(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2014-21774
(P2014-21774A)
(43)公開日　平成26年2月3日(2014.2.3)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/52	(2006.01)	G06F	9/46	475A	5B005
G06F 9/50	(2006.01)	G06F	9/46	462B	5B033
G06F 9/30	(2006.01)	G06F	9/30	350A	
G06F 12/08	(2006.01)	G06F	12/08	523B	

審査請求 未請求　請求項の数 6　ＯＬ　（全 15 頁）

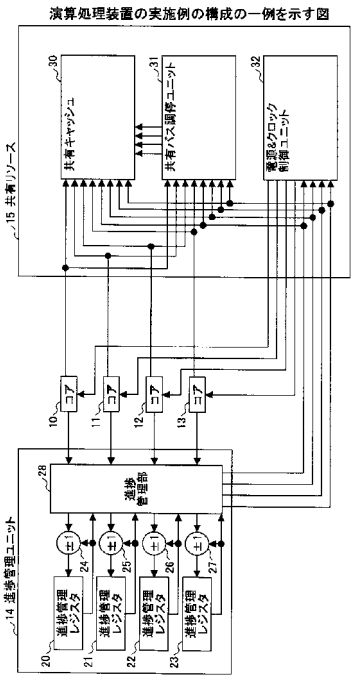
(21)出願番号　特願2012-160696 (P2012-160696)
(22)出願日　平成24年7月19日 (2012.7.19)

(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
(74)代理人　100107766
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

Ｆターム(参考)　5B005 KK13　MM01　TT02　UU42
5B033 BE00

(54)【発明の名称】演算処理装置及び演算処理方法

(57)【要約】
【課題】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を提供する。
【解決手段】演算処理装置は、演算処理を行う複数の演算処理部と、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、複数の演算処理部の各々について、演算処理部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、複数のレジスタのレジスタ値に応じて複数の演算処理部の優先度が変化されることを特徴とする。
【選択図】図１



【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、プログラム中の前記所定位置に挿入された所定のコマンドが実行されると、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】
前記複数の演算処理部のうちの一の演算処理部のプログラム実行箇所がプログラム中の前記所定位置に到達すると、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請求項１又は２記載の演算処理装置。
【請求項４】
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１乃至４何れか一項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、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、
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る各段階を含む演算処理方法。
【発明の詳細な説明】
【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。
【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。

【０００４】

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】

【０００５】

【特許文献１】特開２００７－１０８９４４号公報
【特許文献１】特開２００１－１３４４６６号公報

【発明の概要】

【発明が解決しようとする課題】

【０００６】

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】

【０００７】

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ることを特徴とする。

【発明の効果】

【０００８】

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】

【０００９】

【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。
【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の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１０】優先化装置の構成の一例を示す図である。
【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である。
【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である。

【図１３】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１４】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。
【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。
【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。
【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。
【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。
【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、この位置からコア１０乃至１３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４２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１０乃至１３間の次の同期が確立される。プログラム中の所定位置４３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２０乃至２３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４３は、例えば、コア１０乃至１３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４１とバリア同期位置４２との間の適当な位置に設けられている。複数のコア１０乃至１３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４１とバリア同期位置４２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。

【００１６】

図２の例では、コア１３が矢印４５で示されるように最初にプログラム中の所定位置４３に到達する。この時点での最速のコア１３と最も遅いコア１３とのプログラム実行の進捗度合いの差は、矢印４６の長さに相当する量である。コア１３のプログラム実行箇所がプログラム中の所定位置４３に到達した時点で、コア１３に対応する進捗管理レジスタ２３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２０乃至２３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２３のレジスタ値が残りの進捗管理レジスタ２０乃至２２のレジスタ値よりも大きくなると、進捗管理部２８は、コア１３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１３の優先度を下げるようにする。具体的には、進捗管理部２８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１５のリソース制御部が、コア１３よりも他のコア１０乃至１２を優先的に取り扱う。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。

