(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2005-190430
(P2005-190430A)
(43)公開日　平成17年7月14日(2005.7.14)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 9/38	G06F	9/38　370X	5B013

審査請求 有　請求項の数 3　OL　（全 12 頁）

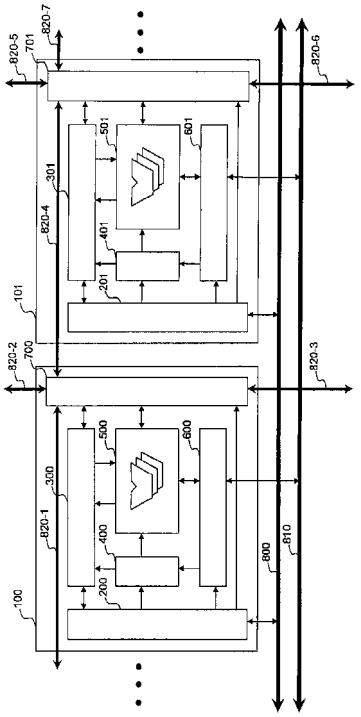
(21)出願番号　特願2003-434625 (P2003-434625)
(22)出願日　平成15年12月26日 (2003.12.26)

(71)出願人　304036743
国立大学法人宇都宮大学
栃木県宇都宮市峰町３５０番地
(74)代理人　100072051
弁理士　杉村　興作
(74)代理人　100100125
弁理士　高見　和明
(74)代理人　100101096
弁理士　徳永　博
(74)代理人　100086645
弁理士　岩佐　義幸
(74)代理人　100107227
弁理士　藤谷　史朗
(74)代理人　100114292
弁理士　来間　清志

最終頁に続く

(54)【発明の名称】自己最適化演算装置

(57)【要約】
【課題】その時々のプログラムの実行挙動に応じた最適化を動的に行うことで究極の最適化（高速化）を達成した自己最適化演算装置を提供する。
【解決手段】複数の単位処理ユニットを具える自己最適化演算装置において、各々の単位処理ユニットが、プログラムを実行する演算処理ユニット、実行中のプログラムの挙動を観測する観測処理ユニット、観測結果に基づいて最適化処理を行う最適化処理ユニット、及び、実行内容の変更など装置全体の資源管理を行う資源管理処理ユニットのうち少なくとも１つとして動作する。
【選択図】図１



【特許請求の範囲】
【請求項１】
複数の単位処理ユニットを具える自己最適化演算装置において、各々の単位処理ユニットが、プログラムを実行する演算処理ユニット、実行中のプログラムの挙動を観測する観測処理ユニット、観測結果に基づいて最適化処理を行う最適化処理ユニット、及び、実行内容の変更など装置全体の資源管理を行う資源管理処理ユニットのうち少なくとも１つとして動作することを特徴とする自己最適化演算装置。
【請求項２】
請求項１に記載の自己最適化演算装置において、前記単位処理ユニットの各々が、前記演算処理ユニットの実行状態および実行プログラムそのものを動的に変更できる機能を有し、前記最適化処理ユニットが、前記観測処理ユニットによって観測されたプログラムの挙動の観測結果をもとに実時間で最適なプログラムコードを生成し、前記演算処理ユニットの実行内容を動的に変更することを特徴とする自己最適化演算装置。
【請求項３】
請求項１又は２に記載の自己最適化演算装置において、プログラムの最適化状況に応じて、前記演算処理ユニット、観測処理ユニット、最適化処理ユニット及び資源管理処理ユニットの数の比率を変えることを特徴とする自己最適化演算装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、演算装置に関し、特に、複数の処理ユニットを具える自己最適化演算装置に関する。
【背景技術】
【０００２】
ひとつの計算機システム内に複数の処理ユニットを擁し、各処理ユニットにプログラムの実行局面に応じた役割を分担させることで、効果的な最適化処理とその結果による処理速度の向上を図ることができる。
【０００３】
第１の従来技術として、特開２００３－３００５０「マルチスレッド実行方法及び並列プロセッサシステム」に記載されているような、並列計算機、マルチスレッドプロセッサ技術がある。この技術は、複数の処理ユニットを用いて二種類の並列性を引き出すことにより高速化を実現する。具体的には、演算処理装置において複数の命令を同時に実行する命令レベル並列と、さらに命令列（スレッド）を単位として並列化するスレッドレベル並列である。これら二種の並列化の組み合わせにより高速化が実現される。並列計算機やマルチスレッド方式による演算処理装置では、内包する複数の処理装置を有効に使用し高速化を果たすために、命令レベル、および、スレッドレベル（ないし並列処理）の各レベルでの並列性を十分に引き出すことが必須である。しかし、一般のアプリケーションプログラムはこれらのレベルでの並列性を十分に引き出す意図では記述されていないために、コンパイラによる並列性抽出を十分に行えない問題がある。すなわち、複数の処理装置があっても、それらを同時並行的に稼動させ高速処理を実現すること、また、その高速処理を持続させることが難しいことが問題である。
【０００４】
第２の従来技術として、特開２００１－１４７８２０「コード最適化方法および記録媒体」に記載されているような、静的最適化、最適化コンパイラ技術がある。この技術は、プログラムとして記述された処理内容を論理的に解析し、上記二種の並列化技術（命令レベル並列、スレッドレベル並列）を適用することで高速化を実現する。いったんプログラムを実行しその時の挙動を記録（プロファイリング）することにより、最適化効果を向上するコンパイラ技術も用いられている。最適化コンパイラは、上述の並列性抽出の問題に応えようとするものであるが、一般的にコンパイルの時点で解析できる範囲は限られており、したがって最適化の効果が限られる問題がある。また、プロファイリングの結果をも

