(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5104863号
(P5104863)
(45)発行日　平成24年12月19日(2012.12.19)　(24)登録日　平成24年10月12日(2012.10.12)

(51)Int.Cl.　F I
G06F 9/38 (2006.01)　G06F 9/38 380X
G06F 9/38 370A

請求項の数 9　(全 22 頁)

(21)出願番号　特願2009-520141 (P2009-520141)
(86)(22)出願日　平成19年6月20日 (2007.6.20)
(86)国際出願番号　PCT/JP2007/000653
(87)国際公開番号　WO2008/155800
(87)国際公開日　平成20年12月24日 (2008.12.24)
審査請求日　平成21年8月7日 (2009.8.7)

(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　100094514
弁理士　林　恒徳
(74)代理人　100094525
弁理士　土井　健二
(72)発明者　秋月　康伸
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　吉田　利雄
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】演算処理装置及び演算処理装置の制御方法

(57)【特許請求の範囲】
【請求項１】
複数の命令をそれぞれ含む複数のスレッドを実行する演算処理装置において、
命令をデコードする命令デコーダと、
前記命令デコーダのデコード結果に基づき、命令を識別する命令識別子と前記命令のスレッドを識別するスレッド識別子とを保持するエントリを複数保持するリザベーションステーションと、
前記リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択するとともに、前記複数のスレッドの内、いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、前記複数のスレッドが交互に実行されるように、前記エントリに保持された前記スレッド識別子に基づいて、前記リザベーションステーションから前記エントリを選択する選択部と、
前記選択されたエントリに保持された命令識別子に対応する命令を実行する実行部とを有する
ことを特徴とする演算処理装置。
【請求項２】
請求項１の演算処理装置において、
前記選択部は、
前記複数のスレッドの内、いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、時間の経過に応じて交互に選択するスレッドを変更するサイク

ルを変更する
ことを特徴とする演算処理装置。
【請求項３】
請求項１の演算処理装置において、
前記選択部は、
前記一定期間内に完了しないことが検出された命令が完了した時は、前記複数のスレッドが交互に実行されるように、各エントリに保持されたスレッド識別子に基づいて前記リザベーションステーションからエントリを選択する制御を終了する
ことを特徴とする演算処理装置。
【請求項４】
請求項１の演算処理装置において、
前記リザベーションステーションは、
前記命令デコーダからの命令識別子を、前記リザベーションステーションが有するエントリに保持させるエントリ生成回路を有し、
前記選択部は、
前記複数のスレッドの内、いずれか一つのスレッドに含まれる命令が一定期間内に完了しないことを検出した場合に、前記複数のスレッドに属する命令を交互に実行するように、スレッド番号を選択するスレッド選択回路と、
前記スレッド選択回路で選択されたスレッド番号と、各エントリに保持されたスレッド識別子が一致した場合、一致したスレッド識別子を保持するエントリが保持する命令識別子に対応する命令を前記リザベーションステーションから実行可能にする実行可能選択回路と、
前記リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択する実行エントリ選択回路とを有する
ことを特徴とする演算処理装置。
【請求項５】
請求項４の演算処理装置において、
前記実行エントリ選択回路は、
前記実行可能な命令に対応する命令識別子を保持するエントリが、先行して選択された先行エントリに保持された命令識別子に対応する命令を実行するタイミング又は前記先行エントリに保持された命令識別子に対応する命令の実行結果を結果レジスタに格納するタイミングと重ならないタイミングで、前記実行可能な命令に対応する命令識別子を保持するエントリを選択する
ことを特徴とする演算処理装置。
【請求項６】
複数の命令をそれぞれ含む複数のスレッドを実行する演算処理装置の制御方法において、
前記演算処理装置が有する命令デコーダが、命令をデコードし、
前記演算処理装置が有するリザベーションステーションが、前記命令デコーダのデコード結果に基づき、命令を識別する命令識別子と命令のスレッドを識別するスレッド識別子を各々有する複数のエントリを保持し、
前記演算処理装置が有する選択部が、前記リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択するとともに、前記複数のスレッドの内いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、前記複数のスレッドが交互に実行されるように、前記エントリに保持された前記スレッド識別子に基づいて、前記リザベーションステーションから前記保持したエントリを選択し、
前記演算処理装置が有する実行部が、前記選択されたエントリに保持された命令識別子に対応する命令を実行する
ことを特徴とする演算処理装置の制御方法。

【請求項７】
請求項６の演算処理装置の制御方法において、
前記選択部は、前記複数のスレッドの内いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、時間の経過に応じて交互に選択するスレッドを変更するサイクルを変更する
ことを特徴とする演算処理装置の制御方法。
【請求項８】
請求項６の演算処理装置の制御方法において、
前記一定期間内に完了しないことが検出された命令が完了した場合、前記選択部は、前記複数のスレッドが交互に実行されるように、各エントリに保持されたスレッド識別子に基づいて前記リザベーションステーションからエントリを選択する制御を終了する
ことを特徴とする演算処理装置の制御方法。
【請求項９】
請求項６の演算処理装置の制御方法において、
前記選択部は、前記実行可能な命令に対応する命令識別子を保持するエントリが、先行して選択された先行エントリに保持された命令識別子に対応する命令を実行するタイミング又は前記先行エントリに保持された命令識別子に対応する命令の実行結果を結果レジスタに格納するタイミングと重ならないタイミングで、前記実行可能な命令に対応する命令識別子を保持するエントリを選択する
ことを特徴とする演算処理装置の制御方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、複数のスレッドが、同時マルチスレッド方式で動作して、アウト・オブ・オーダー処理で、命令制御を行う演算処理装置及び演算処理装置の制御方法に関し、特に、同時マルチスレッド方式で動作する場合のハング状態の発生を防止する演算処理装置及び演算処理装置の制御方法に関する。
【背景技術】
【０００２】
ＣＰＵ（Central Processor Unit）の処理の高速化が、要求されている。このため、従来から様々な技術を使用してＣＰＵの処理を向上させてきた。その方法として、パイプライン処理や、並列に処理を行うスーパースカラ方式や、プログラム命令の順番通りに実行を行わずに、入力データが揃った命令から実行を行うアウト・オブ・オーダー実行方式がある。
【０００３】
アウト・オブ・オーダー実行方式は、先の命令処理に必要なデータが揃っていなくても、後の命令処理に必要なデータが揃えっていた場合、後の命令から先に実行するＣＰＵの性能を向上するための技術である（例えば、特許文献１参照）。
【０００４】
例えば、プログラムに記述された順に命令を処理する場合に、先の命令処理１が、メモリアクセスを伴う命令であり、後の命令処理２が、メモリアクセスを伴わない命令であると仮定すると、命令処理１のメモリアクセスと並行して、命令処理２を実行し、命令処理２の実行後、メモリアクセスの終了した命令処理１を実行する。
【０００５】
さらに、単一のプログラムを走行するのではなく、複数のプログラムを走行することで、ＣＰＵの処理を向上させるマルチスレッド方式も提案されている（例えば、特許文献２参照）。
【０００６】
複数のプログラムを走行するマルチスレッド方式は、ＣＰＵのプログラマブルな資源を、複数組用意しておくことで、ソフトウェアから見ると、見掛け上は、複数のＣＰＵに見

えるために、複数のプログラムを実行することが可能となる構造である。

【０００７】

このマルチスレッドの方式の１つとして、ＶＭＴ(Vertical Multi Threading)方式がある。この方式は、同時に、１つのプログラムしか走行できないが、長時間のデータ待ちが発生したときや、一定の時間の間隔などで、プログラムを切り替えて走行する。ＶＭＴ方式の回路量は、プログラマブルな資源を、プログラムの数分用意しなければならないが、同時に、１つのプログラムが走行するために、追加する回路量も少なく、実現が容易である。

【０００８】

一方、別のマルチスレッドの方式として、複数のプログラムを同時に走行する同時マルチスレッド方式(ＳＭＴ方式）がある。この方式は、複数のプログラムが同時に走行するために、単一のプログラムが走行していたときよりも、回路の制御が複雑になることや、リソースの増加が予想されるが、同時に複数のプログラムが走行するために、効率よく回路を使用することが可能となる。

【０００９】

アウト・オブ・オーダー実行を処理するためのリザベーションステーションの制御は、機能の実行の準備ができたエントリから機能の実行を行うことを可能とする。

【００１０】

機能の実行がパイプライン処理で行われ、機能の実行に必要な時間が異なる種類の命令を実行する場合、リザベーションステーションは、機能の実行の結果を出力するタイミングが、同じにならないように、エントリの実行を制御する。

【００１１】

図１５は、浮動小数点用リザベーションステーションのエントリ実行制御のタイムチャート図である。浮動小数点演算では、パイプライン処理で実行された結果は、結果レジスタに格納されるが、リザベーションステーションは、結果レジスタに格納するタイミングが、重ならないように実行するエントリを選択する。

【００１２】

図１５において、リザベーションステーションから実行されるエントリが、実行に４サイクル必要なエントリ（先行命令）であり、後続の命令が、実行に２サイクル必要なエントリである場合の後続の命令の制御例を示す。

