

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号
特許第6074932号
(P6074932)

(45) 発行日 平成29年2月8日 (2017.2.8)

(24) 登録日 平成29年1月20日 (2017.1.20)

(51) Int. Cl.	F I
G O 6 F 9/52 (2006.01)	G O 6 F 9/46 4 7 5 A
G O 6 F 9/50 (2006.01)	G O 6 F 9/46 4 6 2 B
G O 6 F 9/30 (2006.01)	G O 6 F 9/30 3 5 0 A
G O 6 F 15/167 (2006.01)	G O 6 F 15/167 6 1 5 A
G O 6 F 15/173 (2006.01)	G O 6 F 15/173 6 6 0 D

請求項の数 6 (全 16 頁)

(21) 出願番号	特願2012-160696 (P2012-160696)	(73) 特許権者	000005223
(22) 出願日	平成24年7月19日 (2012.7.19)		富士通株式会社
(65) 公開番号	特開2014-21774 (P2014-21774A)		神奈川県川崎市中原区上小田中4丁目1番1号
(43) 公開日	平成26年2月3日 (2014.2.3)	(74) 代理人	100107766
審査請求日	平成27年4月6日 (2015.4.6)		弁理士 伊東 忠重
		(74) 代理人	100070150
			弁理士 伊東 忠彦
		(74) 代理人	100146776
			弁理士 山口 昭則
		(72) 発明者	近藤 祐史
			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
		審査官	三坂 敏夫

最終頁に続く

(54) 【発明の名称】 演算処理装置及び演算処理方法

(57) 【特許請求の範囲】

【請求項 1】

演算処理を行う複数の演算処理部と、
前記複数の演算処理部のそれぞれに対応して設けられた複数のレジスタと、
を含み、前記複数の演算処理部の各々について、演算処理部がプログラム中の特定の命令を実行すると前記複数のレジスタのうちの対応するレジスタのレジスタ値が変化され、前記特定の命令が実行される度に前記複数のレジスタのレジスタ値に応じて前記複数の演算処理部の優先度が変化され、
前記複数の演算処理部のいずれかが前記特定の命令を実行する度に、前記複数のレジスタのレジスタ値に基づいて進捗状況を判断し、前記複数の演算処理部のうちの相対的に早い演算処理部の優先度を相対的に低下させる
ことを特徴とする演算処理装置。

【請求項 2】

前記複数の演算処理部の各々について、プログラム中の所定位置に挿入された前記特定の命令が実行されると、前記複数のレジスタのうちの対応するレジスタのレジスタ値が変化されることを特徴とする請求項 1 記載の演算処理装置。

【請求項 3】

演算処理を行う複数の演算処理部と、
前記複数の演算処理部のそれぞれに対応して設けられた複数のレジスタと、
を含み、前記複数の演算処理部の各々について、演算処理部がプログラム中の特定の命令

を実行すると前記複数のレジスタのうちの対応するレジスタのレジスタ値が変化され、前記特定の命令が実行される度に前記複数のレジスタのレジスタ値に応じて前記複数の演算処理部の優先度が変化され、

前記複数の演算処理部のうちの一の演算処理部がプログラム中の特定の命令を実行すると、前記一の演算処理部が最も遅い演算処理部でない場合に前記複数のレジスタのうちの前記一の演算処理部に対応するレジスタのレジスタ値を所定値増加させ、前記一の演算処理部が最も遅い演算処理部である場合に前記複数のレジスタのうちの前記一の演算処理部以外の演算処理部に対応するレジスタのレジスタ値を前記所定値減少させることを特徴とする演算処理装置。

【請求項 4】

10

前記複数の演算処理部は共有リソースを共有し、前記優先度が第 1 の値である演算処理部は、前記優先度が前記第 1 の値より低い第 2 の値である演算処理部よりも、優先的に前記共有リソースが割り当てられることを特徴とする請求項 1 乃至 3 何れか一項記載の演算処理装置。

【請求項 5】

前記共有リソースは、キャッシュ、共有バス、及び共有電源電力の少なくとも 1 つであることを特徴とする請求項 4 記載の演算処理装置。

【請求項 6】

複数の演算処理部により演算処理を実行し、

前記複数の演算処理部の各々について、演算処理部がプログラム中の特定の命令を実行すると、前記複数の演算処理部のそれぞれに対応して設けられた複数のレジスタのうちの対応するレジスタのレジスタ値を変化させ、

20

前記特定の命令が実行される度に前記複数のレジスタのレジスタ値に応じて前記複数の演算処理部の優先度を変化させ、

前記複数の演算処理部のいずれかが前記特定の命令を実行する度に、前記複数のレジスタのレジスタ値に基づいて進捗状況を判断し、前記複数の演算処理部のうちの相対的に早い演算処理部の優先度を相対的に低下させる

各段階を含む演算処理方法。

【発明の詳細な説明】

30

【技術分野】

【0001】

本願開示は、演算処理装置及び演算処理方法に関する。

【背景技術】

【0002】

チップマルチプロセッサのコア数は年々増加しており、プロセッサ内にコアが複数存在するメニーコアプロセッサが開発されている。メニーコアプロセッサでは、ソフトウェア的に各コアを平等に扱ったとしても、各コアからの共有資源へのアクセス時間の不平等性や、アクセス競合、他のジッタ等によって、各コアのジョブの進捗に無視できないばらつきが生じる場合がある。

40

【0003】

複数のコア間の同期をとるためには、例えばバリア同期が用いられる。プログラム中に挿入したバリア同期命令にプログラム実行位置が到達すると、コアはプログラム実行を停止し、他の全てのコアのプログラム実行位置が対応バリア同期命令に到達する迄、停止状態で待つ。これにより、バリア同期命令の位置において全てのコアの間で同期が確立される。このようなバリア同期等の同期を確立する時間やプログラム計算の完了の時間は、最後のコアがバリアポイントに到達した時間や最後のコアが計算を完了した時間となる。そのため、コアによるプログラム実行の進捗のばらつきは、計算に必要な時間の増大や、並列化効率の低下を引き起こす。このような進捗のばらつきによる時間の増大や並列化効率の低下は、コア数が増大すると共に大きくなると考えられる。

