様式 C-19

科学研究費補助金研究成果報告書

平成２２年　４月２６日現在

研究種目：若手研究（B）
研究期間：2007～2009
課題番号：１９７００４７
研究課題名（和文）　プロセッサ多コア時代のプロセッサアーキテクチャに関する研究

研究課題名（英文）　Research of a Processor Architecture for the Many-Many Core Processor

研究代表者
松崎　隆哲（MATSUZAKI TAKANORI）
近畿大学・産業理工学部・講師
研究者番号：20363385

研究成果の概要（和文）：本研究では，多数のプロセッサコアを持つプロセッサにおける，スレッド管理機構について研究を行った．イベント駆動型プロセッサアーキテクチャのタスク管理機構に関して，イベント駆動型プロセッサにおいてスレッド実行モデルを変更することで，処理時間，メモリ利用量，スレッド数にどのような変化が起こるかに関して性能評価を行った．タスク管理機構の評価によって，Fuce プロセッサのスレッド実行モデルを変更することで，メモリ資源が不足する問題をある程度解決できるということが判明した．

研究成果の概要（英文）：We research the thread control mechanisms of the Fuce processor and evaluate concurrency performance of the Fuce processor which we described in VHDL. As a result, we understood that the processor has concurrency capability and can control parallelism of thread execution when there is sufficient thread level parallelism.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２００７年度	1,000,000	0	1,000,000
２００８年度	1,200,000	360,000	1,560,000
２００９年度	1,000,000	300,000	1,300,000
年度			
年度			
総　計	3,200,000	660,000	1,860,000

研究分野：計算機アーキテクチャ
科研費の分科・情報学・細目：計算機システム・ネットワーク
キーワード：イベント駆動プロセッサ，マルチコアプロセッサ，タスク管理，細粒度マルチスレッド処理，Fuce プロセッサ，スレッド管理機構，FPGA，オペレーティングシステム

１．研究開始当初の背景

近年のプロセッサアーキテクチャの進歩はとどまるところを知らず，1 チップ上に多数のプロセッサコアを搭載した Many-Many コアプロセッサが研究分野において提唱されつつある．プロセッサが多数のプロセッサコアを持つ傾向は研究分野だけでなく，Intel がクアッドコアの商用プロセッサを実際の製品として 2006 年末には実現化しようとしている．このように，容易に入手可能な汎用

プロセッサにおいても，今後は多数のコアが搭載されたプロセッサが一般的になりつつある．そのため，これまでは非常に高価であった多数のプロセッサを持つ並列分散コンピュータシステムが，今後は汎用プロセッサを用いて安価に実現することができる可能性が出てきた．

これまでのプロセッサアーキテクチャの研究においては，多数のプロセッサコアを搭載することに主眼があるため，Many-Many コアプロセッサ上でどのようにソフトウェアを動作させるのかについての視点が欠けている．プロセッサ技術の進歩により，2010 年には 1 チップ上に 100 以上のプロセッサコアが搭載できると予想されているが，そのプロセッサをどのように使うのかどのようなアプリケーションを実行するのかの観点についてはほとんど検討されてない．

また，応用アプリケーションに着目すると，マルチスレッドやグリッド・クラスタなどのミドルウェア技術によって並列分散処理を行うといった現在利用されている既存の技術を応用することで対応できると考えることができる．しかしながら，基本ソフトウェアであるオペレーティングシステム特にカーネルに関してはほとんど着目されていない．そのため，単一チップに多数のプロセッサコアが載る時代が到来した場合に，計算機の資源を管理するカーネルウェアとしてのオペレーティングシステムを効率よく実現できるかどうかは不明確である．

２．研究の目的

本研究では Many-Many コアプロセッサ時代におけるプロセッサアーキテクチャおよびオペレーティングシステム(カーネルウェア) の制御方法について検討を行う．具体的には，単一チップ上に多数のプロセッサのコアが載るようになった場合，その上で動作する基本ソフトウェアであるオペレーティングシステムをどのように実行することが良いのかを検討する．

特に，ソフトウェア(オペレーティングシステム) 側で，多数のプロセッサコア上の協調動作(制御管理) を行うのか，プロセッサ側にタスクの並列実行を支援するための機構を用意する方が良いのかについて検討を行う．さらには，多数のプロセッサコアが利用可能となった際に，タスクの並列実行の管理を効果的に行うことができるプロセッサアーキテクチャについて，以下の観点から検討を行う．

- これまで通りのノイマン型プロセッサアーキテクチャ上でソフトウェアによるタスク管理を行う
- ノイマン型プロセッサにタスクの並列実行を支援する機構を別途追加する
- データフローの考えを取り入れ，ハードウェアによるタスク制御を実現