【００１３】

図１５において、Ｔ１～Ｔ７は、サイクル、Ｐは、リザベーションステーションから実行エントリを選択する処理、Ｂは、機能の実行に必要なオペランドデータの読み出し処理、Ｘは、機能実行及び最後のサイクルで実行結果を結果レジスタに格納する処理、Ｕは、機能の実行の結果をレジスタ更新バッファに格納する処理を示す。

【００１４】

先行命令は、機能実行にＸ１，Ｘ２，Ｘ３，Ｘ４の４サイクル必要であり、後続命令は、機能実行にＸ１，Ｘ２の２サイクル必要である。サイクルＴ１で、リザベーションステーションが、先行命令を選択した場合、Ｕが重ならないため、Ｔ２のタイミングでは、２サイクルの後続命令は実行可能となる。Ｔ３のタイミングでは、２サイクルの後続命令を実行した場合、４サイクルの先行命令と、結果レジスタに格納するタイミングＵ（Ｔ７）が同じになるため、実行不可能となる。そして、Ｔ４のタイミングでは、２サイクルの後続命令は実行可能となる。

【００１５】

図１６は、このようなパイプライン制御のリザベーションステーションの選択動作により、単一スレッドで動作している場合のエントリ実行例を示す、図中、Ｐ，Ｂ，Ｘ，Ｕが示すものは、図１５で示したものと同一である。

【００１６】

図１６は、実行に４サイクル必要なエントリが連続して、リザベーションステーションから選択されているときに、実行に２サイクル必要なエントリが命令デコーダからデコー

ドされ、その後には、実行に４サイクル必要なエントリが、命令デコーダから連続してデコードされた状況のタイムチャートである。

【００１７】

リザベーションステーションは、実行可能となったエントリからエントリを発行（実行）する。又、同時に幾つも実行可能なエントリがある場合については、デコードされた順番通りにエントリを選択して実行する。

【００１８】

このために、機能の実行の準備ができているエントリであったとしても、先に実行しているエントリの結果を出力するタイミングによって実行可能なエントリにならないエントリが発生する場合がある。

【００１９】

このような状況が連続して長時間続く場合になると、リザベーションステーションから実行することができなくなる。図１６では、実行に２サイクル必要なエントリは、実行可能となって、リザベーションステーションから実行しようとしても、４サイクルの先行命令と結果レジスタに格納するタイミングが同じになるため、実行不可能な状態になる。

【００２０】

単一スレッドの場合には、リザベーションステーションから発行できない状態になってから、ある一定数の命令が、命令デコーダからデコードされると、命令完了制御機能のエントリがＦＵＬＬ状態となる。

【００２１】

即ち、リザベーションステーションから発行できないために、命令の完了ができなくなるためである。後続の命令は、リザベーションステーションからは実行できるが、命令の完了はできない状態となる。

【００２２】

このために、命令の完了を制御する機能のエントリがＦＵＬＬ状態となり、命令デコーダから命令がデコードされない状態（命令デコーダが停止した状態）となる。命令がデコードされないために、リザベーションステーションに新たなエントリが作成されないために、実行できなかったエントリ（図１６の２サイクルのエントリ）が、例えば、サイクルＴ５で、実行することができるようになり、命令の完了も可能となる。

【特許文献１】特開２００７－８７１０８号公報

【特許文献２】特表２００６－５０２５０４号公報（ＷＯ２００４／０３４２０９号）

【発明の開示】

【発明が解決しようとする課題】

【００２３】

一方、同時マルチスレッド方式では、リザベーションステーションのエントリを、スレッド間で共有して構成する場合、リザベーションステーションのエントリのスレッドに関係なく、機能の実行の準備ができたエントリの内、先に実行しているエントリの結果を出力するタイミングが同じにならないようなエントリを、実行可能なエントリとして、リザベーションステーションから選択され、実行される。

【００２４】

この同時マルチスレッド方式においても、シングルスレッド方式と同様に、機能の実行の準備ができているエントリであったとしても、先に実行しているエントリの結果を出力するタイミングによって実行可能なエントリにならないエントリが発生する場合がある。このような状況が連続して長時間続く場合になると、リザベーションステーションから実行することができなくなる。

【００２５】

図１７は、同時マルチスレッドで２つのスレッド０，１が動作しているときに、スレッド０に、実行に４サイクル必要なエントリが、連続してリザベーションステーションから実行され、命令デコーダから、実行に４サイクル必要な命令がデコードされている状況の例を示す。

【００２６】
このような状況において、スレッド１に、実行に２サイクル必要なエントリが命令デコーダからデコードされ、その後には、スレッド０に実行に４サイクル必要なエントリが連続してデコードされると、実行に２サイクル必要なスレッド１のエントリは、リザベーションステーションから実行しようとしても、先行命令と、結果レジスタに格納するタイミングが同じになるため、実行不可能な状態になる。
【００２７】
同時マルチスレッドの場合には、シングルスレッド方式とは、異なり、リザベーションステーションから実行できないエントリが発生した場合でも、他のスレッドは、リソースがＦＵＬＬ状態になることはなく、動作することが可能なために、単一スレッドのように、命令デコーダは停止することがない。
【００２８】
即ち、同時マルチスレッド方式では、スレッド０の命令は、命令を実行した後も、完了することができるので、命令デコーダは、スレッド０の命令をデコードすることが可能である。このため、スレッド０が止まらずに動作し続けることが可能である。
【００２９】
しかし、スレッド１のエントリは、リザベーションステーションから実行できない状態となるために、命令を完了することができなくなり、ハング状態に陥ってしまうことになる。
【００３０】
即ち、リザベーションステーションから実行できない状態の場合、一定期間命令を完了することができていない状態（ハング状態）を異常状態として検出して、ＣＰＵが停止することになる。
【００３１】
従って、本発明の目的は、同時マルチスレッド方式の処理において、ハング状態になる前に、リザベーションステーションのエントリを実行可能な状態にするための演算処理装置及び演算処理装置の制御方法を提供することにある。
【００３２】
又、本発明の他の目的は、同時マルチスレッド方式の処理において、リザベーションステーションのエントリを実行可能な状態にするとともに、大幅な性能低下を防止するための演算処理装置及び演算処理装置の制御方法を提供することにある。
【００３３】
更に、本発明の別の目的は、同時マルチスレッド方式の処理において、リザベーションステーションのエントリを実行可能な状態にし、ＣＰＵの停止を防止するための演算処理装置及び演算処理装置の方法を提供することにある。
【課題を解決するための手段】
【００３４】
この目的の達成のため、本発明の演算処理装置は、複数の命令をそれぞれ含む複数のスレッドを実行する演算処理装置において、命令をデコードする命令デコーダと、前記命令デコーダのデコード結果に基づき、命令を識別する命令識別子と前記命令のスレッドを識別するスレッド識別子とを保持するエントリを複数保持するリザベーションステーションと、前記リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択するとともに、前記複数のスレッドの内、いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、前記複数のスレッドが交互に実行されるように、前記エントリに保持された前記スレッド識別子に基づいて、前記リザベーションステーションから前記エントリを選択する選択部と、前記選択されたエントリに保持された命令識別子に対応する命令を実行する実行部とを有する。
【００３５】
又、本発明の演算処理装置の制御方法は、複数の命令をそれぞれ含む複数のスレッドを実行する演算処理装置の制御方法において、前記演算処理装置が有する命令デコーダが、

命令をデコードし、前記演算処理装置が有するリザベーションステーションが、前記命令デコーダのデコード結果に基づき、命令を識別する命令識別子と命令のスレッドを識別するスレッド識別子を各々有する複数のエントリを保持し、前記演算処理装置が有する選択部が、前記リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択するとともに、前記複数のスレッドの内いずれか一のスレッドに含まれる命令が一定期間内に完了しないことを検出した場合、前記複数のスレッドが交互に実行されるように、前記エントリに保持された前記スレッド識別子に基づいて、前記リザベーションステーションから前記保持したエントリを選択し、前記演算処理装置が有する実行部が、前記選択されたエントリに保持された命令識別子に対応する命令を実行する。

【００３６】

更に、本発明は、好ましくは、前記ハング防止回路は、前記一定期間に完了することができないことを検出したことに応じて、前記リザベーションステーションから実行するエントリのスレッドを同一のスレッドに選択するためのスレッド選択回路を有する。

【００３７】

更に、本発明は、好ましくは、前記ハング防止回路は、前記スレッド選択回路で選択されたスレッドと、前記リザベーションステーションのエントリのスレッドが一致したときに、前記エントリを、前記リザベーションステーションから実行することが可能とするための実行可能選択回路を更に有する。

【００３８】

更に、本発明は、好ましくは、前記スレッド選択回路は、時間の経過によって選択するスレッドを変更するスレッド選択回路で構成される。

【００３９】

更に、本発明は、好ましくは、前記スレッド選択回路は、動作しているスレッドを示す信号に応じて、動作していないスレッドの選択を禁止する。

【００４０】

更に、本発明は、好ましくは、前記ハング防止回路は、前記完了することができなかった命令が完了した時は、前記リザベーションステーションの制御を停止する。

【００４１】

更に、本発明は、好ましくは、前記リザベーションステーションは、前記命令デコーダからの命令を、前記リザベーションステーションにエントリするエントリ生成回路と、前記リザベーションステーションから実行可能なエントリを選択する実行エントリ選択回路とを有する。