50

【 0 0 0 4 】

ハードウェアに起因する進捗のばらつきは、実行タイミング等の再現不可能な要素によって影響を受ける。そのため、アプリケーション作成者が、ハードウェアに起因する進捗のばらつきを予め考慮してプログラミングをすることは難しい。従って、コア間の進捗のばらつきを低減するためには、実際の進捗の状況に応じて進捗速度を調整するハードウェア的な機構を用いることが望ましい。また、コア間でソフトウェアでは回避出来ない作業負荷差が生じた際に、同期に与える影響を小さくするためにも、ハードウェア的な機構を用いてコア間の進捗のばらつきを低減することが望ましい。

【 先行技術文献 】

【 特許文献 】

10

【 0 0 0 5 】

【 特許文献 1 】 特開 2 0 0 7 - 1 0 8 9 4 4 号 公 報

【 特許文献 1 】 特開 2 0 0 1 - 1 3 4 4 6 6 号 公 報

【 発明の概要 】

【 発明が解決しようとする課題 】

【 0 0 0 6 】

以上を鑑みると、演算処理部間の進捗のばらつきを低減する機構を備えた演算処理装置が望まれる。

【 課題を解決するための手段 】

【 0 0 0 7 】

20

演算処理装置は、演算処理を行う複数の演算処理部と、前記複数の演算処理部のそれぞれに対応して設けられた複数のレジスタとを含み、前記複数の演算処理部の各々について、演算処理部がプログラム中の特定の命令を実行すると前記複数のレジスタのうちの対応するレジスタのレジスタ値が変化され、前記特定の命令が実行される度に前記複数のレジスタのレジスタ値に応じて前記複数の演算処理部の優先度が変化され、前記複数の演算処理部のいずれかが前記特定の命令を実行する度に、前記複数のレジスタのレジスタ値に基づいて進捗状況を判断し、前記複数の演算処理部のうちの相対的に早い演算処理部の優先度を相対的に低下させることを特徴とする。

【 発明の効果 】

【 0 0 0 8 】

30

少なくとも 1 つの実施例によれば、演算処理部間の進捗のばらつきを低減する機構を備えた演算処理装置が提供される。

【 図面の簡単な説明 】

【 0 0 0 9 】

【 図 1 】 演算処理装置の実施例の構成の一例を示す図である。

【 図 2 】 進捗管理レジスタのレジスタ値に応じた優先度設定により進捗のばらつきが低減される様子を模式的に示す図である。

【 図 3 】 コアが実行するプログラムの一例を示す図である。

【 図 4 】 図 1 の演算処理装置の動作の一例を示すフローチャートである。

【 図 5 】 最速のコアが最初の管理ポイントに到達した状態の一例を示す図である。

40

【 図 6 】 2 番目に早いコアが最初の管理ポイントに到達した状態の一例を示す図である。

【 図 7 】 最も遅いコアが最初の管理ポイントに到達した状態の一例を示す図である。

【 図 8 】 進捗管理レジスタのレジスタ値が変化する様子 of 一例を示す図である。

【 図 9 】 共有バス調停ユニットにおける共有リソースの割り当て機構の一例を示す図である。

【 図 1 0 】 優先化装置の構成の一例を示す図である。

【 図 1 1 】 優先度に応じたキャッシュのウェイの割り当ての一例を示す図である。

【 図 1 2 】 優先度に応じたキャッシュのウェイの割り当ての一例を示す図である。

【 図 1 3 】 優先度に応じたキャッシュのウェイの割り当ての一例を示す図である。

【 図 1 4 】 優先度に応じたキャッシュのウェイの割り当ての一例を示す図である。

50

【発明を実施するための形態】

【0010】

以下に、本発明の実施例を添付の図面を用いて詳細に説明する。

【0011】

図1は、演算処理装置の実施例の構成の一例を示す図である。演算処理装置は、演算処理部としてのコア10乃至13、進捗管理ユニット14、及び共有リソース15を含む。進捗管理ユニット14は、進捗管理レジスタ20乃至23、加減算器24乃至27、及び進捗管理部28を含む。共有リソース15は、共有キャッシュ30、共有バス調停ユニット31、及び電源&クロック制御ユニット32を含む。なお図1において、各ボックスで示される各機能ブロックと他の機能ブロックとの境界は、基本的には機能的な境界を示すものであり、物理的な位置の分離、電気的な信号の分離、制御論理的な分離等に対応するとは限らない。各機能ブロックは、他のブロックと物理的にある程度分離された1つのハードウェアモジュールであってもよいし、或いは他のブロックと物理的に一体となったハードウェアモジュール中の1つの機能を示したものであってもよい。

10

【0012】

複数のコア10乃至13は、それぞれが演算処理を行う。進捗管理レジスタ20乃至23は、複数のコア10乃至13のそれぞれに対応して設けられている。図1の演算処理装置では、複数のコア10乃至13の各々について、コアのプログラム実行箇所がプログラム中の所定位置に到達すると、複数の進捗管理レジスタ20乃至23のうちの対応するレジスタのレジスタ値を変化させる。例えばコア10のプログラム実行箇所がプログラム中の所定位置に到達すると、それに応答して、コア10に対応する進捗管理レジスタ20に格納されるレジスタ値を例えば1増加させる。具体的には、例えば進捗管理部28が、コア10乃至13からの所定位置到達の報告に応答して、加減算器24乃至27を用いて進捗管理レジスタ20乃至23の現在のレジスタ値を+1し、進捗管理レジスタ20乃至23に増加後の値を格納すればよい。

20

【0013】

上記のようにすれば、進捗管理レジスタ20乃至23に格納されるレジスタ値は、コア10乃至13のプログラム実行箇所がプログラム中の所定位置に到達したか否かを示すことになる。またプログラム中に複数の所定位置が規定されている場合或いはプログラム実行箇所が同一の所定位置を複数回通過する場合等には、進捗管理レジスタ20乃至23に格納されるレジスタ値は、プログラム実行箇所が幾つ目の所定位置に到達したかを示すことになる。従って、進捗管理レジスタ20乃至23に格納されるレジスタ値に基づいて、コア10乃至13のプログラム実行の進捗状況を判断することができる。

30

【0014】

進捗管理部28は、進捗管理レジスタ20乃至23に格納されるレジスタ値に応じて、即ちコア10乃至13のプログラム実行の進捗状況に応じて、複数のコア10乃至13の優先度を変化させる。優先度を変化させる方法については後述する。複数のコア10乃至13の優先度を変化させることにより、プログラム実行の進捗が遅いコアについては、優先度を相対的に高く設定してよい。またプログラム実行の進捗が早いコアについては、優先度を相対的に低く設定してよい。複数のコア10乃至13は、共有リソース15を共有する。例えば優先度が第1の値であるコアは、優先度が第1の値より低い第2の値であるコアよりも、優先的に共有リソース15が割り当てられてよい。なおこの場合、割り当ての直接の対象となる共有リソースとしては、共有キャッシュ30のキャッシュメモリ、共有バス調停ユニット31の管理するバス、電源&クロック制御ユニット32の管理する共有電源等が含まれる。