とに、より高度な最適化効果を得る方法も行われているが、収集されるプログラム実行挙動情報が、観測期間を通しての累積的な結果であるため、実行時間全体を通して平均的な高速化は可能であるが、細かな挙動の変化には追随できない問題がある。また、プログラムの処理内容が入力データの性質に依存する場合には、この技術による高速化効果が得られない問題がある。

【０００５】

第３の従来技術として、特開２００２－２２２０８８「コンパイルシステム、コンパイル方法およびプログラム」に記載されているような、動的最適化技術がある。プログラムの挙動に適応して、プログラム実行中に採取された情報をもとにプログラムコードを最適化する（ないし再コンパイルする）技術も存在する。プログラムの動的な挙動に追随した最適化を行うため、プログラム実行中の挙動を監視し、必要に応じてより適切なプログラムコードを生成する動的最適化がある。この技術では、本来のアプリケーションプログラムに挙動監視用の処理を追加するか、あるいは別個に監視用のプログラムを動作させる必要があり、いずれの場合でも監視の分だけ効率が低下する。さらに、実行途中に最適化処理を行うためのオーバーヘッドが課されるため、最適化による性能向上が相殺される問題がある。

【発明の開示】

【発明が解決しようとする課題】

【０００６】

プログラムの実行挙動に応じて計算機の内部構成やプログラムのコードを変えることにより性能を向上させることが望まれる。本発明は、最適化対象の計算機およびその上で実行されているプログラムを、同じ計算機システム中にありながら第三者的に観測することのできる機構を設置し、その時々のプログラムの実行挙動に応じた最適化を動的に行うことで究極の最適化（高速化）を達成した自己最適化演算装置を提供することを目的とする。本発明では、複数の演算器を持つ処理ユニットを複数個配置するシステムを前提とする。処理ユニット内では命令レベル並列性を抽出可能であり、また、複数の処理ユニットを用いることで並列処理ないしスレッドレベル並列性の抽出を可能にする。そしてマルチスレッド方式の演算処理装置に対して上に述べられた問題を解決し、効率よく動的に最適化を行うための自己最適化演算装置を実現するため、以下の方策をとる。

【課題を解決するための手段】

【０００７】

本発明の第１発明による自己最適化演算装置は、複数の単位処理ユニットを具え、各々の単位処理ユニットが、プログラムを実行する演算処理ユニット、実行中のプログラムの挙動を観測する観測処理ユニット、観測結果に基づいて最適化処理を行う最適化処理ユニット、及び、実行内容の変更など装置全体の資源管理を行う資源管理処理ユニットのうち少なくとも１つとして動作することを特徴とする。すなわち、本来目的とするアプリケーションプログラムの実行を担当する演算処理ユニット群の状況を、アプリケーションプログラムの実行を行わず挙動観測を行う観測処理ユニット群が観測し、その結果を使って最適化処理ユニット群が最適化を行い、全体の動作の管理・制御は資源管理処理ユニット群が司る。

【０００８】

本発明の第２発明による自己最適化演算装置は、前記単位処理ユニットの各々が、前記演算処理ユニットの実行状態および実行プログラムそのものを動的に変更できる機能を有し、前記最適化処理ユニットが、前記観測処理ユニットによって観測されたプログラムの挙動の観測結果をもとに実時間で最適なプログラムコードを生成し、前記演算処理ユニットの実行内容を動的に変更することを特徴とする。これにより、アプリケーションプログラムは常に効率の良い最適なコードで実行されることになる。

【０００９】

本発明の第３発明による自己最適化演算装置は、プログラムの最適化状況に応じて、前記演算処理ユニット、観測処理ユニット、最適化処理ユニット及び資源管理処理ユニット

の数の比率を変えることを特徴とする。最適化が進んでいない状況では観測や最適化処理に単位処理ユニットを多く割当てることで早期に実行効率を改善した最適化コードを得ることが可能になり、最適化時間を短縮する。最適化が進んだ段階では、それ以上の最適化の必要性が薄れるため、アプリケーションプログラムの実行を担当する演算処理ユニット群を多くすることで一層の処理速度の向上を得る。こうしてプログラムの実行局面に応じた最適な役割配分が行える。また、最適化処理が進んだ状況では、プログラムによっては、いったん最適な状態になってもそのまま最適状態が持続するとは限らないが、その場合は、観測処理ユニット群が挙動の変化を検知し、再び観測・最適化処理に多くの処理群ユニットを用いる変更を行うことで、プログラムの挙動変化に対して早期に対応し、早期に最適なプログラムコードを得ることができる。こうした動的な役割変更の処理は、資源管理処理ユニット群が行う。

【発明の効果】

【００１０】

プログラムの実行状況をリアルタイムに観察しながら最適化を行えるため、常にハードウェアの最大能力を出すための制御を行える効果がある。複数の均質な処理ユニットを用い、それらを上記最適化機能により常に最適な状態に保つことで、本発明で目的とする命令レベル並列およびスレッドレベル並列の最大限の抽出が可能になる効果がある。さらに、アプリケーションプログラムを担当する処理ユニットと、観測・最適化・資源管理の処理を行う処理ユニットとに役割分担すること、また、その役割分担を、最適化の状況により動的に変えられる機能を有することで、システム内にある処理ユニットの機能・能力を最大限に引き出せる効果がある。すなわち、最適化が進んでいない状況では、プログラム挙動の観測と最適化処理に注力することで早期に最適化コードを得ることが可能であり、また一方で、最適化が進んだ段階では本来のアプリケーションプログラムの実行に注力することで最大限の実行性能を達成することが可能になる。またさらに、演算処理のために使われることのない処理ユニットを観測・最適化・資源管理の機能に割当てることで、本来のアプリケーションの実行性能に全く影響を与えずに、動的最適化を行うことが可能になる。

