(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6947723号
(P6947723)
(45)発行日　令和3年10月13日(2021.10.13)　(24)登録日　令和3年9月21日(2021.9.21)

(51)Int.Cl.　　F I
G06F 9/48 (2006.01)　　G06F 9/48 300B
G06F 9/52 (2006.01)　　G06F 9/52 120B
　　G06F 9/52 120A

請求項の数 13　(全 29 頁)

(21)出願番号　特願2018-518707(P2018-518707)
(86)(22)出願日　平成28年9月14日(2016.9.14)
(65)公表番号　特表2018-534676(P2018-534676A)
(43)公表日　平成30年11月22日(2018.11.22)
(86)国際出願番号　PCT/US2016/051708
(87)国際公開番号　WO2017/065913
(87)国際公開日　平成29年4月20日(2017.4.20)
審査請求日　令和1年8月21日(2019.8.21)
(31)優先権主張番号　14/884,979
(32)優先日　平成27年10月16日(2015.10.16)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　507364838
クアルコム，インコーポレイテッド
アメリカ合衆国　カリフォルニア　９２１２１　サン　ディエゴ　モアハウス　ドライブ　５７７５
(74)代理人　100108453
弁理士　村山　靖彦
(74)代理人　100163522
弁理士　黒田　晋平
(72)発明者　アルン・ラマン
アメリカ合衆国・カリフォルニア・９２１２１－１７１４・サン・ディエゴ・モアハウス・ドライヴ・５７７５

最終頁に続く

(54)【発明の名称】競合の存在下での効率的なタスクスケジューリングのための方法

(57)【特許請求の範囲】
【請求項１】
コンピューティングデバイス上のタスクグラフにおけるタスクスケジューリングの方法であって、前記方法が、
タスク間の関係を管理するデータ構造を参照することにより、第1のタスクおよび第2のタスクが、タスク競合エッジによって関連付けられていることを決定するステップであって、前記データ構造が、前記第1のタスクおよび前記第2のタスクの実行が重複してはならないことを示す前記タスク競合エッジのインジケータを含む、ステップと、
前記第1のタスクおよび前記第2のタスクが前記タスク競合エッジによって関連付けられていると決定したことに応答して、前記第2のタスクが、前記第1のタスクおよび前記第2のタスクの実行に要求されるリソースを獲得できるか否かを決定するステップと、
前記第2のタスクが前記リソースの獲得に失敗するとの決定に応答して、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるステップであって、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジは、前記第1のタスクが前記リソースを放棄するまで、前記第2のタスクが前記リソースを利用することができないことを示し、前記動的タスク依存エッジの割り当てが、前記動的タスク依存エッジのインジケータを前記データ構造に追加することにより行われ、前記割り当てるステップが、
前記第1のタスクの実行が完了したかどうかを決定するステップと、
前記第1のタスクの前記実行が完了していないと決定したことに応答して、前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるステップを含む

、前記動的タスク依存エッジを割り当てるステップと
を含む、方法。
【請求項２】
前記第2のタスクが前記リソースの獲得に失敗すると決定したことに応答して、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるステップが、
前記第1のタスクをロックするステップと、
前記第1のタスクをロックした後、前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるステップと、
前記第1のタスクをアンロックするステップとを含む、請求項1に記載の方法。
【請求項３】
前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるステップが、
前記タスク競合エッジのインジケータをデータ構造から除去するステップをさらに含む、請求項1に記載の方法。
【請求項４】
前記リソースを前記第1のタスクによって獲得するステップであって、
異なるタスクに対しても適用可能なリソースの所定の処理順序で、前記第1のタスクの実行に要求されるリソースが処理されることを要求するステップを含む、獲得するステップと、
前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てることに応答してアイドル状態に入るように、前記第2のタスクを実行するために割り当てられたプロセッサコアにシグナリングするステップと、
前記第1のタスクが完了したとの信号を受信するステップと、
前記アイドル状態からウェイクアップするように、前記第2のタスクを実行するために割り当てられた前記プロセッサコアにシグナリングするステップとをさらに含む、請求項1に記載の方法。
【請求項５】
前記第2のタスクに対してすでに割り当てた動的タスク依存エッジの数が、所定の数を超えているか否かを決定するステップをさらに含み、
前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるステップが、
前記第2のタスクが前記リソースの獲得に失敗したと決定し、かつ、前記第2のタスクに対してすでに割り当てた動的タスク依存エッジの数が、所定の数を超えていないと決定したことに応答して、前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるステップを含む、
請求項1に記載の方法。
【請求項６】
前記所定の数が、実行タスクに対する電力消費、タスク実行レイテンシ、および前記第2のタスクの緊急性のうちの少なくとも1つに基づいて決定される、請求項5に記載の方法。
【請求項７】
タスクグラフにおけるタスクスケジューリングのために構成されたコンピューティングデバイスであって、前記コンピューティングデバイスが、
タスク間の関係を管理するデータ構造を参照することにより、第1のタスクおよび第2のタスクがタスク競合エッジによって関連付けられていることを決定するための手段であって、前記データ構造が、前記第1のタスクおよび前記第2のタスクの実行が重複してはならないことを示す前記タスク競合エッジのインジケータを含む、手段と、
前記第1のタスクおよび前記第2のタスクが前記タスク競合エッジによって関連付けられていると決定したことに応答して、前記第2のタスクが、前記第1のタスクおよび前記第2のタスクの実行に要求されるリソースを獲得できるか否かを決定するための手段と、

前記第2のタスクが前記リソースの獲得に失敗するとの決定に応答して、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるための手段であって、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジは、前記第1のタスクが前記リソースを放棄するまで、前記第2のタスクが前記リソースを利用することができないことを示し、前記動的タスク依存エッジの割り当てが、前記動的タスク依存エッジのインジケータを前記データ構造に追加することにより行われ、前記割り当てるための手段が、

前記第1のタスクの実行が完了したかどうかを決定するための手段と、

前記第1のタスクの前記実行が完了していないと決定したことに応答して、前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるための手段とを含む、前記動的タスク依存エッジを割り当てるための手段と

を含む、コンピューティングデバイス。

【請求項８】

前記第2のタスクが前記リソースの獲得に失敗すると決定したことに応答して、前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるための手段が、

前記第1のタスクをロックするための手段と、

前記第1のタスクをロックした後、前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるための手段と、

前記第1のタスクをアンロックするための手段とを含む、請求項7に記載のコンピューティングデバイス。

【請求項９】

前記第1のタスクから前記第2のタスクへの動的タスク依存エッジを割り当てるための手段が、

前記タスク競合エッジのインジケータをデータ構造から除去するための手段をさらに含む、請求項7に記載のコンピューティングデバイス。

【請求項１０】

前記リソースを前記第1のタスクによって獲得するための手段であって、

異なるタスクに対しても適用可能なリソースの所定の処理順序で、前記第1のタスクの実行に要求されるリソースが処理されることを要求するための手段を含む、獲得するための手段と、

前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てることに応答してアイドル状態に入るように、前記第2のタスクを実行するために割り当てられたプロセッサコアにシグナリングするための手段と、

前記第1のタスクが完了したとの信号を受信するための手段と、

前記アイドル状態からウェイクアップするように、前記第2のタスクを実行するために割り当てられた前記プロセッサコアにシグナリングするための手段とをさらに含む、請求項7に記載のコンピューティングデバイス。

【請求項１１】

前記第2のタスクに対してすでに割り当てた動的タスク依存エッジの数が、所定の数を超えているか否かを決定するための手段をさらに含み、

前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるための手段が、

前記第2のタスクが前記リソースの獲得に失敗したと決定し、かつ、前記第2のタスクに対してすでに割り当てた動的タスク依存エッジの数が、所定の数を超えていないと決定したことに応答して前記第1のタスクから前記第2のタスクへの前記動的タスク依存エッジを割り当てるための手段を含む、請求項7に記載のコンピューティングデバイス。

【請求項１２】

前記所定の数が、実行タスクに対する電力消費、タスク実行レイテンシ、および前記第2のタスクの緊急性のうちの少なくとも1つに基づいて決定される、請求項11に記載のコンピューティングデバイス。

【請求項１３】

請求項1から6のうちのいずれか一項に記載の方法を実行するための命令を含むコンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、競合の存在下での効率的なタスクスケジューリングのための方法に関する。

【背景技術】

【０００２】

反応が早くて高性能なアプリケーションが、ユーザエクスペリエンスを改善し得る。タスク並列プログラミングモデルは、高性能を提示してユーザへの反応が早いアプリケーションを開発するために広く使用されている。アプリケーション開発のこのモデルでは、アプリケーションにおける計算は、仕事の小さいチャンクであるタスクに分割され、いくつかのタスクは、いくつかの他のタスク終了後に実行するように規定され、これらはタスク依存性と呼ばれる。タスク間の競合を回避して相互依存性をサポートするために並列タスクの実行を管理することが、プログラム上の課題を提示する。

【発明の概要】

【課題を解決するための手段】

【０００３】

様々な実施形態の方法および装置は、コンピューティングデバイスにおいてタスク競合エッジの存在下でのタスクスケジューリングのための回路および方法を提供する。様々な実施形態は、第1のタスクおよび第2のタスクがタスク競合エッジによって関連付けられているかどうかを決定することと、第1のタスクおよび第2のタスクがタスク競合エッジによって関連付けられていると決定したことに応答して、第2のタスクが第1のタスクおよび第2のタスクの実行に要求されるリソースを獲得しているかどうかを決定することと、第2のタスクがリソースの獲得に失敗したと決定したことに応答して、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることとを含み得る。

【０００４】

いくつかの実施形態では、第2のタスクがリソースの獲得に失敗したと決定したことに応答して、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることは、第1のタスクの実行が完了したかどうかを決定することと、第1のタスクの実行が完了していないと決定したことに応答して、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることとを含み得る。

【０００５】

いくつかの実施形態では、第2のタスクがリソースの獲得に失敗したと決定したことに応答して、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることは、第1のタスクをロックすることと、第1のタスクをロックした後、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることと、第1のタスクをアンロックすることとを含み得る。

【０００６】

いくつかの実施形態では、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることは、データ構造からタスク競合エッジのインジケータを除去することと、データ構造に動的タスク依存エッジのインジケータを追加することとを含み得る。

【０００７】

いくつかの実施形態は、リソースを獲得することであって、リソースが、第2のタスクまたは第3のタスクの実行に要求される少なくとも1つ他のリソースに関連する順序で処理されることを要求することによって、デッドロックおよび周期的動的タスク依存性が回避され得る、獲得することと、第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てることに応答してアイドル状態に入るように、第2のタスクを実行するために割り当てられたプロセッサコアにシグナリングすることと、第1のタスクが完了したとの信号を受信することと、アイドル状態からウェイクアップするように、第2のタスクを実行す

るために割り当てられたプロセッサコアにシグナリングすることとを行う第1のタスクをさらに含み得る。

【０００８】

いくつかの実施形態は、競合ポリシーが満足されるかどうかを決定することと、第2のタスクがリソースの獲得に失敗したと決定したことに応答して第1のタスクから第2のタスクへの動的タスク依存エッジを割り当てるのではなく、競合ポリシーが満足されると決定したことに応答して第2のタスクがレディタスクであるかどうかを決定することとをさらに含み得る。

【０００９】

いくつかの実施形態では、競合ポリシーが満足されるかどうかを決定することはさらに、動的タスク依存エッジのしきいの量が割当てセッション内で第2のタスクに割り当てられるかどうかを決定することを含み得る。

【００１０】

いくつかの実施形態では、競合ポリシーは、割当てセッション内で第2のタスクに割り当てられた動的タスク依存エッジのしきいの量を含んでもよく、動的タスク依存エッジのしきいの量は、実行タスクに対する電力消費、タスク実行レイテンシ、および第2のタスクの緊急性のうちの少なくとも1つに基づいて決定される。

【００１１】

様々な実施形態は、タスク競合エッジの存在下でのタスクスケジューリングのために構成されたコンピューティングデバイスを含み得る。コンピューティングデバイスは、第1のタスクおよび第2のタスクの実行に要求される、メモリデバイスを含むリソースと、第1のプロセッサコア、第2のプロセッサコアおよび第3のプロセッサコアを含む互いに通信可能に接続された複数のプロセッサコアとを含んでよく、第2のプロセッサコアおよび第3のプロセッサコアがリソースに通信可能に接続され、第1のプロセッサコアが、上述の実施形態方法のうちの1つまたは複数の動作を実行するようにプロセッサ実行可能命令を用いて構成される。

