

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第5120850号
(P5120850)

(45) 発行日 平成25年1月16日 (2013. 1. 16)

(24) 登録日 平成24年11月2日 (2012. 11. 2)

(51) Int. Cl.

F I

G 0 6 F 12/08 (2006. 01)

G 0 6 F 12/08 5 3 1 B

G 0 6 F 12/08 5 3 1 C

G 0 6 F 12/08 5 1 9 Z

請求項の数 27 (全 38 頁)

(21) 出願番号	特願2008-504137 (P2008-504137)	(73) 特許権者	390009531
(86) (22) 出願日	平成18年3月17日 (2006. 3. 17)		インターナショナル・ビジネス・マシーンズ・コーポレーション
(65) 公表番号	特表2008-535093 (P2008-535093A)		INTERNATIONAL BUSIN
(43) 公表日	平成20年8月28日 (2008. 8. 28)		ESS MACHINES CORPO
(86) 国際出願番号	PCT/US2006/010038		RATION
(87) 国際公開番号	W02006/104747		アメリカ合衆国10504 ニューヨーク
(87) 国際公開日	平成18年10月5日 (2006. 10. 5)		州 アーモンク ニュー オーチャード
審査請求日	平成20年12月9日 (2008. 12. 9)		ロード
(31) 優先権主張番号	11/093, 130	(74) 代理人	100108501
(32) 優先日	平成17年3月29日 (2005. 3. 29)		弁理士 上野 剛史
(33) 優先権主張国	米国 (US)	(74) 代理人	100112690
			弁理士 太佐 種一
		(74) 代理人	100091568
			弁理士 市位 嘉宏

最終頁に続く

(54) 【発明の名称】 ストリーム・レジスタを用いてスヌープ要求をフィルタする方法、装置及びコンピュータ・プログラム

(57) 【特許請求の範囲】

【請求項 1】

複数のプロセッシング・ユニットを有するコンピューティング環境の各プロセッシング・ユニットに関連付けられたスヌープ・フィルタであって、前記複数のプロセッシング・ユニットの1つのプロセッシング・ユニットそれぞれは当該1つのプロセッシング・ユニットに対応する1つのスヌープ・フィルタに関連付けられており、前記複数のプロセッシング・ユニットの1つのプロセッシング・ユニットそれぞれは、当該1つのプロセッシング・ユニットに関連付けられた一つ以上のキャッシュ・メモリを有し、

前記スヌープ・フィルタは、

関連するプロセッシング・ユニットのキャッシュ・メモリ・レベル中にロードされたデータのキャッシュ・ライン・アドレスを追跡するようになされた第一メモリ格納手段であって、前記第一メモリ格納手段は、第一の複数ストリーム・レジスタ・セットを含み、各ストリーム・レジスタ・セットは、1つ又は複数のストリーム・レジスタを含み、各ストリーム・レジスタは、ベース・レジスタおよび対応するマスク・レジスタを含み、前記ベース・レジスタは、前記ストリーム・レジスタによって表される前記キャッシュ・ライン・アドレスすべてに共通するアドレス・ビットを追跡し、前記対応するマスク・レジスタは、これに対応するベース・レジスタに含まれる前に記録されたアドレスとの違いを表すビットを追跡し、当該違いを表すビットの位置は前記ベース・レジスタの対応するビットが判断外であることを示すように前記マスク・レジスタにおいて示される、前記第一メモリ格納手段と、

10

20

一つ以上のメモリ書き込み元からのスヌープ要求を受信する手段と、
受信したスヌープ要求のアドレスを、前記第一メモリ格納手段の前記複数のストリーム・レジスタ・セットに格納されたアドレス群と対比するためのスヌープ・チェック・ロジック手段と、

前記複数のストリーム・レジスタ・セットに格納されたアドレスとの一致を受けて、前記受信したスヌープ要求を前記プロセッシング・ユニットに転送し、又は、一致しない場合は前記スヌープ要求を廃棄するための手段と

を含む、前記スヌープ・フィルタ。

【請求項 2】

これにより、プロセッシング・ユニットに転送されるスヌープ要求の数が削減し、これによって前記コンピューティング環境のパフォーマンスを上げる、請求項 1 に記載のスヌープ・フィルタ。

【請求項 3】

前記スヌープ要求のメモリ書き込み元は、前記複数のプロセッシング・ユニットの一つ、又はダイレクト・メモリ・アクセス (DMA) エンジンを含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 4】

各前記スヌープ・フィルタは

複数のポート・スヌープ・フィルタと連通し、スヌープ要求のサブセットを受信し、エンキューし、そして、該スヌープ・フィルタの前記関連するプロセッシング・ユニットに転送する複数のプロセッサ・スヌープ・フィルタ・キュー手段

をさらに含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 5】

各前記スヌープ・フィルタは

前記スヌープ・フィルタ・キュー手段間をすべて調停し、前記複数のプロセッサ・スヌープ・フィルタ・キュー手段の各々から送られるすべてのスヌープ要求を逐次化し、前記関連するプロセッシング・ユニットに転送する手段

をさらに含む、請求項 4 に記載のスヌープ・フィルタ。

【請求項 6】

実施された各キャッシュ・ロードに対し、前記第一メモリ格納手段のキャッシュ・ライン・アドレスで更新するための手段をさらに含み、

前記更新するための手段は、レジスタ選定基準に基づきどのストリーム・ベース・レジスタ及びマスク・レジスタのセットを更新するかを判定するための手段を含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 7】

前記レジスタ選定基準は、前記ストリーム・レジスタ・セットと前記キャッシュ・ライン・ロード・アドレスとの間の最小ハミング距離、又は、前記ベース・レジスタの最上位部のビット群と前記キャッシュ・ライン・ロード・アドレスの同ビットとの最も近い一致を含む、請求項 6 に記載のスヌープ・フィルタ。

【請求項 8】

前記判定するための手段は、ライン・ロード・アドレスと、すべてのベース・レジスタの内容とを、その関連マスク・レジスタにより対比し、前記選定基準に従い最も近い一致を選定するための手段を含む、請求項 6 に記載のスヌープ・フィルタ。

【請求項 9】

前記第一メモリ格納手段のキャッシュ・ライン・アドレスで更新するための前記手段は、対セットの対応するマスク・レジスタの関連ビットを更新するための手段、又は、対セットの対応するベース・レジスタの内容を置換えるための手段を含む、請求項 6 に記載のスヌープ・フィルタ。

【請求項 10】

前記スヌープ・チェック・ロジック手段は、

キャッシュ・ライン内のオフセットに対応する低位のビットを除去することによって、受信したスヌープ・アドレスをスヌープ・ライン・アドレスに変換するための手段と、

ベース・レジスタの内容を、対応するマスク・レジスタの内容と組み合わせた前記スヌープ・ライン・アドレスと対比し、当該特定のキャッシュ・ライン・アドレスにおける前記キャッシュ・メモリ・レベル中にデータが存在する可能性を示す結果信号を得るロジックを実行するための第一手段と

を含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 1 1】

前記第一手段は、前記スヌープ・ライン・アドレスを、前記第一の複数ストリーム・レジスタ・セットの内容と並行して対比する、請求項 1 0 に記載のスヌープ・フィルタ。

10

【請求項 1 2】

前記スヌープ要求を受信する手段は、前記コンピューティング環境の専用メモリ書き込み元からのスヌープ要求を受信する複数の専用入力ポートを含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 1 3】

前記スヌープ・フィルタは、前記複数の専用入力ポートと連通し並列に動作する複数のポート・スヌープ・フィルタをさらに含む、

前記複数のポート・スヌープ・フィルタの各々がスヌープ・チェック・ロジック手段を実行して、それぞれの前記専用書き込み元から受信したスヌープ要求をフィルタリングし、同時にそれら要求のサブセットをその関連するプロセッシング・ユニットに転送する、請求項 1 2 に記載のスヌープ・フィルタ。

20

【請求項 1 4】

前記スヌープ・フィルタは、

前記複数の専用入力ポートと連通し並列に動作する複数のキュー・デバイスであって、前記複数のキュー・デバイスの各々はそれぞれの専用メモリ書き込み元から受信した受信スヌープ要求をエンキューする、キュー・デバイスと、

前記エンキューされたスヌープ要求を前記スヌープ・チェック・ロジック手段に転送する調停多重化手段と

をさらに含む、請求項 1 2 に記載のスヌープ・フィルタ。

【請求項 1 5】

30

前記キャッシュ・メモリ・レベル中のあらゆるキャッシュ・ラップが、前のキャッシュ・ラップ検知状態から置換えられたがどうかを追跡するためのキャッシュ・ラップ検出手段をさらに含む、請求項 1 に記載のスヌープ・フィルタ。

【請求項 1 6】

前記更新するための手段によってキャッシュ・メモリ中にロードされたデータのキャッシュ・ライン・アドレスを追跡するための、前記第一の複数ストリーム・レジスタ・セットと対称になる第二の複数ストリーム・レジスタ・セットを有する第二メモリ格納手段をさらに含む、

前記スヌープ・チェック・ロジック手段は、前記受信スヌープ要求の前記アドレスを前記第二の複数ストリーム・レジスタ・セット中に格納されたアドレス群と対比する手段を含む、

40

請求項 6 に記載のスヌープ・フィルタ。

【請求項 1 7】

前記第一の複数ストリーム・レジスタ・セットにより追跡されるキャッシュ・メモリ・ラインに対しキャッシュ・ラップ検知状態が検出されると、前記第二の複数ストリーム・レジスタ・セットの前記更新が行われる前に、前記第二の複数ストリーム・レジスタ・セットをリセットするための手段をさらに含む、請求項 1 6 に記載のスヌープ・フィルタ。

【請求項 1 8】

前記第二の複数ストリーム・レジスタ・セットにより追跡されるキャッシュ・メモリ・ラインに対しキャッシュ・ラップ検知状態が検出されると、前記第一の複数ストリーム・

50

レジスタ・セットをリセットするための手段をさらに含み、

前記第一の複数ストリーム・レジスタ・セットの更新が再実施される、

請求項 17 に記載のスヌープ・フィルタ。

【請求項 19】

前記第一の複数ストリーム・レジスタ・セットに対応する第二の複数ストリーム・レジスタ・セットを有する第二メモリ格納手段と、

全キャッシュ・ラインが置換えられたことを示すキャッシュ・ラップ状態の検出を受けて、前記第二の複数ストリーム・レジスタ・セットを前記第一の複数ストリーム・レジスタ・セットの内容に置き換えるための手段であって、前記スヌープ・チェック・ロジック手段は、前記受信したスヌープ要求の前記アドレスを、前記第二の複数ストリーム・レジスタ・セットとさらに対比する、前記置き換えるための手段とをさらに含む、請求項 1 に記載のスヌープ・フィルタ。

10

【請求項 20】

前記第二の複数ストリーム・レジスタ・セットが前記第一の複数ストリーム・レジスタ・セットの前記内容に置換えられた後、前記第一の複数ストリーム・レジスタ・セットをリセットするための手段をさらに含む、請求項 19 に記載のスヌープ・フィルタ。

【請求項 21】

前記第一メモリ格納手段は、関連するプロセッシング・ユニットのキャッシュ・メモリ・レベル中にロードされたデータのキャッシュ・ライン・アドレスを追跡するようになっており、前記キャッシュの対応する一つ以上の区分化サブセットに関する内容を有する一つ以上の第一の複数ストリーム・レジスタ・セットを含み、

20

前記キャッシュ・ラップ検出手段は、前記キャッシュの前記一つ以上の区分化サブセットそれぞれのあらゆるキャッシュ・ラインが、前の各キャッシュ・ラップ検知状態から置き換えられているかどうかを追跡する、

請求項 15 に記載のスヌープ・フィルタ。

【請求項 22】

各々が前記第二の複数ストリーム・レジスタ・セットとして動作する、一つ以上の第二メモリ格納手段と、

キャッシュ・ラップ状態の検出に先だって、前記第一の複数ストリーム・レジスタ・セットの内容を前記一つ以上の第二の複数ストリーム・レジスタ・セットの各々にコピーするための手段であって、前記第一の複数ストリーム・レジスタ・セットの内容は、より高い頻度で更新されリセットされる、前記コピーするための手段と

30

をさらに含む、請求項 21 に記載のスヌープ・フィルタ。

【請求項 23】

複数のプロセッシング・ユニットを有するコンピューティング環境でキャッシュ・コヒーレンスをサポートするためのスヌープ・フィルタリング方法であって、前記複数のプロセッシング・ユニットの 1 つのプロセッシング・ユニットそれぞれは当該 1 つのプロセッシング・ユニットに対応する 1 つのスヌープ・フィルタに関連付けられており、前記複数のプロセッシング・ユニットの 1 つのプロセッシング・ユニットそれぞれは、当該 1 つのプロセッシング・ユニットに関連付けられた一つ以上のキャッシュ・メモリを有し、

40

前記方法は、

プロセッシング・ユニット中の各スヌープ・フィルタに対し、

関連するプロセッシング・ユニットのキャッシュ・メモリ・レベル中にロードされたデータのキャッシュ・ラインのアドレスを追跡し、キャッシュ・ライン・アドレスを第一メモリ格納手段中に格納するステップであって、前記第一メモリ格納手段は、第一の複数ストリーム・レジスタ・セットを含み、各ストリーム・レジスタ・セットは、1 つ又は複数のストリーム・レジスタを含み、各ストリーム・レジスタは、ベース・レジスタおよび対応するマスク・レジスタを含み、前記ベース・レジスタは、前記ストリーム・レジスタによって表される前記キャッシュ・ライン・アドレスすべてに共通するアドレス・ビットを追跡し、前記対応するマスク・レジスタは、これに対応するベース・レジスタに含まれる

50

前に記録されたアドレスとの違いを表すビットを追跡し、当該違いを表すビットの位置は前記ベース・レジスタの対応するビットが判断外であることを示すように前記マスク・レジスタにおいて示される、前記格納するステップと、

一つ以上のメモリ書き込み元からスヌープ要求を受信するステップと、

受信したスヌープ要求のアドレスを、前記第一メモリ格納手段の前記複数のストリーム・レジスタ・セットに格納されたアドレス群と対比するステップと、

前記複数のストリーム・レジスタ・セットに格納されたアドレスとの一致を受けて、前記受信したスヌープ要求を前記プロセッシング・ユニットに転送し、又は一致しない場合は前記スヌープ要求を廃棄するステップと

を含む、前記方法。

10

【請求項 2 4】

これにより、プロセッシング・ユニットに転送されるスヌープ要求の数が削減し、これによって前記コンピューティング環境のパフォーマンスを上げる、請求項 2 3 に記載の方法。

【請求項 2 5】

前記各スヌープ・フィルタは、前記メモリ書き込み元の各々と連通する複数のポート・スヌープ・フィルタを含み、

前記受信したスヌープ要求を転送するステップは、

前記スヌープ・フィルタの前記関連するプロセッシング・ユニットに転送予定のスヌープ要求のサブセットを、ポート・スヌープ・フィルタに対応するそれぞれのプロセッサ・スヌープ・フィルタ・キュー手段中にエンキューするステップ

20

を含む、請求項 2 3 又は 2 4 に記載の方法。

【請求項 2 6】

前記第一メモリ格納手段を、実施された各キャッシュ・ロードに対するキャッシュ・ライン・アドレスで更新するステップをさらに含む、請求項 2 3 又は 2 4 に記載の方法。

【請求項 2 7】

複数のプロセッシング・ユニットを有するコンピューティング環境においてコンピュータにキャッシュ・コヒーレンスをサポートするためのスヌープ・フィルタリングのためのコンピュータ・プログラムであって、前記複数のプロセッシング・ユニットの 1 つのプロセッシング・ユニットそれぞれは当該 1 つのプロセッシング・ユニットに対応する 1 つのスヌープ・フィルタに関連付けられており、前記複数のプロセッシング・ユニットの 1 つのプロセッシング・ユニットそれぞれは、当該 1 つのプロセッシング・ユニットに関連付けられた一つ以上のキャッシュ・メモリを有し、前記コンピュータに請求項 2 3 ～ 2 6 のいずれか一項に記載の方法の各ステップを実行させる前記コンピュータ・プログラム。

30

【発明の詳細な説明】

【技術分野】

【0 0 0 1】

本発明は、一般的に多重プロセッサ・アーキテクチャを有するコンピュータ・システムに関し、さらに具体的には、かかる多重プロセッサ・システムにおけるメモリ・アクセス要求を処理しキャッシュ・コヒーレンスを実現するための、新しい多重プロセッサ・コンピュータ・システムに関する。

40

【背景技術】

【0 0 0 2】

高いパフォーマンスのコンピューティングを実現するために、複数の個別プロセッサが相互連結され、並行処理が可能な多重プロセッサ・コンピュータ・システムが形成されている。複数のプロセッサを単一のチップ又は数個のチップに配置し、各チップに一つ又は数個のプロセッサを包含させ、多重プロセッサ・コンピュータ・システムとして相互連結させることができる。

【0 0 0 3】

多重プロセッサ・コンピュータ・システム中のプロセッサは、自己専用のキャッシュ・

50

メモリを使用し、アクセス・タイムの短縮（キャッシュは、プロセッサに対しローカルで、データへの高速なアクセスが得られる）と主メモリに対するメモリ請求の回数低減とを図っている。しかしながら、多重プロセッサ・システムのキャッシュの管理は複雑である。複数の専用キャッシュは、主メモリのデータの複数のコピーが多重プロセッサ・システム中に同時に存在し得ることに起因する、多重キャッシュのコヒーレンシ問題（又はデータ陳腐化問題）を引き起こす。