【発明を実施するための最良の形態】

【００１１】

図１は、本発明による自己最適化演算装置の一実施例の構成を示すブロック図である。本自己最適化演算装置は、複数個の単位処理ユニット１００、１０１、．．．を具える。図１には、明瞭にするために、単位処理ユニット１００及び１０１のみを示す。これらの複数の処理ユニットは並列動作し、命令レベル並列性とスレッドレベル並列性の両方を引き出す。

【００１２】

代表的に、単位処理ユニット１００は、処理内容保持部４００と、演算処理部５００と、メモリ制御部６００と、ユニット間通信部７００と、プロファイル情報収集部３００と、ユニット制御部２００とを具える。他の単位処理ユニット１０１、．．．も同様の構成要素を具え、例えば、単位処理ユニット１０１は、処理内容保持部４０１と、演算処理部５０１と、メモリ制御部６０１と、ユニット間通信部７０１と、プロファイル情報収集部３０１と、ユニット制御部２０１とを具える。以後、代表的に単位処理ユニット１００及びその構成要素のみを参照して説明する。前記単位処理ユニット間は、制御バス８００と、ユニット間通信路８２０－１、２．．．で接続され、各単位処理ユニットと記憶装置（図示せず）との間をメモリバス８１０で接続する。

【００１３】

例えば、処理内容保持部４００と、演算処理部５００と、メモリ制御部６００との組で、通常のプロセッサ（VLIW：Very Long Instruction Word processor）として動作可能である。たとえば、同じ機能をFPGA（Field Programmable Gate Array）と同様の技術を使った「可塑的ハードウェア」で実現することも可能である。

【００１４】

処理内容保持部４００に格納される処理内容（プログラム）によって、当該単位処理ユニットの動作を変えられるようにする。具体的には、システム全体の資源管理を行う資源管理スレッド（ＲＣ（resource core）と略記）、最適化処理を行う最適化スレッド（ＯＦ（optimizing fork）と略記）、プログラムの挙動を観測しプロファイル情報を収集・解析する観測スレッド（ＰＦ（profiling fork）と略記）、アプリケーションプログラムの実行を行う演算スレッド（ＣＦ（computing fork）と略記）の４つの種類がある。各スレッドは、単位処理ユニットで実行可能な４つの機能、すなわち、実行内容の変更など資源管理をする機能、最適化コードを生成する機能、プログラムの挙動観測機能、アプリケーション実行機能、に対応している。

【００１５】

単位処理ユニット１００には、プログラムのプロファイル情報を収集するための回路が設けられている（プロファイル情報収集部３００）。プロファイル情報収集部３００は、演算機能や記憶機能を持っていても良いし、隣接の単位処理ユニットに情報を転送するだけの機能でも良い。ここで収集されたプロファイル情報は、ユニット間通信部７００によりユニット開通信路８２０－１、２、．．．を介して他の単位処理ユニットに伝えることができる。

【００１６】

・資源管理スレッド（ＲＣ）を実行中の単位処理ユニット１００は、制御バス８００を使い他の単位処理ユニットの処理制御部をアクセスすることによって、他の単位処理ユニットの内部状態を変更できる機能を持つ。たとえば、処理内容保持部４００の内容を変更することで、各単位処理ユニットを任意の役割に変更することができる。また、単位処理ユニットで実行されるアプリケーションプログラムのコード（演算スレッド）を、より最適化されたコードに変更することも可能である。

【００１７】

単位処理ユニットの役割は、実行前に静的に決めることもできるが、上記の変更機能を使うことでプログラム実行中に動的に変更することもできる。

【００１８】

演算処理スレッド（ＣＦ）でのプログラムの実行の状態を、観測スレッド（ＰＦ）が監視する。観測スレッド（ＰＦ）が求めたプロファイル結果を用いて、最適化スレッド（ＯＦ）がより適したプログラム（オブジェクトコード）および処理形態を求める。その結果、実行効率が向上すると判断されれば、資源管理スレッド（ＲＣ）が上記の変更機能を用いることで、システムをより実行に適した状態に変更する。逆に、観測スレッド（ＰＦ）での監視の結果、演算スレッド（ＣＦ）での実行効率が悪化していると判断されれば、資源管理スレッド（ＲＣ）が各処理ユニットの役割分担を変更することにより、プログラムの挙動観測と最適化に．より適した構成に変更することができる。

【００１９】

図２は、前記記憶装置へのアクセスのためにユニット間通信路を使用するように図１の構成を変更した変形例のブロック図である。

【００２０】

図３は本発明による自己最適化演算装置の基本的な考え方を説明するブロック図である。単位処理ユニット１００～１１５は、図１のような内部構成を持つ単位処理ユニットである。図中丸印の中に書かれている記号（ＲＣ、ＰＦ、ＯＦ、ＣＦ）は、各単位処理ユニットで行われている処理機能に該当するスレッドの略称である。楕円９００～９２０で表現されているものは、前記単位処理ユニットを実行中の処理機能ごとに分けたグループ（処理ユニット群）を表している。資源管理処理ユニット群９００、最適化処理ユニット群９１０、観測処理ユニット群９２０、演算処理ユニット群９３０からなる。資源管理ユニット群９００は、システム内の各処理ユニットの制御を行う機能を持つ。このために制御バス（８００－１、２、．．．）を介して各単位処理ユニットをアクセスする。アプリケーションプログラムは演算処理ユニット群９３０で実行する。実行中のプログラムの挙動情報はユニット間通信路８２０－１を介して観測処理ユニット群９２０に逐一伝えられる