【００１２】

様々な実施形態は、上述の実施形態方法のうちの1つまたは複数の機能を実行するための手段を有する、タスク競合エッジの存在下でのタスクスケジューリングのために構成されたコンピューティングデバイスを含み得る。

【００１３】

様々な実施形態は、上述の実施形態方法のうちの1つまたは複数の実施形態方法の動作をコンピューティングデバイスのプロセッサに実行させるように構成された、プロセッサ実行可能命令を記憶している、非一時的プロセッサ可読記憶媒体を含んでもよい。

【００１４】

本明細書に組み込まれ、本明細書の一部を構成する添付の図面は、様々な実施形態のうちの例示的な実施形態を示し、上で与えられた一般的な説明および下記の発明を実施するための形態とともに、特許請求の範囲の特徴を説明するのに役立つ。

【図面の簡単な説明】

【００１５】

【図１】一実施形態を実装するのに適したコンピューティングデバイスを示す構成要素ブロック図である。

【図２】一実施形態を実施するのに適した例示的なマルチコアプロセッサを示す構成要素ブロック図である。

【図３Ａ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。

【図３Ｂ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。

【図３Ｃ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。

【図３Ｄ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。
【図４Ａ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。
【図４Ｂ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。
【図４Ｃ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。
【図４Ｄ】一実施形態による、動的タスク依存性を含む例示的なタスクグラフを示す概略図である。
【図５】タスク競合の存在下での効率的なタスクスケジューリングのための一実施形態方法を示すプロセスフロー図である。
【図６】動的タスク依存性を割り当てるための一実施形態方法を示すプロセスフロー図である。
【図７】タスク競合を動的タスク依存性と置き換えるための一実施形態方法を示すプロセスフロー図である。
【図８】タスク競合を動的タスク依存性と置き換えるための一実施形態方法を示すプロセスフロー図である。
【図９】様々な実施形態とともに使用するのに適した例示的なモバイルコンピューティングデバイスを示す構成要素ブロック図である。
【図１０】様々な実施形態とともに使用するのに適した例示的なモバイルコンピューティングデバイスを示す構成要素ブロック図である。
【図１１】様々な実施形態とともに使用するのに適した例示的なサーバを示す構成要素ブロック図である。
【発明を実施するための形態】
【００１６】
様々な実施形態が、添付図面を参照して詳細に説明される。可能な場合はいつでも、同一のまたは同様の部分を指すために、図面全体を通して同一の参照番号が使用される。特定の例および実施態様に対してなされる参照は、説明が目的であり、特許請求の範囲を限定することを意図していない。
【００１７】
「コンピューティングデバイス」および「モバイルコンピューティングデバイス」という用語は、セルラー電話、スマートフォン、パーソナルまたはモバイルマルチメディアプレーヤ、携帯情報端末(PDA)、ラップトップコンピュータ、タブレットコンピュータ、コンバーチブルラップトップ/タブレット(ツーインワンコンピュータ)、スマートブック、ウルトラブック、ネットブック、パームトップコンピュータ、ワイヤレス電子メール受信機、マルチメディアインターネット対応セルラー電話、モバイルゲームコンソール、ワイヤレスゲームコントローラ、ならびに、メモリおよびマルチコアプログラマブルプロセッサを含む同様のパーソナル電子デバイスのうちのいずれか1つまたはすべてを指すために本明細書で互換的に使用される。様々な実施形態は、限られたメモリおよびバッテリーリソースを有するスマートフォンなどのモバイルコンピューティングデバイスに特に有用であるが、本実施形態は一般に、複数のメモリデバイスおよび限られた電力バジェットを実装する任意の電子デバイスにおいて有用であり、プロセッサの電力消費を低減することは、モバイルコンピューティングデバイスのバッテリー動作時間を延ばすことができる。「コンピューティングデバイス」という用語はさらに、パーソナルコンピュータを含む据置型コンピューティングデバイス、デスクトップコンピュータ、オールインワンコンピュータ、ワークステーション、スーパーコンピュータ、メインフレームコンピュータ、組込み型コンピュータ、サーバ、ホームシアターコンピュータ、およびゲームコンソールを指す場合がある。
【００１８】

本実施形態は、タスク依存性およびタスク競合を考慮するスケジューリング技法を使用して並列タスクの効率的な実行を提供することによってデバイス性能を改善するための、方法およびシステムならびにそのような方法を実装するデバイスを含む。方法、システムおよびデバイスは、タスク間のリソース競合を動的タスク依存性と置き換え、動的タスク依存性を追加するまでタスクがリソース競合の影響を受けやすいリソースを所有するときにだけ動的タスク依存性を追加することが発生することを確保し、再スケジューリング前に特定される競合の制限を指定することによって動的タスク依存性の影響を受けやすいタスクの再スケジューリングの頻度を抑制し得る。

【００１９】

タスク依存性は、タスク実行のアプリオリな順序付けを強制する。依存性t1->t2(「->」は依存性を示す)は、タスクt1はタスクt2が実行を開始する前に終了しなければならないが、その逆ではないことを指定する。いくつかのタスクは、「タスク競合」を通して関連付けられ得る。競合t1=t2(「=」はタスク競合を示す)は、タスクt2が実行を開始する前にタスクt1が終了しなければならないかまたはタスクt1が実行を開始する前にタスクt2が終了しなければならないかのいずれかであるが、タスクt1およびt2の実行は重複してはならないことを指定する。言い換えれば、タスク競合によって関連付けられた2つのタスクは、同時ではないいずれかの順序でタスクスケジューラによって実行するためにスケジュールされ得る。そのようなタスク競合は、典型的には、コンピューティングデバイスの制限された利用可能性の任意の物理的または仮想的構成要素を含み得る、1つまたは複数の共有リソースへの排他的アクセスを要求するタスクによって生じる。リソースの例には、処理構成要素、処理時間、ストレージまたはメモリ構成要素、ストレージまたはメモリロケーション、内部または外部通信構成要素、内部または外部通信帯域幅、電力構成要素、電力、入力/出力構成要素、入力/出力信号、などが含まれ得る。リソースの共有は、たとえば、タスクが、非同期メモリアクセスを有する2つのプロセッサ上のデータバッファなどのメモリデバイスに同時に書き込むときに発生する場合がある。タスク依存性とタスク競合の両方が存在する場合のタスクスケジューリングは困難である。

【００２０】

物理的および仮想的リソースへの参照を含む、様々な実施形態の説明における特定のリソースへの参照は、非限定的な例であることが意図される。コンピューティングデバイスの任意のリソースは、本明細書で与えられる説明および例のいずれかの影響を受けやすく、様々なタイプの任意の物理的または仮想的リソースは、例示的な目的のために本明細書で参照されるリソースで代用され得る。

【００２１】

複数のタスクは、複数のタスクによって同時にアクセスされ得ないリソースを共有する場合があるので、タスク間の依存性および競合の関係は、タスクのスケジューリングに影響を及ぼす場合がある。タスク競合は、リソース獲得問題としてモデル化され得る。一例として、タスク競合t1=t2を考察する。この競合は、両タスクがリソースRへの排他的アクセスを要求することとしてモデル化され得る。タスクが複数の競合を有する場合、そのタスクは、それらの競合に関連付けられたすべてのリソースへの排他的アクセスを獲得しなければならない。競合のない従来のタスクスケジューリングでは、タスクは、それの依存性のすべてが満足されたときに実行の準備ができている。

【００２２】

競合のあるタスクスケジューリングでは、タスクスケジューラは、レディタスクがすべてのリソースへの排他的アクセスを獲得し、競合することなく実行し得るかどうかを決定しなければならない。考慮中のタスクと競合する他のタスクは、後で実行するようにスケジュールされなければならない。同じリソースに同時にアクセスすることを試行する複数のタスクに対して、単一のタスクが、リソースにアクセスするかまたはリソースを所有することを許可され、オーナータスクになり得る。リソースの所有権を獲得することは、いくつかの方法で実施され得る。様々な実施形態では、リソースはメモリ内の名前付きロケーションであってよく、メモリ内のバイトはどのタスクもそのリソースを所有していない

ことを示すためにゼロまたはヌルにセットされ、メモリ内のバイトはタスクがそのリソースを所有していることを示すためにそのタスクのアドレス(または「タスクポインタ」)にセットされる。名前付きリソースの所有権を首尾よく獲得することは、リソースのメモリロケーション内に記憶されている値がゼロまたはヌルであることをアトミックに検査すること、およびその値をタスクポインタと交換することを伴う場合がある。リソースのメモリロケーション内に記憶されているゼロまたはヌル以外の値は、リソースが記憶されている値に対応する異なるタスクによってすでに所有されていることを示し得る。他のタスクは、その所有されたリソースにアクセスして再スケジュールされることを防止され得る。オーナータスクによって所有されているリソースへのアクセスを試行するタスクが、所有されているリソースへのアクセスを継続的に試行し、アクセス試行を繰り返すことによって電力および処理リソースを浪費することがないように、再スケジューリングが実施され得る。

【００２３】

第2のタスク(「t2」)との競合関係を有する、初期化された第1のタスク(「t1」)は、競合の影響を受けやすいリソースにアクセスするために最初の試行を行い得る。第2のタスクがまだリソースのオーナータスクではなく、第1のタスクをリソースのオーナータスクにしているときに、第1のタスクは、リソースに首尾よくアクセスし得る。第2のタスクがすでにリソースのオーナータスクであるときは、第1のタスクは、リソースへのアクセスに失敗する。リソースにアクセスすることの試行に失敗したことに応答して、第1のタスクと第2のタスクとの間の競合関係は、タスクt2が終了しなければならず、それによりリソースの所有権をタスクt1が実行を開始する前に放棄することを指定する、動的タスク依存関係t2=>t1(「=>」は動的タスク依存性を示す)によって置き換えられ得る。

【００２４】

第1のタスク、および第1のタスクへの依存性を有する任意のタスクが待機状態で放置されるのを回避するために、動的タスク依存性を追加することは、オーナータスクが動的タスク依存性を追加するまでリソース競合の影響を受けやすいリソースを所有するときだけ、発生し得る。一実施形態では、第2のタスクがオーナータスクであると決定したことに応答して、第2のタスクは、動的タスク依存性t2=>t1を追加するまで、終了することを防止され得る。これは、動的タスク依存性の追加を終了するためにオーナータスクがロックされることを要求し、それにより動的タスク依存性t2=>t1を追加するまで第2のタスクをロックし、次いで第2のタスクが終了し得るように第2のタスクをアンロックすることによって、達成され得る。一実施形態では、第2のタスクがオーナータスクであると決定したことに応答して、第2のタスクの状態が、動的タスク依存性t2=>t1を追加する前に第2のタスクが終了したかどうかを決定するために検査され得る。いずれの実施形態も、依存性がもはや必要でないときに依存性t2=>t1をセットアップして、第1のタスクに、第2のタスクが終了して競合の影響を受けやすいリソースが利用可能になったとの信号を待たせるが、そのような信号を次回に第2のタスクが実行するまで受信しないか、または第2のタスクが再び実行することはない場合にそのような信号をまったく受信しないという問題を回避する。これはまた、第1のタスクが実行しないことに基づいて、第1のタスクに直接的または間接的に依存する複数のタスクによる待機のカスケード効果を回避する。

【００２５】

第1のタスクの、任意の数の他のタスクとの競合の影響を受けやすいリソースへのアクセスが、検査され得る。競合の影響を受けやすいリソースへのアクセスが拒否されるたびに、新しい動的タスク依存性が第1のタスクに対して追加され得る。第1のタスクの、競合の影響を受けやすいリソースへのアクセスが、すべての競合するタスクに対して検査され得る。第1のタスクは、追加された動的依存性に基づいて再スケジュールされ得る。一実施形態では、動的タスク依存性を追加されたタスクを再スケジュールするためのポリシーは、いくつの動的依存性が検査セッション中に追加されるかに基づいて、どれくらいの頻度で第1のタスクが再スケジュールされるかを制御し得る。言い換えれば、1つのタスクに対して新しく追加される動的依存性の数を制限するポリシーは、第1のタスクと競合して

