(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5387776号
(P5387776)
(45)発行日　平成26年1月15日(2014.1.15)　(24)登録日　平成25年10月18日(2013.10.18)

(51)Int.Cl.　　F I
G06F 9/48 (2006.01)　　G06F 9/46 311B
G06F 12/08 (2006.01)　　G06F 12/08 531B

請求項の数 5　(全 18 頁)

(21)出願番号　特願2012-526234 (P2012-526234)
(86)(22)出願日　平成22年7月27日 (2010.7.27)
(86)国際出願番号　PCT/JP2010/062626
(87)国際公開番号　WO2012/014285
(87)国際公開日　平成24年2月2日 (2012.2.2)
審査請求日　平成25年1月30日 (2013.1.30)

(73)特許権者　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　100104190
弁理士　酒井　昭徳
(72)発明者　山下　浩一郎
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
(72)発明者　山内　宏真
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
(72)発明者　鈴木　貴久
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
最終頁に続く

(54)【発明の名称】割込制御方法、マルチコアプロセッサシステム、および割込制御プログラム

(57)【特許請求の範囲】
【請求項１】
複数のコアと、前記複数のコアのうち任意のコアのキャッシュメモリにデータが書き込まれると前記任意のコア以外のコアの各キャッシュメモリに対してコヒーレンシを実行するキャッシュコヒーレンシ機構と、デバイスと、を備えるマルチコアプロセッサシステムによる割込制御方法であって、
前記複数のコアのうち前記デバイスからの割込信号を検出した第１のコアが、
前記第１のコアのキャッシュメモリ内の割込フラグを規定する領域に前記割込信号の検出を示す第１のデータを書き込む第１の書込工程と、
前記第１の書込工程によって前記第１のデータが書き込まれると前記キャッシュコヒーレンシ機構によって前記複数のコアのうち前記第１のコア以外の他の２以上のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、前記他の２以上のコアに、前記割込信号に対応する割込処理の実行要求を通知する通知工程と、を実行し、
前記他の２以上のコアのうち、前記割込フラグとして書き込まれた前記第１のデータが維持されており、かつ、前記通知工程によって前記実行要求の通知を受けた第２のコアが、
前記割込処理を実行することを特徴とする割込制御方法。
【請求項２】
前記第２のコアが、
前記第２のコアのキャッシュメモリに書き込まれている前記割込フラグを規定する領域

に前記割込信号の非検出を示す第２のデータを上書きする第２の書込工程を実行し、
前記他の２以上のコアのうち前記第２のコアを除く第３のコアが、
前記第２の書込工程によって前記第２のデータが書き込まれると前記キャッシュコヒーレンシ機構によって前記複数のコアのうち前記第２のコアを除く残余のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、前記残余のコアのキャッシュメモリの前記割込フラグが前記第２のデータに上書きされた場合、前記通知工程によって通知された前記実行要求を破棄する破棄工程と、
をさらに実行することを特徴とする請求項１に記載の割込制御方法。
【請求項３】
前記通知工程は、
前記第１の書込工程によって前記第１のデータが書き込まれると前記キャッシュコヒーレンシ機構によって前記複数のコアのうち少なくとも前記第１のコア以外の他の２以上のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、前記複数のコアに、前記割込信号に対応する割込処理の実行要求を通知し、
前記第２のコアは、
前記複数のコアのうち、前記割込フラグとして書き込まれた前記第１のデータが維持されており、かつ、前記通知工程によって前記実行要求の通知を受けたコアであって、
前記第３のコアは、
前記複数のコアのうち前記第２のコアを除くコアであることを特徴とする請求項２に記載の割込制御方法。
【請求項４】
複数のコアと、前記複数のコアのうち任意のコアのキャッシュメモリにデータが書き込まれると前記任意のコア以外のコアの各キャッシュメモリに対してコヒーレンシを実行するキャッシュコヒーレンシ機構と、デバイスと、を備えるマルチコアプロセッサシステムであって、
前記複数のコアのうち前記デバイスからの割込信号を検出した第１のコアにより、前記第１のコアのキャッシュメモリ内の割込フラグを規定する領域に前記割込信号の検出を示す第１のデータを書き込む第１の書込手段と、
前記第１の書込手段によって前記第１のデータが書き込まれると前記キャッシュコヒーレンシ機構によって前記複数のコアのうち前記第１のコア以外の他の２以上のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、前記第１のコアから他の２以上のコアに、前記割込信号に対応する割込処理の実行要求を通知する通知手段と、
前記他の２以上のコアのうち、前記割込フラグとして書き込まれた前記第１のデータが維持されており、かつ、前記通知手段によって前記実行要求の通知を受けた第２のコアにより、前記割込処理を実行する実行手段と、
を備えることを特徴とするマルチコアプロセッサシステム。
【請求項５】
複数のコアと、前記複数のコアのうち任意のコアのキャッシュメモリにデータが書き込まれると前記任意のコア以外のコアの各キャッシュメモリに対してコヒーレンシを実行するキャッシュコヒーレンシ機構と、デバイスと、を備えるマルチコアプロセッサシステムによる割込制御プログラムであって、
前記複数のコアのうち前記デバイスからの割込信号を検出した第１のコアに、
前記第１のコアのキャッシュメモリ内の割込フラグを規定する領域に前記割込信号の検出を示す第１のデータを書き込む第１の書込工程と、
前記第１の書込工程によって前記第１のデータが書き込まれると前記キャッシュコヒーレンシ機構によって前記複数のコアのうち前記第１のコア以外の他の２以上のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、前記他の２以上のコアに、前記割込信号に対応する割込処理の実行要求を通知する通知工程と、を実行させ、
前記他の２以上のコアのうち、前記割込フラグとして書き込まれた前記第１のデータが維持されており、かつ、前記通知工程によって前記実行要求の通知を受けた第２のコアに

、
前記割込処理を実行させることを特徴とする割込制御プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、割込処理を制御する割込制御方法、マルチコアプロセッサシステム、および割込制御プログラムに関する。
【背景技術】
【０００２】
従来から、コンピュータシステムにおいて、デバイスからの割込信号に対して、ＣＰＵで実行中のプロセスを退避し、割込信号に対する割込処理を行うことで、デバイスに対応する処理を高速で行う技術が存在する。デバイスに対応する処理を、デバイスドライバと称し、デバイスドライバの動作としては、１回の割込信号に対して１回の処理を行う単動作型のドライバと、１回の割込信号に対して連続して処理を行う連続動作型のドライバとが存在する。
【０００３】
単動作型のドライバは、デバイスのレジスタセット操作に対し、ソフトウェアフレンドリなインターフェースに変更する。単動作型のドライバは、たとえば、キーボードのドライバ、マウスのドライバ等が当てはまる。連続動作型のドライバは、ＦＩＦＯ（Ｆｉｒｓｔ Ｉｎ, Ｆｉｒｓｔ Ｏｕｔ）で管理されているバッファの枯渇やバッファフル状態の割込信号に対し、ＤＭＡ（Ｄｉｒｅｃｔ Ｍｅｍｏｒｙ Ａｃｃｅｓｓ）設定を行うことで、データ転送を行う。連続動作型のドライバは、たとえば、ディスプレイのドライバや、カメラのドライバ等が当てはまる。
【０００４】
また、連続動作型のドライバには、データ転送となる対象データの仕様により、デッドライン時間が存在する場合が多い。たとえば、ディスプレイのドライバは、ディスプレイが６０［Ｈｚ］にてリフレッシュする場合、デッドライン時間として約０．０１７［秒］以内にデータを転送するという仕様が存在する。
【０００５】
また、複数のＣＰＵを搭載するマルチコアプロセッサシステムでの割込処理の実装形式について、たとえば、複数のＣＰＵのうちマスタＣＰＵに割込処理を全て配置し、マスタＣＰＵにて割込処理を全て実行するという実装形式が存在する（以下、この形式を従来技術１とする。）。また、他の実装形式について、割込処理を各ＣＰＵに分散して配置し、各ＣＰＵで割込処理を実行する実装形式も存在する（以下、この形式を従来技術２とする。）。
【０００６】
たとえば、従来技術２に関して、割込処理を実行するドライバを各ＣＰＵで持ち、割込発生時にはドライバ同士で割込処理を実行するＣＰＵを調停する技術が開示されている（たとえば、下記特許文献１を参照。）。
【０００７】
また、他の技術として、たとえば、割込処理となる関数のアドレスをデータ構造として持つテーブルを各ＣＰＵが参照し、ＯＳ（Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍ）のセマフォ機構を利用して排他をとって割込処理を実行する。これにより、全てのＣＰＵに割込処理を配置することなく、省メモリ効果を得るという技術が開示されている（たとえば、下記特許文献２を参照。）。
【先行技術文献】
【特許文献】
【０００８】
【特許文献１】特開２００６－３３８１８４号公報
【特許文献２】特開２００８－１４０１９１号公報