以上のことを研究することで，本研究では Many-Many コアプロセッサ時代におけるタスク制御手法についての知見を得ることを目的とする．

３．研究の方法

本研究では，単一チップ上に多数のプロセッサコアが搭載可能となった場合に，オペレーティングシステム(カーネルウェア) の動作を念頭に置いたプロセッサアーキテクチャはどのような構成を取るべきかについて，これまで研究を行ってきたプロセッサ設計の経験と成果をフィードバックして，多数コアを持つプロセッサアーキテクチャに関する研究を行った．

具体的には，以下のような内容について研究を行った．

(1) イベント駆動型プロセッサのアーキテクチャに関する研究

イベント駆動型プロセッサにオペレーティングシステムを実装するために，スレッドベースの I/O 処理手法について検討を行い，Fuce プロセッサにオペレーティングシステム向けタスク管理機構の追加を行った．

イベント駆動型プロセッサにおいて，オペレーティングシステムを実行するためには，スレッドの実行制御が必要になった．そこで，プロセッサのタスク管理機構を改良することで，スレッド実行モデルを変更できるようにした．この実行モデル変更によって，処理時間，メモリ利用量，スレッド数にどのような変化が起こるかに関して性能評価を行った．

(2) マルチコアプロセッサにおけるオペレーティングシステム

マルチコアプロセッサにおけるオペレーティングシステムに関する検討として，一般的な UNIX 環境として Linux のカーネルソースコードを調査し，マルチコアプロセッサでどのようにオペレーティングシステムを実現しているかについて調査を行い，マルチコア環境におけるオペレーティングシステムの効率的な実行方法について検討を行った．オペレーティングシステムを動作する環境として，コア数を自由に変更可能なプロセッサシミュレータについて調査し，実際にデータをとることに利用可能かどうか検討を行った．

(3) 並列処理プログラムとしての独立成分分析

マルチコアプロセッサにおける応用アプリケーションとして，処理の並列化が期待できる「独立成分分析を用いた音声処理システム」に関して並列処理の可能性を検討した．独立成分分析をハードウェアに実装するこ

とで，ソフトウェアによる並列アプローチではなく，ハードウェア化を利用した並列化の検討を通して，処理の並列化が実現できないか検討を行った．

４．研究成果

本研究では，大別して，次の 3 つを行った．

(1) イベント駆動型プロセッサのアーキテクチャ
(2) マルチコアプロセッサにおけるオペレーティングシステム
(3) 並列処理プログラムとしての独立成分分析

(1)では，イベント駆動型プロセッサアーキテクチャに関する検討として，イベント駆動型プロセッサにおける I/O 処理手法，タスク管理機構を用いたスレッド実行モデルの変更についての研究を行った．

イベント駆動型プロセッサでオペレーティングシステムを実現するために，ノイマン型プロセッサで広く用いられている割り込みによる I/O 処理でなく，スレッドを用いて I/O 処理を行うタスク管理機構について研究を行った．このタスク管理機構を用いることで，Fuce プロセッサは割り込みによって，処理を中断することなく，I/O 処理を行うことができるようになり，効率の良いスレッド実行が実現できた．

また，I/O 処理を割り込みベースで行った場合，多数のコアを持つノイマン型プロセッサ場合においてはどのコアで割り込み処理(オペレーティングシステムの実行)を行うのかという問題があるが，イベント駆動方式で割り込み処理を行った場合はノイマン型プロセッサにおける問題が生じない事を確認した．その検討結果を受け，イベント駆動型プロセッサではどの程度の処理内容が必要となるか試算を行い，ノイマン型プロセッサにおける I/O 処理に必要な処理量との比較検討を行った．処理量の比較検討により，イベント駆動方式を用いた I/O 処理では，実行中のスレッド(タスク)を中断することなく実現することができ，タスク管理の面だけでなくタスク切り替え等を含んだ実行処理時間の点からも効率的である事を確認した．さらに，Fuce プロセッサにオペレーティングシステム向けタスク管理を追加するために必要となるスレッド管理機構(タスク管理機構)を検討し，その設計を行った．これらの成果は紀要と学会で発表した．

この研究を通じて，イベント駆動型プロセッサに，オペレーティングシステムを実装するためには，スレッドの実行モデルを切り替える機能を持つことが必要になるという知見を得られたため，イベント駆動型プロセッサアーキテクチャのタスク管理機構に関する研究を行った．イベント駆動型プロセッサにおいてスレッド実行モデルを変更することで，処理時間，メモリ利用量，スレッド数にどのような変化が起こるかに関して性能評価を行った．一般的に，スレッド並列処理を行う場合，同時に実行するスレッド数に制限を設けないと，スレッド数が膨大となりメモリ利用量が膨大になる恐れがある．特に，Fuce プロセッサはスレッド実行モデルとして幅優先モデルを採用していることから，プログラムによって，スレッド数が増えすぎないための工夫が必要であった．そこで，タスク管理機構内部のスレッド実行順を切り替えることで，同時に実行することが可能となるスレッド数を減らすことで，同時に実行するスレッド数を制限する機構を Fuce プロセッサのタスク管理機構に追加した．追加したタスク管理機構の評価を，HDL を用いた評価と FPGA 実験ボード上への実装を行うことで行い，Fuce プロセッサのスレッド実行モデルを変更することで，メモリ資源が不足する問題をある程度解決できるということが判明した．

