(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5366552号
(P5366552)
(45)発行日　平成25年12月11日(2013.12.11)　(24)登録日　平成25年9月20日(2013.9.20)

(51)Int.Cl.　F I
G06F 9/50 (2006.01)　G06F 9/46 465C

請求項の数 16 (全 22 頁)

(21)出願番号	特願2008-538384 (P2008-538384)
(86)(22)出願日	平成18年6月8日 (2006.6.8)
(65)公表番号	特表2009-515246 (P2009-515246A)
(43)公表日	平成21年4月9日 (2009.4.9)
(86)国際出願番号	PCT/FR2006/050535
(87)国際公開番号	WO2007/051935
(87)国際公開日	平成19年5月10日 (2007.5.10)
審査請求日	平成21年6月8日 (2009.6.8)
(31)優先権主張番号	0511266
(32)優先日	平成17年11月4日 (2005.11.4)
(33)優先権主張国	フランス(FR)

(73)特許権者　510225292
コミサリア ア レネルジー アトミック エ オ ゼネルジー アルテルナティブ
COMMISSARIAT A L'ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES
フランス, パリ エフ－７５０１５, リュー ルブラン ２５, バティマン ル ポナン デ
Batiment Le Ponant D, 25 rue Leblanc, F-75015 Paris, FRANCE
(74)代理人　100107641
弁理士　鎌田　耕一

最終頁に続く

(54)【発明の名称】集中特化したマルチタスク及びマルチフロー処理をリアルタイム実行する手法及びシステム

(57)【特許請求の範囲】
【請求項１】
特定のマルチタスク及びマルチフロー処理をリアルタイム実行するシステムであって、
（ａ）タスク中の補助プロセッシング部（ＡＰＰ）により処理されない演算処理を実行し、タスク制御を行う中央プロセッサコアと、（ｂ）前記補助プロセッシング部（ＡＰＰ）により特定の演算処理を実行することを要求する追加の命令群を処理する制御ユニット（ＥＳＣＵ）と、を備えた標準プロセッシング部（ＳＰＰ）を備え、
前記補助プロセッシング部（ＡＰＰ）が、（ｉ）それぞれが、前記特定の演算処理に関して高速処理が可能となるように最適化され、かつ与えられた時間内でタスク中の１つの命令ストリームのみを実行するＮ個の補助演算ユニット（ＡＰＵ０、・・・、ＡＰＵＮ－１）と、（ｉｉ）エレメンタリな命令ストリームに分けられ、補助演算ユニットに割り当てる処理の並列処理を実行し、前記命令ストリームの実行環境の管理としてプリエンプション管理および前記補助演算ユニット間のデータ伝送管理を含む管理を行う補助リソース割り当て制御ユニット（ＡＣＵ）と、(iii)内部ネットワークを介して前記補助演算ユニット（ＡＰＵ０、・・・、ＡＰＵＮ－１）間で共有されるメモリ空間（ＳＭＳ）と、を備え、
様々なシステム要素が、前記補助演算ユニット（ＡＰＵ０、・・・、ＡＰＵＮ－１）と前記中央プロセッサコアとの間の通信が前記共有メモリ空間（ＳＭＳ）と前記内部ネットワークとを介して行われるように配置されていることを特徴とするシステム。
【請求項２】

前記補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）により取り扱うすべてのデータとプログラムを格納した大容量メモリ（ＭＭ）をさらに備えたことを特徴とする請求項１に記載のシステム。

【請求項３】

前記大容量メモリ（ＭＭ）を制御するメインメモリ制御部（ＭＭＣ）を備えたことを特徴とする請求項２に記載のシステム。

【請求項４】

前記補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）がプログラマブルユニット、リコンフィギュラブルユニット、専用ユニットを備えたことを特徴とする請求項１から３のいずれか１項に記載のシステム。

【請求項５】

前記中央プロセッサコアが、演算ユニット（ＳＰＵ）と、メモリユニット（Ｌ１　Ｄ－キャッシュ，Ｌ２　Ｉ－キャッシュ，Ｌ２－キャッシュ）と、ローディングユニット（ＬＳＵ）をさらに備えたことを特徴とする請求項１から４のいずれか１項に記載のシステム。

【請求項６】

４から８の補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）を備えたことを特徴とする請求項１から５のいずれか１項に記載のシステム。

【請求項７】

第１にシステムバス（ＳＢ）の通信を優先し、第２に入出力装置（ＩＯ）および前記大容量メモリ（ＭＭ）とともに前記中央プロセッサコアの通信を優先するように管理するバスアービター（ＳＢＡ）を備えたことを特徴とする請求項２または３に記載のシステム。

【請求項８】

システムバス（ＳＢ）に接続された複数個のプロセッサであって、前記プロセッサがそれぞれ、前記中央プロセッサコアと、前記Ｎ個の補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）と、前記共有メモリ空間（ＳＭＳ）と、前記補助リソース割り当て制御ユニット（ＡＣＵ）を備えたものであることを特徴とする請求項１から７のいずれか１項に記載のシステム。

【請求項９】

前記システムバス（ＳＢ）に接続されている複数個のプロセッサ間で共有するシステムバスアービター（ＳＢＡ）を備えたことを特徴とする請求項８に記載のシステム。

【請求項１０】

中央プロセッサコアを備えた少なくとも１つの標準プロセッサ部（ＳＰＰ）と、Ｎ個の補助演算ユニット（ＡＰＵ，・・・，ＡＰＵＮ－１）を備えた補助プロセッシング部（ＡＰＰ）と、前記Ｎ個の補助演算ユニット（ＡＰＵ，・・・，ＡＰＵＮ－１）で内部ネットワークを介して共有されるメモリ空間（ＳＭＳ）と、前記補助プロセッシング部（ＡＰＰ）および前記補助演算ユニット（ＡＰＵ，・・・，ＡＰＵＮ－１）にタスク中の特定の演算処理を実行することを要求する追加の命令群を処理する制御ユニット（ＥＳＣＵ）と、補助リソース割り当て制御ユニット（ＡＣＵ）と、を備えた少なくとも１つのプロセッサを用い、特定のマルチタスク及びマルチフロー処理をリアルタイム実行する方法であって、

前記中央プロセッサコアが、タスクにおける前記補助プロセッシング部（ＡＰＰ）により処理されない演算処理を実行するとともにタスク制御を行い、前記補助演算ユニット（ＡＰＵ，・・・，ＡＰＵＮ－１）が、それぞれ前記特定の演算処理に関して高速処理が可能となるように最適化され、かつ与えられた時間内でタスク中の１つの命令ストリームのみを実行し、前記補助リソース割り当て制御ユニット（ＡＣＵ）が、エレメンタリな命令ストリームに分けられ、前記補助演算ユニット（ＡＰＵ，・・・，ＡＰＵＮ－１）に割り当てる処理の並列処理を実行し、前記命令ストリームの実行環境の管理としてプリエンプション管理および前記補助演算ユニット間のデータ伝送管理を含む管理を行い、

前記補助演算ユニット（ＡＰＵ０、・・・、ＡＰＵＮ－１）間の通信または前記補助演

算ユニット（ＡＰＵ０、・・・、ＡＰＵＮ－１）と前記中央プロセッサコアとの間の通信が前記共有メモリ空間（ＳＭＳ）と前記内部ネットワークとを介して行われることを特徴とする方法。

【請求項１１】

各々の命令ストリームが一つの前記補助演算ユニットのみで実行されることを特徴とする請求項１０に記載の方法。

【請求項１２】

前記補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）で取り扱われるすべてのデータおよびプログラムが大容量メモリ（ＭＭ）に格納され、前記プロセッサがシステムバス（ＳＢ）に接続され、前記中央プロセッサコア、入出力制御部（ＩＯ）および前記大容量メモリ（ＭＭ）からの前記システムバスへの通信がバスアービター（ＳＢＡ）によって管理されていることを特徴とする請求項１０に記載の方法。

【請求項１３】

前記標準プロセッサ部（ＳＰＰ）に割り当てられたタスクが、特定命令が出てくるまで前記標準プロセッサ部（ＳＰＰ）のサイクルごとに処理されてゆき、前記特定命令に関してはデコードされると前記割り当て制御ユニット（ＡＣＵ）に対するコマンドが生成され、前記割り当て制御ユニット（ＡＣＵ）の制御によって前記補助演算ユニット（ＡＰＵ０，・・・，ＡＰＵＮ－１）の一つで実行される命令ストリームが生成され、ひとたび、特定命令がデコードされ、対応する前記コマンドが生成された場合に、命令ストリームが生成され前記補助演算ユニットにおける実行が管理されても、前記標準プロセッサ部では干渉を受けることなく実行中の現タスクの継続が可能であることを特徴とする請求項１０から１２のいずれか１項に記載の方法。

【請求項１４】

トラッピング処理がエクセプション処理、インタラプト処理、またはトラップ処理を伴う場合、トラッピングタイプの機能として、前記プロセッサ内のすべての構成要素が同期しなければならない、強い同期処理が選択される請求項１３に記載の方法。

【請求項１５】

トラッピング処理がエクセプション処理、インタラプト処理、またはトラップ処理を伴う場合、トラッピングタイプの機能として、前記標準プロセッシング部に対応する実行環境は同期するが、補助リソース割り当て制御ユニット（ＡＣＵ）は補助演算ユニットにて独立してアクティブとなっているファンクションの実行を継続させる、弱い同期処理が選択される請求項１３または１４に記載の方法。