【発明の概要】
【発明が解決しようとする課題】
【０００９】
しかしながら、上述した従来技術において、従来技術１にかかる技術では、マスタＣＰＵに割込処理が集中した際に、負荷が集中し、割込処理を処理すべき時間を越えてしまい、リアルタイム処理における応答性能が低下するという問題があった。また、従来技術２にかかる技術では、割込信号が発生した際に、発生した割込信号に対応する割込処理が、どのＣＰＵで実行される割込処理であるかを判断する仕分け処理が発生してしまうという問題があった。また、特許文献２にかかる技術では、割込処理を各ＣＰＵから実行することができるが、負荷の高いＣＰＵに割込処理を割り当ててしまい、割込処理を処理すべき時間を越えてしまい、応答性能が低下する可能性があるという問題があった。
【００１０】
本発明は、上述した従来技術による問題点を解消するため、応答性能を向上させ、割込処理の負荷を分散できる割込制御方法、マルチコアプロセッサシステム、および割込制御プログラムを提供することを目的とする。
【課題を解決するための手段】
【００１１】
上述した課題を解決し、目的を達成するため、開示の割込制御方法は、複数のコアと、複数のコアのうち任意のコアのキャッシュメモリにデータが書き込まれると任意のコア以外のコアの各キャッシュメモリに対してコヒーレンシを実行するキャッシュコヒーレンシ機構と、デバイスと、を備えるマルチコアプロセッサシステムによる割込制御方法であって、複数のコアのうちデバイスからの割込信号を検出した第１のコアが、第１のコアのキャッシュメモリ内の割込フラグを規定する領域に割込信号の検出を示す第１のデータが書き込まれると、キャッシュコヒーレンシ機構によって複数のコアのうち少なくとも第１のコア以外の他のコアのキャッシュメモリに対してコヒーレンシが実行されたことにより、他のコアに、割込信号に対応する割込処理の実行要求を通知し、他のコアのうち、割込フラグとして書き込まれた第１のデータが維持されており、かつ、実行要求の通知を受けた第２のコアが、割込処理を実行し、また、第２のコアのキャッシュメモリに書き込まれている割込フラグを規定する領域に割込信号の非検出を示す第２のデータを上書きする。
【発明の効果】
【００１２】
本割込制御方法、マルチコアプロセッサシステム、および割込制御プログラムによれば、割込処理を最速で実行可能なコアに割込処理を実行させることができ、応答性能を向上させ、また割込処理の負荷を分散できるという効果を奏する。
【図面の簡単な説明】
【００１３】
【図１】本実施の形態にかかるマルチコアプロセッサシステム１００のハードウェアを示すブロック図である。
【図２】マルチコアプロセッサシステム１００のハードウェアの一部とデータ構造とソフトウェアを示すブロック図である。
【図３】マルチコアプロセッサシステム１００の機能を示すブロック図である。
【図４Ａ】割込処理実行の概要を示す説明図（その１）である。
【図４Ｂ】割込処理実行の概要を示す説明図（その２）である。
【図５】マルチコアプロセッサシステム１００の起動時におけるドライバ設定処理を示すフローチャートである。
【図６】割込制御処理を示すフローチャートである。
【発明を実施するための形態】
【００１４】
以下に添付図面を参照して、本発明にかかる割込制御方法、マルチコアプロセッサシステム、および割込制御プログラムの好適な実施の形態を詳細に説明する。

【００１５】
（マルチコアプロセッサシステム１００のハードウェア）
図１は、本実施の形態にかかるマルチコアプロセッサシステム１００のハードウェアを示すブロック図である。図１において、マルチコアプロセッサシステム１００は、ＣＰＵを複数搭載するＣＰＵｓ１０１と、ＲＯＭ（Ｒｅａｄ‐Ｏｎｌｙ Ｍｅｍｏｒｙ）１０２と、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）１０３と、を有する。また、マルチコアプロセッサシステム１００は、フラッシュＲＯＭ１０４と、フラッシュＲＯＭコントローラ１０５と、フラッシュＲＯＭ１０６と、を有する。また、マルチコアプロセッサシステム１００は、ユーザやその他の機器との入出力装置として、ディスプレイ１０７と、Ｉ／Ｆ（Ｉｎｔｅｒｆａｃｅ）１０８と、キーボード１０９と、を有する。また、各部はバス１１０によってそれぞれ接続されている。
【００１６】
ここで、ＣＰＵｓ１０１は、マルチコアプロセッサシステム１００の全体の制御を司る。ＣＰＵｓ１０１は、シングルコアのプロセッサを並列して接続した全てのＣＰＵを指している。ＣＰＵｓ１０１の詳細は、図２にて後述する。また、マルチコアプロセッサシステム１００は、コアが複数搭載されたプロセッサを含むコンピュータのシステムである。コアが複数搭載されていれば、複数のコアが搭載された単一のプロセッサでもよく、シングルコアのプロセッサが並列されているプロセッサ群でもよい。なお、本実施の形態では、説明を単純化するため、シングルコアのプロセッサが並列されているプロセッサ群を例に挙げて説明する。
【００１７】
ＲＯＭ１０２は、ブートプログラムなどのプログラムを記憶している。ＲＡＭ１０３は、ＣＰＵｓ１０１のワークエリアとして使用される。フラッシュＲＯＭ１０４は、ＯＳ（Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍ）などのシステムソフトウェアやアプリケーションソフトウェアなどを記憶している。たとえば、ＯＳを更新する場合、マルチコアプロセッサシステム１００は、Ｉ／Ｆ１０８によって新しいＯＳを受信し、フラッシュＲＯＭ１０４に格納されている古いＯＳを、受信した新しいＯＳに更新する。
【００１８】
フラッシュＲＯＭコントローラ１０５は、ＣＰＵｓ１０１の制御に従ってフラッシュＲＯＭ１０６に対するデータのリード／ライトを制御する。フラッシュＲＯＭ１０６は、フラッシュＲＯＭコントローラ１０５の制御で書き込まれたデータを記憶する。データの具体例としては、マルチコアプロセッサシステム１００を使用するユーザがＩ／Ｆ１０８を通して取得した画像データ、映像データなどである。フラッシュＲＯＭ１０６は、たとえば、メモリカード、ＳＤカードなどを採用することができる。
【００１９】
ディスプレイ１０７は、カーソル、アイコンあるいはツールボックスをはじめ、文書、画像、機能情報などのデータを表示する。このディスプレイ１０７は、たとえば、ＴＦＴ液晶ディスプレイなどを採用することができる。
【００２０】
Ｉ／Ｆ１０８は、通信回線を通じてＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、ＷＡＮ（Ｗｉｄｅ Ａｒｅａ Ｎｅｔｗｏｒｋ）、インターネットなどのネットワーク１１１に接続され、ネットワーク１１１を介して他の装置に接続される。そして、Ｉ／Ｆ１０８は、ネットワーク１１１と内部のインターフェースを司り、外部装置からのデータの入出力を制御する。Ｉ／Ｆ１０８には、たとえばモデムやＬＡＮアダプタなどを採用することができる。
【００２１】
キーボード１０９は、数字、各種指示などの入力のためのキーを有し、データの入力を行う。また、キーボード１０９は、タッチパネル式の入力パッドやテンキーなどであってもよい。
【００２２】

図２は、マルチコアプロセッサシステム１００のハードウェアの一部とデータ構造とソフトウェアを示すブロック図である。図２で示されるハードウェアは、ＲＡＭ１０３と、デバイス２０１＃０～デバイス２０１＃２と、割込コントローラ２０２と、キャッシュコヒーレンシ機構２０３と、ＣＰＵｓ１０１に含まれるＣＰＵ＃０～ＣＰＵ＃３とである。ＲＡＭ１０３と、デバイス２０１＃０～デバイス２０１＃２と、割込コントローラ２０２と、キャッシュコヒーレンシ機構２０３と、ＣＰＵ＃０～ＣＰＵ＃３はバス１１０によって接続されている。

【００２３】

ＣＰＵ＃０～ＣＰＵ＃３は、各々がアクセス可能なキャッシュメモリを有する。また、本実施の形態では、ＣＰＵ＃０がマスタＣＰＵとなりマルチコアプロセッサシステム１００の全体を制御し、ＣＰＵ＃１～ＣＰＵ＃３がスレイブＣＰＵとなりＣＰＵ＃０の配下となる。