【0004】

小規模の共用メモリ多重処理システムは、単一のバスで相互連結された複数プロセッサ（又はその群）を有する。しかし、プロセスの速度が向上するにつれ、バスを効果的に共用できるプロセッサの実施可能数が減少する。

10

【0005】

複数プロセッサ間のコヒーレンスを維持するプロトコルは、キャッシュ・コヒーレンス・プロトコルと呼ばれる。キャッシュ・コヒーレンス・プロトコルは、プロセッサ間の一切のデータ・ブロックの共用を追跡する。どのようにデータ共用を追跡するかによって、キャッシュ・コヒーレンス・プロトコルを2つのクラスに分類することができる。すなわち、1) ディレクトリ・ベースと、2) スヌーピングである。

【0006】

ディレクトリ・ベースのアプローチでは、物理メモリのブロックの共用状態は、コヒーレンシ・ディレクトリと呼ばれるただ一つのロケーションに置かれる。コヒーレンシ・ディレクトリは、一般に、多重プロセッサ・コンピュータ・システム中のどのプロセッサが、どのメモリのラインを有しているかを追跡記録する大きなメモリ・ブロックである。不都合なことに、コヒーレンシ・ディレクトリは、通常、大きくて遅い。これらは、全体的システム・パフォーマンスを大きく低下させる。というのは、これらのディレクトリでは、メモリへの各アクセスが共用ディレクトリを通過することが必要なので、あらゆるメモリ・アクセス要求に対し余分な待ち時間をもたらすからである。

20

【0007】

図1は、キャッシュ・コヒーレンシに対しコヒーレンシ・ディレクトリによるアプローチを用いた、典型的な従来技術の多重プロセッサ・システム10を示す。多重プロセッサ・システム10は、共用バス24を介し、それぞれメモリ・コントローラ22a、22bを通して主メモリ20a、20bに相互接続されたいくつかのプロセッサ15a、…、15dを含む。各プロセッサ15a、…、15dは、それぞれ自分専用のキャッシュ17a、…、17dを有し、Nウェイ・セット・アソシアティブである。プロセッサからのメモリに対する各要求は、プロセッサ・バス24上に置かれ、コヒーレンシ・コントローラ26に向けられる。多くの場合、コヒーレンシ・コントローラ中には、特定のサブシステム中に保持されているキャッシュ・ラインのロケーションを追跡するモジュールが包含されており、キャッシング・エージェントすべてに不必要なスヌープ要求を一せいで同報する必要性をなくす。このユニットはしばしば「スヌープ・コントローラ」又は「スヌープ・フィルタ」と呼ばれる。また、I/Oシステム28からの全メモリ・アクセス要求もコヒーレンシ・コントローラ26に向けられる。主メモリの代わりに、主メモリに接続された二次的なキャッシュを使うことができる。プロセッサを、プロセッサ・クラスタにグループ化し、各クラスタに自己のクラスタ・バスを持たせ、これをコヒーレンシ・コントローラ26に連結することができる。各メモリ要求がコヒーレンシ・ディレクトリを通過する際に、各要求に対し、要求されるメモリ・ブロックの状態をチェックするための追加サイクルが加えられる。

30

40

【0008】

スヌーピング・アプローチにおいては、集中体制は取られず、代わりに各キャッシュが、データ・ブロックの共用状態をローカルに保存する。キャッシュは、通常、共用メモリ・バスにつながっており、すべてのキャッシュ・コントローラはバスを探り（モニタし）、コントローラが、要求するデータ・ブロックのコピーを持っているかどうかを判断する。一般に用いられるスヌーピング方法は、「write-invalidate」プロト

50

コルである。このプロトコルでは、プロセッサは、データに書き込みをする前に、当該データに対して自分だけがアクセスすることを確実にする。書き込みをする毎に、他のすべてのキャッシュ中の当該データの他のコピー全部が無効にされる。2つ以上のプロセッサが同時に同一のデータに書き込みをしようとする、その一つだけが勝ち残り他のプロセッサのコピーは無効にされる。

【0009】

`write-invalidate`プロトコル・ベースのシステムにおいて書き込みを行うため、プロセッサは共用バスを必要とし、バスを介して無効にされるアドレスを一せいで同報する。すべてのプロセッサは、バスをスヌープして、当該データが自分のキャッシュ中にあるかどうかをチェックする。ある場合、それらデータは無効にされる。かくして、共用バスを使って書き込みの逐次化が行われる。

10

【0010】

不都合なこととして、スヌーピング・アプローチにおけるあらゆるバス・トランザクションでは、キャッシュ・アドレス・タグを確認しなければならず、これがCPUのキャッシュ・アクセスを妨げる可能性がある。最新式のアーキテクチャでは、通常、アドレス・タグを複製し、CPUの要求とスヌーピング要求とが並行して進められるようにして、これを低減している。別のアプローチでは、内包型の多層レベル・キャッシュを用い、一次キャッシュへのあらゆる入力を低レベルのキャッシュ中に複製する。こうして、スヌープ作業を二次レベルのキャッシュで行い、CPUの作業を妨げないようにする。

20

【0011】

図2は、キャッシュ・コヒーレンシのためスヌーピング・アプローチを使った典型的な従来技術の多重プロセッサ・システム50を示す。多重プロセッサ・システム50は、共用バス56を介して主メモリ58に相互接続されたいくつかのプロセッサ52a、…、52cを含む。各プロセッサ52a、…、52cは、それぞれ自分専用のキャッシュ54a、…、54cを有し、Nウェイ・セット・アソシアティブである。プロセッサからのメモリに対する各書き込み要求は、プロセッサ・バス56上に置かれる。すべてのプロセッサは、バスをスヌープし、自分のキャッシュをチェックして書き込み対象アドレスが自分のキャッシュ中に所在しているかどうかを見る。所在する場合、当該アドレスに対応するデータは無効にされる。いくつかの多重プロセッサ・システムでは、各プロセッサに対しローカルにモジュールを付加し、無効にされるキャッシュ・ラインが特定のキャッシュ中に保持されているかどうかを追跡させ、ローカルでのスヌーピング作業を効果的に削減している。このユニットは、しばしば「スヌープ・フィルタ」と呼ばれる。主メモリの代わりに、主メモリに連結された二次キャッシュを使うことができる。

30

【0012】

バスにつながるプロセッサ数が増えるにつれ、同様にスヌーピング作業も増加する。キャッシュに対する不必要なスヌープ要求は、プロセッサのパフォーマンスを低下させ、各スヌープ要求のキャッシュ・ディレクトリへのアクセスによりパワーが消費される。さらに、スヌーピング作業を支援するため、あらゆるプロセッサに対しキャッシュ・ディレクトリを複製することは、チップのサイズを大幅に増大させる。このことは、単一チップ上に設けられたパワー量の限定されたシステムに対しては特に重要である。

40

【0013】

以下は、多重プロセッサ・システムで判明した、従来のスヌーピング・アプローチのさまざまな問題に取り組んだ従来技術の参考文献の内容である。

【0014】

具体的には、米国特許出願公開第2003/0135696(A1)号、及び米国特許第6,704,845(B2)号の双方は、スヌープ・フィルタを含むコヒーレンス・ディレクトリ・ベースのアプローチのための、スヌープ・フィルタのエントリを置換える置換え基準を記載している。このスヌープ・フィルタは、キャッシュ格納されたメモリ・ブロックに関する情報を包含しており、キャッシュ・ラインとその状態がキャッシュ格納されている。米国特許出願公開第2004/0003184(A1)号は、遠隔ノードにア

50

クセスされるローカル・キャッシュ・ラインを記録する偶数番及び奇数番のアドレス・ラインを記録するためのサブスヌープ・フィルタを包含するスヌープ・フィルタを記載している（サブフィルタは、同じフィルタリング・アプローチを使う）。これらの開示のいずれも、多重プロセッサ・システムにおいて、各キャッシュに送られるスヌープ要求の数をローカル的に低減するためのシステム及び方法については教示も提案もしていない。また、これらは、各種のフィルタリング方法を使っていくつかのスヌープ・フィルタを結合することについても、教示も提案もしておらず、さらに、スヌーピング情報の2点間相互接続をキャッシュに具えることも教示も提案もしていない。

【0015】

米国特許出願公開第2003/0070016(A1)号及び米国特許出願公開第2003/0005843(A1)号は、スヌープ・フィルタを包含する中央コヒーレンシ・ディレクトリを備えた多重プロセッサ・システムについて記載している。これらの出願に記載されたスヌープ・フィルタはスヌープ要求を処理するサイクル数を減少させるが、キャッシュに送られるスヌープ要求の数は低減しない。

【0016】

米国特許第5,966,729号は、キャッシュ・コヒーレンスのためのスヌーピング・アプローチと各プロセッサ群にローカルに関連付けられたスヌープ・フィルタとを使う、共用バスの多重プロセッサ・システムを記載している。スヌーピング作業を削減するため、特定キャッシュ・ラインに「関与のある」遠隔プロセッサ群と「関与のない」群とのリストが保持される。スヌープ要求は「関与のある」とマークされたプロセッサ群にだけ送られ、これにより一せい同報されるスヌープ要求の数は削減される。該出願は、ローカルのプロセッサに対するスヌープ要求の数をいかに削減するかでなく、むしろ、「関与のない」とマークされた他のプロセッサ群に送られるスヌープ要求の数をいかに低減するかを記載している。この対処策は、プロセッサ群に対するキャッシュ中の各ライン毎にこれに関与するプロセッサ群の情報を含むリストを保持することが必要となり、これは、サイズにおいて当該プロセッサ群中の各プロセッサのキャッシュ・ディレクトリを複製するのに匹敵し、従ってチップのサイズを大幅に拡大する。

【0017】

米国特許第6,389,517(B1)号は、2つのアクセス・キューを具え、プロセッサ及びスヌープ・アクセスの両方からのキャッシュへの同時アクセスを可能にする、キャッシュ・コヒーレンスをスヌーピングする方法を記載している。開示された実施形態は共用バス構成を対象としている。該特許は、キャッシュに渡されるスヌープ要求の数を減少させる方法については記載していない。

【0018】

米国特許第5,572,701号は、低速のバスの高速バス及びプロセッサへの妨害を低減するために、バス・ベースのスヌープ方法を記載している。スヌープ・バス制御ユニットは、プロセッサが高速バスへのリリースを完了するまで、低速のバスからのアドレス及びデータを一時格納する。該ユニットは、その後にデータを転送し、キャッシュ中の対応するラインを無効にする。この開示は、全構成要素が高速バスを介して交信する多重プロセッサ・システムについては記載していない。

【0019】

A. Moshovos (モシュボス)、G. Memik (メミク)、B. Falsafi (フォルサフィ) 及び A. Choudhary (チョードリ) は、「JETTY: filtering snoops for reduced energy consumption in SMP servers (JETTY: SMPサーバにおける低エネルギー消費のためのフィルタリング・スヌープ)」(「Jetty」) という題名の文献の中で、ハードウェア・フィルタを用いてスヌープ要求を削減するためのいくつかの提案を記載している。該文献は、スヌープ要求が共用システム・バスを介して配送される多重プロセッサ・システムを記載している。プロセッサに送られるスヌープ要求の数を削減するため、一つ又は数個のさまざまなフィルタが用いられる。

【発明の開示】

【発明が解決しようとする課題】

【0020】

しかしながら、J e t t y に記載されたシステムは、パフォーマンス、支援システム、さらに具体的には相互接続アーキテクチャに関し重大な限界があり、マルチポート化への対応が欠けている。さらに具体的には、J e t t y に記載されたアプローチは、システム全体に亘って共通イベント順序付けを設定した共用システム・バスに基づいている。このようなグローバルな時間順序付けは、フィルタ・アーキテクチャを単純化する上では望ましいが、単一共用バスを使うシステムに対しては可能なシステム構成を制限してしまう。残念なことに、共用バス・システムは、単一のグローバル資源に対する競合に起因して、拡張性に限界があることが知られている。さらに、グローバルなバスは、これらに付属する複数の構成要素の大きな負荷に起因して低速になりがちで、チップ型多重プロセッサ中に配置するのは非効率的である。

10

【0021】

従って、高度に最適化された高帯域システムにおいては、スター型、又は2点間実装のような別のシステム・アーキテクチャを具えるのが望ましい。これらには利点がある、というのは、これらは、単一組の送信者と伝送者とを有し、負荷を低減し、高速プロトコルの使用を可能にし、チップ多重プロセッサのレイアウトを単純化するからである。また、2点間プロトコルを使うことによって、いくつもの送信を同時に進行させることができ、これにより、データ伝送の並行処理及び全体的データ・スループットが増大する。

20

【0022】

J e t t y の他の限界には、いくつもの要求に対するスヌープ・フィルタリングを同時に実施できないことが含まれ、J e t t y にあるように、いくつものプロセッサからの同時のスヌープ要求は、システム・バスがそれらを逐次化しなければならない。いくつものスヌープ要求の同時処理を可能にすることによって、一度に取り扱える要求の数の大幅な増加が得られ、これにより全体的システム・パフォーマンスが向上されよう。

【0023】

従来技術の限界を示してきたが、必要とされるものは、システム設計の選択肢を制限することなく、スヌープ・フィルタを組み込み、全体的パフォーマンス及びパワー効率を増進させるシステム、さらに具体的には、共用バスを必要としないシステムにおいてスヌープ・フィルタリングをサポートする方法及び装置であることは明らかである。

30

【0024】

さらに、2点間接続を用いるシステムをサポートし、スヌープ・フィルタリングを用いた高パフォーマンスのシステム実行を可能にする、スヌープ・フィルタ・アーキテクチャが必要である。

【0025】

複数のスヌープ・フィルタ・ユニットを同時に動作させて、複数メモリ書き込み元からの要求を同時にフィルタし、システム・パフォーマンスを向上させることがさらに求められる。

【0026】

パイプライン方式で実行できる新規の高パフォーマンス・スヌープ・フィルタを実現し、このようなスヌープ・フィルタを利用するシステムにおいて高いシステム・クロック速度を可能にすることがさらに求められる。

40

【0027】

従来技術の限界を乗り越えた高いフィルタリング効率持つスヌープ・フィルタがさらに求められる。

【0028】

従って、本発明の目的は、キャッシュ・コヒーレント多重プロセッサ・システムにおいて、一つのプロセッサに送られるスヌープ要求の数を減らすための簡単な方法及び装置を提供することである。

50

【0029】

本発明のさらなる目的は、入来するスヌープ要求をフィルタする簡単なハードウェア・デバイスを、各プロセッサにローカル的に加えることによって、ローカルでのスヌーピング作業を効果的に削減する方法及び装置を提供することである。前述のスヌープ・フィルタは、プロセッシング・ユニットに関連付けられたローカル・キャッシュ・メモリ中にロードされたデータを追跡するためにストリーム・レジスタを活用し、キャッシュ・ミスとなる可能性の高いスヌープ要求を識別し、そのほとんどをフィルタ除去するが、ローカルにキャッシュ格納する対象としてストリーム・レジスタに示されているデータに対するスヌープ要求を除去することはしない。プロセッサあたりのスヌープ要求の数を低減することによってシステム・パフォーマンスが向上されパワーが節減される。

10

【課題を解決するための手段】

【0030】

本発明の第一の態様によれば、複数のプロセッシング・ユニットを備えたコンピューティング環境中の単一のプロセッシング・ユニットに関連付けられたスヌープ・フィルタ装置が提供され、各プロセッシング・ユニットはそれに関連付けられた一つ以上のキャッシュ・メモリを有し、該スヌープ・フィルタは、

関連するプロセッサのキャッシュ・メモリ・レベル中にロードされているデータのキャッシュ・ライン・アドレスを追跡するようになされた第一メモリ格納手段と、

一つ以上のメモリ書き込み元からのスヌープ要求を受信するための手段と、

受信したスヌープ要求のアドレスを前記メモリ格納手段に格納されているアドレスと対比する手段と、

20

メモリ格納手段中のアドレスとの一致を受けて受信したスヌープ要求を転送するか、さもなくば、スヌープ要求を廃棄するための手段と、
を含み、

これらによりプロセッシング・ユニットに転送されるスヌープ要求の数が大幅に削減され、それによって、コンピューティング環境のパフォーマンスが増進される。

【0031】

さらに具体的には、スヌープ・フィルタは各プロセッシング・ユニットに関連付けられており、ストリーム・レジスタ・セットと関連するストリーム・レジスタ対比ロジックとの使用に基づいてフィルタリング方法を実行する少なくとも一つのスヌープ・フィルタ基本要素を含む。どの時点においても、複数のストリーム・レジスタ・セットの中の少なくとも一つのストリーム・レジスタ・セットはアクティブであり、少なくとも一つのストリーム・レジスタ・セットは履歴と標識される。さらに、スヌープ・フィルタ・ブロックは、キャッシュ・ラップ検知ロジックと動作可能に結合され、これにより、キャッシュ・ラップ状態が検知されると、アクティブなストリーム・レジスタ・セットの内容は履歴ストリーム・レジスタ・セットに移し替えられ、少なくとも一つのアクティブなストリーム・レジスタ・セットの内容がリセットされる。各フィルタ基本要素は、ストリーム・レジスタ対比ロジックを実行し、受信したスヌープ要求をプロセッサに転送するかあるいは廃棄するかを判断する。受信されたスヌープ要求は、ストリーム・レジスタと対比され、判定が行われ、キャッシュ中にエントリがあり得るかどうかを示されるが、その実際のレジデンス状態は示されない。