【請求項１６】

トラッピング処理が補助演算ユニットのローカルイベントを伴う場合、当該関係する前記補助演算ユニットのみが前記トラッピング処理を制御し、他のプロセッサとは独立して同期制御が実行される、選択的ローカル同期が有効となる請求項１３から１５のいずれか１項に記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は集中特化したマルチタスク及びマルチフロー処理をリアルタイム実行する手法及びシステムに関する。

【０００２】

本発明は特に組み込み型マルチプロセッサアーキテクチャに適している。

【背景技術】

【０００３】

本発明は以下の特徴を備えたシステムにおける処理に関する問題解決手段の提供を目的とする。

・高性能：組み込み型アプリケーションはますます複雑化しつつある。それは組み込み型システムにより多くの機能を実装する必要性にせまられていること（マルチメディア、ゲ

ーム、テレコミュニケーション、携帯電話におけるＧＰＳ機能利用等）、および、処理データ量が増大していること（ビデオセンサ、高速コンバーター等の能力向上）からも明らかである。組み込み型システムでは複数の情報処理を同時に並列処理する能力が求められている。それゆえシステム内に分散されている各ユニットにおいて並列処理に必要なすべての情報を効率的に収集、分配、処理することが独立に行われる必要がある。この多数の情報処理を同時に並列処理するという必要性は、マルチタスク実行環境においても同じである。

・柔軟性：ターゲットとなるシステムではオープン性が要求される。システムを利用するどのユーザーでも行いたい業務が自由に実行できなければならない。それゆえシステムのアーキテクチャは多様な利用用途に適するように十分な柔軟性を備えていなければならない。このオープン性ゆえにアーキテクチャ全体にわたってアプリケーションコンテンツの実行前のオフライン状態での最適化は設計段階では十分に行うことができない。アルゴリズムによってはプロセスを単に静的に分割する並列制御が適当なもの（実行前のオフライン状態でも決められるもの）もあるが、その他のアルゴリズムでは実行中の動的ストリーム制御が要求されており、組み込みアプリケーションの複雑性の増大に伴ってこの傾向はますます強まるであろう。

・動作環境への統合性：システムは動作環境に統合されるように開発されていなければならない。この統合性はリアルタイム性、消費電力、コスト、信頼性などの諸条件が反映される。

・異種処理(ヘテロプロセッシング)：アプリケーションの多様性と組み込みシステムにおける制御の流れの複雑性のため、様々なタイプの処理が組み込みアーキテクチャ内で共存せざるを得ない。それゆえ、集中処理タスクはアプリケーションの異なる要素間において干渉し合っても優先されるべき制御となっているタスクとして実行される必要がある。

【０００４】

以上まとめると、ターゲットとする組み込みシステムは、異なるデータストリームを実行環境に動的に適合させつつ処理する高い演算性能と通信性能とを備えている必要がある。組み込みシステムは同時に外部環境（消費電力、リアルタイム性等）により大きく制限され、オープンであることが要求され、複数マシンでの利用が前提となる。この環境には様々なタスクが（実行中において）動的に生成され、保持され、消去等されるというマルチアプリケーションシステム環境が含まれている。

【０００５】

このような組み込みシステムでは、アーキテクチャの実行前のオフラインでの最適化を行うと、リソース利用の正確な手順の決定が不可能であるために、リソースの利用不足を招いてしまうことが問題となる。そこで逆に、実行中のオンラインでの最適化に注力し、実行前にオフラインですべての処理動作を事前に予測する必要性を低減した方が良い。しかし、実行前にオフラインによるアーキテクチャの最適化ができないとなると余力あるコストパフォーマンスが高い高価な制御メカニズムが必要となってしまう。本発明は動的制御が統合されていない環境においてもパフォーマンスが低下しない計算機構造を提供することを目的とする。

【０００６】

実行中のシステムでの実行環境の競合において、並列処理を用いることはアプリケーションでのオペレーションレベルまたは命令レベルの並列処理での効果をもたらすものと考えられてきた。しかし、命令レベルでの高度な並列処理を可能とするアーキテクチャを研究することに注力されてきたものの、このアプローチには限界があることが明らかである。また、組み込みアプリケーションの複雑化により、単一の制御ストリームの形でモデリングすることは非常に難しいものである。それでもユーザやアーキテクチャ設計者はタスクレベルでの並列処理を求めている。その結果、現在の有力な技術トレンドは同じシリコン基板上にたくさんのプロセッサコアを組み込み、同一の回路基板上でタスクの並列処理を行うものとなっている。

【０００７】

パフォーマンスを向上させるものとして、並列処理を用いた手法に分類される多くの解決策が提案されている。主要なモデルは、同時マルチスレッディング（Simultaneous MultiThreading：ＳＭＴ）と、チップマルチプロセッシング（Chip MultiProcessing：ＣＭＰ）と、チップマルチスレッディング（Chip MultiThreading：ＣＭＴ）である。

【０００８】

例えば、ＳＭＴテクノロジーは最新のインテル製品、ＩＢＭおよびＨＰのアルファプロセッサに実装されている。これらにおいて、複数の命令のストリームから選ばれた命令を実行処理する演算ユニットを割り当ててゆくために複数のプログラムカウンタが用いられている。タスクの相互依存性が限定的であるので、プロセッサにおける命令レベルでの並列処理（ＩＬＰ）が増加するため、プロセッサパフォーマンスも結果として向上する。これらプロセッサの実装は難しく、読み込みステージと命令分配ステージの複雑性が増す。結果として、これらアーキテクチャは大規模回路となってしまい、組み込みシステムの条件、特にコストと消費電力の条件面において合わなくなってしまう。

【０００９】

図１ＡはＳＭＴアーキテクチャ理論を示すブロック図である。演算ユニットまたはファンクションユニットＦＵは、タスク割り当て器ＴＤと連動する単一の制御リソースであるＣＰによりプロセスが与えられる。各サイクルにおいて、制御ブロックＣＰはタスク割り当て器ＴＤと連動し、ファンクションユニットＦＵに対してデータ処理に利用可能な機能と処理上の問題点とを通知する。それぞれのファンクションユニットは共有メモリ空間ＳＭＳを共用し合う。

【００１０】

図１Ｂは、４つのファンクションユニットＦＵを備えた構成における処理操作の例を示している。この図では、各々のブロック１は命令を表わしており、縦軸２は命令割り当てと制御タスクを表わしている。

【００１１】

×印がついているブロック３は、データやリソースの依存性によりファンクションユニットには利用されていないタイムスロットに対応している。

【００１２】

次に、ＣＭＰを用いた技術は、比較的シンプルな実装で良いため、もともと組み込みシステムに適したものである。

【００１３】

このＣＭＰを用いた技術は、利用可能性に応じて各々の演算リソースにタスクを分散するというものである。各々の演算リソースは次々と割り当てられたタスクを実行してゆく。これらアーキテクチャはホモ構造とヘテロ構造の２つのファミリーに分けることができる。

・ヘテロ構造：この構造はヘテロ構造で与えられたアプリケーション領域に最適化された演算ユニットが組み込まれ、コンパイル時に前もって認識されたリソースに対してタスクを分散するものである。コンパイル時にパーティション化されたソフトウェアは実行時における（動的な）タスク分散のためにそのメカニズムが簡素化されている。これらアプリケーション指向のソリューションではＯＭＡＰ、ＶＩＰＥＲ、ＰＮＸおよびノマディックプラットフォームを含んでいる。

・ホモ構造：これらの構造は、ＩＢＭセルプラットフォームやＡＲＭのＭＰコアプラットフォームや、与えられたアプリケーション領域に最適化されたもの、例えばＭＰＥＧ４－ＡＶＣコーディング／デコーディング向けに最適化されたクレイドルテクノロジー社のＣＴ３４００のように、一般的に用いられるホモ構造の演算ユニットが組み込まれたものがベースとなっている。前者は広い範囲の問題を解決することをターゲットとしている。しかし後者は明らかに特定のアプリケーション領域に最適化されている。

【００１４】

図２Ａは、ＣＭＰアーキテクチャの理論を示すブロック図である。共有メモリ空間ＳＭＳと協働している演算ユニット（ファンクションユニット）ＦＵに対して、タスク割り当

て器ＴＤと連動した単一の制御リソースＣＰによって処理が供給される。タスク割り当て器ＴＤに接続された制御ユニットＣＰは、タスクの実行準備が整っているか決定する。演算リソースが開放されるとすぐにタスクを割り当て、データがロードされるとすぐに処理が開始される。図２Ｂは４つのファンクションユニットＦＵで構成された例を示しており、ハッチングが付けられて示されているブロック４がタスク処理開始を示している。ここで、ブロック１は命令を表し、縦線２は命令の割り当てとタスク制御を表している。

【００１５】

次に、マルチプロセスアンドＣＭＴアーキテクチャは、前記２つのモデルの組み合わせである。ＣＭＰコンセプトでは並列処理を専用の演算構成における複数タスクの実行処理まで拡張している。

【００１６】

この技術は本質的にサーバータイプでの実行環境も想定されている。

【００１７】

図３Ａは、汎用的ＣＭＴアーキテクチャモデルを示している。演算ユニット（ファンクションユニット）にはタスク割り当て器ＴＤに接続されている単一の制御リソースＣＰによって処理が供給される。ファンクションユニットＦＵは共有メモリ空間ＳＭＳと協働している。