【００２４】

デバイス２０１＃０～デバイス２０１＃２は、マルチコアプロセッサシステム１００に接続される周辺機器となるハードウェアであり、たとえばディスプレイ１０７、キーボード１０９などである。割込コントローラ２０２は、デバイス２０１＃０～デバイス２０１＃２からの割込信号を受信し、マスタＣＰＵであるＣＰＵ＃０に通知する。このとき、割込コントローラ２０２は、デバイス２０１＃０～デバイス２０１＃２の優先順位に基づいて、ＣＰＵ＃０に割込信号を通知する。

【００２５】

キャッシュコヒーレンシ機構２０３は、各ＣＰＵのキャッシュメモリについてコヒーレンシを実行することで、キャッシュメモリの整合性をとる。たとえば、ＣＰＵ＃０がキャッシュメモリに値を書き込んだ場合、キャッシュコヒーレンシ機構２０３は、ＣＰＵ＃１～ＣＰＵ＃３のキャッシュメモリに値を反映する。または、ＣＰＵ＃０がキャッシュメモリに値を書き込んだ場合、キャッシュコヒーレンシ機構２０３は、ＣＰＵ＃１～ＣＰＵ＃３が該当の値にアクセスするときに、ＣＰＵ＃１～ＣＰＵ＃３のキャッシュメモリに値を反映してもよい。

【００２６】

キャッシュコヒーレンシ機構２０３の具体例としては、スヌープ方式、ディレクトリ方式、共有キャッシュ等が存在する。本実施の形態では、キャッシュコヒーレンシ機構２０３がスヌープ方式である場合を例として説明を行うが、他の例であってもよい。

【００２７】

図２で示されるデータ構造は、割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３と、ドライバコンテキスト２０５＃０～ドライバコンテキスト２０５＃２とである。割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３は、デバイス２０１＃０～デバイス２０１＃２から割込信号を検出したか否かという割込フラグをデバイス２０１ごとに記憶するテーブルである。たとえば、割込フラグテーブル２０４は、先頭ビットにデバイス２０１＃０から割込信号を受信したか否かを記録し、次のビットにデバイス２０１＃１から割込信号を受信したか否かを記録し、というようにビット単位で記録してもよい。

【００２８】

また、割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３は、各ＣＰＵのキャッシュメモリの保護領域に展開されている。キャッシュメモリの保護領域とは、キャッシュメモリの領域のうち、ＣＰＵに割り当てられるプロセスなどによって使用され、データが入れ替わる領域とは別の領域であり、データが消去されないように保護されている領域である。

【００２９】

たとえば、デバイス２０１＃０から割込信号を受信すると、ＣＰＵ＃０は、割込ハンドラ２０７＃０によって割込フラグテーブル２０４＃０の先頭ビットに“１”を書き込むことで割込信号を受信したことを記録する。以下、割込フラグが“１”のデータを割込フラ

グＯＮ、割込フラグが“０”のデータを割込フラグＯＦＦとする。また、ＣＰＵ＃０が記録を行うと、キャッシュコヒーレンシ機構２０３が、割込フラグテーブル２０４＃１～割込フラグテーブル２０４＃３の先頭ビットもＯＮにする。

【００３０】

ドライバコンテキスト２０５＃０～ドライバコンテキスト２０５＃２は、デバイス２０１＃０～デバイス２０１＃２からの割込処理を実行するために用いられ、ＲＡＭ１０３上に展開されているコンテキストである。たとえば、ドライバコンテキスト２０５＃０には、割込処理２３４で使用されるプログラムカウンタなどのレジスタの値、一時変数の値などが記憶されている。また、ドライバコンテキスト２０５は、デバイス２０１ごとに生成される。具体的には、ドライバコンテキスト２０５＃０は、デバイス２０１＃０のコンテキストであり、ドライバコンテキスト２０５＃１は、デバイス２０１＃１のコンテキストであり、ドライバコンテキスト２０５＃２は、デバイス２０１＃２のコンテキストである。

【００３１】

なお、一般的なコンピュータシステムでは、割込の要因に番号を振り、番号ごとに各番号に対応する割込処理となる関数へのアドレスを格納した割込ベクタテーブルが存在する。したがって、割込フラグテーブル２０４では、割込ベクタテーブルに記載された番号順にデバイス２０１からの割込フラグを対応づけてもよい。具体的には、デバイス２０１＃０からの割込が割込ベクタテーブルに０番として登録されている場合、割込フラグテーブル２０４の先頭ビットがデバイス２０１＃０からの割込フラグであるとして割り当ててもよい。また、ドライバコンテキスト２０５は割込ベクタテーブルに格納された割込処理となる関数へのアドレスによって生成されてもよい。

【００３２】

図２で示されるソフトウェアは、ＯＳ２０６＃０～ＯＳ２０６＃３と、割込ハンドラ２０７＃０～割込ハンドラ２０７＃３と、プロセス２２１～プロセス２３３と、割込処理２３４とである。ＯＳ２０６＃０～ＯＳ２０６＃３は、マルチコアプロセッサシステム１００を制御するプログラムである。具体的には、ＯＳ２０６＃０～ＯＳ２０６＃３は、ＣＰＵ＃０～ＣＰＵ＃３に割り当てられるプロセスが使用するライブラリを提供する。また、ＯＳ２０６＃０～ＯＳ２０６＃３は、ＣＰＵ＃０～ＣＰＵ＃３が実行するプロセスのスケジューリング処理を行う。たとえば、ＯＳ２０６＃０～ＯＳ２０６＃３は、指定されたタイムスライスが満了するたびに、実行するプロセスを切り替える。

【００３３】

割込ハンドラ２０７＃０～割込ハンドラ２０７＃３は、ＯＳ２０６＃０～ＯＳ２０６＃３上で動作するプログラムであり、割込信号を受信した際に実行される。また、プロセス２２１～プロセス２２４は、ＣＰＵ＃０で実行される高優先プロセスである。また、プロセス２２５～プロセス２２７は、ＣＰＵ＃１で実行される低優先プロセスである。また、プロセス２２８は、ＣＰＵ＃２で実行される高優先プロセスであり、プロセス２２９は、ＣＰＵ＃２で実行される低優先プロセスである。また、プロセス２３０～プロセス２３３は、ＣＰＵ＃３で実行される低優先プロセスである。また、高優先プロセスの実行中は、割込禁止設定などにより、割込処理が実行されない状態である。

【００３４】

たとえば、割込ハンドラ２０７＃０は、ＣＰＵ＃０がデバイス２０１＃０の割込信号を受信した際に呼びだされ、割込フラグテーブル２０４＃０の先頭ビットにフラグＯＮを書き込む。フラグＯＮを書き込まれた割込フラグテーブル２０４のコヒーレンシが実行された後、ＣＰＵ＃０はソフトウェア割込信号をＣＰＵ＃０～ＣＰＵ＃３に通知する。ここで、ソフトウェア割込信号の受信タイミングを、ＣＰＵ＃０は高優先プロセス２２２のタイムスライスが満了し、高優先プロセス２２３の実行中とする。同様に、ＣＰＵ＃１は低優先プロセス２２６のタイムスライス満了直前とし、ＣＰＵ＃２は高優先プロセス２２８の実行中、ＣＰＵ＃３は低優先プロセス２３２の実行中とする。

【００３５】

ソフトウェア割込信号を受けたときの各ＣＰＵの動作状況は以下の通りである。ＣＰＵ＃０では、高優先プロセス２２３が実行中であるため、ソフトウェア割込信号に対応する処理がすぐに行われず、割込ハンドラ２０７＃０が呼び出されない。また、ＣＰＵ＃１では、低優先プロセス２２６が実行中であるため、ディスパッチ可能時刻となった際に、ソフトウェア割込信号に対応する処理が行われ、割込ハンドラ２０７＃１が呼び出される。

【００３６】

また、ＣＰＵ＃２では、高優先プロセス２２８が実行されており、ソフトウェア割込信号に対応する処理が行われないため、割込ハンドラ２０７＃２がすぐに呼び出されない。高優先プロセス２２８のタイムスライスが満了すると、ＣＰＵ＃２は、ソフトウェア割込信号に対応する処理を行い、割込ハンドラ２０７＃２を実行する。また、ＣＰＵ＃３では、低優先プロセス２３２が実行中であるため、ディスパッチ可能時刻となった際に、ソフトウェア割込信号に対応する処理が行われ、割込ハンドラ２０７＃３が呼び出される。呼び出された割込ハンドラ２０７＃１～割込ハンドラ２０７＃３は、各々の割込フラグテーブル２０４を読み込み、ＯＮであるかを判断する。

【００３７】

前述の例より、割込ハンドラ２０７＃０～割込ハンドラ２０７＃３のうち、割込フラグテーブル２０４に最速でアクセスしたのが割込ハンドラ２０７＃１であると想定する。先頭ビットがＯＮであると判断した割込ハンドラ２０７＃１は、割込フラグテーブル２０４＃１の先頭ビットをＯＮからＯＦＦに書き換える。続けて、割込ハンドラ２０７＃１は、ドライバコンテキスト２０５＃０を展開し、割込処理２３４を実行する。また、割込フラグテーブル２０４＃１の先頭ビットがＯＮからＯＦＦに書き換えられたことで、キャッシュコヒーレンシ機構２０３が割込フラグテーブル２０４＃１以外の先頭ビットもＯＦＦに書き換える。

