクルに対応付けることで、図２（ｃ）のプログラムを生成する。そして、図２（ｃ）のプログラムについて、図２（ａ）のプログラムの作成時にはそれ程意識されない小変数化を実現するために、処理サイクルが変わる度に変数を開放して識別番号の小さい変数から順番に使用することで、図２（ｄ）のプログラムを生成する。この後、図２（ｄ）のプログラムをコード化して命令情報（命令コード）を生成する。本発明では、演算（図２のプロセスＰ０～Ｐ１０）の実行に使用されるハードウェア上のレジスタや演算器を命令情報で一意に決めず、演算の実行に使用されるレジスタや演算器はハードウェア上で決定される。例えば、命令情報は、処理サイクルあたりの演算の数を示す情報を含むとともに、演算毎に、演算の種類を示す情報と演算のオペランドを識別するための情報とを含むように生成される。
【００２７】
例えば、図２（ｄ）のプログラムをコード化して生成される第ｔサイクルの命令情報は、演算の数が６個であることを示す情報、第０演算の種類が”加算”であり第０演算のオ

ペランドがデータ”ａ”、”ｂ”であることを示す情報、第１演算の種類が”減算”であり第１演算のオペランドがデータ”ａ”、”ｂ”であることを示す情報、第２演算の種類が”加算”であり第２演算のオペランドがデータ”ｂ”、”ｃ”であることを示す情報、第３演算の種類が”減算”であり第３演算のオペランドがデータ”ｂ”、”ｃ”であることを示す情報、第４演算の種類が”加算”であり第４演算のオペランドがデータ”ｃ”、”ａ”であることを示す情報、第５演算の種類が”減算”であり第５演算のオペランドがデータ”ｃ”、”ａ”であることを示す情報を含むように生成される。従って、本発明で用いる命令情報を生成する際には、アーキテクチャのリソース制限がない。このため、コンパイラの負荷が軽減される。

【００２８】

図３は、演算へのリソースの割り当ての一例を示している。例えば、図２（ｄ）のプログラムから生成された命令情報が示す演算に対してリソースを割り当てる場合について説明する。演算処理部のリソースの数（ＲＥＧの数、ＡＬＵの数およびＭＵＬの数）は、十分に大きいものとする。ＡＬＵは、加算器および減算器の双方の機能を有するものとする。また、第ｔ－１サイクルにおいて、外部から入力インタフェース部に供給された外部入力データ”ａ”、”ｂ”、”ｃ”がＲＥＧ（０）、ＲＥＧ（１）、ＲＥＧ（２）にそれぞれ格納されているものとする。

【００２９】

第ｔサイクルにおいて、プロセスＰ０、Ｐ１を実行するために、演算対象データ（オペランド）の入力元としてＲＥＧ（０）、ＲＥＧ（１）を共通に割り当て、演算器としてＡＬＵ（０）を共通に割り当て、演算結果データの出力先としてＲＥＧ（０）、ＲＥＧ（１）を別々に割り当てる。プロセスＰ２、Ｐ３を実行するために、演算対象データの入力元としてＲＥＧ（１）、ＲＥＧ（２）を共通に割り当て、演算器としてＡＬＵ（１）を共通に割り当て、演算結果データの出力先としてＲＥＧ（２）、ＲＥＧ（３）を別々に割り当てる。プロセスＰ４、Ｐ５を実行するために、演算対象データの入力元としてＲＥＧ（２）、ＲＥＧ（０）を共通に割り当て、演算器としてＡＬＵ（２）を共通に割り当て、演算結果データの出力先としてＲＥＧ（４）、ＲＥＧ（５）を別々に割り当てる。

【００３０】

第ｔ＋１サイクルにおいて、プロセスＰ６を実行するために、演算対象データの入力元としてＲＥＧ（０）、ＲＥＧ（１）を割り当て、演算器としてＭＵＬ（０）を割り当て、演算結果データの出力先としてＲＥＧ（０）を割り当てる。プロセスＰ７を実行するために、演算対象データの入力元としてＲＥＧ（２）、ＲＥＧ（３）を割り当て、演算器としてＭＵＬ（１）を割り当て、演算結果データの出力先としてＲＥＧ（１）を割り当てる。プロセスＰ８を実行するために、演算対象データの入力元としてＲＥＧ（４）、ＲＥＧ（５）を割り当て、演算器としてＭＵＬ（２）を割り当て、演算結果データの出力先としてＲＥＧ（２）を割り当てる。

【００３１】

第ｔ＋２サイクルにおいて、プロセスＰ９を実行するために、演算対象データの入力元としてＲＥＧ（０）、ＲＥＧ（１）を割り当て、演算器としてＡＬＵ（０）を割り当て、演算結果データの出力先としてＲＥＧ（０）を割り当てる。また、プロセスＰ８の演算結果データの転送を実行するために、転送元としてＲＥＧ（２）を割り当て、転送先としてＲＥＧ（１）を割り当てる。

【００３２】

第ｔ＋３サイクルにおいて、プロセスＰ１０を実行するために、演算対象データの入力元としてＲＥＧ（０）、ＲＥＧ（１）を割り当て、演算器としてＡＬＵ（０）を割り当て、演算結果データの出力先としてＲＥＧ（０）を割り当てる。これにより、出力インタフェース部は、第ｔ＋４サイクルにおいて、データ”ｘ”を外部に出力することが可能な状態になる。

【００３３】

このように、演算処理部のリソースの数が十分に大きいものと仮定すると、命令情報が

示す最大限に並列化された演算が同一の処理サイクルで実行されるように、リソースが割り当てられるため、演算処理部の処理速度は非常に速くなる。しかしながら、入力インタフェース部および出力インタフェース部の動作速度がそれに比べて遅い場合、演算処理部の処理速度が過剰に速くなっている可能性がある。演算処理部の処理速度は、演算処理装置のスループットを満たす範囲であれば遅くても問題はない。

【００３４】

図４および図５は、本発明における演算へのリソースの割り当ての基本概念を示している。データ系列０、１に対する２つの演算処理を演算処理部で実施することを考える。例えば、２つの演算処理は図２（ａ）の演算処理で同一であり、図２（ｄ）のプログラムから生成された命令情報が示す演算に対してリソースを割り当てる場合について説明する。演算処理部のリソースの数（ＲＥＧの数、ＡＬＵの数およびＭＵＬの数）は、十分に大きいものとする。ＡＬＵは、加算器および減算器の双方の機能を有するものとする。また、第ｔ－１サイクルにおいて、外部から系列０の入力インタフェース部０に供給された外部入力データ”ａ”、”ｂ”、”ｃ”がＲＥＧ（０）、ＲＥＧ（１）、ＲＥＧ（２）に格納されており、外部から系列１の入力インタフェース部１に供給されたデータ”ａ”、”ｂ”、”ｃ”がＲＥＧ（３）、ＲＥＧ（４）、ＲＥＧ（５）に格納されているものとする。また、系列０に対する演算処理が系列１に対する演算処理よりも優先して実施されるものとする。

【００３５】

このような場合、図４に示すように、第ｔサイクルにおいて、系列０のプロセスＰ０～Ｐ５を実行するために、図３と同様に、リソースを割り当てる。続いて、系列１のプロセスＰ０、Ｐ１を実行するために、演算対象データの入力元としてＲＥＧ（３）、ＲＥＧ（４）を共通に割り当て、演算器としてＡＬＵ（３）を共通に割り当て、演算結果データの出力先としてＲＥＧ（６）、ＲＥＧ（７）を別々に割り当てる。系列１のプロセスＰ２、Ｐ３を実行するために、演算対象データの入力元としてＲＥＧ（４）、ＲＥＧ（５）を共通に割り当て、演算器としてＡＬＵ（４）を共通に割り当て、演算結果データの出力先としてＲＥＧ（８）、ＲＥＧ（９）を別々に割り当てる。系列１のプロセスＰ４、Ｐ５を実行するために、演算対象データの入力元としてＲＥＧ（５）、ＲＥＧ（３）を共通に割り当て、演算器としてＡＬＵ（５）を共通に割り当て、演算結果データの出力先としてＲＥＧ（１０）、ＲＥＧ（１１）を別々に割り当てる。

【００３６】

第ｔ＋１サイクルにおいて、系列０のプロセスＰ６～Ｐ８を実行するために、図３と同様に、リソースを割り当てる。続いて、系列１のプロセスＰ６を実行するために、演算対象データの入力元としてＲＥＧ（６）、ＲＥＧ（７）を割り当て、演算器としてＭＵＬ（３）を割り当て、演算結果データの出力先としてＲＥＧ（３）を割り当てる。系列１のプロセスＰ７を実行するために、演算対象データの入力元としてＲＥＧ（８）、ＲＥＧ（９）を割り当て、演算器としてＭＵＬ（４）を割り当て、演算結果データの出力先としてＲＥＧ（４）を割り当てる。系列１のプロセスＰ８を実行するために、演算対象データの入力元としてＲＥＧ（１０）、ＲＥＧ（１１）を割り当て、演算器としてＭＵＬ（５）を割り当て、演算結果データの出力先としてＲＥＧ（５）を割り当てる。

【００３７】

第ｔ＋２サイクルにおいて、系列０におけるプロセスＰ９およびプロセスＰ８の演算結果データの転送を実行するために、図３と同様に、リソースを割り当てる。続いて、系列１のプロセスＰ９を実行するために、演算対象データの入力元としてＲＥＧ（３）、ＲＥＧ（４）を割り当て、演算器としてＡＬＵ（１）を割り当て、演算結果データの出力先としてＲＥＧ（２）を割り当てる。また、系列１におけるプロセスＰ８の演算結果データの転送を実行するために、転送元としてＲＥＧ（５）を割り当て、転送先としてＲＥＧ（３）を割り当てる。

【００３８】

第ｔ＋３サイクルにおいて、系列０のプロセスＰ１０を実行するために、図３と同様に

、リソースを割り当てる。続いて、系列１のプロセスＰ１０を実行するために、演算対象データの入力元としてＲＥＧ（２）、ＲＥＧ（３）を割り当て、演算器としてＡＬＵ（１）を割り当て、演算結果データの出力先としてＲＥＧ（１）を割り当てる。これにより、系列０の出力インタフェース部０は、第ｔ＋４サイクルにおいて、データ”ｘ”を外部に出力することが可能な状態になる。また、系列１の出力インタフェース部１も、第ｔ＋４サイクルにおいて、データ”ｘ”を外部に出力することが可能な状態になる。

【００３９】

このように、演算処理部のリソースの数が十分に大きいものと仮定すると、多数のリソースを使用して最速に２つの演算処理を実施できる。しかしながら、このリソース割り当てでは、瞬間的に（第ｔサイクルのみで）リソースが多く使用され、その他の期間ではリソースがあまり使用されないため、演算処理装置の系列０、１のスループットが低ければ、相対的にリソースが過剰に確保されていることになる。

【００４０】

そこで、例えば、ＲＥＧの数、ＡＬＵの数、ＭＵＬの数がそれぞれ１２個、５個、６個であるものとして、演算にリソースを割り当てる場合について説明する。このような場合、図５に示すように、第ｔサイクルにおいて、系列０のプロセスＰ０～Ｐ５を実行するために、図３と同様に、リソースを割り当てる。系列１のプロセスＰ０～Ｐ５を実行するためには、ＡＬＵが３個必要であるが、未割り当てのＡＬＵは２個しかない。このため、系列１のプロセスＰ０～Ｐ５を実行すべき処理サイクルを次の処理サイクル（第ｔ＋１サイクル）に変更する。なお、プロセスＰ４、Ｐ５だけではなく、プロセスＰ０～Ｐ３も次の処理サイクルにシフトするのは、ハードウェアの設計を容易にするためである。この際、ＲＥＧ（３）、ＲＥＧ（４）、ＲＥＧ（５）にそれぞれ格納されているデータ”ａ”、”ｂ”、”ｃ”を次の処理サイクルで使用するために退避させる必要がある。従って、ＲＥＧ（３）に格納されているデータ”ａ”を退避させるために、転送元としてＲＥＧ（３）を割り当て、転送先としてＲＥＧ（６）を割り当てる。ＲＥＧ（４）に格納されているデータ”ｂ”を退避させるために、転送元としてＲＥＧ（４）を割り当て、転送先としてＲＥＧ（７）を割り当てる。ＲＥＧ（５）に格納されているデータ”ｃ”を退避させるために、転送元としてＲＥＧ（５）を割り当て、転送先としてＲＥＧ（８）を割り当てる。

【００４１】

第ｔ＋１サイクルにおいて、系列０のプロセスＰ６～Ｐ８を実行するために、図３と同様に、リソースを割り当てる。続いて、第ｔサイクルで実行されるはずであった系列１のプロセスＰ０、Ｐ１を実行するために、演算対象データの入力元としてＲＥＧ（６）、ＲＥＧ（７）を共通に割り当て、演算器としてＡＬＵ（０）を共通に割り当て、演算結果データの出力先としてＲＥＧ（３）、ＲＥＧ（４）を別々に割り当てる。第ｔサイクルで実行されるはずであった系列１のプロセスＰ２、Ｐ３を実行するために、演算対象データの入力元としてＲＥＧ（７）、ＲＥＧ（８）を共通に割り当て、演算器としてＡＬＵ（１）を共通に割り当て、演算結果データの出力先としてＲＥＧ（５）、ＲＥＧ（６）を別々に割り当てる。第ｔサイクルで実行されるはずであった系列１のプロセスＰ４、Ｐ５を実行するために、演算対象データの入力元としてＲＥＧ（８）、ＲＥＧ（６）を共通に割り当て、演算器としてＡＬＵ（２）を共通に割り当て、演算結果データの出力先としてＲＥＧ（７）、ＲＥＧ（８）を別々に割り当てる。

【００４２】

第ｔ＋２サイクルにおいて、系列０におけるプロセスＰ９およびプロセスＰ８の演算結果データの転送を実行するために、図３と同様に、リソースを割り当てる。続いて、第ｔ＋１サイクルで実行されるはずであった系列１のプロセスＰ６を実行するために、演算対象データの入力元としてＲＥＧ（３）、ＲＥＧ（４）を割り当て、演算器としてＭＵＬ（０）を割り当て、演算結果データの出力先としてＲＥＧ（２）を割り当てる。第ｔ＋１サイクルで実行されるはずであった系列１のプロセスＰ７を実行するために、演算対象データの入力元としてＲＥＧ（５）、ＲＥＧ（６）を割り当て、演算器としてＭＵＬ（１）を割り当て、演算結果データの出力先としてＲＥＧ（３）を割り当てる。第ｔ＋１サイクル

で実行されるはずであった系列１のプロセスＰ８を実行するために、演算対象データの入力元としてＲＥＧ（７）、ＲＥＧ（８）を割り当て、演算器としてＭＵＬ（２）を割り当て、演算結果データの出力先としてＲＥＧ（４）を割り当てる。

【００４３】

第ｔ＋３サイクルにおいて、系列０のプロセスＰ１０を実行するために、図３と同様に、リソースを割り当てる。これにより、系列０の出力インタフェース部０は、第ｔ＋４サイクルにおいて、データ”ｘ”を外部に出力することが可能な状態になる。続いて、第ｔ＋２サイクルで実行されるはずであった系列１のプロセスＰ９を実行するために、演算対象データの入力元としてＲＥＧ（２）、ＲＥＧ（３）を割り当て、演算器としてＡＬＵ（１）を割り当て、演算結果データの出力先としてＲＥＧ（１）を割り当てる。また、第ｔ＋２サイクルで実行されるはずであった系列１におけるプロセスＰ８の演算結果データの転送を実行するために、転送元としてＲＥＧ（４）を割り当て、演算結果データの出力先としてＲＥＧ（２）を割り当てる。

【００４４】

第ｔ＋４サイクルにおいて、第ｔ＋３サイクルで実行されるはずであった系列１のプロセスＰ１０を実行するために、演算対象データの入力元としてＲＥＧ（１）、ＲＥＧ（２）を割り当て、演算器としてＡＬＵ（０）を割り当て、演算結果データの出力先としてＲＥＧ（０）を割り当てる。これにより、系列１の出力インタフェース部１は、第ｔ＋５サイクルにおいて、データ”ｘ”を外部に出力することが可能な状態になる。

【００４５】

このような場合、第ｔサイクルにおいて系列１のプロセスＰ４、Ｐ５を実行できなくなるため、系列１のプロセスＰ４、５がプロセスＰ０～Ｐ３と共に第ｔ＋１サイクルへシフトされる。これに伴い、系列１のプロセスＰ６～Ｐ１０が１サイクル後ろにシフトされる。この結果、系列１の出力インタフェース部１は、第ｔ＋４サイクルではなく第ｔ＋５サイクルでデータ”ｘ”を外部に出力することが可能な状態になるが、系列１の出力インタフェース部１に要求されるデータ出力タイミングが第ｔ＋５サイクルであれば問題はない。つまり、外部から第ｔ－１サイクル、第ｔ＋５サイクル、第ｔ＋１１サイクル（図示せず）というコンスタントな時間間隔（６サイクル周期）でデータが入力されても、系列０は第ｔ＋４サイクル、第ｔ＋１０サイクル（図示せず）、第ｔ＋１６サイクル（図示せず）に結果を出力することができ、系列１は第ｔ＋５サイクル、第ｔ＋１１サイクル（図示せず）、第ｔ＋１７サイクル（図示せず）に結果を出力することができる。すなわち、外部からのデータの入力時間間隔（６サイクル）を所望のスループットとする場合に、これを満たすことができる。従って、演算処理装置のスループットを保ちながらリソース（ＡＬＵ）を削減できたことになる。この例では、ＡＬＵのみならず、ＲＥＧやＭＵＬも削減できる。本発明は、以上のような基本概念に基づいてなされたものである。

