Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

# 高応答性と統合容易性を両立する マルチコア活用ソフトウェア統合ミドルウェア

田中勇気[†1]　石郷岡祐[†1]　大塚敏史[†1]　吉村健太郎[†1]
小田裕弘[†2]　内藤智[†3]　成沢文雄[†3]

**概要**：自動車制御システムではソフトウェアの開発規模増加に伴い，並列開発における効率的な統合技術が求められている．これを達成する従来技術に時間駆動スケジューリングがあるが，割込み処理の応答性が低下する課題がある．本研究では高応答性と効率的なソフトウェア統合を両立するマルチコアスケジューリングを提案する．提案手法の特徴は，複数スケジューリングの共存及び共有データアクセスの監視である．提案手法は割込み処理を実行する優先度ベーススケジューリングのコアと，アプリケーションを実行する時間駆動スケジューリングのコアで構成する．コア間の共有データのアクセス競合回避には排他制御を用いるが，排他制御中のスロット切替により，優先度ベーススケジューリングのコアの処理に大きな遅延が発生するため，これを防止する監視付き排他制御方法を示す．提案する排他制御手法は事前に測定した排他制御時間に基づいて排他制御を禁止する区間を設け，実行時には区間侵入の有無を監視しつつ，排他制御を実施する．提案手法をミドルウェアとして実装，評価の結果，割込み処理の遅延は応答性重視の優先度ベーススケジューリングと比較した際，約 4%のオーバーヘッド増加で実現可能なことを確認し，高応答性と統合容易性を両立可能な見込みを得た．

**キーワード**：スケジューリング，時間駆動，排他制御，マルチコア

## 1. はじめに

自動車制御システムでは電動化や自動運転などの機能の高度化に伴い，ソフトウェアの開発規模が増加する一方で，開発期間の短縮が求められている．開発期間の短縮には，多数の開発チームの並列開発が必須な一方で，アプリケーション統合時の共有データ(共有リソース)の競合による影響評価や実行順序の妥当性検証に工数を要するために，効率的なソフトウェア統合技術が求められている．従来技術として，タスクの実行タイミングを設計時に規定することで実行時の共有データにアクセスするタイミングを固定化し，競合発生を避ける時間駆動スケジューリングが提案されているが，自動車分野では通信による割り込み処理の応答性が低下する課題がある．

本研究では高応答性と効率的なソフトウェア統合を両立するマルチコアスケジューリングを提案する．提案手法の特徴は，複数スケジューリングの共存及び共有データアクセスの監視である．提案手法は割り込みを処理する優先度ベーススケジューリングを実行するコア（優先度コア）と，アプリケーションを実行する時間駆動スケジューリングのコア（時間駆動コア）で構成する．時間駆動コアにおいては，実行するタスク間で実行周期が異なる場合には長い周期タスクを割り当てるスロットを複数に分割した後に，すべてのタスクの実行順序が保てるようにデータフローに基づいてタスクをスロットに割り当てる．時間駆動コア内ではアクセス競合をスケジューリングによって回避するが，優先度コアと時間駆動コア間の共有データのアクセス競合回避には排他制御を用いる．しかしながら，単純な排他制御では時間駆動コアが排他制御中にスロット切替が発生すると，排他制御を取得したい優先度コアの処理に大きな遅延が発生するという課題がある．そこで我々は排他制御中のスロット切替を防止する監視付き排他制御方法を示す．提案する排他制御手法は事前に測定した排他制御時間に基づいて排他制御を禁止する区間を設け，実行時には区間侵入の有無を監視しつつ，排他制御を実施する．

本論文の構成を述べる．まず第２章で従来研究について述べ，既存手法の問題点を指摘する．第３章で提案アルゴリズムについて説明する．第４章で提案アルゴリズムを評価，第５章で関連研究について述べ，第６章で結論と今後の課題を述べる．

## 2. 従来研究と自動車制御システムへの適用

本章ではソフトウェア統合向けの既存スケジューリング手法について説明する．

### 2.1 優先度ベーススケジューリング

自動車のソフトウェア標準規格である AUTOSAR OS では割込みへの高い応答性を有するスケジューリング手法のひとつである優先度ベーススケジューリング[3]を採用している．優先度ベーススケジューリングの概要を図 1 に示す．各ソフトウェアに優先度を設定し，複数のソフトウェ

†1 (株)日立製作所
Hitachi Ltd.
†2 (株)日立産業制御ソリューションズ
Hitachi Industry & Control Solutions, Ltd.
†3 日立オートモティブシステムズ(株)
Hitachi Automotive Systems Ltd.

Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

アが同時に ready 状態に入った場合，優先度の高い制御ソフトウェアを優先して実行する．他のソフトウェアが実行している最中に優先度の高い制御ソフトウェアが ready 状態となった場合，実行中の制御ソフトウェアを一時中断し，高優先度の制御ソフトウェアを実行する．割込みの優先度を高く設定しておくことにより，割込みへの応答性を確保することが可能となる．

優先度ベーススケジューリングは高い応答性を有する一方で，統合容易性に課題がある．統合容易性とは，制御ソフトウェア統合後に各制御ソフトウェアの実行完了および結果出力タイミングのジッターが抑制されており，検証範囲が限定可能なことを意味する．優先度の低い制御ソフトウェアは割込み等優先度の高い制御ソフトウェアにより実行中断されてしまうため，実行完了タイミングの保障が困難となる．また，各制御ソフトウェアが予想と異なる時間プロセッサを使用した場合，制御ソフトウェア間の干渉によって他の制御ソフトウェアの実行タイミングにずれが生じる．この干渉により制御ソフトウェアの実行遅延，遅延によるデッドラインミスの発生および実行順番の変動などが発生する（課題１）．そのため，ソフトウェア統合時のテストケースが膨大となる．

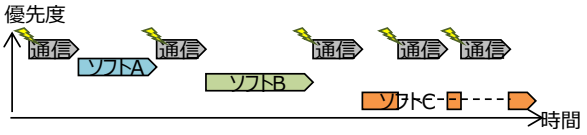


図 1　優先度ベーススケジューリング

## 2.2 時間駆動スケジューリング

同一マイコン上に統合された制御ソフトウェア間の干渉を防ぐスケジューリング手法のひとつに時間駆動スケジューリング[4]がある．時間駆動スケジューリングの概要を図 2 に示す．時間駆動スケジューリングは制御ソフトウェアへのプロセッサ割り当てを時間で分割することにより，特定の時間で各制御ソフトウェアの実行を保証する方式である．分割された時間はスロットやタイムウィンドウと呼ばれる．実行段階では，設計時に規定した時間通りに処理を実行することで，設計時に検討した処理実行タイミングを保証できる．もし，処理が設計時の予想を超えてスロット時間に達した場合には，違反が検知され，処理が中断される．これにより割込みや異常が発生した制御ソフトウェアによる他制御ソフトウェアへの干渉を防止し，ソフトウェア統合の容易性を確保可能である．

また，時間駆動スケジューリングは制御ソフトウェア間の不干渉による統合容易性を実現可能な一方で，応答性に課題がある．あらかじめ割り当てられた制御ソフトウェアのみが実行可能なため，割込みなどの高い応答性が必要な処理への対応が困難である（課題２）．

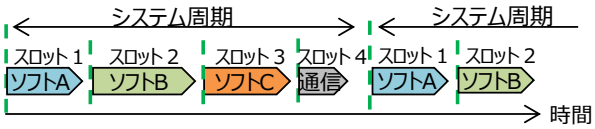


図 2　時間駆動スケジューリング

## 2.3 自動車制御システムへ適用

自動車制御システムにおいて優先ベーススケジューリングはすでに多く使用されている．一方で，時間駆動スケジューリングの適用には課題がある．課題の１つは，タスクの分割である．時間駆動スケジューリングでは各制御ソフトウェアを設定された周期および実行時間に従って実行させる．別々に作成された制御ソフトウェアは周期や実行時間が異なり，組み合わせによっては制御ソフトウェア間で実行タイミングが重なる可能性がある．これを回避するため，制御ソフトウェアのタスクを分割して複数のタスクとして扱う．これにより実行タイミングの重複を回避する．

一方で，タスクの分割は制御ソフトウェアの再設計が必要となる．統合する制御ソフトウェアのいずれかに周期や実行時間の変更が起きた場合，および新規に制御ソフトウェアが追加された場合に再設計が必要となり，再設計にかかるコストが大きい．そのため，時間駆動スケジューリング上でのスロットプリエンプションが求められる．周期や実行時間の異なる制御ソフトウェアを統合する場合，他制御ソフトウェアが配置されるスロットをまたいでスロットを配置する．そのため，異なる制御周期を有する自動車アプリケーションを統合する場合，スロットを分割して配置する必要がある（課題３）．スロットまたがりの例を図 3 に示す．例えば異なる周期をもつ安全系制御ソフトウェア（図 3 のソフト A）と非安全系制御ソフトウェア（図 3 のソフト B）を同一コア上で動かす際に，非安全系制御ソフトウェアのスロットの途中で安全系制御ソフトウェアの起動タイミングになった場合，安全系制御ソフトウェアのスロットを優先配置する必要がある．この際に非安全系制御ソフトウェアのスロットは複数に分割され，安全系制御ソフトウェアのスロット終了後に再度配置される．この場合，非安全系制御ソフトウェアは１つ目のスロット（ソフト B1）の終端にて処理を一時中断し，２つ目のスロット（ソフト B2）の始端にて処理を再開する．異なる制御間隔で動作する制御ソフトウェアを共存させるため，スロットまたがりへの対応が必要である．