【００３８】

これにより、割込ハンドラ２０７＃０、割込ハンドラ２０７＃２、割込ハンドラ２０７＃３が、割込フラグテーブル２０４にアクセスしたときには先頭ビットがＯＦＦになっているため、割込処理２３４を実行しない。結果的に、割込フラグテーブル２０４に最速でアクセスしたＣＰＵ＃１が割込処理を実行することになる。

【００３９】

（マルチコアプロセッサシステム１００の機能）

次に、マルチコアプロセッサシステム１００の機能について説明する。図３は、マルチコアプロセッサシステム１００の機能を示すブロック図である。マルチコアプロセッサシステム１００は、書込部３０１と、通知部３０２と、書込部３０３と、実行部３０４と、書込部３０５と、実行部３０６と、破棄部３０７と、を含む。各機能部のうち、書込部３０１、通知部３０２は、マスタＣＰＵとしての機能であり、書込部３０３～破棄部３０７は、スレイブＣＰＵとしての機能である。

【００４０】

この制御部となる機能（書込部３０１～破棄部３０７）は、記憶装置に記憶されたプログラムをＣＰＵ＃０～ＣＰＵ＃３が実行することにより、その機能を実現する。記憶装置とは、具体的には、たとえば、図１に示したＲＯＭ１０２、ＲＡＭ１０３、フラッシュＲＯＭ１０４、フラッシュＲＯＭ１０６などである。または、Ｉ／Ｆ１０８を経由して他のＣＰＵが実行することにより、その機能を実現してもよい。

【００４１】

また、マルチコアプロセッサシステム１００は、複数のコアと、デバイスと、を含んでいる。複数のコアとは、ＣＰＵ＃０～ＣＰＵ＃３である。デバイスは、デバイス２０１＃０～デバイス２０１＃２である。また、マルチコアプロセッサシステム１００は、複数のコアのうち任意のコアのキャッシュメモリにデータが書き込まれると任意のコア以外のコアの各キャッシュメモリに対してコヒーレンシを実行するキャッシュコヒーレンシ機構２０３を含んでいる。

【００４２】

書込部３０１は、デバイスからの割込信号を検出した第１のコアにより、第１のコアのキャッシュメモリ内の割込フラグを規定する領域に割込信号の検出を示す第１のデータを書き込む機能を有する。割込フラグを規定する領域とは、割込フラグテーブル２０４内の割込フラグを記憶する領域である。第１のデータは、割込信号の検出を示していればどのようなデータでもよく、たとえば、割込フラグテーブル２０４の割込フラグに“１”と書かれたデータである。または、第１のデータは、“ＯＮ”と書かれたデータであってもよい。

【００４３】

たとえば、書込部３０１は、デバイス２０１＃０からの割込信号を検出したＣＰＵ＃０により、割込フラグテーブル２０４＃０の先頭ビットにＯＮを書き込む。なお、書き込まれたデータは、キャッシュコヒーレンシ機構２０３によって、コヒーレンシが実行される。

【００４４】

通知部３０２は、第１のコアから少なくとも第１のコア以外の他のコアに、割込信号に対応する割込処理の実行要求を通知する機能を有する。また、通知部３０２は、書込部３０１によって第１のデータが書き込まれるとキャッシュコヒーレンシ機構２０３によって他のコアのキャッシュメモリに対してコヒーレンシが実行されてから実行要求を通知する。また、通知部３０２は、複数のコアに、実行要求を通知してもよい。実行要求は、コア間のどのような通信を用いて通知してもよいが、たとえば、第１のコアは、ソフトウェア割込によって実行要求を通知してもよい。

【００４５】

具体的には、マルチコアプロセッサシステム１００にて、ソフトウェア割込の特定の割込番号を割込信号に対応する割込処理の実行要求として設定し、ＣＰＵ＃０からＣＰＵ＃１～ＣＰＵ＃３にソフトウェア割込を通知する。なお、通知した実行要求は、ＣＰＵ＃０のレジスタやキャッシュメモリ等に記憶されてもよい。

【００４６】

書込部３０３、書込部３０５は、第２のコアにより、第２のコアのキャッシュメモリに書き込まれている割込フラグを規定する領域に割込信号の非検出を示す第２のデータを上書きする機能を有する。第２のコアとは、他のコアのうち、割込フラグとして書き込まれた第１のデータが維持されており、かつ、通知部３０２によって実行要求の通知を受けたコアである。また、第２のコアは、複数のコアのうち、割込フラグとして書き込まれた第１のデータが維持されており、かつ、通知部３０２によって実行要求の通知を受けたコアであってもよい。

【００４７】

第２のデータは、第１のデータとは異なるデータであり、割込信号の非検出を示していればどのようなデータでもよい。たとえば、第２のデータは、割込フラグテーブル２０４の割込フラグに“０”と書かれたデータである。または、第２のデータは、“ＯＦＦ”と書かれたデータであってもよい。

【００４８】

たとえば、ＣＰＵ＃１～ＣＰＵ＃３のうち、各々の割込フラグテーブル２０４に割込フラグＯＮが維持されており、かつソフトウェア割込を受けたＣＰＵをＣＰＵ＃１であると想定する。書込部３０３は、ＣＰＵ＃１により、割込フラグテーブル２０４＃１の先頭ビットにＯＦＦを上書きする。なお、書き込まれたデータは、キャッシュコヒーレンシ機構２０３によって、コヒーレンシが実行される。

【００４９】

また、たとえば、実行部３０４がＣＰＵ＃１により実行中である場合に、書込部３０１がデバイス２０１＃１から割込信号を検出し、割込フラグテーブル２０４＃０の先頭から２番目のビットにＯＮを書き込んだ場合を想定する。このとき、書込部３０５は、ＣＰＵ＃２により、割込フラグテーブル２０４＃２の先頭から２番目のビットにＯＦＦを上書きする。

【００５０】
実行部３０４、実行部３０６は、書込部３０３、書込部３０５を実行したコアにより、割込処理を実行する機能を有する。たとえば、書込部３０３を実行したＣＰＵ＃１は、ドライバコンテキスト２０５＃０を展開し、割込処理を実行する。また、たとえば、書込部３０５を実行したＣＰＵ＃２は、ドライバコンテキスト２０５＃１を展開し、割込処理を実行する。
【００５１】
破棄部３０７は、複数のコアのうち第１のコア以外となる他のコアのうち、第２のコアを除く第３のコアにより、通知部３０２によって通知された実行要求を破棄する機能を有する。また、破棄部３０７は、書込部３０３、書込部３０５によって第２のデータが書き込まれると、キャッシュコヒーレンシ機構２０３によってコヒーレンシが実行され、残余のコアのキャッシュメモリの割込フラグが第２のデータに上書きされた場合に、実行する。また、キャッシュコヒーレンシ機構２０３のコヒーレンシ対象は、複数のコアのうち第２のコアを除く残余のコアのキャッシュメモリに対してである。また、第３のコアは、複数のコアのうち第２のコアを除いたコアであってもよい。
【００５２】
なお、破棄部３０７は、図３ではＣＰＵ＃３の機能となっているが、ＣＰＵ＃０～ＣＰＵ＃２でも、実行部３０４、実行部３０６による割込処理の実行を行わなかったＣＰＵにて実行される機能である。たとえば、破棄部３０７は、ＣＰＵ＃３により、通知部３０２により通知されたソフトウェア割込を破棄する。なお、破棄されたという情報は、ＣＰＵ＃３のレジスタ、キャッシュメモリなどに記憶されてもよい。
【００５３】
図４Ａ、図４Ｂは、割込処理実行の概要を示す説明図である。また、図４Ａ、図４Ｂの説明内にて割込フラグテーブル２０４の符号に付随する“＿ｔ０”、“＿ｔ１”、“＿ｔ４”、“＿ｔ７”は、時刻ｔ０、時刻ｔ１、時刻ｔ４、時刻ｔ７での割込フラグテーブル２０４を示している。
【００５４】
時刻ｔ０にて、ＣＰＵ＃０は、割込コントローラ２０２によってデバイス２０１＃０～デバイス２０１＃２のいずれか１つから割込信号を受け取ると、イベント発生とみなし、割込ハンドラ２０７＃０の処理を開始する。図４Ａの例では、デバイス２０１＃０から割込信号を受け取った場合を想定する。同時刻にて、ＣＰＵ＃１は、低優先プロセス２２５を実行しており、ＣＰＵ＃２は、高優先プロセス２２８を実行しており、ＣＰＵ＃３は、低優先プロセス２３０を実行している。また、割込フラグテーブル２０４＃０＿ｔ０～割込フラグテーブル２０４＃３＿ｔ０について、全てのフラグがＯＦＦとなっている。
【００５５】
時刻ｔ１にて、ＣＰＵ＃０は、割込ハンドラ２０７＃０によって、割込フラグテーブル２０４＃０のイベント発生したデバイス２０１＃０に対応する割込フラグをＯＮにする。具体的に、ＣＰＵ＃０は、割込フラグテーブル２０４＃０の先頭ビットをＯＮにする。これにより、割込フラグテーブル２０４＃０＿ｔ１は、先頭ビットがＯＮになった状態となる。
【００５６】
また、同時刻にて、ＣＰＵ＃１は、低優先プロセス２２６を実行しており、ＣＰＵ＃２は、高優先プロセス２２８を実行し続けており、ＣＰＵ＃３は、低優先プロセス２３１を実行している。また、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃０の更新を検出し、割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３のコヒーレンシを実行する。具体的には、割込フラグテーブル２０４＃０＿ｔ１と同じ値を持つように、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃１＿ｔ１～割込フラグテーブル２０４＃３＿ｔ１の先頭ビットをＯＮにする。
【００５７】
なお、キャッシュコヒーレンシ機構２０３の採用された機構によっては、特定のＣＰＵ