30

40

【0032】

さらに、ストリーム・レジスタ・セットに基づくフィルタ基本要素は、複数の、ベース・レジスタとマスク・レジスタとの対セットを含む。前述のスヌープ・フィルタ・ユニットが関連付けられたキャッシュ階層中にロードされる各新規データに対し、メモリ要求のアドレスは、きっちりと一つのベース・レジスタに書き込まれ、対となるマスク・レジスタは更新される。本発明によれば、選定されたストリーム・レジスタのマスク・レジスタは更新されて、値弁別性を減ずることによって、前に記録されていたアドレスとの差異を追跡記録する。

【0033】

50

本発明の第二の態様によれば、複数のプロセッシング・ユニットを有するコンピューティング環境においてキャッシュ・コヒーレンシをサポートするためのスヌープ・フィルタリング方法が提供され、各プロセッシング・ユニットは、それに関連付けられた一つ以上のキャッシュ・メモリ、及び関連するスヌープ・フィルタ・デバイスを有し、該方法は、プロセッシング・ユニット中の各スヌープ・フィルタ・デバイスについて、

関連するプロセッサのキャッシュ・メモリ・レベル中にロードされているデータのキャッシュ・ライン・アドレスを追跡記録し、キャッシュ・ライン・アドレスを第一メモリ格納手段に保存するステップと、

複数のメモリ書き込み元からスヌープ要求を受信するステップと、

受信したスヌープ要求のアドレスをメモリ格納手段中に格納されたアドレスと対比するステップと、

メモリ格納手段中のアドレスとの一致を受けて、受信したスヌープ要求をプロセッサに転送するか、さもなければ、スヌープ要求を廃棄するステップと、
を含み、

これにより、プロセッシング・ユニットに転送されるスヌープ要求の数が大幅に削減され、コンピューティング環境のパフォーマンスが向上する。

【0034】

本発明は、有益にいくつかの要求のスヌープ・フィルタリングを同時に実施することを可能にするが、従来技術のシステムでは、いくつかのプロセッサからの同時のスヌープ要求についてはシステム・バスがこれらを逐次化しなければならない。いくつかのスヌープ要求の同時処理を可能にすることによって、一度に取り扱える要求の数が大幅に増加でき、これにより全体的システム・パフォーマンスが向上される。

【0035】

当業者は、添付の図面と組合せ、以下の詳細説明を考察することにより、本発明の目的、特徴及び利点を明確に理解できよう。

【発明を実施するための最良の形態】

【0036】

ここで、図面類さらに具体的には図3を参照すると、キャッシュ・コヒーレンシに対する、スヌーピング・アプローチを用いた多重プロセッサ・システムの総体的なベース・アーキテクチャが示されている。この好適な実施形態において、多重プロセッサ・システムは、各自のローカルL1データ及び命令用のキャッシュと、その各々に関連付けられたL2キャッシュ120a、…、120nとを備えたNケのプロセッサ100a、…、100n（又はDCU1～DCUNと名付けられたCPU群）から成る。主メモリ130は共用され、これをオンチップ又はチップ外に実装することができる。別の実施形態において、主メモリの代わりに、主メモリにアクセスを有する共用L3を使うことができる。該好適な実施形態において、プロセッサ・コア100a、…、100nは、PPC440又はPPC405のようなPowerPC（IBM社の登録商標）コアであるが、本発明の範囲から逸脱することなく他の任意のコアを用いることができ、あるいは、一つの多重プロセッサ・システムの中にさまざまなプロセッサを組み合わせ用いることもできる。プロセッサ・コア100a、…、100nは、システム・ローカル・バス150によって相互連結されている。

【0037】

プロセッサに渡されるスヌープ要求の数を減らし、これにより、プロセッサ及びシステム・パフォーマンスに対するスヌーピングの影響を低減し、不必要なスヌープ要求によって消費されるアワーを削減するために、多重プロセッサ・システム10中の各プロセッサ・コア100a、…、100nのそれぞれに対しスヌープ・フィルタ140a、…、140nが具えられている。スヌーピング要求を伝送するため、該好適な実施形態は、通常従来技術のシステムで見られるようなシステム・バス150は使用せず、代わりに2点間相互接続160を実施し、これにより、各プロセッサの関連スヌープ・フィルタは、システム中の他のあらゆるプロセッサに関連付けられた各々のスヌープ・フィルタに直接連結さ

れる。かくして、スヌープ要求は、システム・ローカル・バスを介して伝送される他のメモリ要求のすべてから切り離され、しばしばシステムのボトルネックとなるバスの混雑を軽減する。一つのプロセッサに対するすべてのスヌープ要求は、そのスヌープ・フィルタ 140a、…、140nに送られ、これらのフィルタについては、後記でさらに詳細を説明するように、同一のフィルタリング方法を具備した、又はいくつもの異なるフィルタリング方法を具備した、あるいはその2つの任意の組み合わせのいくつかのサブフィルタを含む。スヌープ・フィルタは各スヌープ要求を処理し、全要求のうち、プロセッサのキャッシュの中にある可能性のある一部だけをプロセッサに渡す。

【0038】

各プロセッサのスヌープ要求は、2点間相互接続160を用いて他のすべてのプロセッサのスヌープ・フィルタに直接接続される。これにより、異なったプロセッサからのいくつものスヌープ要求（書き込み及び無効化の試みからくる）を同時に行うことができる。システム・バスを使いバスが要求の逐次化を行う典型的なスヌーピング・アプローチと違って、これらの要求はもはや逐次化されない。すなわち、後記でさらに詳細を説明するように、複数のスヌープ要求をスヌープ・フィルタで同時に処理することが可能である。プロセッサは一つのスヌープ・ポートしか持たないので、スヌープ・フィルタに除去されなかったスヌープ要求はプロセッサに送られる順にキューに並べられる。しかしながら、プロセッサに渡される要求の数は、フィルタ前の全スヌープ要求の数よりもずっと少なく、キャッシュ・コヒーレンス実行がシステム・パフォーマンスに与える影響を軽減する。

【0039】

スヌープ・フィルタ・ブロック中に包含されるキューのキュー・オーバーフロー状態を防止するために、各々の2点間リンクに対し、トークンベースのフロー制御システムを実施し、同一時に競合する要求の数を制限する。該トークンベース・フロー制御によれば、各書き込み元は、それが直接2点間接続を有するスヌープ・フィルタ・ブロックの全ポートに対して使えるトークンさえ持っていれば、次の書き込み要求を送信することができ、該要求は、他のすべてのプロセッシング・ユニット及び関連するスヌープ・フィルタ・ブロックに対するスヌープ要求をも起動する。接続されている遠隔ポートの少なくとも一つに対し利用できるトークンがない場合、そのスヌープ・フィルタ・ポートに対し少なくとも一つのトークンが再度利用できるまでは、当該メモリ書き込み元からスヌープ要求を発信することはできない。

【0040】

図4は、キャッシュ・コヒーレンシのため、スヌーピング要求のための2点間相互接続を使うスヌーピング・アプローチを用いたベース多重プロセッサ・システムを含む本発明の別の実施形態を示しており、該実施形態では、スヌープ・フィルタは、L2キャッシュと主メモリ230との間に配置されている。この実施形態による多重プロセッサ・システムは、各々のデータ及び命令用ローカルL1キャッシュと、関連するL2キャッシュ220a、…、220nとを備えたNヶのプロセッサ200a、…、200n（又はDCU1～DCUNと名付けられたCPU群）を含む。主メモリ230は共用され、これをオンチップ又はチップ外に実装することができる。別の実施形態において、主メモリの代わりに、主メモリにアクセスを有する共用L3キャッシュを使うことができる。プロセッサ200a、…、200nからのメモリ・アクセス要求のすべては、システム・ローカル・バス250を介して伝送される。図4に示された実施形態において、多重プロセッサ・システム中のプロセッサの各々は、それぞれのスヌープ・フィルタ240a、…、240nと対になっている。この好適な実施形態では、スヌープ要求を転送するのに2点間相互接続が使われ、システム・バスの混雑が軽減されている。この2点間相互接続スキーム260では、各プロセッサに関連付けられたスヌープ・フィルタは、システム中の他のあらゆるプロセッサに関連付けられた各々のスヌープ・フィルタに直接連結される。一つのプロセッサに対するスヌープ要求はすべてそのプロセッサのスヌープ・フィルタに送られ、これが各々のスヌープ要求を処理し、全要求のうち適切な一部だけをプロセッサに渡す。この実施形態においては、スヌープ要求はL2キャッシュ・レベルで（図3に示した前の実施形

10

20

30

40

50

態におけるようなL1ではなく) フィルタされるが、本発明は、任意のキャッシュ・レベルに適用でき、本発明の範囲から逸脱することなく他のレベルのキャッシュ階層に対し用いることができる。

【0041】

ここで図5を参照すると、本発明によるスヌープ・フィルタのハイレベルのブロック図が示されている。1からNまでの他のすべてのプロセッサからのスヌープ要求は、専用の2点間相互接続入力ライン300a、…、300nを介してスヌープ・ブロック310に送られる。スヌープ・ブロック310は、入来するスヌープをフィルタし、適切なサブセットをプロセッサ・スヌープ・インタフェース340を介してプロセッサ320に送る。さらに、スヌープ・ブロック310は、プロセッサとL1キャッシュとのブロック320からL2キャッシュ330へのメモリ・アクセス要求のすべてをモニタする。これらは、L1キャッシュ中の欠落に対する要求だけである。スヌープ・ブロックは、読み取りアドレスをすべてモニタし、これに応じ信号360及び362を制御してそのフィルタを更新する。

10

【0042】

図6は、図5に示されたスヌープ・ブロック310のハイレベル概略図を示す。図6に示されるように、スヌープ・ブロック310は、並行して動作し各々がNヶのメモリ書き込み元(プロセッサ又はDMAエンジンのサブシステムなど)の一つだけに専任する複数(「Nヶ」)のポート・スヌープ・フィルタ400a、…、400nを含む。ポート・スヌープ・フィルタ400a、…、400nの各々は、直接2点間接続された単一の送信元からのスヌープ要求をその専用入力ライン410a、…、410nに受信する。後記で説明するように、一つのポート・スヌープ・フィルタには、さまざまなスヌープ・フィルタ方法を含めることができる。スヌープ・ブロック310は、ストリーム・レジスタ・ブロック430及びスヌープ・トークン制御ブロック426をさらに含む。加えて、各ポート・スヌープ・フィルタ400a、…、400nは、関連するプロセッサからの、プロセッサのL1レベル・キャッシュ中の欠落に対するメモリ読み取りアクセス要求412をすべてモニタする。また、この情報は、後記でさらなる詳細を説明するように、ストリーム・レジスタ・ブロックの使用のために提供される。

20

【0043】

作動において、ポート・スヌープ・フィルタ400a、…、400nは、入来するスヌープ要求を処理し、全スヌープ要求からのサブセットを、各スヌープ・ポートに対して各一つのキューが対応するスヌープ・キュー420a、…、420nのそれぞれに送る。スヌープ・キュー420全部の間を調停し、スヌープ・キュー420からのスヌープ要求すべてを適正に逐次化するキュー・アービトレーション・ブロック422が具備されている。スヌープ・キューのオーバーフロー状態を検知するロジックが具えられ各キューの状態が、遠隔メモリ書き込み元からのスヌープ要求の流れを制御するスヌープ・トークン制御ユニット426に入力される。プロセッサ又はDMAエンジンなどのメモリ書き込み元は、すべてのスヌープ・フィルタに対し利用できるトークンを持ってさえいれば、メモリに対し書き込みを、全スヌープ・フィルタに対してはスヌープ要求を提起することができる。プロセッサが、書き込みを提起するため、相手からの利用可能なトークンを必要としないスヌープ・フィルタは、プロセッサ自体のローカル・スヌープ・フィルタしかない。このメカニズムによって、スヌープ・キューがオーバーフローしないことが確実になる。スヌープ要求は、調停機能422に選定されたスヌープ・キューから、プロセッサ・スヌープ・インタフェース408を介してプロセッサに送られる。

30

40

【0044】

図7は、一つのスヌープ・ポート・フィルタ400のハイレベル概略図を示す。スヌープ・ポート・フィルタ・ブロック400は、さまざまなフィルタリング・アルゴリズムを実行する複数のフィルタ・ユニットを含む。この好適な実施形態では、各々が異なるフィルタ・アルゴリズムを実行する3つのスヌープ・フィルタ・ブロック440、444、及び448が並行して動作している。該スヌープ・フィルタ・ブロックは、スヌープ・キャ

50

ッシュ４４０、ストリーム・レジスタ・チェック・ユニット４４４、及びレンジ・フィルタ４４８と名付けられている。一つの実施形態において、並列のスヌープ・フィルタ・ブロックの各々は、単一の発信元からの同一のスヌープ要求４１０を同時に各自の入力部に受信する。さらに、スヌープ・キャッシュ４４０は、プロセッサからのＬ１レベル・キャッシュ中の欠落に対するメモリ読み取りアクセス要求４１２をすべてモニタし、ストリーム・レジスタ・チェック・ユニット４４４は、図６に示されたストリーム・レジスタ・ユニット４３０から状態入力４３２を受信する。

【００４５】

該好適な実施形態によれば、スヌープ・キャッシュ・ブロック４４０は、スヌープ要求の現時の局所的特性に基づいたアルゴリズムを使ってスヌープ要求４１０をフィルタする、すなわち、特定のロケーションに対し一つのスヌープ要求が行われたならば、同じロケーションへの別の要求は迅速に行える可能性があるということである。スヌープ・キャッシュは、ローカル・キャッシュに対し行われるあらゆるロードをモニタし、必要に応じその状態を更新する。ストリーム・レジスタ・チェック・ブロック４４４は、現在のローカル・キャッシュ内容のサブセットを判断するアルゴリズムを使って、スヌープ要求４１０をフィルタする。キャッシュ内容の概要値がストリーム・レジスタ・ブロック４３０（図６）に含まれており、ストリーム・レジスタ状態４３２は各スヌープ・ポート・フィルタ４００に転送される。この状態に基づいて、各新規スヌープ要求４１０に対して、スヌープ・アドレスをローカル・キャッシュに含めることができる可能性が判断される。スヌープ・ポート・フィルタ中の第三のフィルタリング・ユニットは、レンジ・フィルタ４４８である。このフィルタリング・アプローチのため、２つのレンジ・アドレスが指定される、すなわち最小レンジ・アドレスと最大レンジ・アドレスとである。スヌープ要求のフィルタリングは、まず、当該スヌープ要求がこれらの２つのレンジ・アドレスで規定されたアドレス範囲内にあるかどうか判断することから行われる。この条件を満たす場合、スヌープ要求は廃棄され、そうでなければ、スヌープ要求は判断ロジック・ブロック４５０に送られる。逆に、要求がアドレス範囲内に入っていれば送り、そうでなければ廃棄することもでき、これも本発明の要旨の範囲内である。具体的に、判断ロジック・ブロック４５０は、３つのフィルタ・ユニットのすべて、４４０、４４４、及び４４８の結果４５６を、各個別スヌープ・フィルタ・ユニットを有効又は無効にする制御信号４５４とともに受信する。対応する制御信号が有効化されているスヌープ・フィルタ・ユニットの結果だけが各フィルタリング判断の対象となる。フィルタリング・ユニット４４０、４４４、又は４４８のいずれかが、スヌープ要求４１０を廃棄すべきと判断すればそのスヌープ要求は廃棄される。判断ロジック・ユニットの結果出力は、スヌープ要求に対応するスヌープ・キュー４５２に加えるか、あるいは、スヌープ要求を廃棄し、スヌープ・トークン４５８を廃棄されたスヌープ要求を発行した遠隔プロセッサ又はDMAユニットに返却するかのいずれかとなる。

【００４６】

該好適な実施形態においては、上記のアルゴリズムを実行する３つのフィルタリング・ユニットだけがポート・スヌープ・フィルタに含まれているが、当業者は、本発明の範囲から逸脱することなく、他の任意の数のスヌープ・フィルタ・ユニットを一つのポート・スヌープ・フィルタに含めることができること、又はポート・スヌープ・フィルタ中で何らかの他のスヌープ・フィルタ・アルゴリズムを実行できること、又はスヌープ・アルゴリズムの組み合わせが可能なことをよく理解しているであろう。

【００４７】

図８及び９は、図６のスヌープ・フィルタ３１０の２つの別な実施形態のハイレベル概略図を示す。図６に関連して前に説明したように、スヌープ・ブロックには、多様なフィルタリング・アプローチ、同一のフィルタリング・アプローチ、又はこれら２つの組合せを用いた複数のスヌープ・フィルタを含めることができる。図８に示されるように、Ｎヶのポート・スヌープ・フィルタ４６０a、…、４６０nは、一つずつが並行してＮヶの遠隔メモリ書き込み元の各々に対応して動作している。ポート・スヌープ・フィルタ４６０

a、…、460nの各々は、2点間接続された専任の発信元からのスヌープ要求を、それぞれの入力ライン462a、…、462nに受信する。さらに、各スヌープ・フィルタ460a、…、460nは、ローカル・プロセッサの、L1レベル・キャッシュ中の欠落に対するメモリ・ロード要求464のすべてをモニタする。また、スヌープ・ブロックの他のユニットからの他の信号も、実行されているフィルタ・アルゴリズムが要求する場合にはポート・スヌープ・フィルタに供給する必要があるだろう。必要とされる信号の内容は、一つのポート・スヌープ・フィルタ460内で実行されている一つ以上のスヌープ・フィルタ・アルゴリズムによって決まる。さらに、全ポート・スヌープ・フィルタが、必ずしも同一セットのフィルタリング・アルゴリズムを実行する必要はないことを理解しておくべきである。