【００４２】

更に、本発明は、好ましくは、前記演算処理リザベーションステーションは、前記実行の準備ができたエントリが、前記先行エントリの実行又は実行結果の格納タイミングと重ならないようなタイミングで、前記実行の準備ができたエントリを実行する。

【発明の効果】

【００４３】

リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択する同時マルチスレッドで動作しているときに、命令が、一定期間完了しない状態を検出し、選択部が強制的に交互にスレッド選択制御するため、リザベーションステーションから実行できないエントリが存在する状態の場合には、実行され続けている１のスレッドの実行を停止することで、実行できない他のスレッドのエントリを実行可能な状態にすることができる。又、このようにしても、交互にスレッド選択するので、１のスレッドも選択でき、マルチスレッドの性能低下を防止できる。更に、エントリにスレッド識別子を付加しているので、マルチスレッドのリザベーションステーションからエントリを容易に交互に選択できる。

【図面の簡単な説明】

【００４４】

【図１】本発明の情報処理装置の一実施の形態のブロック図である。

【図２】本発明の一実施の形態の命令実行制御装置の構成図である。
【図３】図２の命令実行制御装置の動作フロー図である。
【図４】図２の命令実行制御装置の動作説明図である。
【図５】図２の命令実行制御装置により、スレッド選択動作の説明図である。
【図６】図２のスレッド選択回路のブロック図である。
【図７】図６の実行可能選択回路の実行選択処理フロー図である。
【図８】図６のスレッド選択回路のスレッド選択動作の説明図である。
【図９】図２の浮動小数点リザベーションステーションの動作説明図である。
【図１０】図２の固定小数点リザベーションステーションの動作説明図である。
【図１１】図６のスレッド選択回路のスレッド時間変更動作の説明図である。
【図１２】図６のスレッド決定回路の回路図である。
【図１３】図６のスレッドＩＤ生成回路の回路図である。
【図１４】図６のスレッド切り替え時間選択回路の回路図である。
【図１５】従来のリザベーションステーションのエントリ実行動作の説明図である。
【図１６】従来のシングルスレッド方式のエントリ実行動作の説明図である。
【図１７】従来のマルチスレッド方式のエントリ実行動作の説明図である。
【符号の説明】
【００４５】
１　命令フェッチアドレス生成器
２　１次命令キャッシュ
３　命令バッファ
４　命令デコーダ
５，６，７　リザベーションステーション
１０　オペランドアドレス生成器
１２、１５　演算器
１３，１６　更新バッファ
１４、１７　レジスタ
３０　スレッド選択回路
５０,６０,７０　エントリ生成回路
５２，６２，７２　実行可能選択回路
５６，６６，７６　実行エントリ選択回路
５４　主記憶リザベーションステーション
６４，７４　演算処理リザベーションステーション
【発明を実施するための最良の形態】
【００４６】
以下、本発明の実施の形態を、図面に従い、情報処理装置、命令実行制御装置、ハング防止機構、スレッド選択回路、他の実施の形態の順で説明する。しかし、本発明は、下記実施の形態に限らず、種々の変形が可能である。
【００４７】
（情報処理装置）
図１は、本発明の情報処理装置の一実施の形態の全体図である。図１に示すように、１次命令キャッシュ２と、１次データキャッシュ１１は、図示しない主記憶に接続された２次キャッシュに接続する。
【００４８】
命令フェッチを行うために、命令フェッチアドレス生成器１は、命令アドレスを選択し、選択された命令アドレスに対して命令フェッチリクエストを、１次命令キャッシュ２に与える。１次命令キャッシュ２からフェッチされた命令は、命令バッファ３に格納される。命令バッファ３からプログラムの順番通りに、命令デコーダ４に、命令の供給を行う。
【００４９】
命令デコーダ４は、プログラムの順番通りに命令のデコードを行う。命令デコーダ４は

、デコードする命令の種類に従って、命令の実行を制御する主記憶オペランドアドレス生成用リザベーションステーションユニット（ＲＳＡ:Reservation Station for Address generate）５、固定小数点演算用リザベーションステーションユニット（ＲＳＥ:Reservation
Station for Execute）６、浮動小数点演算用リザベーションステーションユニット（ＲＳＦ:Reservation Station for
Floating）７、分岐命令用リザベーションステーションユニット（ＲＳＢＲ:Reservation Station for BRanch）８に、必要なエントリを作成する。

【００５０】

即ち、命令デコーダ４は、フェッチしてきた命令を、イン・オーダーでデコードし、デコードされた命令は、機能の実行を制御するリザベーションステーションユニット５，６，７，８，９に、命令の種類によって、それぞれ格納される。そして、リザベーションステーションユニットは、演算用のリザベーションステーションユニット６，７と、主記憶オペランドアドレス生成用のリザベーションステーションユニット５を備える。

【００５１】

また、全てのデコードされた命令に対して、命令の完了を制御するコミットスタックエントリ（ＣＳＥ:Commit Stack Entry）９に、エントリを作成する。

【００５２】

デコードされた命令が、ＲＳＡ５にエントリを作成したときに、ロード命令であった場合には、ＲＳＡ５は、オペランドアドレス生成器１０にオペランドアドレスの生成を指示し、１次データキャッシュ１１から対応するデータを、固定小数点更新バッファ（ＧＵＢ:General Update Buffer）１３と、浮動小数点更新バッファ（ＦＵＢ:Floating Update Buffer）１６に読み出す。

【００５３】

又、デコードされた命令が、ＲＳＥ６，ＲＳＦ７にエントリを作成した場合には、各々演算器１２，１５を動作し、対応する演算処理を行う。デコードされた命令が、ＲＳＡ５、ＲＳＥ６，ＲＳＦ７にエントリを作成する場合に、ＧＵＢ１３とＦＵＢ１６に対応するレジスタリネームを行うことで、アウト・オブ・オーダー実行を行うことが可能となり、実行結果は、ＧＵＢ１３、ＦＵＢ１６に格納される。

【００５４】

リザベーションステーション５，６，７により、アウト・オブ・オーダーで実行された命令は、ＣＳＥ９の制御により、プログラムの順番通りに、命令の完了を行う。そして、完了した命令に対してのみ、固定小数点レジスタ１４や浮動小数点レジスタ１７やプログラムカウンタ（ＰＣ、ＮＥＸＴ＿ＰＣ）１８，１９などのプログラマブルな資源の更新を行う。

【００５５】

分岐予測機構２１は、分岐命令用リザベーションステーション８からの命令により、分岐予測を行い、命令フェッチアドレス生成器１を制御する。

【００５６】

従って、後述するように、リザベーションステーションユニット５，６，７，８により、演算サイクル毎に、スレッドを選択し、選択されたスレッドのエントリの実行を、オペランドアドレス生成器１０、演算器１２，１５に指示し、且つレジスタ１４，１７から選択されたスレッドのオペランドデータの読み出し、書込みを行い、同時マルチスレッド処理を実行する。

【００５７】

（命令実行制御装置）

図２は、本発明の命令実行制御装置の一実施の形態のブロック図、図３は、図２の構成の動作フロー図、図４は、図３の動作の説明図、図５は、図２乃至図４の動作による実行スレッドの説明図である。図２は、図１のＲＳＥ５、ＲＳＥ６，ＲＳＦ７の詳細図を示す。又、この実施の形態では、２つのスレッド（スレッド０と１）が同時に動作する場合に

ついて説明するが、スレッド数が３つ以上になっても実現は可能である。

【００５８】

図２において、図１で示したものと同一のものは、同一の記号で示してあり、リザベーションステーションユニット５，６，７は、各々、エントリ生成回路５０，６０，７０、実行可能選択回路５２，６２，７２、リザベーションステーション５４，６４，７４、実行エントリ選択回路５６，６６，７６とで構成される。この各実行可能選択回路５２，６２，７２に、スレッド選択回路３０が接続される。

【００５９】

このリザベーションステーションユニット５，６，７のエントリは、スレッド０と１とを共有して使用する。即ち、エントリは、エントリが有効であること示すＶＡＬＩＤ信号、エントリのスレッドを示すスレッドＩＤ、オペランドデータをアーキテクチャレジスタから読み出すことを示す信号と読み出しアドレス、レジスタ更新バッファから読み出すことを示す信号と読み出しアドレス、命令デコード時に命令ごとに割り当てられる命令の番号を示す命令識別子などを格納している。

【００６０】

スレッド選択回路３０は、図６にて詳細に説明するように、一定時間完了していない命令を、図１のＣＳＥ９が検出したことに応じて、ハング防止モードに走行モードを切り替え、ハング防止スレッドＩＤを選択する。

【００６１】

実行可能選択回路５２，６２，７２は、ハング防止モードにおいて、エントリ生成回路５０，６０，７０、リザベーションステーション５４，６４，７４のエントリと、スレッド選択回路３０のスレッドＩＤを比較し、実行可能性のあるエントリを選択する。