によるキャッシュメモリの更新後、他のＣＰＵがアクセスした際に、値が更新したかを全てのキャッシュメモリに問い合わせてコヒーレンシを実行する機構も存在する。具体的には、スヌープ方式の無効型プロトコルであるＭＥＳＩプロトコルでは、時刻ｔ１の段階では、割込フラグテーブル２０４＃０のみが割込フラグＯＮとなり、他の割込フラグテーブル２０４は、値を持たない状態である。後述する時刻ｔ４で、ＣＰＵ＃１からのアクセスにより、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃０から値を取得し、割込フラグテーブル２０４＃１を割込フラグＯＮとする。

【００５８】

または、前述のように、特定のＣＰＵによる更新を検出してすぐにコヒーレンシを実行する機構も存在する。本実施の形態にかかるキャッシュコヒーレンシ機構２０３は、どのような形態であってもよいが、説明のし易さから、更新を検出してすぐにコヒーレンシを実行する機構にて説明を行う。

【００５９】

時刻ｔ２にて、ＣＰＵ＃０は、ＣＰＵ間のソフトウェア割込をＣＰＵ＃１～ＣＰＵ＃３に通知する。なお、ＣＰＵ＃０は、ソフトウェア割込をＣＰＵ＃０自身に送信してもよい。また、高優先プロセスが多く割り当てられるように設計されているＣＰＵには、ソフトウェア割込を送信しなくてもよい。これにより、割込処理による高優先プロセスの実行遅延を防ぐことができる。図４Ａの例では、ＣＰＵ＃０に高優先プロセスが多く割り当てられているため、ＣＰＵ＃０は、ＣＰＵ＃０自身にソフトウェア割込を通知していない。

【００６０】

同時刻にて、ＣＰＵ＃１は、ＣＰＵ間のソフトウェア割込を捕捉し、ＣＰＵ＃２は、高優先プロセス２２８を実行し続けている。ＣＰＵ＃３は、ソフトウェア割込を捕捉する前に、低優先プロセス２３２を実行している状態を想定している。

【００６１】

時刻ｔ３にて、ＣＰＵ＃０は、高優先プロセス２２３を実行する。ＣＰＵ＃１は、ソフトウェア割込を受け、割込ハンドラ２０７＃１の処理を開始する。ＣＰＵ＃２は、高優先プロセス２２８を実行し続けている。ＣＰＵ＃３は、ＣＰＵ間のソフトウェア割込を捕捉する。

【００６２】

時刻ｔ４にて、ＣＰＵ＃０は、高優先プロセス２２４を実行する。ＣＰＵ＃１は、割込フラグテーブル２０４＃１がＯＮとなっている割込フラグが存在するか否かをチェックする。図４Ａの例では、時刻ｔ１にて割込フラグテーブル２０４＃１の先頭ビットがＯＮになっているため、ＣＰＵ＃１は、先頭ビットをＯＦＦにする。これにより、割込フラグテーブル２０４＃１＿ｔ４は、先頭ビットがＯＦＦとなった状態になる。また、同時刻にて、ＣＰＵ＃２は、ＣＰＵ間のソフトウェア割込を捕捉し、ＣＰＵ＃３は、割込ハンドラ２０７＃３の処理を開始する。

【００６３】

また、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃１の更新を検出し、割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３のコヒーレンシを実行する。具体的には、割込フラグテーブル２０４＃１＿ｔ４と同じ値を持つように、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃０＿ｔ４の先頭ビットをＯＦＦにする。また、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃２＿ｔ４、割込フラグテーブル２０４＃３＿ｔ４の先頭ビットもＯＦＦにする。

【００６４】

時刻ｔ５にて、ＣＰＵ＃０は、高優先プロセス２２１を実行する。ＣＰＵ＃１は、時刻ｔ１にてＯＮであった割込フラグに対応する割込処理を実行する。具体的には、ＣＰＵ＃１は、先頭ビットに対応するドライバコンテキスト２０５＃０を展開し、割込処理２３４を実行する。割込処理終了後、ＣＰＵ＃１は、次の時刻にて通常のプロセスを実行する。ＣＰＵ＃２は、割込ハンドラ２０７＃２の処理を開始する。ＣＰＵ＃３は、割込フラグテーブル２０４＃３がＯＮとなっている割込フラグが存在するか否かをチェックする。図４

Ａの例では、割込フラグテーブル２０４＃３＿ｔ４にて全てのビットがＯＦＦとなっているため、ＣＰＵ＃３は、割込処理を行わず、次の時刻にて通常のプロセスを実行する。

【００６５】

時刻ｔ６にて、ＣＰＵ＃０は、割込コントローラ２０２によってデバイス２０１＃０～デバイス２０１＃２のいずれか１つから割込信号を受け取ると、イベント発生とみなし、割込ハンドラ２０７＃０の処理を開始する。図４Ｂの例では、デバイス２０１＃１から割込信号を受け取った場合を想定する。同時刻にて、ＣＰＵ＃１は、割込処理２３４を実行し続けている。ＣＰＵ＃３は、割込フラグテーブル２０４＃２がＯＮとなっている割込フラグが存在するか否かをチェックする。図４Ａの例では、割込フラグテーブル２０４＃２＿ｔ４にて全てのビットがＯＦＦとなっているため、ＣＰＵ＃２は、割込処理を行わず、次の時刻にて通常のプロセスを実行する。ＣＰＵ＃３は、低優先プロセス２３３を実行する。

【００６６】

時刻ｔ７にて、割込ハンドラ２０７＃０によって、ＣＰＵ＃０は、割込フラグテーブル２０４＃０のイベント発生したデバイス２０１に対応する割込フラグをＯＮにする。具体的には、ＣＰＵ＃０は、デバイス２０１＃１に対応する割込フラグテーブル２０４＃０の先頭から２番目のビットをＯＮにする。これにより、割込フラグテーブル２０４＃０＿ｔ７は、先頭から２番目のビットがＯＮとなった状態となる。ＣＰＵ＃１は、割込処理２３４を実行し続けている。ＣＰＵ＃２は、低優先プロセス２２９を実行する。ＣＰＵ＃３は、低優先プロセス２３０を実行する。

【００６７】

また、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃０の更新を検出し、割込フラグテーブル２０４＃０～割込フラグテーブル２０４＃３のコヒーレンシを実行する。具体的には、割込フラグテーブル２０４＃０＿ｔ７と同じ値を持つように、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃１＿ｔ７の先頭から２番目のビットをＯＮにする。同様に、キャッシュコヒーレンシ機構２０３は、割込フラグテーブル２０４＃２＿ｔ７、割込フラグテーブル２０４＃３＿ｔ７の先頭から２番目のビットをＯＮにする。

【００６８】

時刻ｔ８にて、ＣＰＵ＃０は、ＣＰＵ間のソフトウェア割込をＣＰＵ＃１～ＣＰＵ＃３に通知する。同時刻にて、ＣＰＵ＃１は、割込処理２３４を実行し続けている。ＣＰＵ＃２は、ＣＰＵ間のソフトウェア割込を捕捉する。ＣＰＵ＃３は、ソフトウェア割込を捕捉する前に、低優先プロセス２３１を実行している状態を想定している。