【００１７】

上記のようにして、コア１３の優先度が下がることにより、コア１３のプログラム進行が遅くなる。その結果、コア１３のプログラム実行箇所がバリア同期位置４２に到達したときには、最速のコア１３と最も遅いコア１０とのプログラム実行の進捗度合いの差は、矢印４７の長さに相当する量となる。この量は、矢印４６が示す優先度調整の無い状態での最速のコア１３と最も遅いコア１０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１３のプログラム実行箇所がバリア同期位置４２に到達したときには、矢印４６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１３と最も遅いコア１０との間に発生していたことになる。

【００１８】

図３は、コア１０乃至１３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１０乃至１３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１０乃至１３のそれぞれが実行することにより、コア１０乃至１３がそれぞれの配列ｂの値の和ａを求め、最後のコマンド"allreduce-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５１は、最初のバリア同期命令である。バリア同期命令５１の位置は、図２に対応させると、バリア同期位置４１に相当する。プログラム中の命令５２は、２番目のバリア同期命令である。バリア同期命令５２の位置は、図２に対応させると、バリア同期位置４２に相当する。命令５３は、進捗管理ユニット１４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５３の位置は、図２に対応させると、プログラム中の所定位置４３に相当する。

【００１９】
進捗状況報告命令５３のパラメータｍｙｒａｎｋは、当該プログラムを実行するコアの番号を示す。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。

【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、この通知に応答して、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。

【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。

【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。

【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アではないので、ステップＳ４で当該コア１３の進捗管理レジスタ２３が１増加される。それに続きステップＳ５で、当該コア１３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へ必要な通知が行われる。図６に示す例の場合も、第１の管理ポイント６１に到達したコア１１は最も進捗の遅いコアではないので、当該コア１１の進捗管理レジスタ２１が１増加され、当該コア１１の共有リソース１５へのアクセスの優先度が低下される。
【００２５】
図４を再び参照し、ステップＳ３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Ｓ６に進む。ステップＳ６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Ｓ３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。
【００２６】
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。
【００２７】
図７の例の場合、第１の管理ポイント６１に到達したコア１２は最も進捗の遅いコアであるので、ステップＳ６において、当該コア以外のコア１０，１１，１３の進捗管理レジスタ２０，２１，２３を１減少させる。これにより、最も進捗の遅いコア１２に対応する進捗管理レジスタ２２のレジスタ値は０のままとなる。
【００２８】
図４を再び参照し、ステップＳ７において、進捗管理部２８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Ｓ８において、共有リソース１５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。
【００２９】
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１３が管理ポイントに到達し、コア１３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１２が管理ポイントに到達し、コア１２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１１が管理ポイントに到達し、コア１１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１０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１１乃至１３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１０乃至１３に対応する進捗管理レジスタの値は全て０になる。

【００３０】
その後、コア１２、コア１１、コア１２、コア１０がこの順番で管理ポイントに到達することにより、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。
【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に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。
【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。
【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】
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、ＬＲＵ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、ＬＲＵ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、ＬＲＵ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、ＬＲＵ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、ＬＲＵ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、ＬＲＵ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ＬＲＵ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、ＬＲＵ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ＬＲＵ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、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、ウェィ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内のコア数が増える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１４に、そのようにウェイを割り当てた例が示される。

【００４９】

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。

【００５０】

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。

【００５１】

例えば、進捗管理レジスタ２０乃至２３のレジスタ値の書き換えや優先度の調整は進捗管理部２８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１０乃至１３により分散的に実行されてもよい。例えば、各コア１０乃至１３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２０乃至２３のレジスタ値を直接に書き換えてよい。また各コア１０乃至１３が、進捗管理レジスタ２０乃至２３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。