いるすべてのタスクに対して競合の影響を受けやすいリソースへのアクセスのすべてが検査されたとは限らないとしても、制限に達して第1のタスクが再スケジュールされ得るとき、第1のタスクに対する、競合の影響を受けやすいリソースへのアクセスを検査するプロセスを中断してもよい。競合の影響を受けやすいリソースへの残りのアクセスは、第1のタスクを実行する後続の試行時に検査され得る。ポリシーは、異なるタスクに対して、および一定量のタスクレイテンシおよび/またはリソース消費を与えることなど、異なる目的に対してセットされ得る。

【００２６】

動的依存エッジを十分検討せずに単純に追加することで、もともと非周期的なグラフに周期が生じる場合がある。複数のリソースの獲得を複数のタスクが同時に試行するときに、デッドロックが生じる場合がある。たとえば、第2のタスクが、R2に次いでR1を獲得しようと試行している間に同時に、第1のタスクが、第1のリソース(「R1」)および第2のリソース(「R2」)などのリソースをR1およびR2の順序で獲得しようと試行する場合がある。各タスクは他方のタスクによって所有されるリソースの利用可能性を待って、リソースを獲得するため待機するタスクのデッドロックを生じる場合がある。リソース獲得の間のデッドロックと周期的依存性のデッドロックの両方は、いくつかの方法によって回避され得る。様々な実施形態では、タスクに対するリソースは常に、正規順序で、たとえばR1およびR2に対して対応するリソースハンドルのポインタアドレスのソートされた順序、または各リソースに割り当てられた一意のリソース識別子(ID)のソートされた順序で、処理され得る。様々な実施形態では、デッドロック検出およびリカバリ技法が使用され得る。

【００２７】

図1は、様々な実施形態とともに使用するのに適したリモートコンピューティングデバイス50と通信中のコンピューティングデバイス10を含むシステムを示す。コンピューティングデバイス10は、プロセッサ14、メモリ16、通信インターフェース18、およびストレージメモリインターフェース20を有するシステムオンチップ(SoC)12を含み得る。コンピューティングデバイスは、ワイヤードモデムまたはワイヤレスモデムなどの通信構成要素22、ストレージメモリ24、ワイヤレスネットワーク30へのワイヤレス接続32を確立するためのアンテナ26、および/またはインターネット40へのワイヤード接続44に接続するためのネットワークインターフェース28をさらに含んでもよい。プロセッサ14は、様々なハードウェアコア、たとえばいくつかのプロセッサコアのいずれかを含み得る。

【００２８】

「システムオンチップ」(SoC)という用語は、通常、限定はしないが、ハードウェアコア、メモリ、および通信インターフェースを含む相互接続された電子回路のセットを指すために本明細書で使用される。ハードウェアコアは、汎用プロセッサ、中央処理ユニット(CPU)、デジタル信号プロセッサ(DSP)、グラフィックス処理ユニット(GPU)、加速処理ユニット(APU)、補助プロセッサ、シングルコアプロセッサ、およびマルチコアプロセッサなどの様々な異なるタイプのプロセッサを含んでよい。ハードウェアコアは、フィールドプログラマブルゲートアレイ(FPGA)、特定用途向け集積回路(ASIC)、他のプログラマブル論理回路、個別ゲート論理、トランジスタ論理、性能監視ハードウェア、ウォッチドッグハードウェア、および時間基準などの、他のハードウェアおよびハードウェアの組合せをさらに具現化してもよい。集積回路は、集積回路の構成要素がシリコンなどの単体の半導体材料上に存在するように構成されてよい。SoC12は、1つまたは複数のプロセッサ14を含むことができる。コンピューティングデバイス10は、2つ以上のSoC12を含み、それによって、プロセッサ14およびプロセッサコアの数を増加させることができる。コンピューティングデバイス10は、SoC12に関連付けられていないプロセッサ14を含んでもよい。個々のプロセッサ14は、図2を参照しながら以下で説明するようなマルチコアプロセッサであってよい。プロセッサ14は各々、コンピューティングデバイス10の他のプロセッサ14と同じであっても異なっていてもよい特定の目的のために構成されてよい。同じまたは異なる構成のプロセッサ14およびプロセッサコアのうちの1つまたは複数は、一緒にグループ化されてもよい。プロセッサ14またはプロセッサコアのグループは、マルチプロセッサクラス

タと呼ばれてもよい。

【００２９】

SoC12のメモリ16は、プロセッサ14によるアクセスのためにデータおよびプロセッサ実行可能コードを記憶するように構成された、揮発性または不揮発性のメモリとすることができる。コンピューティングデバイス10および/またはSoC12は、様々な目的のために構成された1つまたは複数のメモリ16を含むことができる。一実施形態では、1つまたは複数のメモリ16は、ランダムアクセスメモリ(RAM)もしくはメインメモリ、またはキャッシュメモリなど、揮発性メモリを含んでもよい。これらのメモリ16は、限られた量の、データセンサまたはサブシステムから受信されたデータ、様々な因子に基づいて将来アクセスされることを予期して不揮発性メモリから要求され、不揮発性メモリからメモリ16にロードされるデータおよび/またはプロセッサ実行可能コード命令、ならびに/あるいはプロセッサ14によって生成され、不揮発性メモリには記憶されずに将来迅速にアクセスできるように一時的に記憶される中間処理データおよび/またはプロセッサ実行可能コード命令を、一時的に保持するように構成され得る。

【００３０】

メモリ16は、プロセッサ14のうちの1つまたは複数によるアクセスのために、別のメモリ16またはストレージメモリ24など、別のメモリデバイスからメモリ16にロードされるデータおよびプロセッサ実行可能コードを、少なくとも一時的に記憶するように構成されてもよい。メモリ16にロードされたデータまたはプロセッサ実行可能コードは、プロセッサ14による機能の実行に応答してロードされてもよい。機能の実行に応答してデータまたはプロセッサ実行可能コードをメモリ16にロードすることは、要求されたデータまたはプロセッサ実行可能コードがメモリ16内にないために、不成功またはミスであるメモリ16へのメモリアクセス要求から生じる場合がある。ミスに応答して、要求されたデータまたはプロセッサ実行可能コードを他のメモリデバイス16またはストレージメモリ24からメモリデバイス16にロードするために、別のメモリ16またはストレージメモリ24へのメモリアクセス要求が行われてもよい。機能の実行に応答してデータまたはプロセッサ実行可能コードをメモリ16にロードすることは、別のメモリ16またはストレージメモリ24へのメモリアクセス要求から生じる場合があり、データまたはプロセッサ実行可能コードは、後のアクセスのためにメモリ16にロードされてもよい。

【００３１】

通信インターフェース18、通信構成要素22、アンテナ26、および/またはネットワークインターフェース28は、コンピューティングデバイス10がワイヤレス接続32を介してワイヤレスネットワーク30上でおよび/またはワイヤードネットワーク44上でリモートコンピューティングデバイス50と通信することが可能になるように、調和して動作してもよい。コンピューティングデバイス10が、リモートコンピューティングデバイス50とデータを交換できるようにするためのインターネット40に接続するのを可能にするために、ワイヤレスネットワーク30は、たとえば、ワイヤレス通信用に使用される無線周波数スペクトルを含む様々なワイヤレス通信技術を使用して実装されてもよい。

【００３２】

ストレージメモリインターフェース20およびストレージメモリ24は、コンピューティングデバイス10が不揮発性記憶媒体上にデータとプロセッサ実行可能コードとを記憶することが可能になるように、調和して動作してもよい。ストレージメモリ24は、ストレージメモリ24がプロセッサ14のうちの1つまたは複数によってアクセスするためのデータまたはプロセッサ実行可能コードを記憶してもよいメモリ16の一実施形態とほとんど同じに構成されてもよい。不揮発性であるストレージメモリ24は、コンピューティングデバイス10の電源が切断された後でも情報を保持し得る。電源が再び投入され、コンピューティングデバイス10が再起動すると、コンピューティングデバイス10がストレージメモリ24上に記憶された情報を利用できるようになる。ストレージメモリインターフェース20は、ストレージデバイス24へのアクセスを制御し、プロセッサ14がストレージメモリ24からデータを読み取り、ストレージメモリ24にデータを書き込むことを可能にすることができる。

【００３３】
コンピューティングデバイス10の構成要素の一部または全部は、やはり必要な機能を果たしながら異なって構成されてよく、かつ/または組み合わせられてもよい。その上、コンピューティングデバイス10は、構成要素の各々のうちの1つに限定されなくてよく、各構成要素の複数のインスタンスがコンピューティングデバイス10の様々な構成の中に含まれてよい。
【００３４】
図2は、様々な実施形態を実装するのに適したマルチコアプロセッサ14を示す。マルチコアプロセッサ14は、複数の同種または異種のプロセッサコア200、201、202、203を有し得る。単一のプロセッサ14のプロセッサコア200、201、202、203が同じ目的のために構成され、同じまたは同様の性能特性を有してもよいという点で、プロセッサコア200、201、202、203は同種であってもよい。たとえば、プロセッサ14は汎用プロセッサであってもよく、プロセッサコア200、201、202、203は同種の汎用プロセッサコアであってもよい。代替的に、プロセッサ14はグラフィックス処理ユニットまたはデジタル信号プロセッサであってもよく、プロセッサコア200、201、202、203はそれぞれ、同種のグラフィックスプロセッサコアまたはデジタル信号プロセッサコアであってもよい。参照しやすいように、「プロセッサ」および「プロセッサコア」という用語は、本明細書で互換的に使用される場合がある。
【００３５】
単一のプロセッサ14のプロセッサコア200、201、202、203が異なる目的のために構成され、ならびに/あるいは異なる性能特性を有してもよいという点で、プロセッサコア200、201、202、203は異種であってもよい。そのような異種プロセッサコアの異種性には、異なる命令セットアーキテクチャ、パイプライン、動作周波数などを含めてもよい。そのような異種のプロセッサコアの例は、より低速で低電力のプロセッサコアがより強力で電力を大量に消費するプロセッサコアと結合される場合がある「big.LITTLE」アーキテクチャとして知られているものを含んでもよい。同様の実施形態では、SoC12は、いくつかの均一なまたは不均一なプロセッサ14を含んでもよい。
【００３６】
図2に示す例では、マルチコアプロセッサ14は、4つのプロセッサコア200、201、202、203(すなわち、プロセッサコア0、プロセッサコア1、プロセッサコア2、およびプロセッサコア3)を含む。説明を簡単にするために、本明細書の例では、図2に示す4つのプロセッサコア200、201、202、203を参照する場合がある。しかしながら、図2に示し、本明細書で説明する4つのプロセッサコア200、201、202、203は単に一例として提供され、決して様々な実施形態を4コアプロセッサシステムに限定するものではない。コンピューティングデバイス10、SoC12、またはマルチコアプロセッサ14は、図示され本明細書で説明される4つのプロセッサコア200、201、202、203よりも少数またはより多数のプロセッサコアを個々にまたは組合せで含むことができる。
【００３７】
図3A～図3Dは、様々なタスクグラフ構成300a～300dにおける例示的な1つのタスクグラフを示す。タスク302a～302d、タスク304a～304c、タスク依存エッジ(異なるタスク302a～302dおよび304aの間の矢印によって示される)、タスク競合エッジ306aおよび306b、ならびに動的タスク依存エッジ308a～308cを含むタスクグラフおよびそれの様々なタスクグラフ構成300a～300dは単なる例であり、様々な数のタスクおよびタスク間の様々な関係を用いて構成され得る。
【００３８】
図3Aは、タスク依存エッジによって互いに関連するタスク302a～302dおよび304aを有する例示的なタスクグラフ構成300aを示す。言い換えれば、1つのタスクは別のタスクに従属しており、それにより、従属タスクが従属しているタスクより前に、従属タスクは実行することができない。たとえば、タスク302bはタスク302aに従属しており、したがって、タスク302aが実行を完了する前にタスク302bが実行することはできない。同様にして、タ