40

【0015】

図2は、進捗管理レジスタのレジスタ値に応じた優先度設定により進捗のばらつきが低減される様子を模式的に示す図である。図2は、複数のコア10乃至13のそれぞれがプログラム実行することにより、プログラム実行箇所が進行していく様子を示している。バリア同期位置41は、各プログラム中に挿入されているバリア同期命令の位置であり、こ

50

の位置からコア 1 0 乃至 1 3 のプログラム実行が同時に開始（再開）される。バリア同期位置 4 2 は、各プログラム中に挿入されている次のバリア同期命令の位置であり、この位置においてコア 1 0 乃至 1 3 間の次の同期が確立される。プログラム中の所定位置 4 3 は、この位置にプログラム実行箇所が到達すると、進捗管理レジスタ 2 0 乃至 2 3 のレジスタ値が変化するような位置である。プログラム中の所定位置 4 3 は、例えば、コア 1 0 乃至 1 3 がそれぞれ実行するプログラム中に挿入された特定の命令の位置であってよい。この特定の命令は、バリア同期位置 4 1 とバリア同期位置 4 2 との間の適当な位置に設けられている。複数のコア 1 0 乃至 1 3 がそれぞれ実行する複数のプログラムの内容が互いに実質的に同一又は対応していれば、この特定の命令は、各プログラム中の実質的に同一又は対応する位置に設けられてよい。複数のプログラムの内容が互いに異なれば、この特定の命令は、各プログラム中においてバリア同期位置 4 1 とバリア同期位置 4 2 との間でプログラム進捗量が同等である位置に設けられてよい。

10

【 0 0 1 6 】

図 2 の例では、コア 1 3 が矢印 4 5 で示されるように最初にプログラム中の所定位置 4 3 に到達する。この時点での最速のコア 1 3 と最も遅いコア 1 3 とのプログラム実行の進捗度合いの差は、矢印 4 6 の長さに相当する量である。コア 1 3 のプログラム実行箇所がプログラム中の所定位置 4 3 に到達した時点で、コア 1 3 に対応する進捗管理レジスタ 2 3 のレジスタ値が例えば 1 増加される。なお複数の進捗管理レジスタ 2 0 乃至 2 3 のレジスタ値は、初期状態で全て 0 であってよい。進捗管理レジスタ 2 3 のレジスタ値が残りの進捗管理レジスタ 2 0 乃至 2 2 のレジスタ値よりも大きくなると、進捗管理部 2 8 は、コア 1 3 のプログラム実行が他のコアのプログラム実行よりも進捗していると判断し、コア 1 3 の優先度を下げようとする。具体的には、進捗管理部 2 8 からの通知（例えば各コアの優先度を示す優先度情報の通知）に基づいて、共有リソース 1 5 のリソース制御部が、コア 1 3 よりも他のコア 1 0 乃至 1 2 を優先的に取り扱う。ここで共有リソース 1 5 のリソース制御部とは、例えば共有キャッシュ 3 0 のキャッシュ制御部、共有バス調停ユニット 3 1、電源 & クロック制御ユニット 3 2 等であってよい。

20

【 0 0 1 7 】

上記のようにして、コア 1 3 の優先度が下がることにより、コア 1 3 のプログラム進行が遅くなる。その結果、コア 1 3 のプログラム実行箇所がバリア同期位置 4 2 に到達したときには、最速のコア 1 3 と最も遅いコア 1 0 とのプログラム実行の進捗度合いの差は、矢印 4 7 の長さに相当する量となる。この量は、矢印 4 6 が示す優先度調整の無い状態での最速のコア 1 3 と最も遅いコア 1 0 とのプログラム実行の進捗度合いの差を考慮すると、十分に小さな量となっている。なお、仮に優先度調整が全く行われなかったとすると、コア 1 3 のプログラム実行箇所がバリア同期位置 4 2 に到達したときには、矢印 4 6 の長さの 2 倍の長さに相当する進捗度合いの差が、最速のコア 1 3 と最も遅いコア 1 0 との間に発生していたことになる。

30

【 0 0 1 8 】

図 3 は、コア 1 0 乃至 1 3 が実行するプログラムの一例を示す図である。この例では、コア 1 0 乃至 1 3 の各々が、図 3 に示される同一の内容のプログラムを実行する。このプログラムをコア 1 0 乃至 1 3 のそれぞれが実行することにより、コア 1 0 乃至 1 3 がそれぞれの配列 b の値の和 a を求め、最後のコマンド "all reduce-sum" により、各コアが求めた和 a の総和を求める。プログラム中の命令 5 1 は、最初のバリア同期命令である。バリア同期命令 5 1 の位置は、図 2 に対応させると、バリア同期位置 4 1 に相当する。プログラム中の命令 5 2 は、2 番目のバリア同期命令である。バリア同期命令 5 2 の位置は、図 2 に対応させると、バリア同期位置 4 2 に相当する。命令 5 3 は、進捗管理ユニット 1 4 に対して、プログラム実行箇所が所定位置に到達したことを報告する進捗状況報告命令である。進捗状況報告命令 5 3 の位置は、図 2 に対応させると、プログラム中の所定位置 4 3 に相当する。

40

【 0 0 1 9 】

進捗状況報告命令 5 3 のパラメータ `my rank` は、当該プログラムを実行するコアの

50

番号を示す。例えばコア 10 が実行するプログラムにおいて、パラメータ `myrank` は 0 に設定される。例えばコア 11 が実行するプログラムにおいて、パラメータ `myrank` は 1 に設定される。例えばコア 12 が実行するプログラムにおいて、パラメータ `myrank` は 2 に設定される。例えばコア 13 が実行するプログラムにおいて、パラメータ `myrank` は 3 に設定される。またパラメータ `ngroupe` は、当該プログラムを実行するコアが所属するグループを規定する。例えば、コア 10 乃至 13 を、コア 10 及びコア 11 が所属する第 1 のグループと、コア 12 及びコア 13 が所属する第 2 のグループとに分け、それぞれのグループにおいて独立に進捗のばらつきを調整してよい。即ち、第 1 のグループでは、コア 10 とコア 11 とのうち早い方のコアの進行速度を遅くするように優先度を調整し、第 2 のグループでは、コア 12 とコア 13 とのうち早い方のコアの進行速度を遅くするように優先度を調整してよい。また或いは、コア 10 乃至 13 の全てが同一のグループに属するようにパラメータ `ngroupe` を設定し、コア 10 乃至 13 間での相対的な進捗度合いに応じて、各コアの優先度を調整してよい。