10

【0048】

ポート・スヌープ・フィルタ460a、…、460nは入来するスヌープをフィルタし、フィルタ除去されなかったスヌープ要求の適切なサブセットを、それぞれのキュー466a、…、466n中を経由してキュー・アービトレーション・ブロック468に送る。ここでスヌープ要求は逐次化され、次の共用スヌープ・フィルタ470に送られ、該共用フィルタは遠隔メモリ書き込み元すべてからの入力を取り扱う。この共用スヌープ・フィルタ470は、渡されたスヌープ要求のすべてを処理し、全要求のサブセットをスヌープ・キュー472に送る。スヌープ要求は、スヌープ・キュー472からプロセッサ・スヌープ・インタフェース474を介してプロセッサに送られる。なお、図8に示された構成に換えて、共用スヌープ・フィルタ470を複数にしたり省略することも可能である。複数共用フィルタの場合、これらフィルタを並列にも直列にも（この場合、例えば一つのフィルタの出力が次の入力となる）配列することができる。フィルタが複数の発信元からの入力を扱う（すなわち複数発信元の間で共用されている）場合、該フィルタはそれ自体の入力キュー及び調停機能を具え、スヌープ要求を逐次化する必要がある。最終的に順序付けされた全スヌープ要求のサブセットはスヌープ・キュー472に置かれ、スヌープ要求は、プロセッサ・スヌープ・インタフェース474を介してプロセッサに送られる。随意により、スヌープ・キューが満杯になったとき、それを示すスヌープ・キュー満杯表示信号476を具え、スヌープ・キューの中のスヌープの数が所定のレベルを下回るまで、一部又はすべてのメモリ書き込み元がさらなるスヌープ要求を発行するのを止めさせる。

20

【0049】

同様に、図9は、スヌープ・ブロック310中のスヌープ・フィルタの異なった編成による別の実施形態を示す。各々が遠隔メモリ書き込み元（スヌープ・フィルタが付随されるプロセッサは除く）の一つからのスヌープ要求だけを受信するNヶのポート・スヌープ・フィルタ480a、…、480nは並行して動作する。各ポート・スヌープ・フィルタ480a、…、480nは、そのそれぞれの入力ラインに、単一の発信元だけからのスヌープ要求482a、…、480nをそれぞれ受信する。共用スヌープ・フィルタ484は、ポート・スヌープ・フィルタ・デバイス480a、…、480nと並列に連結されている。別の実施形態では、複数の共用スヌープ・フィルタを並列に配置することができる。共用スヌープ・フィルタ484は、Nヶすべての遠隔メモリ書き込み元からの入力を取り扱う。複数の入力を受信するので、共用フィルタ484は、スヌープ要求を逐次化するためにそれ自体の入力キュー486及びキュー・調停機能488を有する。さらに、図9に示された実施形態において、ポート・スヌープ・フィルタ480a、…、480nのすべて及び共用スヌープ・フィルタ484は、ローカル・プロセッサからの、L1レベル・キャッシュ中の欠落に対するメモリ読み取りアクセス要求490をすべてモニタする。スヌープ・フィルタ480a、…、480n及び484は、入来するスヌープ要求をフィルタし、フィルタ除去されなかった適切なサブセットを次の共用スヌープ・フィルタの入力キュー492a、…、492nに送る。次いで、フィルタ除去されなかったスヌープ要求はキュー・調停機能494によって逐次化され、プロセッサ・スヌープ・インタフェース496を介してプロセッサに送られる。スヌープ・キュー・デバイス492a、…、492nの一つ又は486が満杯の場合、スヌープ・キュー満杯表示498が作動され、スヌー

30

40

50

プ・キューの中のスヌープの数が所定のレベルを下回るまで、すべての（又は一部の）メモリ書き込み元がさらなるスヌープ要求を発行するのを止めさせる。

【0050】

次に図10を参照すると、スヌープ・フィルタ・ブロックのさらなる実施形態310が示されている。該ブロックは、ポート・スヌープ・フィルタ400、（図8及び9の）460a、…、460n、及び480a、…、480nに対応する、Nヶのポート・スヌープ・フィルタ500a、…、500nを包含する。各ポート・スヌープ・フィルタ500a、…、500nは、スヌープ・キャッシュ・デバイス502a、…、502n、及びスヌープ・チェック・ロジック504a、…、504nを含む。スヌープ・キャッシュ・デバイス502a、…、502nは、一つの発信元からの最新のスヌープ要求の経過を追跡記録するスヌープ・フィルタリング・アルゴリズムを実行し、このスヌープ要求の発信元を、別のプロセッサ、DMAエンジン、あるいは何か他のユニットとすることができる。一つの発信元からの各新規スヌープ要求に対し、該スヌープ要求のアドレスが、スヌープ・チェック・ロジック・ブロック504中のスヌープ・キャッシュと照合される。この対比結果が一致すれば、すなわちスヌープ・キャッシュ中に該スヌープ要求が見つければ、スヌープされたデータはプロセッサのローカルL1キャッシュ中にないことが保証される。従って、スヌープ要求は、スヌープ・キュー506及びスヌープ・キュー・調停機能508には送られない。スヌープ・キャッシュ502a、…、502nの中に現下のスヌープ要求に対し一致するものが見つからなければ、信号514a、…、514nを使って該スヌープ要求のアドレスがスヌープ・キャッシュに追加される。同時に、スヌープ要求はスヌープ・キュー506に送られる。

【0051】

また、スヌープ・キャッシュ・デバイス502a、…、502nのすべては、ローカル・プロセッサから、読み取りアドレス及び要求を受信し、該メモリ読み取りアクセス・アドレスをスヌープ・キャッシュ502a、…、502n中のエントリと対比する。要求がスヌープ・キャッシュ中のエントリの一つと合致すれば、そのエントリは、キャッシュ・ラインがプロセッサの第一レベルのキャッシュに配置されることになり、スヌープ・キャッシュからは除去される。該好適な実施形態において、並行して動作し、各々が一つの遠隔メモリ書き込み元からのスヌープ要求を追跡記録する複数のスヌープ・キャッシュが用いられる。フィルタリングの後、スヌープ要求のフィルタ除去されなかった部分を、実施形態に応じて、次のポート・スヌープ・フィルタに送るなり、又は一つ以上の共用スヌープ・フィルタのキューに入れるなり、又はプロセッサ・インタフェースのスヌープ・キューに置くなりすることができる。

【0052】

なお、単一のスヌープ・キャッシュ・デバイス502は、各エントリが2つのフィールド、すなわちアドレス・タグ・フィールドと有効ライン・ベクトルとを有するMヶのキャッシュ・ライン（エントリ）の内部編成を含む。スヌープ・キャッシュのアドレス・タグ・フィールドは、通常、ローカル・プロセッサのL1キャッシュのアドレス・タグと同じではなく、これより、有効ライン・ベクトル中に表現されるビット数の分短い。具体的には、有効ライン・ベクトルは、いくつかの連続するキャッシュ・ラインの群を符号化し、すべてのラインが、対応するアドレス・タグ・フィールドによって表現される同一の上位ビットを共用している。しかして、2nの連続するL1キャッシュ・ラインを符号化するためアドレスの最下位ビットn桁が使われる。nがゼロの極端な場合、スヌープ・キャッシュ中の全エントリはただ1本のL1キャッシュ・ラインで表わされる。この場合、有効ライン・ベクトルは「バリッド」ビットに対応するただ一つのビットとなる。

【0053】

スヌープ・キャッシュ中のアドレス・タグ・フィールドのサイズは、L1キャッシュ・ラインのサイズ及び有効ライン・ベクトルの符号化に使われたビットの数により決まる。ある実施形態において、32ビットのアドレス長（31:0）、L1キャッシュ・ラインが32バイトの長さ、及び32ビットの有効ライン・ベクトルに対し、アドレス・ビット

結果がゼロになるかどうかを確認することによって有効ライン・ベクトルのチェックを行うことができる。なお、本発明の範囲から逸脱することなく、他の実施方法を追加して用いることができよう。さらに、有効ライン・ベクトルのヒットのチェックは、アドレス・タグのヒットのチェックと並行して実施することができる。

【0056】

ステップ606において、「タグ」フィールドが一致し、且つ有効ライン・ベクトル中に一致するビットがセットされているかどうかの判断が行われる。「タグ」フィールドが一致し、且つ有効ライン・ベクトル中に対応するビットがセットされていれば、ステップ606に示されるように、スヌープ要求はキャッシュ中ではないことが保証される。しかし、当該スヌープ要求は、キャッシュには送られず、ステップ608に示されるようにフィルタ除去される。

10

【0057】

それとは別に、「タグ」フィールドはスヌープ・キャッシュ中をヒットしたが有効ライン・ベクトル中のビットが一致していないか、あるいは、スヌープ・キャッシュ中でタグがヒットしない場合、このことは、ラインがキャッシュにあり得ることを示している。従って、スヌープ要求は、ステップ612に示されるように、スヌープ・キュー中に置かれて、キャッシュに送られる。また、ステップ610に示されるように、このスヌープ要求は、新しいエントリとしてスヌープ・キャッシュに加えられる。

【0058】

次に図12を参照すると、スヌープ・キャッシュ中に新しい情報を追加するプロセスを記載した、ステップ610（図11）の詳細が示されている。これは、以下に説明するように、いくつかのタスクによって達成される。ステップ614において、まず、アドレス・タグが、既にスヌープ・キャッシュの中に格納されているかどうか（すなわち、アドレス・タグがヒットしたか）を判定する。このステップにおいて、ステップ602（図11）で計算された情報を使うことができる。アドレス・タグがヒットした場合には、プロセスはステップ624に進み、そこでスヌープ要求に一致する選定されたスヌープ・キャッシュ・エントリの有効ライン・ベクトル中のビットがセットされる。ステップ614においてアドレス・タグ・チェックがヒット無しであった場合、その新しいアドレス・タグに対して新規スヌープ・キャッシュ・エントリを割り当てなければならない、プロセスは616に進み、スヌープ・キャッシュの中に利用可能な空きエントリがあるかどうか判断される。空きエントリが利用可能と判断された場合、ステップ620で示されるように最初の利用可能空きエントリが選定される。スヌープ・キャッシュに利用可能な空きエントリが無い場合、ステップ618に示されるように、スヌープ・キャッシュ中の有効エントリの一つが置換え用に選定される。この置換え基準については、ラウンド・ロビン処理、最近最少使用、ランダム、又は、本発明の範囲から逸脱することなく当業者にとって公知の他の置換え基準とすることができる。ステップ622に進み、次いで新規のアドレス・タグが選定されたスヌープ・キャッシュ・ラインに書き込まれ、対応する有効ライン・ベクトルは消去される。次いで、ステップ624に示されるように、スヌープ要求の有効ライン・ベクトル中のビット・セットに一致させて、選定されたスヌープ・キャッシュ・エントリの有効ライン・ベクトル中にビットがセットされる。

20

30

40

【0059】

さらに別の実施形態では、スヌープ要求がスヌープ・キャッシュの中でヒットしたかミスしたかだけに基づいて新しい情報をスヌープ・キャッシュをスヌープ・キャッシュに加えることはせず、代わりに、スヌープ・キャッシュ・ライン全体であれ、有効ライン・ベクトル中に単一ビットをセットするだけのことであれ、新しい値の追加は、判断ロジック・ブロック450（図7）の判断に基づいて行う。この実施形態においては、判断ロジック・ブロックがスヌープ要求をフィルタ除去しない場合にだけ、新しい情報がスヌープ・キャッシュに加えられる。スヌープ・ポート・フィルタ・ブロック400（図7）中の他のどれかのフィルタがスヌープ要求をフィルタ除去した場合、（すなわち、データがローカルL1キャッシュに無いと判断した場合）、スヌープ・キャッシュに新しい情報は加え

50

られないが、作業ステップはスヌープ・キャッシュのヒットの場合と同じである。この実施形態の利点は、冗長な情報の格納が少なくなるのでスヌープ・キャッシュの処理パフォーマンスが良くなることである。

【0060】

次に図13を参照すると、スヌープ・キャッシュからエントリを除去するための制御フローが示されている。プロセッサの、ローカルL1レベル・キャッシュ中の欠落に対するメモリ読み取りの各要求に対して、メモリ要求のアドレスは、全スヌープ要求元に関連する、すべてのスヌープ・キャッシュ中の全エントリと照合される。ステップ630において、メモリ読み取り要求のアドレスは、アドレス・タグ・フィールドの中及び有効ライン・ベクトルの符号化に使われるビットの中に構文分解される。これは、ステップ630で行われる。ステップ632において、一つ以上のタグ・ヒットがあるかどうかの判断が行われる。これは、メモリ要求の「タグ」フィールドを、全スヌープ元に関連するすべてのスヌープ・キャッシュ中の全タグ・フィールドと照合することによって遂行される。該タグ・チェックがミスの場合、このアドレスはフィルタ除去されず、何もしてはならない。しかし、制御フローはステップ630にループ・バックし、プロセッサからの次のキャッシュ欠落読み取りを待つ。

【0061】

ステップ632に戻ると、アドレス・タグの全スヌープ・キャッシュとの対比から一つ以上のヒットが得られたと判断された場合、ヒットされたすべてのスヌープ・キャッシュから情報を除去しなければならない。そこで、ステップ634において、メモリ・読み取りアドレスの適切な下位ビットは、有効ライン・ベクトルに復号され、ステップ635に示されるように、ヒットされたスヌープ・キャッシュ・エントリの有効ライン・ベクトルと対比される。次に、ステップ636に進み、読み取りアドレス・ベクトル中の固有のビット・セットが、同様にスヌープ・キャッシュの有効ライン・ベクトル中にセットされているかどうか判断される。このような有効ライン・ベクトルのヒットがない場合（アドレス・タグ・フィールドのヒットの如何を問わず）、このメモリ・アドレスはフィルタ除去されず、該特定のスヌープ・キャッシュ中の何も変更してはならない。かくして、制御フローはステップ640に進み、すべてのアドレス・タグのヒットの処理が済んだかどうかをチェックし、まだであれば、プロセスはステップ635に戻る。

【0062】

また一方、ステップ636で、読み取りアドレス・ベクトルが有効ライン・ベクトル中でヒットしたと判断されれば、該読み取りアドレスはフィルタ除去されることになる。該メモリ読み取りアドレスは第一レベルのキャッシュにロードされることになるので、対応する有効ライン・ベクトルのビットは消去されねばならない。この有効ライン・ベクトルからの対応ビットの消去はステップ638で行われる。有効ライン・ベクトルからの対応ビットの除去の後で、該有効ライン・ベクトルのビット・セットの数がゼロになる場合、スヌープ・キャッシュからアドレス・タグ・フィールドがさらに除去され、エントリが空になる。次にステップ640に示されるように、有効ライン・ベクトル・ビットのチェック、その消去、及びアドレス・タグの消去の同じプロセスが、必要に応じ、ローカルL1キャッシュの欠落に対する要求であるメモリ読み取り要求をヒットした、スヌープ・キャッシュすべてに対して繰り返し行われる。ヒットしたアドレス・タグ・ラインすべてが処理された状態になっているかどうかステップ640でチェックされる。キャッシュ・ラインのすべてがチェックされたならばプロセスはステップ630に戻る。

【0063】

さらに別の実施形態において、当該ローカル・メモリ要求は、全スヌープ・キャッシュ中の全アドレス・タグに対し同時に対比される。同時に、ローカル・メモリ要求の有効ライン・ベクトルの符号を、ヒットがあった全スヌープ・キャッシュ中の全有効ライン・ベクトルと一せいに対比することができる。次いで、これら2つの結果—アドレス・タグのヒット及び有効ライン・ベクトルのヒット—の組合せにより、対応する有効ライン・ベクトルのビットを除去しなければならないすべてのスヌープ・キャッシュ・ラインを判定し

、全スヌープ・キャッシュ中のヒットしたキャッシュ・ラインから、これらビットのすべてを同時に除去することができる。

【0064】

次に図14を参照すると、ストリーム・レジスタを実装したスヌープ・フィルタ・デバイスのブロック図が示されている。一つの好適な実施形態において、スヌープ・フィルタ・ユニットは以下のエレメントを含む：2つのストリーム・レジスタ及びマスク・セット700、スヌープ・チェック・ロジック・ブロック702、キャッシュ・ラップ検知ロジック・ブロック706、ストリーム・レジスタ選択ロジック・ブロック704、スヌープ要求キュー703、及び、プロセッサ・調停及び多重化ロジック710。後述でさらに詳細を説明するように、キャッシュ中に無いものを追跡するスヌープ・キャッシュ・フィルタと違って、ストリーム・レジスタ及びマスク・セット700は、プロセッサのキャッシュにロードされた最新のデータを追跡記録する。さらに正確には、ストリーム・レジスタは、少なくともキャッシュ中にあるラインを追跡記録するが、一部のラインについて、実際はキャッシュに無いのに、キャッシュ格納されていると想定することがある。しかし、一部の不必要なスヌープ要求をキャッシュに送ったとしても正確さに影響することはない。

10

【0065】

ストリーム・レジスタの中心はストリーム・レジスタ700自体である。キャッシュが新しいラインをロードする毎にこれらレジスタの一つが更新され、新ラインは適切な制御信号716とともにストリーム・レジスタに渡される。図14中のストリーム・レジスタ選択ロジック・ブロック704は、特定のレジスタを選択し、現在のストリーム・レジスタの状態と、信号716中のキャッシュの中にロードされている新しいラインのアドレスとに基づいて、該レジスタを更新する役割を果たす。