【００６２】

実行エントリ選択回路５６，６６，７６は、リザベーションステーション５４，６４，７４のエントリから実行可能性のあるエントリを選択し、機能実行部１０，１２，１５へ発行する。

【００６３】

この動作を説明する。複数のスレッドが、同時マルチスレッド方式で動作する場合、リザベーションステーション５４，６４，７４の実行は、アウト・オブ・オーダー実行が可能である。即ち、リザベーションステーション５４，６４，７４のエントリは、実行するための必要なオペランドデータの準備ができたエントリから、実行することが可能となる。

【００６４】

実行するエントリを選択する実行エントリ選択回路４６，５６，６６で選択されたエントリが、リザベーションステーション５４,６４，７４から実行される。実行する準備のできたエントリが、同時に幾つもある場合には、実行エントリ選択回路４６，５６，６６は、デコードされた順番通りにエントリを選択して実行する。

【００６５】

又、浮動小数点用のリザベーションステーション７４から実行されたエントリは、機能の実行がパイプライン処理で実行される。機能の実行が終了すると、結果レジスタに、機能の実行の結果を格納する。また、機能の実行が終了したことを、命令の完了する制御を行う機能ＣＳＥ９へ実行の完了を報告する。

【００６６】

命令完了制御機能ＣＳＥ９は、プログラムの順番通りに完了する制御を行う。機能の実行が完了すると、命令の完了を行うことが可能となるが、プログラムの順番通りに完了するために、最も古い命令が完了しないと、後続の命令の実行が完了していても、命令の完了をすることができない。命令の完了は、スレッド別に、完了していくために、他のスレッドによって、命令の完了の制御を邪魔されることはなく、命令の完了を行うことが可能である。

【００６７】

更に、図１５で説明したように、機能の実行が、パイプライン処理で実行された結果は、結果レジスタに格納されるが、リザベーションステーション７４は、結果レジスタに格納するタイミングが重ならないように、実行するエントリを選択する。

【００６８】

図３乃至図５を参照して、図２の構成の動作を説明する。スレッド選択回路３０は、通常走行状態で、ＣＳＥ９が、一定時間完了していない命令を検出した信号を発したかを判定する（Ｓ１０）。スレッド選択回路３０は、ＣＳＥ９から一定時間完了していない命令を検出した信号を受けると、ハング防止モードに、走行モードを変更する（Ｓ１２）。

【００６９】

図４に示す通常走行時には、スレッド選択回路３０が関与していないため、図５のように、リザベーションステーション５，６，７から実行する命令のスレッドは、各リザベーションステーション５，６，７の各々で選択されたエントリが、実行するスレッド番号になる。即ち、図５に示すように、リザベーションステーションのエントリ選択にスレッドの制限は与えられず、実行するスレッド番号は、リザベーションステーション５，６，７ごとに異なる。ただし、各リザベーションステーション５，６，７が個々に選択するために、偶然に、実行するスレッド番号が同じになることはあり得る。

【００７０】

一方、ハング防止モード時には、ハング防止モード中に動作するスレッド選択回路３０によって、選択されたスレッド番号のみ実行可能なスレッド番号となる。即ち、図５のように、リザベーションステーション５，６，７から実行するスレッド番号が同じになるように制御される。

【００７１】

そして、スレッド選択回路３０は、一定時間完了できなかった命令が完了したことをＣＳＥ９から通知されると、図４のように、通常走行モードに戻り、スレッド選択動作を停止する（Ｓ１４）。

【００７２】

このように、通常走行時は、リザベーションステーションが、実行可能なエントリをスレッドに関係なく実行していたものを、ハング防止モードで、各リザベーションステーションから決められたスレッドしか実行することができないように制御する。

【００７３】

このため、実行できない一のスレッドのエントリより後続の他のスレッドのエントリが、次々に実行されている状況のために、実行され続けている他のスレッドの実行を停止することで、実行できない一のスレッドのエントリを実行可能な状態にすることができる。また、リザベーションステーションが原因でないときに、ハング状態となった場合でも、リザベーションステーションからは、決められたスレッドしか実行することができないように制御する。

【００７４】

ただし、この制御を行うと、ハング状態になっていないスレッドのエントリは、実行を停止することになるために、ハング状態になった場合でも、大幅な性能低下にならないような制御を行い、ハング状態になっているスレッドを実行する。

【００７５】

（ハング防止機構）

図２の命令実行制御装置を更に詳細に説明する。図６は、図２のスレッド選択回路３０のブロック図、図７は、図２、図６の実行可能選択回路の処理フロー図、図８は、図６の構成によるスレッド選択方法の説明図、図９は、図６のリザベーションステーションのエントリの実行選択動作の説明図、図１０は、図６の固定小数点リザベーションステーションのエントリの実行選択動作の説明図である。

【００７６】

図６乃至図９は、図２の浮動小数点用リザベーションステーションユニット７の例で説明する。尚、図２の固定小数点用リザベーションステーションユニット６、オペランド生

成リザベーションユニット５も同様の構成である。

【００７７】

図６により、スレッド選択回路３０の構成を説明する。スレッド選択回路３０は、基本的に、時間軸に沿って、交互にスレッド番号を変更して、出力する。図６に示すように、スレッド選択回路３０は、タイマーカウンタ３２と、スレッド切り替え時間選択回路３４と、スレッドＩＤ生成回路３６と、スレッド決定回路３８と、ハング防止スレッドＩＤレジスタ４０と、ハング防止モード起動回路４２とを有する。

【００７８】

ハング防止モード起動回路４２は、図１のＣＳＥ９の一定時間完了していない命令を検出した検出信号に応じて、スレッド切り替え時間選択回路３４と、スレッド決定回路３８と、実行可能選択回路５２，６２，７２とを起動する。

【００７９】

スレッド切り替え時間選択回路３４は、同じスレッドを連続して選択する時間を示すスレッド切り替え時間を備え、スレッド切り替え時間になると、スレッド決定回路３８にスレッド切り替えを指示する。スレッド切り替え時間は、１サイクルで１加算されるタイマーカウンタ３２によって、カウンタの値がある値になったら、スレッド切り替え時間を変更することが可能である。

【００８０】

スレッド切り替え時間とタイマーカウンタのカウンタの値によって、スレッド決定回路３８は、スレッドＩＤ生成回路３６のスレッドＩＤを選択する。スレッドＩＤ生成回路３６は、ハング防止スレッドＩＤレジスタ４０のスレッドＩＤと異なるスレッドＩＤを、スレッド決定回路３８に出力する。このスレッド決定回路３８で選択されたスレッドが、ハング防止スレッドＩＤとなり、実行可能選択回路５２，６２，７２に送られ、リザベーションステーション５４，６４，７４から実行することのできるエントリを選択する。

【００８１】

従って、スレッド切り替え時間選択回路３４で、スレッド切り替え時間を選択することにより、時間の経過により同じスレッドを選択する時間を変更できる。

【００８２】

次に、図７により、実行可能選択回路５２，６２，７２を説明する。ハング防止モード中は、実行エントリ選択回路５６，６６，７６の１サイクル前のサイクルで、リザベーションステーション５４，６４，７４のエントリのスレッドが、スレッド選択回路３０で選択されたスレッドと一致するかを判定する（Ｓ２２）。一致しないと、ステップＳ２６に進む。

【００８３】

一方、スレッドが一致したエントリについては、実行する準備のできているかを判定し、実行準備ができていれば、そのエントリが、次サイクルでの実行エントリ選択回路５６，６６，７６で選択される可能性のあるエントリに決定する（Ｓ２４）。例えば、エントリにフラグを付与する。

【００８４】

逆に、選択されたスレッドと一致しないエントリは、次サイクルの実行エントリ選択回路で選択される可能性がないと決定する（Ｓ２６）。

【００８５】

又、命令がデコードされて、リザベーションステーションに新しくエントリに登録するサイクルにおいて、エントリ生成回路５０，６０，７０から登録するエントリのスレッドが、スレッド選択回路３０で選択されたスレッドと一致するかを判定する（Ｓ２０）。一致しないと、ステップＳ２６に進む。

【００８６】

一方、スレッドが一致したエントリについては、実行する準備のできているかを判定し、実行準備ができていれば、そのエントリが、次サイクルでの実行エントリ選択回路５６，６６，７６で選択される可能性のあるエントリに決定する（Ｓ２４）。例えば、エント

リにフラグを付与する。

【００８７】

逆に、選択されたスレッドと一致しないエントリは、次サイクルの実行エントリ選択回路で選択される可能性がないと決定する（Ｓ２６）。

【００８８】

スレッド選択回路３０のスレッド選択例を、図８で説明する。あるスレッドの命令が一定期間完了していないことを検出したときから、ハング防止モードとなり、スレッド選択回路３０が動作する。

【００８９】

ハング防止モードに移行すると、最初はスレッド切り替え時間が１サイクルモードとなり、スレッド選択回路３０で１サイクルごとに異なるスレッドを選択する。

【００９０】

タイマーカウンタで、１００サイクルカウントしたときに、ハング防止モードが続いている場合には、スレッド切り替え時間が２サイクルモードとなり、スレッド選択回路３０で２サイクルごとに異なるスレッドを選択する。