【００５２】

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。

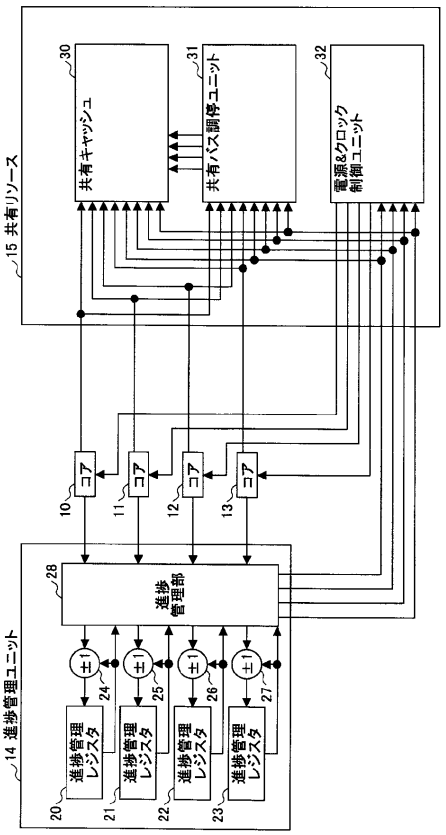
【符号の説明】

【００５３】

１０，１１，１２，１３　コア
１４　進捗管理ユニット
１５　共有リソース
２０，２１，２２，２３　進捗管理レジスタ
２４，２５，２６，２７　加減算器
２８　進捗管理部
３０　共有キャッシュ
３１　共有バス調停ユニット
３２　電源＆クロック制御ユニット