【００４６】

以下、図面を用いて本発明の実施形態を説明する。図６は、本発明の一実施形態を示している。図７は、図６の入力制御部の詳細を示している。図８は、図７の入力優先レベル判定部の動作例を示している。図９は、図６の出力制御部の詳細を示している。図１０は、図９の出力優先レベル判定部の動作例を示している。図１１は、図６の命令情報メモリに格納される命令情報の概要を示している。図１２は、図６の命令情報メモリに格納される命令情報の具体例を示している。図１３は、図６の命令情報メモリに格納される命令情報の別の具体例を示している。図１４は、図６のシステム制御部の詳細を示している。図１５は、図１４の総合優先レベル判定部で用いられる優先レベル定義テーブルの具体例を示している。図１６は、図１４のリソース割当部で用いられる内部変数を示している。図１７および図１８は、図１４のリソース割当部の動作を示している。

【００４７】

図６に示すように、演算処理装置ＯＰＤは、入力バッファ部ＩＢ０～ＩＢｎ（ｎ：１以上の整数）、演算処理部ＯＰ、出力バッファ部ＯＢ０～ＯＢｎ、命令情報メモリＩＮＳＭ０～ＩＮＳＭｎおよびシステム制御部ＳＹＳＣを有している。

入力バッファ部ＩＢｉ（ｉ＝０、・・・、ｎ）は、系列ｉに対応して設けられている。入力バッファ部ＩＢｉは、入力ＦＩＦＯ部ＩＦｉおよび入力制御部ＩＣｉを有している。入力ＦＩＦＯ部ＩＦｉは、外部から供給されるライトコマンドＷＣＭＩｉに応答して、入力制御部ＩＣｉから供給されるライトアドレスＷＡＤＩｉが示す番地に、外部から供給される外部入力データＤＩｉを格納する。入力ＦＩＦＯ部ＩＦｉは、入力制御部ＩＣｉから供給されるリードコマンドＲＣＭＩｉに応答して、入力制御部ＩＣｉから供給されるリードアドレスＲＡＤＩｉが示す番地に格納されている外部入力データを、処理対象データＤＰＯｉとして演算処理部ＯＰに出力する。

【００４８】

図７に示すように、入力制御部ＩＣｉは、ライト制御部ＷＣＩｉ、リード制御部ＲＣＩｉ、カウンタＣＩｉおよび入力優先レベル判定部ＩＬＤｉを有している。ライト制御部ＷＣＩｉは、外部から供給されるライトコマンドＷＣＭＩｉに応答して、ライトアドレスＷＡＤＩｉを入力ＦＩＦＯ部ＩＦｉに出力する。リード制御部ＲＣＩｉは、システム制御部ＳＹＳＣから供給される入力許可通知ＩＥＮｉに応答して、リードコマンドＲＣＭＩｉおよびリードアドレスＲＡＤＩｉを入力ＦＩＦＯ部ＩＦｉに出力する。カウンタＣＩｉは、ライトコマンドＷＣＭＩｉに応答してアップカウントし、入力許可通知ＩＥＮｉに応答してダウンカウントする。すなわち、カウンタＣＩｉのカウンタ値ＣＶＩｉは、ライトコマンドＷＣＭＩｉに応答して増加し、入力許可通知ＩＥＮｉに応答して減少する。従って、カウンタＣＩｉのカウンタ値ＣＶＩｉは、入力ＦＩＦＯ部ＩＦｉのデータ蓄積量に対応している。カウンタＣＩｉは、カウンタ値ＣＶＩｉを入力優先レベル判定部ＩＬＤｉに出力する。

【００４９】

入力優先レベル判定部ＩＬＤｉは、ソフトウェアにより閾値ＶＩ０ｉ～ＶＩ３ｉ（ＶＩ０ｉ＜ＶＩ１ｉ＜ＶＩ２ｉ＜ＶＩ３ｉ）が設定される閾値レジスタＴＲＩｉを有している。入力優先レベル判定部ＩＬＤｉは、カウンタＣＩｉから供給されるカウンタ値ＣＶＩｉを閾値レジスタＴＲＩｉに設定されている閾値ＶＩ０ｉ～ＶＩ３ｉと比較することで優先レベルを決定し、決定した優先レベルを示す入力優先レベル情報ＩＬＩｉをシステム制御部ＳＹＳＣに出力する。例えば、図８に示すように、入力優先レベル判定部ＩＬＤｉは、カウンタ値ＣＶＩｉが閾値ＶＩ０ｉ未満であるとき、優先レベルを”０”に決定する。入力優先レベル判定部ＩＬＤｉは、カウンタ値ＣＶＩｉが閾値ＶＩ０ｉ以上かつ閾値ＶＩ１ｉ未満であるとき、優先レベルを”１”に決定する。入力優先レベル判定部ＩＬＤｉは、カウンタ値ＣＶＩｉが閾値ＶＩ１ｉ以上かつ閾値ＶＩ２ｉ未満であるとき、優先レベルを”２”に決定する。入力優先レベル判定部ＩＬＤｉは、カウンタ値ＣＶＩｉが閾値ＶＩ２ｉ以上かつ閾値ＶＩ３ｉ未満であるとき、優先レベルを”３”に決定する。入力優先レベル判定部ＩＬＤｉは、カウンタ値ＣＶＩｉが閾値ＶＩ３ｉ以上であるとき、優先レベルを”４”に決定する。このように、入力優先レベル判定部ＩＬＤｉは、カウンタＣＩｉのカウンタ値ＣＶＩｉが大きいほど高い優先レベルに決定する。換言すれば、入力優先レベル判定部ＩＬＤｉは、入力ＦＩＦＯ部ＩＦｉのデータ蓄積量が多いほど高い優先レベルに決定する。

【００５０】

図６に示すように、演算処理部ＯＰは、レジスタ群ＲＧＲ、演算器群ＯＧＲ、メモリ群ＭＧＲおよびマルチプレクサＭＰＸを有している。レジスタ群ＲＧＲは、複数のレジスタＲＥＧを備えて構成されている。演算器群ＯＧＲは、”演算Ａ”を実施するための複数の演算器ＯＰＡと、”演算Ｂ”を実施するための複数の演算器ＯＰＢと、”演算Ｃ“を実施するための複数の演算器ＯＰＣとを備えて構成されている。例えば、演算器ＯＰＡ、ＯＰＢ、ＯＰＣは、２入力１出力の演算器である。メモリ群ＭＧＲは、複数のメモリＭＥＭを備えて構成されている。マルチプレクサＭＰＸは、システム制御部ＳＹＳＣから供給される接続情報ＣＩに基づいて、レジスタ群ＲＧＲにおけるレジスタＲＥＧ、演算器群ＯＧＲにおける演算器ＯＰＡ、ＯＰＢ、ＯＰＣおよびメモリ群ＭＧＲにおけるメモリＭＥＭの接続関係を変更する。このような演算処理部ＯＰは、接続情報ＣＩに応じて内部構成を変更

することで、入力バッファ部ＩＢｉから供給される処理対象データＤＰＯｉに対して系列ｉ用の演算処理を実施して処理結果データＤＰＲｉを出力バッファ部ＯＢｉに出力する。

【００５１】

出力バッファ部ＯＢｉは、系列ｉに対応して設けられている。出力バッファ部ＯＢｉは、出力ＦＩＦＯ部ＯＦｉおよび出力制御部ＯＣｉを有している。出力ＦＩＦＯ部ＯＦｉは、出力制御部ＯＣｉから供給されるライトコマンドＷＣＭＯｉに応答して、出力制御部ＯＣｉから供給されるライトアドレスＷＡＤＯｉが示す番地に、演算処理部ＯＰから供給される処理結果データＤＰＲｉを格納する。出力ＦＩＦＯ部ＯＦｉは、外部から供給されるリードコマンドＲＣＭＯｉに応答して、出力制御部ＯＣｉから供給されるリードアドレスＲＡＤＯｉが示す番地に格納されている処理結果データを、外部出力データＤＯｉとして外部に出力する。

【００５２】

図９に示すように、出力制御部ＯＣｉは、ライト制御部ＷＣＯｉ、リード制御部ＲＣＯｉ、カウンタＣＯｉおよび出力優先レベル判定部ＯＬＤｉを有している。ライト制御部ＷＣＯｉは、システム制御部ＳＹＳＣから供給される出力許可通知ＯＥＮｉに応答して、ライトコマンドＷＣＭＯｉおよびライトアドレスＷＡＤＯｉを出力ＦＩＦＯ部ＯＦｉに出力する。リード制御部ＲＣＯｉは、外部から供給されるリードコマンドＲＣＭＯｉに応答してリードアドレスＲＡＤＯｉを出力ＦＩＦＯ部ＯＦｉに出力する。カウンタＣＯｉは、出力許可通知ＯＥＮｉに応答してアップカウントし、リードコマンドＲＣＭＯｉに応答してダウンカウントする。すなわち、カウンタＣＯｉのカウンタ値ＣＶＯｉは、出力許可通知ＯＥＮｉに応答して増加し、リードコマンドＲＣＭＯｉに応答して減少する。従って、カウンタＣＯｉのカウンタ値ＣＶＯｉは、出力ＦＩＦＯ部ＯＦｉのデータ蓄積量に対応している。カウンタＣＯｉは、カウンタ値ＣＶＯｉを出力優先レベル判定部ＯＬＤｉに出力する。

【００５３】

出力優先レベル判定部ＯＬＤｉは、ソフトウェアにより閾値ＶＯ０ｉ～ＶＯ３ｉ（ＶＯ０ｉ＜ＶＯ１ｉ＜ＶＯ２ｉ＜ＶＯ３ｉ）が設定される閾値レジスタＴＲＯｉを有している。出力優先レベル判定部ＯＬＤｉは、カウンタＣＯｉから供給されるカウンタ値ＣＶＯｉを閾値レジスタＴＲＯｉに設定されている閾値ＶＯ０ｉ～ＶＯ３ｉと比較することで優先レベルを決定し、決定した優先レベルを示す出力優先レベル情報ＯＬＩｉをシステム制御部ＳＹＳＣに出力する。例えば、図１０に示すように、出力優先レベル判定部ＯＬＤｉは、カウンタ値ＣＶＯｉが閾値ＶＯ０ｉ未満であるとき、優先レベルを”４”に決定する。出力優先レベル判定部ＯＬＤｉは、カウンタ値ＣＶＯｉが閾値ＶＯ０ｉ以上かつ閾値ＶＯ１ｉ未満であるとき、優先レベルを”３”に決定する。出力優先レベル判定部ＯＬＤｉは、カウンタ値ＣＶＯｉが閾値ＶＯ１ｉ以上かつ閾値ＶＯ２未満であるとき、優先レベルを”２”に決定する。出力優先レベル判定部ＯＬＤｉは、カウンタ値ＣＶＯｉが閾値ＶＯ２ｉ以上かつ閾値ＶＯ３ｉ未満であるとき、優先レベルを”１”に決定する。出力優先レベル判定部ＯＬＤｉは、カウンタ値ＣＶＯｉが閾値ＶＯ３ｉ以上であるとき、優先レベルを”０”に決定する。このように、出力優先レベル判定部ＯＬＤｉは、カウンタＣＯｉのカウンタ値ＣＶＯｉが大きいほど低い優先レベルに決定する。換言すれば、出力優先レベル判定部ＯＬＤｉは、出力ＦＩＦＯ部ＯＦｉのデータ蓄積量が多いほど低い優先レベルに決定する。

【００５４】

図６に示すように、命令情報メモリＩＮＳＭｉは、系列ｉに対応して設けられている。命令情報メモリＩＮＳＭｉは、系列ｉ用の演算処理プログラムをコンパイルして生成された複数の命令情報を格納している。命令情報メモリＩＮＳＭｉは、システム制御部ＳＹＳＣから供給されるリードコマンドＲＣＭＭｉに応答して、システム制御部ＳＹＳＣから供給されるリードアドレスＲＡＤＭｉが示す番地に格納されている命令情報を、システム制御部ＳＹＳＣに供給する命令情報ＩＮＳｉとして選択する。

【００５５】

例えば、図１１に示すように、命令情報メモリＩＮＳＭｉに格納されている命令情報は、共通情報およびｍ＋１個（ｍ：０以上の整数）の演算情報で構成されている。共通情報は、ｅｘｅｃ＿ｎｕｍおよびｏｐｄ＿ｎｕｍをフィールドとして有している。ｅｘｅｃ＿ｎｕｍは、処理サイクルあたりの演算の数を示すフィールドである。ｏｐｄ＿ｎｕｍは、演算処理部ＯＰ（レジスタ群ＲＧＲ）のレジスタＲＥＧが入力元であるオペランドの数を示すフィールドである。

【００５６】

第ｊ演算情報は、ｅｘｅｃ＿ｔｙｐｅ[ｊ]、ｏｐｄ０[ｊ]、ｏｐｄ１[ｊ]、ｆｉｎ０[ｊ]、ｆｉｎ１[ｊ]およびｆｏｕｔ[ｊ]をフィールドとして有している。ｅｘｅｃ＿ｔｙｐｅ[ｊ]は、第ｊ演算の種類を示すフィールドである。例えば、第ｊ演算の種類が”加算”であるとき、ｅｘｅｃ＿ｔｙｐｅ[ｊ]は”０”に設定される。第ｊ演算の種類が”減算”であるとき、ｅｘｅｃ＿ｔｙｐｅ[ｊ]は”１”に設定される。第ｊ演算の種類が”乗算”であるとき、ｅｘｅｃ＿ｔｙｐｅ[ｊ]は”２”に設定される。第ｊ演算の種類が”転送”であるとき、ｅｘｅｃ＿ｔｙｐｅ[ｊ]は”３”に設定される。ｏｐｄ０[ｊ]は、第ｊ演算のオペランド０（第ｊ演算で用いられる２つの演算対象データの一方）の識別番号を示すフィールドである。ｏｐｄ１[ｊ]は、第ｊ演算のオペランド１（第ｊ演算で用いられる２つの演算対象データの他方）の識別番号を示すフィールドである。

【００５７】

ｆｉｎ０[ｊ]は、第ｊ演算のオペランド０の格納場所を示すフィールドである。例えば、第ｊ演算のオペランド０の格納場所が演算処理部ＯＰのレジスタＲＥＧである場合、ｆｉｎ０[ｊ]は”０”に設定される。第ｊ演算のオペランド０の格納場所が入力バッファ部ＩＢｉの入力ＦＩＦＯ部ＩＦｉである場合、ｆｉｎ０[ｊ]は”１”に設定される。ｆｉｎ１[ｊ]は、第ｊ演算のオペランド１の格納場所を示すフィールドである。例えば、ｆｉｎ０[ｊ]と同様に、第ｊ演算のオペランド１の格納場所が演算処理部ＯＰのレジスタＲＥＧである場合、ｆｉｎ１[ｊ]は”０”に設定される。第ｊ演算のオペランド１の格納場所が入力バッファ部ＩＢｉの入力ＦＩＦＯ部ＩＦｉである場合、ｆｉｎ１[ｊ]は”１”に設定される。ｆｏｕｔ[ｊ]は、第ｊ演算の演算結果データの格納場所を示すフィールドである。例えば、第ｊ演算の演算結果データの格納場所が演算処理部ＯＰのレジスタＲＥＧである場合、ｆｏｕｔ[ｊ]は”０”に設定される。第ｊ演算の演算結果データの格納場所が出力バッファ部ＯＢｉの出力ＦＩＦＯ部ＯＦｉである場合、ｆｏｕｔ[ｊ]は”１”に設定される。例えば、図２（ａ）の演算処理プログラムをコンパイルして生成される命令情報は、図１２に示すとおりである。また、図１３（ａ）の演算処理プログラムをコンパイルして得られる命令情報は、図１３（ｅ）に示すとおりである。