【００１８】

図３ＢはファンクションユニットＦＵにおける処理の例を示している。

【００１９】

タスク割り当て器ＴＤと接続されている制御ユニットＣＰはタスクの実行準備が整ったか決定する。演算リソースが開放されるとすぐにタスクを割り当て、データがロードされるとすぐに処理が開始される。図３Ｂにハッチングが付けられている領域４によってタスク処理開始が示されており、ブロック１は命令を表し、縦線２は命令の割り当てとタスク制御を表している。

【００２０】

各々の演算リソースは多数のタスクを同時に管理している。例えばキャッシュ容量が足りなくなったなどの理由でタスク処理が滞るとすぐに新しいファンクションユニットＦＵが割り当てられる。このような環境では、ファンクションユニット内でのタスクの切り替え処理には実行環境のロードペナルティが発生しない。

【００２１】

実行性能を高めるために命令ストリーム（スレッド）の並列処理を用いるこれらアーキテクチャをベースとするエミュレーションにかかわらず、ＳＭＴであってもＣＭＰであってもＣＭＴであっても、これらアーキテクチャは組み込みシステムにおける諸問題を部分的にしか解決することができない。この事情の主な原因は、アプリケーションには異なる処理クラスのものが混在しておりそれらを区別ができないからである。そのため同じ処理リソースの中において、実行時間の観点からクリティカルであり制御が優先されるべき処理であっても他の通常処理と同一レベルに扱われてしまう。演算リソースは通常処理のサポートもクリティカル処理のサポートと同様に行ってしまい、最適化されていない演算結果しか得られないシステムとなり、消費電力の観点、コストパフォーマンスの観点、信頼性の観点の三重の観点からアプリケーションの要求には適さないものとなっている。

【００２２】

しかしながら、ＣＭＰタイプのシステムでも通常処理とクリティカル処理を区別できるものがある。これらのアーキテクチャは特化した処理を実行できる専用の演算リソースが搭載されているものである。ここでイレギュラーな処理は汎用プロセッサのシステムソフトウェアを用いている。特化した処理を実行できる専用の演算リソースが搭載されているので、パフォーマンスや消費電力効率が改善されるように最適化されうるが、タスク処理間の通信、タスク処理とシステムソフトウェアまたは制御プロセス間の通信が非効率的であり、システムレベルではその最適化の恩恵を受けられない。アーキテクチャ内の様々な要素間の通信ではシステムバスが用られるが、通信バンド幅の不足によるペナルティを招

きやすい。そのため、システムには伝送制御情報が遅延してしまうというペナルティと、ビットレートが遅くなるというペナルティが発生し、データ伝送が乱れてしまう。これらペナルティにより応答性の遅いアーキテクチャということとなってしまい、システムソフトウェアは演算リソースを最適には使用できない。

【００２３】

このオーバーヘッドを低減するため、米国特許出願ＵＳ２００５／０１４９９３７Ａ１では演算リソース間の同期制御機構が専用構成となっているが、その解決方法ではタスク間のデータ伝送の問題には適用できない。米国特許出願ＵＳ２００４／００８８５１９Ａ１では高性能プロセッサの実行環境におけるタスクの並列処理管理による解決方法を示しているが、その解決方法ではコスト面から組み込みシステムには適用できない。

【発明の開示】

【発明が解決しようとする課題】

【００２４】

従来技術において開発されているタスクレベルでの並列処理を用いる解決方法は、上記問題のすべてを解決することはできない。ＳＭＴタイプの解決手段では、例えば、典型的には汎用プロセッサをベースとし、追加の制御ステージを付加するものであるが、しかしながら、その解決方法では従来の汎用プロセッサが持っている消費電力の問題を解決することはできず、加えて、多数のスレッドを並列処理管理するために複雑性が増大してしまう。

【００２５】

ＣＭＰタイプアーキテクチャの実装には多様なものがあるが、どれも上記問題を解決するために組み込みシステムに対して採用することは難しい。第１に、アプリケーション指向の解決は十分な柔軟性をもたらすことはできない。第２に、汎用アーキテクチャは演算による解決手段を提供できるものでなく、汎用プロセッサを開発するというコストのかかる解決方法をベースとし続けなければならない。同様に、ＣＭＴによる解決手段は、アーキテクチャによる並列処理を拡張したものであるが、消費電力の要求を解決するものでなく、また、回路内での通信されるデータの一致性が保たれるように管理しなければならないという問題に直面してしまう。

【００２６】

本発明は、上記した障害を解決することを目的とし、特に、プロセッサにハイレベルの演算リソースを搭載することを可能とするものである。

【課題を解決するための手段】

【００２７】

上記目的を達成するために本発明のシステムは、特定のマルチタスク及びマルチフロー処理をリアルタイム実行するシステムであって、アプリケーションのスレッドのうち、クリティカルでないスレッドは中央プロセッサコア自身で実行されるように割り当て、集中処理すべきまたは特定のスレッドは特定の命令に対して高速処理が可能なように最適化されているＮ個の補助演算ユニットを備えた補助プロセッシング部で実行されるように割り当て制御をする制御ユニットを備え、システムソフトウェアをサポートする中央プロセッサコアと、内部ネットワークを介して前記補助演算ユニット間で共有されるメモリ空間と、前記集中処理すべきまたは特定のスレッドに対応する各々の命令ストリームをまず並列に前記補助プロセッシング部に割り当て、次にこれら前記命令ストリームの実行を同期制御し、前記命令ストリームの実行環境を管理するように各々の補助リソースに対する制御を行う補助リソース割り当て制御ユニットとを備え、様々なシステム要素が、前記補助演算ユニット間の通信または前記補助演算ユニットと前記中央プロセッサコア間の通信が前記共有メモリ空間と前記内部ネットワークを介して行われるように配置されていることを特徴とする。

【００２８】

上記本発明のシステムは前記中央プロセッサコアに接続されたシステムバスを備えている。

【００２９】
また、上記本発明のシステムは、前記補助演算ユニットにより取り扱うすべてのデータとプログラムを格納した大容量メモリを備えている。
【００３０】
メインメモリ制御部は、前記大容量メモリに接続されている。
【００３１】
上記本発明のシステムにおいて、少なくとも一つの入出力周辺機器と接続された入出力制御部を備えている。入出力信号は共有メモリ空間を介して他のシステム要素において利用可能とされる。したがってクリティカルタイムにおいてもシステムの処理を行うことができる。
【００３２】
前記補助演算ユニットがプログラマブルユニット、リコンフィギュラブルユニット、専用ユニットから選ばれるユニットを備えた構成である。
【００３３】
前記共有メモリ空間が複数のメモリリソースと前記メモリリソースを統合する内部ネットワークを備え、前記共有メモリ空間において前記補助演算ユニットで取り扱うすべてのデータが格納されている。
【００３４】
前記共有メモリ空間がさらに、前記演算に関するリソースと前記共有メモリ空間内の前記メモリリソース間のリンクを確立せしめるメモリ空間制御部を備えている。
【００３５】
前記中央プロセッサコアが、演算ユニットと、メモリユニットと、ローディングユニットを備えている。
【００３６】
前記制御ユニットが、前記補助プロセッシング部を制御するための追加の命令群を備えている。
【００３７】
前記ローディングユニットが、前記中央プロセッサコアと前記補助プロセッシング部間でデータ交換するための追加レジスタキューを備えている。
【００３８】
前記補助演算ユニットのそれぞれが、一時にタスク中の一つの命令ストリームのみを処理するように制御され、アプリケーション向けに最適化され、Ｎ個の数が２から１００で構成されている。
【００３９】
補助リソース割り当て制御ユニットが、動的消費電力管理、障害管理、クライシスモード管理の少なくとも一つ以上の管理機能を担っている。
【００４０】
一実施例として、第１に前記システムバスの通信を優先し、第２に前記中央プロセッサコアと前記大容量メモリ間の通信を優先するように管理するバスアービターを備えた構成がある。
【００４１】
また、一実施例として、システムバスに接続された複数個のプロセッサであって、前記プロセッサがそれぞれ、前記中央プロセッサコアと、前記Ｎ個の補助演算ユニットと、前記共有メモリ空間と、前記補助リソース割り当て制御ユニットを備えた構成がある。
【００４２】
上記システム構成として、前記システムバスと前記中央プロセッサコア間の通信を管理するバスアービターを備えた構成がある。
【００４３】
上記システム構成として、前記複数のプロセッサ間で共有される大容量メモリを備えた構成がある。

【００４４】
また、本発明は、中央プロセッサコアと、コントロールユニットと、Ｎ個の補助演算ユニットと、前記Ｎ個の補助演算ユニットで内部ネットワークを介して共有されるメモリ空間と、補助リソース割り当て制御ユニットを備えた少なくとも一つのプロセッサ上で用いられる特定のマルチタスク及びマルチフロー処理をリアルタイム実行する方法であって、前記中央プロセッサコアが、システムソフトウェアを実行するとともに、前記制御ユニットが、アプリケーションのスレッドのうち、クリティカルでないスレッドは前記中央プロセッサコア自身で実行されるように割り当て、集中処理すべきまたは特定のスレッドは特定の命令に対して高速処理が可能なように最適化されているＮ個の補助演算ユニットを備えた補助プロセッシング部で実行されるように前記補助リソース割り当て制御ユニットを介して割り当て制御をし、前記補助リソース割り当て制御ユニットが前記集中処理すべきまたは特定のスレッドに対応する各々の命令ストリームをまず並列に前記補助ユニットに割り当て、次にこれら前記命令ストリームの実行を同期制御し、前記命令ストリームの実行環境を管理するようにし、少なくとも、データ通信が前記補助演算ユニット間の通信または前記補助演算ユニットと前記中央プロセッサコア間の通信が前記共有メモリ空間と前記内部ネットワークを介して行われることを特徴とする。