【００６９】

時刻ｔ９にて、ＣＰＵ＃０は、高優先プロセス２２２を実行する。ＣＰＵ＃１は、ＣＰＵ間のソフトウェア割込を捕捉する。ＣＰＵ＃２は、ソフトウェア割込を受け、割込ハンドラ２０７＃２の処理を開始する。ＣＰＵ＃３は、ＣＰＵ間のソフトウェア割込を捕捉する。

【００７０】

以降の時刻については特に図示していないが、最も速くＣＰＵ間のソフトウェア割込を捕捉したＣＰＵ＃２が、デバイス２０１＃１に対応する割込処理を実行することになる。このように、ソフトウェア割込を行い、割込フラグテーブル２０４に最速でアクセスしたＣＰＵ＃１が割込処理を行うことになる。これにより、割込処理に最も速くとりかかれるＣＰＵが割込処理を行うことで、リアルタイム性能を向上させることができる。

【００７１】

また、時刻ｔ０にて発生しているイベント以降、図４Ｂの例では時刻ｔ６にて、時刻ｔ０にて割込が発生したデバイス２０１＃０とは別のデバイスであるデバイス２０１＃１から割込が発生した場合を想定する。この場合、デバイス２０１＃０の割込処理および高優先プロセス以外を実行中のＣＰＵ＃２が、最も速くＣＰＵ間のソフトウェア割込に対応して割込フラグテーブル２０４にアクセスし、デバイス２０１＃１の割込処理を実行するこ

とになる。このように、デバイス処理を行うＣＰＵを分散させることができ、負荷の一極集中を避けることができる。

【００７２】

図４Ａ、図４Ｂにて示した割込処理を実現するために、マルチコアプロセッサシステム１００は、後述する図５、図６のフローチャートを実行する。図５では、マルチコアプロセッサシステム１００の起動時におけるドライバ設定処理について記載している。図６では、マルチコアプロセッサシステム１００が起動完了し、通常運用としてタイムスライス等に基づいて各プロセスを実行中に、イベント発生としてデバイス２０１＃０～デバイス２０１＃２から割込信号が通知された場合の処理について記載している。

【００７３】

図５は、マルチコアプロセッサシステム１００の起動時におけるドライバ設定処理を示すフローチャートである。図５で示すフローチャートは、マスタＣＰＵをＣＰＵ＃０とし、スレイブＣＰＵをＣＰＵ＃１～ＣＰＵ＃３とした場合のフローチャートである。ＣＰＵ＃０は、ブート処理を実行する（ステップＳ５０１）。ブート処理により、ＣＰＵ＃０はＯＳ２０６＃０～ＯＳ２０６＃３のメモリイメージをＲＡＭ１０３に展開する。展開後、ＣＰＵ＃０は、カーネル起動し（ステップＳ５０２）、ＣＰＵ＃１～ＣＰＵ＃３もカーネル起動する（ステップＳ５０３）。

【００７４】

以下、スレイブＣＰＵのフローチャートについては、説明の簡略化のため、ＣＰＵ＃１について記述する。また、ステップＳ５１０の処理での割込ハンドラ２０７、ステップＳ５１２の処理での割込フラグテーブル２０４について、図５に示すフローチャートではＣＰＵ＃１～ＣＰＵ＃３の各々に対応する意味で、接尾記号である“＃１”～“＃３”の記載を行っていない。しかし、下記説明文中では、ＣＰＵ＃１について説明しているため、割込ハンドラ２０７＃１、割込フラグテーブル２０４＃１として記載する。

【００７５】

ＣＰＵ＃０は、全てのドライバを起動したか否かを判断する（ステップＳ５０４）。起動していないドライバが存在する場合（ステップＳ５０４：Ｎｏ）、ＣＰＵ＃０は、起動していないドライバについて、ドライバコンテキスト２０５を確保する（ステップＳ５０５）。確保後、ＣＰＵ＃０は、ドライバコンテキスト２０５のアドレスをＣＰＵ＃１～ＣＰＵ＃３に通知する（ステップＳ５０６）。通知後、ＣＰＵ＃０は、割込ハンドラ２０７＃０に確保したドライバコンテキスト２０５を設定し（ステップＳ５０７）、ステップＳ５０４の処理に移行する。

【００７６】

ＣＰＵ＃１は、全てのドライバコンテキスト２０５を設定したかを判断する（ステップＳ５０８）。全てのドライバコンテキスト２０５を設定していない場合（ステップＳ５０８：Ｎｏ）、ＣＰＵ＃１は、ステップＳ５０６にてＣＰＵ＃０から通知されたドライバコンテキスト２０５のアドレスを取得する（ステップＳ５０９）。取得後、ＣＰＵ＃１は、割込ハンドラ２０７＃１に取得したドライバコンテキスト２０５を設定し（ステップＳ５１０）、ステップＳ５０８の処理に移行する。

【００７７】

全てのドライバを起動した場合（ステップＳ５０４：Ｙｅｓ）、ＣＰＵ＃０は、キャッシュ保護領域上に、割込フラグテーブル２０４＃０を確保し（ステップＳ５１１）、ドライバ設定処理を終了する。また、全てのドライバコンテキスト２０５を設定した場合（ステップＳ５０８：Ｙｅｓ）、ＣＰＵ＃１は、キャッシュ保護領域上に、割込フラグテーブル２０４＃１を確保し（ステップＳ５１２）、ドライバ設定処理を終了する。

【００７８】

図６は、割込制御処理を示すフローチャートである。図６で示すフローチャートも、図５と同様に、マスタＣＰＵをＣＰＵ＃０とし、スレイブＣＰＵをＣＰＵ＃１～ＣＰＵ＃３とする。また、説明の簡略化のため、スレイブＣＰＵのフローチャートについては、ＣＰＵ＃１について記述する。また、ステップＳ６０８の処理での割込ハンドラ２０７、ステ

ップＳ６０９の処理での割込フラグテーブル２０４について、図６に示すフローチャートではＣＰＵ＃１～ＣＰＵ＃３の各々に対応する意味で、接尾記号である“＃１”～“＃３”の記載を行っていない。しかし、下記説明文中では、ＣＰＵ＃１について説明しているため、割込ハンドラ２０７＃１、割込フラグテーブル２０４＃１として記載する。

【００７９】

ＣＰＵ＃０は、デバイス２０１＃０～デバイス２０１＃２からのイベント発生を検出したかを判断する（ステップＳ６０１）。イベント発生を検出していない場合（ステップＳ６０１：Ｎｏ）、ＣＰＵ＃０は、通常運用としてプロセスをディスパッチし（ステップＳ６０２）、ステップＳ６０１の処理に移行する。

【００８０】

イベント発生を検出した場合（ステップＳ６０１：Ｙｅｓ）、ＣＰＵ＃０は、割込ハンドラ２０７＃０の処理を開始する（ステップＳ６０３）。具体的には、ＣＰＵ＃０は、今まで実行中であったプロセスをプロセスのコンテキスト領域に退避し、割込ハンドラ２０７＃０を実行できるようにする。続けて、ＣＰＵ＃０は、割込ハンドラ２０７＃０の処理として、割込フラグテーブル２０４＃０にて、イベント発生したデバイス２０１に対応する割込フラグにＯＮを書き込む（ステップＳ６０４）。書き込み後、ＣＰＵ＃０は、割込処理の実行要求となるＣＰＵ間のソフトウェア割込をＣＰＵ＃１～ＣＰＵ＃３に通知し（ステップＳ６０５）、ステップＳ６０１の処理に移行する。

【００８１】

ＣＰＵ＃１は、ＣＰＵ間のソフトウェア割込を捕捉したかを判断する（ステップＳ６０６）。ＣＰＵ間のソフトウェア割込を捕捉していない場合（ステップＳ６０６：Ｎｏ）、ＣＰＵ＃１は、通常運用としてプロセスをディスパッチし（ステップＳ６０７）、ステップＳ６０６の処理に移行する。

【００８２】

ステップＳ６０５によるＣＰＵ＃０の通知により、ＣＰＵ間のソフトウェア割込を捕捉した場合（ステップＳ６０６：Ｙｅｓ）、ＣＰＵ＃１は、割込ハンドラ２０７＃１の処理を開始する（ステップＳ６０８）。続けて、ＣＰＵ＃１は、割込ハンドラ２０７＃１の処理として、割込フラグテーブル２０４＃１に、ＯＮとなる割込フラグが存在するかを判断する（ステップＳ６０９）。

【００８３】

割込フラグが全てＯＦＦである場合（ステップＳ６０９：Ｎｏ）、ＣＰＵ＃１は、割込処理の実行要求を破棄することになり、ステップＳ６０６の処理に移行する。ＯＮとなる割込フラグが存在した場合（ステップＳ６０９：Ｙｅｓ）、ＣＰＵ＃１は、該当の割込フラグにＯＦＦを書き込む（ステップＳ６１０）。なお、ステップＳ６１０の処理前にて、ＣＰＵ＃１は、ＯＮであった割込フラグのアドレス等をレジスタ等に格納することで、どの割込フラグがＯＮとなっていたかを記憶しておく。書き込み後、ＣＰＵ＃１は、ＯＮであった割込フラグに対応する割込処理を実行し（ステップＳ６１１）、ステップＳ６０６の処理に移行する。