【００９１】

タイマーカウンタで３００サイクルカウントしたときに、ハング防止モードが続いている場合には、スレッド切り替え時間が４サイクルモードとなり、スレッド選択回路３０で４サイクルごとに異なるスレッドを選択する。

【００９２】

タイマーカウンタで６００サイクルカウントしたときに、ハング防止モードが続いている場合には、スレッド切り替え時間が１６サイクルモードとなり、スレッド選択回路で１６サイクルごとに異なるスレッドを選択する。

【００９３】

このようにスレッド選択回路３０で選択するスレッドは、時間の経過によって、ある一定期間で連続して同じスレッドを選択して動作することがいくつものパターンで変更することが可能である。

【００９４】

又、命令の完了ができていなかった命令が完了すると、ハング防止モードから通常走行状態に戻ると、スレッド選択回路３０もリセットされ、もう一度ハング防止モードになった場合には、最初の状態（図８の場合は、１サイクルごとに異なるスレッドを選択）から選択される。

【００９５】

このハング防止モードが動作するのは、同時マルチスレッドで動作するスレッドが２つ以上のときであり、単一のスレッドが動作している場合には、スレッドを選択する必要がないために、ハング防止モードになることがない。また、ハング防止モードになったときに、動作していないスレッドがある場合には、動作していないスレッドには、スレッド選択回路で選択することがないように動作することが可能である。

【００９６】

図９は、このスレッド選択により、図１７の状態から、ハング防止モードが起動したときのリザベーションステーションの制御状態の図である。

【００９７】

時刻Ｔ１で、ハング防止モードが起動する。時刻Ｔ２では、時刻Ｔ１で、スレッド選択回路３０により、スレッド０が選択されたために、スレッド０のエントリを実行する。時刻Ｔ３では、時刻Ｔ２で、スレッド選択回路３０によりスレッド１が選択されるが、先行命令と結果レジスタに格納するタイミングが同じになるために実行不可能である。

【００９８】

時刻Ｔ４では、時刻Ｔ３で、スレッド選択回路３０によりスレッド０が選択されたために、スレッド０のエントリを実行する。時刻Ｔ５では、時刻Ｔ４で、スレッド選択回路３０によりスレッド１が選択されたために、今まで実行できなかったスレッド１のエントリ

を実行することが可能となる。時刻Ｔ６では、時刻Ｔ５で、スレッド選択回路３０によりスレッド０が選択されたために、スレッド０のエントリを実行する。

【００９９】

この後、スレッド１のエントリの命令を完了すると、リザベーションステーションの制御は、ハング防止モードから通常走行状態に戻る。

【０１００】

上述の説明は、浮動小数点リザベーションステーションを例に説明した。固定小数点リザベーションステーションでは、図１０に示すような、エントリの実行制約がある。即ち、固定小数点用のリザベーションステーションは、実行されたエントリは、機能の実行が終了するまで後続のエントリに対して、機能の実行を行わないように制御される。

【０１０１】

リザベーションステーションから実行されるエントリが、実行に２サイクル必要なエントリ（先行命令）であり、後続の命令も実行に２サイクル必要なエントリの場合は、図１０に示すように、Ｔ１で、先行命令がリザベーションステーションから実行される。Ｔ２のタイミングで、後続命令を実行した場合、先行命令と実行（Ｘ）が重なってしまうために、実行不可能となり、Ｔ３のタイミングでは、実行可能となる。

【０１０２】

従って、浮動小数点のエントリ実行制御に、置き換えれば、図９と同様のスレッドのエントリの実行が可能となる。

【０１０３】

このように、複数のスレッドが同時マルチスレッドで動作しているときに、機能の実行の結果レジスタに格納する又は機能の実行のタイミングによって、リザベーションステーションから実行することができなくなったエントリが、ハング防止モードで、リザベーションステーションから実行するエントリのスレッドを選択することで、実行することが可能となり、ハング状態を防止できる。

【０１０４】

次に、このように切り替えサイクルを変更する利点を説明する。図１１は、同時マルチスレッドで動作しているときに、命令フェッチしたデータを得る時間が長時間必要であったために、リザベーションステーションのスレッド１のエントリに、命令デコーダから命令がデコードされないことが原因で、ハング防止モードになった場合の例を示す。この例は、スレッド０には、実行に２サイクル必要なエントリが連続して、固定小数点用のリザベーションステーションに、命令デコーダからデコードされて実行されているときに、スレッド１に実行に２サイクル必要なエントリが作成されたときの状況である。

【０１０５】

図１１に示すように、時刻Ｔ１で、スレッド選択回路３０は、１サイクルモードで、スレッド０を選択する。時刻Ｔ２で、スレッド選択回路３０で、選択されたスレッド０のエントリが実行可能となる。このときに、スレッド選択回路３０は、１サイクルモードのためにスレッド１を選択する。

【０１０６】

時刻Ｔ３で、実行に必要なオペランドデータが揃ったために実行可能となったエントリが実行しようとしても、先行命令の実行と重なるために実行することができない。このときに、スレッド選択回路は、１サイクルモードのために、スレッド０を選択する。時刻Ｔ４で、スレッド０が実行され、スレッド選択回路３０でスレッド１を選択するが、時刻Ｔ５で、スレッド１を実行することができない。

【０１０７】

このときに、スレッド切り替え時間により、１サイクルモードから２サイクルモードに変更する。そして、時刻Ｔ７で、スレッド選択回路３０で、スレッド１が選択され、時刻Ｔ８で、スレッド１を実行することが可能となる。

【０１０８】

図１１では、ハング状態になっていないスレッド０の命令の実行に必要な時間が２サイ

クルのために、スレッド１のエントリは、２サイクルモードになったときに実行可能となった。しかし、スレッド０の命令が、実行に必要な時間が４サイクルの場合には、４サイクルモードにならないと、スレッド１のエントリは実行可能とならない。

【０１０９】

このように、リザベーションステーションが、直接の原因でなく、ハング防止モードになった場合で、機能の実行が後続命令に対して、ブロッキング処理を行う実行制御については、ハング防止モードのスレッド選択方法である同一のスレッドを連続して選択する時間を、機能の実行に必要な時間の最も長い時間より長い時間を設定可能となるようにすることが望ましい。

【０１１０】

このため、スレッド選択回路は、時間の経過によって選択するスレッドを変更することが可能とし、同じスレッドを連続して選択する時間により、スレッドを切り替える制御を行う。

【０１１１】

また、時間によって、同じスレッドを連続して選択する時間も変更することが可能である。

【０１１２】

スレッド選択回路で選択したスレッドのエントリのみが実行可能となるために、通常走行時より性能が下がることが予想され、スレッド選択回路で、同じスレッドを連続して選択する時間が長い程、性能は低下するために、段階的に同じスレッドを選択する時間を長くしていくことで、大幅な性能低下が発生するのを遅らせることができる。

【０１１３】

（スレッド選択回路）

次に、前述のスレッドを選択するスレッド選択回路３０を説明する。前述のスレッド選択回路３０は、論理回路で構成される。図１２は、図６のスレッド決定回路３８の回路図、図１３は、スレッドＩＤ生成回路３６の回路図、図１４は、スレッド切り替え時間選択回路３４の回路図である。

【０１１４】

図１２において、スレッド決定回路３８は、ＡＮＤゲート３８０、ＯＲゲート３８２、４つのＡＮＤゲート３８４－１～３８４－４とからなる。この図１１において、+WARNING_TO_HUNG_MODE_VALID信号は、ハング防止モードであることを示す信号を１回ラッチした信号である。+MULTI_THREAD_MODE信号は、マルチスレッド状態で走行していることを示す信号である。ＡＮＤゲート３８０は、ハング防止モードであり、且つマルチスレッド状態である時に、開く。即ち、ハング防止モードでない、又はマルチスレッド状態でない（シングルスレッド状態である）場合には、動作しない。

【０１１５】

又、+HUNG_THREAD_ID_1T_MODE信号は、１サイクルごとに、選択するスレッドを変更することを示す信号であり、+NEXT_1T_MODE_THREAD_ID信号が、選択されるスレッドを示す。この選択スレッドを示す信号は、図１２から得られる。

【０１１６】

この組み合わせが、1サイクル、２サイクル、４サイクル、１６サイクルの４つ備えられていて、４つＡＮＤゲート３８４－１～３８４－４から１つ選択されたスレッドが、ＯＲゲート３８２を介し、ＡＮＤゲート３８０に入力し、次サイクルのハング防止スレッドＩＤとなる。

【０１１７】

次に、図１３のスレッドＩＤ生成回路は、1サイクル、２サイクル、４サイクル、１６サイクルのスレッドを決める回路である。１サイクルモードのときは、毎サイクルスレッドが変更するために、ハング防止スレッドＩＤを示す+HUNG_MODE_THREAD_IDの極性を、反転回路３６０で、反転する。

