車載制御向けマルチコアプログラミングフレームワーク

Masataka Ogawa[1,a)] Shinya Honda[1] Hiroaki Takada[1]

概要：エンジン等を制御するパワトレアプリは高機能化が著しく，マルチコアによる処理が必要になるが，車種ごとにハードウェアのアーキテクチャのバリエーションが異なるため，そのすべてに対応させるのは難しいと予想される．そこで本研究では，パワトレアプリをモデル化し，そのモデルからランタイムを自動生成することにより，アーキテクチャの変更を可能とするフレームワークを開発した．実際のパワトレアプリの一部に本フレームワークを適応し，同一のモデルから，2 プロセッサ及び 4 プロセッサ用のランタイムが生成できることを確認した．

Keywords: 車載システム，マルチコア，ソフトウェアフレームワーク

1. はじめに

自動車の高性能化に伴って，車載制御が高機能化，複雑化してきている．特にエンジンやトランスミッションなどを制御するパワートレイン・アプリケーション (パワトレアプリ）ではその傾向が強い．そのため，シングルコアでは処理が間に合わず，コア数を増加させる動きがある [1], [2]．パワトレアプリは多くの車種で共通的に使用され，車種毎に使用するハードウェアのアーキテクチャが異なる．マルチコアのハードウェアーキテクチャはコア数やメモリ構成等のバリエーションが多く，そのすべてに単一のソースコードで対応させることは難しい．そこで本研究では，現状のパワトレアプリの構成をベースにモデルを定義し，そのモデルからアーキテクチャに応じたランタイムを生成するフレームワークを実現した．実際のパワトレアプリの一部に本フレームワークを適応し，同一のモデルから，2 プロセッサ及び 4 プロセッサ用のランタイムが生成できることを確認した．

2. パワトレアプリ

パワトレアプリでは，時間もしくはエンジンのクランク角に同期して処理が実行される．機能毎にモジュールに分けられており，モジュールの中の機能要素毎に，起動の要因となるイベントや起動周期，オフセットが異なる．機能要素の追加，周期変更を容易に実現するため，ソフトウェアは，次のような構成となっている．

- **レイヤ**: エンジン，トランスミッションなど，同一の機能を実現するための処理の集合である．
- **レイヤマネージャ**:タスクや ISR として実現する，機能を構成する処理の集合である．
- **サブレイヤ**:レイヤマネージャから呼び出される，同一の周期や優先度を持つ処理の集合である．1 つのレイヤマネージャから複数のサブレイヤを呼び出す．
- **レイヤ内公開関数 (公開関数)**:サブレイヤから呼び出される関数であり，実行順序に依存関係がある．

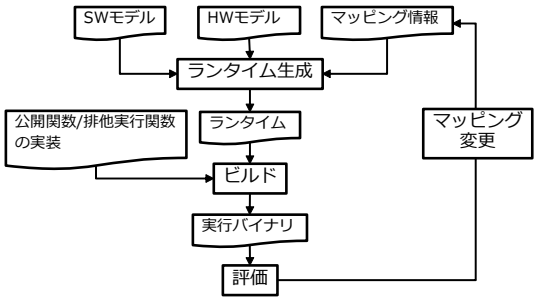


図 1 提案手法のフロー

現状のパワトレアプリはシングルプロセッサを前提として，全体が C 言語で記述されている．レイヤマネージャは，公開関数間の依存関係を満たすように順に実行していく．現状のソフト構成のまま，マルチコアに対応するためには，使用するコアの数に応じて公開関数の依存関係を満たすようにレイヤマネージャをその都度記述する必要がある．前述のように，パワトレアプリは複数の車種で共通的に使用されるため，車種毎にレイヤマネージャを開発するのは現実的ではない．

3. 車載制御向けマルチコアプログラミングフレームワーク

前章で述べた課題に対応するため，現状のパワトレアプリの構成をベースとしたモデルを定義して，そこからレイヤマネージャに相当するランタイムを自動生成するプログラミングフレームワークを提案する．提案手法による設計