スク304aもまたタスク302aに従属しており、タスク302cはタスク302bに従属しており、タスク302dはタスク302cおよびタスク304bに従属している。タスクグラフ構成300aはまた、タスク競合エッジ306aおよび306bによって互いに関連するタスク304a～304cを含む。言い換えれば、互いに競合する2つ以上のタスクが任意の順序で実行し得るという点で、1つのタスクは別のタスクに依存しないが、互いに競合する2つのタスクが同時に実行することはない。タスク304a～304cの間の競合エッジ306aおよび306bは、互いに競合するタスクが、実行するために同じリソースへのアクセスを要求する場合があることを表し得る。そのようなものとして、この例では、タスク304aおよびタスク304bは、実行するために同じリソースを要求する場合があり、同時に実行することはない。同様に、タスク304bおよびタスク304cは、実行するために同じリソースを要求する場合があり、同時に実行することはない。

【００３９】

様々な実施形態では、タスクスケジューラ(図示せず)は、タスク競合エッジを動的タスク依存エッジと置き換え得る。タスク競合エッジを動的タスク依存エッジと置き換えることは、タスク間の関係を管理するために、データ構造内でタスクを関連付けることを含み得る。データ構造内でタスクを関連付けることは、データ構造内のそれらの配置がそれらの関係を暗示するようにタスクをデータ構造に追加すること、またはタスク間の新しい関係を示すように既存のエントリが変更されるようにデータ構造を更新することを含み得る。タスク競合エッジが動的タスク依存エッジによって置き換えられるかどうかは、タスク競合エッジに関連付けられたタスクがレディタスクであるかどうか、およびタスク競合エッジに関連付けられた1つまたは複数のリソースが、レディタスクによる使用のために利用可能であるかどうかまたはタスク競合が関連するタスクによって所有されているかどうかに依存する場合がある。

【００４０】

典型的には、すでに所有されているリソースの獲得に失敗するレディタスクに対して、レディタスクを実行するために割り当てられたスレッドおよび/またはプロセッサはスピンし、成功するまでリソースの獲得を連続的に試行することになり、そのことが、獲得までに複数の試行を行い、試行ごとにコンピューティングデバイスの電力、処理および帯域幅のリソースを使用することになる。スケジューラは、タスク間の関係を追跡し得る。スケジューラは、動的タスク依存エッジが持続する限り、レディタスクを実行するために割り当てられたスレッドおよび/またはプロセッサがアイドルに入ること、または他の無関係のレディタスクに取り組むことを可能にし得る。完了すると、リソースオーナータスクはスケジューラにシグナリングし、スケジューラは動的タスク依存エッジを除去し、レディタスクを実行するために割り当てられたスレッドおよび/またはプロセッサに、ウェイクアップして、完了されたタスクによって放棄されたリソースの獲得を試行するようにシグナリングし得る。

【００４１】

様々な実施形態では、レディタスクは動的優位依存性を介して複数の他のタスクと関連してもよく、スケジューラは、要求されたリソースが利用可能になったときに要求されたリソースをレディタスクが獲得することを可能にするか、または要求されたリソースのすべてが利用可能であるときにだけ要求されたリソースをレディタスクが獲得することを可能にしてもよい。リソース利用可能性は、様々な技法を使用して追跡され得る。様々な実施形態では、リソースはメモリ内の名前付きロケーションであってよく、メモリ内のバイトはどのタスクもそのリソースを所有していないことを示すためにゼロまたはヌルにセットされ、メモリ内のバイトはタスクがそのリソースを所有していることを示すためにそのタスクのアドレス(または「タスクポインタ」)にセットされる。名前付きリソースの所有権を獲得することは、リソースのメモリロケーション内に記憶されている値がゼロまたはヌルであることをアトミックに検査すること、およびその値をタスクポインタと交換することを伴う場合がある。リソースのメモリロケーション内に記憶されているゼロまたはヌル以外の値は、リソースがすでに記憶されている値に対応する異なるタスクによって所有

されていることを示し得る。デッドロックが周期的動的依存エッジによって引き起こされ、第1のタスクが第2のタスクによって所有されているリソースを獲得するのを待っており、第2のタスクが第1のタスクによって所有されているリソースを獲得するのを待っているという結果をもたらす場合がある。そのようなデッドロックを回避するために、様々な実施形態では、タスクに対するリソースは常に、正規順序で、たとえばリソースに対して対応するリソースハンドルのポインタアドレスのソートされた順序、または各リソースに割り当てられた一意のリソース識別子(ID)のソートされた順序で、処理され得る。様々な実施形態では、デッドロック検出およびリカバリ技法が使用され得る。言い換えれば、デッドロックおよび周期的動的タスク依存性は、第1のタスクの実行に要求されるリソースが、別のタスクの実行に要求される少なくとも1つの他のリソースに関連付けられた順序で処理されることを要求することによって回避され得る。

【００４２】

図3Bは、動的タスク依存エッジ308aがタスク競合エッジ306aを置き換えたことを除いて、図3Aを参照しながら説明したタスクグラフ構成300aと同じ構成を有する例示的なタスクグラフ構成300bを示す。この例では、タスク304aはレディタスクであり得る。そのようなものとして、タスク302aは実行を完了しており、タスク304aのタスク302aへの依存性を満足する。タスク304aを実行するために、コンピューティングデバイスのプロセッサまたはプロセッサコアのうちの1つによって実装される、タスク304aを実行するためのスレッドは、タスク304aを実行するために要求されるリソースの獲得を試行して成功することができる。本明細書で説明するように、タスク304aとタスク304bとの間のタスク競合エッジ306aは、タスク304aとタスク304bの両方が、実行するために同じリソースを要求することを示す。

【００４３】

いくつかの状況では、タスク304bもまたレディタスクであり、タスク304bを実行するためのスレッドは、タスク304aと304bの両方が実行するために要求するリソースを獲得することに失敗する場合がある。コンピューティングデバイスのタスクスケジューラまたは他のモジュールは、タスク304aと304bの両方が実行するために要求するリソースは、すでにタスク304aによって所有されていると決定する場合がある。それゆえ、タスク304bは、タスク304aがそのリソースの所有権を放棄するまで実行することはできないことになる。この状況では、コンピューティングデバイスのタスクスケジューラは、タスク304aとタスク304bとの間の競合エッジ306aを動的タスク依存エッジ308aと置き換えることができる。競合エッジ306aを動的タスク依存エッジ308aに換えることは、タスク304bが、今や、タスク304aの完了、それゆえタスク304bが実行するのに要求されるリソースの放棄に依存することを知らせ得る。動的タスク依存エッジ308aを追加することは、動的タスク依存エッジ308aが除去され、タスク304aが実行を完了して、タスク304bが実行するために要求されるリソースを放棄したことの表示を通知されるまで、スレッドはアイドルであるかまたは他の無関係のレディタスクに取り組んでもよいことを、タスク304bを実行するためのスレッドに知らせ得る。この例では、タスク304cは、タスク304aがリソースを所有しているかどうかにかかわらず、実行し得る。

【００４４】

図3Cは、動的タスク依存エッジ308aがタスク競合エッジ306aを置き換え、動的タスク依存エッジ308bがタスク競合エッジ306bを置き換えたことを除いて、図3Aを参照しながら説明したタスクグラフ構成300aと同じ構成を有する例示的なタスクグラフ構成300cを示す。タスクグラフ構成300cは、図3Bを参照しながら説明したタスクグラフ構成300bの例と同様の例を示す。すなわち、動的タスク依存エッジ308aおよび308bの各々は、それらのそれぞれのタスク304a～304cによって要求される異なるリソースに応じて、それらのそれぞれのタスク304a～304cを関連付け得る。言い換えれば、例示的なタスクグラフ構成300cは、タスク304aとタスク304bとの間の動的タスク依存エッジ308aを決定するプロセスを第1のリソースに対して適用し、タスク304bとタスク304cとの間の動的タスク依存エッジ308bを決定するプロセスを第2のリソースに対して適用し得る。その結果、動的タスク依存エッジ3

08aおよび308bは、互いに依存することはない。しかしながら、この例では、タスク304a～304cの実行は、事実上、互いに依存してもよい。タスク304aおよび304cは同じリソースを要求しないにもかかわらず、タスク304cはタスク304bに動的に依存する場合があり、タスク304bはタスク304aに動的に依存する場合がある。それゆえ、タスク304cは、タスク304bまで実行することができず、タスク304bは、タスク304aまで実行することができない。

【００４５】

タスクグラフ構成300cは、タスク304a～304cの各々が実行するために同じリソースを要求する一例を示す。そのような状況における様々な実施形態では、タスク競合エッジ306aを動的タスク依存エッジ308aと置き換えることは、図3Bを参照しながら説明したタスクグラフ構成300bの例と同様の方法で発生し得る。加えて、スケジューラは、タスク304cが、動的タスク依存エッジ308aの影響を受けやすいリソースと同じリソースを、実行するために要求する場合があると決定し得る。タスク304aによって放棄され、タスク304bおよび304cによって要求されるリソースに対する競合条件を回避するために、様々な実施形態を実施するタスクスケジューラは、タスク競合エッジ306bを動的タスク依存エッジ308bと置き換えることによってタスク304bおよび304cの実行の順序を優先順位付け得る。そのようにして、タスク304cがアイドルのままであるか、または他の無関係のレディタスクに取り組む間に、タスク304bはタスク304aによって放棄されたリソースの利用可能性を通知され、タスク304cはタスク304bによってリソースが放棄された後、そのリソースの利用可能性を通知され得る。

【００４６】

図3Dは、動的タスク依存エッジ308bがタスク競合エッジ306bを置き換えたことを除いて、図3Aを参照しながら説明したタスクグラフ構成300aと同じ構成を有する例示的なタスクグラフ構成300dを示す。タスクグラフ構成300dは、図3Bを参照しながら説明したタスクグラフ構成300bの例と同様の例を示す。すなわち、例示的なタスクグラフ構成300dでは、タスク304bはタスク304bおよび304cによって要求されるリソースを獲得するレディタスクである。リソースはタスク304bによって所有されているので、タスク304cは、それが実行するために要求するリソースを獲得することに失敗し、それゆえ、様々な実施形態によるタスクスケジューラは、タスク競合エッジ306bをタスク依存エッジ308bと置き換える。タスク304cを実行するためのスレッドおよび/またはプロセッサは、タスク304bが実行している間に、アイドルであることまたは他の無関係のレディタスクに取り組むことを許容され、タスク304bが実行を完了した後、タスク304bによって放棄されたリソースを獲得するようにシグナリングされる。この例では、タスク304aは、タスク304bがリソースを所有するかどうかにかかわらず、実行し得る。

【００４７】

図4A～図4Dは、様々なタスクグラフ構成400a～400dにおける例示的な1つのタスクグラフを示す。タスク302a～302d、タスク304a～304c、タスク依存エッジ(異なるタスク302a～302dおよび304aの間の矢印によって示され、タスク依存エッジ402を含む)、タスク競合エッジ306aおよび306b、ならびに動的タスク依存エッジ308aおよび308bを含む、タスクグラフおよびそれの様々なタスクグラフ構成400a～400dは単なる例であり、様々な数のタスクおよびタスク間の様々な関係を用いて構成され得る。

【００４８】

図4Aは、図3Aを参照しながら説明したタスクグラフ構成300aと同じ構成を有し、タスク304aと304cとの間にタスク依存エッジ402が追加された、例示的なタスクグラフ構成400aを示す。そのようなものとして、タスクグラフ構成400aは、タスク304cがタスク304aに従属し、したがってタスク304cはタスク304aの前に実行することができないことを除いて、タスクグラフ構成300aと同様の特性を含む。

【００４９】

図4Bは、図3Bを参照しながら説明したタスクグラフ構成300bと同じ構成を有し、タスク304aと304cとの間にタスク依存エッジ402が追加された、例示的なタスクグラフ構成400bを示す。タスク304aおよび304bの実行は、タスクグラフ構成300bに対して説明したものと

同じ例の影響を受けやすい場合がある。タスク304cの実行は、タスク競合エッジ306aを動的タスク依存エッジ308aと置き換えたことによる影響を受けない場合がある。しかしながら、この例では、タスク304aと304bとの間の動的タスク依存エッジ308aは、タスク304bが、タスク304aおよび304bを実行するために要求されるリソースを最初に獲得するように反転される。したがって、タスク304cは、事実上、タスク304aを介してタスク304bに依存するようになり得る。