【０１１８】

２サイクルモードの回路３６２，３６４，３６６は、スレッドを変更してから２サイクル経過したことを示す+2T_MODE_THREAD_ID_CHANGEがオンになっていれば、ハング防止スレッドＩＤを示す+HUNG_MODE_THREAD_IDの極性を、ＡＮＤゲート３６６で、反転する。又、+2T_MODE_THREAD_ID_CHANGEがオフになっていれば、ハング防止スレッドＩＤを示す+HUNG_MODE_THREAD_IDを、ＡＮＤゲート３６４から出力する。

【０１１９】

ＯＲゲート３６２の出力は、+2T_MODE_THREAD_ID_CHANGEがオンになっていれば、スレッドを変更して、オフであれば、スレッドは変更しない信号を出力する。

【０１２０】

４サイクルと１６サイクルも、同様の２つのＡＮＤゲート３７０，３７２と３７６と３７８と、ＯＲゲート３６８，３７４と組み合わせ回路により、４サイクル経過したことを示す+4T_MODE_THREAD_ID_CHANGE、１６サイクル経過したことを示す+16T_MODE_THREAD_ID_CHANGEがオンになっていれば、スレッドを変更する。

【０１２１】

更に、図１４は、スレッド切り替え時間の変更回路である。ＡＮＤゲート３４２に入力するハング防止モードであることを示す、+SET_WARNING_TO_HUNG_MODEがオンであり、この信号を1回ラッチしたことを示す、+WARNING_TO_HUNG_MODE_VALIDがオフのときに、ＡＮＤゲート３４２、ＯＲゲート３４０から、１サイクルモードを起動する。（ハング防止モードのスタートとなる。）

１サイクルモードが開始すると、２サイクルモードに切り替える時間を示す、+1T_MODE_TIMER_CHANGE信号がオフであり、1サイクルモードであることを示す、+HUNG_THREAD_ID_1T_MODEがオンのときには、ＡＮＤゲート３５２から、ＡＮＤゲート３４４より、１サイクルモードを続ける。

【０１２２】

１サイクルモード中に、切り替え時間がきたら、+1T_MODE_TIMER_CHANGE信号がオンとなり、ＡＮＤゲート３５６、ＯＲゲート３５４、ＡＮＤゲート３４６を介し、２サイクルモードに変更する。

【０１２３】

同様に、２サイクルから４サイクル、４サイクルから１６サイクルへのモード変更についても同様の回路である。即ち、ＡＮＤゲート３５３，３５５とＯＲゲート３５１、ＡＮＤゲート３４８、ＡＮＤゲート３５９，３４９とＯＲゲート３５７、ＡＮＤゲート３５０の構成である。

【０１２４】

このように、スレッド切り替え時間が有効なときは、ハング防止中のみとするので、ハング防止モードであること示す信号を１回ラッチしたことを示す信号とＡＮＤをとる。この信号がスレッド切り替え信号となり、１回ラッチした信号が図１２につながって、スレッドを選択することが可能となる。

【０１２５】

（他の実施の形態）

前述の実施の形態では、２つのスレッド（スレッド０と１）が同時に動作する同時マルチスレッド方式で説明したが、３つ以上のスレッドが同時に動作するものにも適用できる。

【０１２６】

以上、本発明を実施の形態により説明したが、本発明の趣旨の範囲内において、本発明は、種々の変形が可能であり、本発明の範囲からこれらを排除するものではない。

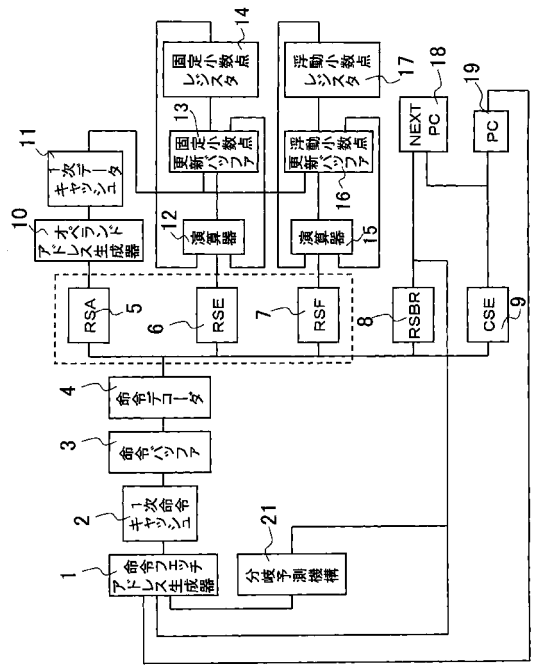
【産業上の利用可能性】

【０１２７】

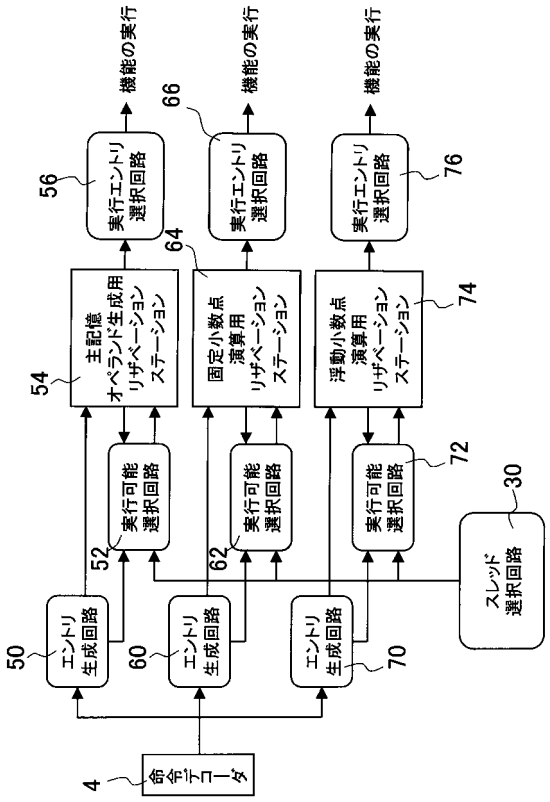
リザベーションステーションから実行可能な命令に対応する命令識別子を保持するエントリを選択する同時マルチスレッドで動作しているときに、命令が、一定期間完了しない状態を検出し、選択部が強制的に交互にスレッド選択制御するため、リザベーションステ

ーションから実行できないエントリが存在する状態の場合には、実行され続けている１のスレッドの実行を停止することで、実行できない他のスレッドのエントリを実行可能な状態にすることができる。又、このようにしても、交互にスレッド選択するので、１のスレッドも選択でき、マルチスレッドの性能低下を防止できる。更に、エントリにスレッド識別子を付加しているので、マルチスレッドのリザベーションステーションからエントリを容易に交互に選択できる。