10

【0020】

あるコアにより進捗状況報告命令 53 が実行されると、パラメータ `myrank` とパラメータ `ngroupe` とが、当該コアから進捗管理部 28 に通知される。進捗管理部 28 は、この通知にตอบสนองして、パラメータ `myrank` が示す進捗管理レジスタのレジスタ値を変化させる（例えば 1 増加させる）。このようにして、複数のコア 10 乃至 13 の各々は、プログラム中の所定位置に挿入された所定のコマンドを実行すると、進捗管理レジスタ 20 乃至 23 のうちの対応するレジスタのレジスタ値を変化させる。進捗管理部 28 は、進捗管理レジスタ 20 乃至 23 のレジスタ値に基づいてコア 10 乃至 13 の優先度を変化させる際に、パラメータ `ngroupe` が示すグループ分けに応じて優先度を変化させてよい。

20

【0021】

図 4 は、図 1 の演算処理装置の動作の一例を示すフローチャートである。ステップ S1 において、あるコアのプログラム実行箇所が管理ポイント（即ちプログラム中の所定位置）に到達する。これにより、管理ポイントに到達した旨の報告が当該コアから進捗管理部 28 に送信される。

【0022】

ステップ S2 において、進捗管理部 28 が、進捗管理レジスタ 20 乃至 23 を参照してレジスタ値をチェックする。ステップ S3 において、進捗管理部 28 は、今回管理ポイントに到達したコア以外のコアが対応する管理ポイントに既に到達しているか否かを判定する。即ち、今回管理ポイントに到達したコアが最も進捗の遅いコアであるか否かを判定する。今回管理ポイントに到達したコア以外のコアが対応する管理ポイントに既に到達していない場合、即ち、今回管理ポイントに到達したコアが最も進捗の遅いコアでない場合、ステップ S4 で、当該コアの進捗管理レジスタを 1 増加させる。それに続きステップ S5 で、進捗管理部 28 は、当該コアの共有リソース 15 へのアクセスの優先度を低下させるように、共有リソース 15 へ必要な通知（例えば各コアの優先度を示す優先度情報の送信）を行う。

30

【0023】

図 5 は、最速のコアが最初の管理ポイントに到達した状態の一例を示す図である。図 6 は、2 番目に早いコアが最初の管理ポイントに到達した状態の一例を示す図である。図 7 は、最も遅いコアが最初の管理ポイントに到達した状態の一例を示す図である。これらの例において、バリア同期位置 41 及びバリア同期位置 42 は、図 2 で説明したものと同様である。この例では、3 つのプログラム中の所定位置として、3 つの管理ポイント 61 乃至 63 が設定されている。コア 13 が第 1 の管理ポイント 61 に最初に到達し、コア 11 が第 1 の管理ポイント 61 に 2 番目に到達し、コア 12 が第 1 の管理ポイント 61 に最も遅く到達している。

40

【0024】

図 5 に示す例の場合、第 1 の管理ポイント 61 に到達したコア 13 は最も進捗の遅いコ

50

アではないので、ステップ S 4 で当該コア 1 3 の進捗管理レジスタ 2 3 が 1 増加される。それに続きステップ S 5 で、当該コア 1 3 の共有リソース 1 5 へのアクセスの優先度を低下させるように、共有リソース 1 5 へ必要な通知が行われる。図 6 に示す例の場合も、第 1 の管理ポイント 6 1 に到達したコア 1 1 は最も進捗の遅いコアではないので、当該コア 1 1 の進捗管理レジスタ 2 1 が 1 増加され、当該コア 1 1 の共有リソース 1 5 へのアクセスの優先度が低下される。

【 0 0 2 5 】

図 4 を再び参照し、ステップ S 3 において、今回管理ポイントに到達したコア以外のコアが対応する管理ポイントに既に到達している場合、即ち、今回管理ポイントに到達したコアが最も進捗の遅いコアである場合、ステップ S 6 に進む。ステップ S 6 において、当該コア以外のコアの進捗管理レジスタを 1 減少させる。前述のように、あるコアのプログラム実行箇所がプログラム中の所定位置に到達すると、当該コアが最も遅いコアでない場合には、複数のレジスタのうちの当該コアに対応するレジスタのレジスタ値を所定値（この例では 1）増加させる。但しステップ S 3 に示すように、当該コアが最も遅いコアである場合には、複数のレジスタのうちの当該コア以外のコアに対応するレジスタのレジスタ値を所定値（この例では 1）減少させてよい。

【 0 0 2 6 】

なおこのステップにおける 1 減少させる処理は必ずしも必要ではないが、この処理によりある管理ポイントに全コアが到達した場合に進捗管理レジスタのレジスタ値を 1 減少させることで、最も遅いコアのレジスタ値を常に 0 の状態に保つことができる。従って、レジスタ間でのレジスタ値の比較をする必要なく、ある進捗管理レジスタのレジスタ値のみに基づいて、当該レジスタに対応するコアが相対的にどれだけ進捗しているのかを判断することができる。またこのようにすることで、今回管理ポイントに到達したコア以外のコアが対応する管理ポイントに既に到達しているか否かを判断するためには、他のコアの進捗管理レジスタの値が全て 1 以上であるか否かを判断すればよい。

【 0 0 2 7 】

図 7 の例の場合、第 1 の管理ポイント 6 1 に到達したコア 1 2 は最も進捗の遅いコアであるので、ステップ S 6 において、当該コア以外のコア 1 0 , 1 1 , 1 3 の進捗管理レジスタ 2 0 , 2 1 , 2 3 を 1 減少させる。これにより、最も進捗の遅いコア 1 2 に対応する進捗管理レジスタ 2 2 のレジスタ値は 0 のままととなる。