。観測処理ユニット群９２０では、この情報を解析してプログラム実行の様子を観測する。もし演算処理ユニット群９３０での実行効率が不十分でありさらに最適化する余地があれば、収集したプロファイル情報をユニット間通信路８２０－２を経由することで最適化処理ユニット群９１０に伝える。最適化処理ユニット群９１０では、プログラムをより効率的に実行するためのコードを生成する。生成したコードは、資源管理処理ユニット群９００の制御の下で、演算処理ユニット群９３０に伝えられる。この際、各処理ユニットの役割分担の変更が必要と判断されれば、資源管理処理ユニット群９００の制御により、各処理ユニット群に属する処理ユニットを変更する。各処理ユニット群とも、所定の処理を行うために必要な情報を保持するため、メモリバス８１０－１、－２、－３を介して記憶装置１０００にアクセスすることができる。

【００２１】

図４は、各処理ユニット群の動作を時間順に説明した図である。図中、１００、１０１－１～ｎ、１０２－１～ｎ、１０３－１～ｎ、１０４－１～ｎ、１０５－１～ｎ、１０６－１～ｎは前記単位処理ユニットを示している。上の説明と同様に、各単位処理ユニットで実行している機能スレッドを丸印の中に略記している。図中の楕円９００は資源管理処理ユニット群であり、９３０－１、９３０－２は演算処理ユニット群、９２０－１、９２０－２は観測処理ユニット群、９１０－１、９１０－２は最適化処理ユニット群である。各処理ユニット群の中に単位処置ユニットが記されている。各処理ユニット群に割当てられた単位処理ユニットを重畳して書くことで、当該処理ユニット群の内部で並列的に処理されていることを表現している。また、重畳度の増減により、処理ユニット群に割当てられた単位処理ユニットの数の増減を表現している。図４は、システム内でアプリケーションプログラムの実行を開始したときの状態から示している。アプリケーションプログラムは、あらかじめコンパイルされており、実行可能なオブジェクトコードが用意されているものとする。まず、資源管理処理ユニット群９００が動作し、他の各処理ユニットの役割分担を決め、演算処理ユニット群９３０－１、観測処理ユニット群９２０－１、最適化処理ユニット群９１０－１に属する単位処理ユニットを決定する。資源管理処理ユニット群９００は、制御バスを介して他の処理ユニット群で実行するスレッドを決め、必要な設定をするなどの準備を行う（ｂ１００）。準備が完了したら、演算処理ユニット群９３０－１、観測処理ユニット群９２０－１に対して指令（ｂ１１０－１、ｂ１１０－２）を送り、各々の処理ユニット群の実行を開始する（ｂ１０１）。実行開始後は、当面、資源管理処理スレッド群の役割はなく、処理スレッドを休止する（ｂ１０２）。演算処理ユニット群９３０－１は、与えられたプログラムの実行を行い（ｂ１２０）、実行中の情報を観測処理スレッドに送る（ｂ１３０－１～ｎ）。観測処理ユニット群９２０－１は、演算処理ユニット群９３０－１から送られてくる実行情報を逐一解析し、最適化が必要な状況に達したか否かを判断している（ｂ１４０）。もし最適化が必要と判断すれば（ｂ１４１）、その情報を資源管理処理ユニット群９００に送る（ｂ１１１－１）。資源管理処理ユニット群９００は、この情報を受けると休止状態から復帰し（ｂ１０３）、最適化処理ユニット群９１０－１の動作を起動する（ｂ１１１－２）。その後、資源管理処理ユニット群９００は休止状態となり、次のイベントが発生するまで待つ（ｂ１０４）。最適化処理ユニット群９１０－１は、起動後、観測処理ユニット群９２０－１から、プログラムのプロファイル情報（ｂ１５０－１～ｎ）を受け取り、この情報を元に最適化処理を行う（ｂ１６０）。最適化処理が終わると（ｂ－１６１）、資源管理処理ユニット群９００に対してその旨を通知し（ｂ１１２－１）、自身は休止状態になる（ｂ１６２）。演算処理ユニット群９３０－１、観測処理ユニット群９２０－１は、最適化処理ユニット群９１０－１で最適化処理を行っている間も、そのまま各々の実行を継続する（ｂ１２０、ｂ１４２）。最適化処理終了の通知を受けた資源管理処理ユニット群９００は、休止状態から復帰し（ｂ１０５）、演算処理ユニット９３０－１、観測処理ユニット９２０－１を一時停止させる（ｂ１１２－２、ｂ１１２－３）。ここで各処理ユニットは、資源管理処理ユニット群９００の管理下で役割分担の変更を施される（ｂ１２１、ｂ１４３）。その結果、新たな構成に変更され、演算処理ユニット群９３０－２、観測処理ユニット群９２０－２となる。