【００５８】

図１４に示すように、システム制御部ＳＹＳＣは、総合優先レベル判定部ＴＬＤ０～ＴＬＤｎ、調停部ＡＢＴ、メモリアクセス部ＭＡ０～ＭＡｎおよびリソース割当部ＲＡを有している。総合優先レベル判定部ＴＬＤｉは、系列ｉに対応して設けられている。総合優先レベル判定部ＴＬＤｉは、入力バッファ部ＩＢｉから供給される入力優先レベル情報ＩＬＩｉと出力バッファ部ＯＢｉから供給される出力優先レベル情報ＯＬＩｉとを受けて優先レベル定義テーブルを参照することで優先レベルを決定し、決定した優先レベルを示す総合優先レベル情報ＴＬＩｉを調停部ＡＢＴに出力する。例えば、総合優先レベル判定部ＴＬＤｉは、入力優先レベル情報ＩＰＬｉが示す優先レベルが”ａ”（ａ＝０、１、・・・、４）であり、かつ出力優先レベル情報ＯＰＬｉが示す優先レベルが”ｂ”（ｂ＝０、１、・・・、４）であるとき、図１５に示すような優先レベル定義テーブルを参照することで優先レベルを”ａ＋ｂ”に決定し、”ａ＋ｂ”を示す総合優先レベル情報ＴＬＩｉを調停部ＡＢＴに出力する。

【００５９】

図１４に示すように、調停部ＡＢＴは、総合優先レベル判定部ＴＬＤ０～ＴＬＤｎからそれぞれ供給される総合優先レベル情報ＴＬＩ０～ＴＬＩｎに基づいて、系列０～ｎの優先順位を決定し、決定した優先順位を示す優先順位情報ＰＩをリソース割当部ＲＡに出力

する。具体的には、調停部ＡＢＴは、系列０～ｎの優先順位を、対応する総合優先レベル情報が示す優先レベルが高い系列ほど優先順位が高くなるように決定する。

【００６０】

メモリアクセス部ＭＡｉは、プログラムカウンタＰＣｉおよびリード制御部ＲＣＭｉを有している。プログラムカウンタＰＣｉは、リソース割当部ＲＡから供給されるアクセス要求ＡＲｉに応答してカウンタ値ＰＣＶｉを更新する。プログラムカウンタＰＣｉは、カウンタ値ＰＣＶｉをリード制御部ＲＣＭｉに出力する。リード制御部ＲＣｉは、プログラムカウンタＰＣｉから供給されるカウンタ値ＰＣＶｉの変化に伴って、カウンタ値ＰＣＶｉに応じたリードアドレスＲＡＤＭｉをリードコマンドＲＣＭＭｉと共に命令情報メモリＩＮＳＭｉに出力する。

【００６１】

リソース割当部ＲＡは、命令情報メモリＩＮＳＭ０～ＩＮＳＭｎからそれぞれ供給される命令情報ＩＮＳ０～ＩＮＳｎと、調停部ＡＢＴから供給される優先順位情報ＰＩとを受けて、後述の内部変数を用いた動作を実施することで、接続情報ＣＩ、入力許可通知ＩＥＮ０～ＩＥＮｎ、出力許可通知ＯＥＮ０～ＯＥＮｎおよびアクセス要求ＡＲ０～ＡＲｎを所望のタイミングでそれぞれ出力する。

【００６２】

リソース割当部ＲＡは、図１６に示すｅｘｅｃ＿ｎｕｍ[ｉ]、ｏｐｄ＿ｎｕｍ[ｉ]、ｅｘｅｃ＿ｔｙｐｅ[ｉ][ｊ]、ｏｐｄ０[ｉ][ｊ]、ｏｐｄ１[ｉ][ｊ]、ｆｉｎ０[ｉ][ｊ]、ｆｉｎ１[ｉ][ｊ]、ｆｏｕｔ[ｉ][ｊ]、ｖｒ、ｖａ、ｖｂ、ｖｃ、ｏｆｓｔ＿ｒ、ｏｆｓｔ＿ａ、ｏｆｓｔ＿ｂ、ｏｆｓｔ＿ｃ、ｏｆｓｔ＿ｒ＿ｌ[ｉ]、ｓｔａｄ＿ｆｉｎ[ｉ]、ｓｔａｄ＿ｆｏｕｔ[ｉ]およびｐｒｉを内部変数として有している。

【００６３】

ｅｘｅｃ＿ｎｕｍ[ｉ]およびｏｐｄ＿ｎｕｍ[ｉ]は、命令情報ＩＮＳｉが命令情報メモリＩＮＳＭｉからリソース割当部ＲＡに供給されたときに、命令情報ＩＮＳｉにおける共通情報のｅｘｅｃ＿ｎｕｍおよびｏｐｄ＿ｎｕｍの値にそれぞれ設定される。ｏｐｄ０[ｉ][ｊ]、ｏｐｄ１[ｉ][ｊ]、ｆｉｎ０[ｉ][ｊ]、ｆｉｎ１[ｉ][ｊ]およびｆｏｕｔ[ｉ][ｊ]は、命令情報ＩＮＳｉが命令情報メモリＩＮＳＭｉからリソース割当部ＲＡに供給されたときに、命令情報ＩＮＳｉにおける第ｊ演算情報のｏｐｄ０[ｊ]、ｏｐｄ１[ｊ]、ｆｉｎ０[ｊ]、ｆｉｎ１[ｊ]およびｆｏｕｔ[ｊ]の値にそれぞれ設定される。すなわち、ｅｘｅｃ＿ｎｕｍ[ｉ]は、系列ｉにおける演算の数を示す変数である。ｏｐｄ＿ｎｕｍ[ｉ]は、系列ｉにおける演算処理部ＯＰのレジスタＲＥＧが入力元であるオペランドの数を示す変数である。ｅｘｅｃ＿ｔｙｐｅ[ｉ][ｊ]は、系列ｉにおける第ｊ演算の種類を示す変数である。ｏｐｄ０[ｉ][ｊ]は、系列ｉにおける第ｊ演算のオペランド０の識別番号を示す変数である。ｏｐｄ１[ｉ][ｊ]は、系列ｉにおける第ｊ演算のオペランド１の識別番号を示す変数である。ｆｉｎ０[ｉ][ｊ]は、系列ｉにおける第ｊ演算のオペランド０の格納場所を示す変数である。ｆｉｎ１[ｉ][ｊ]は、系列ｉにおける第ｊ演算のオペランド１の格納場所を示す変数である。ｆｏｕｔ[ｉ][ｊ]は、系列ｉにおける第ｊ演算の演算結果データの格納場所を示す変数である。

【００６４】

ｖｒは、レジスタＲＥＧ割り当て用の一時変数である。ｖａは、演算器ＯＰＡ割り当て用の一時変数である。ｖｂは、演算器ＯＰＢ割り当て用の一時変数である。ｖｃは、演算器ＯＰＣ割り当て用の一時変数である。ｏｆｓｔ＿ｒは、レジスタＲＥＧ割り当て用のオフセットを示す変数である。ｏｆｓｔ＿ａは、演算器ＯＰＡ割り当て用のオフセットを示す変数である。ｏｆｓｔ＿ｂは、演算器ＯＰＢ割り当て用のオフセットを示す変数である。ｏｆｓｔ＿ｃは、演算器ＯＰＣ割り当て用のオフセットを示す変数である。ｏｆｓｔ＿ｒ＿ｌ[ｉ]は、前回の処理サイクルにおける系列ｉのレジスタＲＥＧ割り当て用のオフセットを示す変数である。ｓｔａｄ＿ｆｉｎ[ｉ]は、系列ｉにおける入力ＦＩＦＯ部ＩＦｉの読み出し開始アドレスを示す変数である。ｓｔａｄ＿ｆｏｕｔ[ｉ]は、系列ｉにおける出力ＦＩＦＯ部ＯＦｉの書き込み開始アドレスを示す変数である。ｐｒｉは、優先順位番

号を示す変数である。

【００６５】

リソース割当部ＲＡは、このような内部変数を使用して、図１７および図１８に示すステップＳ１～Ｓ３４を適宜実施する。

ステップＳ１において、リソース割当部ＲＡは、ｖｒ＝０、ｖａ＝０、ｖｂ＝０、ｖｃ＝０、ｏｆｓｔ＿ｒ＿ｌ[０]＝０、・・・、ｏｆｓｔ＿ｒ＿ｌ[ｎ]＝０に設定する。リソース割当部ＲＡは、例えば、演算処理装置ＯＰＤのリセット解除の直後に、この動作を実施する。これと同時に、リソース割当部ＲＡは、アクセス要求ＡＲ０～ＡＲｎを出力することで、系列０～ｎの最初に実行すべき演算を示す命令情報ＩＮＳ０～ＩＮＳｎを取得する。この後、リソース割当部ＲＡの動作はステップＳ２に移行する。

【００６６】

ステップＳ２において、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＝０、ｏｆｓｔ＿ａ＝０、ｏｆｓｔ＿ｂ＝０、ｏｆｓｔ＿ｃ＝０、ｐｒｉ＝０に設定する。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

ステップＳ３において、リソース割当部ＲＡは、ｐｒｉの値が系列数（ｎ＋１）以上であるか否かを判定する。ｐｒｉの値が系列数以上であるとき、リソース割当部ＲＡの動作はステップＳ３０に移行する。ｐｒｉの値が系列数未満であるとき、リソース割当部ＲＡの動作はステップＳ４に移行する。

【００６７】

ステップＳ４において、リソース割当部ＲＡは、優先順位情報ＰＩが示す系列０～ｎの優先順位に基づいて、処理対象の系列として第ｐｒｉ優先の系列ｉ（優先順位が”ｐｒｉ＋１”番目に高い系列ｉ）を選択する。この後、リソース割当部ＲＡの動作はステップＳ５に移行する。

ステップＳ５において、リソース割当部ＲＡは、残りのリソースの数（演算処理部ＯＰにおける未割り当てのレジスタＲＥＧ、演算器ＯＰＡ、演算器ＯＰＢおよび演算器ＯＰＣの数）が、必要なリソースの数（命令情報ＩＮＳｉが示す演算を実行するために必要なレジスタＲＥＧ、演算器ＯＰＡ、演算器ＯＰＢおよび演算器ＯＰＣの数）以上であるか否かを判定する。残りのリソースの数が必要なリソースの数以上であるとき、リソース割当部ＲＡの動作はステップＳ６に移行する。残りのリソースの数が必要なリソースの数未満であるとき、リソース割当部ＲＡの動作はステップＳ３１に移行する。

【００６８】

ステップＳ６において、リソース割当部ＲＡは、ｖｒの値がｅｘｅｃ＿ｎｕｍ[ｉ]の値未満であるか否かを判定する。ｖｒの値がｅｘｅｃ＿ｎｕｍ[ｉ]の値未満であるとき、リソース割当部ＲＡの動作はステップＳ７に移行する。ｖｒの値がｅｘｅｃ＿ｎｕｍ[ｉ]の値以上であるとき、リソース割当部ＲＡの動作はステップＳ２５に移行する。

ステップＳ７において、リソース割当部ＲＡは、ｆｉｎ０[ｉ][ｖｒ]の値が”０”であるか否かを判定する。ｆｉｎ０[ｉ][ｖｒ]の値が”０”であるとき、リソース割当部ＲＡの動作はステップＳ８に移行する。ｆｉｎ０[ｉ][ｖｒ]の値が”１”であるとき、リソース割当部ＲＡの動作はステップＳ９に移行する。

【００６９】

ステップＳ８において、リソース割当部ＲＡは、第ｖｒ演算のオペランド０の入力元として、演算処理部ＯＰのレジスタＲＥＧ（ｏｆｓｔ＿ｒ＿ｌ[ｉ]＋ｏｐｄ０[ｉ][ｖｒ]）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１０に移行する。

ステップＳ９において、リソース割当部ＲＡは、第ｖｒ演算のオペランド０の入力元として、入力バッファ部ＩＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ０[ｉ][ｖｒ]）を割り当てる。ここで、入力バッファ部ＩＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ０[ｉ][ｖｒ]）は、入力ＦＩＦＯ部ＩＦｉの”ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ０[ｉ][ｖｒ]”番地を意味している。この後、リソース割当部ＲＡの動作はステップＳ１０に移行する。

【００７０】

ステップＳ１０において、リソース割当部ＲＡは、ｆｉｎ１[ｉ][ｖｒ]の値が”０”であるか否かを判定する。ｆｉｎ１[ｉ][ｖｒ]の値が”０”であるとき、リソース割当部ＲＡの動作はステップＳ１１に移行する。ｆｉｎ１[ｉ][ｖｒ]の値が”１”であるとき、リソース割当部ＲＡの動作はステップＳ１２に移行する。

ステップＳ１１において、リソース割当部ＲＡは、第ｖｒ演算のオペランド１の入力元として、演算処理部ＯＰのレジスタＲＥＧ（ｏｆｓｔ＿ｒ＿ｌ[ｉ]＋ｏｐｄ１[ｉ][ｖｒ]）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１３に移行する。

【００７１】

ステップＳ１２において、リソース割当部ＲＡは、第ｖｒ演算のオペランド１の入力元として、入力バッファ部ＩＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ１[ｉ][ｖｒ]）を割り当てる。ここで、入力バッファ部ＩＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ１[ｉ][ｖｒ]）は、入力ＦＩＦＯ部ＩＦｉの”ｓｔａｄ＿ｆｉｎ[ｉ]＋ｏｐｄ１[ｉ][ｖｒ]”番地を意味している。この後、リソース割当部ＲＡの動作はステップＳ１３に移行する。

【００７２】

ステップＳ１３において、リソース割当部ＲＡは、ｅｘｅｃ＿ｔｙｐｅ[ｉ][ｖｒ]に基づいて、系列ｉにおける第ｖｒ演算の種類が”転送”であるか否かを判定する。第ｖｒ演算の種別が”転送”であるとき、リソース割当部ＲＡの動作はステップＳ２１に移行する。第ｖｒ演算の種別が”転送”ではないとき、リソース割当部ＲＡの動作はステップＳ１４に移行する。

【００７３】

ステップＳ１４において、リソース割当部ＲＡは、ｅｘｅｃ＿ｔｙｐｅ[ｉ][ｖｒ]に基づいて、系列ｉにおける第ｖｒ演算の種別を判別する。第ｖｒ演算の種類が”演算Ａ”であるとき、リソース割当部ＲＡの動作はステップＳ１５に移行する。第ｖｒ演算の種類が”演算Ｂ”であるとき、リソース割当部ＲＡの動作はステップＳ１６に移行する。第ｖｒ演算の種類が”演算Ｃ”であるとき、リソース割当部ＲＡの動作はステップＳ１７に移行する。

【００７４】

ステップＳ１５において、リソース割当部ＲＡは、第ｖｒ演算の演算器として、演算器ＯＰＡ（ｏｆｓｔ＿ａ＋ｖａ）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１８に移行する。

ステップＳ１６において、リソース割当部ＲＡは、第ｖｒ演算の演算器として、演算器ＯＰＢ（ｏｆｓｔ＿ｂ＋ｖｂ）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１９に移行する。

【００７５】

ステップＳ１７において、リソース割当部ＲＡは、第ｖｒ演算の演算器として、演算器ＯＰＣ（ｏｆｓｔ＿ｃ＋ｖｃ）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ２０に移行する。

ステップＳ１８において、リソース割当部ＲＡは、ｖａ＝ｖａ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ２１に移行する。

【００７６】

ステップＳ１９において、リソース割当部ＲＡは、ｖｂ＝ｖｂ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ２１に移行する。

ステップＳ２０において、リソース割当部ＲＡは、ｖｃ＝ｖｃ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ２１に移行する。

ステップＳ２１において、リソース割当部ＲＡは、ｆｏｕｔ[ｉ][ｖｒ]の値が”０”であるか否かを判定する。ｆｏｕｔ[ｉ][ｖｒ]の値が”０”であるとき、リソース割当部ＲＡの動作はステップＳ２２に移行する。ｆｏｕｔ[ｉ][ｖｒ]の値が”１”であるとき、リソース割当部ＲＡの動作はステップＳ２３に移行する。

【００７７】

ステップＳ２２において、リソース割当部ＲＡは、系列ｉにおける第ｖｒ演算の演算結果データの出力先として、演算処理部ＯＰのレジスタＲＥＧ（ｏｆｓｔ＿ｒ＋ｖｒ）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ２４に移行する。

ステップＳ２３において、リソース割当部ＲＡは、系列ｉにおける第ｖｒ演算の演算結果データの出力先として、出力バッファ部ＯＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｏｕｔ[ｉ]＋ｖｒ）を割り当てる。ここで、出力バッファ部ＯＢｉのレジスタＲＥＧ（ｓｔａｄ＿ｆｏｕｔ[ｉ]＋ｖｒ）は、出力ＦＩＦＯ部ＯＦｉの”ｓｔａｄ＿ｆｏｕｔ[ｉ]＋ｖｒ”番地を意味している。この後、リソース割当部ＲＡの動作はステップＳ２４に移行する。

【００７８】

ステップＳ２４において、リソース割当部ＲＡは、ｖｒ＝ｖｒ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ６に移行する。

ステップＳ２５において、リソース割当部ＲＡは、系列ｉの命令情報を更新するために、メモリアクセス部ＭＡｉへのアクセス要求ＡＲｉを出力する。これにより、メモリアクセス部ＭＡｉにおいて、プログラムカウンタＰＣｉのカウンタ値ＰＣＶｉが更新され、命令情報メモリＩＮＳＭｉへのリードコマンドＲＣＭＭｉおよびリードアドレスＲＡＤＭｉが出力される。従って、次の処理サイクル用の命令情報ＩＮＳｉが命令情報メモリＩＩＭｉからリソース割当部ＲＡに供給される。この後、リソース割当部ＲＡの動作はステップＳ２６に移行する。