【 0 0 2 8 】

図 4 を再び参照し、ステップ S 7 において、進捗管理部 2 8 は、全てのコアの進捗管理レジスタの値が 0 であるか否かを判定する。全てのコアの進捗管理レジスタの値が 0 である場合、ステップ S 8 において、共有リソース 1 5 に対する全コアのアクセス優先度をリセットして、初期状態のアクセス優先度に戻す。即ち、最も遅いコアがある管理ポイントに到達した時点で、何れのコアもその次の管理ポイントには未だ到達していない場合、コア間の進捗状況の差は十分に小さいとの判断に基づいて、初期状態のアクセス優先度に戻す。アクセス優先度の初期状態は、例えば、全てのコアに対して同一の優先度が設定されている状態、或いは優先度の設定無しの状態等であってよい。

【 0 0 2 9 】

図 8 は、進捗管理レジスタのレジスタ値が変化する様子の一例を示す図である。最初に、コア 1 3 が管理ポイントに到達し、コア 1 3 に対応する進捗管理レジスタが 0 から 1 になる。次に、コア 1 2 が管理ポイントに到達し、コア 1 2 に対応する進捗管理レジスタが 0 から 1 になる。次に、コア 1 1 が管理ポイントに到達し、コア 1 1 に対応する進捗管理レジスタが 0 から 1 になる。次に、コア 1 0 が管理ポイントに到達すると、他の全てのコアが対応管理ポイントに到達しているので、コア 1 1 乃至 1 3 に対応する進捗管理レジスタが 1 減少して 1 から 0 になる。即ち、コア 1 0 乃至 1 3 に対応する進捗管理レジスタの値は全て 0 になる。

【 0 0 3 0 】

その後、コア 1 2、コア 1 1、コア 1 2、コア 1 0 がこの順番で管理ポイントに到達す

10

20

30

40

50

ることにより、コア 10 乃至 13 に対応する進捗管理レジスタの値は 1, 1, 2, 0 となる。この時点で、コア 13 が管理ポイントに到達すると、他の全てのコアが対応管理ポイントに到達しているので、コア 10 乃至 12 に対応する進捗管理レジスタがそれぞれ 1 減少する。この結果、コア 10 乃至 13 に対応する進捗管理レジスタの値は 0, 0, 1, 0 となる。

【0031】

以上のように変化する進捗管理レジスタ 20 乃至 23 のレジスタ値に基づいて、図 1 を参照して述べたように、進捗管理部 28 が、共有リソース 15 に対して優先度調整のための通知（例えば各コアの優先度を示す優先度情報の通知）を行う。この通知に基づいて、共有リソース 15 のリソース制御部が、共有リソースの割り当てを調整する。ここで共有リソース 15 のリソース制御部とは、例えば共有キャッシュ 30 のキャッシュ制御部、共有バス調停ユニット 31、電源 & クロック制御ユニット 32 等であってよい。

10

【0032】

まず、電源 & クロック制御ユニット 32 による共有リソースの割り当てについて説明する。一般に、コアの消費電力と周波数とは密接な関係がある。コアの動作周波数を上げて処理速度を増加させるためには、電源電圧を上げることが好ましく、その結果、コアの消費電力は大きくなる。その際、放熱の問題、環境の問題、更にはコスト等の観点から、プロセッサが用いる電力に上限を設定することがある。このように使用電力に上限の設定がある場合、周波数や電力も各コアの共有資源と考えることができる。限られた電力の分配をコアの優先度に応じて調整することによって、進捗の遅いコアの周波数を相対的に高くし、進捗の早いコアの周波数を相対的に遅くすることが考えられる。

20

【0033】

即ち、図 1 に示されるように、電源 & クロック制御ユニット 32 は、進捗管理部 28 から各コアの優先度を示す優先度情報を受け取る。電源 & クロック制御ユニット 32 は、優先度情報に基づいて、コア 10 乃至 13 に供給する電源電圧及びクロック周波数を変化させる。この際、進捗管理部 28 から、電源 & クロック制御ユニット 32 に対して、電源電圧及びクロック周波数の変化を要求するようにしてもよい。電源 & クロック制御ユニット 32 は、進捗が早いために優先度が低いコアに対しては、供給する電源電圧及びクロック周波数を低下させてよい。また同様に、電源 & クロック制御ユニット 32 は、進捗が遅いために優先度が高いコアに対して、供給する電源電圧及びクロック周波数を増加させてもよい。

30

【0034】

図 9 は、共有バス調停ユニット 31 における共有リソースの割り当て機構の一例を示す図である。図 9 には、コア 10 乃至 13、進捗管理ユニット 14、優先化装置 71、LRU ユニット 72、AND 回路 73 乃至 76、OR 回路 77、及び 2 次キャッシュ 78 が示される。図 1 の共有バス調停ユニット 31 は、優先化装置 71 及び LRU ユニット 72 を含んでよく、AND 回路 73 乃至 76、OR 回路 77、及び 2 次キャッシュ 78 は、図 1 の共有キャッシュ 30 に含まれてよい。なお優先化装置 71 は、共有バス調停ユニット 31 側ではなく進捗管理ユニット 14 側に含まれてもよい。

【0035】

40

1 次キャッシュはコア 10 乃至 13 の各々に内蔵されており、2 次キャッシュ 78 は、メモリ階層において、外部メモリ装置と 1 次キャッシュとの間に存在する。1 次キャッシュへのアクセスにおいてキャッシュミスが発生した場合、2 次キャッシュ 78 へのアクセスが実行される。LRU ユニット 72 は、複数のコア 10 乃至 13 のうちで最後に 2 次キャッシュ 78 にアクセスしてから最も時間の経過している LRU (Least Recently Used) コアが何れのコアであるのかを示す情報を保持している。LRU ユニット 72 は、コア 10 乃至 13 に対して特に優先度の設定が無い場合、他のコアに優先して LRU コアに 2 次キャッシュ 78 へのバス (OR 回路 77 の出力が接続される部分) へのアクセスを許可する。具体的には、例えばコア 11 が LRU コアである場合、コア 11 がアクセス先のアドレスを出力し且つアクセス許可を要求するアクセスリクエスト信号をアサートすると、

50

L R Uユニット72は、対応するAND回路74への信号を1に設定してアクセスを許可する。即ち、アクセス許可されたコア11の出力するアドレス信号が、AND回路74及びOR回路77を介して2次キャッシュ78に供給される。コア11がアクセスリクエスト信号をアサートしている状態で、他のコアが2次キャッシュ78にアクセスしようとしても、L R Uコアであるコア11が優先されるので、他のコアは2次キャッシュ78にアクセスすることはできない。即ち、L R Uコアであるコア11以外のコア10, 12, 13からアクセスリクエスト信号を受け取っても、L R Uユニット72は、それぞれ対応するAND回路73, 75, 76への信号を0のまま保持する。