本研究では、幅優先モデル、深さ優先モデル、ハイブリッドモデルにおいて実プログラムを動作させ、インスタンス数、スレッド数、スレッドキューの長さを測定する実験を行い、測定結果を基に各スレッド実行モデルに対して評価を行った。評価結果から、深さ優先モデルにおいてはスレッド生成数が不足する可能性があり、プログラムによっては Fuce プロセッサの性能を引き出せない可能性が確認できた。また、ハイブリッドモデルにおいては、適正な閾値を設定することで Fuce プロセッサの性能を引き出せる十分なスレッドを生成でき、メモリの利用量を抑えることができると判明した。

これらの成果については，測定データをまとめており，7 月の国際会議で発表する予定である．

なお，当初計画していた，イベント駆動型プロセッサにおけるオペレーティングの実装や Fuce プロセッサの FPGA 実験ボード上への実装などといった項目については，十分に行うことができなかったが，本研究を通して，これらを行う準備は整えた．これらのやり残した項目については，今後，研究を進めていく予定である．

(2)では，マルチコアプロセッサにオペレーティングシステムを実装するために必要となる技術の調査として Linux カーネルのタスク管理やプロセッサ内部の動作データを測定するためのプロセッサシミュレータについて調査を行った．

既存のオペレーティングシステムである，Linux カーネルのタスク管理機構を調査することで，マルチコアプロセッサ上にオペレー

ティングシステムを実装する際に必要となる機能を確認したため，イベント駆動型プロセッサのタスク管理に必要な機能を取り入れた．また，ノイマン型コンピュータにもとづくマルチコアプロセッサのシミュレーション環境として，コア数を自由に変更可能なプロセッサシミュレータについて調査し，実際にデータをとることに利用可能かどうか検討を行ったが，有効なデータを得ることは困難であると予想されたため，オペレーティングシステムに関する研究は，イベント駆動型プロセッサをベースに行うことにした．

なお，当初計画していたタスク管理機構を持つオペレーティングシステムについては，検討を行うことができなかったが，本研究を通して，イベント駆動型プロセッサのオペレーティングシステムを実装するために必要となる知見が得られた．今後，オペレーティングシステム実装に関する研究を進めていく予定である．

(3)では，マルチコアプロセッサで実行する実アプリケーションとして，独立成分分析に関して調査を行った．並列処理システムにおいて，処理対象となるアプリケーションの並列性能は重要であることから，実行するアプリケーションには並列性が要求されている．独立成分分析を用いた音声処理は，その処理内容に十分な並列性を持っており，並列性を持つスレッドを抽出できるであろうと確認した．

並列処理スレッドプログラムとしての独立成分分析では，現状の逐次型プログラムでは多大な時間が必要となるブラインド信号分離を実現する独立成分分析について，多数のスレッドで並列化することによる高速化の可能性を検討した．この検討は，独立成分分析を用いた音声処理システムをハードウェアに実装することで，ソフトウェアによる並列アプローチではなく，ハードウェア化を利用した並列化の検討を通して行った．これは，これまでとは異なった観点から，処理のスレッド並列化が実現できないか検討を行うことで，並列スレッドによる独立成分分析の実装を検討した．今後，この結果をもとにして，並列処理スレッドプログラムの検討を行う予定である．

これらの検討については，まだ手をつけ始めた段階であり，独立成分分析をハードウェアで実現するために必要な技術要素の調査を行っている段階である．しかしながら，少しずつ成果が得られて始めており，今後，独立成分分析を並列実装することで，高速化を図る研究を進めていく予定である．

５．主な発表論文等
（研究代表者，研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 1 件）

① 泉雅昭，松崎隆哲，日下部茂，谷口秀夫，長谷川隆三，雨宮真人，Fuceプロセッサ上での継続モデルによる並列I/O処理，九州大学大学院システム情報科学紀要，査読有，第 13 巻第 2 号，2009，81－88

〔学会発表〕（計 2 件）

① 津城忠史，松崎隆哲，音声処理システムのハードウェア化に関する検討，情報処理学会 火の国情報シンポジウム，2009 年 3 月 13 日，九州産業大学
② 平分亮，泉雅昭，雨宮聡史，松崎隆哲，長谷川隆三，雨宮真人，Fuceアーキテクチャによる細粒度マルチスレッディングの検討，電子情報通信学会九州支部学生会講演会，C-003, pp.159-160，2007 年 9 月 20 日，琉球大学

６．研究組織
(1)研究代表者
松崎　隆哲（MATSUZAKI TAKANORI）
近畿大学・産業理工学部・講師
研究者番号：20363385