【００７９】

ステップＳ２６において、リソース割当部ＲＡは、今回の処理サイクルにおける系列ｉのレジスタ割り当て用のオフセットを保存するために、ｏｆｓｔ＿ｒ＿ｌ[ｉ]＝ｏｆｓｔ＿ｒに設定する。この後、リソース割当部ＲＡの動作はステップＳ２７に移行する。

ステップＳ２７において、リソース割当部ＲＡは、レジスタＲＥＧ割り当て用のオフセット、演算器ＯＰＡ割り当て用のオフセット、演算器ＯＰＢ割り当て用のオフセットおよび演算器ＯＰＣ割り当て用のオフセットを更新するために、ｏｆｓｔ＿ｒ＝ｏｆｓｔ＿ｒ＋ｖｒ、ｏｆｓｔ＿ａ＝ｏｆｓｔ＿ａ＋ｖａ、ｏｆｓｔ＿ｂ＝ｏｆｓｔ＿ｂ＋ｖｂ、ｏｆｓｔ＿ｃ＝ｏｆｓｔ＿ｃ＋ｖｃに設定する。この後、リソース割当部ＲＡの動作はステップＳ２８に移行する。

【００８０】

ステップＳ２８において、リソース割当部ＲＡは、ｖｒ＝０、ｖａ＝０、ｖｂ＝０、ｖｃ＝０に設定する。この後、リソース割当部ＲＡの動作はステップＳ２９に移行する。

ステップＳ２９において、リソース割当部ＲＡは、ｐｒｉ＝ｐｒｉ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

ステップＳ３０において、リソース割当部ＲＡは、演算処理部ＯＰにより演算を実行するために、これまでのリソース割り当て結果を示す接続情報ＣＩを生成して演算処理部ＯＰに出力する。この際、オペランド０、１の入力元として入力ＦＩＦＯ部が割り当てられた演算が存在する場合、リソース割当部ＲＡは、対応する入力ＦＩＦＯ部への入力許可通知を出力する。また、演算結果データの出力先として出力ＦＩＦＯ部が割り当てられた演算が存在する場合、リソース割当部ＲＡは、対応する出力ＦＩＦＯ部に出力許可通知を出力する。この後、リソース割当部ＲＡの動作はステップＳ２に移行する。

【００８１】

ステップＳ３１において、リソース割当部ＲＡは、残りのレジスタＲＥＧの数（演算処理部ＯＰにおける未割り当てのレジスタＲＥＧの数）がｏｐｄ＿ｎｕｍ[ｉ]の値未満であるか否かを判定する。残りのレジスタＲＥＧの数がｏｐｄ＿ｎｕｍ[ｉ]の値未満であるとき、リソース割当部ＲＡの動作はステップＳ２６に移行する。残りのレジスタＲＥＧの数がｏｐｄ＿ｎｕｍ[ｉ]の値以上であるとき、リソース割当部ＲＡの動作はステップＳ３２に移行する。

【００８２】

ステップＳ３２において、リソース割当部ＲＡは、ｖｒの値がｏｐｄ＿ｎｕｍ[ｉ]の値未満であるか否かを判定する。ｖｒの値がｏｐｄ＿ｎｕｍ[ｉ]の値未満であるとき、リソ

ース割当部ＲＡの動作はステップＳ３３に移行する。ｖｒの値がｏｐｄ＿ｎｕｍ[ｉ]の値以上であるとき、リソース割当部ＲＡの動作はステップＳ２６に移行する。

ステップＳ３３において、リソース割当部ＲＡは、演算処理部ＯＰのレジスタＲＥＧ（ｏｆｓｔ＿ｒ＿ｌ[ｉ]＋ｖｒ）の退避先として、演算処理部ＯＰのレジスタＲＥＧ（ｏｆｓｔ＿ｒ＋ｖｒ）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ３４に移行する。

【００８３】

ステップＳ３４において、リソース割当部ＲＡは、ｖｒ＝ｖｒ＋１に設定する。この後、リソース割当部ＲＡの動作はステップＳ３２に移行する。

ここで、演算処理装置ＯＰＤの内部構成（演算処理部ＯＰにおけるレジスタＲＥＧの数、演算器ＯＰＡの数、演算器ＯＰＢの数、演算器ＯＰＣの数）を決定する方法について、具体例を用いて説明する。この方法は、演算処理装置ＯＰＤの回路記述（ＲＴＬ記述など）を用いた機能シミュレーションを利用して実施される。なお、機能シミュレーションで用いる演算処理装置ＯＰＤの回路記述において、レジスタ群ＲＧＲにおけるレジスタＲＥＧの数と、演算器群ＯＧＲにおける演算器ＯＰＡの数、演算器ＯＰＢの数および演算器ＯＰＣの数と、メモリ群ＭＧＲにおけるメモリＭＥＭの数およびメモリＭＥＭの容量と、入力ＦＩＦＯ部ＩＦｉの容量と、出力ＦＩＦＯ部ＯＦｉの容量と、入力優先レベル判定部ＩＬＤｉの閾値ＶＩ０ｉ～ＶＩ４ｉと、出力優先レベル判定部ＯＬＤｉの閾値ＶＯ０ｉ～ＶＯ４ｉとは、パラメータ化されている。

【００８４】

例えば、系列数が”２”である場合（ｎ＝１）について説明する。なお、命令情報メモリＩＮＳＭ０には、図１２に示した命令情報が格納されているものとする。命令情報メモリＩＮＳＭ１には、図１３に示した命令情報が格納されているものとする。入力ＦＩＦＯ部ＩＦ０において、”ｓｔａｄ＿ｆｉｎ[０]”番地に外部入力データ”ａ”が格納され、”ｓｔａｄ＿ｆｉｎ[０]＋１”番地に外部入力データ”ｂ”が格納され、”ｓｔａｄ＿ｆｉｎ[０]＋２”番地に外部入力データ”ｃ”が格納されているものとする。また、入力ＦＩＦＯ部ＩＦ１において、”ｓｔａｄ＿ｆｉｎ[１]”番地に外部入力データ”ｄ”が格納され、”ｓｔａｄ＿ｆｉｎ[１]＋１”番地に外部入力データ”ｅ”が格納され、”ｓｔａｄ＿ｆｉｎ[１]＋２”番地に外部入力データ”ｆ”が格納され、”ｓｔａｄ＿ｆｉｎ[１]＋３”番地に外部入力データ”ｇ”が格納されているものとする。この例における機能シミュレーションの実施中は、常に、第０優先の系列が系列０であり、第１優先の系列が系列１であるものとする。演算器ＯＰＡ、ＯＰＢ、ＯＰＣは、それぞれ加算器ＡＤＤ、減算器ＳＵＢ、乗算器ＭＵＬであるものとする。

【００８５】

まず、レジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数のパラメータ値を十分に大きな値（例えば、”１０２４”）に設定して、演算処理装置ＯＰＤの回路記述を用いた１回目の機能シミュレーションを実施する。

図１９は、１回目の機能シミュレーション（リソースの数を大きく設定した場合の機能シミュレーション）におけるリソース割当状態および内部変数状態を示している。１回目の機能シミュレーションにおいて、リソース割当部ＲＡは、以下のように動作する。

【００８６】

ステップＳ１において、リソース割当部ＲＡは、演算処理装置ＯＰＤのリセットが解除されると、ｖｒ＝０、ｖａ＝０、ｖｂ＝０、ｖｃ＝０、ｏｆｓｔ＿ｒ＿ｌ[０]＝０、ｏｆｓｔ＿ｒ＿ｌ[１]＝０に設定する。このとき、命令情報メモリＩＮＳ０からリソース割当部ＲＡに第ｔサイクルの命令情報（図１２）が供給されるとともに、命令情報メモリＩＮＳＭ１からリソース割当部ＲＡに第ｔサイクルの命令情報（図１３）が供給される。この後、リソース割当部ＲＡの動作はステップＳ２に移行する。

【００８７】

ステップＳ２において、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＝０、ｏｆｓｔ＿ａ＝０、ｏｆｓｔ＿ｂ＝０、ｏｆｓｔ＿ｃ＝０、ｐｒｉ＝０に設定する。この後、リソース割当

部ＲＡの動作はステップＳ３に移行する。
ステップＳ３において、ｐｒｉ＝０であり、かつ系列数が”２”であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ４に移行する。
【００８８】
ステップＳ４において、ｐｒｉ＝０であるため、リソース割当部ＲＡは、処理対象の系列として第０優先の系列０を選択する。この後、リソース割当部ＲＡの動作はステップＳ５に移行する。
ステップＳ５において、残りのリソースの数（レジスタＲＥＧ：１０２４個、加算器ＡＤＤ：１０２４個、減算器ＳＵＢ：１０２４個、乗算器ＭＵＬ：１０２４個）が、系列０の第ｔサイクルの命令情報が示す演算を実行するために必要なリソースの数（レジスタＲＥＧ：６個、加算器ＡＤＤ：３個、減算器ＳＵＢ：３個、乗算器ＭＵＬ：０個）より大きいため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ６へ移行する。
【００８９】
ステップＳ６において、ｖｒ＝０、ｅｘｅｃ＿ｎｕｍ[０]＝６であるため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ７に移行する。
ステップＳ７において、ｆｉｎ０[０][０]＝１であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ９に移行する。
【００９０】
ステップＳ９において、ｏｐｄ０[０][０]＝０であるため、リソース割当部ＲＡは、系列０における第０演算のオペランド０の入力元として、入力バッファ部ＩＦ０のレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[０]）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１０に移行する。
ステップＳ１０において、ｆｉｎ１[０][０]＝１であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ１２に移行する。
【００９１】
ステップＳ１２において、ｏｐｄ１[０][０]＝１であるため、リソース割当部ＲＡは、系列０における第０演算のオペランド１の入力元として、入力バッファ部ＩＢ０のレジスタＲＥＧ（ｓｔａｄ＿ｆｉｎ[０]＋１）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１３に移行する。
ステップＳ１３において、系列０における第０演算の種類は”加算”（ｅｘｅｃ＿ｔｙｐｅ[０][０]＝０）であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ１４に移行する。
【００９２】
ステップＳ１４において、リソース割当部ＲＡは、系列０における第０演算の種類を判別する。系列０における第０演算の種類は”加算”であるため、リソース割当部ＲＡの動作はステップＳ１５に移行する。
ステップＳ１５において、ｖａ＝０、ｏｆｓｔ＿ａ＝０であるため、リソース割当部ＲＡは、系列０における第０演算の演算器として、加算器ＡＤＤ（０）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ１８へ移行する。
【００９３】
ステップＳ１８において、リソース割当部ＲＡは、ｖａ＝１に設定する。この後、リソース割当部ＲＡの動作はステップＳ２１へ移行する。
ステップＳ２１において、ｆｏｕｔ[０][０]＝０であるため、リソース割当部ＲＡは、は、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ２２に移行する。
【００９４】
ステップＳ２２において、ｏｆｓｔ＿ｒ＝０、ｖｒ＝０であるため、リソース割当部Ｒ

Ａは、系列０における第０演算の演算結果データの出力先として、演算処理部ＯＰのレジスタＲＥＧ（０）を割り当てる。この後、リソース割当部ＲＡの動作はステップＳ２４に移行する。

ステップＳ２４において、リソース割当部ＲＡは、ｖｒ＝１に設定する。このときのリソース割当状態および内部変数状態は、図１９＜１＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ６に移行する。

【００９５】

ステップＳ６において、ｖｒ＝１、ｅｘｅｃ＿ｎｕｍ[０]＝６であるため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ７に移行する。そして、系列０の第ｔサイクルの第０演算に対する処理と同様に、系列０の第ｔサイクルの第１演算に対する処理が実施される。系列０の第ｔサイクルの第１演算に対する処理が系列０の第ｔサイクルの第０演算に対する処理と異なる点は、第ｔサイクルの第１演算の種類が”減算”であるため、ステップＳ１５、Ｓ１８の代わりにステップＳ１６、Ｓ１９が実施される点である。系列０の第ｔサイクルの第１演算に対する処理におけるステップＳ２４の実施後のリソース割当状態および内部変数状態は、図１９＜２＞に示すとおりである。以降、同様に、系列０の第ｔサイクルの第２演算～第５演算に対する処理が順次実施される。系列０の第ｔサイクルの第２演算～第５演算に対する処理におけるステップＳ２４の実施後のリソース割当状態および内部変数状態は、図１９＜３＞～図１９＜６＞に示すとおりである。系列０の第ｔサイクルの第５演算に対する処理におけるステップＳ２４の実施後に、リソース割当部ＲＡの動作はステップＳ６に移行する。

【００９６】

ステップＳ６において、ｖｒ＝６、ｅｘｅｃ＿ｎｕｍ[０]＝６であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ２５に移行する。

ステップＳ２５において、リソース割当部ＲＡは、系列０の命令情報を更新するために、アクセス要求ＡＲ０を出力する。これにより、命令情報メモリＩＮＳＭ０からリソース割当部ＲＡに、次の処理サイクル（第ｔ＋１サイクル）の命令情報が供給される。この後、リソース割当部ＲＡの動作はステップＳ２６に移行する。

【００９７】

ステップＳ２６において、ｏｆｓｔ＿ｒ＝０であるため、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＿ｌ[０]＝０に設定する。この後、リソース割当部ＲＡの動作はステップＳ２７に移行する。

ステップＳ２７において、ｖｒ＝６、ｖａ＝３、ｖｂ＝３、ｖｃ＝０、ｏｆｓｔ＿ｒ＝０、ｏｆｓｔ＿ａ＝０、ｏｆｓｔ＿ｂ＝０、ｏｆｓｔ＿ｃ＝０であるため、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＝６、ｏｆｓｔ＿ａ＝３、ｏｆｓｔ＿ｂ＝３、ｏｆｓｔ＿ｃ＝０に設定する。この後、リソース割当部ＲＡの動作はステップＳ２８に移行する。

【００９８】

ステップＳ２８において、リソース割当部ＲＡは、ｖｒ＝０、ｖａ＝０、ｖｂ＝０、ｖｃ＝０に設定する。この後、リソース割当部ＲＡの動作はステップＳ２９に移行する。

ステップＳ２９において、リソース割当部ＲＡは、ｐｒｉ＝１に設定する。このときの内部変数状態は、図１９＜７＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

【００９９】

ステップＳ３において、ｐｒｉ＝１、系列数＝２であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ４に移行する。

ステップＳ４において、ｐｒｉ＝１であるため、処理対象の系列として第１優先の系列１を選択する。この後、リソース割当部ＲＡの動作はステップＳ５に移行する。

ステップＳ５において、残りのリソースの数（レジスタＲＥＧ：１０１８個、加算器ＡＤＤ：１０２１個、減算器ＳＵＢ：１０２１個、乗算器ＭＵＬ：１０２４個）が、系列１の第ｔサイクルの命令情報が示す演算を実行するために必要なリソースの数（レジスタＲ

ＥＧ：２個、加算器ＡＤＤ：２個、減算器ＳＵＢ：０個、乗算器ＭＵＬ：０個）より大きいため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ６へ移行する。

【０１００】

ステップＳ６において、ｖｒ＝０、ｅｘｅｃ＿ｎｕｍ[１]＝２であるため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ７に移行する。以降、系列０の第ｔサイクルの第０演算～第５演算に対する処理と同様に、系列１の第ｔサイクルの第０演算および第１演算に対する処理が順次実施される。系列１の第ｔサイクルの第０演算および第１演算に対する処理におけるステップＳ２４の実施後のリソース割当状態および内部変数状態は、図１９＜８＞および図１９＜９＞に示すとおりである。系列１の第ｔサイクルの第１演算に対する処理におけるステップＳ２４の実施後に、リソース割当部ＲＡの動作はステップＳ６に移行する。

【０１０１】

ステップＳ６において、ｖｒ＝２、ｅｘｅｃ＿ｎｕｍ[１]＝２であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ２５に移行する。そして、系列０の場合と同様に、ステップＳ２５～２９が順次実施される。ステップＳ２９の実施後の内部変数状態は、図１９＜１０＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

【０１０２】

ステップＳ３において、ｐｒｉ＝２であり、かつ系列数が”２”であるため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ３０に移行する。

ステップＳ３０において、リソース割当部ＲＡは、これまでのリソース割り当て結果を示す接続情報ＣＩを出力するとともに、入力許可通知ＩＥＮ０、１を出力する。従って、演算処理部ＯＰにより系列０の第ｔサイクルの第０演算～第５演算および系列１の第ｔサイクルの第０演算および第１演算が第ｔサイクルで並列に実行される。これにより、第ｔサイクルの処理が完了する。この後、リソース割当部ＲＡの動作はステップＳ２に移行する。