【0036】

進捗管理ユニット14によりコア10乃至13に対して優先度設定がされている場合、優先化装置71により、L R Uユニット72によるアクセス許可動作を調整する。具体的には、優先化装置71は、コア10乃至13の優先度に関する優先度情報を進捗管理ユニット14から受け取り、当該優先度情報に基づいて、優先度の相対的に低いコアに対しては、L R Uユニット72へのアクセスリクエスト信号を遮断する。即ち、コア10乃至13からのアクセスリクエスト信号は、通常は優先化装置71を介してL R Uユニット72に供給されるが、優先度の相対的に低いコアからのアクセスリクエスト信号は、優先化装置71により遮断され、L R Uユニット72に供給されない。

【0037】

図10は、優先化装置71の構成の一例を示す図である。優先化装置71は、AND回路80-1乃至80-4、OR回路81-1乃至81-4、2入力のうち一方が負論理入力であるAND回路82-1乃至82-4及び83-1乃至83-4、AND回路84-1乃至84-4、及びOR回路85-1乃至85-4を含む。進捗管理ユニット14は、進捗管理レジスタのレジスタ値が0である場合に1となり且つ当該レジスタ値が0以外の時に0となる優先度情報を、AND回路80-1乃至80-4の第1の入力に印加する。この優先度情報は更に、AND回路83-1乃至83-4及びAND回路84-1乃至84-4の第1の入力にも印加される。例えばコア10に対して優先度情報が0の場合、このコア10の進捗管理レジスタ20の値は1以上であり、コア10が相対的に進捗していること、即ちコア10の優先度は低いことを示す。また例えばコア10に対して優先度情報が1の場合、このコア10の進捗管理レジスタ20の値は0であり、コア10が相対的に遅れていること、即ちコア10の優先度は高いことを示す。

【0038】

コア10乃至13は、アクセス要求時にアクセスリクエスト信号を1にアサートし、これらアクセスリクエスト信号はAND回路80-1乃至80-4の第2の入力に印加される。またこれらアクセスリクエスト信号は、AND回路82-1乃至82-4の第1の入力、及びAND回路84-1乃至84-4の第2の入力に印加される。AND回路82-1乃至82-4の出力が、AND回路83-1乃至83-4の第2の入力に印加される。またAND回路82-1乃至82-4の第2の入力には、OR回路81-1乃至81-4の出力が印加される。

【0039】

例えばコア10に対する優先度情報が印加されるAND回路83-4及び84-4に着目した場合、コア10の優先度情報が1である（即ち優先度が高い）場合、コア10からのアクセスリクエスト信号はAND回路84-4側の経路を通る。即ち、コア10の優先度情報が1である（即ち優先度が高い）場合、コア10からのアクセスリクエスト信号は、AND回路84-4を通過し、OR回路85-4を介して優先化装置71から出力される。出力された信号は、優先化装置71からL R Uユニット72に供給される。

【0040】

またコア10の優先度情報が0である（即ち優先度が低い）場合、コア10からのアクセスリクエスト信号はAND回路83-4側の経路を通る。但し、AND回路80-2乃至80-4及びOR回路81-4による論理演算に相当する所定の条件が満たされた場合のみ、アクセスリクエスト信号は、AND回路82-4及びAND回路83-4を通過し

10

20

30

40

50

、OR回路85-4を介して優先化装置71から出力される。出力された信号は、優先化装置71からLRUユニット72に供給される。

【0041】

AND回路80-1乃至80-4はそれぞれ、対応するコア10乃至13がアクセスリクエスト信号をアサートし且つ対応優先度が高いときにのみ、その出力を1にする。OR回路81-4は、AND回路80-2乃至80-4の出力のOR演算を行い、OR演算結果を出力する。従って、OR回路81-4の出力が1になるのは、コア10以外の少なくとも1つのコアで優先度の高いものがアクセスリクエスト信号をアサートした場合である。それ以外の場合、OR回路81-4の出力は0になる。

【0042】

従ってコア10の優先度が低い場合、コア10以外の少なくとも1つのコアで優先度の高いものがアクセスリクエスト信号をアサートすれば、コア10のアクセスリクエスト信号はLRUユニット72に供給されない。コア10の優先度が低い場合、コア10のアクセスリクエスト信号がLRUユニット72に供給されるのは、コア10以外のコアで優先度の高いものがアクセスリクエスト信号をアサートしていないときのみである。

【0043】

図11乃至図14は、優先度に応じたキャッシュのウェイの割り当ての一例を示す図である。共有キャッシュ30は、進捗管理部28からの優先度情報に基づいて、ウェイの割り当てを制御してよい。複数のコア10乃至13は、それぞれが保有する専用の1次キャッシュとは別に、2次キャッシュである共有キャッシュ30にアクセスできる。この際、共有キャッシュ30の共有リソースであるウェイの使用においては、コア10乃至13間での競合に起因してキャッシュミスが発生する場合がある。競合によるキャッシュミスはCPU内のコア数が増えたと増加する傾向にある。そこで、コア間の競合によるキャッシュミスの頻度を下げるために、コアに対して動的なキャッシュのウェイ分割をすることが考えられる。その際、ウェイ分割の仕方をコアの優先度に基づいて調整することで、進捗の遅いコアに対しては優先的にウェイを割り当てることが考えられる。

【0044】

図1に示す進捗管理ユニット14からの優先度情報に基づいて、共有キャッシュ30がキャッシュのウェイ分割を行う例について以下に説明する。以下の説明において、ウェイの数（即ち各インデックスに対応するタグの数）は16であるとする。

【0045】

図11乃至図14の例では、縦16行が16のウェイを示し、横4列がそれぞれ4つのインデックスに対応する。コア10乃至13の進捗状況が同一である場合、図11に示すように、各コアには4つずつウェイを占有させてよい。なお"0"はコア10に占有させるウェイ、"1"はコア11に占有させるウェイ、"2"はコア12に占有させるウェイ、"3"はコア13に占有させるウェイを示す。

【0046】

例えばコア10が進んでおり他のコア11乃至13が遅れている場合、共有キャッシュ30における動的なキャッシュのウェイ割り当てにより、コア10が1つのウェイを占有し、他のコア11乃至13が5つずつウェイを占有するようにしてよい。図12に、そのようにウェイを割り当てた例が示される。