【００４５】
上記システムにおいて、前記制御ユニットが、標準のリード命令／標準のライト命令または標準のエクセプション命令によって、前記補助リソースの前記割り当て制御ユニットを制御する。

【００４６】
前記制御ユニットが、通信および同期処理に特化した特別の命令群によって、前記補助リソースの前記割り当て制御ユニットを制御する。

【００４７】
与えられた時間内で、前記補助演算ユニットのそれぞれが一時にタスク中の一つの命令ストリームのみを処理し、各々の命令ストリームが一つの前記補助演算ユニットで実行される。

【００４８】
前記補助演算ユニットで取り扱われるすべてのデータおよびプログラムが大容量メモリに格納されている。

【００４９】
クリティカル時には、入出力信号が共有メモリ空間を介して前記補助演算ユニットに伝送される。

【００５０】
前記プロセッサはシステムバスに接続されている。

【００５１】
一実施例として、第１に前記システムバスの通信を優先し、第２に前記中央プロセッサコアと前記入出力装置間の通信を優先するように管理するバスアービターを備えた構成がある。

【００５２】
前記中央プロセッサコアに割り当てられたタスクが、特定命令が出てくるまで前記中央プロセッサコアのサイクルごとに処理されてゆき、前記特定命令に関してはデコードされると前記割り当て制御ユニットに対するコマンドが生成され、前記割り当て制御ユニットの制御によって前記補助演算ユニットの一つで実行される命令ストリームが生成され、ひとたび、特定命令がデコードされ、対応する前記コマンドが生成された場合に、命令ストリームが生成され前記補助演算ユニットにおける実行が管理されても、前記中央プロセッサコアでは干渉を受けることなく実行中の現タスクの継続が可能である。

【００５３】
トラッピング処理がエクセプション処理、インタラプト処理、またはトラップ処理を伴う場合、トラッピングタイプの機能として、前記プロセッサ内のすべての構成要素が同期

しなければならない、強い同期処理が有効となる。
【００５４】
また、トラッピング処理がエクセプション処理、インタラプト処理、またはトラップ処理を伴う場合、トラッピングタイプの機能として、前記標準プロセッシング部に対応する実行環境は同期するが、補助リソース割り当て制御ユニット（ＡＣＵ）は補助演算ユニットにて独立して命令ストリームの実行を継続させる、弱い同期処理が選択される。
【００５５】
トラッピング処理が補助演算ユニットのローカルイベントを伴う場合、当該関係する前記補助演算ユニットのみが前記トラッピング処理を制御し、他のプロセッサとは独立して同期制御が実行される、選択的ローカル同期が有効となる。
【００５６】
従来技術とは異なり、本発明はプロセッサ内の演算リソースを強く統合することができる新しい結合メカニズムを実現している。
【００５７】
本発明のシステムアーキテクチャは、第１のサブシステムは中央プロセッサコアを形成する標準プロセッシング部（ＳＰＰ）を備え、第２のサブシステムは補助プロセッシング部（ＡＰＰ）を備え、補助プロセッシング部（ＡＰＰ）には補助演算ユニットと制御割り当て補助リソースと共有メモリ空間が実装されている。
【００５８】
２つのサブシステムは異なる特性と機能を持っているが、タスクを実行するという同じ目的を持っている。結果として、これらの機能はデータ処理および制御レベルで強く結合されている。
【発明を実施するための最良の形態】
【００５９】
本発明の他の特徴や利点は以下の実施例に関する詳細な説明と参照する添付図面から明らかになるであろう。
【００６０】
システム１０は、アプリケーション１１、１２、さらにタスク２１から２５、最終的に命令（スレッド）３１から３３の流れに細分化されているが、まず、いわゆる“ライトプロセス”と呼ばれるものを図４を参照しつつ説明する。
【００６１】
組み込みシステム１０は典型的には多数のアプリケーション１１や１２などのプロセスの並列処理に用いられる。アプリケーションは組み込みシステムにより提供される機能やサービスを用いる。組み込みシステムで処理されるどのアプリケーション１１、１２もタスク２１から２５の形に分割され、アプリケーションの記述による制御依存性に応じて一連のものにまとめられている。これらのタスク２１から２５は、並列処理が可能となるようにシーケンシャルに実行される操作処理ごとに並列スレッド３１から３３に分割される。
【００６２】
この詳細な説明において、スレッドという語は、他のプロセスとアドレス空間全体を共有することができる実行ストリームであるライトプロセスを表わすものとして使用されている。
【００６３】
図５は、本発明のプロセッサアーキテクチャの例を示す図である。第１のサブシステムは中央プロセッサコアを形成する標準プロセッシング部ＳＰＰを備え、第２のサブシステムは補助プロセッシング部ＡＰＰを備えている。
【００６４】
標準プロセッシング部ＳＰＰは一般的なタスクの実行処理を担っている。処理されるべきプログラム命令とシステムソフトウェアとを含んでいる。従来技術のプロセッサとは違い、標準プロセッシング部ＳＰＰは、補助プロセッシング部ＡＰＰの補助実行ユニットＡ

ＰＵ０、ＡＰＵ１、・・・ＡＰＵＮ－２、ＡＰＵＮ－１をコールして、強力な演算パワーを必要とするアプリケーション部分を実行させる。

【００６５】

本発明は、補助演算ユニットをコールという方法によって特定のプロセスを実行する。

【００６６】

標準プロセッシング部ＳＰＰは、アプリケーションにおける汎用的な演算処理を担う。また、標準プロセッシング部ＳＰＰは、リソース共有とタスク制御とを管理するシステムソフトウェアも処理する。標準プロセッシング部ＳＰＰは、汎用プロセッサにより形成されている。それゆえ、以下の従来型の４つのユニットを含んでいる。

１．コントロールユニットＥＳＣＵ：このユニットは、命令読み込み処理と、デコード処理を担っている。このユニットの複雑さは多様である。多数の命令を同時に管理することができ、また、アプリケーションでの記述順序によらず、実行準備が整った順に命令を選ぶことができる。このユニットは命令分岐予測に必要数の予測機構を実装している。このユニットはアーキテクチャ内の他のユニットに対する命令としてコマンドを送信する。

２．演算ユニットＳＰＵ：このユニットは命令により記述されている汎用的演算の実行を担っている。このユニットはコントロールユニットＥＳＣＵが複数の命令を同時に管理できるように複数の演算リソースを実装している。

３．メモリユニット：このユニットはプログラムに関連するデータと命令の格納を担っている。メモリユニットはハーバード実行モデルの２階層レベルのキャッシュメモリ階層をベースとし、統合レベルの２つのキャッシュを伴っている。

このメモリユニットは、レベル１のキャッシュメモリであるＬ１　Ｄ－キャッシュ、Ｌ１　Ｉ－キャッシュ、レベル２のキャッシュメモリであるＬ２－キャッシュを備えている。

４．ローディングユニットＬＳＵ：ローディングユニットはメモリに格納されているデータと演算ユニットＳＰＵによって稼動しているユニットとの間にリンクを張る処理を担っている。このリンクは標準プロセッシング部ＳＰＰ内のサイクルあたりの命令処理数の能力に応じて決まる数のポート数のレジスタキューという形となっている。

標準プロセッシング部ＳＰＰと補助プロセッシング部ＡＰＰ間の密接なカップリングを提供するため、標準的な中央プロセッサコアと比較し、コントロールユニットＥＳＣＵとローディングユニットＬＳＵに対して幾つかの特徴が加えられている。

【００６７】

コントロールユニットＥＳＣＵは、補助プロセッシング部ＡＰＰを制御するための追加命令群を備えている。例えば、これらの命令群はクリティカルプロセスの実行を要求するものである。クリティカルプロセスは標準的なメカニズムによっても実行はできるが、追加命令群では命令実行によって新たな命令を要求することがない（例えば、実行後にメモリ空間へのマッピング処理を伴うようなメソッド）。

【００６８】

ローディングユニットＬＳＵは追加のレジスタキューが実装されている。このように標準の汎用レジスタキューに第２のレジスタキューをローディングユニットＬＳＵに追加実装し、ＳＰＰとＡＰＰの２つのサブシステム間でのデータ交換を可能としている。

【００６９】

構造の点から見て、ローディングユニットＬＳＵレベルの補助レジスタＡＲＦ列と汎用レジスタＧＰＲＦ列では違いはない（図８および図９参照）。プロセッサはアドレスの違いにより汎用レジスタと補助レジスタとを区別している。このＳＰＰとＡＰＰの２つのサブシステム間のコミュニケーションモードは特に少量のデータの伝送には適したものといえる。

【００７０】

補助プロセッシング部ＡＰＰはアプリケーション内の特化した及び／又は集中的な演算処理を担っている。補助プロセッシング部ＡＰＰは独自のメモリ空間ＳＭＳを共有し合っている多数の補助演算ユニットＡＰＵ０、ＡＰＵ１、・・・、ＡＰＵＮ－２、ＡＰＵＮ－１が実装されている。補助演算ユニットＡＰＵ０、ＡＰＵ１、・・・、ＡＰＵＮ－２、Ａ

ＰＵＮ－１の数Ｎは特に制限は受けない。同様に、これら演算要素は、相互に区別ができずＡＰＵで単純に定義されるものであるか、同期ロジックまたは非同期ロジックをベースとするものであるかによって違いはない。それゆえ補助プロセッシング部ＡＰＰはＧＡＬＳ（グローバル非同期制御・ローカル同期制御）タイプの構成の実装に非常に便利である。補助プロセッシング部ＡＰＰは典型的には４個から８個の演算要素ＡＰＵを備えている。サイクル内では補助演算ユニットＡＰＵは一つのスレッドのみを実行し、一つのスレッドは一つの補助演算ユニットＡＰＵのみで実行される。