【００５０】

図4Cは、図3Cを参照しながら説明したタスクグラフ構成300cと同じ構造を有し、タスク304aと304cとの間にタスク依存エッジ402が追加された、例示的なタスクグラフ構成400cを示す。タスク304aおよび304bの実行は、タスクグラフ構成300cに対して説明したものと同じ例の影響を受けやすい場合がある。さらに、タスク304cは、今や、タスク304aに従属し、動的タスク依存エッジ308aおよび308bの影響を受けやすいリソースが同じものであるか否かにかかわらず、タスク304bに動的に従属する場合がある。しかしながら、タスク304cを実行するために要求される第1のリソースをタスク304aによって放棄することは、タスク304aに直接的に従属しているにもかかわらず、タスク304cを実行するのに十分ではない場合がある。なぜならば、タスク304cを実行するために要求される第2のリソースが、タスク304bによって所有されている場合があるからである。様々な実施形態では、スケジューラは、タスク304aによって放棄された第1のリソースを獲得するために、タスク304cを実行するためのスレッドおよび/またはプロセッサをウェイクアップし得る。様々な実施形態では、スケジューラは、タスク304cを実行するためのスレッドおよび/またはプロセッサが、タスク304bによって第2のリソースが放棄されるのを待っている間、アイドルに戻ることを可能にし得る。様々な実施形態では、第1と第2の両方のリソースがタスク304aおよび304bによって放棄されるときにだけ、スケジューラは、タスク304cを実行するためのスレッドおよび/またはプロセッサをウェイクアップし得る。様々な実施形態では、第1と第2の両方のリソースが利用可能であるときにだけ、タスク304cを実行するためのスレッドおよび/またはプロセッサをウェイクアップすることを待つことは、別のタスクが、実行するために第1または第2のリソースのうちの1つを要求するときに実施され得る。

【００５１】

いくつかの状況では、同じリソースが、動的タスク依存エッジ308aおよび依存エッジ402の影響を受けやすい場合がある。タスク304aによって放棄され、タスク304bおよび304cによって要求されるリソースに対する競合条件を回避するために、様々な実施形態を実施するタスクスケジューラは、タスク304bおよび304cの実行の順序を優先順位付け得る。そのようにして、より優先度の低いタスクがアイドルのままであるか、または他の無関係のレディタスクに取り組む間に、より優先度の高いタスクは、タスク304aによって放棄されたリソースの利用可能性を通知され、より優先度の低いタスクは、より優先度の高いタスクによってリソースが放棄された後、そのリソースの利用可能性を通知され得る。様々な実施形態では、タスクスケジューラは、依存エッジ402より上に動的タスク依存エッジ308aおよび308bを優先順位付けてもよく、それにより、タスク304bがタスク304cとのタスク競合の影響を受けやすいリソースを獲得してタスク304cの前に実行するように、タスクスケジューラは、動的タスク依存エッジ308bを追加してもよい。

【００５２】

図4Dは、図3Dを参照しながら説明したタスクグラフ構成300dと同じ構造を有し、タスク304aと304cとの間にタスク依存エッジ402が追加された、例示的なタスクグラフ構成400dを示す。タスク304aおよび304bの実行は、タスクグラフ構成300dに対して説明したものと同じ例の影響を受けやすい場合がある。図4Cを参照しながら例示的なタスクグラフ構成400cに対して説明したように、スケジューラは、動的タスク依存エッジ308bを使用してタスク304bおよび304cの実行を優先順位付けてもよい。そのようなものとして、タスク304aによって放棄された第1のリソースの利用可能性にもかかわらず、タスク304cの実行はタスク304bが完了するまで遅延される場合がある。なぜならば、動的タスク依存エッジ308bは、タスク304bがタスク304cの前に第2のリソースを獲得することを可能にするように、タ

スクスケジューラに指示する場合があるからである。

【００５３】

図5は、タスク競合の存在下でのタスクスケジューリングのための一実施形態方法500を示す。方法500は、汎用ハードウェアまたは専用ハードウェア内のプロセッサで実行するソフトウェアにおいて、コンピューティングデバイス内で実施されてよく、任意のそのような構成におけるタスクスケジューラとして実施されてもよい。様々な実施形態では、方法500は、複数のプロセッサまたはハードウェア構成要素上の複数のスレッドによって実施され得る。

【００５４】

決定ブロック502では、コンピューティングデバイス(たとえば、コンピューティングデバイス内に実装されたタスクスケジューラ)は、タスクtがレディタスクであるかどうかを決定し得る。タスクtがレディタスクであるかどうかを決定するために、コンピューティングデバイスは、タスクtが任意の他のタスクに従属しているかどうかを決定し得る。いくつかの実施形態では、コンピューティングデバイスは、タスクtが、別のタスクへの依存性および/または動的な依存性を示すデータ構造内に含まれるかどうかを決定し得る。これは、単一のまたは複数のデータ構造における単一のエントリまたは複数のエントリを分析することを含み得る。そのようなデータ構造の各々は、特定のタスク関係に対する専用であってもよく、または複数のタイプのタスク関係を含んでもよい。タスクtが別のタスクに従属するとして入力されているとの決定は、タスクtはレディタスクでないことを示し得る。タスクtが別のタスクに従属しないとして入力されているかまたはデータ構造内に入力されていないとの決定は、タスクtは別のタスクに従属しておらず、それゆえレディタスクであることを示し得る。

【００５５】

タスクtがレディタスクではない(すなわち、決定ブロック502=「No」)と決定したことに応答して、コンピューティングデバイスは、決定ブロック502において、タスクtがレディタスクであるかどうかの検査を継続し得る。

【００５６】

タスクtがレディタスクである(すなわち、決定ブロック502=「Yes」)と決定したことに応答して、コンピューティングデバイスは、決定ブロック504において、タスクtがタスクtjとのタスク競合を有するかどうかを決定し得る。図6を参照しながらさらに説明するように、タスクtが別のタスクとの競合を有するかどうかの決定は、いくつかのタスクに対して実施され得る。タスクtが別のタスクに従属しているかどうかの決定と同様に、コンピューティングデバイスは、タスクtがタスクtjとの競合を有するかどうかを決定するためにデータ構造を分析し、タスクtとタスクtjとの間の競合を示すエントリを探索し得る。様々な実施形態では、データ構造は、競合関係を示すように構成されてよく、または各タスクによって要求されるリソースを示してもよく、要求されるリソースは、競合が存在するかどうかを決定するために比較されてもよい。たとえば、少なくとも1つの同じリソースを要求する2つのタスクは、互いに競合する場合がある。

【００５７】

タスクtはタスクtjと競合しない(すなわち、決定ブロック504=「No」)と決定したことに応答して、コンピューティングデバイスは、ブロック512においてタスクtが実行するために要求されるリソースを獲得し得る。たとえば、タスクtが実行している間に他のタスクがそのタスクにアクセスしないように、ロックが、要求されるリソースに取り付けられ得る。様々な実施形態では、リソースを獲得することはリソースに対する排他的アクセスを要求することはなく、たとえば読取りアクセスだけが要求されるとき、またはリソースの異なる部分がアクセスされるときに、リソースは他のタスクによって共有され得る。ブロック514において、コンピューティングデバイスは、獲得されたリソースを使用してタスクtを実行し得る。

【００５８】

タスクtはタスクtjとの競合を有する(すなわち、決定ブロック504=「Yes」)と決定した

ことに応答して、コンピューティングデバイスは、ブロック506においてタスクtを実行するために、タスク競合の影響を受けやすいリソースを含むリソースの獲得を試行し得る。コンピューティングデバイスは、ブロック512に対して説明したものと同様の方法で、リソースの獲得を試行し得る。様々な実施形態では、リソースの獲得を試行することは、図3Aを参照しながら説明したリソース利用可能性追跡および/またはデッドロック回避技法を使用し得る。様々な実施形態では、コンピューティングデバイスは、リソースを獲得する前にリソースの利用可能性を検査し得る。利用可能なリソースに対して、コンピューティングデバイスはリソースを獲得してよく、利用可能でないリソースに対して、コンピューティングデバイスはリソースの獲得をスキップしてよい。様々な実施形態では、コンピューティングデバイスは、個々のリソースが利用可能であるときに個々の利用可能なリソースを獲得してよく、または要求されたリソースのセットもしくは全部が利用可能であるときにだけリソースを獲得してもよい。

【００５９】

決定ブロック508において、コンピューティングデバイスは、タスクtを実行するために、タスクtが、タスク競合の影響を受けやすいリソースを含むリソースを獲得するかどうかを決定し得る。コンピューティングデバイスは、実行に対するタスクtの必要性に応じて、タスクtがリソースの排他的所有権または共有所有権を取ることを許容しないリソースの所有権を、他のタスクが有するとき、タスクtを実行するために要求されるリソースのいずれかの獲得に失敗する場合がある。様々な実施形態では、タスクtがリソースを獲得するかどうかに関する決定は、リソースの獲得を示す信号をタスクから受信すること、またはリソースに対してメモリ内にバイトでセットされたタスクのアドレス(または「タスクポインタ」)を検査することによってリソース利用可能性追跡を使用することを含み得る。

【００６０】

コンピューティングデバイスが、タスクtを実行するために、タスク競合の影響を受けやすいリソースを含むリソースの獲得に失敗したと決定した(すなわち、決定ブロック508=「No」)ことに応答して、コンピューティングデバイスは、ブロック510において、タスクtjからタスクtへの動的タスク依存エッジを割り当て得る。言い換えれば、コンピューティングデバイスは、タスクtをタスクtjに動的に従属させてよく、タスクtが実行され得る前にタスクtの実行に要求されるリソースをタスクtjが放棄することを要求する。タスクtjからタスクtへの動的タスク依存エッジを割り当てることが、図6を参照しながらさらに説明される。

【００６１】

コンピューティングデバイスが、タスクtを実行するために、タスク競合の影響を受けやすいリソースを含むリソースを獲得したと決定した(すなわち、決定ブロック508=「Yes」)ことに応答して、コンピューティングデバイスは、ブロック514において、獲得したリソースを使用してタスクtを実行し得る。

【００６２】

図6は、動的タスク依存性を割り当てるための一実施形態方法600を示す。方法600は、汎用ハードウェアまたは専用ハードウェア内のプロセッサで実行するソフトウェアにおいて、コンピューティングデバイス内で実施されてよく、任意のそのような構成におけるタスクスケジューラとして実施されてもよい。様々な実施形態では、方法600は、複数のプロセッサまたはハードウェア構成要素上の複数のスレッドによって実施され得る。様々な実施形態では、方法600は、図5を参照しながら説明したように、ブロック510の一部として実施され得る。

【００６３】

決定ブロック602において、コンピューティングデバイス(たとえば、コンピューティングデバイス内で実行するタスクスケジューラ)は、競合ポリシーが満足されるかどうかを決定し得る。様々な実施形態では、競合ポリシーは、いくつの動的タスク依存エッジを、割当てセッションごとにタスクtに割り当てるかを決定し得る。言い換えれば、タスクtと

の競合について分析されるべき他のタスクがまだ残っているかどうかにかかわらず、一定の数の動的タスク依存エッジがタスクtに割り当てられたことに応答して、競合ポリシーは割当てセッションを中断し得る。タスクポリシーは、タスクレイテンシ、電力消費、およびタスクの緊急性など、様々な因子に応じてセットされ得る。たとえば、タスクレイテンシを増加させるために、ポリシーは、タスクtに割り当てられている動的タスク依存エッジの数を増加させてよく、タスクレイテンシを減少させるために、動的タスク依存エッジの数を減少させてもよい。競合ポリシーが満足されたかどうかの決定はまた、時間、他のタスクの実行の数、プロセッササイクル、および他のメトリックに基づく場合がある。

【００６４】

タスクレイテンシは、動的タスク依存エッジの数に関連付けられ得る。なぜならば、要求されるリソースの獲得を試行する前に終了するのをタスクtが待たなければならないタスクが多いほど、それを待つために、より長い時間がかかるからである。同様に、動的タスク依存エッジが少ないほど、より高い頻度で、それゆえ潜在的により早く、要求されるリソースの獲得をタスクtが試行することを可能にし得る。別の例では、動的タスク依存エッジの数に対する電力消費の関係は、レイテンシと逆関係にあり得る。動的タスク依存エッジが少ないほど、より高い頻度で要求されるタスクの獲得をタスクtが試行し、したがってアイドル時間はより小さくなり、タスクtを実行するために割り当てられたスレッドおよび/またはプロセッサに対してよりアクティブになる。別の例では、タスクの緊急性が高いほど、より少ない動的タスク依存エッジが割り当てられ、それによりタスクは、潜在的に、実行に際して緊急の期間により一致して実行し得る。