20

【0066】

作動において、Nヶの遠隔プロセッサの一つから受信されたスヌープ要求は、図15の右側に示された信号714として着信する。スヌープ・チェック・ロジック・ブロック702は、着信したスヌープ要求714のアドレスをストリーム・レジスタ700の状態と対比して該スヌープ要求がキャッシュにある可能性を判断するポート・フィルタのセットを含む。可能性がある場合、該要求はスヌープ要求キュー703に送られ、そこで、実際のキャッシュ・スヌープとしてキャッシュに送られるのを待つ。図14の待ち行列構造では、Nヶの遠隔プロセッサの各々は専用のスヌープ要求キュー703を有し、最大のスヌープ要求レート設定ができるようになっており、大きな数のスヌープ要求をフィルタ除去し、エンキューの対象にならないようにする。本発明の一般的範囲から逸脱することなく、違った待ち行列構造にすることも可能である。

30

【0067】

調停及び多重化ロジック・ブロック710は、単に、Nヶのスヌープ要求キュー703の間で、キャッシュのスヌープ・インタフェースを適正に割り振りし、すべての要求に対し転送の進捗を保証する。

【0068】

以下に、一つのストリーム・レジスタがどのように更新されるのかを説明する。ストリーム・レジスタは、実際は「ベース」及び「マスク」のレジスタ対とバリッド・ビットとを含む。ベース・レジスタは、ストリーム・レジスタによって表されるキャッシュ・ラインのすべてに共通なアドレス・ビットを追跡記録し、対応するマスク・レジスタは、どのビットがあるかを追跡記録する。バリッド・ビットは単に、ストリーム・レジスタが使用中であることと、遠隔スヌープ要求714をフィルタするかどうかを決める際にスヌープ・チェック・ロジック・ブロック702が参照するのは当該ストリーム・レジスタであることとを示している。以下の例示をわかりやすくするために、32バイトのキャッシュ・ライン・サイズを備えた232バイトのアドレス・スペースを考えてみる。この場合、キャッシュ・ライン・ロード・アドレスは27ビット長で、ストリーム・レジスタのベース及びマスク・レジスタの長さは27ビット長である。

40

50

【0069】

まず、バリッド・ビットはゼロにセットされ、ストリーム・レジスタが使用されていないことが示され、ベース及びマスク・レジスタの内容は意味を持たない。最初のキャッシュ・ライン・ロード・アドレスがこのストリーム・レジスタに加えられたとき、バリッド・ビットは1にセットされ、ベース・レジスタは該ライン・アドレスにセットされ、マスク・レジスタの全ビットが1にセットされ、これでベース・レジスタ中のすべてのビットが有効になる。すなわち、ベース・レジスタに格納されたアドレスにちょうど一致するアドレスが、キャッシュ中にあると見なされ、ビットが一つでも異なればそのアドレスは不在とされる。例えば、第1キャッシュ・ライン・ロード・アドレスが、0x1708fb1（接頭部の0xは16進数であることを示す）として与えられれば、ロードの後のスト

10

ベース=0x1708fb1 マスク=0x7fffff バリッド=1
となる。

【0070】

次に、このストリーム・レジスタに第2のキャッシュ・ライン・ロード・アドレスが加えられると、第2のアドレスがベース・レジスタと対比され、どのビットが異なるかが判断される。ついで、マスク・レジスタが更新され、異なる位置のマスクの中のビットがゼロにされる。しかして、これらのゼロは、ベース・レジスタの対応ビットは「判断外」であり、どの値（ゼロ又は1）をも取れると見なすことができることを示す。従って、これらのビットは、ストリーム・レジスタとの対比にはもはや使えない。例えば、第2のキャ

20

ベース=0x1708fb1 マスク=0x7fffffc バリッド=1
となる。

【0071】

すなわち、第2アドレスとベース・レジスタとは最下位の2ビットが違っており、これにより、マスク・レジスタ中のこれらのビットは消去された。この時点で、ストリーム・レジスタは、アドレス0x1708fb0、0x1708fb1、0x1708fb2、及び0x1708fb3のすべてがキャッシュ中にあり得ることを表す、なぜなら、該レジスタは最下位2つの有効ビットをもはや区別できないからである。但し、実際にロードされた2つのアドレスは、キャッシュ中にあると見なされていることに留意するのが重要である。かくして、このメカニズムは、ストリーム・レジスタに提示されたことがあるすべてのアドレスが、該レジスタ中に含まれることを確実にする。限界になると、マスク・レジスタはすべてゼロになり、あらゆる可能なアドレスが該レジスタ中に含まれ、キャッシュの中にあると見なされる。記載したメカニズムを使って、ストリーム・レジスタにアドレスを加え続けることができるのは明らかである。

30

【0072】

あらゆるキャッシュ・ライン・ロード・アドレスは、きっちりと、複数のストリーム・レジスタの一つに加えられる。従って、ストリーム・レジスタの集合により、完全なキャッシュ状態が表される。どのレジスタを更新するか判断は、図14のストリーム・レジスタ選択ロジック・ブロック704によって行われる。一つの考えられる選択基準として、ライン・ロード・アドレスから最小のハミング距離を有するストリーム・レジスタ（すなわち、マスク・レジスタのゼロに変化するビットの数が最小になるストリーム・レジスタ）を選択することがある。さらに、別の選定基準としてベース・レジスタの最上位のビット群がライン・ロード・アドレスのものと一致するストリーム・レジスタを選択することがある。他にも選定基準があり、本発明の範囲を逸脱することなくそれらを実施することができる。

40

【0073】

更新するストリーム・アドレス・レジスタの選定において、ライン・ロード・アドレスは、対応マスク・レジスタと併せ、ベース・レジスタのすべてと、並行して対比される。

50

次いで、上記のように、該ライン・ロード・アドレスは選定されたストリーム・レジスタに加えられる。

【0074】

スヌープ・チェック・ロジック・ブロック702は、下記のように、スヌープ・アドレス714をストリーム・レジスタのすべてと対比することによって、それがキャッシュの中にある可能性を判断する：スヌープ・アドレス714は、キャッシュ・ライン内のオフセットに対応する低位のビットを除去することによって、ライン・アドレスに変換される。このライン・アドレスは、一つのストリーム・レジスタと対比され、ベース・レジスタとスヌープ・ライン・アドレスとの間のビット単位で排他的OR論理が適用され、引き続き、その結果とマスク・レジスタとのビット単位でのAND論理が適用される。これら2つの論理演算の最終結果に、いずれかゼロでないビットがある場合、該スヌープ・アドレスは、ストリーム・レジスタにおいて「ミス」となり、そのストリーム・レジスタが関与する範囲ではキャッシュの中にあることが分かる。同じ対比が、ストリーム・レジスタのすべてに対して並行して行われ、スヌープ・ライン・アドレスがストリーム・レジスタすべてにおいてミスであれば、該スヌープ・アドレスは、キャッシュ中にあることが分かり除去することができる（すなわち、キャッシュには送られない）。逆に、スヌープ・アドレスがどれかのストリーム・レジスタの中でヒットすれば、該アドレスをキャッシュに転送する必要がある。

10

【0075】

スヌープ・チェック・ロジック702は、Nヶの遠隔スヌープ要求ポートの各々に対して複製されるが、これらポートは同一のストリーム・レジスタ700のセットを共用する。

20

【0076】

時間が経過し、キャッシュ・ライン・ロード・アドレスがストリーム・レジスタに加えられると、これらレジスタは、実際にキャッシュ中にあるものに対する知識について次第に不正確になる。上記の例で説明したように、ゼロ化したあらゆるマスク・ビットの各々が、対応ストリーム・レジスタが、当該キャッシュ中にあると特定するキャッシュ・ラインの数を2倍にする。一般に、無用なスヌープ要求をプロセッサに送る（すなわち、それらをフィルタし損ねる）問題は、ゼロのマスク・ビットの数が増すにつれ悪化する。従って、ストリーム・レジスタ・スヌープ・フィルタには、レジスタを初期状態に戻しリサイクルするメカニズムが具えられる。一般に、このメカニズムは、キャッシュ中にロードされたラインは既にあるラインと置換わるという事象に基づく。ラインを置換えることによって、何時でも該ラインをストリーム・レジスタから除去することができる、というのは、これらレジスタは、どのラインがキャッシュ中にあるかしか追跡していないからである。キャッシュを完全に置換えてしまう場合は常に、ストリーム・レジスタ・スヌープ・フィルタは、個々のラインを除去するのではなく一回にまとめて除去し、レジスタを一掃する。但し、この置換えを行った新規のキャッシュ・ラインも、ストリーム・レジスタに加えられるので、これらレジスタの中身を簡単に廃棄することはできない。

30

【0077】

このジレンマを解決するために、ストリーム・レジスタ・スヌープ・フィルタは、初期キャッシュ状態から始めて、前述のようなストリーム・レジスタの更新を行う。キャッシュ・ラップ検知ロジック・ブロック706には、キャッシュ更新信号717が表すキャッシュ更新情報をモニタし、初期状態に存在したキャッシュ・ラインのすべてを何時新しいラインで上書きするか、すなわちキャッシュを「ラップ」させるかを判断する機能を具える。この時点で、全ストリーム・レジスタの内容（これを「アクティブ」セットという）はストリーム・レジスタの第二「履歴」セットにコピーされ、アクティブ・セット中のストリーム・レジスタは、無効状態に戻され、改めてキャッシュ・ライン・ロード・アドレスの累積を開始する。さらに、ラップ時のキャッシュ状態が、次のキャッシュ・ラップを検知する目的のための新規の初期状態となる。履歴セット中のストリーム・レジスタが更新されることはない。但し、スヌープ・チェック・ロジック・ブロック702が、スヌープ・

40

50

アドレスがキャッシュにある可能性があるかどうかを判断する際には、アクティブ・セットと同じように取り扱われる。このメカニズムを使って、キャッシュが上書きされて、ストリーム・レジスタは定期的にはリサイクルされる。

【0078】

キャッシュ更新基準及びキャッシュ更新信号717に応じて、キャッシュ・ラッピング時を検知できるいくつかのやり方がある。例えば、キャッシュが、上書きされるラインを指定する場合、簡単なスコアボードを用いて、任意の特定ラインが初回に上書きされた時期を知ることができ、カウンタを使ってすべてのラインが上書きされた少なくとも1回の時期を知ることができる。本発明の範囲から逸脱することなく、キャッシュ・ラッピング時を検知するための任意のメカニズムを用いることができる。

10

【0079】

図15は、ストリーム・レジスタ・スヌープ・フィルタの別の実施形態を示し、ここでは、フィルタは、Nヶの遠隔プロセッサによって完全に共有されている。すなわち、個別のスヌープ要求ポート714は、図14に関連して説明した実施形態に示されような、それ自体のスヌープ・チェック・ロジック・ブロック702を持たない。この実施形態において、スヌープ要求は、共有のスヌープ・チェック・ロジック・ブロック701に入力される前に、待ち行列構造708にエンキューされる。待ち行列に入れられた要求は、調停及び多重化ロジック705を経由して適正にスヌープ・チェック・ロジック・ブロック701に送られる。スヌープ・チェック・ロジック・ブロック701の機能は、その他の点では、前に図14に関連して説明した前記のストリーム・レジスタ・スヌープ・チェック・ロジックと同じである。本発明の一般的範囲を逸脱することなく、別の待ち行列構造708が可能なのは明らかである。

20

【0080】

好適な実施形態において、ストリーム・レジスタの2つのセットが用いられるが、本発明の範囲を逸脱することなく2つ以上のセットを使うことが可能である。例えば、4セットのストリーム・レジスタを実装したある実施形態においては、2セットのアクティブ・レジスタ、A及びB、ならびに対応する2セットの履歴レジスタが実装される。該実施形態において、Aセットのレジスタにはキャッシュの一つのサブセットに関する情報を含めることができ、Bセットのレジスタにはキャッシュの別のサブセットに関する情報を含めることができる。ストリーム・レジスタの各セット、A及びB、に割り当てられた部分へのキャッシュ区分については、キャッシュを2つの均等な部分に区分けすることができるが、他の区分法を用いることもできる。さらに、ストリーム・レジスタのセットの数を2よりも多くすることができる。例えば、ストリーム・レジスタの一つのセットを、セットアソシアティブ・キャッシュの各キャッシュ・セットに割り当てることができる。

30

【0081】

さらに別の実施形態において、ストリーム・レジスタの複数の履歴セットを形成し、アクティブ・セットをもっと高い頻度でリサイクルすることができる。但し、レジスタに含まれているキャッシュラインがまだキャッシュ中にある可能性があるときに、当該レジスタを決して消去しないように、キャッシュ・ラップ検知に関して履歴レジスタの管理には注意を払わなければならない。レジスタが決して消去されないことを確実にする一つのやり方は、ストリーム・レジスタのアクティブ・セットに履歴レジスタを加え、キャッシュをラップさせる際には、これら履歴レジスタ（及びアクティブ・レジスタ）のすべてを履歴レジスタの第二セットにコピーすることである。これは、基本的には、前に記載したストリーム・レジスタ・スヌープ・フィルタの好適な実施形態に、履歴の第二の「次元」を加えることである。

40

【0082】

次に図16を参照すると、対になったベース・レジスタとマスク・レジスタとのセットを用いたスヌープ・フィルタに対する制御フローの詳細プロセス流れ図が示されている。作業の開始において、ステップ730に示されるように全ストリーム・レジスタ、全マスク及び全スヌープ・キューはリセットされ、ステップ732に示されるように、システ

50

ムはいずれかのスヌープ元からの次のスヌープ要求を待つ。新しいスヌープ要求が受信されると、ステップ734で示されるように、該スヌープ要求のアドレスはすべてのアドレス・ストリーム・レジスタ及びマスク（ストリーム・レジスタの両セット）と照合される。スヌープ要求のアドレスは、付随するマスクと組み合わせられた全ストリーム・レジスタ（すなわち、全アドレス・ストリーム・レジスタとマスク（ストリーム・レジスタの両セット））と照合される。ステップ736での判断において、現下のスヌープ要求の照合が、対のマスク・レジスタと組み合わせられたあるストリーム・レジスタと一致すると、当該スヌープ・キャッシュ・ラインがキャッシュの中にある可能性があり、スヌープ要求は、ステップ740において、スヌープ・キューに置かれてキャッシュに送られる。プロセスはステップ732に戻り次のスヌープ要求を待つ。しかしながら、スヌープ要求が、ストリーム・レジスタの両方のセット中の、対のマスクと組み合わせたいずれのストリーム・レジスタとも一致しない場合は、当該スヌープ・キャッシュ・ラインがキャッシュ中にないことが保証される。かくて、このスヌープ要求は、ステップ738でフィルタ除去され、プロセスは732に戻る。

10

【0083】

次に図17を参照すると、置換えられたキャッシュ・ラインに対し、2つのストリーム・レジスタ・セット、及びキャッシュ・ラップ検知ロジック・ブロックを更新するための制御フローが示されている。作業の開始において、ステップ750で示されるように、ストリーム・レジスタ及びマスクのすべてはリセットされ、キャッシュ・ラップ検知ロジックはクリアされて、レジスタの第一セットがアクティブにされる。プロセッサの、L1キャッシュの欠落に対するメモリ要求（ロード又は格納オペレーション要求のいずれも含む）の各々に対し、該メモリ要求のアドレスがストリーム・レジスタの第一セットに加えられ、該セットをアクティブ・アドレス・ストリーム・レジスタ・セットと呼ぶ。レジスタの第一セットからのアドレス・ストリーム・レジスタのすべてがチェックされて、実装されているレジスタ選定基準の指定により最善に一致するものが選定される：又はこれに換えて、最初の空ストリーム・レジスタを選定してもよい。メモリ要求のアドレスは、ステップ752に示されるように、アクティブ・レジスタ・セット中の選定されたストリーム・アドレス・レジスタの中に格納され、対となるマスクは更新され、該アドレスのどのビットが関連しどのビットが関連しないのかを反映する。次いで、ステップ754において、キャッシュ・ラップ検知ロジックが更新され、キャッシュにロードされた新しいデータを反映する。キャッシュ・ラップ検知ブロックは、アクティブ・レジスタが最初に使用開始されて以来、キャッシュ中のラインすべてが置換えられたかどうかを追跡記録する。かくして、ステップ756において、キャッシュ・ラップさせる状態になっているかどうか判断がされる。ステップ756においてキャッシュ・ラップ状態が検知されなければ、制御フローはステップ752にループ・バックし、システムは次のプロセッサ・メモリ要求を待つ。一方、キャッシュ・ラップ状態が検知された場合、制御はステップ758へと続き、キャッシュ・ラップ検知ロジック・ブロックはクリアされ、第二ストリーム・レジスタ及びマスク・セットがステップ758においてクリアされる。次のステップ760に進み、システムは次のプロセッサ・メモリ要求を待つ。新しいメモリ要求に対し、レジスタの第二セットの全アドレス・ストリーム・レジスタがチェックされ最善に一致するものが選定され、例えば、実装されたレジスタ選定基準に指定されたレジスタ、又は最初の空ストリーム・レジスタが選定される。メモリ要求のアドレスは、ステップ760に示されるように、第二レジスタ・セット中の選定されたストリーム・アドレス・レジスタのなかに格納され、対となるマスクは更新され、該アドレスのどのビットが関連しているかを反映する。ステップ762に進み、キャッシュ・ラップ検知ロジックは更新され、キャッシュにロードされた新規データを反映する。キャッシュ・ラップ検知ロジックは、レジスタの第二セットが開始されて以来、置換えられたキャッシュ中のラインすべてを追跡記録しており、これにより、ステップ764において、キャッシュ・ラップ状態が存在するかどうかを決める判断が行われる。ステップ764においてラップ・イベントが検知されなければ、システムは、ステップ760に戻り、次のプロセッサ・メモリ要求を待つ。一方、キャッシュ

