報告番号	※甲　　第　　号

主　論　文　の　要　旨

論文題目　車載制御マルチコアシステムにおけるマッピングとランタイムコード生成技術

氏　　名　　小 川　真 彩 高

論 文 内 容 の 要 旨

近年，自動車制御に用いられるアプリケーションは，排ガス規制や低燃費化，そして安全性への要求の高まりといった要件から非常に複雑化，高度化してきている．中でもパワートレインの制御に用いられるアプリケーション（パワトレアプリ）は，その傾向が強い．従来のパワトレアプリはシングルコアアーキテクチャ上で動作していたが，耐熱性，耐ノイズ性の観点からシングルコアの動作周波数は限界に達しており，上述の要件を満たすためにはパワトレアプリにマルチコアアーキテクチャを用いることが必要となってきている．そのため，従来のシングルコアアーキテクチャ向けのパワトレアプリをマルチコアアーキテクチャに対応させる手法が必要である．

既存のマルチコアアーキテクチャ上のパワトレアプリの開発フローでは，まず使用するアーキテクチャを決定し，次に処理間のデータ通信の方法やタスク間の依存関係を成立させるための同期方法を決定するためにランタイム構造の設計を行う．そしてシングルコアアーキテクチャ向けにすでに作成されているアプリを複数のコア上で分散実行するために手動でアプリを分割し，分割した機能のコアへの配置を決定する．その後，実機上で実行できるようにランタイム実装を行い，実機による動作検証を行う．動作検証の結果，デッドラインミスのような問題が発生する場合には，その問題の原因を究明し，コアマッピングにフィードバックを行う．このようにして，最終的な実装を得る．

しかし既存の開発フローには，いくつかの課題が存在する．まず，コアの異なる処理間の実行依存関係を成立させるには同期が必要となるため，コア間同期による性能の悪化や制御の複雑化が課題となる．また，マルチコアアーキテクチャはコア数やメモリ構成が多様であり，アーキテクチャごとにこのような同期を実装することは非常に開発コストがかかる．さらに，マッピングの変更や機能の追加がある場合にランタイムを再実装する必要があることも課題となる．次に，安全性への影響が高いタスクのデッドラインミスが起きない処理のマッピングを探索する必要があ

ることが挙げられる．ここで並列性を出すために細かい粒度でタスク分割するとマッピングのパターンが膨大なものとなることも課題となる．最後に，所属するコアの異なるタスク間通信では，他のタスクの実行状況やコア間で共有するリソースへのアクセスのために必要な排他制御によって通信のタイミングが変動するため，動作検証を行うことが困難である．検証容易とするため，通信のタイミングが決定的となるようなランタイム構造が必要となる．

本論文では，これらの課題を解決できるような開発フローを提案する．提案する開発フローでは，パワトレアプリと実装するマルチコアアーキテクチャをモデル化し，それらに加えて処理のマッピング情報を用いてランタイムを自動生成する．マッピング情報としては最悪応答時間解析によってデッドラインミスが生じないように保証されたものを使用する．さらに，生成されるランタイムでは，通信のタイミングを決定的とするために，あらかじめ決められたタイミング（LET 区間）で通信を行う Logical Execution Time（LET）を用いて実装される．

提案する開発フローを実現するために本論文では 3 つの研究を行った．1 つ目は，Powertrain Multicore Programing Framework（PMPF）の提案である．PMPF では，パワトレアプリとハードウェアのアーキテクチャをモデル化し，それに加えて処理やデータのマッピング情報を入力として，ランタイムコードを自動生成する．これにより様々なマルチコアアーキテクチャへの対応を容易にすることが可能となる．PMPF では，タスクの途中で同期が必要ないように処理をタスクに分割し，タスク間の実行依存関係はタスク起動によって実現することで同期を不要としている．

2 つ目は，最悪余裕時間の静的解析の結果を用いて処理のコアマッピングの候補を選定する手法の提案である．まず，解析の候補を限定するために処理の集合を実行依存関係や排他アクセスが必要なデータ数によって PF グループという単位に分割する．そして PF グループ単位でのコアマッピングに対して最悪余裕時間解析を行う．最悪余裕時間が 0 以上であれば，デッドライン内で処理が完了することを保証することができる．最悪余裕時間の解析には既存手法の MIF をマルチコア向けに拡張したものを用いる．解析によって得られた最悪余裕時間が長い処理のコアマッピングを，実機に実装するコアマッピングとして選定する．

3 つ目は，LET をパワトレアプリに適用する手法の提案である．LET を実現するための既存手法では，処理の負荷を柔軟に変更することを可能とするサブスケジューリングやエンジンが高回転の場合に生じる安全性への影響が低いタスクのデッドラインミスに対応していない．そこで本研究では，サブスケジューリング下での適切な LET 区間の設定方法とデッドラインミスによって不整合なデータが生じないようなランタイム構造を提案する．また，既存手法では LET の通信を実現するための処理は実行オーバヘッドが大きいため，そのような処理を複数のコアに分散し，効率的に実装する手法を提案する．

以上 3 つの研究により，前述した課題に対応した開発フローを実現することが可能となった．