【０１０３】

ステップＳ２において、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＝０、ｏｆｓｔ＿ａ＝０、ｏｆｓｔ＿ｂ＝０、ｏｆｓｔ＿ｃ＝０、ｐｒｉ＝０に設定する。このときの内部変数状態は、図１９＜１１＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。以降、第ｔサイクルの処理と同様に、第ｔ＋１サイクルの処理、第ｔ＋２サイクルの処理および第ｔ＋３サイクルの処理が順次実施される。これらの処理におけるリソース割当状態および内部変数状態は、図１９＜１２＞～＜２７＞に示すとおりである。これにより、１回目の機能シミュレーションが完了する。

【０１０４】

１回目のシミュレーションにおいては、レジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数のパラメータ値が”１０２４”であるのに対して、演算の実行に使用されたレジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数は、各処理サイクルにおいて、以下に示すとおりである。これらの数は、例えば、ステップＳ３０の実施毎に、ｏｆｓｔ＿ｒ、ｏｆｓｔ＿ａ、ｏｆｓｔ＿ｂ、ｏｆｓｔ＿ｃの値を記憶しておくことで取得できる。

（第ｔサイクル）　ＲＥＧ：８個、ＡＤＤ：５個、ＳＵＢ：３個、ＭＵＬ：０個
（第ｔ＋１サイクル）ＲＥＧ：４個、ＡＤＤ：１個、ＳＵＢ：０個、ＭＵＬ：３個
（第ｔ＋２サイクル）ＲＥＧ：３個、ＡＤＤ：１個、ＳＵＢ：０個、ＭＵＬ：１個
（第ｔ＋３サイクル）ＲＥＧ：１個、ＡＤＤ：１個、ＳＵＢ：０個、ＭＵＬ：０個

従って、１回目の機能シミュレーションにおけるリソース割り当てにより系列０、１の演算処理を実施するためには、演算処理部ＯＰにおいて、レジスタＲＥＧが８個、加算器ＡＤＤが５個、減算器ＳＵＢが３個、乗算器ＭＵＬが３個設けられていれば十分であるこ

とが分かる。しかしながら、このように、レジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数を、各処理サイクルで要求される数のピーク値に決定すると、演算処理装置ＯＰＤに要求されるスループットに対して、系列０、１の演算処理が必要以上に高速に実行される場合が多い。

【０１０５】

そこで、レジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数、乗算器ＭＵＬの数のパラメータ値を１回目のシミュレーションで演算の実行に使用された数より小さく設定して、演算処理装置ＯＰＤの回路記述を用いた２回目の機能シミュレーションを実施する。例えば、レジスタＲＥＧの数のパラメータ値を”８”に設定し、加算器ＡＤＤの数のパラメータ値を”３”に設定、減算器ＳＵＢの数のパラメータ値を”３”に設定し、乗算器ＭＵＬの数のパラメータ値を”３”に設定する。

【０１０６】

図２０は、２回目の機能シミュレーション（リソースの数を小さく設定した場合の機能シミュレーション）におけるリソース割当状態および内部変数状態を示している。２回目の機能シミュレーションにおいて、リソース割当部ＲＡは、以下のように動作する。

まず、１回目の機能シミュレーションと同様に、系列０の第ｔサイクルの命令情報（第０演算～第５演算）に対する処理が実施される。系列０の第ｔサイクルの命令情報に対する処理におけるステップＳ２４の実施後のリソース割当状態および内部変数状態は、図２０＜１＞～＜６＞に示すとおりである。系列０の第ｔサイクルの命令情報に対する処理におけるステップＳ２９の実施後の内部変数状態は、図２０＜７＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

【０１０７】

ステップＳ３において、ｐｒｉ＝１であり、かつ系列数が”２”であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ４に移行する。

ステップＳ４において、ｐｒｉ＝１であるため、リソース割当部ＲＡは、処理対象の系列として第１優先の系列１を選択する。この後、リソース割当部ＲＡの動作はステップＳ５に移行する。

【０１０８】

ステップＳ５において、残りのリソースの数（レジスタＲＥＧ：２個、加算器ＡＤＤ：０個、減算器ＳＵＢ：０個、乗算器ＭＵＬ：３個）が、系列１の第ｔサイクルの命令情報が示す演算を実行するために必要なリソースの数（レジスタＲＥＧ：２個、加算器ＡＤＤ：２個、減算器ＳＵＢ：０個、乗算器ＭＵＬ：０個）より小さい、すなわち、レジスタＲＥＧと減算器ＳＵＢと乗算器ＭＵＬとは足りるが、加算器ＡＤＤが足りないため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ３１に移行する。

【０１０９】

ステップＳ３１において、残リのレジスタＲＥＧの数が”２”であり、かつｏｐｄ＿ｎｕｍ[１][０]＝０であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当ＲＡの動作はステップＳ３２に移行する。

ステップＳ３２において、ｖｒ＝０、ｏｐｄ＿ｎｕｍ[１][０]＝０であるため、リソース割当部ＲＡは、”偽”と判定する。従って、リソース割当部ＲＡの動作はステップＳ２６に移行する。そして、ステップＳ２６～Ｓ２９が順次実施される。ステップＳ２９の実施後の内部変数状態は、図２０＜８＞に示すとおりである。このとき、ステップＳ２５が実施されないため、系列１の次の処理サイクル（第ｔ＋１サイクル）の命令情報は供給されない。従って、今回の処理サイクル（第ｔサイクル）の命令情報が示す演算が次の処理サイクル（第ｔ＋１サイクル）で実行されることになる。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。

【０１１０】

ステップＳ３において、ｐｒｉ＝２であり、かつ系列数が”２”であるため、リソース

割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ３０に移行する。

ステップＳ３０において、系列０の第ｔサイクルの命令情報が示す演算に対してのみリソースが割り当てられているため、系列０の第ｔサイクルの第０演算～第５演算のみが第ｔサイクルで並列に実行される。この後、リソース割当部ＲＡの動作はステップＳ２に移行する。

【０１１１】

ステップＳ２において、リソース割当部ＲＡは、ｏｆｓｔ＿ｒ＝０、ｏｆｓｔ＿ａ＝０、ｏｆｓｔ＿ｂ＝０、ｏｆｓｔ＿ｃ＝０、ｐｒｉ＝０に設定する。このときの内部変数状態は、図２０＜９＞に示すとおりである。この後、リソース割当部ＲＡの動作はステップＳ３に移行する。そして、系列０の第ｔ＋１サイクルの命令情報に対する処理が実施される。系列０の第ｔ＋１サイクルの命令情報に対する処理におけるリソース割当状態および内部変数状態は、図２０＜１０＞～＜１３＞に示すとおりである。系列０の第ｔ＋１サイクルの命令情報に対する処理におけるステップＳ２９が実施された後、リソース割当部ＲＡの動作はステップＳ５に移行する。

【０１１２】

ステップＳ５において、残りのリソースの数（レジスタＲＥＧ：５個、加算器ＡＤＤ：３個、減算器ＳＵＢ：３個、乗算器ＭＵＬ：０個）が、系列１の第ｔサイクルの命令情報が示す演算を実行するために必要なリソースの数（レジスタＲＥＧ：２個、加算器ＡＤＤ：２個、減算器ＳＵＢ：０個、乗算器ＭＵＬ：０個）より大きい、すなわち、全ての種類のリソースが足りるため、リソース割当部ＲＡは、”真”と判定する。従って、リソース割当部ＲＡの動作はステップＳ６へ移行する。そして、系列１の第ｔサイクルの命令情報に対する処理が実施される。系列１の第ｔサイクルの命令情報に対する処理におけるリソース割当状態および内部変数状態は、図２０＜１４＞～＜１７＞に示すとおりである。これにより、第ｔ＋１サイクルでは、系列０の第ｔ＋１サイクルの命令情報が示す演算が実行されるとともに、系列１の第ｔサイクルの命令情報が示す演算が実行される。以降、第ｔ＋２サイクルの処理および第ｔ＋３サイクルの処理が順次実施される。これらの処理におけるリソース割当状態および内部変数状態は、図２０＜１８＞～＜２８＞に示すとおりである。これにより、２回目の機能シミュレーションが完了する。

【０１１３】

２回目のシミュレーションにおいては、レジスタＲＥＧの数のパラメータ値が”８”であり、加算器ＡＤＤの数のパラメータ値が”３”であり、減算器ＳＵＢの数のパラメータ値が”３”であり、乗算器ＭＵＬの数のパラメータ値が”３”であるのに対して、演算の実行に使用されたレジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数は、各処理サイクルにおいて、以下に示すとおりである。

（第ｔサイクル）　ＲＥＧ：６個、ＡＤＤ：３個、ＳＵＢ：３個、ＭＵＬ：０個
（第ｔ＋１サイクル）ＲＥＧ：５個、ＡＤＤ：２個、ＳＵＢ：０個、ＭＵＬ：３個
（第ｔ＋２サイクル）ＲＥＧ：３個、ＡＤＤ：２個、ＳＵＢ：０個、ＭＵＬ：０個
（第ｔ＋３サイクル）ＲＥＧ：２個、ＡＤＤ：１個、ＳＵＢ：０個、ＭＵＬ：１個

従って、２回目の機能シミュレーションでのリソース割り当てにより系列０、１の演算処理を実施するためには、演算処理部ＯＰにおいて、レジスタＲＥＧが６個、加算器ＡＤＤが３個、減算器ＳＵＢが３個、乗算器ＭＵＬが３個設けられていれば十分であることが分かる。この例では、系列１の命令情報が示す演算が１サイクル後ろにずらされて実行される。しかしながら、系列１の演算処理が何回も呼び出されて連続して実行されるものであったとしても、系列１の出力バッファ部ＯＢ１に要求されるデータ出力タイミングが満たされていれば問題はない。すなわち、系列１において、１回の演算処理のデータが４サイクルに１回、入力ＦＩＦＯ部ＩＦ１に格納され、出力ＦＩＦＯ部ＯＦ１から出力されるような場合、３サイクルで処理される系列１の演算処理は、第ｔサイクル～第ｔ＋３サイクルのいずれか３サイクルで実施されれば、演算処理装置ＯＰＤのスループットは保たれる。

【０１１４】
この例においては、機能シミュレーションの実施中に調停部ＡＢＴからリソース割当部ＲＡに送られてくる優先順位情報ＰＩが、常に系列０の優先順位が系列１の優先順位より高いことを示す場合について説明したが、仮に系列１のリソース割り当てが次々に先送りにされると、系列１の入力データが溜まり、系列１の出力データが減ってくるので、やがて系列１の優先順位が系列０の優先順位より高い状態になる。すると、まず系列１のリソース割り当てが実施され、続いて系列０のリソース割り当てが実施される。このように、リソース割り当てがどちらかの系列に偏ることなく、どちらの系列のスループットも保たれるように制御される。

【０１１５】
この後、レジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数、乗算器ＭＵＬの数のパラメータ値を２回目のシミュレーションで演算の実行に使用された数より小さく設定して、演算処理装置ＯＰＤの回路記述を用いた３回目の機能シミュレーションを実施する。例えば、レジスタＲＥＧの数のパラメータ値を”８”に設定し、加算器ＡＤＤの数のパラメータ値を”２”に設定、減算器ＳＵＢの数のパラメータ値を”３”に設定し、乗算器ＭＵＬの数のパラメータ値を”３”に設定する。系列０の演算処理を実施するためには、演算処理部ＯＰに加算器ＡＤＤが少なくとも３個設けられている必要があるため、３回目の機能シミュレーションは、系列０の出力バッファ部ＯＢ０に要求されるデータ出力タイミングを満たせなくなり強制終了されることになる。

【０１１６】
従って、演算処理部ＯＰにおける最終的なレジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数を、２回目の機能シミュレーションで演算の実行に使用されたレジスタＲＥＧの数（６個）、加算器ＡＤＤの数（３個）、減算器ＳＵＢの数（３個）および乗算器ＭＵＬの数（３個）に決定する。これにより、演算処理装置ＯＰＤのスループットを低下させることなく、演算処理部ＯＰにおけるレジスタＲＥＧの数、加算器ＡＤＤの数、減算器ＳＵＢの数および乗算器ＭＵＬの数を最小限に抑えることができる。すなわち、演算処理装置ＯＰＤの処理性能を保ったまま回路面積を削減できる。なお、ここでは、系列数が”２”である場合について説明したが、系列数が大きくなるほど本発明は多大な効果を奏する。系列数が大きいほど演算処理部ＯＰのリソース（レジスタＲＥＧ、加算器ＡＤＤ、減算器ＳＵＢおよび乗算器ＭＵＬ）が演算に割り当てられる機会が多くなるため、非常に長い期間で考えれば、レジスタＲＥＧ、加算器ＡＤＤ、減算器ＳＵＢおよび乗算器ＭＵＬは、平均的に最大限にそれぞれ利用され、かつその数は最小限に抑えられる。

【０１１７】
図２１は、図６の演算処理装置の応用例を示している。マルチプロセッサ１００は、系列０～３の演算処理をそれぞれ担当するプロセッサコア１１０～１１３と、プロセッサコア１１０～１１３に共通して設けられる共有演算処理部１２０と、マルチプロセッサ１００の全体を制御するメインＣＰＵ１３０と、各種データを格納するメモリ１４０と、タイマ機能や通信インタフェース機能等を具現する周辺回路１５０と、外部とのデータ授受を実施するＩ／Ｏ回路１６０と、ネットワーク１７０とを備えて構成されている。ネットワーク１７０は、プロセッサコア１１０～１１３、メインＣＰＵ１３０、メモリ１４０、周辺回路１５０およびＩ／Ｏ回路１６０を相互に接続し、これらの間でのデータ授受を可能にする。なお、マルチプロセッサ１００では、プロセッサコア１１０～１１３、メインＣＰＵ１３０、メモリ１４０、周辺回路１５０およびＩ／Ｏ回路１６０がネットワーク１７０を介して接続されているが、これらがバスを介して接続されるようにしてもよい。

【０１１８】
共有演算処理部１２０は、図６の演算処理装置ＯＰＤ（ｎ＝３）により構成されている。共有演算処理部１２０において、入力バッファ部ＩＢ０～ＩＢ３（図６）へのライトコマンドＷＣＭＩ０～ＷＣＭＩ３および外部入力データＤＩ０～ＤＩ３と、出力バッファ部ＯＢ０～ＯＢ３（図６）へのリードコマンドＲＣＭＯ０～ＲＣＭＯ３とは、プロセッサコ

ア１１０～１１３からそれぞれ供給される。また、出力バッファ部ＯＢ０～ＯＢ３からの外部出力データＤＯ０～ＤＯ３は、プロセッサコア１１０～１１３にそれぞれ供給される。

【０１１９】

このような構成のマルチプロセッサ１００では、プロセッサコア１１０～１１３の各々は、自身が担当する演算処理のうち、高速性が要求される演算処理を共有演算処理部１２０に実施させる。共有演算処理部１２０は、プロセッサコア１１０～１１３からの演算処理要求に従って、プロセッサコア１００～１１３に代わって演算処理を実施する。その際、共有演算処理部１２０において、プロセッサコア１１０～１１３からの演算処理要求の優先順位に従って演算処理が実施され、かつプロセッサコア１１０～１１３の全てについて、要求される処理速度に遅延が発生しないようにバランスがとられるため、処理性能の面およびリソースの使用効率の面でマルチプロセッサ１００の全体として高い最適化が図られる。

【０１２０】

また、マルチプロセッサ１００では、プロセッサコア１１０～１１３にアプリケーションをそれぞれ実装する際、プロセッサコア１１０～１１３の各々におけるソフトウェア設計において、演算処理用のリソースの数が非常に大きいものとして扱える。そして、リソースの使用効率の最適化が共有演算処理部１２０により実施されるため、従来のマルチプロセッサにおけるアプリケーションの実装に比べて、ソフトウェア設計が非常に容易になる。

【０１２１】

図２２は、マルチプロセッサにおけるリソースの使用効率を示している。図２２では、４個のプロセッサコアの各々について拡張演算処理部が設けられたリソース非共有型マルチプロセッサ（第１従来例）と、４個のプロセッサコアの各々についてプログラマブル拡張演算処理部が設けられたリソース非共有型マルチプロセッサ（第２従来例）と、４個のプロセッサコアに共通して共有演算処理部が設けられたリソース共有型マルチプロセッサ（図２１のマルチプロセッサ１００）とについて、演算処理用のリソースの使用効率を示している。図中、横軸はリソース数（リソース面積）を示し、縦軸は時間を示している。また、網掛されることなくＡ～Ｄが記載された部分は、あるアプリケーションの演算処理Ａ～Ｄに使用されているリソースに対応している。網掛されることなくＡ’～Ｄ’が記載された部分は、別のアプリケーションの演算処理Ａ’～Ｄ’に使用されているリソースに対応している。網掛された部分は、演算処理に使用されていないリソースに対応している。

【０１２２】

図２２を参照すると、リソース非共有型マルチプロセッサ（第１従来例）では、アプリケーションの非動作状態がそのまま演算処理用のリソースの未使用状態として表れるため、リソースの使用効率が非常に悪いことが分かる。また、リソース非共有型マルチプロセッサ（第２従来例）では、プロセッサコア毎に演算処理用のリソースが独立して設けられているため、２つのアプリケーションの演算処理が時分割多重方式で実施されても、マルチプロセッサの全体としてはリソースの未使用状態が多くなることが分かる。これに対して、リソース共有型マルチプロセッサ（図２１のマルチプロセッサ１００）では、プロセッサコアが演算処理用のリソース（図２１の共有演算処理部１２０）を共有するため、リソースの数が最小限に抑えられ、リソースが効率良く使用されることが分かる。