【００７１】

一組のライトプロセス（スレッド）は、コントロールユニットＥＳＣＵから補助プロセッシング部ＡＰＰに含まれている割り当て制御ユニットＡＣＵを介して、次の実行処理を要求する補助演算ユニットＡＰＵに割り当てられる。

【００７２】

スレッドの補助演算ユニットＡＰＵへの物理割り当て処理、実行管理、異なるスレッドに含まれている同期処理は割り当て制御ユニットＡＣＵが担う。

【００７３】

補助プロセッシング部ＡＰＰには、クリティカル入出力コントローラＩＯが実装されている。これらは例えば、高速ＡＤコンバータ、ラジオ周波数インターフェイス、ビデオセンサ等のクリティカル入出力周辺機器にダイレクトにリンクされている。これらＭ個の入出力コントローラＩＯ０からＩＯＭ－１は割り当て制御ユニットＡＣＵにより補助演算ユニットＡＰＵとして取り扱われる。割り当て制御ユニットＡＣＵは入出力コントローラに入出力アクセスを管理できるようにタスクを割り当てなければならない。データは共有メモリ空間ＳＭＳに対して送信され、または共有メモリ空間ＳＭＳから受信される。しかしながら、クリティカル入出力は、例えば、キーボードまたはマウス操作に応じて、システムバスＳＢなどのより汎用的手段によって標準プロセッシング部ＳＰＰを用いることができる。

【００７４】

補助プロセッシング部ＡＰＰは補助演算ユニットで取り扱われるすべてのデータとプログラムを格納する大容量メモリＭＭを含んでいる。このメモリＭＭは、システム（図５ではシステムバスＳＢとして示されている）と特化した演算スレッド間のデータ伝送を行うスレッドを割り当てる割り当て制御ユニットＡＣＵを制御するコントローラＭＭＣを含んでいる。このコントローラＭＭＣは共有メモリ空間ＳＭＳと大容量メモリＭＭ間のデータ伝送にも関連している。

【００７５】

補助演算ユニットＡＰＵは、特定の処理に関して高速処理が可能なように最適化されている。補助演算ユニットＡＰＵは機能に応じてパフォーマンス、フレキシビリティ、コスト、消費電力間においてトレードオフが成り立っている。どの種類の演算ユニットを選択するかはアプリケーション実行環境に強く影響される。

【００７６】

補助演算ユニットは、プログラマブルユニット、リコンフィギュラブルユニット、専用ユニットを備える構成も可能である。

・プログラマブルユニット：このユニットタイプは、組み込み演算に対しては汎用プロセッサコア（ＭＩＰＳ、ＡＲＭ等）または最適化プロセッサコア（ＤＳＰ、ＳＴ２ｘｘ等）に相当するものである。演算に最適化されているため、結果として制御構造がシンプルなものとなっており、例えば分岐予測機構、割り込み処理機構、擬似データ処理機構などが省かれている。これらユニットは浮動小数点演算やベクトル演算などに特化した演算ユニットを構成することができる。

・リコンフィギュラブルユニット：リコンフィギュラブルユニットは演算アクセラレータ同等のものとして用いられる。大規模構造はその処理能力から再構成処理には有利であり、処理能力とフレキシビリティとはトレードオフの関係となる。小規模構成は、非常にフレキシビリティが必要とされる処理または非常に小さいサイズ（１ビットから４ビット）

程度のデータ処理に適している。再構成処理のためには長い時間が必要となるため、プリエンプションによる優先割り当てを避けられるように小規模構成のリソースは別々に管理されることが好ましい。
・専用ユニット：特定のクリティカルな処理に最適化され、コンポーネントに組み込まれている。専用アクセラレータは、プログラマブルまたはリコンフィギュラブルな構成では十分な演算パワーを提供できない場合に、クリティカルな処理を担当することが想定されている。高速暗号処理や入出力ストリーム管理処理はこの専用ユニットを用いる良い対象である。

【００７７】

ユニットタイプにかかわらず、補助演算ユニットＡＰＵは特定の記憶要素として利用することができる。アクセスを高速化するために中間データを記憶する用途や共有メモリ空間のバンド幅を最小化する用途、実行中のプログラム命令を記憶する用途のいずれでも利用することができる。タスク割り当て段階を高速化するために実行中のプログラムをローカルに記憶することができる。

【００７８】

２つのサブシステムＳＰＰおよびＡＰＰは他のシステムへのアクセス手段を共有することができる。通信経路はシステムバスＳＢであり、バスアービターＳＢＡによりシステムバスの共有使用が管理される。例えばメインメモリと入出力コントローラＩＯなど補助プロセッシング部ＡＰＰの２つの要素がシステムバスに対してアクセスを要求することができる。標準プロセッシング部ＳＰＰからみれば、システムバスＳＢへのアクセスは、キャッシュメモリＬ２－キャッシュに大容量メモリまたは周辺機器から渡されるデータと命令がロードされることとなる。プロセッサの一つ以上の要素から同時にアクセス要求が出された場合、バスアービターＳＢＡはシステムバスＳＢを介した通信を保証するためアクセス要求を順番に並べる。

【００７９】

アクセスする各構成要素から要求されるバンド幅はアプリケーション実行環境の機能に応じて可変となっている。このスキームはたくさんの構成要素が同時に大きいバンド幅を要求するようなアプリケーションに適している。システムのすべての構成要素に対して十分なバンド幅を提供するため第２の（場合によっては第３の）システムバスを追加する構成も可能となる。

【００８０】

一構成例を詳しく下記に示す。

【００８１】

標準プロセッシング部ＳＰＰで実行中のシステムソフトウェアによって同じＳＰＰに対してタスクが割り当てられると、ＳＰＰはプログラマブルプロセッサで従来から用いられている方法にてプログラムを実行してゆく。命令処理は特定の命令が出てくるまでサイクルごとに進んでゆく。特定の命令が制御ユニットＥＳＣＵによりデコードされると、割り当て制御ユニットＡＣＵに対してコマンドが生成され、補助演算ユニットの一つで実行されるスレッドが生じる。このような状況下、割り当て制御ユニットＡＣＵは実行管理を担う。この実行モデルはライブラリの最適ファンクションがコールされることにより演算スレッドがアクティブとなるようなプログラマブルモデルに適している。このアプローチは組み込みソフトウェアの分野では既に広く使われており、例えば汎用プロセッサのＡｌｔｉＶｅｃ命令やＭＭＸ命令に相当するものである。

【００８２】

制御ユニットＥＳＣＵによって特定命令が渡されると、標準プロセッシング部ＳＰＰは補助プロセッシング部ＡＰＰによるスレッド管理に干渉することなくプログラムの実行を継続する。このプログラム実行は、スレッドの生成や破棄や補助プロセッシング部ＡＰＰにおいてデータ読み込みなどを生じさせる次の特定命令の処理に至るまで継続される。

【００８３】

実行や割り込みやトラップを伴うトラッピング処理は以下の３つの動作を前提としてい

る。
１．強い同期性：（サブシステムＡＰＰおよびＳＰＰの両方の）すべてのプロセッサの構成要素が同期している。この同期処理には長時間を要するので、部分的な同期手法を採ればマルチタスク処理の実行環境におけるペナルティを低減させることができる。実行環境の書き換えを高速化するためにビクティムキャッシュなどを用いて大容量メモリへ書き込む内容はしばらくの間保持される。
２．弱い同期性：標準プロセッシング部ＳＰＰに関する実行環境のみが同期しているものである。この状態では補助プロセッシングユニットＡＰＵによりアクティブとなっているファンクションは補助プロセッシング部ＡＰＰにおいて維持されている。割り当て制御ユニットＡＣＵは補助リソースの割り当てのみを担う。ＡＰＰの自律的処理はスレッドが標準プロセッシング部ＳＰＰのタスクが生成したデータをコールしない限り継続される。
３．ローカル同期性：トラッピングが例えば０の除算など補助演算ユニットＡＰＵのイベントコールを伴う場合、ユニットはトラッピングのみを管理し、他のプロセッサからは独立した同期性をとる。
【００８４】
割り当て制御ユニットＡＣＵは制御ユニットＥＳＣＵからの専用命令の処理を担う。制御ユニットＥＳＣＵが割り当て制御ユニットＡＣＵと連動するカップリング処理の詳細を図６に示すモデルによって説明する。
【００８５】
図６は制御ユニットＥＳＣＵを示しており、標準プロセッシング部ＳＰＰのローディングユニットＬＳＵ、演算ユニットＳＰＵ、メモリユニットＬ１　Ｉ－キャッシュと共に示されている。図６には補助プロセッシング部ＡＰＰの割り当て制御ユニットＡＣＵも示されている。
【００８６】
標準プロセッシング部ＳＰＰの標準命令は制御ユニットＥＳＣＵのリードステージ、デコードステージにおいてそれぞれリード、デコードされ、ローディングユニットＬＳＵと演算ユニットＳＰＵが制御される。逆に、専用命令の場合は制御ユニットＥＳＣＵは割り当て制御ユニットＡＣＵのコマンドの流れとしてリダイレクトする。
【００８７】
これら専用命令は以下の異なる種類の命令に関連付けることもできる。
－スレッドの生成／破棄すること
－タスクに関連づけてスレッドを破棄すること
－メインメモリＭＭからシステムバスＳＢへまたは逆方向へデータを転送すること
－サブシステムＳＰＰとＡＰＰ間でデータを転送すること
【００８８】
標準プロセッシング部ＳＰＰへのタスクの割り当てにおいて、システムソフトウェアは補助演算ユニットＡＰＵへのスレッドの仮想割り当てを行う。割り当て制御ユニットＡＣＵは最適な割り当てを決定するパラメタをすべて勘案して物理的な割り当てを行う。割り当てとは別に、割り当て制御ユニットＡＣＵはスレッド間の同期とクリティカルな共有リソースのアクセスも制御する。この割り当て制御ユニットＡＣＵは例えばプリエンプション管理またはタスクのアップデートリスト管理などのシステムソフトウェアのサポートも担っている。
【００８９】
これらファンクションにより割り当て制御ユニットＡＣＵは補助プロセッシング部ＡＰＰにおいて実行されている各々のスレッドの実行環境が調整される。弱い同期性を持っている場合、割り当て制御ユニットＡＣＵのみがスレッドの展開を担っている。結果として、タスクが標準プロセッシング部ＳＰＰに対して再割り当てされると、演算が進んでいるスレッドに対してその旨を通知する必要がある。これにより、標準プロセッシング部ＳＰＰではターミネートされずに補助プロセッシング部ＡＰＰにおいて実行されているプロセスのスレッドが再アクティブ化されてしまうことがなくなる。割り当て制御ユニットＡＣ