こうして、プログラムをより効率的に実行できるように変更した後、各処理ユニット群９３０－２、９２０－２の動作を起動する（ｂ１２２、ｂ１４４）。ここで、アプリケーションプログラムはｂ１２１で中断したときの続きを実行することになる。演算処理ユニット群９３０－２の実行中の情報を逐一、観測処理ユニット群９２０－２に転送する（ｂ１３１－１～ｎ）動作は、前と同様に行われる。観測処理ユニット群９２０－２が、再度、最適化の必要な状況を検出すれば（ｂ１４５）、ｂ１４１以降の動作と同様に、最適化が必要な旨の情報を資源管理処理ユニット群９００に送り（ｂ１１３－１）、これを受けて資源管理処理ユニット群９００が休止状態から回復し（ｂ１０７）、最適化処理ユニット群（９１０－２）に対して指示を送り（ｂ１１３－２）、処理を起動する（ｂ１６３）。最適化処理ユニット群９１０－２は、必要なプロファイル情報を観測処理ユニット群９２０－２から受け取り（ｂ１５１－１）、最適化処理を行う（ｂ１６３）。この間も、演算処理ユット群９３０－２、観測処理ユニット群９２０－２は継続して実行している（ｂ１２２、ｂ１４６）。

【００２２】

図５は、図４で説明した各単位処理ユニットの役割分担の変更の様子を説明する図である。この図はシステムの単位処理ユニットの役割分担の状況を示した３つの図からなる。上段の図は、プログラムの初期段階において最適化があまり進んでいない状況を示している。観測処理ユニット群９２０－１、最適化処理ユニット群９１０－１に多くの単位処理ユニットを割当てることで、プログラム実行の早期に最適化対象を特定し、最適化処理結果を求めることが可能となる。下段左側の図は、中程度に最適化が進んだ状況を示している。演算処理ユニット群９３０－２にやや多くの単位処理ユニットを割り当て、処理性能を向上させながら、それと並行して、観測処理ユニット群９２０－２、最適化処理ユニット群９１０－２で更に最適化ができるポイントを探し、最適化する。下段右側の図は、高度に最適化が進んだ状況を示している。高度に最適化された結果、それ以上に最適化を行う可能性は低くなる。このために、観測処理ユニット群（９２０－３）、最適化処理ユニット群（９１０－３）に割当てる単位処理ユニットの数を抑える。その分を演算処理ユニット群（９３０－３）に割り当てて最大の処理性能を達成する。観測処理ユニット群（９２０－３）での観測の結果、演算処理ユニット群（９３０－３）での実行効率が悪化していると判断されれば、資源管理処理ユニット群９００が制御することにより、各処理ユニット群の割り振りを変更し、これら３つの図の間を遷移することにより、状況に応じた最適な処理形態とする（図中の双方向矢印）。

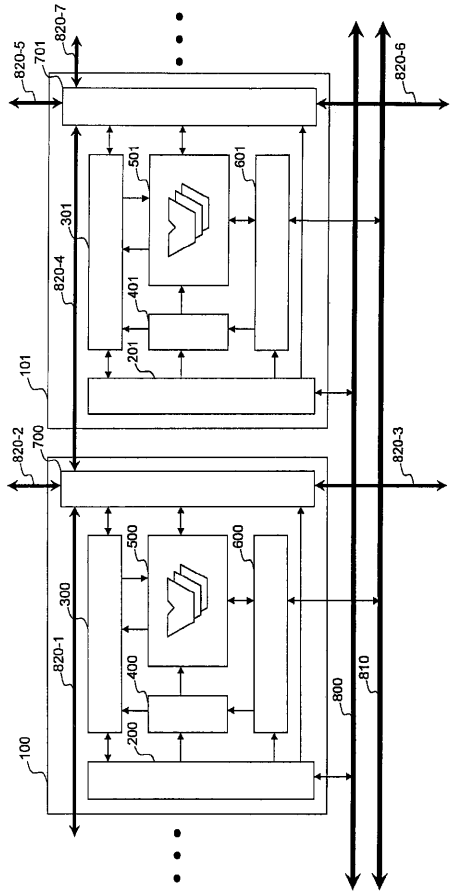
【００２３】

図６～図８は、図５中に示した各構成を、図１を元に説明した図である。図中、１００～１１１は単位処理ユニットを示す。単位処理ユニット内の各部の番号は記載を省略している。ただし、各単位処理ユニットで実行している機能処理の内容を、処理内容保持部（図１の４００、４０１）の位置に、処理スレッドの略称で表示している。たとえば、図６の単位処理ユニット１００は資源管理スレッドを実行するため、処理内容保持部にＲＣと記されている。アプリケーションプログラムの実行を開始するとき（初期状態）、たとえば、図６のような役割分担を行う。すなわち、資源管理処理ユニット群９００の管理の下で、最適化処理ユニット群９１０、観測処理ユニット群９２０、演算処理ユニット群９３０に分かれる。最適化処理が進むと、図７に示すように、演算処理ユニット群９３０の比率を上げ、観測処理ユニット群９２０、最適化処理ユニット群９１０の比率を相対的に下げる。全体の単位処理ユニットの数が少ない場合には、ひとつの単位処理ユニットが複数の役割を分担することも可能である。図７の場合、単位処理ユニット１００が、資源管理スレッド（ＲＣ）と最適化スレッド（ＯＦ）の２つを担当している。このため、資源管理・最適化処理ユニット群９４０ができている。さらに最適化が進み、最大限に最適化された状態を示したものが図８である。ここでは、最適化を最大限に施した結果、プログラムの実行を司る演算処理ユニット群９３０に単位処理ユニットの大半を割当てている状況を示している。残りのごく少数の単位処理ユニット（図８では１個）を、資源管理・最適化・観測の処理（ＲＣ、ＯＦ、ＰＦ）に割当てている（資源管理・最適化・観測処理ユニッ

