5A-04

短周期制御機器向け並列処理対応 S/W ランタイムの開発

矢吹　潤†　　岡部　亮†　　攝津　敦†
三菱電機(株)　情報技術総合研究所†

1.はじめに

短周期と非周期処理が共存する制御機器には高応答で細やかな制御が求められ，複数のCPUコアを搭載するマルチコアマイコンでは，並列処理による制御処理プログラムの高速化が有効である[1]．プログラムを並列化するための並列化コンパイラが開発されているが，コア間での開始要求や終了通知などの連携が必要であり，これらを実行するS/Wランタイムを開発する必要がある．また，リアルタイム性が求められる制御処理においては，これらのコア間連携に掛かるオーバヘッドを低くすることが重要である．

本稿では，低オーバヘッドでコア間連携を実現する並列処理対応S/Wランタイムの方式について述べる．

2.背景

制御機器は，図 1に示すように，センサなどで取得した情報をマイコンに入力し，マイコン内で制御処理を行い，制御信号を出力することで制御を行う．

制御の高機能化を実現するために，周期時間内により多くの複雑な制御処理を実行する必要が生じている．このような制御の高度化によって，制御処理を完了するまでの時間が増加すると，決められた周期時間内に処理を完了できず，適切な制御ができなくなる．この問題を解決するために，マイコンでの制御処理を高速化することが求められている．

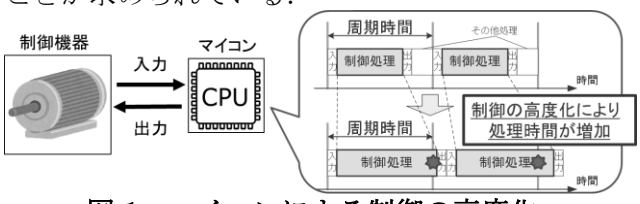

図 1　マイコンによる制御の高度化

マイコンは CPU 動作周波数が高いほど，高速な処理が可能であるが，消費電力や発熱量の問題により，CPU 動作周波数の向上は限界を迎えている．そこで，高速な処理を実現する方法として，図 2に示すように，複数の CPU コアで処理を分担するマルチコア化が期待されている．また，マルチコア化に合わせて，複数のコアで処理を分担させるためのコードを自動的に生成する，並列化コンパイラの開発も進んでいる．

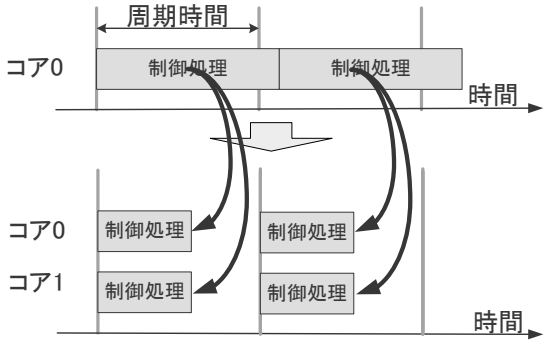

図 2　並列処理による高速化

制御処理の分割は，並列化コンパイラを使用することで可能であるが，並列化された制御処理を連携させるための並列処理対応 S/W ランタイムが必要となる．

3.課題

並列化コンパイラによって生成されたコードは，様々なプラットフォームに対応するために H/W や OS に非依存となっている．コア間連携のように OS や H/W に依存する部分は，OS や H/W に備わっている機能を使うなどして，対象とするプラットフォーム毎に開発が必要である．

並列処理対応 S/W ランタイムによって提供されるコア間連携にも実行時間が発生し，この実行時間は制御処理の並列処理を行う上でオーバヘッドとなる．一般的な OS にはコア間連携機能が備わっているが，リアルタイム性が求められる制御処理に対しては性能的に要件を満たせないことが多い．そのため，制御処理において並列処理による高速化を達成するために，低オーバヘッドでコア間連携を実現する並列処理対応 S/W ランタイムの開発が必要となる．さらに，低オーバヘッドでコア間連携を実現した上で，OS 上アプリケーションにも CPU リソースを活用できることが望ましい．

4.解決策

本章では，OSから提供されるコア間連携機能を使用する，従来方式の並列処理対応S/Wランタイムに加えて，低オーバヘッドでの並列処理が見込まれる提案方式1,2を述べる．図3～5では各方式において，並列化された制御処理の開始時に，コア0からコア1に開始要求を出す挙動を示している．

Development of Parallel Processing S/W Runtime for Short Cycle Controller
†Jun Yabuki, Ryo Okabe, Atsushi Settsu
Information Technology R&D Center, Mitsubishi Electric Corporation

従来方式：OS機能を使用する方式
コア0,1でOSを起動する．制御処理開始時，コア0はOSが提供するコア間割り込みを使用してコア1に通知する[2]．コア1はOS上アプリケーションの実行が可能である．