Ｕのローカル実行環境の利用管理によりタスクがプロセッサに割り当てられる状態が維持されることが確保される。このことは、標準プロセッシング部ＳＰＰにおいてタスクが正常に実行されない場合になおさらである。

【００９０】

基本となるサービスのもと、割り当て制御ユニットＡＣＵはアプリケーションドメインに関連したファンクションを担っている。これらファンクションはダイナミック消費電力管理、フォールト管理、クライシスモード管理などである。

【００９１】

補助演算ユニットＡＰＵで取り扱われる全てのデータは共有メモリ空間ＳＭＳに格納されている。共有メモリ空間ＳＭＳはマルチメモリリソースと同じ空間内にあるすべてのリソースをユニット化して内部接続するネットワークを備えている。メモリ空間コントローラＭＳＣは演算リソースとメモリリソース間のリンク構築を担っている。割り当て制御ユニットＡＣＵは、補助演算ユニットＡＰＵによって取り扱われる共有メモリ空間の仮想アドレス（変数名と変数の位置、例えばイメージ名とピクセルインデックス）と、メモリリソースで使用されることが宣言されている物理アドレスとのリンクに関する情報を供給する。図７はＡＰＵｐと表示されているプロデューサ側の補助演算ユニットＡＰＵと、ＡＰＵｃと表示されているコンシューマー側の補助演算ユニットＡＰＵとの間のデータ伝送のデータアクセス機構を示している。この補助プロセッシング部ＡＰＰ内でのデータアクセス機構は２つのステップに分けることができ、図７中のサークル１、サークル２として示されている。

【００９２】

データアクセスの第１のフェーズは補助演算ユニットＡＰＵが初めて変数にアクセスしたときに用いられる。この状態ではまだデータとメモリ間のリンクはない。メモリ情報を得るため、まず補助演算ユニットＡＰＵは割り当て制御ユニットＡＣＵに問い合わせる。ＡＣＵは変数名と関連付けることによりデータアクセスが実行できるように、メモリ空間管理ユニットＭＳＭＵと協働する。アクセスするデータがＡＣＵにより特定されると、管理ユニットＭＳＭＵは変数を格納しているメモリの特定情報を送信する。逆に、補助演算ユニットＡＰＵが参照情報のないデータを書き込もうとすると（例えば変数を最初に書き込む場合）、管理ユニットＭＳＭＵは、共有メモリ空間ＳＭＳ中の利用できるメモリから書き込むメモリを選んで割り当てる制御を行う。一度メモリが割り当てられるとメモリユニットは記憶されているメモリの変数名によって関連付けられ、更新されるたびにメモリの特定情報が補助演算ユニットＡＰＵに対して送信される。補助演算ユニットＡＰＵが参照情報のないデータを読み出そうとすると（例えば変数を最初に読み出す場合）、管理ユニットＭＳＭＵが取り出しを担い（メインメモリ制御部ＭＭＣと関連付けられる）、共有メモリ空間ＳＭＳ中の利用できるメモリからメモリを選んで割り当てる制御を行う。パーマネントデータが書き込まれると（最終結果として）、変数名が対応しているメモリのユニットのエントリが開放され、大容量メモリに書き戻されるべきデータのリクエストがメインメモリ制御部ＭＭＣに対して送信される。

【００９３】

データアクセスの第２フェーズでは補助演算ユニットＡＰＵとデータを含むメモリの間にリンクを確立することがシステマティックに行われる。補助演算ユニットＡＰＵはアクセスしたいメモリを知っているので、共有メモリ空間ＳＭＳに対して必要となるデータアドレスとメモリ制御信号とを同一サイクルタイム内に送る。メモリ空間制御部ＭＳＣはこれら信号の適切なメモリへのルーティングを担う（その結果データを返す）。この動作はメモリリソースの内部接続構成のタイプにより異なる。図７に示したモデルにおけるチップ内のネットワーク実行環境では、このメモリ空間制御部ＭＳＣによるデータのパケット化とは、例えば、ネットワークのルーティング情報をデータを加えることとなる。クロスバータイプのポイントツーポイントネットワークでは、メモリ空間制御部ＭＳＣはハードウェアのパスの形成を担っている。

【００９４】

補助演算ユニットＡＰＵのレベルのデータアドレスの管理は、例えば、データストリームやバーストコールを管理できる適切な能力を備えた専用の入出力ユニットにより取り扱われる。データ管理を担う構成要素は補助演算ユニットＡＰＵにより取り扱われるデータのリカバリも担うこととなる。データへのアクセスができない場合またはアクセスが阻害された場合、制御モジュールは不安定なデータへの処理をさせないために補助演算ユニットＡＰＵにおけるスレッドの実行をフリーズする。

【００９５】

アプリケーションドメインと共有メモリ空間ＳＭＳの処理結果において、データにアクセスするリクエストNb_{access}の数がメモリのポートNb_{ports}の数またはネットワークのノード数よりも大きい場合には実行中にコンフリクトが発生しうる。この阻害要因は発生しうることを考慮しておかなければならず、システムのディメンジョンを決定し、メモリアクセスタイムT_{access}を知り、以下の式（１）を考慮しなければならない。

【００９６】

【数１】

$$Tacces = Tmem \times Sup\left(\frac{Nb_{Access}}{Nb_{Port}}\right) \qquad (1)$$

式中のNb_{access}はアクセス数を表わし、
Nb_{port}は共有メモリ空間ＳＭＳのポート数又はネットワークノード数を表わし、
T_{mem}は最小メモリアクセスタイムを表わしている。

【００９７】

オーバーヘッドを最小化するため、異なるメモリバンクのデータデカップリングにより仮想のメモリポート数を増やすことができ、同時マルチアクセスが可能となる。データ書き込み時間とデータ読み出し時間の兼ね合いはシステム全体のパフォーマンスを最適化するようにユーザが決定する。コンフリクトが発生している間は、特別なアクセスプライオリティポリシーは必要ない。メモリアクセス時間を最大限確保するなどのコンフリクト処理の機能を提供するためには単純な先入れ先出しタイプのソリューションで良い。

【００９８】

データが演算要素自体ではなくメモリ構造に依存しているので採用されているメモリ構造は重要である。実行環境の変更は演算リソース間のデータ伝送を必要としないためプリエンプションは即座に実行することができる。

【００９９】

小容量データは、図５に示すように補助プロセッシング部ＡＰＰと標準プロセッシング部ＳＰＰ間で交換できる。データ伝送はソースプログラムにおいて以下の情報を指定した命令として直接書き込むことができる。

－補助プロセッシング部ＡＰＰから標準プロセッシング部ＳＰＰ間へのデータ伝送またはその逆方向のデータ伝送
－標準プロセッシング部におけるターゲットレジスタ
－補助プロセッシング部ＡＰＰにおけるターゲットスレッド
－スレッドのデータ

【０１００】

図８に示した例のようにロード命令Ｒｘ，Ｔｙ，Ｒｚの読み込み処理は、標準プロセッシング部ＳＰＰのレジスタＲｘに補助プロセッシング部ＡＰＰで実行されているスレッドＴｙの変数Ｒｚがロードされることで実行される。制御ユニットＥＳＣＵはこの命令をデコードし、以下の３つのコマンドを生成する。

１．Ｒｅａｄ（Ｒｚ）：このコマンドは補助演算ユニットＡＰＵの変数Ｒｚの読み込みを行うものである。

２．Ｓｅａｒｃｈ（Ｔｙ）：このコマンドは割り当て制御ユニットＡＣＵに対してどの補助演算ユニットＡＰＵがスレッドＴｙを実行中であるかの識別子を送る。この識別子は、

補助演算ユニットＡＰＵの割り当て制御ユニットＡＣＵ内でアクティブなスレッドに関連付けられたトランスレーションルックアサイドバッファ（ＴＬＢ）と呼ばれるページのテーブルという形で示される。もしＴＬＢが補助演算ユニットＡＰＵの識別子を送り返して来なかった場合、標準プロセッシング部ＳＰＰが処理待ちの実行中のスレッドやタスクが存在しないことを意味する。スレッドが実行されている場合、ＴＬＢは当該スレッドを実行している補助演算ユニットＡＰＵの識別子を送り返す。この識別子は標準プロセッシング部ＳＰＰが補助プロセッシング部ＡＰＰの補助レジスタキューに送るべきデータを選択するために利用される。この識別子は共有レジスタキューＳＲＦのリードデータを有効化するために補助演算ユニットＡＰＵにおいても利用される。