【００８４】

以上説明したように、割込制御方法、マルチコアプロセッサシステム、および割込制御プログラムによれば、マスタとなる第１のコアがコヒーレンシ領域の割込フラグをＯＮにし、他のコアのうち割込フラグをＯＦＦにした第２のコアが割込処理を実行する。これにより、マルチコアプロセッサシステムは、割込処理を最速で実行可能な第２のコアに実行させることができ、応答性能を向上させることができる。

【００８５】

また、マルチコアプロセッサシステムは、第２のコアが割込処理を実行中に、別のデバイスからの割込信号を検出すると、別の割込フラグをＯＮにし、ＯＮからＯＦＦにしたコアが、別のデバイスからの割込信号に対応する割込処理を実行する。このように、マルチコアプロセッサシステムは、割込処理を複数のコアに分散して実行することができる。また、通常のプロセスを実行中の各コアは、各プロセスの動作を乱すことなく、ソフトウェ

ア割込を受けた時点で即座に応答可能なコアが、割込処理を分散することが可能になる。

【００８６】

また、マルチコアプロセッサシステムは、複数のコアのうち第１および第２のコアを除く第３のコアが、通知された実行要求を破棄してもよい。これにより、割込処理を最速で実行可能な第２のコア以外で実行要求を通知されたコアが割込処理を行うことがないように排他制御されることになり、割込処理を多重に実行されることを防ぐことができる。また、排他制御に関して、キャッシュコヒーレンシ機構を利用して排他制御を行っているため、たとえば、特許文献２にかかる技術で行っているＯＳのセマフォ機構を用いた排他制御より、排他制御にかかる時間コストを短くすることができる。

【００８７】

また、マルチコアプロセッサシステムは、マスタとなる第１のコアを含めた複数のコアに、割込処理の実行要求を通知してもよい。これにより、マルチコアプロセッサシステムは、マスタＣＰＵの負荷が少なければ、マスタＣＰＵも割込処理の実行対象とすることができる。複数のコアが、マスタＣＰＵと１つのスレイブＣＰＵという、コアが２つである場合であっても、割込処理を分散することができる。

【００８８】

また、マルチコアプロセッサシステムは、割込処理を最速で取りかかれるコアで実行することになり、割込処理を行うデバイスドライバのうち、特に単動作型のドライバについて、イベント発生から動作応答までの時間を短くすることができる。連続動作型のドライバについては、デッドライン時間が存在するケースが多いため、デッドライン時間に基づいて実行開始予測を行うこともできる。しかしながら、単動作型のドライバについては、基準となる時間が存在しなく、可能な限り速く処理するといったケースになることが多いため、割込処理を最速で取りかかれるコアで実行するという本実施の形態が有効である。

【００８９】

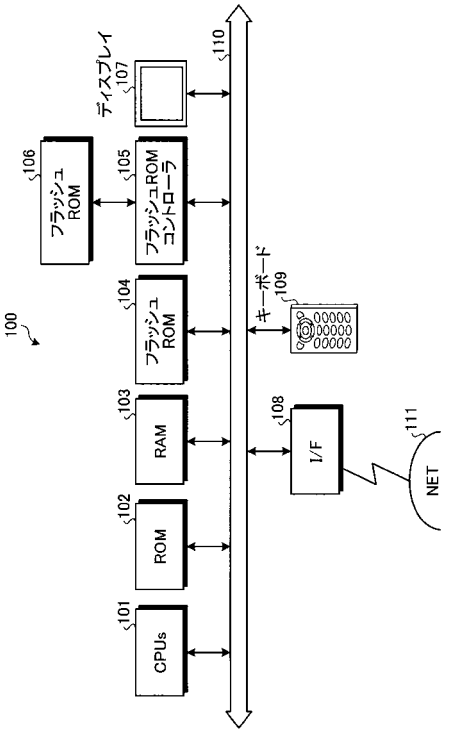
なお、本実施の形態で説明した割込制御方法は、予め用意されたプログラムをパーソナル・コンピュータやワークステーション等のコンピュータで実行することにより実現することができる。本割込制御プログラムは、ハードディスク、フレキシブルディスク、ＣＤ－ＲＯＭ、ＭＯ、ＤＶＤ等のコンピュータで読み取り可能な記録媒体に記録され、コンピュータによって記録媒体から読み出されることによって実行される。また本割込制御プログラムは、インターネット等のネットワークを介して配布してもよい。

【符号の説明】

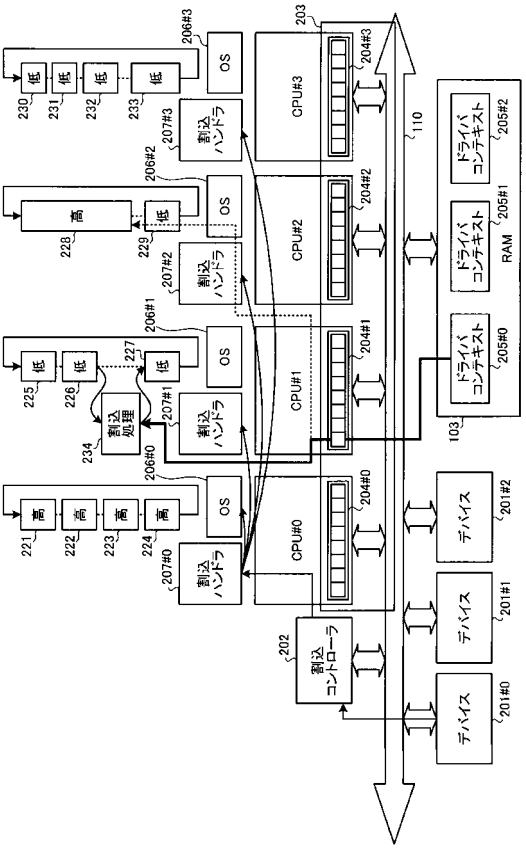
【００９０】

＃０～＃３　ＣＰＵ
１０３　ＲＡＭ
１１０　バス
２０１　デバイス
２０３　キャッシュコヒーレンシ機構
２０４　割込フラグテーブル
２０５　ドライバコンテキスト
３０１　書込部
３０２　通知部
３０３　書込部
３０４　実行部
３０５　書込部
３０６　実行部
３０７　破棄部

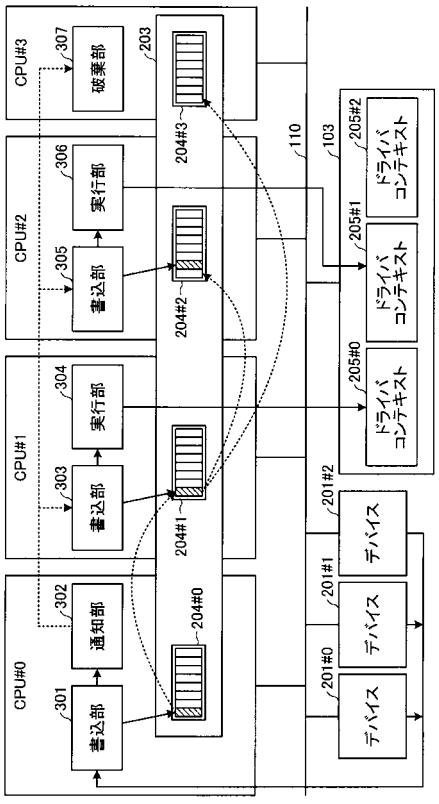
【図１】


100
CPUs
101
ROM
102
RAM
103
フラッシュROM
104
フラッシュROMコントローラ
105
フラッシュROM
106
ディスプレイ
107
I/F
108
キーボード
109
110
NET
111

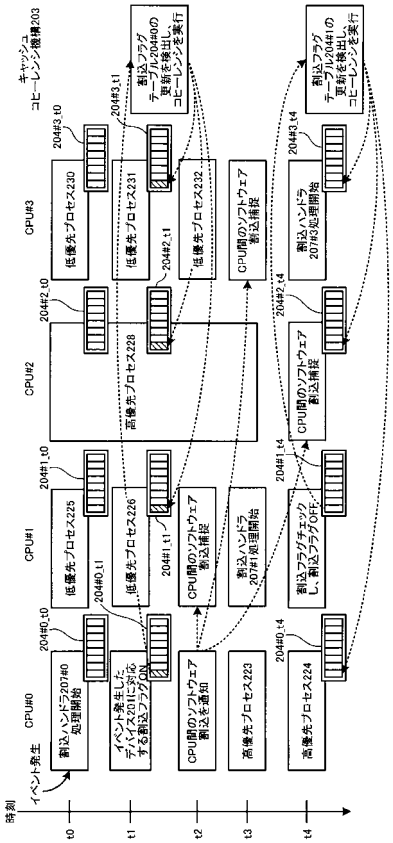
【図２】


CPU#0
CPU#1
CPU#2
CPU#3
割込コントローラ
202
デバイス
201#0
201#1
201#2
ドライバコンテキスト
205#0
205#1
205#2
RAM
205#1
割込ハンドラ
207#0
207#1
207#2
207#3
割込処理
234
OS
高
低
221
222
223
224
225
226
227
228
229
230
231
232
233
204#0
204#1
204#2
204#3
103
110