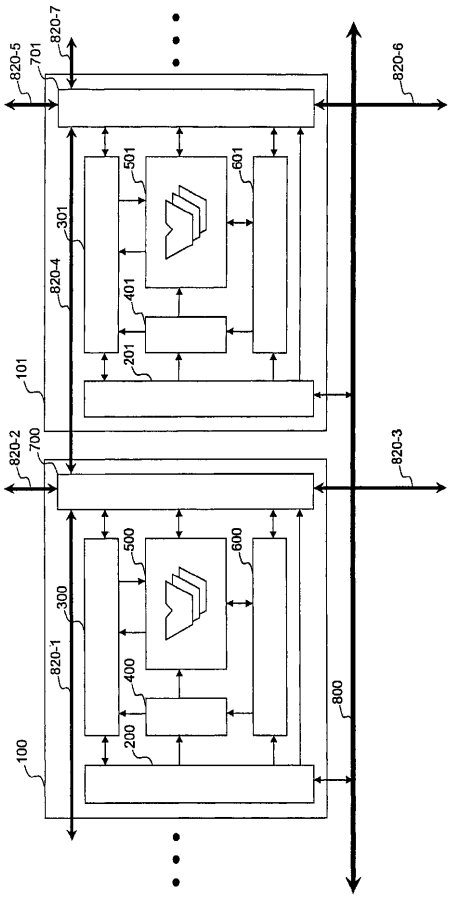
ト群９５０）。
【００２４】
図９および図１０は、各処理ユニット群の配置に関する一例を示す図である。処理ユニット群の範囲を図上で見やすくするため、ハッチングを施している。上述の説明では処理ユニット群に割当てる処理ユニットの個数に言及するだけで、配置方法については触れていなかった。上に挙げた本発明による実施例では、処理ユニット間の通信がユニット間通信路（図１中の８２０）を介して行われるため、通信の状況を勘案して処理ユニット群を配置しないと、ユニット間通信路を通る情報が輻輳し、性能向上を妨げる要因になる可能性がある。このため、現実的には、ユニット間通信路の負荷が最も少なくなるように処理ユニット群の配置を考える必要がある。図９は、最適化があまり進んでいない状態（あるいは初期状態）での、処理ユニット群の配置例である。ここでは演算処理ユニット群９３０に２つの処理ユニットが割当てられ、相互に通信している。この演算処理ユニット群９３０を取り囲むように観測処理ユニット群９２０を配置する。演算処理ユニット群での実行挙動の情報は、演算処理ユニット群９３０から外側に向かって流れるため、演算処理ユニット群内部での通信を阻害しない。さらにこの図では、観測処理ユニット群９２０の結果が、最適化処理ユニット群９１０に抵抗なく流れるように考慮されている。図１０は、最適化が進んだ状態での、処理ユニット群の配置例である。この例では、演算処理ユニット群９３０が環状の通信路を形成している。この環状通信路に沿った通信を阻害しないように観測処理ユニット群９２０、資源管理・最適化処理ユニット群９４０を配置している。
【産業上の利用可能性】
【００２５】
本発明によれば、処理ユニットを複数用いることによりアプリケーションプログラムの高速化を実現する演算処理装置において、該アプリケーションプログラム実行中に得られる情報を用いることで動的な最適化を行い、一層の高速化を果たすことができる。したがって、本発明は、高速な処理性能が求められる高性能電子計算機、汎用マイクロプロセッサ、機器組み込み装置など広い分野で適用が可能である。
【図面の簡単な説明】
【００２６】
【図１】本発明による自己最適化演算装置の一実施例の構成を示すブロック図である。
【図２】図１の自己最適化演算装置の変形例の構成を示すブロック図である。
【図３】本発明による自己最適化演算装置の基本的な考え方を説明するブロック図である。
【図４】本発明による自己最適化演算装置の各処理ユニット群の動作を時間順に説明した図である。
【図５】本発明による自己最適化演算装置の各単位処理ユニットの役割分担の変更の様子を説明する図である。
【図６】図５中に示した各構成を、図１を元に説明した図である。
【図７】図５中に示した各構成を、図１を元に説明した図である。
【図８】図５中に示した各構成を、図１を元に説明した図である。
【図９】本発明による自己最適化装置の各処理ユニット群の配置に関する一例を示す図である。
【図１０】本発明による自己最適化装置の各処理ユニット群の配置に関する他の例を示す図である。
【符号の説明】
【００２７】
１００、１０１、１０２　単位処理ユニット
２００、２０１　ユニット制御部
３００、３０１　プロファイル情報収集部
４００、４０１　処理内容保持部

５００、５０１　演算処理部
６００、６０１　メモリ制御部
７００、７０１　ユニット間通信部
８００　制御バス
８１０　メモリバス
８２０　ユニット間通信路
９００　資源管理処理ユニット群
９１０　最適化処理ユニット群
９２０　観測処理ユニット群
９３０　演算処理ユニット群
９４０　資源管理・最適化処理ユニット群
９５０　資源管理・最適化・観測処理ユニット群
１０００　記憶装置