【００６５】

競合ポリシーが満足されると決定した(すなわち、決定ブロック602=「Yes」)ことに応答して、コンピューティングデバイスは、決定ブロック502においてタスクtがレディタスクかどうかを決定してよく、図5を参照しながら説明したように方法500を継続する。

【００６６】

競合ポリシーが満足されないと決定した(すなわち、決定ブロック602=「No」)ことに応答して、コンピューティングデバイスは、ブロック604において、タスクtとタスクtjとの間のタスク競合エッジをタスクtjからタスクtへの動的タスク依存エッジと置き換え得る。図5を参照しながら説明したように、コンピューティングデバイスは、タスク間の関係を示すデータ構造を維持し得る。タスクtとタスクtjとの間のタスク競合エッジをタスクtjからタスクtへの動的タスク依存エッジと置き換えるために、コンピューティングデバイスは、1つまたは複数のデータ構造内で1つのエントリを生成または更新して、タスク競合エッジの除去と動的タスク依存エッジの追加とを示し得る。様々な実施形態では、様々な関係および様々なタスクを示すエントリが、1つまたは複数のデータ構造内で維持され得る。様々な実施形態では、タスク競合エッジエントリを除去または編集する必要はない場合がある。なぜならば、動的タスク依存エッジエントリは、単に、他の関係エントリにとって代わり得るからである。

【００６７】

ブロック606において、コンピューティングデバイスは、tjが関連しているタスクを、たとえばタスクtjをtj+1にセットすることによって変更し得る。本明細書で説明するように、動的タスク依存エッジの割当ては、競合ポリシーが満足されるまで、またはタスクtと競合するタスクがなくなるまで継続し得る。しかしながら、タスクが残っており、競合ポリシーが満足されない限り、タスクtは次のタスクと比較され得る。決定ブロック504において、コンピューティングデバイスは、タスクtがタスクtjとのタスク競合を有するかどうかを決定してよく、図5を参照しながら説明したように方法500を続ける。

【００６８】

様々な実施形態では、競合しているタスクtjによって所有されている、実行するために要求されるリソースをタスクtが獲得できないと決定することと、タスク競合エッジを動的タスク依存エッジと置き換えることとの間の時間内に、タスクtjが完了しないことを確保することが重要である場合がある。この制約は、動的タスク依存エッジが追加される前

にリソースが利用可能になったので、タスクtが、リソースが利用可能であることをシグナリングされるのを待ちながら放置されることを防止するのに役立つ場合がある。それゆえ、タスクtjの実行が次に完了するまで、または図6を参照しながら示した何らかの他のメトリックが次に完了するまで、タスクtはリソースの獲得を試行しない場合がある。この遅延はまた、タスクtに直接的または間接的に従属する他のタスクにつながる場合がある。図7および図8を参照しながら説明する方法は、タスクtjの所有権の原子性を確保すること、およびタスクtjからタスクtへの動的タスク依存エッジの追加に関する。

【００６９】

図7は、タスク競合を動的タスク依存性と置き換えるための一実施形態方法700を示す。方法700は、汎用ハードウェアまたは専用ハードウェア内のプロセッサで実行するソフトウェアにおいて、コンピューティングデバイス内で実施されてよく、任意のそのような構成におけるタスクスケジューラとして実施されてもよい。様々な実施形態では、方法700は、複数のプロセッサまたはハードウェア構成要素上の複数のスレッドによって実施され得る。様々な実施形態では、方法700は、図6を参照しながら説明したように、ブロック604の一部として実施され得る。

【００７０】

決定ブロック702において、コンピューティングデバイス(たとえば、コンピューティングデバイス内で実行するタスクスケジューラ)は、タスクtjの実行が完了されたかどうかを決定し得る。様々な実施形態では、コンピューティングデバイスは、タスクtjまたはタスクtjを実行するスレッドおよび/もしくはプロセッサから、実行が完了したとの信号を受信し得る。信号は、コンピューティングデバイスのスケジューラによって受信され得る。タスクtjの実行が完了していないと決定した(すなわち、決定ブロック702=「No」)ことに応答して、コンピューティングデバイスは、図6を参照しながらブロック604において説明したように、ブロック704においてタスク競合エッジをタスクtjからタスクtへの動的タスク依存エッジと置き換え得る。ブロック704においてタスク競合エッジをタスクtjからタスクtへの動的タスク依存エッジと置き換えた後、タスクtjの実行が完了したと決定した(すなわち、決定ブロック702=「Yes」)ことに応答して、コンピューティングデバイスは、図6を参照しながら説明するように、ブロック606においてtjが関連するタスクを変更してよく、方法600を進める。

【００７１】

図8は、タスク競合を動的タスク依存性と置き換えるための一実施形態方法800を示す。方法800は、汎用ハードウェアまたは専用ハードウェア内のプロセッサで実行するソフトウェアにおいて、コンピューティングデバイス内で実施されてよく、任意のそのような構成におけるタスクスケジューラとして実施されてもよい。様々な実施形態では、方法800は、複数のプロセッサまたはハードウェア構成要素上の複数のスレッドによって実施され得る。様々な実施形態では、方法800は、図6を参照しながら説明したように、ブロック604の一部として実施され得る。

【００７２】

ブロック802において、コンピューティングデバイス(たとえば、コンピューティングデバイス内で実行するタスクスケジューラ)は、タスクtjをロックし得る。様々な実施形態では、コンピューティングデバイスは、タスクtjを実行するために割り当てられたスレッドおよび/またはプロセッサへの割込み信号または命令を用いてタスクtjの実行を中断させることによってタスクtjをロックし、それによりタスクtjの実行を休止させることができる。様々な実施形態では、コンピューティングデバイスは、実行のためにタスクtjによって要求される1つまたは複数のリソースへの排他的アクセスを獲得することによって、またはさもなければタスクtjがそれらのリソースにアクセスするのを防止することによってタスクtjをロックし、タスクtjの実行を同様に休止させることができる。

【００７３】

ブロック804において、コンピューティングデバイスは、図6を参照しながら説明した方法600のブロック604における動作と同様に、タスク競合エッジをタスクtjからタスクtへ

の動的タスク依存エッジと置き換え得る。

【００７４】

ブロック806において、コンピューティングデバイスは、タスクtjをアンロックし得る。様々な実施形態では、コンピューティングデバイスは、タスクtjを実行するために割り当てられたスレッドおよび/またはプロセッサへの再開信号または命令を用いてタスクtjの実行を再開させることによってタスクtjをアンロックし、それによりタスクtjの実行を休止解除させることができる。様々な実施形態では、コンピューティングデバイスは、実行のためにタスクtjによって要求される1つまたは複数のリソースへの、排他的アクセスを放棄することによって、またはさもなければタスクtjがそれらのリソースにアクセスすることを可能にすることによってタスクtjをアンロックし、タスクtjの実行を同様に休止解除させることができる。次いで、コンピューティングデバイスは、図6を参照しながら説明したように、ブロック606においてtjが関連するタスクを変更してよく、方法600を進める。

【００７５】

(図1～図8を参照して上で論じた実施形態を含むが、それらに限定されない)様々な実施形態は、図9に示す様々な実施形態とともに使用するのに適した例示的なモバイルコンピューティングデバイスを含み得る多種多様なコンピューティングシステム内で実装され得る。モバイルコンピューティングデバイス900は、タッチスクリーンコントローラ904と内部メモリ906とに結合されたプロセッサ902を含んでもよい。プロセッサ902は、汎用または特定の処理タスクに指定された1つまたは複数のマルチコア集積回路であってもよい。内部メモリ906は、揮発性メモリまたは不揮発性メモリであってもよく、また、セキュアおよび/もしくは暗号化メモリであっても、または非セキュアおよび/もしくは非暗号化メモリであっても、あるいはそれらの任意の組合せであってもよい。活用されてもよいメモリタイプの例には、限定はしないが、DDR、LPDDR、GDDR、WIDEIO、RAM、SRAM、DRAM、P-RAM、R-RAM、M-RAM、STT-RAM、および埋め込みDRAMが含まれる。タッチスクリーンコントローラ904およびプロセッサ902は、抵抗感知タッチスクリーン、静電容量感知タッチスクリーン、赤外線感知タッチスクリーンなどのタッチスクリーンパネル912にも結合され得る。加えて、コンピューティングデバイス900のディスプレイは、タッチスクリーン能力を有する必要がない。

【００７６】

モバイルコンピューティングデバイス900は、互いにおよび/またはプロセッサ902に結合された、通信を送信および受信するための1つまたは複数の無線信号トランシーバ908(たとえば、Peanut、Bluetooth(登録商標)、Zigbee、Wi-Fi、RF無線)と、アンテナ910とを有してもよい。トランシーバ908およびアンテナ910は、様々なワイヤレス伝送プロトコルスタックおよびインターフェースを実装するために、上述した回路で使用されてもよい。モバイルコンピューティングデバイス900は、セルラーネットワークを介した通信を可能にし、プロセッサに結合されたセルラーネットワークワイヤレスモデムチップ916を含み得る。

【００７７】

モバイルコンピューティングデバイス900は、プロセッサ902に結合された周辺デバイス接続インターフェース918を含んでもよい。周辺デバイス接続インターフェース918は、1つのタイプの接続を受け入れるように単独で構成されてよく、あるいはUSB、FireWire、Thunderbolt、またはPCIeなどの一般または独自の様々なタイプの物理接続および通信接続を受け入れるように構成されてもよい。周辺デバイス接続インターフェース918はまた、同様に構成された周辺デバイス接続ポート(図示せず)に結合されてもよい。

【００７８】

モバイルコンピューティングデバイス900は、オーディオ出力を生成するためのスピーカー914を含む場合もある。モバイルコンピューティングデバイス900はまた、本明細書で論じる構成要素のすべてまたはいくつかを収容するための、プラスチック、金属、または材料の組合せから構築されたハウジング920を含み得る。モバイルコンピューティングデ

バイス900は、使い捨てバッテリーまたは充電式バッテリーなどの、プロセッサ902に結合された電源922を含み得る。充電式バッテリーはまた、モバイルコンピューティングデバイス900の外部にあるソースから充電電流を受けるために周辺デバイス接続ポートに結合され得る。モバイルコンピューティングデバイス900はまた、ユーザ入力を受け取るための物理ボタン924を含んでもよい。モバイルコンピューティングデバイス900は、モバイルコンピューティングデバイス900をオンオフするための電源ボタン926を含んでもよい。

【００７９】

(図1～図8を参照して上で論じた実施形態を含むが、それらに限定されない)様々な実施形態は、図10に示すラップトップコンピュータ1000など、様々なモバイルコンピューティングデバイスを含み得る多種多様なコンピューティングシステム内で実装され得る。多くのラップトップコンピュータは、コンピュータのポインティングデバイスとして働くタッチパッドのタッチ面1017を含み、したがって、タッチスクリーンディスプレイを装備した上述のコンピューティングデバイス上で実装されるものと同様のドラッグジェスチャ、スクロールジェスチャ、およびフリックジェスチャを受信してもよい。ラップトップコンピュータ1000は通常、揮発性メモリ1012、およびフラッシュメモリのディスクドライブ1013などの大容量不揮発性メモリに結合されたプロセッサ1011を含む。加えて、コンピュータ1000は、プロセッサ1011に結合されたワイヤレスデータリンクおよび/または携帯電話トランシーバ1016に接続される場合がある、電磁放射を送受信するための1つまたは複数のアンテナ1008を有する場合がある。コンピュータ1000は、プロセッサ1011に結合されたフロッピーディスクドライブ1014およびコンパクトディスク(CD)ドライブ1015を含む場合もある。ノートブック構成では、コンピュータのハウジングは、すべてがプロセッサ1011に結合された、タッチパッド1017、キーボード1018、およびディスプレイ1019を含む。コンピューティングデバイスの他の構成はよく知られているように、(たとえば、USB入力を介して)プロセッサに結合されたコンピュータマウスまたはトラックボールを含んでよく、それらはまた、様々な実施形態とともに使用され得る。