【０１２３】

なお、前述の実施形態では、演算に割り当てるリソースとしてメモリＭＥＭを利用しない例について述べたが、本発明はかかる実施形態に限定されるものではない。例えば、ある演算の演算結果データが長期間使用されない場合に、その演算の演算結果データの格納先としてレジスタＲＥＧではなくメモリＭＥＭを割り当てるようにしてもよい。また、前述の実施形態では、図１７および図１８に示すステップＳ２～Ｓ３４が同一の処理サイクルで実施される例について述べたが、本発明はかかる実施形態に限定されるものではない

。例えば、パイプライン方式を使用して処理の効率化を図ってもよい。

【０１２４】

前述の実施形態では、機能シミュレーションを実施する際に、メモリ群ＭＧＲにおけるメモリＭＥＭの数およびメモリＭＥＭの容量、入力ＦＩＦＯ部ＩＦｉの容量、出力ＦＩＦＯ部ＯＦｉの容量、入力優先レベル判定部ＩＬＤｉの閾値ＶＩ０ｉ～ＶＩ４ｉ、出力優先レベル判定部ＯＬＤｉの閾値ＶＯ０ｉ～ＶＯ４ｉのパラメータ値を変更しない例について述べたが、本発明はかかる実施形態に限定されるものではない。例えば、演算処理部ＯＰのリソースの数を更に削減するために、これらのパラメータ値を変更しながら機能シミュレーションを実施するようにしてもよい。

【０１２５】

以上の実施形態で説明した発明を整理して、付記として開示する。

（付記１）

複数のデータ系列に対応して設けられ、対応する外部入力データを格納して処理対象データとして出力する複数の入力バッファ部と、

前記データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する演算処理部と、

前記データ系列に対応して設けられ、対応する処理結果データを格納して外部出力データとして出力する複数の出力バッファ部と、

処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、命令情報が示す演算を実行可である場合に、演算に割り当てる前記演算処理部の構成要素を決定し、前記演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、前記出力バッファ部のデータ出力タイミングを満たしたうえで演算を実行すべき処理サイクルを次の処理サイクルに移行させる制御処理を所定順序で実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する制御部とを備えることを特徴とする演算処理装置。

（付記２）

付記１記載の演算処理装置において、

前記各入力バッファ部は、

対応する外部入力データをライト要求に応答して格納し、格納している外部入力データをリード要求に応答して処理対象データとして出力する第１データ蓄積部と、

前記第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を出力する第１優先度情報生成部とを備え、

前記各出力バッファ部は、

対応する処理結果データをライト要求に応答して格納し、格納している処理結果データをリード要求に応答して外部出力データとして出力する第２データ蓄積部と、

前記第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を出力する第２優先度情報生成部とを備え、

前記制御部は、

前記入力バッファ部から供給される複数の第１優先度情報と前記出力バッファ部から供給される複数の第２優先度情報とに応じて前記データ系列の優先順位を決定する優先順位判定部と、

処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、対応するデータ系列の前記優先順位判定部により決定された優先順位が高い順に前記制御処理を実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する構成要素割当部とを備えることを特徴とする演算処理装置。

（付記３）

付記２記載の演算処理装置において、

前記第１優先度情報生成部は、

前記第１データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第１データ蓄

積部のデータ出力動作に伴ってダウンカウントする第１カウンタと、
前記第１カウンタのカウンタ値を第１閾値と比較して優先度を決定し、決定した優先度を示す第１優先度情報を出力する第１優先度判定部とを備えることを特徴とする演算処理装置。
（付記４）
付記３記載の演算処理装置において、
前記第１優先度判定部は、前記第１閾値を設定するための第１閾値設定部を備えることを特徴とする演算処理装置。
（付記５）
付記２記載の演算処理装置において、
前記第２優先度情報生成部は、
前記第２データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第２データ蓄積部のデータ出力動作に伴ってダウンカウントする第２カウンタと、
前記第２カウンタのカウンタ値を第２閾値と比較して優先度を決定し、決定した優先度を示す第２優先度情報を出力する第２優先度判定部とを備えることを特徴とする演算処理装置。
（付記６）
付記５記載の演算処理装置において、
前記第２優先度判定部は、前記第２閾値を設定するための第２閾値設定部を備えることを特徴とする演算処理装置。
（付記７）
付記２記載の演算処理装置において、
前記優先順位判定部は、
前記データ系列に対応して設けられ、対応する入力バッファ部から供給される第１優先度情報、対応する出力バッファ部から供給される第２優先度情報、および優先度定義テーブルを参照して優先度を決定し、決定した優先度を示す第３優先度情報を出力する複数の第３優先度判定部と、
前記第３優先度判定部から供給される複数の第３優先度情報に応じて、前記データ系列の優先順位を決定する調停部とを備えることを特徴とする演算処理装置。
（付記８）
付記１記載の演算処理装置において、
前記演算処理部は、
構成要素として設けられる複数のレジスタ、複数の演算器および複数のメモリと、
前記接続情報に応じて、前記レジスタ、前記演算器および前記メモリの接続関係を変更する接続関係変更部とを備えることを特徴とする演算処理装置。
（付記９）
付記１記載の演算処理装置において、
前記制御部に供給される命令情報は、処理サイクルあたりの演算の数を示す情報を含むとともに、演算毎に、演算の種類を示す情報と演算対象データを識別するための情報とを含むことを特徴とする演算処理装置。
（付記１０）
複数のデータ系列に対応して設けられ、対応する外部入力データを格納して処理対象データとして出力する複数の入力バッファ部と、
前記データ系列毎に処理対象データに対して所望の演算処理を実施して処理結果データを出力するために、接続情報に応じて内部の構成要素の接続関係を確立することで演算処理機能を変更する演算処理部と、
前記データ系列に対応して設けられ、対応する処理結果データを格納して外部出力データとして出力する複数の出力バッファ部と、
処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、命令情報が示す演算を実行可である場合に、演算に割り当てる前記演算処理部の構成要素を決定し、前

記演算処理部の構成要素の不足により命令情報が示す演算を実行不可である場合に、演算を実行すべき処理サイクルを次の処理サイクルに移行させる制御処理を所定順序で実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する制御部とを備える演算処理装置の内部構成を決定する方法であって、

前記演算処理部の構成要素の数をパラメータ化した前記演算処理装置の回路記述を用いて、前記演算処理部の構成要素の数を、前記制御処理で常に演算実行可となる値に設定してシミュレーションを実施する第１工程と、

前記第１工程で実施したシミュレーションの結果から、そのシミュレーションで前記制御部により演算に割り当てられた前記演算処理部の構成要素の数を仮設計値として取得する第２工程と、

前記演算処理部の構成要素の数をパラメータ化した前記演算処理装置の回路記述を用いて、前記演算処理部の構成要素の数を、前記仮設計値より小さい値を初期値として順次減少させながら、前記出力バッファ部のデータ出力タイミングが満たされなくなるまでシミュレーションを繰り返し実施する第３工程と、

前記第３工程で実施したシミュレーションのうち、前記出力バッファ部のデータ出力タイミングが満たされた最後のシミュレーションの結果から、そのシミュレーションで前記制御部により演算に割り当てられた前記演算処理部の構成要素の数を実設計値として取得する第４工程とを含むことを特徴とする演算処理装置の内部構成決定方法。

（付記１１）

付記１０記載の演算処理装置の内部構成決定方法において、

前記演算処理部は、構成要素として設けられる複数のレジスタ、複数の演算器および複数のメモリと、前記接続情報に応じて、前記レジスタ、前記演算器および前記メモリの接続関係を変更する接続関係変更部とを備え、

前記第１および第３工程において、前記演算処理装置の構成要素の数として、前記レジスタの数、前記演算器の数および前記メモリの数をパラメータ化した前記演算処理装置の回路記述を用いてシミュレーションを実施することを特徴とする演算処理装置の内部構成決定方法。

（付記１２）

付記１１記載の演算処理装置の内部構成決定方法において、

前記第１および第３工程において、前記レジスタの数、前記演算器の数および前記メモリの数に加えて、前記各メモリの容量をパラメータ化した前記演算処理装置の回路記述を用いてシミュレーションを実施することを特徴とする演算処理装置の内部構成決定方法。

（付記１３）

付記１０記載の演算処理装置の内部構成決定方法において、

前記各入力バッファ部は、対応する外部入力データをライト要求に応答して格納し、格納している外部入力データをリード要求に応答して処理対象データとして出力する第１データ蓄積部と、前記第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を出力する第１優先度情報生成部とを備え、

前記各出力バッファ部は、対応する処理結果データをライト要求に応答して格納し、格納している処理結果データをリード要求に応答して外部出力データとして出力する第２データ蓄積部と、前記第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を出力する第２優先度情報生成部とを備え、

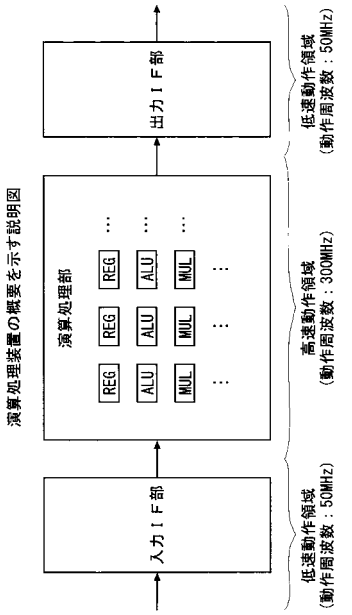
前記制御部は、前記入力バッファ部から供給される複数の第１優先度情報と前記出力バッファ部から供給される複数の第２優先度情報とに応じて前記データ系列の優先順位を決定する優先順位判定部と、処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、対応するデータ系列の前記優先順位判定部により決定された優先順位が高い順に前記制御処理を実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する構成要素割当部とを備え、

前記第１優先度情報生成部は、前記第１データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第１データ蓄積部のデータ出力動作に伴ってダウンカウントする第１カ

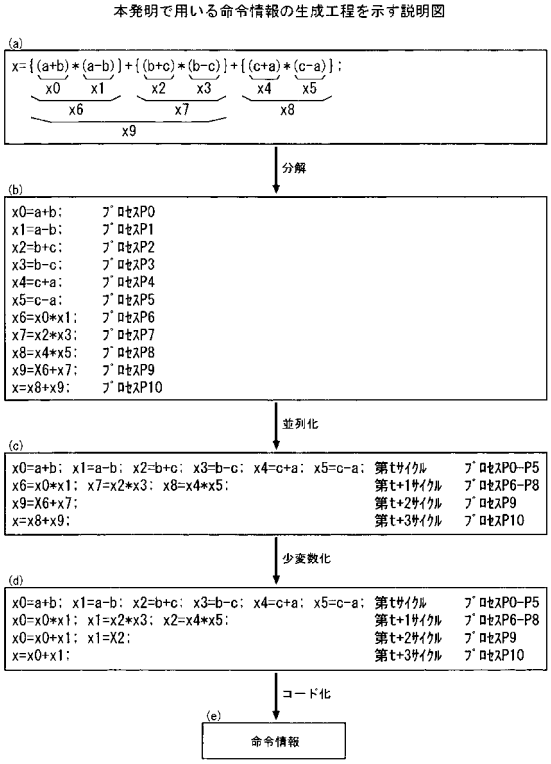
ウンタと、前記第１カウンタのカウンタ値を第１閾値と比較して優先度を決定し、決定した優先度を示す第１優先度情報を出力する第１優先度判定部とを備え、
　前記第１および第３工程において、前記演算処理部の構成要素の数に加えて、前記第１データ蓄積部の容量および前記第１閾値をパラメータ化した前記演算処理装置の回路記述を用いてシミュレーションを実施することを特徴とする演算処理装置の内部構成決定方法。
（付記１４）
　付記１０記載の演算処理装置の内部構成決定方法において、
　前記各入力バッファ部は、対応する外部入力データをライト要求に応答して格納し、格納している外部入力データをリード要求に応答して処理対象データとして出力する第１データ蓄積部と、前記第１データ蓄積部のデータ蓄積量が多いほど高い優先度を示す第１優先度情報を出力する第１優先度情報生成部とを備え、
　前記各出力バッファ部は、対応する処理結果データをライト要求に応答して格納し、格納している処理結果データをリード要求に応答して外部出力データとして出力する第２データ蓄積部と、前記第２データ蓄積部のデータ蓄積量が多いほど低い優先度を示す第２優先度情報を出力する第２優先度情報生成部とを備え、
　前記制御部は、前記入力バッファ部から供給される複数の第１優先度情報と前記出力バッファ部から供給される複数の第２優先度情報とに応じて前記データ系列の優先順位を決定する優先順位判定部と、処理サイクル毎に、前記データ系列に対応する複数の命令情報に対して、対応するデータ系列の前記優先順位判定部により決定された優先順位が高い順に前記制御処理を実施し、前記制御処理における前記演算処理部の構成要素の割り当て結果を前記接続情報として出力する構成要素割当部とを備え、
　前記第２優先度情報生成部は、前記第２データ蓄積部のデータ格納動作に伴ってアップカウントし、前記第２データ蓄積部のデータ出力動作に伴ってダウンカウントする第２カウンタと、前記第２カウンタのカウンタ値を第２閾値と比較して優先度を決定し、決定した優先度を示す第２優先度情報を出力する第２優先度判定部とを備え、
　前記第１および第３工程において、前記演算処理部の構成要素の数に加えて、前記第２データ蓄積部の容量および前記第２閾値をパラメータ化した前記演算処理装置の回路記述を用いてシミュレーションを実施することを特徴とする演算処理装置の内部構成決定方法。
（付記１５）
　複数のデータ系列に対応する複数のメイン演算処理部と、
　前記メイン演算処理部に共通して設けられ、前記メイン演算処理部に代わって演算処理を実施するサブ演算処理部とを備え、
　前記サブ演算処理部は、付記１～付記９のいずれかに記載の演算処理装置により構成されることを特徴とする演算処理システム。
【０１２６】
　以上、本発明について詳細に説明してきたが、前述の実施形態およびその変形例は発明の一例に過ぎず、本発明はこれらに限定されるものではない。本発明を逸脱しない範囲で変形可能であることは明らかである。
【図面の簡単な説明】
【０１２７】
【図１】演算処理装置の概要を示す説明図である。
【図２】本発明で用いる命令コードの生成工程を示す説明図である。
【図３】演算へのリソースの割り当ての一例を示す説明図である。
【図４】本発明における演算へのリソースの割り当ての基本概念を示す説明図（その１）である。
【図５】本発明における演算へのリソースの割り当ての基本概念を示す説明図（その２）である。
【図６】本発明の一実施形態を示すブロック図である。

【図７】図６の入力制御部の詳細を示すブロック図である。
【図８】図７の入力優先レベル判定部の動作例を示す説明図である。
【図９】図６の出力制御部の詳細を示すブロック図である。
【図１０】図９の出力優先レベル判定部の動作例を示す説明図である。
【図１１】図６の命令情報メモリに格納される命令情報の概要を示す説明図である。
【図１２】図６の命令情報メモリに格納される命令情報の具体例を示す説明図である。
【図１３】図６の命令情報メモリに格納される命令情報の別の具体例を示す説明図である。
【図１４】図６のシステム制御部の詳細を示すブロック図である。
【図１５】図１４の総合優先レベル判定部で用いられる優先レベル定義テーブルの具体例を示す説明図である。
【図１６】図１４のリソース割当部で用いられる内部変数を示す説明図である。
【図１７】図１４のリソース割当部の動作を示すフロー図（その１）である。
【図１８】図１４のリソース割当部の動作を示すフロー図（その２）である。
【図１９】リソースの数を大きく設定した場合の機能シミュレーションにおけるリソース割当状態および内部変数状態を示す説明図である。
【図２０】リソースの数を小さく設定した場合の機能シミュレーションにおけるリソース割当状態および内部変数状態を示す説明図である。
【図２１】図６の演算処理装置の応用例を示すブロック図である。
【図２２】マルチプロセッサにおけるリソースの使用効率を示す説明図である。
【符号の説明】
【０１２８】
ＡＢＴ‥調停部；ＣＩ０～ＣＩｎ‥カウンタ；ＣＯ０～ＣＯｎ‥カウンタ；ＩＢ０～ＩＢｎ‥入力バッファ部；ＩＣ０～ＩＣｎ‥入力制御部；ＩＦ０～ＩＦｎ‥入力ＦＩＦＯ部；ＩＬＤ０～ＩＬＤｎ‥入力優先レベル判定部；ＩＮＳＭ０～ＩＮＳＭｎ‥命令情報メモリ；ＭＡ０～ＭＡｎ‥メモリアクセス部；ＭＧＲ‥メモリ群；ＭＰＸ‥マルチプレクサ；ＯＢ０～ＯＢｎ‥出力バッファ部；ＯＣ０～ＯＣｎ‥出力制御部；ＯＦ０～ＯＦｎ‥出力ＦＩＦＯ部；ＯＧＲ‥演算器群；ＯＬＤ０～ＯＬＤｎ‥出力優先レベル判定部；ＯＰ‥演算処理部；ＯＰＤ‥演算処理装置；ＰＣ０～ＰＣｎ‥プログラムカウンタ；ＲＡ‥リソース割当部；ＲＣＩ０～ＲＣＩｎ‥リード制御部；ＲＣＭ０～ＲＣＭｎ‥リード制御部；ＲＣＯ０～ＲＣＯｎ‥リード制御部；ＲＧＲ‥レジスタ群；ＳＹＳＣ‥システム制御部；ＴＬＤ０～ＴＬＤｎ‥総合優先レベル判定部；ＴＲＩ０～ＴＲＩｎ‥閾値レジスタ；ＴＲＯ０～ＴＲＯｎ‥閾値レジスタ；ＷＣＩ０～ＷＣＩｎ‥ライト制御部；ＷＣＯ０～ＷＣＯｎ‥ライト制御部；１００‥マルチプロセッサ；１１０～１１３‥プロセッサコア；１２０‥共有演算処理部；１３０‥メインＣＰＵ；１４０‥メモリ；１５０‥周辺回路；１６０‥Ｉ／Ｏ回路；１７０‥ネットワーク