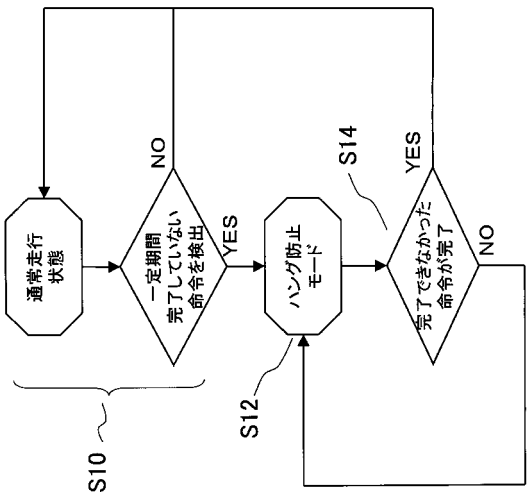
【図１】



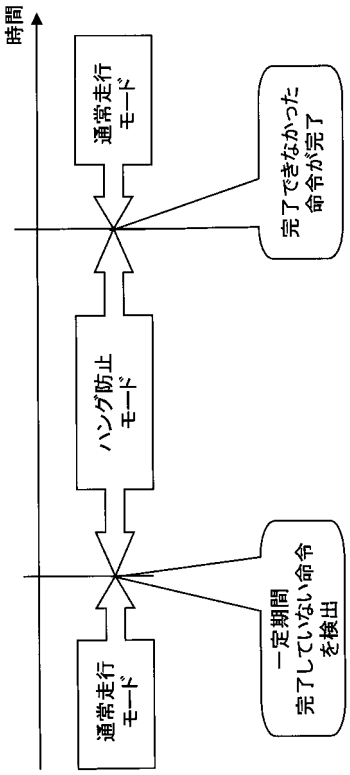
【図２】



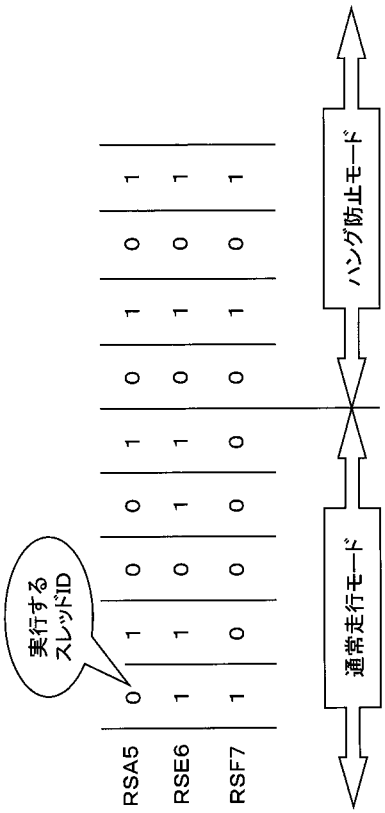
【図３】


通常走行状態
一定期間完了していない命令を検出
NO
YES
ハング防止モード
完了できなかった命令が完了
YES
NO
S10
S12
S14

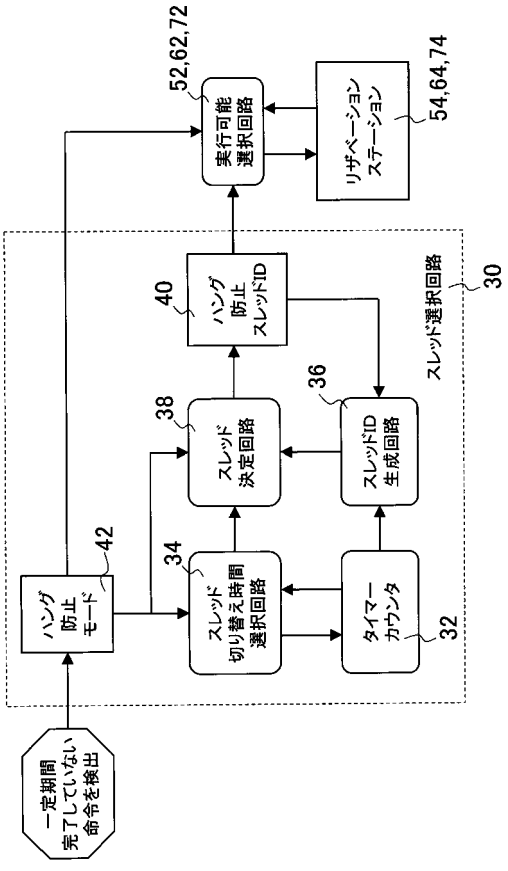
【図４】


時間
通常走行モード
ハング防止モード
通常走行モード
一定期間完了していない命令を検出
完了できなかった命令が完了

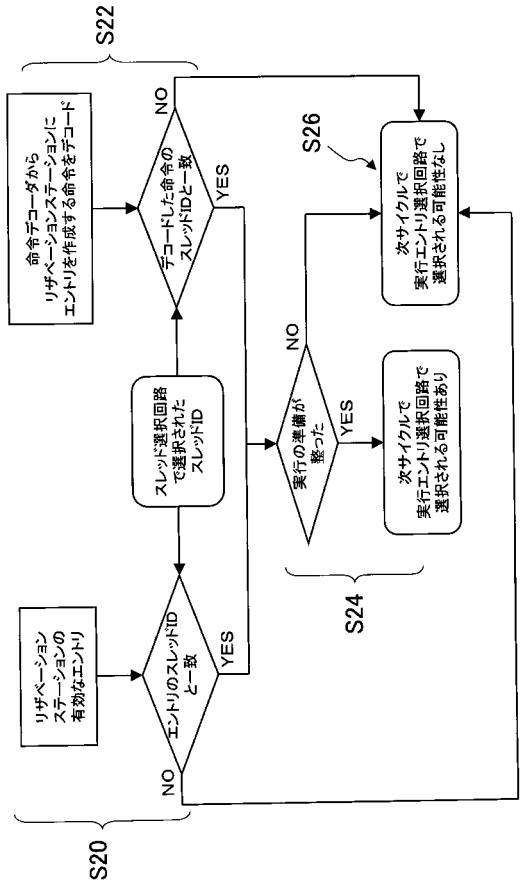
【図５】


実行するスレッドID
RSA5
RSE6
RSF7
ハング防止モード
通常走行モード

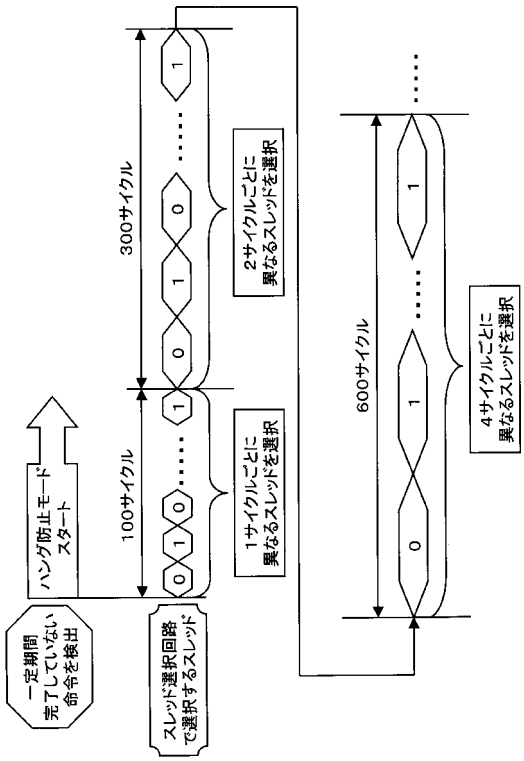
【図６】


52,62,72
実行可能選択回路
リザベーションステーション
54,64,74
スレッド選択回路
30
ハング防止スレッドID
40
スレッド決定回路
38
スレッドID生成回路
36
スレッド切り替え時間選択回路
34
タイマーカウンタ
32
ハング防止モード
42
一定期間完了していない命令を検出

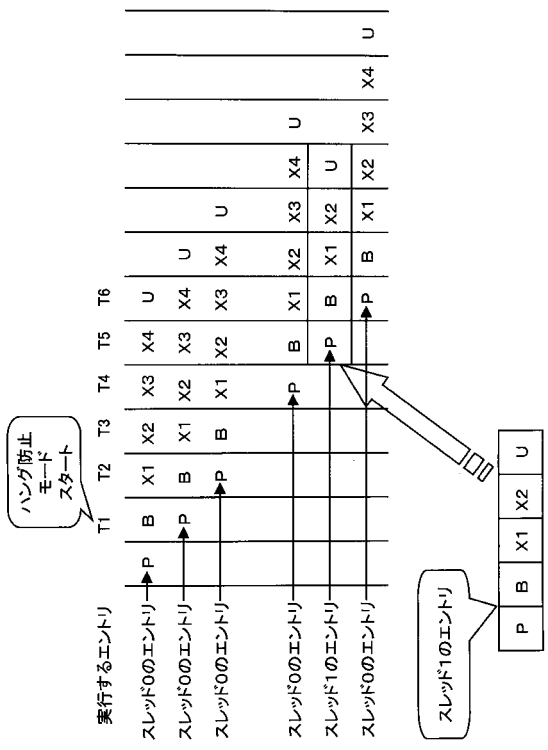
【図７】


S22
命令デコーダから
リザベーションステーションに
エントリを作成する命令をデコード
デコードした命令の
スレッドIDと一致
NO
YES
S26
次サイクルで
実行エントリ選択回路で
選択される可能性なし
スレッド選択回路
で選択された
スレッドID
実行の準備が
整った
NO
YES
次サイクルで
実行エントリ選択回路で
選択される可能性あり
S24
リザベーション
ステーションの
有効なエントリ
エントリのスレッドID
と一致
YES
NO
S20

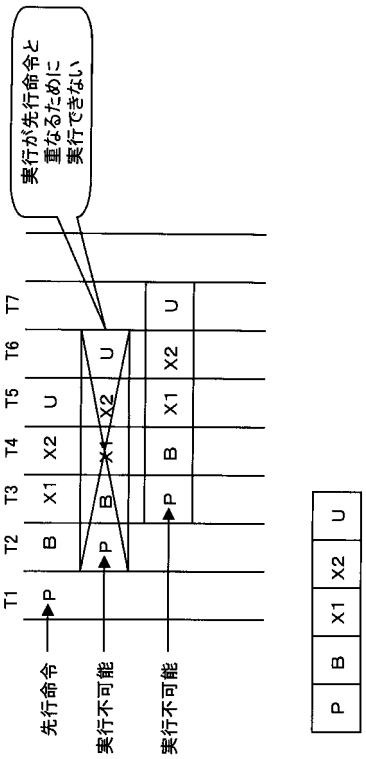
【図８】


300サイクル
100サイクル
600サイクル
ハング防止モード
スタート
一定期間
完了していない
命令を検出
スレッド選択回路
で選択するスレッド
1サイクルごとに
異なるスレッドを選択
2サイクルごとに
異なるスレッドを選択
4サイクルごとに
異なるスレッドを選択