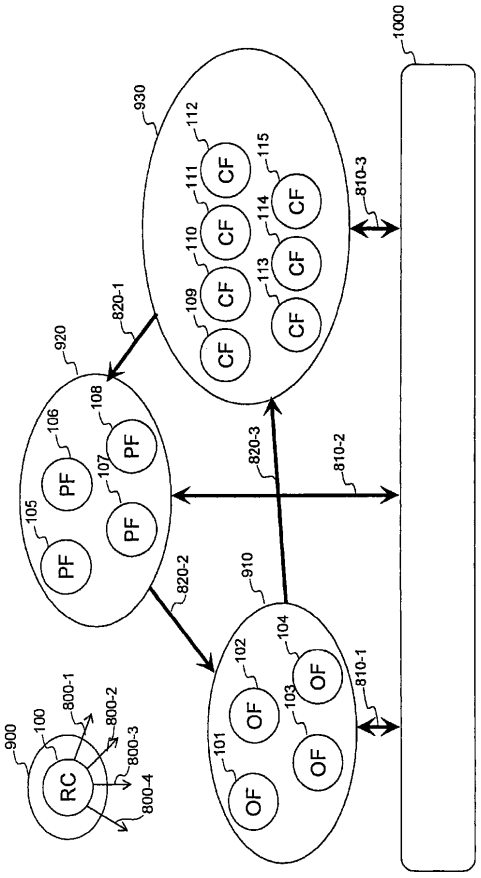
【図１】



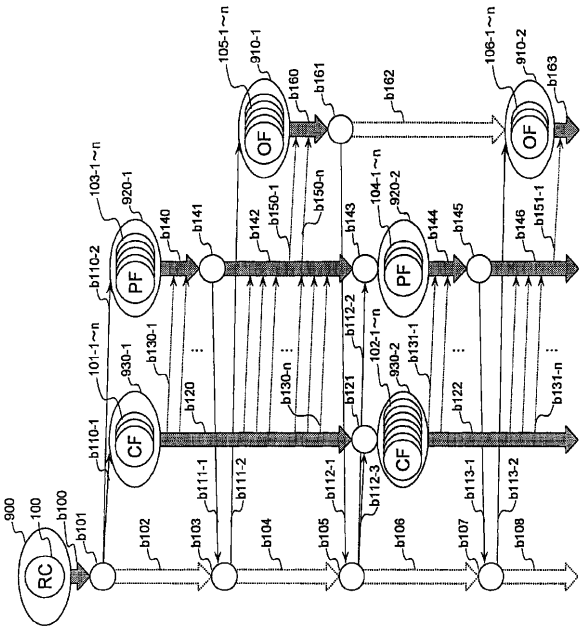
【図２】



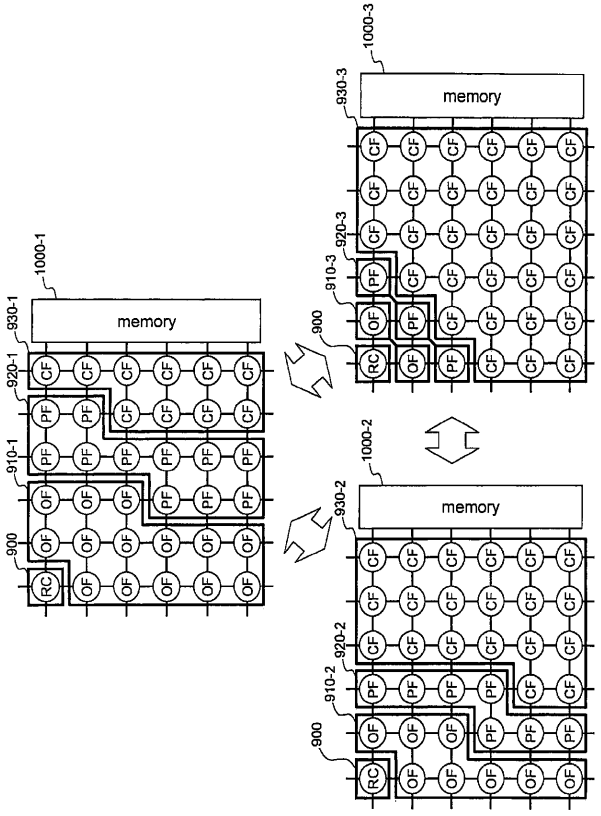
【図３】


1000
930
112
111
110
109
115
114
113
CF
810-3
820-1
920
108
106
107
105
PF
820-3
810-2
820-2
910
104
102
103
101
OF
810-1
900
100
RC
800-1
800-2
800-3
800-4

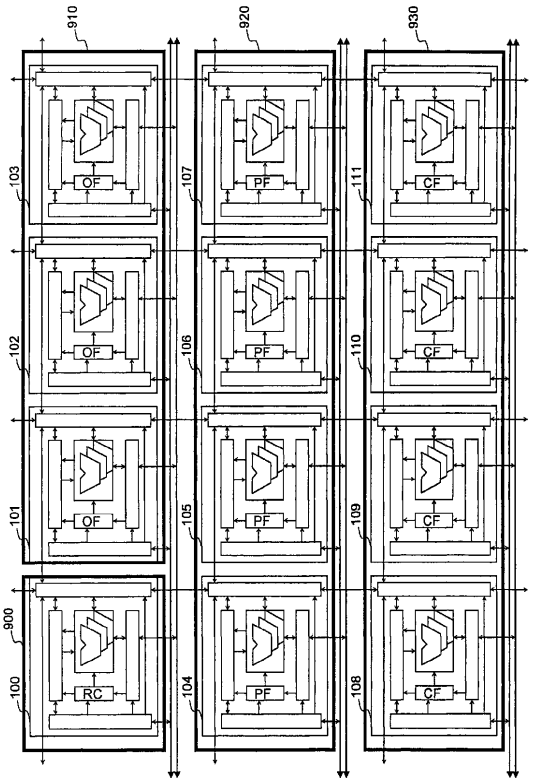
【図４】


105-1~n
910-1
b160
b161
b162
106-1~n
910-2
b163
OF
103-1~n
920-1
b140
b141
b142
b150-1
b150-n
b143
104-1~n
920-2
b144
b145
b146
b151-1
PF
b110-2
b130-1
b120
b130-n
b112-2
102-1~n
b131-1
b122
b131-n
101-1~n
930-1
930-2
b110-1
CF
b121
b111-1
b111-2
b112-1
b112-3
b113-1
b113-2
900
100
b100
b101
RC
b102
b103
b104
b105
b106
b107
b108