20

30

40

50

・ラップ・イベントが検知された場合は、レジスタ及びマスクの第一セットが再び使われることになる。かくして、ステップ766において、レジスタの第一セットのレジスタと対のマスクとのすべてがリセットされ、キャッシュ・ラップ検知ロジックはクリアされる。レジスタの第一セットは、再びキャッシュの内容を見積もるためのアクティブとして使われ、制御フローはステップ752にループ・バックされる。

【0084】

ストリーム・レジスタ・スヌープ・フィルタの使用について前に説明したように、ゼロにセットされたマスク・ビットの数が増えるにつれ、各ストリーム・レジスタ・フィルタのスヌープ要求を阻止する力は低減する。例えば、全マスク・ビットがゼロの場合、スヌープ要求はすべて通過してしまう。また一方、仮にこれらのマスク・ビットが一回に1ビットだけゼロにセットされるとすれば（すなわち、各ロードが1ビットだけストリーム・レジスタと異なる）、このような場合、丁度2ビットのストリーム・レジスタと異なるビットを持つアドレスに対するスヌープ要求は、たとえそのアドレスがキャッシュにある可能性がない場合でも通過させることになる。こうして、異なるビットの数など、さらに複雑又は微細な違いの検出を可能にするシグネチャ・フィルタを実装することによって、追加されたフィルタリング能力が具備される。一般的な考え方は、マスク・フィルタとシグネチャ・フィルタとの双方が、該アドレスがキャッシュにある可能性を示す場合に限って、スヌープをストリーム・レジスタから転送するというものである。

【0085】

図18を参照すると、入力を取り入れるシグネチャ機能900、アドレス901、ストリーム・レジスタ902、及びストリーム・レジスタに関連する、アドレスのシグネチャ903の計算がある。下記のような、考えられる多くのシグネチャ機能がある。

1. アドレス中の、ストリーム・レジスタ・アドレスと異なるビットの数。

この数を s で表す。例えば、シグネチャをある定数 M に対する $\min(M, s)$ に設定するなど、短縮係数を用いてスペースを節減することができる。

2. アドレスが N ビットの場合、シグネチャは、あらゆるビットがゼロ値の長さ $B = (N + 1)$ のベクトルである、但し、 $s = i$ の場合ビット i は1とする。スペースを節減するため、これを長さ $B + 1$ ($B + 1 < N$)に切り詰めることができよう、 $\min(s, B) = i$ の場合ビット i は1である。

3. アドレスを k ヶ ($k > 1$) のビット群に分割する。群 i の長さは $L(i)$ であり、 $M(i) = L(i) + 1$ とする。 $s(i)$ を、群 i 中のストリーム・レジスタと異なる群 i 中のアドレス・ビットの数とする。このとき、シグネチャは、($s(1), s(2), \dots, s(k)$)で与えられ、これは単に各群の異なったビットの数である。これらの群については、互いに素なビットのセットにも、部分的にオーバーラップしているビットのセット（すなわち、アドレスの一部のビットが複数の群に在る）にも構成することができる。シグネチャの長さは、 $B(1) + \dots + B(k)$ であり、 $B(i)$ は、 $s(i)$ が取り得るあらゆる値を表すために必要なビットの数である。

4. 上記の(2)と(3)との組み合わせ。この組み合わせにおいて、シグネチャは、各々の群に対応する k ビットのベクトルから成る。 $s(j) = i$ の場合、群 j 中のビット i は1にセットされる。群 i が $L(i)$ ビット長さの場合、 $s(i)$ が取り得るあらゆる値を符号化するためには $M(i) = (L(i) + 1)$ のビットが必要となる。シグネチャは $M(1) + \dots + M(k)$ ビットの長さとなる。例えば、ある定数 M に対して、 $\min(M, s(j)) = i$ とし群 j 中のビット i を1に設定するなどして、短縮係数を使いスペースを節減することができる。

5. 上記の(3)と同様であるが、 $s(1), \dots, s(k)$ の別の一意的な組み合わせとして $M(1) \times \dots \times M(k)$ がある。

6. アドレスを k ヶ ($k > 1$) のビット群に分割し、 $p(i)$ を、群 i 中のアドレス・ビットのパリティとする。このとき、シグネチャは、($p(1), p(2), \dots, p(k)$)で与えられる。

7. 上記の(6)と同様であるが、 $2k$ のパリティの組み合わせの各々を整数 q に

10

20

30

40

50

符号化し、長さ $2k$ のゼロのビット・ベクトルを戻す、但しビット q は 1 とする。

この他に多くのシグネチャが可能なことは言うまでもない。

【0086】

アドレス 901 がキャッシュにロードされると、シグネチャ 903 が、シグネチャ更新機能 904 に供給される。また、更新機能は、入力としてシグネチャ・レジスタ 905 の前の値も取り込み、それを新しい値 906 で置換える。シグネチャ・レジスタを更新する適切な方法は、シグネチャの種類の種類による。S__old はシグネチャ・レジスタの古い値を表し、S__new がシグネチャ・レジスタの新しい値を表し、V はシグネチャ 903 の値を表すものとする。上記のシグネチャ機能に対応して、シグネチャ更新機能 904 は、以下の項目を計算する。

1. S__new = max (S__old, V)。これは、ストリーム・レジスタと異なるビットの最大数を追跡記録する。

2. S__new = S__old ビット単位又は (bit-width - or) V。これは、異なるビットのスコアボードを保持する。

3. S__new = max (S__old, V)。これは、各群中のストリーム・レジスタと異なるビットの最大数を追跡記録する。

4. S__new = S__old ビット単位又は V。これは、各群中の異なるビットのスコアボードを保持する。

5. S__new = S__old ビット単位又は V。これは、各群中の同時に生じた異なるビットのスコアボードを保持する。

6. S__new = S__old ビット単位又は V。これは各群中のパリティのスコアボードを保持する。

7. S__new = S__old ビット単位又は V。これは各群中の同時に生じたパリティのスコアボードを保持する。

【0087】

スヌープ要求が着信すると、そのシグネチャが計算され、シグネチャ・レジスタと対比される。これが一致しなければ、そのアドレスがキャッシュの中に在ることはなく、通常のストリーム・レジスタ及びマスク・フィルタが要求がキャッシュの中にある可能性を示していたとしても、該要求はフィルタ除去される。スヌープは、シグネチャ・レジスタとマスク・レジスタとの双方ともが当該アドレスがキャッシュ中にある可能性を示した場合にだけ転送される。

【0088】

シグネチャ・フィルタリングのメカニズムが図 19 に示されている。キャッシュへのロード・アドレス 1001 は、マスク更新ロジック 1002 に送信され、該ロジックは、先に説明したように動作し、前のマスク・レジスタ 1003、ストリーム・レジスタ 1004 を取り込み、該マスク・レジスタ 1003 を更新する。また、このアドレス 1001 はシグネチャ機能 1005 にも供給され、該機能も入力としてストリーム・レジスタ 1004 を取り込みシグネチャ 1006 を生成する。シグネチャ 1006 及び前のシグネチャ・レジスタ 1008 は、シグネチャ更新ロジック 1007 に供給され、該ロジックはシグネチャ・レジスタ 1008 に対する新規の値を生成する。

【0089】

スヌープ・アドレス 1009 a 要求が着信すると、マスク・フィルタ 1010 がこれを受信して処理し、マスク・スヌープ要求 1011 を生成する。さらに、この同じスヌープ・アドレス (1009 b として示す) 及びストリーム・レジスタ 1004 が、シグネチャ機能 1012 に供給され、シグネチャ 1013 が生成される。なお、シグネチャ機能 1005 と 1012 とは同じロジックでなければならない、すなわち、これらは同一の入力を得た場合同一の出力を生成することになる。スヌープ要求 1013 のシグネチャとシグネチャ・レジスタがシグネチャ・フィルタ 1014 に供給される。

【0090】

このフィルタは、このシグネチャを有する要求がキャッシュ中にある可能性があるかど

10

20

30

40

50

うかを判定しなければならず、その作業の正確さはシグネチャの種類に如何による。シグネチャ更新機能が「スコアボード」型の場合、スヌープのシグネチャは、ビット単位でシグネチャ・レジスタとの論理積を取られる。この結果がゼロでない場合、スヌープ・シグネチャ要求 1 0 1 5 が作成される（要求が作成される場合は、当該信号が 1 にセットされ、されない場合は 0 となる）。シグネチャ更新機能が「変更されたビットの最大数」型の場合、スヌープ・シグネチャがシグネチャ・レジスタ以下の数かどうかを点検するチェックが行われる（各群毎に一対比）。このような対比のすべてが真であれば、当該アドレスがキャッシュ中にある可能性があり、シグネチャ・スヌープ要求 1 0 1 5 が作成される。マスク・スヌープ要求 1 0 1 1 とシグネチャ・スヌープ要求 1 0 1 5 とは、ロジック・エレメント 1 0 1 6 において論理積が取られ、スヌープ要求信号 1 0 1 7 が生成される。この信号が 1 の場合、スヌープ・ベクトル・リスト、又は適用レンジ・フィルタ（図 7 参照）によって排除されない限り、スヌープ要求が生成されることになる。但し、正確に言えば、このようなスヌープ要求を、別のストリーム・レジスタによるシグネチャマスク・フィルタの結果によって排除することはできない。

10

【0091】

シグネチャ・レジスタは、ストリーム・レジスタが最初にセット又はリセットされるのと同時に適切にセットされる。スコアボード型及び最大型のシグネチャ対しては、シグネチャ・レジスタはすべてゼロにセットされる（ストリーム・レジスタとビット差異がないことを示す）。

【0092】

20

ストリーム・レジスタは、特定の開始状態に関連して、キャッシュの全体内容をどの時点で置換えるか一本明細書ではこれをキャッシュ・ラップ状態という一の認知に依存している。セットアソシアティブ・キャッシュは、キャッシュ内のセットのすべてが置換えられたとき、ラップしたと見なされる。通常、一部のセットは、他のものよりも早く置換えられ、セットすべてが置換えられキャッシュがラップする前に、更新状態を継続する。従って、キャッシュ・ラップ検知の開始点は、前回のキャッシュ・ラップ時のキャッシュ・セットの状態である。

【0093】

一つの実施形態において、キャッシュはセットアソシアティブであり、ラウンドロビン置換えアルゴリズムを用いているが、他の置換え法実施も可能である。例えば、キャッシュが、最近最少使用置換及びランダム置換を含む、任意置換方式を実行している場合、キャッシュ・ラップ検知を活用することができる。以下に述べるように、セットアソシアティブ（SA）キャッシュは、いくつかの数のセットを含み、各セットは、複数のライン（各々が同一のセット・インデックスを持つ）を格納することができる。セット内のラインは「ウェイ」と呼ばれる。従って、2 ウェイ・セットアソシアティブキャッシュは、セットあたり 2 本のラインを有する。セット内のウェイは、すべてルックアップ時に同時に探索でき、更新時にはその一つだけが置換えられる。さらに、ウェイのサブセット各々のパーティションに割り当てられるように、セットを区分けすることができる。例えば、4 ウェイ SA キャッシュを 2 つの 2 ウェイ SA キャッシュに区分けすることができる。仮想メモリ・ページ・テーブル（及び変換索引バッファ（TLB））は、特定のメモリ参照がどのパーティションを対象とするのかを（検索及び更新の双方に対し）特定するパーティション識別子を提供することができる。キャッシュ・ラップで更新されるウェイを格納するレジスタは、ウェイ番号を格納するのに十分な大きさが必要である。例えば、4 ウェイ SA キャッシュに対しては 2 ビット、又は 3 2 ウェイ SA キャッシュに対しては 5 ビットである。各セットが異なった時期にラップできるように、セット毎にこのようなレジスタが一つある。

30

40

【0094】

本発明の一つの実施形態において、キャッシュは 3 つのパーティションに区分可能であり、各パーティションは、キャッシュ・ウェイの連続するサブセットを含み、これらサブセットは各キャッシュ・セット内では同様である。メモリ参照は、3 つのパーティション

50

のいずれかの中にキャッシュ格納される、プロセッサのメモリ管理ユニットにより指定される。あるパーティションに対する更新は、他のパーティションとは関係なく行われ、一つのパーティションが全体のキャッシュ・ラップのずっと前にラップすることができる。しかしながら、パーティションが何時更新されるかが分かれば、パーティションのラッピングを検知するのと、キャッシュ全体のラッピングを検知するのとは同じことである。しかして、以降に述べるように、キャッシュ・ラッピングは、パーティション・ラッピングもキャッシュ全体ラッピングも含む。

【0095】

外部のロジックがキャッシュ更新を検知するためには、キャッシュが更新が発生していること、及びどのラインが上書きされるのかの表示を提示しなければならない。該好適な実施形態のロジックは、この情報がセット仕様、ウェイ仕様、及び更新インジケータを使って提示されることを前提としている。

10

【0096】

図20(a)及び(b)は、Nウェイ・セットアソシアティブ・キャッシュの好適な実施形態によるキャッシュ・ラップ検知ロジックを示す。この実施形態では、セットの更新は必ずラウンドロビン順序で実施されることが前提となっている。すなわち、選択された「犠牲」ウェイが、前回上書きされたウェイに続いて上書きされる。

【0097】

図20(a)は、ロジック・ブロック920内に、単一のセット(示された実施形態ではセット「i」)の単一のパーティションのラップを検知するため実装されたロジックの一つの実施形態を示す。このロジックがセットi中にラップを検知した場合、該ロジックはset_wrap(i)信号910をアサートする。図20(b)は、キャッシュの全Nヶセットからの個々のset_wrap(i)信号910を、OR論理関数を使って、どのように組合せて、cash_wrap912を生成するかを示しており、信号912は、キャッシュ全体が(すなわち全セット)がラップ完了するとアサートする。なお、図20(a)及び(b)に示されているロジック及び回路は単なる一つの実施例であり、当業者は、本発明の範囲から逸脱することなくこれに多くの変形及び変更を加えることができることを認識するであろう。

20

【0098】

図20(a)の左側に、ラッピングのためモニタされているパーティションがキャッシュ更新対象に入る時期を判断する、パーティション検知ロジック・ブロック922が示されている。このロジックは、「下部」916で特定されるウェイと「上部」918で特定されるウェイとの間にあるパーティションを担当している。従って、セット・ラップ・パーティションを検知するロジックは、更新が行われ当該パーティションが更新対象となるときにだけその状態を変える。なお、パーティション検知ロジック922は、セット・ラップ検知ロジックのNヶすべてのコピーに共通のものである。

30

【0099】

セット・ラップ検知ロジック内の、共通パーティション更新インジケータは、更新が当該ロジックに関連する特定のセットiに対するものである場合にだけ作動するようさらに条件付けされている。これは、セット指定子924を、セット・ラップ検知ロジック926のインデックスと対比することによって実施される。

40

【0100】

ロジック回路は次のように機能する。始めに、flip-flop driving set_wrap(i)930はクリアされセットがラップしていないことを示しており、レジスタ928は、更新してセット・ラップを完了しなければならないウェイを含むと仮定する。この状態で該レジスタはその値を保持する。キャッシュ更新が発生すると、ウェイ914は、コンパレータ・デバイス919にレジスタ928の内容と対比、判断され、flip-flop driving set_wrap(i)930にロジック1がロードされて、これによりset_wrap(i)910がアサートする。その後、キャッシュの更新により更新されたウェイ914がレジスタ928中に格納され、レジスタ928は

50

効果的にこれらの更新を追跡する。キャッシュ・セットのすべてがラップすると、図20 (b)に示されるように、組み合わされた`cash_wrap912`信号がアサートされ、これにより`flip-flop_driving_set_wap(i)930`はクリアされる(`Rest`が`Load`より優先すると仮定する)。これにより、回路は初期の状態に戻り、レジスタ928は更新が必要なウェイを格納し、次のセット・ラップを表示する。

【0101】

しかして、セット毎に一つのレジスタがあり、該レジスタはあるウェイの番号とそのウェイが上書きされ次いでセットがラップされる時期を格納していることが理解される。しかしながら、各セットは異なる時期にラップ(アクセス・パターン如何による)するので、すべてのセットがラップするまではキャッシュ全体がラップしたとは見なされない。この全体ラップ時点で、犠牲ウェイのポインタ(すなわち最後に上書きされたウェイに対するポインタ)の状態は、次のキャッシュ・ラップを検知するため初期状態となる。この第一実施形態は、それ自体がラップした時とキャッシュ全体がラップした時との間に上書きされたウェイの追跡記録をする上記のレジスタを具えることによってこの必要条件を満たしている。キャッシュ全体がラップすると、該実施形態は上書きされたウェイの追跡を止め、セットが次にラップする時期を判断する比較のためのベース状態となる。

【0102】

キャッシュ・ラップ検出ロジックの第二の実施形態においては、カウンタが実装され、キャッシュ全体がラップすると、すべてのセット・カウンタがパーティションの中のウェイの数にリセットされる。ウェイが上書きされるにつれ、カウンタは、カウント・ダウンされ、カウンタがゼロになると対応セットがラップされる。全カウンタがゼロに達すると、当該キャッシュがラップしたことになり、プロセスが再び開始される。