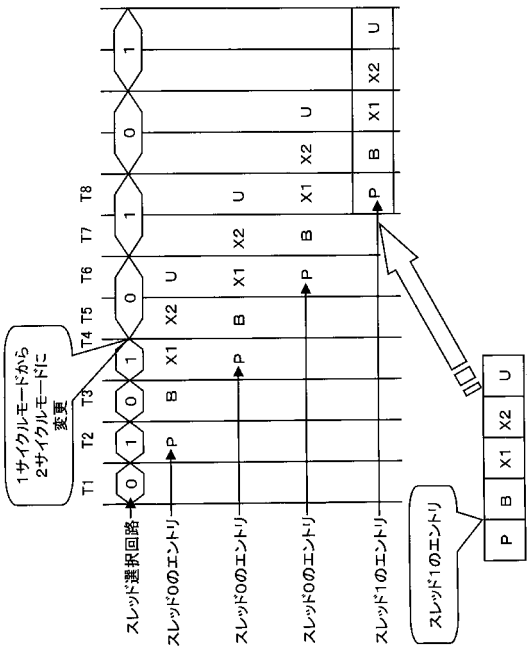
【図９】


ハング防止
モード
スタート
実行するエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド1のエントリ
スレッド0のエントリ
スレッド1のエントリ
T1 T2 T3 T4 T5 T6
P B X1 X2 X3 X4 U

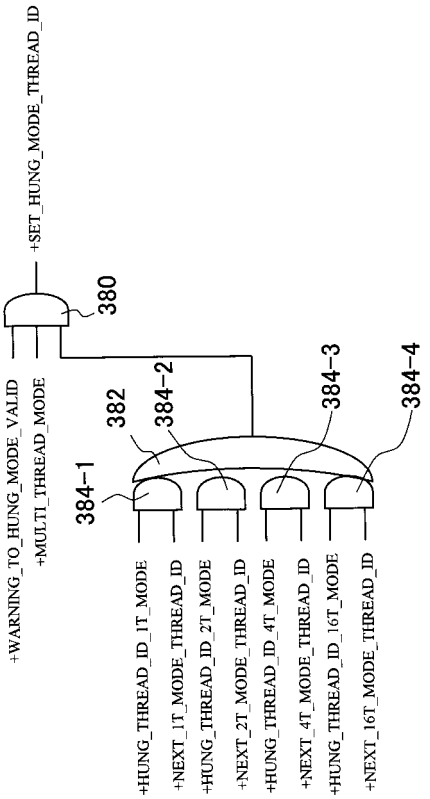
【図１０】


実行が先行命令と
重なるために
実行できない
先行命令
実行不可能
実行不可能
T1 T2 T3 T4 T5 T6 T7
P B X1 X2 U

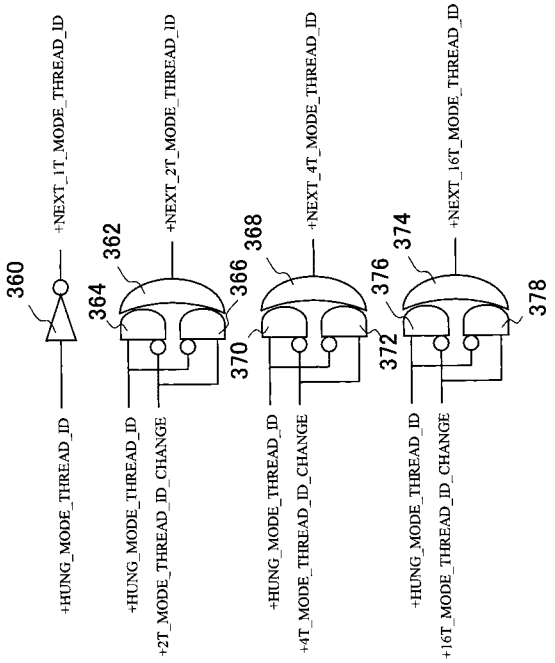
【図１１】



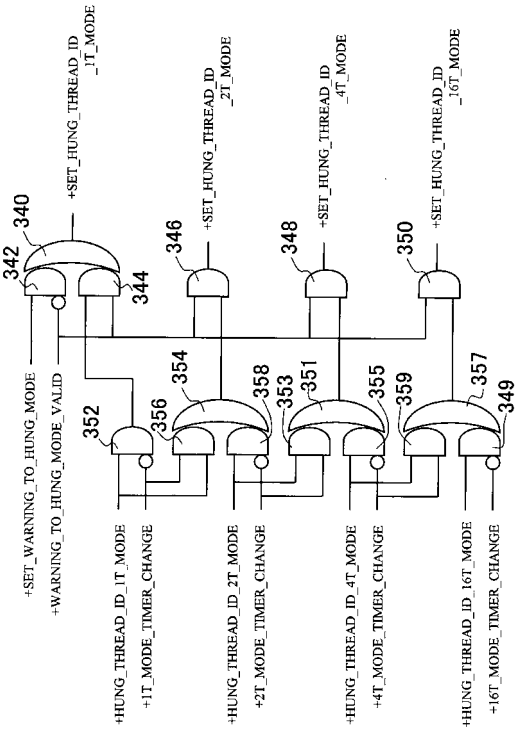
【図１２】



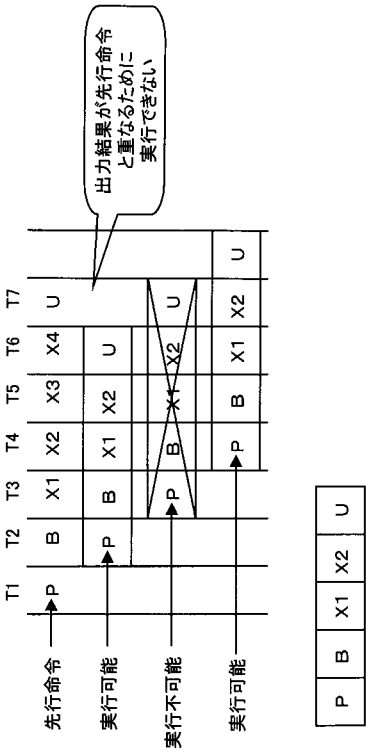
【図１３】



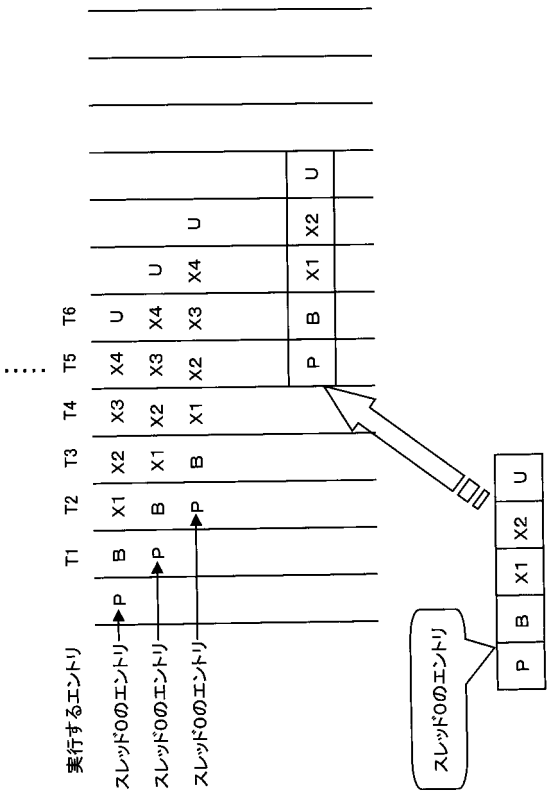
【図１４】



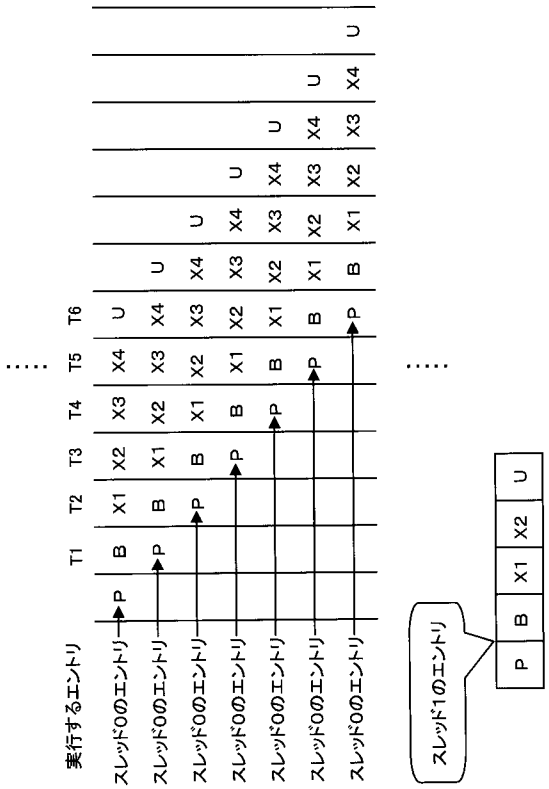
【図１５】


出力結果が先行命令と重なるために実行できない
T1
T2
T3
T4
T5
T6
T7
先行命令
実行可能
実行不可能
実行可能
P
B
X1
X2
X3
X4
U

【図１６】


実行するエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
T1
T2
T3
T4
T5
T6
P
B
X1
X2
X3
X4
U

【図１７】


実行するエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド0のエントリ
スレッド1のエントリ
T1
T2
T3
T4
T5
T6
P
B
X1
X2
X3
X4
U

フロントページの続き

審査官　林　毅

(56)参考文献　国際公開第２００６／１１４８７４（WO，Ａ１）
特許第３７１４５９８（ＪＰ，Ｂ２）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　9/38