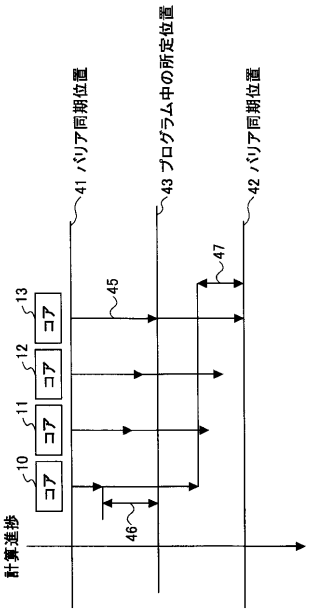
【図１】

演算処理装置の実施例の構成の一例を示す図



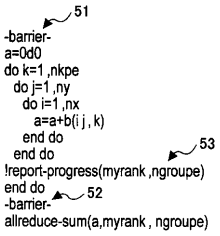
【図２】

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



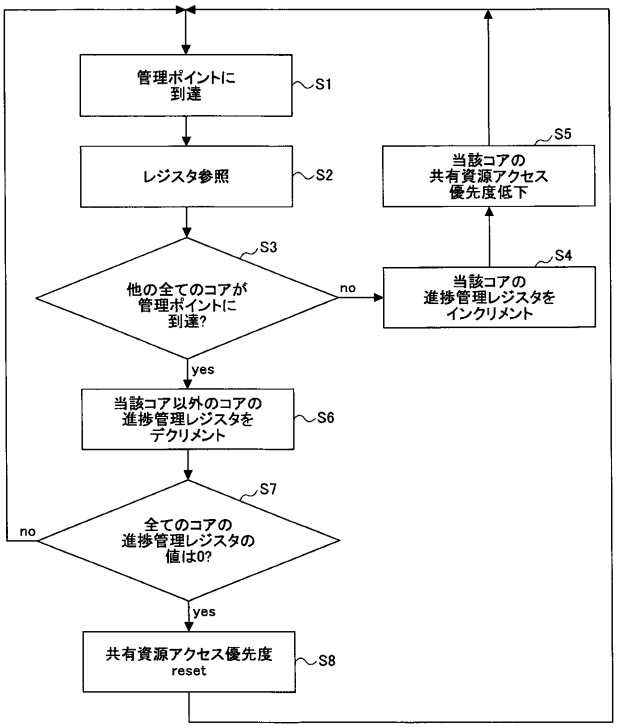
【図３】

コアが実行するプログラムの一例を示す図



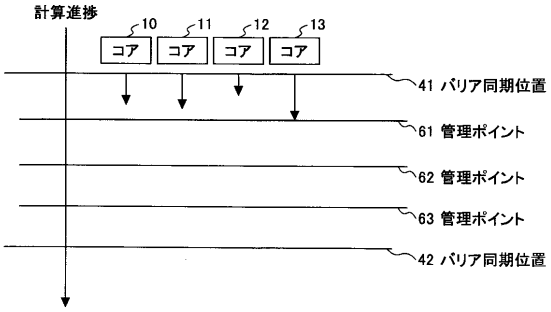
【図４】

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



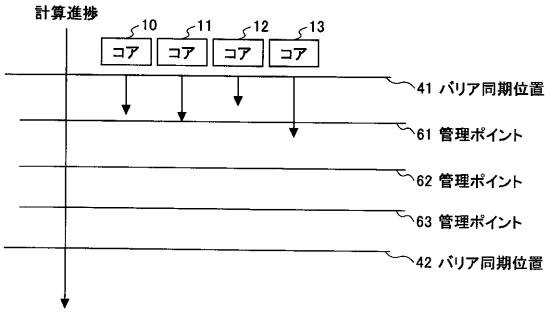
【図５】

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



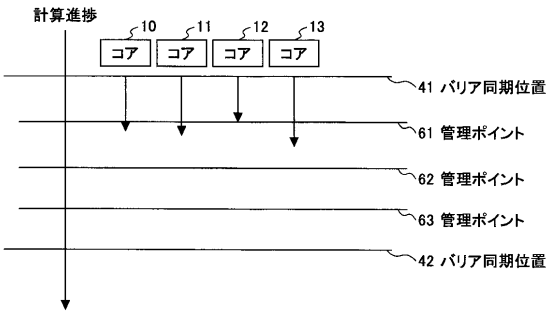
【図６】

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



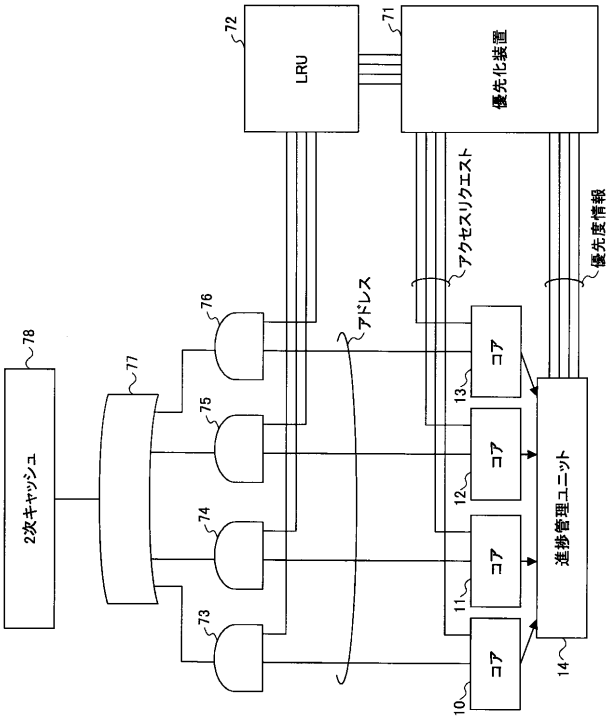
【図８】

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

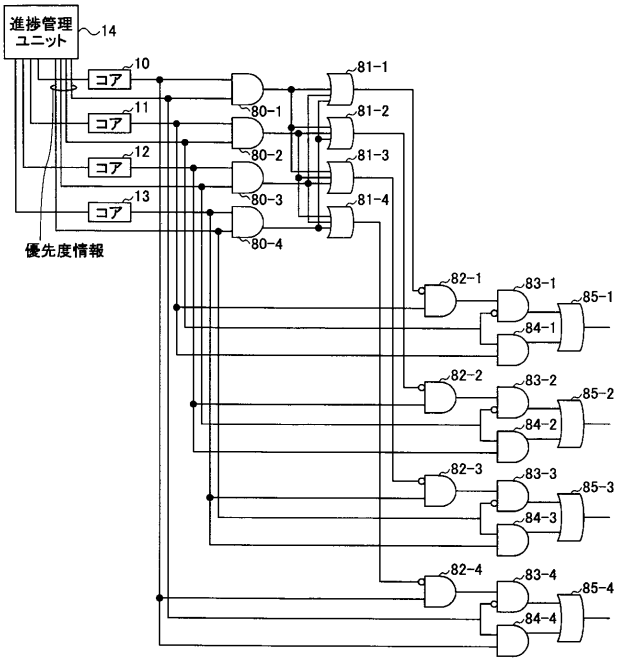
【図９】

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
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
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3

【図１３】

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
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
3	3	3	3
3	3	3	3

【図１４】

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
1	1	1	1
1	1	1	1
1	1	1	1
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
3	3	3	3
3	3	3	3
3	3	3	3