【0103】

この第二実施形態によれば、図20(a)に示されたボックス920内に具えられたラップ済みセット検出ロジックは、レジスタ及びコンパレータではなく、ロード可能なカウンタに基づいている。このロジックは図21に示されている。このロジックでは、ダウンカウンタ・デバイス932には、パーティション936中のウェイの数がロードされており、`set_wrap(i)910`がアサートされている(`Load`は`Down`に優先すると仮定する)。全セットがラップし、`cash_wrap912`がアサートされると、`flip-flop_driving_set_wap(i)930`はクリアされ、カウンタ932のロードはなくなる。その後、ロジックが、パーティション914及びセット934の各々の更新を追跡し、その都度カウンタ932が1つカウント・ダウンされる。これが0に達すると`flip-flop_driving_set_wap(i)930`には論理1がロードされ、これにより`set_wrap(i)910`がアサートされ、ロジックは初期状態に戻る。

【0104】

図22に示された、キャッシュ・ラップ検出ロジックの第三の実施形態は、最近最少使用法及びランダム法を含め、どのような置換え基準を実行するキャッシュに対しても機能する。このケースでは、スコアボード940を使って、上書きされる正確なキャッシュ・ウェイ914が追跡記録される。具体的には、これを使って、一切のウェイに対する最初の書き込みが検知される。さらに、カウンタ942は、スコアボード・ビットが最初にセットされた(すなわち、0から1へのセット)回数を追跡記録する。該カウンタは、単に、上書きされたビット(旧いビット)がゼロのスコアボード書き込みをカウントすることによってこれを行う。カウンタ942には、パーティション・サイズ936(すなわち、当該パーティションの中のウェイの数)がプリロードされ、このカウンタが0に達すると、全部のキャッシュ・パーティションがラップしたことになる。これは、`cash_wrap912`信号のアサートによって示され、これにより、カウンタ942は再ロードされ(`Load`は`Down`に優先すると仮定する)、スコアボード940はクリア(リセット)される。

10

20

30

40

50

【0105】

本発明の好適な実施形態は、ライトスルー・キャッシュと併せて実施され、書き込み要求に対してだけスヌーピングが行われてスヌープ処置の結果はローカル・データ・コピーを無効にすることであるが、本発明は前記に限定されない。例えば、ライトバック・キャッシュ編成と併用して本発明を実施できる。ライトバック・キャッシュによる場合は、コヒーレンス・プロトコルは、例えば、以下に限定はされないが、周知のMESIプロトコル、又は他のコヒーレンス・プロトコルを含め、追加のトランザクションを包含することになる。ライトバック・キャッシュのコヒーレンス・プロトコルによれば、遠隔プロセッサへの書き込みトランザクションでは、スヌープ処置が実施され、遠隔キャッシュが主メモリに関する最新のコンピュータを有しているかどうか判断される。この場合、以下に限定はされないが、最新のデータを有するプロセッサに該データを主メモリに書き込ませる方法、最新コピー所有のプロセッサから要求元に直接データを転送する方法、又は、他の任意の特定プロトコルのスヌープ介入によるデータ伝送方法を含め、いくつかのやり方の一つを使ってデータ転送を実施する。本発明によれば、スヌープ・フィルタリング処置を用いてより速くスヌープ対応を決定することができる。

10

【0106】

これら好適な実施形態を、固定的な相互接続形態、及び固定的なスヌープ・フィルタリング・オペレーションに関連させて説明してきたが、本発明の一つの態様において、スヌープ・フィルタリング・システムは、一つ又はそれより多くのレベルのスヌープ・フィルタ階層においてプログラムが可能な特質を有する。本発明のプログラム可能な特質面の一つの実施形態によれば、相互接続形態が選択される。プログラム可能な接続形態の一つの実施例によれば、接続形態に関し、各種フィルタの間で1対1関係及び1対多関係を選択できる。プログラム可能な実施形態の別の態様によれば、第一スヌープ・フィルタの次に第二スヌープ・フィルタにアクセスする、又はこれに換えて、第一及び第二スヌープ・フィルタに並行してアクセスする順番をプログラム制御により設定することができる。

20

【0107】

本発明のプログラム可能な特質面のさらに別の実施形態のさらに別の態様によれば、フィルタのサブユニットの動作のプログラムが可能である。これは、例えば、スヌープされるキャッシュの結合性、実装されるコヒーレンス・アーキテクチャなどといったプログラム可能な特性の設定を行うことによって、スヌープ・フィルタの構成が可能な態様の形で実施することができる。プログラム可能なフィルタ・サブユニットの別の態様において、フィルタ・サブユニットはプログラム可能なマイクロコードとして実施され、これにより、プログラム可能なエンジンが、命令のシーケンスを実行して本明細書に記載した一つ以上の好適な実施形態の機能を実行する。一つの態様では、これは汎用マイクロコード・エンジンである。別の態様において、これは、用途最適化されたプログラム可能マイクロコード・エンジンであり、該プログラム可能マイクロコード・エンジンは、スヌープ・フィルタ固有の状態を検知するための専用対応ロジック、ならびに随意により、「キャッシュ・ラップ条件分岐」及び、例えば、「キャッシュ・ラップ条件による割り込み」のようなマイクロコード・エンジンに配信されるマイクロコード・エンジン固有の例外通知の形による専用の通知イベントなどの専用のオペレーションを有する。

30

40

【0108】

本発明のプログラム可能な特質面のさらに別の実施形態において、スヌープ・フィルタリング態様の一部又は全部は、プログラム可能スイッチ・マトリックス、又はプログラム可能ゲート・アレイ・ファブリックを組み込んで実施される。これらの態様の一つにおいて、スヌープ・サブユニット間の経路指定は、プログラム可能スイッチ・マトリックスを設定することによって行われる。このプログラム可能な実施形態の別の態様では、スヌープ・フィルタ・ユニットの動作は、プログラム可能ゲート・アレイ・ロジック・ブロックを設定することによって実施される。本発明の別の態様において、スヌープ・フィルタ・ブロック全体が、少なくとも一つのフィールド・プログラマブル・ゲート・アレイ・セルを設定することによって実施される。

50

【0109】

本発明のプログラム可能な特質面の別の実施形態によれば、一つ以上のスヌープ・フィルタ・サブシステムを無効化することができ、特定のスヌープ・フィルタリング・ステップをバイパスすることができ、又は、スヌープ・フィルタリングを完全に無効化することができる。一つの実施形態において、これは、コンフィギュレーション・レジスタの中にスヌープ・フィルタの構成を書き込むことによって達成される。別の実施形態では、入力信号によってこの構成を選択することができる。

【0110】

本発明の好適な実施形態と考えられるものを提示し説明してきたが、当然ながら、本発明の精神から逸脱することなく、形態又は細部についてさまざまな変形及び変更を容易に行えることが理解されよう。従って、本発明は、説明され図示されたとおりの形態に限定されるものでなく、添付の請求範囲内に入るあらゆる変形が含まれると解釈すべきものとして意図されている。

【図面の簡単な説明】

【0111】

【図1】従来技術による、キャッシュ・コヒーレンシのためのコヒーレンシ・ディレクトリを備えたベース多重プロセッサ・アーキテクチャを示す。

【図2】従来技術による、キャッシュ・コヒーレンシのためスヌーピング・アプローチを用いるベース多重プロセッサ・アーキテクチャを示す。

【図3】本発明に従って描かれた、キャッシュ・コヒーレンシのため2点間接続を使ったスヌーピング・アプローチを用いるベース多重プロセッサ・アーキテクチャを示す。

【図4】別の実施形態による、キャッシュ・コヒーレンシのため2点間接続を使ったスヌーピング・アプローチを用いるベース多重プロセッサ・アーキテクチャを示し、この形態では、スヌープ・フィルタはL2キャッシュと主メモリとの間に配置されている。

【図5】本発明の好適な実施形態によるスヌープ・フィルタ・ブロックのハイレベル概略図を示す。

【図6】本発明による、複数のスヌープ・フィルタを包含するスヌープ・ブロックのハイレベル概略図である。

【図7】本発明による、単一のスヌープ・ポート・フィルタのハイレベル概略図を示す。

【図8】本発明による、スヌープ・ブロックの2つの別の実施形態のハイレベル概略図を示す。

【図9】本発明による、スヌープ・ブロックの2つの別の実施形態のハイレベル概略図を示す。

【図10】本発明のさらなる実施形態による、複数のポート・スヌープ・フィルタを含むスヌープ・ブロックのハイレベル概略図である。

【図11】本発明による、一つのスヌープ元に対しスヌープ・キャッシュを実行するスヌープ・フィルタの制御フローを示す。

【図12】本発明による、ポート・スヌープ・キャッシュに新しいエントリを加えるための制御フロー・ロジックを示す。

【図13】本発明による、スヌープ・キャッシュからエントリを除去するための制御フロー・ロジックを示す。

【図14】本発明による、ストリーム・レジスタを実装したスヌープ・フィルタのブロック図を示す。

【図15】本発明による、ストリーム・レジスタのフィルタリング・アプローチを実装したスヌープ・フィルタの別の実施形態を示す。

【図16】本発明による、対になったストリーム・レジスタとマスクとのセットを用いるスヌープ・フィルタに対する制御フローを示すブロック図である。

【図17】本発明による、置換えられたキャッシュラインに対し、2つのストリーム・レジスタ及びキャッシュ・ラップ検知ロジックを更新するための制御フローを示すブロック図である。

10

20

30

40

50

【図 18】ストリーム・レジスタに追加のフィルタリング能力を付与するためのシグネチャ・フィルタのブロック図を示す。

【図 19】本発明による、シグネチャ・ファイルを用いるフィルタリング・メカニズムのブロック図である。

【図 20】Nウェイ・セットアソシアティブ・キャッシュに対する例示的キャッシュ・ラップ検知ロジック回路（レジスタ及びコンパレータ）を示す。

【図 21】本発明の第二実施形態による、Nウェイ・セットアソシアティブ・キャッシュのための例示的キャッシュ・ラップ検知ロジック回路を示し、これはロード可能なカウンタに基づいている。

【図 22】本発明の第三実施形態による、Nウェイ・セットアソシアティブ・キャッシュのための例示的キャッシュ・ラップ検知ロジック回路を示し、これはスコアボード・レジスタに基づいている。

10

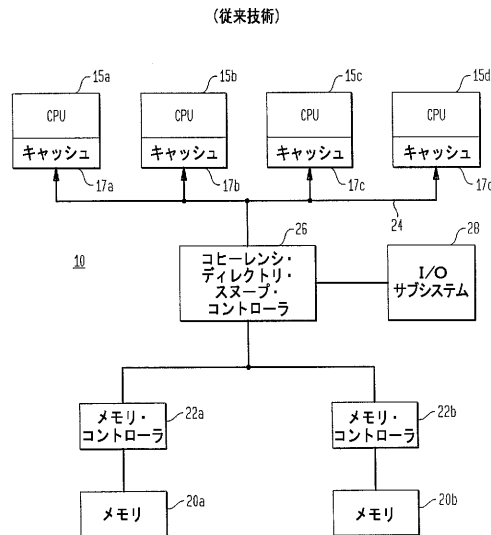
【符号の説明】

【0112】

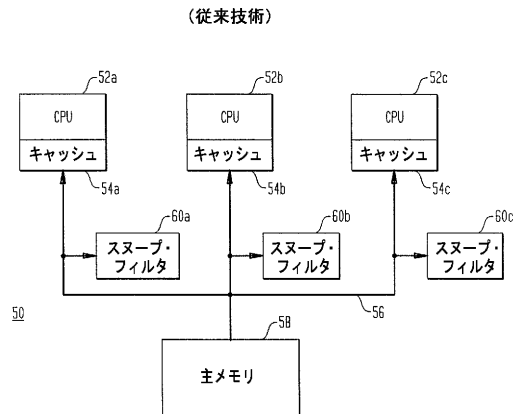
200a L1 キャッシュ 1
 200b L1 キャッシュ 2
 200n L1 キャッシュ N
 250 システム・ローカル・バス
 260 2点間相互接続
 240a～240n スヌープ・フィルタ
 230 主メモリ