【0047】

また例えばコア10及び11が進んでおり他のコア12及び13が遅れている場合、共有キャッシュ30における動的なキャッシュのウェイ割り当てにより、コア10及び11がそれぞれ2つのウェイを占有し、他のコア12及び13が6つずつウェイを占有するようにしてよい。図13に、そのようにウェイを割り当てた例が示される。

【0048】

また例えばコア10乃至12が進んでおり他のコア13が遅れている場合、共有キャッシュ30における動的なキャッシュのウェイ割り当てにより、コア10乃至12がそれぞれ3つのウェイを占有し、他のコア13が7つのウェイを占有するようにしてよい。図1

10

20

30

40

50

4 に、そのようにウェイを割り当てた例が示される。

【 0 0 4 9 】

上記のウェイの割り当て例はあくまで一例であり、限定を意図するものではない。上記以外の様々なウェイの割り当てが可能である。

【 0 0 5 0 】

以上、演算処理装置を実施例に基づいて説明したが、本発明は上記実施例に限定されるものではなく、特許請求の範囲に記載の範囲内で様々な変形が可能である。

【 0 0 5 1 】

例えば、進捗管理レジスタ 2 0 乃至 2 3 のレジスタ値の書き換えや優先度の調整は進捗管理部 2 8 により集中管理的に実行される例について説明したが、そのように集中管理的にではなく各コア 1 0 乃至 1 3 により分散的に実行されてもよい。例えば、各コア 1 0 乃至 1 3 が、所定の命令を実行することにより、対応する進捗管理レジスタ 2 0 乃至 2 3 のレジスタ値を直接に書き換えてよい。また各コア 1 0 乃至 1 3 が、進捗管理レジスタ 2 0 乃至 2 3 のレジスタ値を参照して、自らの優先度を下げるように、各共有リソースの制御部に働きかけてもよい。

10

【 0 0 5 2 】

また同期ポイントは、バリア同期によるものでなくとも、任意の方式で同期を確立するものであってよい。また同期ポイント間の進捗管理ポイント（進捗をチェックする所定位置）の数は、1 つであっても複数であってもよい。また同期ポイントが設けられることなく、プログラム動作開始から終了までの間に 1 つ又は複数の管理ポイントが設けられていてもよい。

20

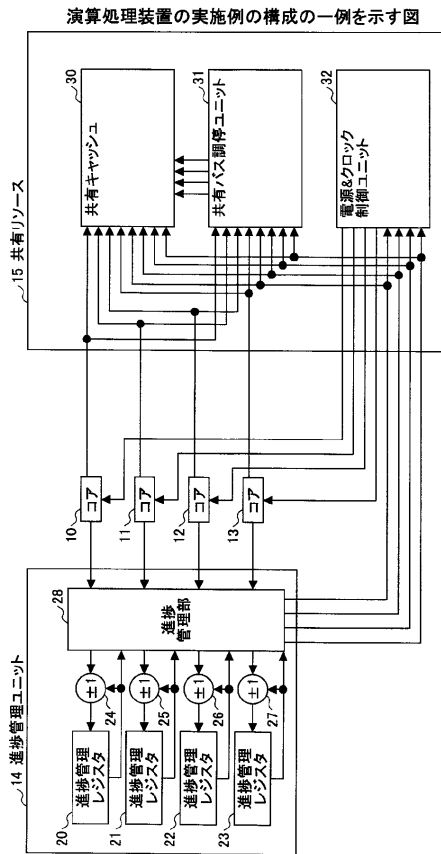
【符号の説明】

【 0 0 5 3 】

1 0 , 1 1 , 1 2 , 1 3 コア
1 4 進捗管理ユニット
1 5 共有リソース
2 0 , 2 1 , 2 2 , 2 3 進捗管理レジスタ
2 4 , 2 5 , 2 6 , 2 7 加減算器
2 8 進捗管理部
3 0 共有キャッシュ
3 1 共有バス調停ユニット
3 2 電源 & クロック制御ユニット

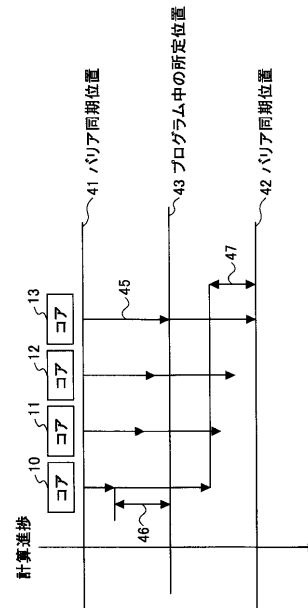
30

【図 1】



【図 2】

進捗管理レジスタのレジスタ値に応じた優先度設定により
進捗のばらつきが低減される様子を模式的に示す図



【図 3】

コアが実行するプログラムの一例を示す図

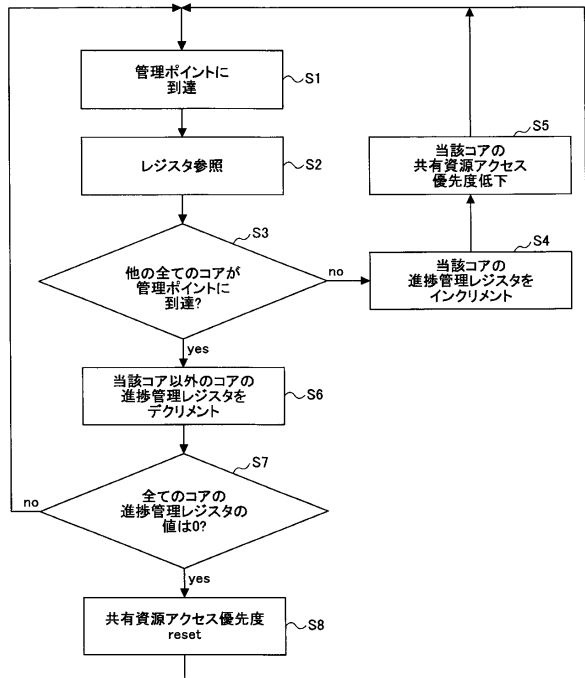
```

51
-barrier-
a=0d0
do k=1,nkpe
do j=1,ny
do i=1,nx
a=a+b(i,j,k)
end do
end do
report-progress(myrank,ngroupe)
end do
-barrier-
allreduce-sum(a,myrank,ngroupe)
52
53