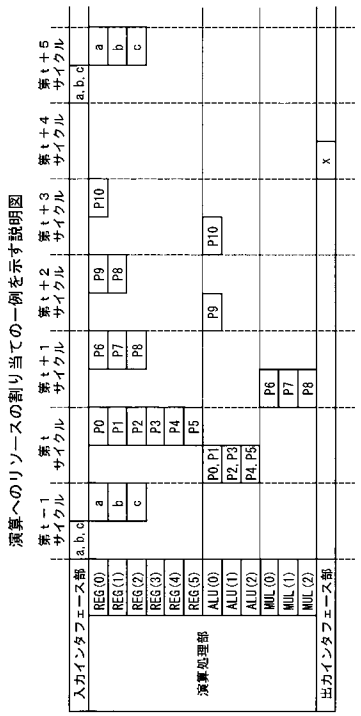
【図１】


演算処理装置の概要を示す説明図
出力ＩＦ部
演算処理部
REG
ALU
MUL
入力ＩＦ部
低速動作領域
（動作周波数：50MHz）
高速動作領域
（動作周波数：300MHz）
低速動作領域
（動作周波数：50MHz）

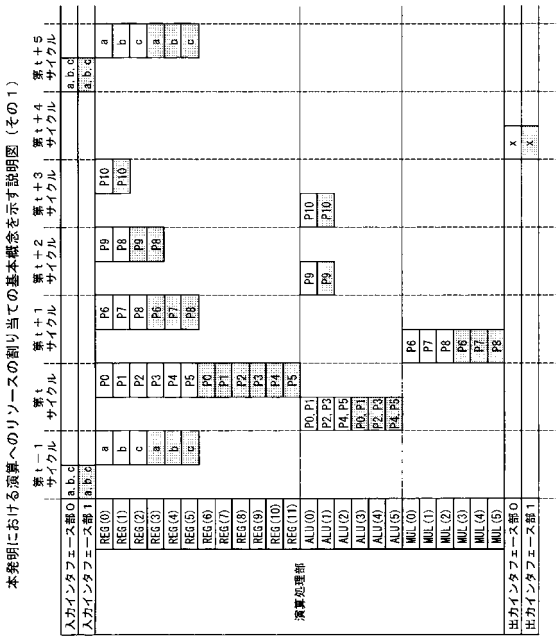
【図２】


本発明で用いる命令情報の生成工程を示す説明図
(a)
x={(a+b)*(a-b)}+{(b+c)*(b-c)}+{(c+a)*(c-a)};
x0 x1 x2 x3 x4 x5
x6 x7 x8
x9
分解
(b)
x0=a+b; プロセスP0
x1=a-b; プロセスP1
x2=b+c; プロセスP2
x3=b-c; プロセスP3
x4=c+a; プロセスP4
x5=c-a; プロセスP5
x6=x0*x1; プロセスP6
x7=x2*x3; プロセスP7
x8=x4*x5; プロセスP8
x9=X6+x7; プロセスP9
x=x8+x9; プロセスP10
並列化
(c)
x0=a+b; x1=a-b; x2=b+c; x3=b-c; x4=c+a; x5=c-a; 第tサイクル プロセスP0-P5
x6=x0*x1; x7=x2*x3; x8=x4*x5; 第t+1サイクル プロセスP6-P8
x9=X6+x7; 第t+2サイクル プロセスP9
x=x8+x9; 第t+3サイクル プロセスP10
少変数化
(d)
x0=a+b; x1=a-b; x2=b+c; x3=b-c; x4=c+a; x5=c-a; 第tサイクル プロセスP0-P5
x0=x0*x1; x1=x2*x3; x2=x4*x5; 第t+1サイクル プロセスP6-P8
x0=x0+x1; x1=X2; 第t+2サイクル プロセスP9
x=x0+x1; 第t+3サイクル プロセスP10
コード化
(e)
命令情報

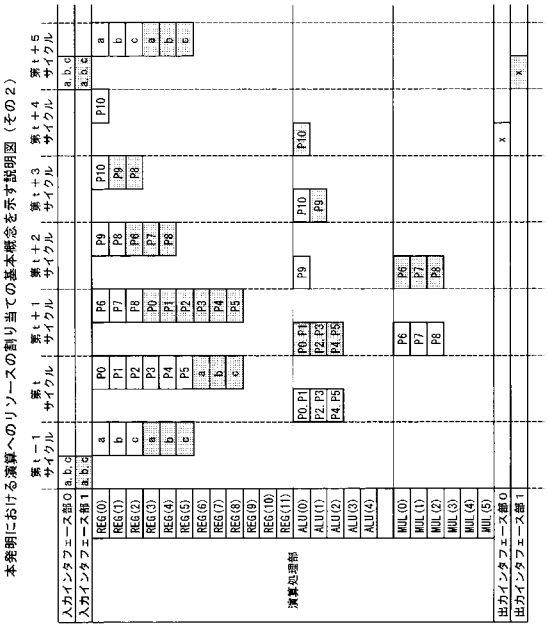
【図３】


演算へのリソースの割り当ての一例を示す説明図
第t－1サイクル
第tサイクル
第t＋1サイクル
第t＋2サイクル
第t＋3サイクル
第t＋4サイクル
第t＋5サイクル
入力インタフェース部
演算処理部
出力インタフェース部

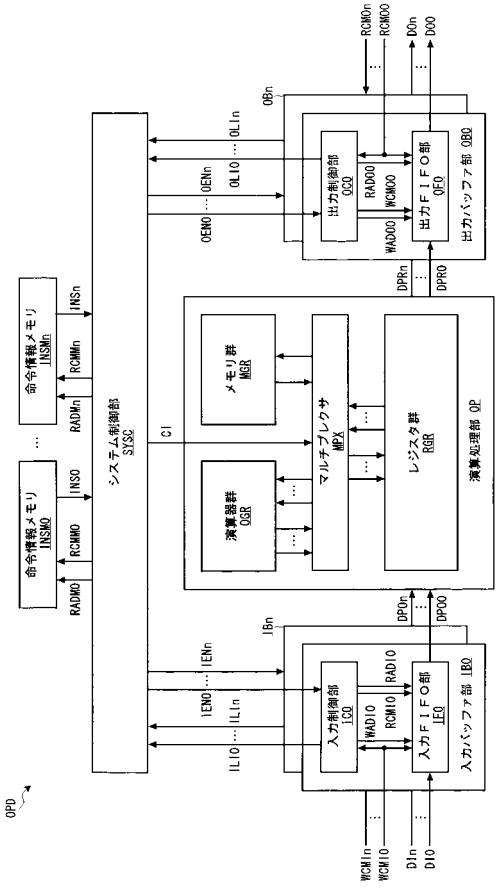
【図４】


本発明における演算へのリソースの割り当ての基本概念を示す説明図（その１）
第t－1サイクル
第tサイクル
第t＋1サイクル
第t＋2サイクル
第t＋3サイクル
第t＋4サイクル
第t＋5サイクル
入力インタフェース部0
入力インタフェース部1
演算処理部
出力インタフェース部0
出力インタフェース部1

【図５】


本発明における演算へのリソースの割り当ての基本概念を示す説明図（その２）
第ｔ－１サイクル
第ｔサイクル
第ｔ＋１サイクル
第ｔ＋２サイクル
第ｔ＋３サイクル
第ｔ＋４サイクル
第ｔ＋５サイクル
入力インタフェース部０
入力インタフェース部１
演算処理部
出力インタフェース部０
出力インタフェース部１

【図６】


本発明の一実施形態を示すブロック図
命令情報メモリ INSMO
命令情報メモリ INSMn
システム制御部 SYSC
演算器群 OGR
メモリ群 MGR
マルチプレクサ MPX
レジスタ群 RGR
演算処理部 OP
入力制御部 ICO
入力ＦＩＦＯ部 IFO
入力バッファ部 IBO
出力制御部 OCO
出力ＦＩＦＯ部 OFO
出力バッファ部 OBO

【図７】

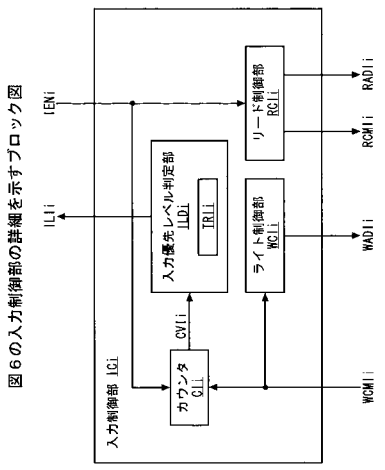


図６の入力制御部の詳細を示すブロック図
入力制御部 ICi
カウンタ CLi
入力優先レベル判定部 ILDi
ライト制御部 WCIi
リード制御部 RCIi

【図８】

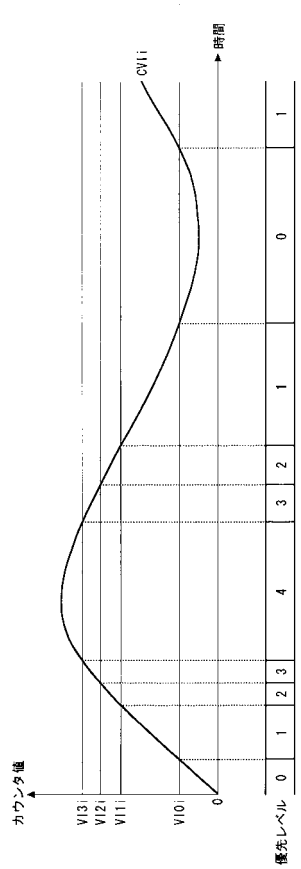
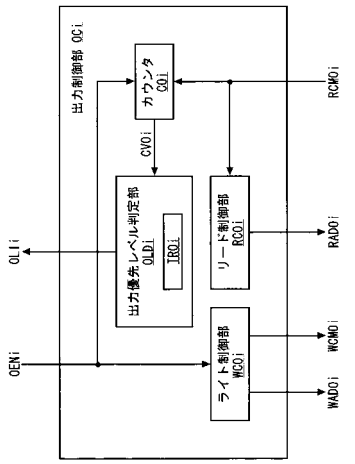


図７の入力優先レベル判定部の動作例を示す説明図
カウンタ値
時間
優先レベル

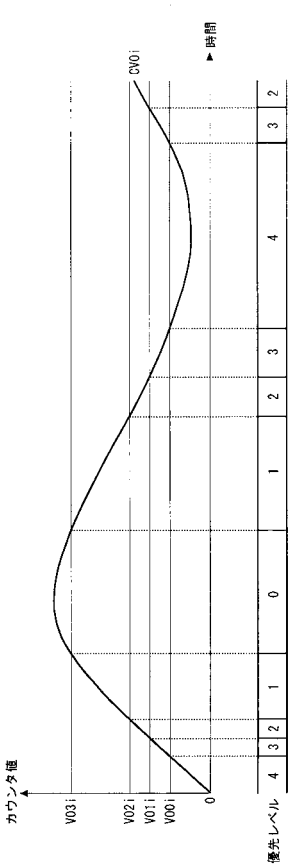
【図９】

図６の出力制御部の詳細を示すブロック図



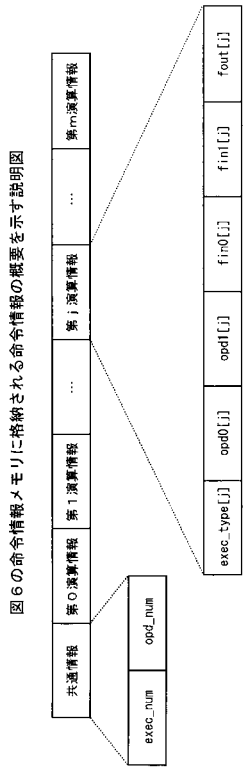
【図１０】

図９の出力優先レベル判定部の動作例を示す説明図



【図１１】

図６の命令情報メモリに格納される命令情報の概要を示す説明図



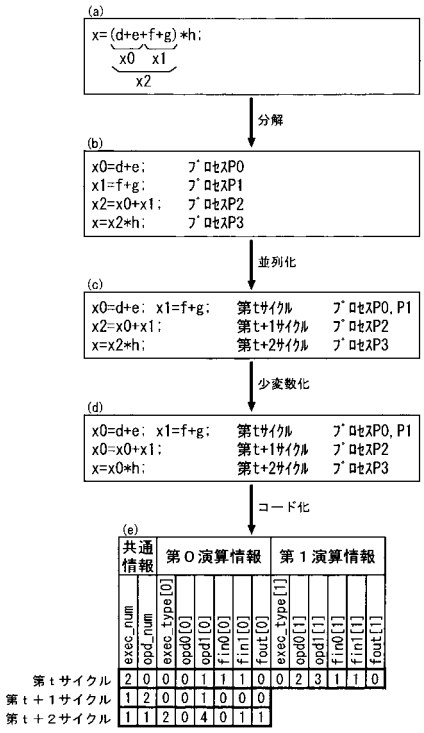
【図１２】

図６の命令情報メモリに格納される命令情報の具体例を示す説明図

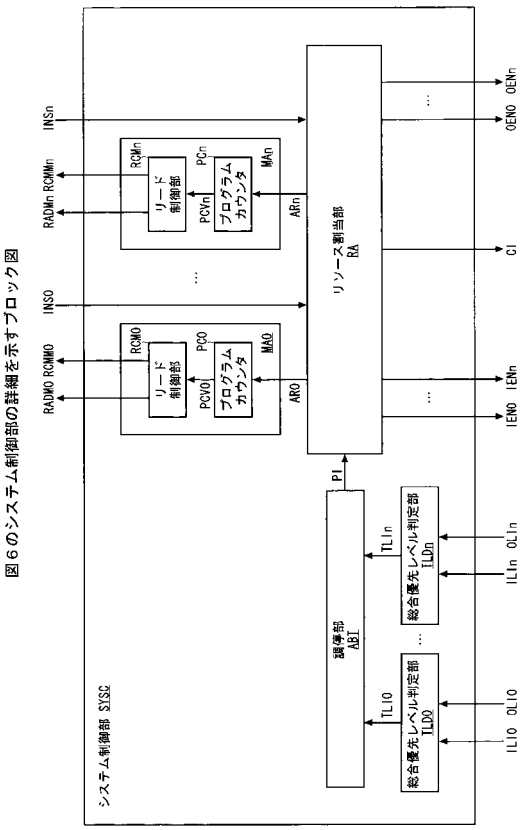
	共通情報	第０演算情報						第１演算情報						第２演算情報						第３演算情報						第４演算情報						第５演算情報					
	exec_num	exec_type[0]	opd_num	opd0[0]	opd1[0]	fin0[0]	fin1[0]	fout[0]	exec_type[1]	opd0[1]	opd1[1]	fin0[1]	fin1[1]	fout[1]	exec_type[2]	opd0[2]	opd1[2]	fin0[2]	fin1[2]	fout[2]	exec_type[3]	opd0[3]	opd1[3]	fin0[3]	fin1[3]	fout[3]	exec_type[4]	opd0[4]	opd1[4]	fin0[4]	fin1[4]	fout[4]	exec_type[5]	opd0[5]	opd1[5]	fin0[5]	fin1[5]	fout[5]
第ｔサイクル	6	0	0	1	1	1	1	0	1	0	1	1	1	0	0	1	2	1	1	0	1	1	2	1	1	0	0	2	0	1	1	0	1	2	0	1	1	0
第ｔ＋１サイクル	3	6	2	0	1	0	0	0	2	2	3	0	0	0	2	4	5	0	0	0																		
第ｔ＋２サイクル	2	3	0	0	1	0	0	0	3	2	*	0	*	0																								
第ｔ＋３サイクル	1	2	0	1	0	1																																