20

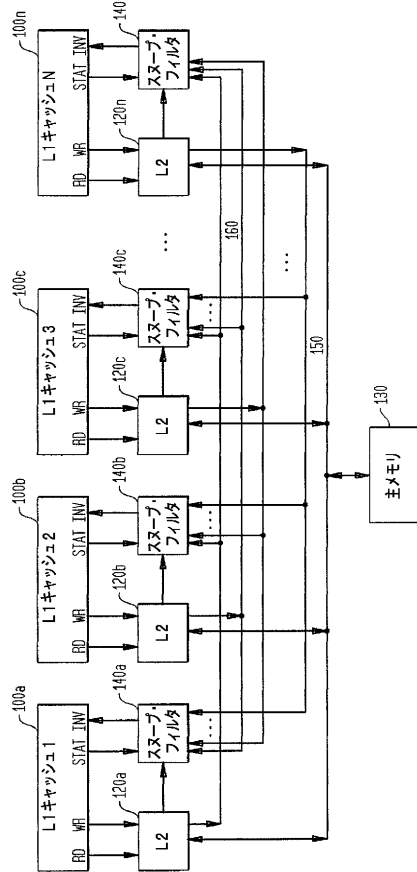
【図 1】



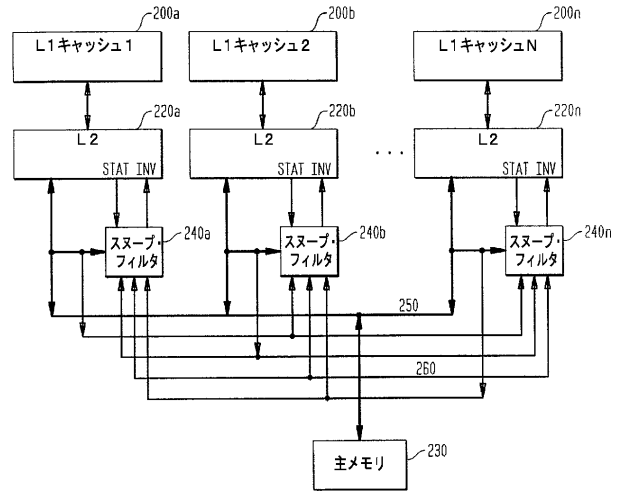
【図 2】



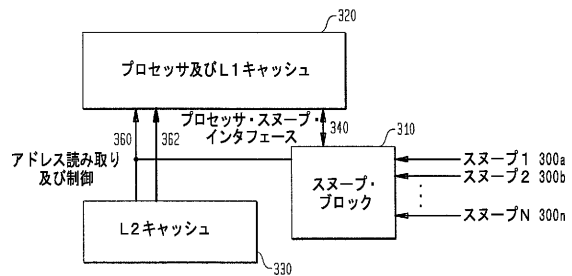
【図 3】



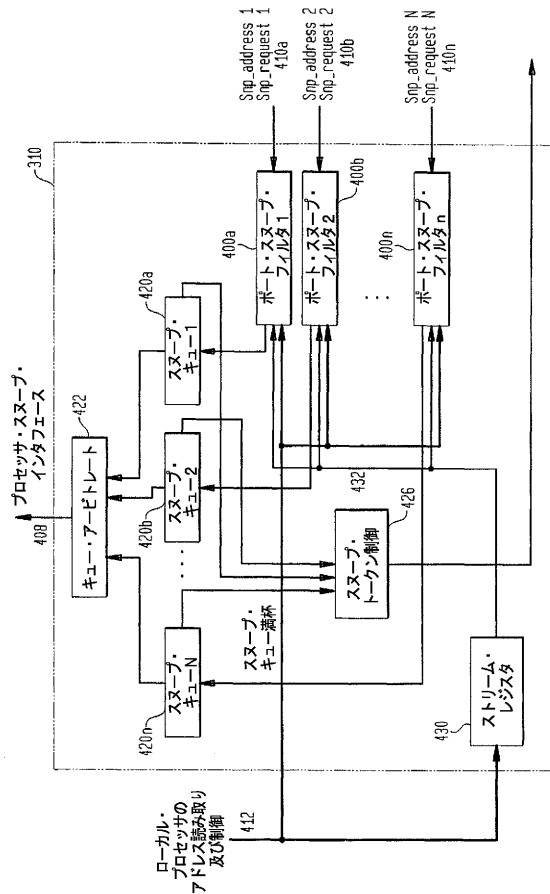
【図 4】



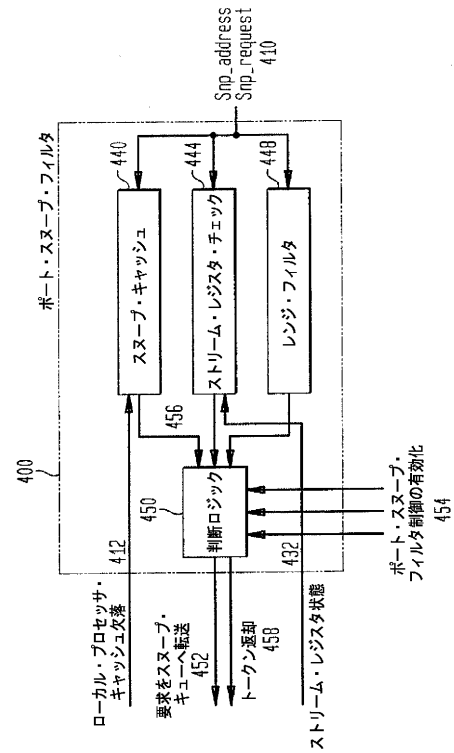
【図 5】



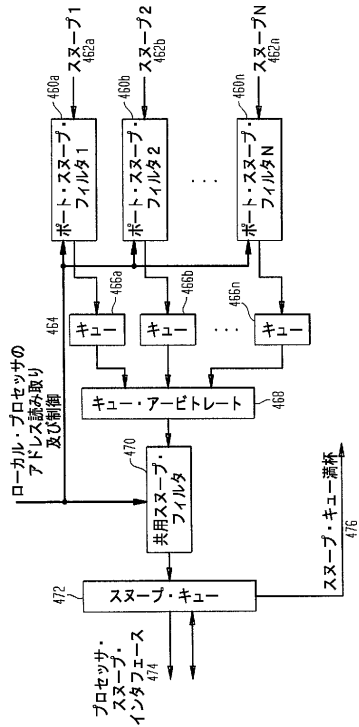
【図 6】



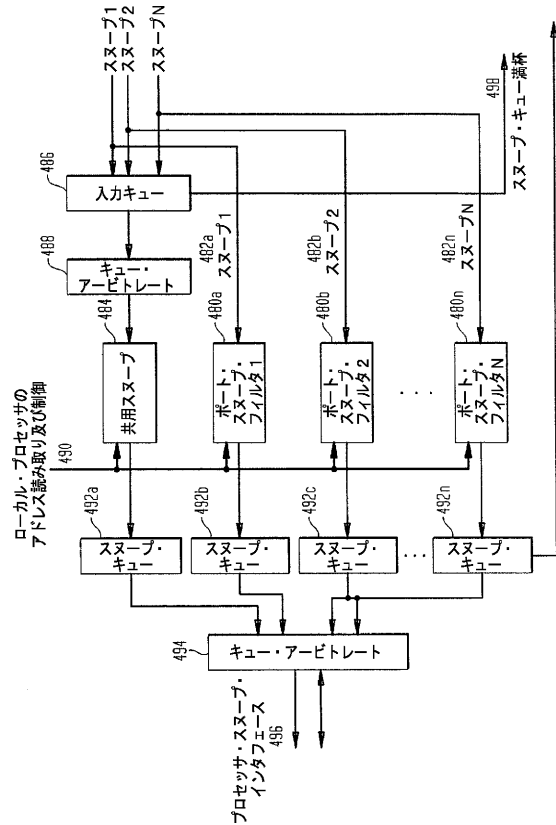
【图 7】



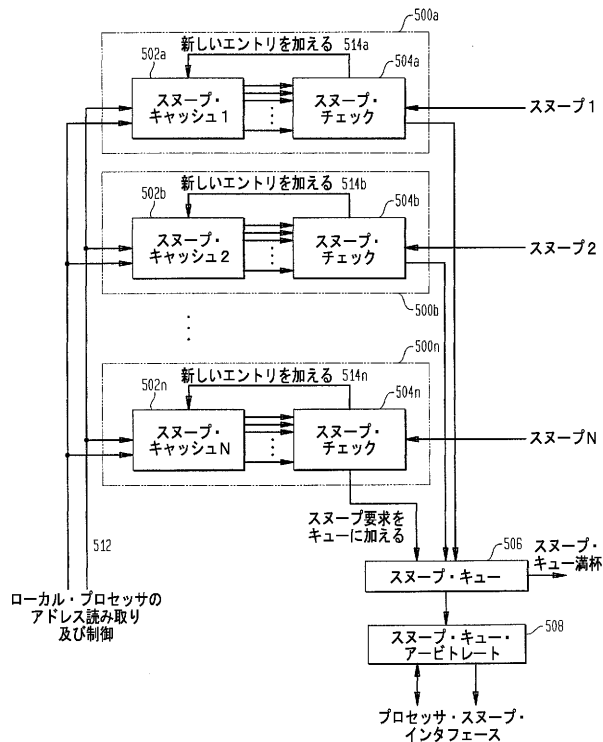
【図 8】



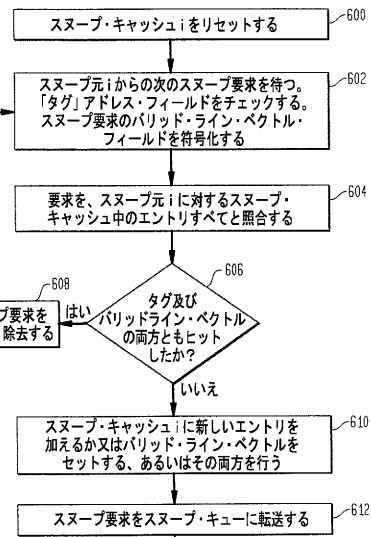
【図 9】



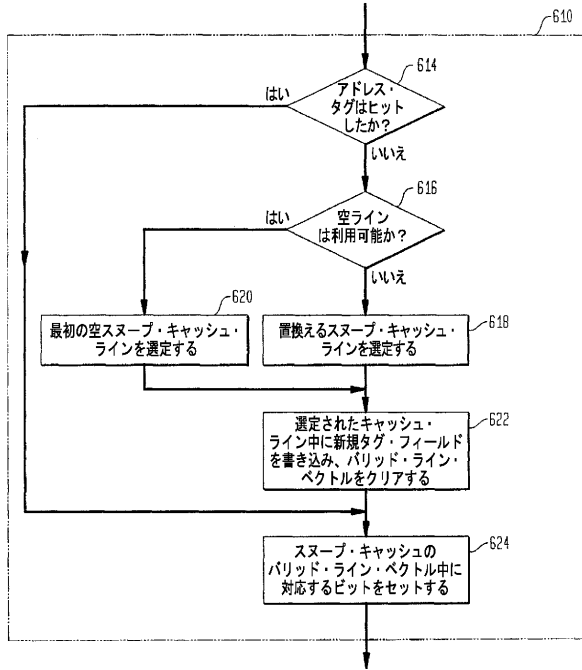
【図 10】



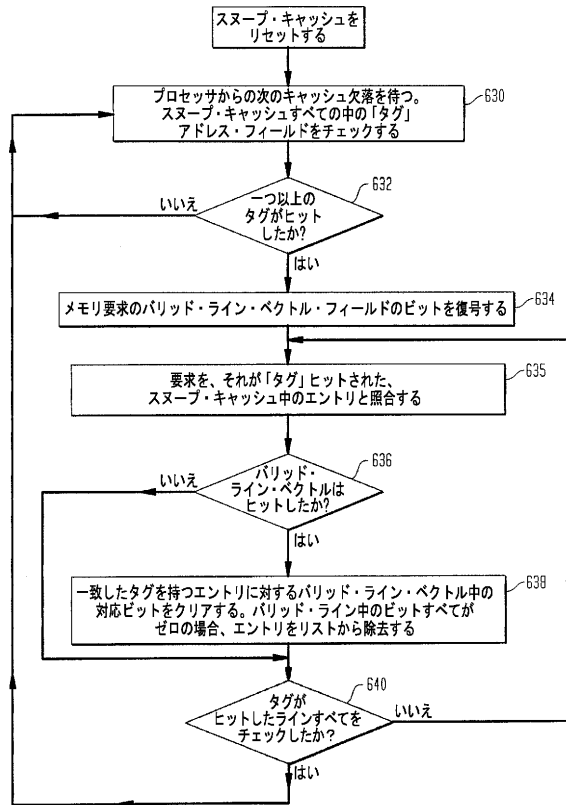
【図 11】



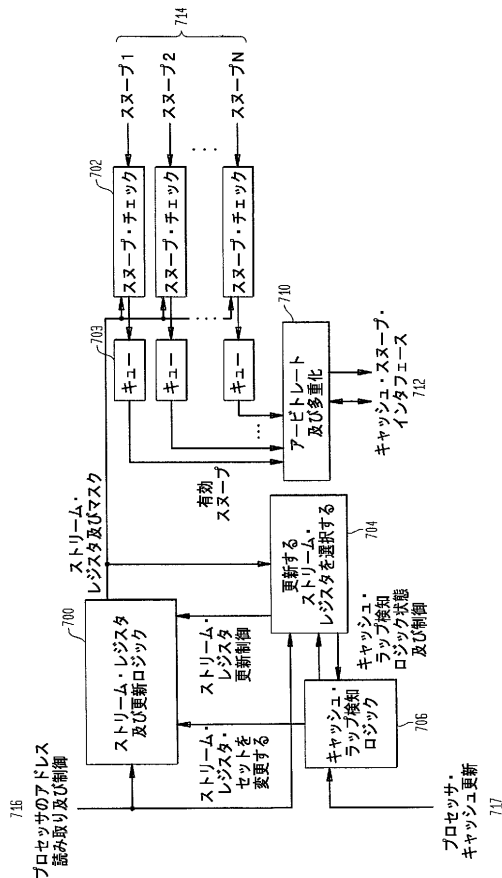
【図 12】



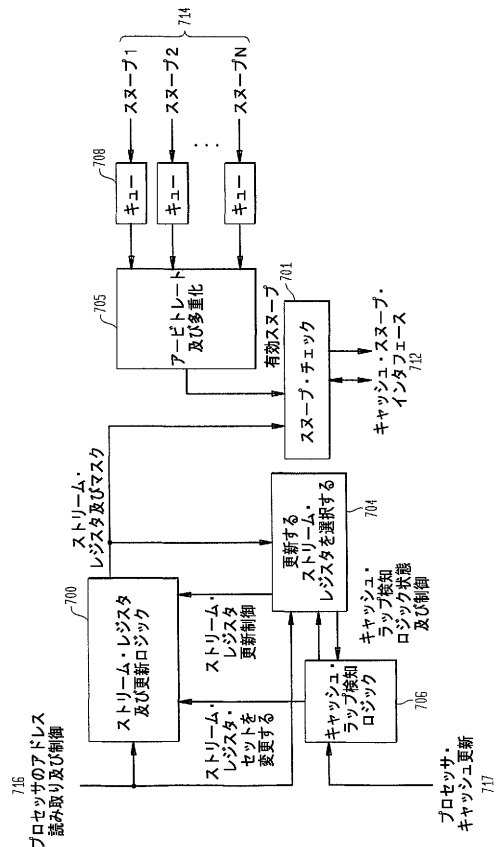
【図 13】



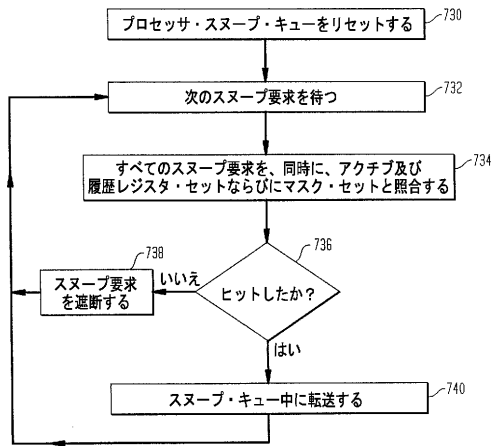
【図 14】



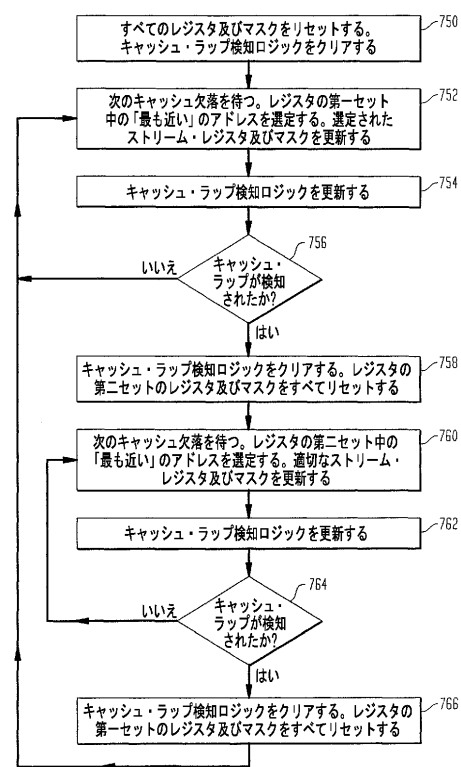
【図 15】



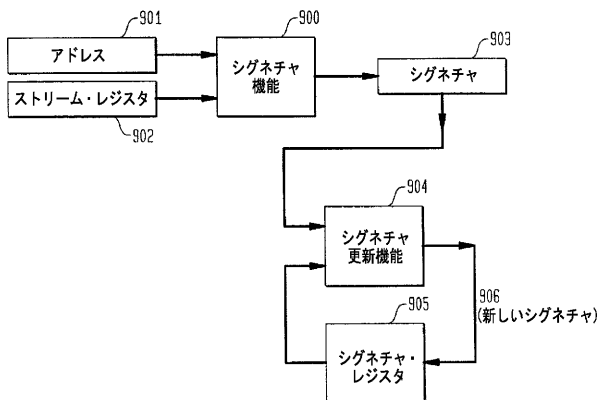
【図 16】



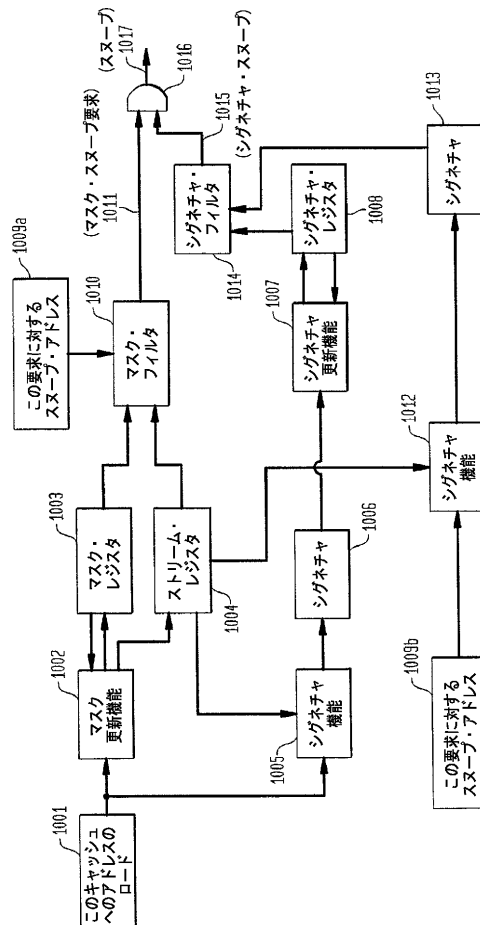
【図 17】



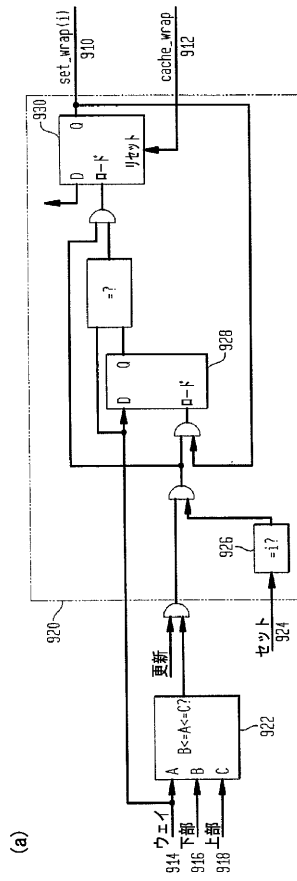
【図 18】



【図 19】



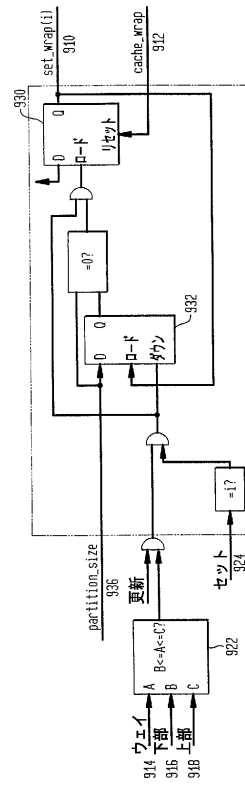
【図 20】



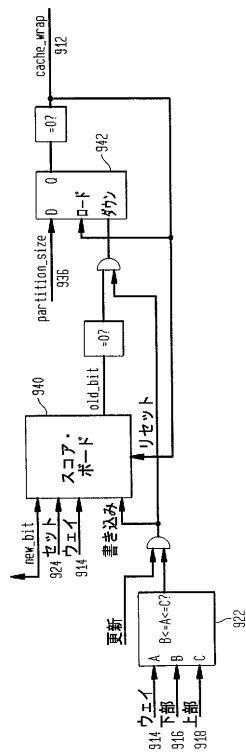
(b)



【図 21】



【図 22】



フロントページの続き

(74)代理人 100086243

弁理士 坂口 博

(72)発明者 ブルームリッチ、マシアス、エー.

アメリカ合衆国06877 コネチカット州 リッジフィールド フロリダ・ヒル・ロード 76

(72)発明者 ガラ、アラン、ジー.

アメリカ合衆国10549 ニューヨーク州 マウント・キスコ マリオン・アヴェニュー 38

(72)発明者 サラプラ、ヴァレンティナ

アメリカ合衆国10514 ニューヨーク州 チャパクア ブルーク・レーン 31

審査官 野田 佳邦

(56)参考文献 F.Aono 外1名, The Azusa 16-way titanium server , IEEE Micro, 米国, IEEE Computer Society , 2002年 8月 6日, Volume: 20, Issue: 5 , p. 54 - 60

A. Moshovos 外3名, JETTY: filtering snoops for reduced energy consumption in SMP servers, The Seventh International Symposium on High-Performance Computer Architecture, 2001. HPCA., 米国, IEEE Computer Society, 2001年, p. 85 - 96

(58)調査した分野(Int.Cl., DB名)

G06F 12/08-12/12