```

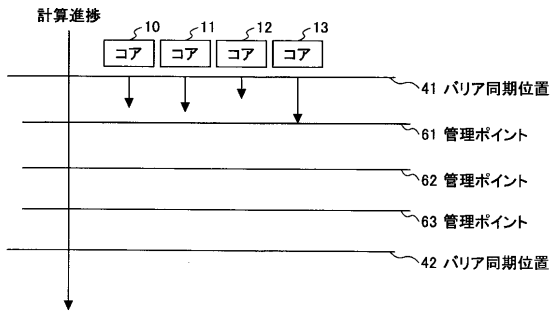
【図 4】

図1の演算処理装置の動作の一例を示すフローチャート



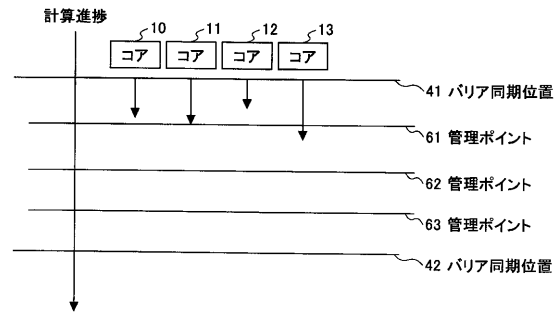
【図 5】

最速のコアが最初の管理ポイントに到達した状態の一例を示す図



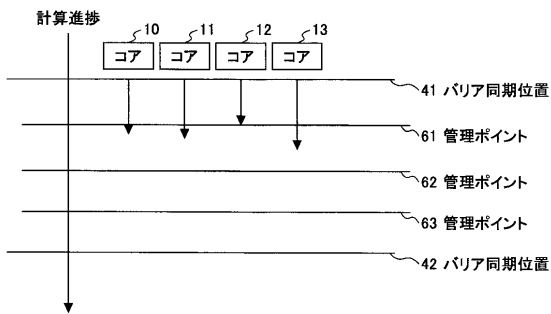
【図 6】

2番目に早いコアが最初の管理ポイントに到達した状態の一例を示す図



【図 7】

最も遅いコアが最初の管理ポイントに到達した状態の一例を示す図



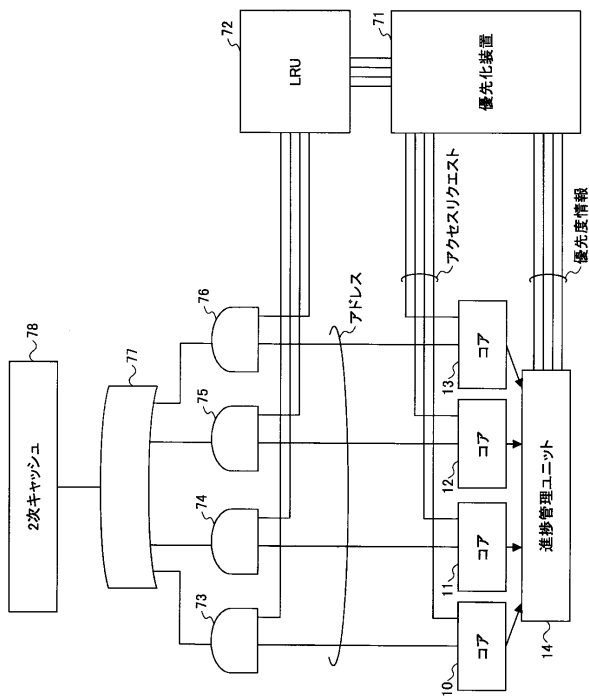
【図 8】

進捗管理レジスタのレジスタ値が変化する様子の一例を示す図

進捗管理レジスタ				
コア10	コア11	コア12	コア13	event
0	0	0	0	job start
0	0	0	1	コア13 管理ポイント到達
0	0	1	1	コア12 管理ポイント到達
0	1	1	1	コア11 管理ポイント到達
0	0	0	0	コア10 管理ポイント到達
0	0	1	0	コア12 管理ポイント到達
0	1	1	0	コア11 管理ポイント到達
0	1	2	0	コア12 管理ポイント到達
1	1	2	0	コア10 管理ポイント到達
0	0	1	0	コア13 管理ポイント到達
0	0	1	1	コア13 管理ポイント到達
1	0	1	1	コア10 管理ポイント到達
0	0	0	0	コア11 管理ポイント到達

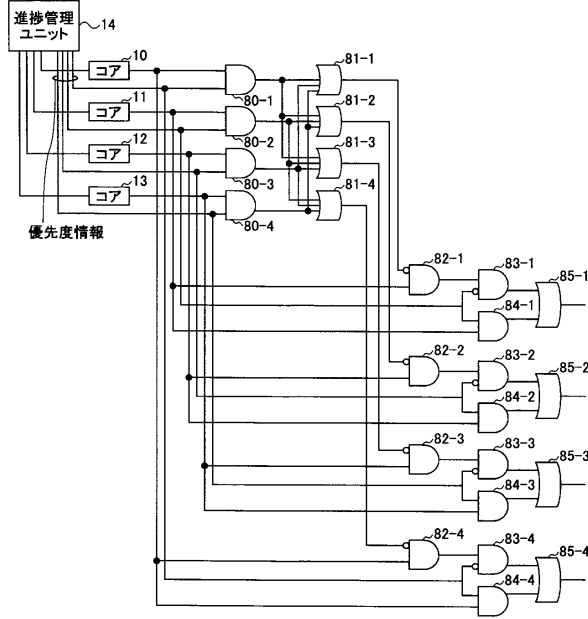
【図 9】

共有バス調停ユニットにおける共有リソースの割り当て機構の一例を示す図



【図 10】

優先化装置の構成の一例を示す図



【図 11】

優先度に応じたキャッシュのウェイの割り当ての一例を示す図

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3

【図 12】

優先度に応じたキャッシュのウェイの割り当ての一例を示す図

0	0	0	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3

【 図 1 4 】

優先度に応じたキャッシュのウェイの割り当ての一例を示す図

[illegible]

フロントページの続き

(56)参考文献 特開2009-230590(JP,A)
特開平07-271614(JP,A)
特開平07-248967(JP,A)
特開2004-038767(JP,A)
特開2012-128809(JP,A)
特開2009-025939(JP,A)

(58)調査した分野(Int.Cl., DB名)

G 0 6 F	9 / 5 2
G 0 6 F	9 / 3 0
G 0 6 F	9 / 5 0
G 0 6 F	1 5 / 1 6 7
G 0 6 F	1 5 / 1 7 3