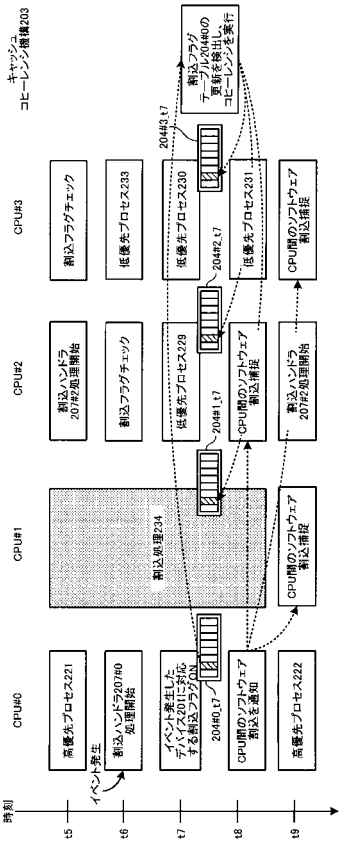
【図３】


CPU#0
CPU#1
CPU#2
CPU#3
書込部
301
通知部
302
書込部
303
実行部
304
書込部
305
実行部
306
破棄部
307
204#0
204#1
204#2
204#3
デバイス
201#0
201#1
201#2
ドライバコンテキスト
205#0
205#1
205#2
103
110

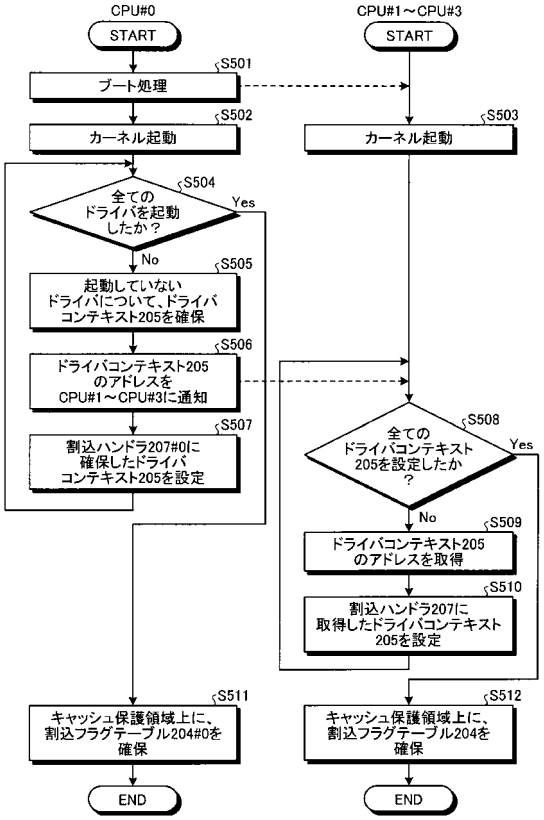
【図４Ａ】


時刻
t0
t1
t2
t3
t4
イベント発生
CPU#0
CPU#1
CPU#2
CPU#3
キャッシュコヒーレンシ機構203
割込ハンドラ207#0処理開始
イベント発生したデバイス201に対応する割込フラグON
CPU間のソフトウェア割込を通知
高優先プロセス223
高優先プロセス224
低優先プロセス225
低優先プロセス226
CPU間のソフトウェア割込捕捉
割込ハンドラ207#1処理開始
割込フラグチェックし、割込フラグOFF
高優先プロセス228
低優先プロセス230
低優先プロセス231
低優先プロセス232
割込ハンドラ207#3処理開始
割込フラグテーブル204#0の更新を検出しコヒーレンシを実行
割込フラグテーブル204#1の更新を検出しコヒーレンシを実行
割込フラグテーブル204#3の更新を検出しコヒーレンシを実行

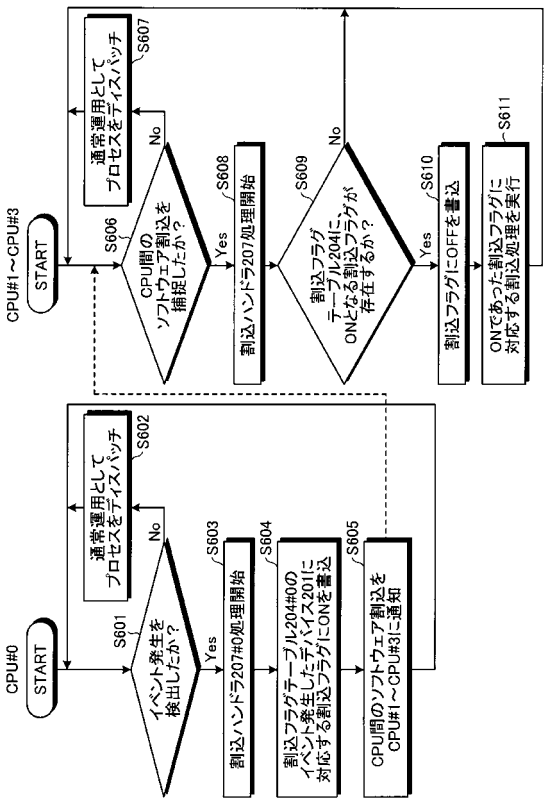
【図４Ｂ】


キャッシュ
コヒーレンシ機構203
CPU#3
CPU#2
CPU#1
CPU#0
割込フラグチェック
低優先プロセス233
低優先プロセス230
低優先プロセス231
CPU間のソフトウェア
割込捕捉
割込ハンドラ
207#2処理開始
割込フラグチェック
低優先プロセス229
CPU間のソフトウェア
割込捕捉
割込ハンドラ
207#2処理開始
割込処理234
CPU間のソフトウェア
割込捕捉
高優先プロセス221
割込ハンドラ207#0
処理開始
イベントに応じた
ディスパッチ
CPU間のソフトウェア
割込を通知
高優先プロセス222
イベント発生
204#3,t7
204#2,t7
204#1,t7
204#0,t7
割込フラグ
テーブル204#0の
更新を検出し、
コヒーレンシを実行
時刻
t5
t6
t7
t8
t9

【図５】


CPU#0
START
S501
ブート処理
S502
カーネル起動
S504
全ての
ドライバを起動
したか？
Yes
No
S505
起動していない
ドライバについて、ドライバ
コンテキスト205を確保
S506
ドライバコンテキスト205
のアドレスを
CPU#1～CPU#3に通知
S507
割込ハンドラ207#0に
確保したドライバ
コンテキスト205を設定
S511
キャッシュ保護領域上に、
割込フラグテーブル204#0を
確保
END
CPU#1～CPU#3
START
S503
カーネル起動
S508
全ての
ドライバコンテキスト
205を設定したか
？
Yes
No
S509
ドライバコンテキスト205
のアドレスを取得
S510
割込ハンドラ207に
取得したドライバコンテキスト
205を設定
S512
キャッシュ保護領域上に、
割込フラグテーブル204を
確保
END

【図６】


CPU#1～CPU#3
START
S606
CPU間の
ソフトウェア割込を
捕捉したか？
No
S607
通常運用として
プロセスをディスパッチ
Yes
S608
割込ハンドラ207処理開始
S609
割込フラグ
テーブル204に、
ONとなる割込フラグが
存在するか？
No
Yes
S610
割込フラグにOFFを書込
S611
ONであった割込フラグに
対応する割込処理を実行
CPU#0
START
S601
イベント発生を
検出したか？
No
S602
通常運用として
プロセスをディスパッチ
Yes
S603
割込ハンドラ207#0処理開始
S604
割込フラグテーブル204#0の
イベント発生したデバイス201に
対応する割込フラグにONを書込
S605
CPU間のソフトウェア割込を
CPU#1～CPU#3に通知

フロントページの続き

(72)発明者　栗原　康志
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　篠塚　隆

(56)参考文献　特開２００１－２３６２３８（ＪＰ，Ａ）
特表２０１０－５０７１６０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４６－９／５４
１２／０８－１２／１２