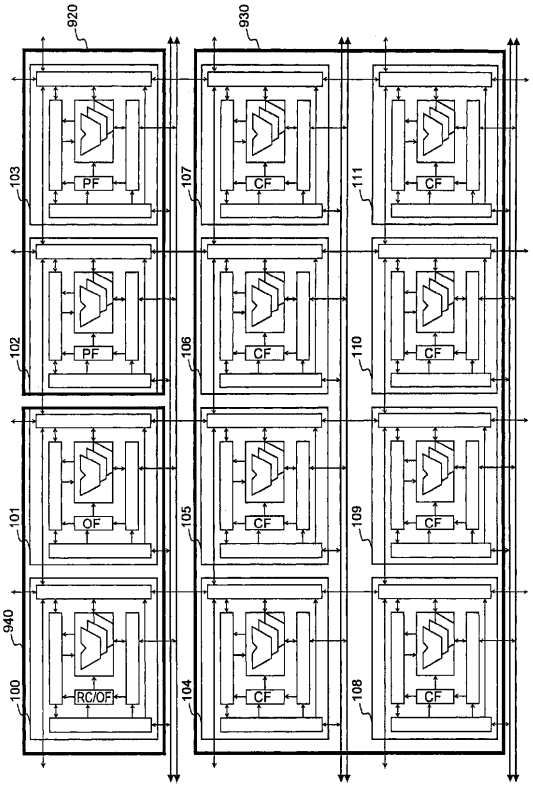
【図５】


1000-3
930-3
memory
1000-1
930-1
920-1
910-1
900
1000-2
930-2
920-2
910-2
CF
PF
OF
RC

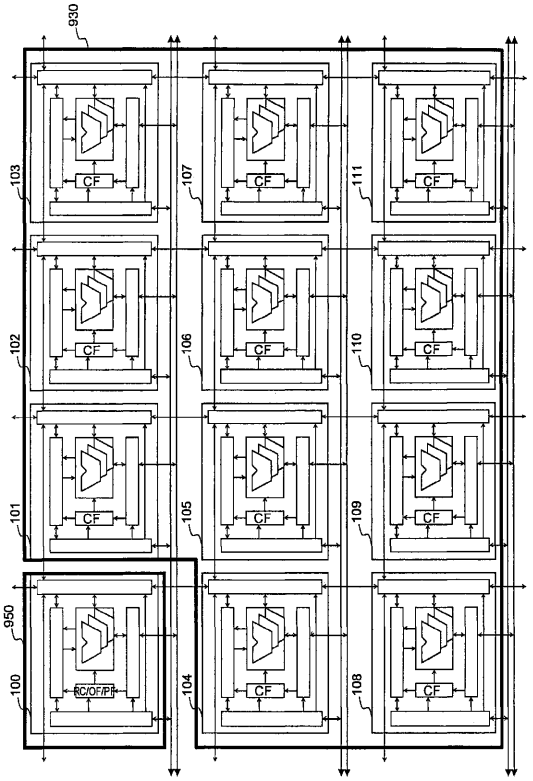
【図６】


910
920
930
103
107
111
102
106
110
101
105
109
900
104
108
100
OF
PF
CF
RC

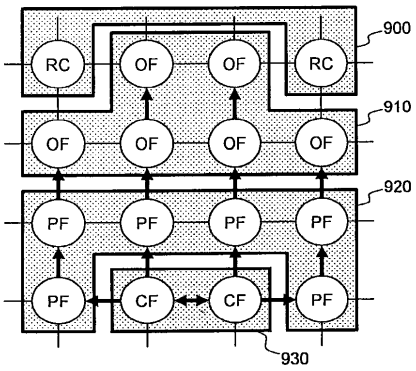
【図７】


920
930
103
107
111
PF
CF
102
106
110
PF
CF
101
105
109
OF
CF
940
100
104
108
RC/OF
CF

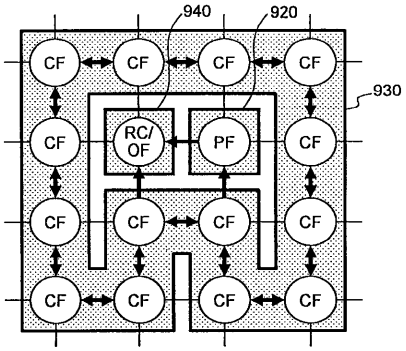
【図８】


930
103
107
111
CF
102
106
110
CF
101
105
109
CF
950
100
104
108
RC/OF/PF
CF

【図９】


900
RC
OF
OF
RC
910
OF
OF
OF
OF
920
PF
PF
PF
PF
PF
CF
CF
PF
930

【図１０】


940
920
CF
CF
CF
CF
930
CF
RC/
OF
PF
CF
CF
CF
CF
CF
CF
CF
CF
CF
CF

フロントページの続き

(74)代理人 100119530
弁理士 冨田 和幸
(72)発明者 馬場 敬信
栃木県宇都宮市西の宮２の２０の２５
(72)発明者 横田 隆史
栃木県宇都宮市若草４の３の２１
(72)発明者 大津 金光
栃木県宇都宮市峰４の３の２１ サンワマンション４０５

Ｆターム(参考) 5B013 DD04