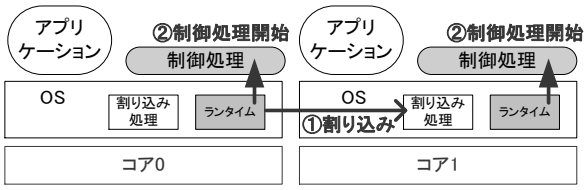

図 3　従来方式の S/W 構成図

提案方式1：ポーリング起動方式
コア0のみOSを起動する．コア1は共有メモリ上の開始要求フラグを常時監視する．制御処理開始時，コア0はフラグを更新することでコア1に通知する．コア1への高速な通知が可能であるが，コア1は開始要求フラグを常時監視するため，OS上アプリケーションの実行が不可能である．

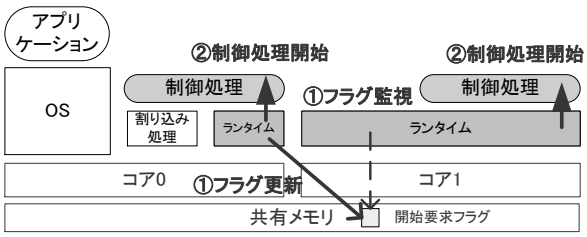

図 4　提案方式 1 の S/W 構成図

提案方式2：割り込み起動方式
コア0,1でOSを起動する．制御処理開始時，コア0はOSの機能とは異なるコア間割り込みを使用してコア1に通知する．従来方式のOS機能を使用する方式と比較して，このコア間割り込みはメモリ保護やシステムサービスなどOSの機能が使用不可となるが，高速なコア間割り込みが可能である．コア1はOS上アプリケーションの実行が可能である．

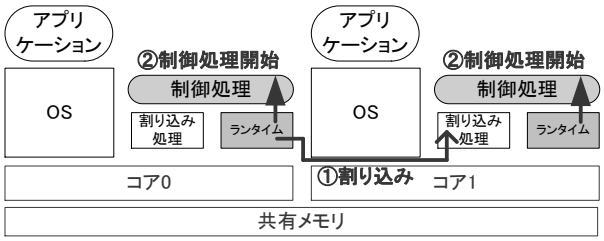

図 5　提案方式 2 の S/W 構成図

5.評価と考察

従来方式および提案方式 1,2 の並列処理対応 S/W ランタイムが提供するコア間連携のオーバヘッドを測定した．従来方式と比較した提案方式 1,2 のオーバヘッドの比率を図 6に示す．制御処理の種類によってコア間連携に掛かる時間は異なるため，測定で対象とするコア間連携は，並列処理を行う上で最低限必要となる，開始要求，終了通知のみとした．具体的には，コア 0 からコア 1 に開始要求を出し，コア 0 およびコア 1 で空の制御処理を起動した後，コア 1 は即座に終了通知を送り，コア 0 で終了通知を受け取るまでの時間を測定した．

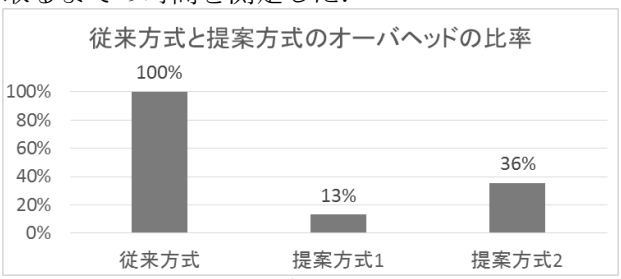

図 6　従来方式と提案方式のオーバヘッドの比率

従来方式に対し，提案方式 1 では 87%削減，提案方式 2 では 64%削減となり，低オーバヘッドでのコア間連携を実現した．

並列処理による高速化は，並列処理対応 S/W ランタイムが提供するコア間連携のオーバヘッドだけではなく，処理の並列化可能箇所や制御処理中に発生する同期の回数なども関係する．これらを総合的に評価し，CPU リソースの活用効率は下がるが低オーバヘッドを重要視する場合は，最も低オーバヘッドで動作する提案方式 1 を，オーバヘッドが許容でき OS 上アプリケーションにも CPU リソースを活用する場合は，提案方式 2 を使用するように，要求に応じて使い分けることが重要である．

6.おわりに

本稿では，マルチコアを使用した並列処理による制御処理の高速化を目指し，並列処理に必要となる S/W ランタイムの開発にむけて，低オーバヘッドと CPU リソースの有効活用を考慮した 2 通りの方式の提案を行った．今後はさらにオーバヘッド削減を図り，適用可能な制御処理の範囲を拡大させていくことが望まれる．

参考文献

[1] 梅田弾ほか：エンジン基本制御ソフトウェアモデルのマルチコア上での並列処理，情報処理学会研究報告，Vol.2012-ARC-201, No.22, pp. 1-7, 2012
[2] 菅井尚人ほか：シングルチップマルチプロセッサ上のハイブリッド OS 環境の実現，情処理学会第 66 回全国大会, 2004