【００８０】

(限定はしないが、図1～図8を参照して上記で説明した実施形態を含む)様々な実施形態は、サーバキャッシュメモリ内にデータを圧縮するための様々な市販のサーバのうちのいずれかを含んでもよい多種多様なコンピューティングシステムにおいて実装されてもよい。例示的なサーバ1100が、図11に示されている。そのようなサーバ1100は通常、揮発性メモリ1102と、ディスクドライブ1104などの大容量不揮発性メモリとに結合された、1つまたは複数のマルチコアプロセッサアセンブリ1101を含む。図11に示すように、マルチコアプロセッサアセンブリ1101は、それらをアセンブリのラックに挿入することによって、サーバ1100に付加されてもよい。サーバ1100はまた、プロセッサ1101に結合されたフロッピーディスクドライブ、コンパクトディスク(CD)、またはデジタル多用途ディスク(DVD)ディスクドライブ1106を含み得る。サーバ1100はまた、他のブロードキャストシステムコンピュータおよびサーバに結合されたローカルエリアネットワーク、インターネット、公衆交換電話網、ならびに/またはセルラーデータネットワーク(たとえば、CDMA、TDMA、GSM(登録商標)、PCS、3G、4G、LTE、もしくは任意の他のタイプのセルラーデータネットワーク)など、ネットワーク1105とのネットワークインターフェース接続を確立するための、マルチコアプロセッサアセンブリ1101に結合されたネットワークアクセスポート1103を含み得る。

【００８１】

様々な実施形態の動作を実行するためのプログラマブルプロセッサ上での実行のためのコンピュータプログラムコードまたは「プログラムコード」は、C、C++、C#、Smalltalk、Java(登録商標)、JavaScript(登録商標)、Visual Basic、Structured Query Language(たとえば、Transact-SQL)、Perlなどの高水準プログラミング言語または様々な他のプログラミング言語で記述されてもよい。本出願において使用される、コンピュータ可読記憶媒体上に記憶されたプログラムコードまたはプログラムは、そのフォーマットがプロセッサによって理解可能である(オブジェクトコードなどの)機械語コードを指すことがある。

【００８２】
上記の方法の説明およびプロセスフロー図は、単に説明のための例として提供されており、様々な実施形態の動作が提示された順序で実行されなければならないことを要求または暗示するものではない。当業者によって諒解されるように、上記の実施形態における動作の順序は、任意の順序で実行されてもよい。「その後」、「次いで」、「次に」などの単語は、動作の順序を限定するものではなく、これらの単語は単に、方法の説明を通じて読者を導くために使用される。さらに、たとえば、冠詞「a」、「an」、または「the」を使用する単数形でのクレーム要素へのいかなる参照も、要素を単数形に限定するものとして解釈すべきではない。

【００８３】
様々な実施形態に関して説明した様々な例示的な論理ブロック、モジュール、回路、およびアルゴリズム動作は、電子ハードウェア、コンピュータソフトウェア、または両方の組合せとして実装され得る。ハードウェアとソフトウェアのこの互換性を明確に示すために、様々な例示的な構成要素、ブロック、モジュール、回路、および動作について、上記では概してそれらの機能に関して説明した。そのような機能性がハードウェアまたはソフトウェアのどちらで実施されるのかは、特定の応用例と、システム全体に課せられる設計制約とに依存する。当業者は、説明した機能を特定の適用例ごとに様々な方法で実装してよいが、そのような実装決定は、特許請求の範囲からの逸脱を引き起こすものとして解釈されるべきではない。

【００８４】
本明細書で開示される実施形態に関連して説明された様々な例示的な論理、論理ブロック、モジュール、および回路を実施するのに使用されるハードウェアは、汎用プロセッサ、デジタル信号プロセッサ(DSP)、特定用途向け集積回路(ASIC)、フィールドプログラマブルゲートアレイ(FPGA)もしくは他のプログラム可能論理デバイス、ディスクリートゲート論理もしくはトランジスタ論理、ディスクリートハードウェア構成要素、または本明細書で説明される機能を実行するように設計されたその任意の組合せを用いて実施されまたは実行され得る。汎用プロセッサはマイクロプロセッサであってよいが、代替として、プロセッサは任意の従来のプロセッサ、コントローラ、マイクロコントローラ、またはステートマシンであってよい。プロセッサはまた、コンピューティングデバイスの組合せ、たとえば、DSPとマイクロプロセッサとの組合せ、複数のマイクロプロセッサ、DSPコアと連携した1つもしくは複数のマイクロプロセッサ、または任意の他のそのような構成として実装され得る。代替的に、いくつかの動作または方法は、所与の機能に固有の回路機構によって実施され得る。

【００８５】
1つまたは複数の実施形態では、説明した機能は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せにおいて実装されてもよい。ソフトウェアにおいて実装される場合、機能は、1つまたは複数の命令またはコードとして、非一時的コンピュータ可読媒体または非一時的プロセッサ可読媒体上に記憶され得る。本明細書で開示する方法またはアルゴリズムの動作は、非一時的コンピュータ可読記憶媒体または非一時的プロセッサ可読記憶媒体上に存在し得るプロセッサ実行可能ソフトウェアモジュールにおいて具現化され得る。非一時的コンピュータ可読記憶媒体または非一時的プロセッサ可読記憶媒体は、コンピュータまたはプロセッサによってアクセスされ得る任意の記憶媒体であり得る。限定ではなく例として、そのような非一時的コンピュータ可読またはプロセッサ可読媒体は、RAM、ROM、EEPROM、FLASHメモリ、CD-ROMもしくは他の光ディスクストレージ、磁気ディスクストレージもしくは他の磁気ストレージデバイス、または命令もしくはデータ構造の形態で所望のプログラムコードを記憶するために使用され、コンピュータによってアクセスされ得る任意の他の媒体を含んでもよい。本明細書で使用される場合、ディスク(disk)およびディスク(disc)は、コンパクトディスク(disc)(CD)、レーザーディスク(登録商標)(disc)、光ディスク(disc)、デジタル多用途ディスク(disc)(DVD)、フロッピーディスク(disk)、およびblu-rayディスク(disc)を含み、ここで、ディスク(disk

)は、通常、データを磁気的に再生するが、ディスク(disc)は、レーザを用いて光学的にデータを再生する。上記の組合せも、非一時的コンピュータ可読およびプロセッサ可読媒体の範囲に含まれる。追加として、方法またはアルゴリズムの動作は、コンピュータプログラム製品に組み込まれてもよい、非一時的プロセッサ可読媒体および/またはコンピュータ可読媒体上のコードおよび/または命令の1つまたは任意の組合せまたはセットとして存在してもよい。

【００８６】

開示される実施形態の前述の説明は、任意の当業者が特許請求の範囲を製作または使用することを可能にするために提供される。これらの実施形態への様々な修正が当業者には容易に明らかになり、本明細書において定義される一般原理は、特許請求の範囲から逸脱することなく他の実施形態に適用されてもよい。したがって、本開示は、本明細書において示される実施形態に限定されるものではなく、以下の特許請求の範囲ならびに本明細書において開示された原理および新規の特徴に一致する最大の範囲を与えられるものである。

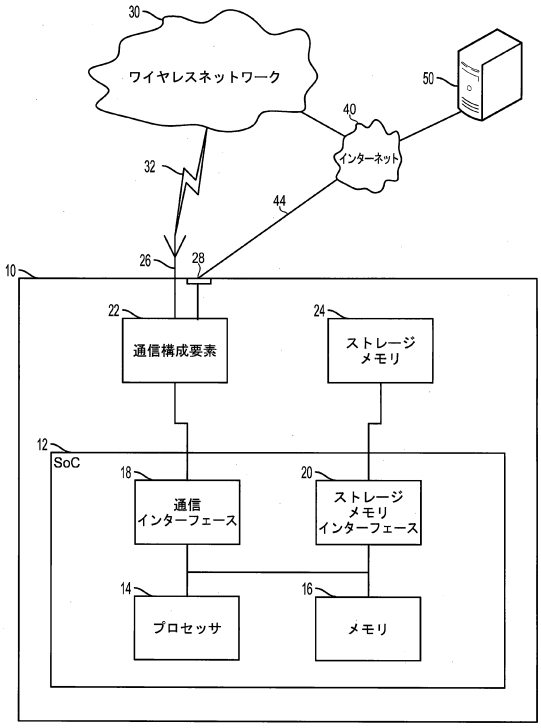
【符号の説明】

【００８７】

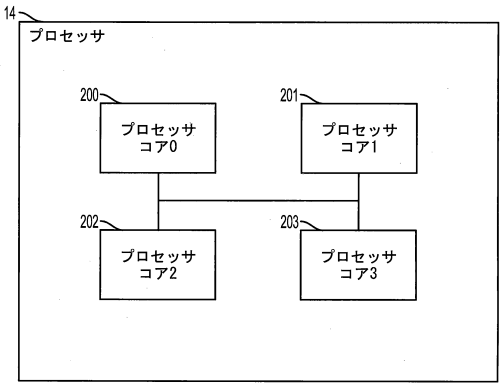
10 コンピューティングデバイス
12 システムオンチップ(SoC)
14 プロセッサ
16 メモリ
18 通信インターフェース
20 ストレージメモリインターフェース
22 通信構成要素
24 ストレージメモリ
26 アンテナ
28 ネットワークインターフェース
30 ワイヤレスネットワーク
32 ワイヤレス接続
40 インターネット
44 ワイヤード接続
50 コンピューティングデバイス
200 プロセッサコア
201 プロセッサコア
202 プロセッサコア
203 プロセッサコア
300a タスクグラフ構成
300b タスクグラフ構成
300c タスクグラフ構成
300d タスクグラフ構成
302a タスク
302b タスク
302c タスク
302d タスク
304a タスク
304b タスク
304c タスク
306a タスク競合エッジ
306b タスク競合エッジ
308a 動的タスク依存エッジ
308b 動的タスク依存エッジ

400a タスクグラフ構成
400b タスクグラフ構成
400c タスクグラフ構成
400d タスクグラフ構成
402 タスク依存エッジ
900 モバイルコンピューティングデバイス
902 プロセッサ
904 タッチスクリーンコントローラ
906 内部メモリ
908 トランシーバ
910 アンテナ
912 タッチスクリーンパネル
914 スピーカー
916 セルラーネットワークワイヤレスモデムチップ
918 周辺デバイス接続インターフェース
920 ハウジング
922 電源
924 物理ボタン
926 電源ボタン
1000 ラップトップコンピュータ
1008 アンテナ
1011 プロセッサ
1012 揮発性メモリ
1013 ディスクドライブ
1014 フロッピーディスクドライブ
1015 コンパクトディスク(CD)ドライブ
1016 携帯電話トランシーバ
1017 タッチ面
1018 キーボード
1019 ディスプレイ
1100 サーバ
1101 マルチコアプロセッサアセンブリ
1102 揮発性メモリ
1103 ネットワークアクセスポート
1104 ディスクドライブ
1105 ネットワーク
1106 デジタル多用途ディスク(DVD)ディスクドライブ