３．Ｗｒｉｔｅ（Ｒｘ）：このコマンドは、補助プロセッシング部ＡＰＰから返された書き込みデータを補助レジスタキューのレジスタＲｘへ書き込むものである。

【０１０１】

コマンドＡＰＵｉを補助演算ユニットＡＰＵ０、ＡＰＵ１、ＡＰＵＮ－２、ＡＰＵＮ－１に送ることはオプションであり、伝送モードを邪魔することなく除去することができ、ユーティリティのないレジスタキューにアクセスすることを防ぐことができる。

【０１０２】

二重構造のメカニズムによって標準プロセッシング部ＳＰＰから補助プロセッシング部ＡＰＰにデータを伝送することができる。この機構は図９に示されており、図８と似通っているが、ＳＴＯＲＥ Ｒｘ、Ｔｙ，Ｒｚ，Ｗｒｉｔｅ（Ｒｚ）、Ｒｅａｄ（Ｒｘ）という命令が、ＬＯＡＤ Ｒｘ，Ｔｙ，Ｒｚ，Ｒｅａｄ（Ｒｚ）、Ｗｒｉｔｅ（Ｒｘ）命令の代わりにある。

【０１０３】

下記に説明する実施例では、システムへのアクセスがメインメモリMMを介したものとなっている。メインメモリ制御部ＭＭＣ、割り当て制御ユニット（ＡＣＵ）、メモリ空間制御部ＭＳＣが、使用される通信タイプに応じて、通信制御に関与している。メインメモリMMは４つの通信タイプに関与している。

１．システムバスＳＢからメインメモリMMへの通信：データ通信の第１のタイプは、データをシステム外部から補助プロセッシング部ＡＰＰのメインメモリMMへ取り入れることである。この伝送は制御ユニットＥＳＣＵの特定命令のデコード後に発生しうる。特定命令は割り当て制御ユニットＡＣＵによりメインメモリ制御部ＭＭＣに対して割り当てられるデータ伝送処理を発生させる。後者はダイレクトメモリアクセス（ＤＭＡ）制御部と同様である。同時に、メインメモリ制御部ＭＭＣはロードされているデータとメインメモリMMにおけるアドレスのリンクを確立できるようにテーブルを埋める。

２．メインメモリMMからシステムバスＳＢへの通信：対称的に、通信されるデータはメインメモリMMから制御ユニットＥＳＣＵの特定命令により識別されるデータ伝送の到着を知らせるシステムリマインダーデータである。メインメモリMMから送信することはデータが最終結果であると通信内容のテーブル内のエントリが破棄される。制御ユニットＥＳＣＵによってデコードされた特定命令は破棄伝送か破棄しない伝送かの区別を決める。

３．メインメモリMMから共有メモリ空間ＳＭＳへの通信：補助演算ユニットＡＰＵが共有メモリ空間ＳＭＳに存在しないデータにアクセスしようとすると、共有メモリ空間ＳＭＳにデータをルーティングするため伝送要求が割り当て制御ユニットＡＣＵにより制御部ＭＭＣに送られる。補助演算ユニットＡＰＵは伝送処理の間、ブロックされる。

４．共有メモリ空間ＳＭＳからメインメモリMMへの通信：この伝送は補助プロセッシング部ＡＰＰの共有メモリ空間ＳＭＳに再リードされない最終結果の書き込みにおいて、補助演算ユニットＡＰＵからの特定データの伝送である。これらの伝送処理は実行環境の格納において強い同期性の状態でも実行できる。例えば、共有メモリ空間ＳＭＳは割り当て制御ユニットＡＣＵを介してリクエストデータをメインメモリ制御部ＭＭＣに対して送る。

【０１０４】

ターゲットアプリケーション空間の機能拡充のため、使用用途が限定されていない大容

量メモリの実装が可能である。この状態は大容量メモリＭＭが他のメモリリソースと同じ共有メモリ空間ＳＭＳに実装されている環境と同様である。このような環境下では、データが共有メモリ空間ＳＭＳとシステムバスの間で直接交換される。この交換は通信スレッドという形で管理され、メインメモリ制御部ＭＭＣの専用ユニットのどの補助演算ユニットＡＰＵにおいても実行できるものである。

【０１０５】

本発明のメソッドと拡張実装可能なアーキテクチャは補助演算ユニットＡＰＵの実装数をサポートできる。

【０１０６】

実際には、アーキテクチャのパフォーマンスは補助演算ＡＰＵの数が大きすぎると、例えば、数百個のオーダーの数であると、低下してしまう。

【０１０７】

この問題を解決する手段の一つは、共有メモリマルチプロセッサシステムのアーキテクチャが適用されることである。このような実施例は図１０に示されており、本発明が適用された２つのプロセッサが搭載されている例を示している。図５を参照して示したような上記の標準プロセッシング部ＳＰＰと補助プロセッシング部ＡＰＰがカップリングされているものと同じ構成のコアを持った多数のプロセッサが搭載されたものも可能である。

【０１０８】

共有メモリマルチプロセッサシステムにおいて、図１０に示すように、専用バスを介したコア間で共有されているシステムバスアービターＳＢＡや補助プロセッシング部ＡＰＰの大容量メモリＭＭや高速入出力コントローラなどの構成要素を共有することは有利である。

【０１０９】

要するに、本発明は、組み込みアーキテクチャでのスレッド制御やスレッド割り当てのデバイスやメソッドに関するものであり、マルチプルプロセッシングリソースに実装され、リアルタイムで集中特化したマルチタスク演算およびマルチストリーム演算に適したものである。

【０１１０】

さらに、下記の構成要素を備えたリアルタイムパラレル演算アーキテクチャにも適用され得るものである。

－クリティカルではないタスク処理とシステムソフトウェアサポートの実行を担う中央プロセッサコアＳＰＰ

－プログラマブルでリコンフィギュラブルまたは特定処理の高速処理に最適化された補助演算ユニットＡＰＵ

－補助演算ユニットＡＰＵで内部ネットワークを介して共有されるメモリ空間ＳＭＳ

－補助演算ユニットＡＰＵｉによる集中特化した処理の並列処理の実行を管理する補助リソースを制御し割り当てるユニットＡＣＵ

【０１１１】

さらに、特に、様々な補助演算ユニットＡＰＵ間の通信、補助演算ユニットＡＰＵと中央プロセッサコアＳＰＰ間の通信が、共有メモリ空間ＳＭＳまたは内部ネットワークを介して行われるものである。

【０１１２】

タスク割り当てとタスク処理の手法において、中央プロセッサコアＳＰＰで実行される制御タスクが、補助演算ユニットＡＰＵで実行される集中特化した演算タスクとは分離されている。割り当て制御ユニットＡＣＵは集中特化した演算タスクを様々な補助演算ユニットＡＰＵに対して割り当てる管理を並列処理にて行う。この補助制御部ＡＣＵは、補助演算ユニットＡＰＵが、中央プロセッサコアＳＰＰで実行されているタスクとは異なるスレッドの処理ができるという、いわゆる弱い同期機構により実装されている。このシステム状態はユニークな実行環境で表わされており、ノイマン型アーキテクチャとは異なるものとなっている。クリティカル入出力は、補助演算ユニットＡＰＵによって共有メモリ空

間に直接リンクされている。このアーキテクチャおよびこの割り当て手法は、データローディング時間の低減や、異なアプリケーションへの適合など、リアルタイムマルチタスクプロセッシングの最適化を実現できる。

【図面の簡単な説明】

【０１１３】

【図１】図１Ａおよび１Ｂは、それぞれ汎用的なＳＭＴアーキテクチャモデルおよび操作処理例を表わす図である。

【図２】図２Ａおよび２Ｂは、それぞれ汎用的なＣＭＰアーキテクチャモデルおよび操作処理例を表わす図である。

【図３】図３Ａおよび３Ｂは、それぞれ汎用的なＣＭＴアーキテクチャモデルおよび操作処理例を表わす図である。

【図４】図４は、システム動作をアプリケーション、タスク、さらに命令（スレッド）の流れに分解して表わした図である。

【図５】図５は、本発明のプロセッサのアーキテクチャの主要構成要素を示すブロック図である。

【図６】図６は、補助プロセッシング部と標準プロセッシング部で構築されるメカニズムを示すブロック図である。

【図７】図７は、補助プロセッシング部におけるデータアクセス処理のメカニズムを示す図である。

【図８】図８は、補助プロセッシング部と標準プロセッシング部の間でのデータ転送処理のメカニズムを示す図である。

【図９】図９は、標準プロセッシング部と補助プロセッシング部の間でのデータ転送処理のメカニズムを示す図である。

【図１０】図１０は、本発明の実施例である共有メモリマルチプロセッサシステムの構成例を示すブロック図である。

【図１】

FIG.1A

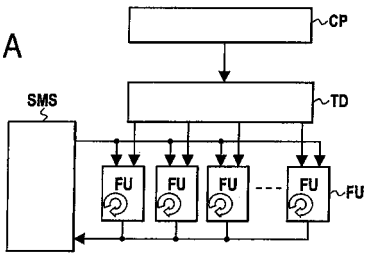
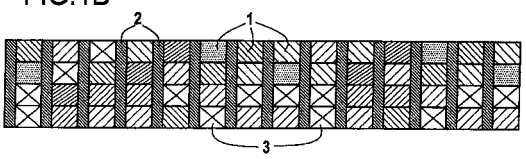