[1] 名古屋大学大学院 情報科学研究科
[a)] masa-bach@ertl.jp

ⓒ 2015 Information Processing Society of Japan

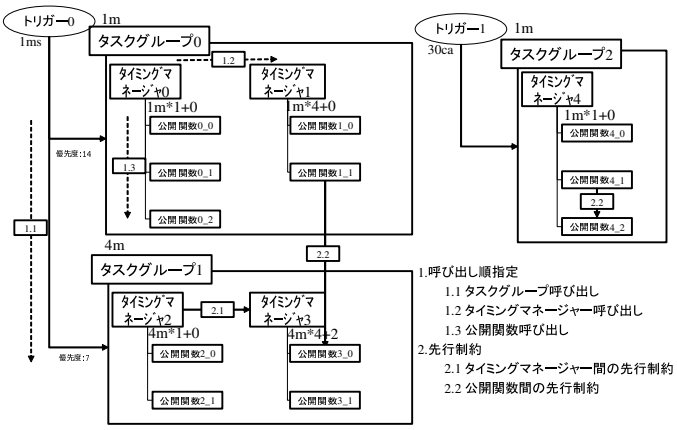


図 2 提案手法によるモデル例

フローを**図 1** に，提案モデルの例を**図 2** に示す．主要な構成要素について説明する．

- **トリガー**:処理の契機となるオブジェクトで，周期割込みやクランク角割込みから発生しタスクグループを起動する．
- **タスクグループ**：優先度（デッドライン）が同じ処理をまとめて扱うためのオブジェクトである．
- **タイミングマネージャ**：同一タスクグループ内で周期とオフセットが同じ処理の集合でタスクグループの中でも実行タイミングが同じ処理をまとめて扱うためのオブジェクトである．
- **公開関数**：処理の本体でモデル記述とは別に，公開関数の実装を C 言語で記述する．実行時間やアクセスするデータについての情報を記述する．

3.1 依存関係

モデルでは，2 種類の方法で依存関係を記述する．一つは，既存のパワトレアプリとの互換性のためにサポートする，同じタスクグループ/タイミングマネージャに含まれるすべてのタイミングマネージャ/公開関数の実行順序を指定する呼び出し順指定である．もつ一つはタイミングマネージャ，公開関数間の依存関係を個別に設定し，依存関係を持たないオブジェクト同士を並列に実行できるようにした先行制約である．この 2 つの依存関係をまとめて実行開始条件と呼ぶ．これらの依存関係がないタイミングマネージャ. 公開関数同士は並列して実行可能である．最終的には並列度の高い先行制約で実装することを想定しているが，このような並列性の高い依存関係に一度に対応することは困難であるため，まずは呼び出し順指定で実装してから徐々に先行制約に移行していくことを想定している．

3.2 ランタイム生成

実装ではリアルタイム OS を使用するため，依存関係を満たすように公開関数をタスクに割り付けて呼び出すランタイムを生成する．同じ実行開始条件をもつ公開関数の集合を一連実行グループと呼び，一連実行グループ単位でタスク化する．同じ一連実行グループに含まれる条件は，同一のタスクグループに属していること，同一のコアに割り当てられていること，共通の実行開始条件を持つ公開関数であることの 3 つである．図 2 の例において，タスクグループ 1 の公開関数がすべて同じコアに割り付けられた場合，タスクグループ 0 が終了したあとに実行できる公開関数 2_0，2_1 と，公開関数 2_0，2_1 が終了したことで実行できるタイミングマネージャ 3 内の公開関数 3_1 が同じ一連実行グループとなる．公開関数 3_0 は公開関数 1_1 が終了しないと実行できないため，同じ一連実行グループには含めない．

4. 評価

既存のパワトレアプリの一部を対象として，モデル化を行い，人手で 2 コア及び 4 コアにマッピング（マッピング 1）して，提案ツールによりランタイムを生成した．評価結果を**表 1** に示す．実行時間は，ある周期における全処理の終了までの時間を示す．

評価の結果，モデル記述は生成ランタイムと比較して，記述量が 10 分の 1 程度となることを確認した．また 2 コアと 4 コアの記述の違いはマッピング指定のみであり，単一のモデル記述からコア数の異なるランタイムが生成できることを確認した．また，公開関数のコア配置の見直しを行い評価を実施した（マッピング 2）．実行時間は 2 コアでは 7.1%，4 コアでは 27.6%短縮することが出来た．この際，マッピングを変更するのにかかった行数は 3，9 行であった．

表 1 評価結果

マッピング	コア数	モデル記述 [行数]	変更量 [行数]	ランタイム [行数]	実行時間 [μsec]	改善率 [%]
1	2	1,660	-	16,295	815	-
	4	1,662	-	17,477	682	-
2	2	1,660	3	15,674	757	7.1
	4	1,664	9	17,202	494	27.6

5. 今後の課題

今後の課題としては，対象とするアプリケーションの範囲を広げてツールの適用性を向上することと，マッピングの自動化が挙げられる．

謝辞 本研究を進めるにあたりご協力いただいたトヨタ自動車株式会社に深く御礼申し上げます．

参考文献

[1] 相庭裕史, 本田晋也, 高田広章, ”対称型マルチコアシステムのエンジン制御ソフトウェアへの適用”, 情報処理学会論文誌, Vol. 51, No. 12, pp. 2238-2249, Dec. 2010.

[2] Monot, Aurlien, et al. ”Multicore scheduling in automotive ECUs.” Embedded Real Time Software and Systems-ERTSS 2010. 2010.

ⓒ 2015 Information Processing Society of Japan