【図１３】

図６の命令情報メモリに格納される
命令情報の別の具体例を示す説明図



【図１４】



【図１５】

図１４の総合優先レベル判定部で用いられる
優先レベル定義テーブルの具体例を示す説明図

優先レベル (ILIi)	優先レベル (OLIi)	優先レベル (TLIi)
0	0	0
0	1	1
1	0	1
0	2	2
1	1	2
2	0	2
0	3	3
1	2	3
2	1	3
3	0	3
0	4	4
1	3	4
2	2	4
3	1	4
4	0	4
1	4	5
2	3	5
3	2	5
4	1	5
2	4	6
3	3	6
4	2	6
3	4	7
4	3	7
4	4	8

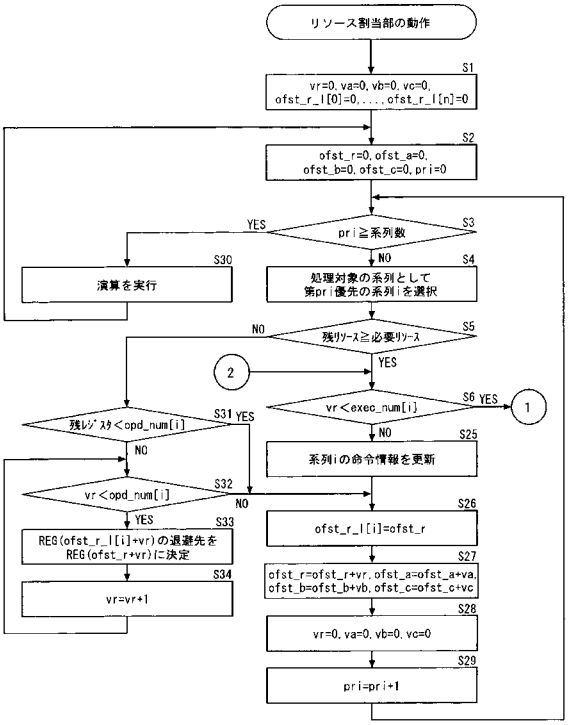
【図１６】

図１４のリソース割当部で用いられる内部変数を示す説明図

内部変数	意味
exec_num[i]	系列ｉにおける演算の数
opd_num[i]	系列ｉにおけるレジスタＲＥＧが入力元であるオペランドの数
exec_type[i][j]	系列ｉにおける第ｊ演算の種別
opd0[i][j]	系列ｉにおける第ｊ演算のオペランド０の識別番号
opd1[i][j]	系列ｉにおける第ｊ演算のオペランド１の識別番号
fin0[i][j]	系列ｉにおける第ｊ演算のオペランド０の格納場所
fin1[i][j]	系列ｉにおける第ｊ演算のオペランド１の格納場所
fout[i][j]	系列ｉにおける第ｊ演算結果データの格納場所
vr	レジスタＲＥＧ割り当て用の一時変数
va	演算器ＯＰＡ割り当て用の一時変数
vb	演算器ＯＰＢ割り当て用の一時変数
vc	演算器ＯＰＣ割り当て用の一時変数
ofst_r	レジスタＲＥＧ割り当て用のオフセット
ofst_a	演算器ＯＰＡ割り当て用のオフセット
ofst_b	演算器ＯＰＢ割り当て用のオフセット
ofst_c	演算器ＯＰＣ割り当て用のオフセット
ofst_r_i[i]	前回の処理サイクルでの系列ｉのレジスタＲＥＧ割り当て用オフセット
stad_fin[i]	系列ｉにおける入力ＦＩＦＯ部の読み出し開始アドレス
stad_fout[i]	系列ｉにおける出力ＦＩＦＯ部の書き込み開始アドレス
pri	優先順位番号

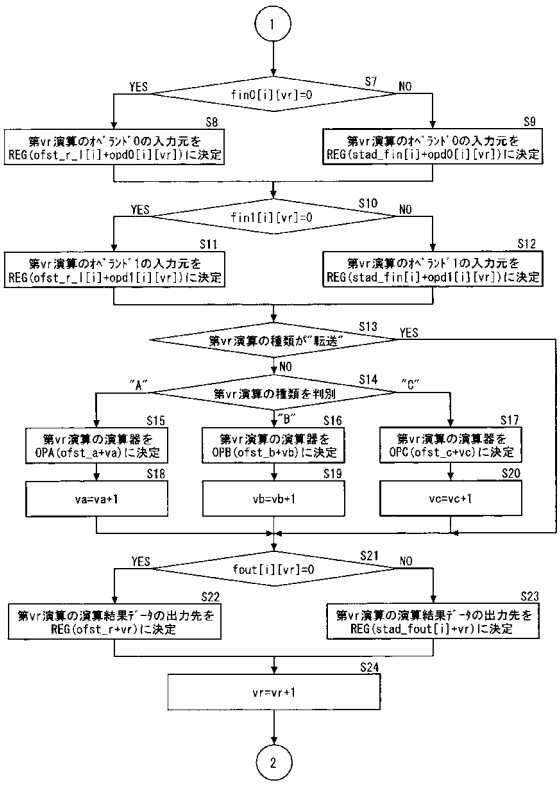
【図１７】

図１４のリソース割当部の動作を示すフロー図（その１）



【図１８】

図１４のリソース割当部の動作を示すフロー図（その２）



【図１９】

リソースの数を大きく設定した場合の機能シミュレーションにおけるリソース割当状態および内部変数状態を示す説明図

サイクル	系列	演算	オペランド0の入力元	オペランド1の入力元	演算器	演算結果の出力先	pri	vr	va	vb	vc	ofst_r	ofst_a	ofst_b	ofst_c	ofst_r_l[0]	ofst_r_l[1]	
t	0	0	REG(stad_fin[0])="a"	REG(stad_fin[0]+1)="b"	ADD(0)	REG(0)	0	1	1	0	0	0	0	0	0	0	0	<1>
		1	REG(stad_fin[0])="a"	REG(stad_fin[0]+1)="b"	SUB(0)	REG(1)	0	2	1	1	0	0	0	0	0	0	0	<2>
		2	REG(stad_fin[0]+1)="b"	REG(stad_fin[0]+2)="c"	ADD(1)	REG(2)	0	3	2	1	0	0	0	0	0	0	0	<3>
		3	REG(stad_fin[0]+1)="b"	REG(stad_fin[0]+2)="c"	SUB(1)	REG(3)	0	4	2	2	0	0	0	0	0	0	0	<4>
		4	REG(stad_fin[0]+2)="c"	REG(stad_fin[0])="a"	ADD(2)	REG(4)	0	5	3	2	0	0	0	0	0	0	0	<5>
		5	REG(stad_fin[0]+2)="c"	REG(stad_fin[0])="a"	SUB(2)	REG(5)	0	6	3	3	0	0	0	0	0	0	0	<6>
			系列0に対するステップS２９実施後				1	0	0	0	0	6	3	3	0	0	0	<7>
	1	0	REG(stad_fin[1])="d"	REG(stad_fin[1]+1)="e"	ADD(3)	REG(6)	1	1	1	0	0	6	3	3	0	0	0	<8>
		1	REG(stad_fin[1]+2)="f"	REG(stad_fin[1]+3)="g"	SUB(4)	REG(7)	1	2	2	0	0	6	3	3	0	0	0	<9>
			系列1に対するステップS２９実施後				2	0	0	0	0	0	0	0	0	0	0	<10>
t+1	0	0	REG(0)	REG(1)	MUL(0)	REG(0)	0	1	0	1	0	0	0	0	0	0	6	<11>
		1	REG(2)	REG(3)	MUL(1)	REG(1)	0	2	0	2	0	0	0	0	0	0	6	<12>
		2	REG(4)	REG(5)	MUL(2)	REG(2)	0	3	0	3	0	0	0	0	0	0	6	<13>
			系列0に対するステップS２９実施後				1	0	0	0	0	6	3	3	0	0	6	<14>
	1	0	REG(6)	REG(7)	ADD(3)	REG(3)	1	1	1	0	0	6	3	3	0	0	6	<15>
			系列1に対するステップS２９実施後				2	0	0	0	0	0	0	0	0	0	6	<16>
t+2	0	0	REG(0)	REG(1)	ADD(0)	REG(0)	0	1	1	0	0	6	3	3	0	0	3	<17>
		1	REG(2)	転送		REG(1)	0	2	1	1	0	0	0	0	0	0	3	<18>
			系列0に対するステップS２９実施後				1	0	0	0	0	0	0	0	0	0	3	<19>
	1	0	REG(3)	REG(4)	ADD(1)	REG(2)	1	1	1	0	0	6	3	3	0	0	3	<20>
			系列1に対するステップS２９実施後				2	0	0	0	0	0	0	0	0	0	3	<21>
t+3	0	0	REG(0)	REG(1)	ADD(0)	REG(stad_fout[0])="x"	0	1	1	0	0	6	3	3	0	0	2	<22>
			系列0に対するステップS２９実施後				1	0	0	0	0	0	0	0	0	0	2	<23>
	1	0	REG(2)	REG(3)	ADD(1)	REG(stad_fout[1])="y"	1	1	1	0	0	6	3	3	0	0	2	<24>
			系列1に対するステップS２９実施後				2	0	0	0	0	0	0	0	0	0	2	<25>

【図２０】

リソースの数を小さく設定した場合の機能シミュレーションにおけるリソース割当状態および内部変数状態を示す説明図

サイクル	系列	演算	オペランド0の入力元	オペランド1の入力元	演算器	演算結果の出力先	pri	vr	va	vb	vc	ofst_r	ofst_a	ofst_b	ofst_c	ofst_r_l[0]	ofst_r_l[1]	
t	0	0	REG(stad_fin[0])="a"	REG(stad_fin[0]+1)="b"	ADD(0)	REG(0)	0	1	1	0	0	0	0	0	0	0	0	<1>
		1	REG(stad_fin[0])="a"	REG(stad_fin[0]+1)="b"	SUB(0)	REG(1)	0	2	1	1	0	0	0	0	0	0	0	<2>
		2	REG(stad_fin[0]+1)="b"	REG(stad_fin[0]+2)="c"	ADD(1)	REG(2)	0	3	2	1	0	0	0	0	0	0	0	<3>
		3	REG(stad_fin[0]+1)="b"	REG(stad_fin[0]+2)="c"	SUB(1)	REG(3)	0	4	2	2	0	0	0	0	0	0	0	<4>
		4	REG(stad_fin[0]+2)="c"	REG(stad_fin[0])="a"	ADD(2)	REG(4)	0	5	3	2	0	0	0	0	0	0	0	<5>
		5	REG(stad_fin[0]+2)="c"	REG(stad_fin[0])="a"	SUB(2)	REG(5)	0	6	3	3	0	0	0	0	0	0	0	<6>
			系列0に対するステップS２９実施後				1	0	0	0	0	6	3	3	0	0	0	<7>
	1	0	REG(stad_fin[1])="d"	REG(stad_fin[1]+1)="e"	MUL(0)	REG(0)	0	1	0	1	0	0	0	0	0	0	6	<8>
		1	REG(stad_fin[1]+2)="f"	REG(stad_fin[1]+3)="g"	MUL(1)	REG(1)	0	2	0	2	0	0	0	0	0	0	6	<9>
			系列1に対するステップS２９実施後				1	0	0	0	0	0	0	0	0	0	6	<10>
t+1	0	0	REG(0)	REG(1)	MUL(0)	REG(0)	0	1	0	1	0	0	0	0	0	0	6	<11>
		1	REG(2)	REG(3)	MUL(1)	REG(1)	0	2	0	2	0	0	0	0	0	0	6	<12>
		2	REG(4)	REG(5)	MUL(2)	REG(2)	0	3	0	3	0	0	0	0	0	0	6	<13>
			系列0に対するステップS２９実施後				1	0	0	0	0	6	3	3	0	0	6	<14>
	1	0	REG(stad_fin[1])="d"	REG(stad_fin[1]+1)="e"	ADD(0)	REG(3)	1	1	1	0	0	3	0	3	0	0	6	<15>
		1	REG(stad_fin[1]+2)="f"	REG(stad_fin[1]+3)="g"	ADD(1)	REG(4)	1	2	2	0	0	3	0	3	0	0	6	<16>
			系列1に対するステップS２９実施後				2	0	0	0	0	5	2	3	0	0	3	<17>
t+2	0	0	REG(0)	REG(1)	ADD(0)	REG(0)	0	1	1	0	0	0	0	0	0	0	3	<18>
		1	REG(2)	—	転送	REG(1)	0	2	1	1	0	0	0	0	0	0	3	<19>
			系列0に対するステップS２９実施後				1	0	0	0	0	0	0	0	0	0	3	<20>
	1	0	REG(3)	REG(4)	ADD(1)	REG(2)	1	1	1	0	0	2	1	1	0	0	3	<21>
			系列1に対するステップS２９実施後				2	0	0	0	0	3	2	1	0	0	2	<22>
t+3	0	0	REG(0)	REG(1)	ADD(0)	REG(stad_fout[0])="x"	0	1	1	0	0	0	0	0	0	0	2	<23>
			系列0に対するステップS２９実施後				1	0	0	0	0	0	0	0	0	0	2	<24>
	1	0	REG(2)	REG(3)	MUL(0)	REG(stad_fout[1])="y"	1	1	1	0	0	2	1	1	0	0	2	<25>
			系列1に対するステップS２９実施後				2	0	0	0	0	3	2	1	0	0	1	<26>
																		<27>

【図２１】

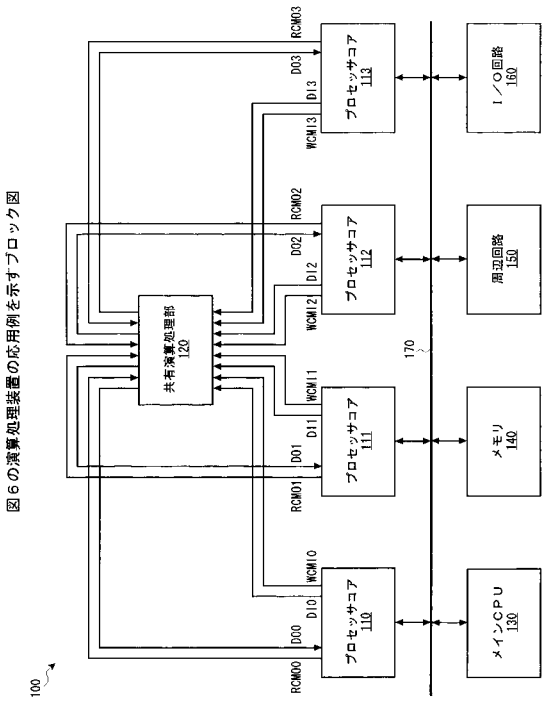
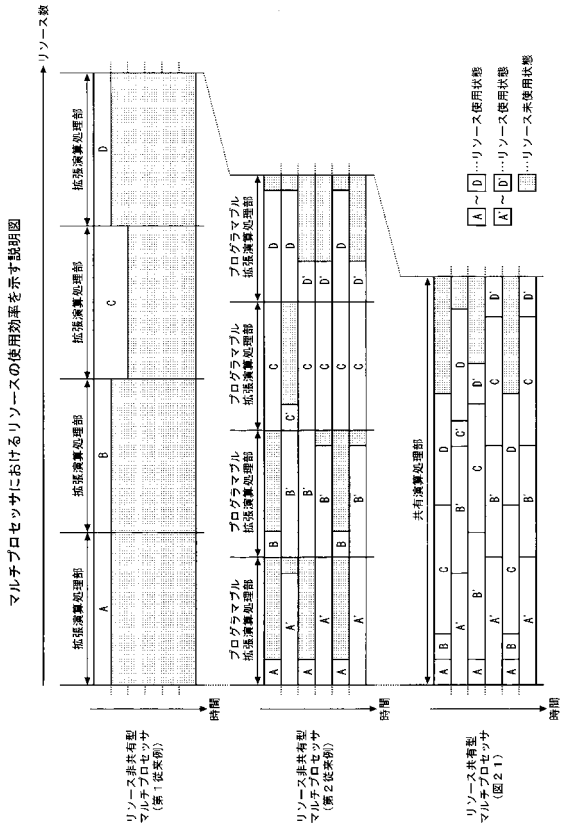


図６の演算処理装置の応用例を示すブロック図
100
共有演算処理部 120
プロセッサコア 110
プロセッサコア 111
プロセッサコア 112
プロセッサコア 113
メインCPU 130
メモリ 140
周辺回路 150
I／O回路 160
170
RCMO0
RCMO1
RCMO2
RCMO3
WCMI0
WCMI1
WCMI2
WCMI3
DO0
DO1
DO2
DO3
DI0
DI1
DI2
DI3

【図２２】


マルチプロセッサにおけるリソースの使用効率を示す説明図
リソース数
時間
拡張演算処理部
プログラマブル拡張演算処理部
共有演算処理部
A～D …リソース使用状態
A'～D' …リソース使用状態
…リソース未使用状態
リソース非共有型マルチプロセッサ（第１従来例）
リソース非共有型マルチプロセッサ（第２従来例）
リソース共有型マルチプロセッサ（図２１）