FIG.1B



【図２】

FIG.2A

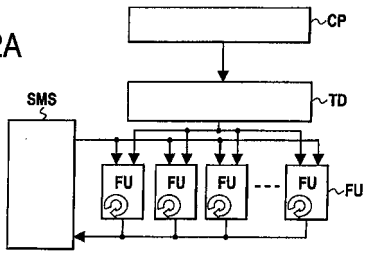
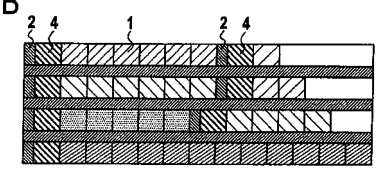


FIG.2B



【図３】

FIG.3A

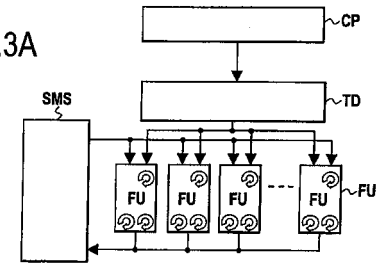
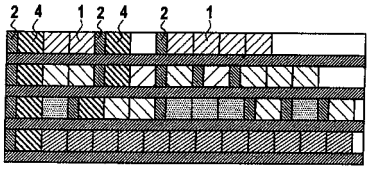


FIG.3B



【図４】

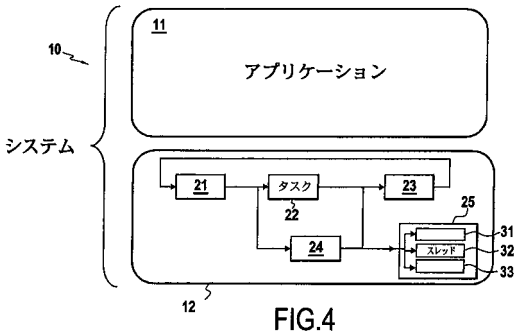


FIG.4

【図５】

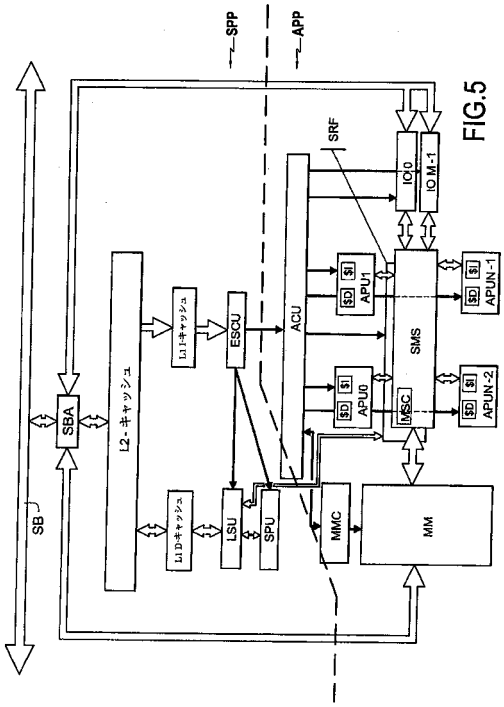


FIG.5

【図６】

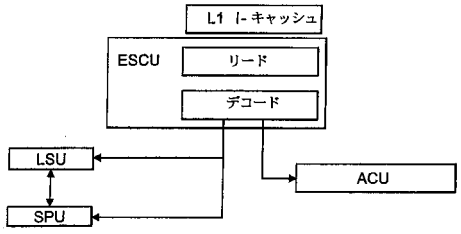


FIG.6

【図７】

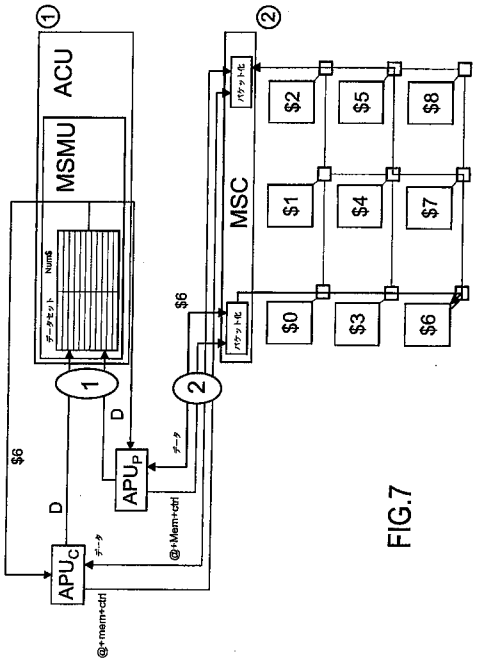


FIG.7

【図８】

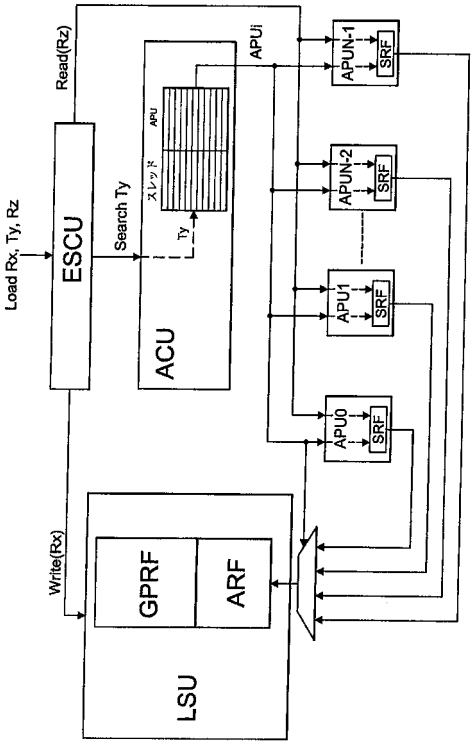


FIG.8

【図９】

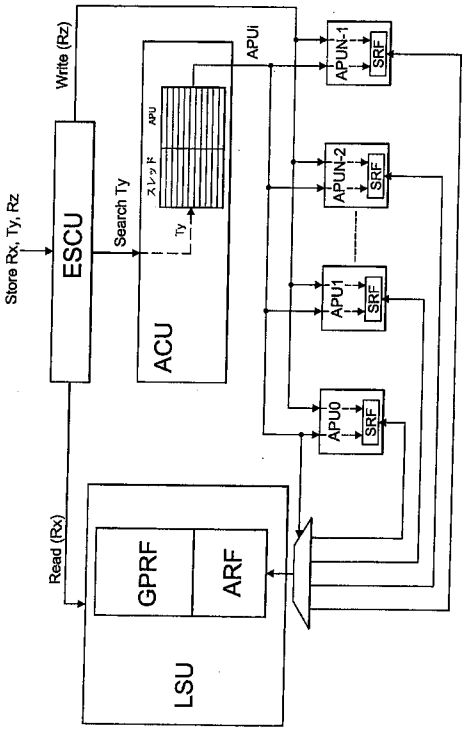


FIG.9

【図１０】

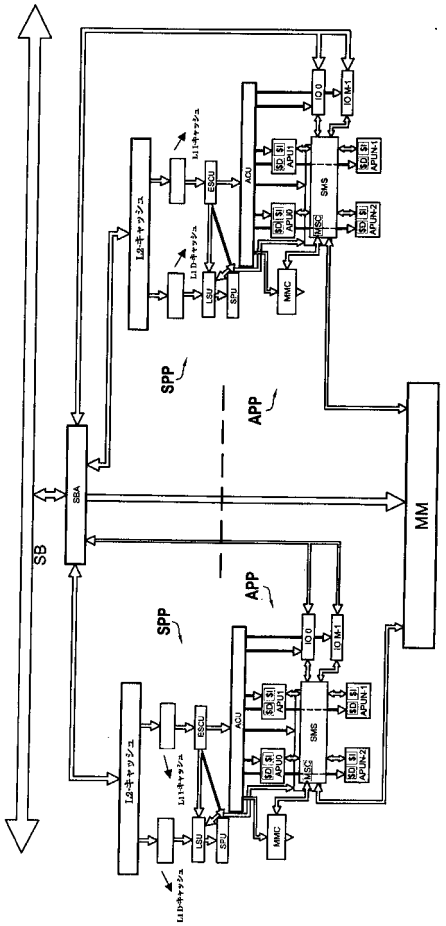


FIG.10

フロントページの続き

(72)発明者 ダヴィ，ラファエル
フランス、エフ－９１４４０ ビュル シュル イヴェット、アヴニュ シャルル ドゥ ゴール １７
(72)発明者 ダヴィ，ヴァンサン
フランス、エフ－７８０００ ヴェルサイユ、リュ デ ションティエ ４７
(72)発明者 ヴァントルー，ニコラ
フランス、エフ－９１４４０ ビュル シュル イヴェット、リュ シャルル ドゥ ゴール ２８
(72)発明者 コレット，ティエリー
フランス、エフ－９１１２０ パレゾー、リュ マルソー １２４

審査官 川崎 優

(56)参考文献 Wolf,W., The Future of Multiprocessor Systems-on-Chips, Proc. of the 41st annual Design Automation Conf., ２００４年 7月１１日, P.681-685

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／４６－５４