【図１】



【図２】



【図３Ａ】

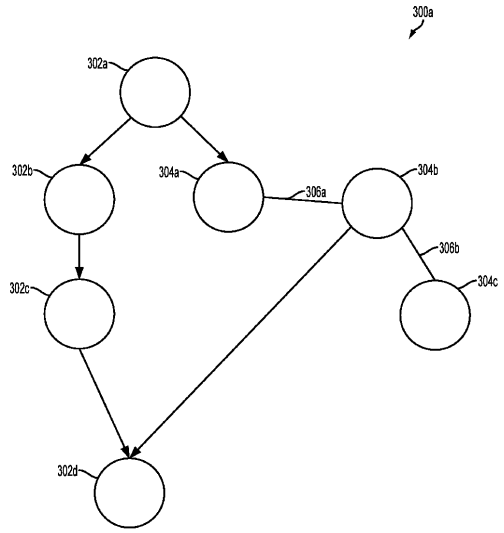


FIG. 3A

【図３Ｂ】

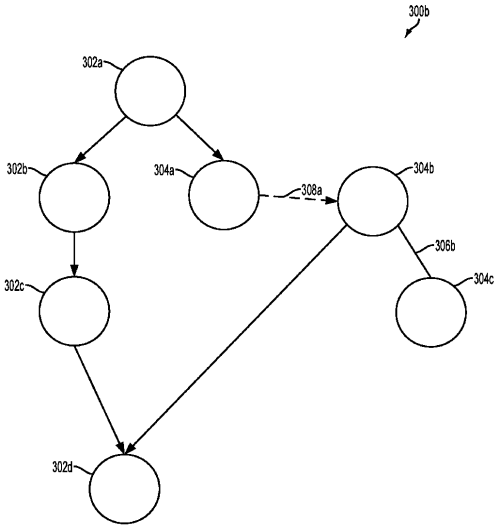


FIG. 3B

【図３Ｃ】

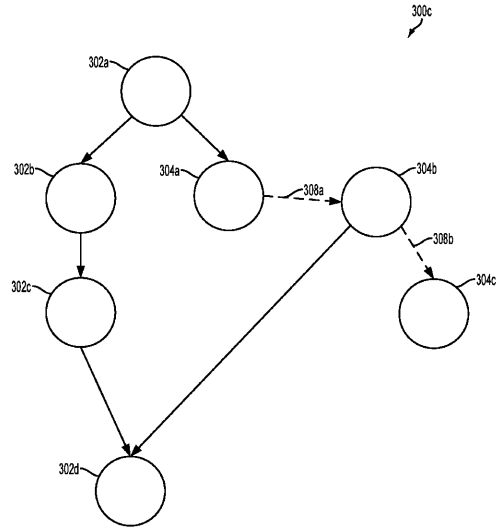


FIG. 3C

【図３Ｄ】

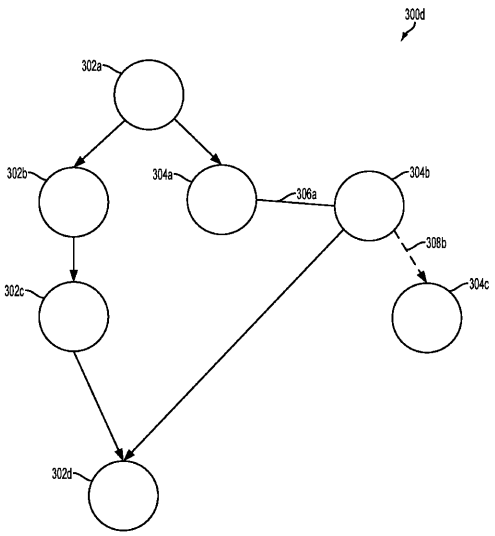


FIG. 3D

【図４Ａ】

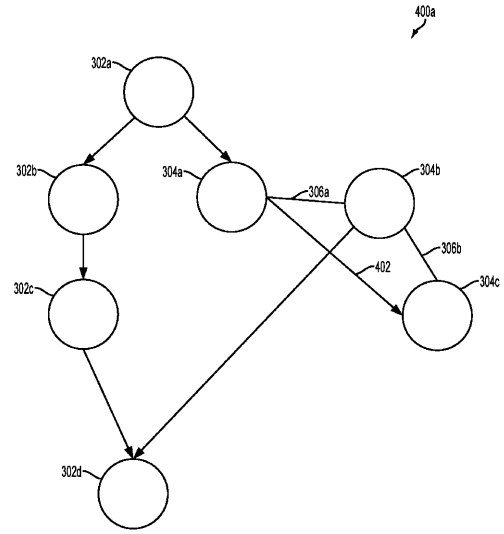


FIG. 4A

【図４Ｂ】

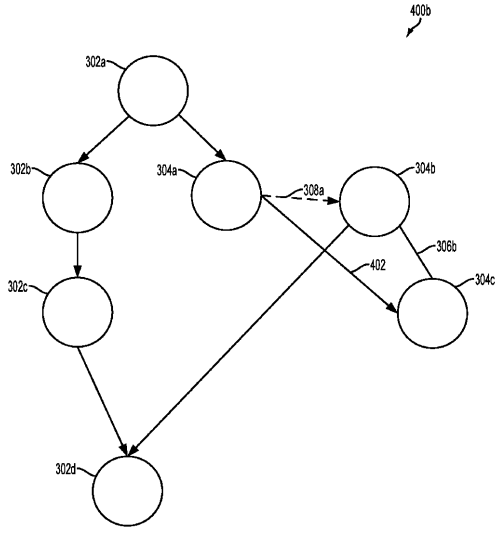


FIG. 4B

【図４Ｃ】

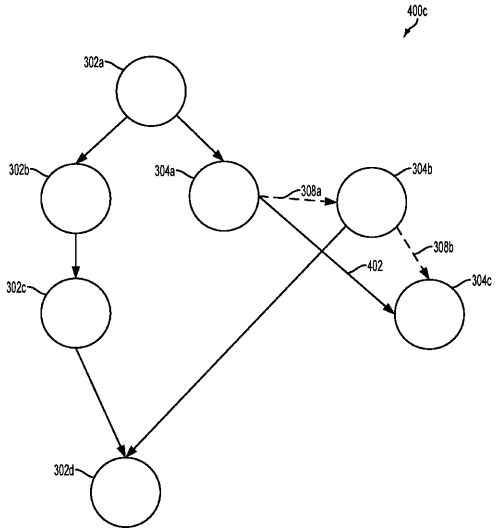


FIG. 4C

【図４Ｄ】

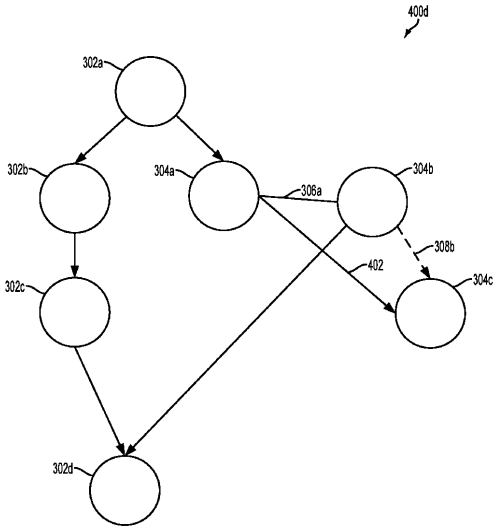
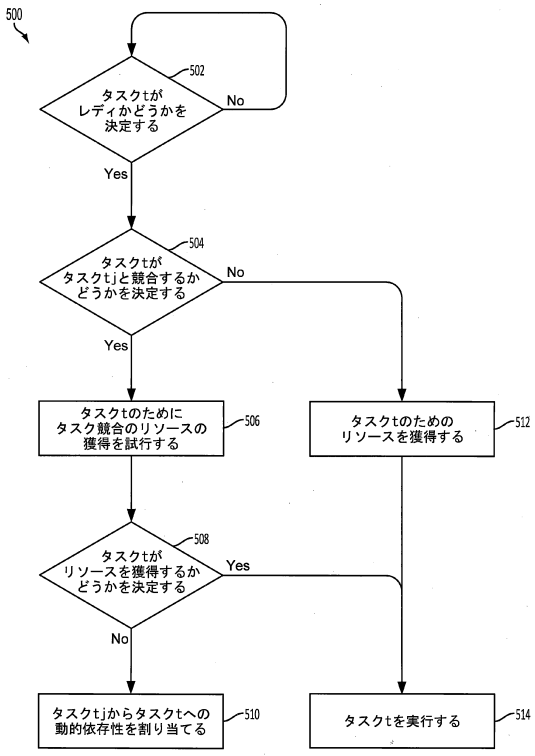
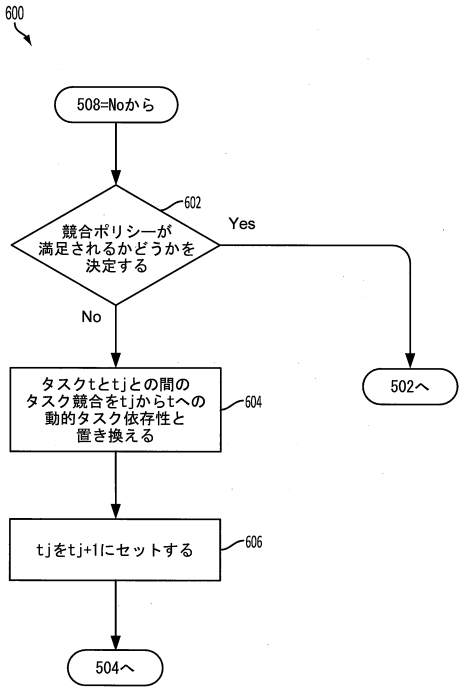


FIG. 4D

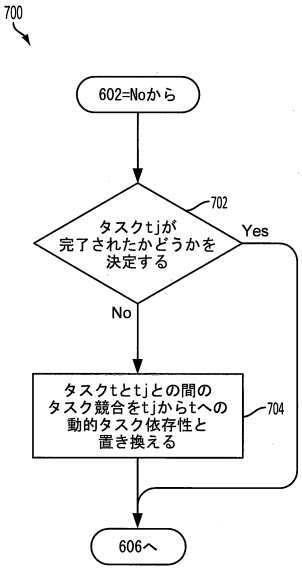
【図５】



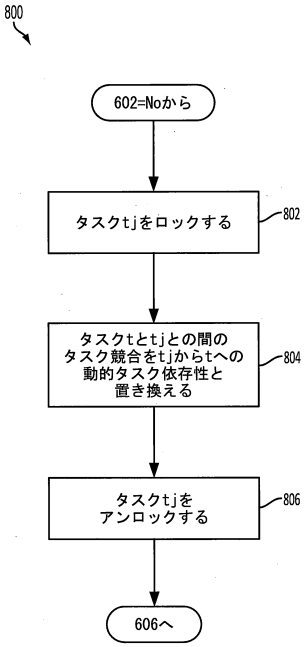
【図６】



【図７】



【図８】



【図９】

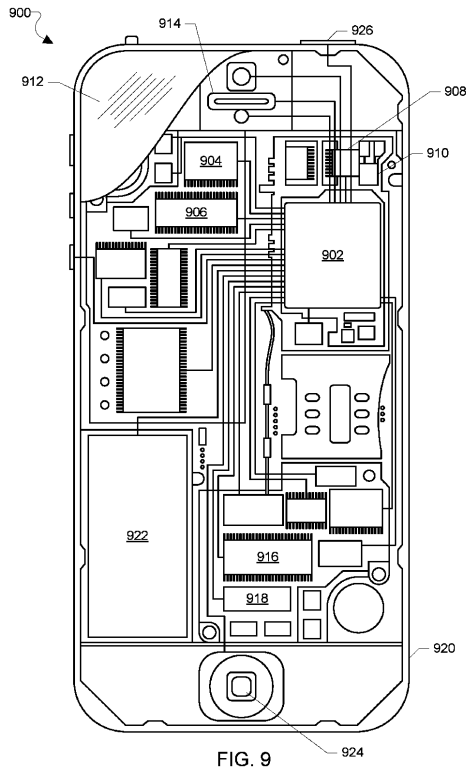


FIG. 9

【図１０】

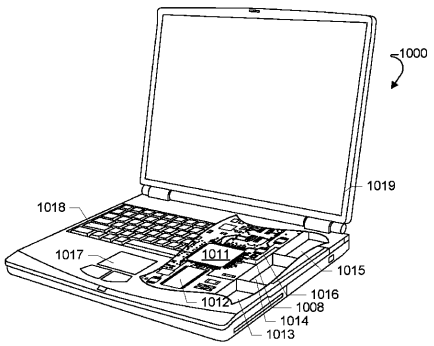


FIG. 10

【図１１】

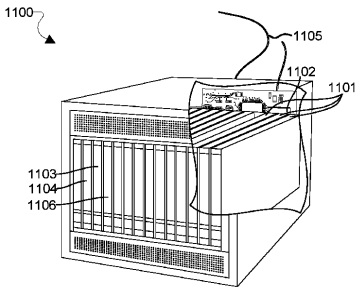


FIG. 11

フロントページの続き

(72)発明者　トゥシャー・クマール
アメリカ合衆国・カリフォルニア・９２１２１－１７１４・サン・ディエゴ・モアハウス・ドライヴ・５７７５

審査官　井上　宏一

(56)参考文献　特表２００７－５１７３２６（ＪＰ，Ａ）
特表２００７－５０５３７４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４５５－９／５４