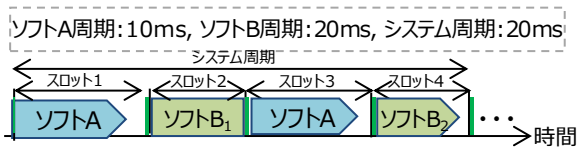


図 3　スロットまたがり

Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

## 3. 提案手法

本章では 2 章で述べた課題を解決するため，図 4 に示す高応答性と統合容易性を両立可能とするスケジューリング手法を提案する．

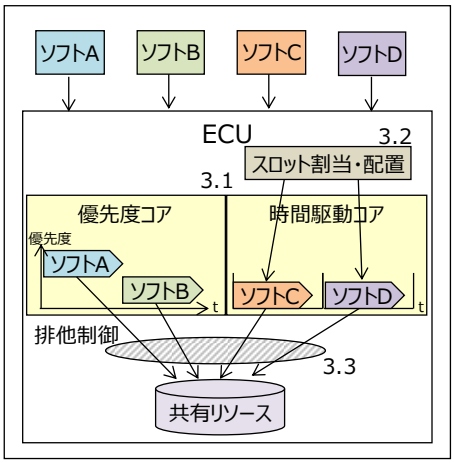


図 4　提案手法概要

### 3.1 ハイブリッドスケジューリング

（課題１)および（課題２)を解決するため，高応答性と統合容易性の両立を実現する，複数のスケジューリング手法を使用したハイブリッドスケジューリングを提案する．ハイブリッドスケジューリングの概要を図 5 に示す．ハイブリッドスケジューリングでは，制御ソフトウェア間の干渉防止を実現する時間駆動スケジューリングと高応答性を実現する優先度ベーススケジューリングを使用する．マルチコアシステムを活用し，時間駆動スケジューリングを採用したコアと優先度ベーススケジューリングを採用したコアを用意する．このように提案するハイブリッドスケジューリングはマルチコアシステム上で２つのスケジューリング手法が共存する特徴を持つ．

時間駆動スケジューリングを採用したコアでは，制御ソフトウェアを実行するタイミングをスロットと呼ばれる単位で固定化し，一定周期で反復実行することで，実行タイミングの差異による組合せを排除する．各制御ソフトウェアの実行時間を保証するため，各制御ソフトウェアの実行周期，最悪実行時間の情報から各制御ソフトウェア専用のスロットを作成する．制御ソフトウェアが割り当てられた時間を超えて動作しようとした場合に，共有リソースを開放及び制御ソフトウェアを停止させることで，割り当てられた時間内に実行完了しなかった制御ソフトウェアによる時間干渉を防止し，他の制御ソフトウェアの実行時間を保証する．これにより制御ソフトウェア間のプロセッサ時間干渉を防止し，統合容易性を実現する．

時間駆動スケジューリングでは通信の受信割込のような高応答処理に対応困難なため，割込みは優先度ベーススケジューリングを採用した別のコアで対処する．優先度ベーススケジューリングを採用したコアでは，ready 状態の制御ソフトウェアが複数存在する場合，割り当てられた優先度の高い制御ソフトウェアを優先して実行する．割込みなどの高い応答性を必要とする制御ソフトウェアの優先度を高く設定することにより，高応答性を実現する．

複数のスケジューリング手法をマルチコアシステム上で共存させることにより，制御ソフトウェア間の干渉防止および割込み処理の即時対応を可能とし，割込み発生タイミングの予測が困難な場合においても高応答性と統合容易性を両立可能となる．

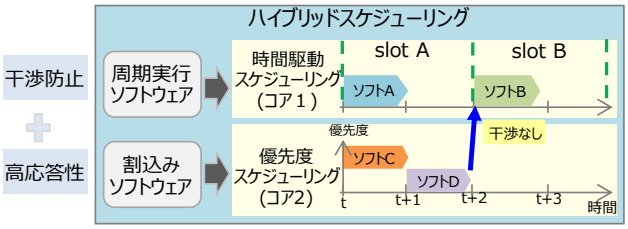


図 5　ハイブリッドスケジューリング

### 3.2 スロット割当・配置アルゴリズム

複数の制御ソフトウェアを統合する時間駆動スケジューリングでは，各制御ソフトウェアをスロットに割当・配置する作業が必要となる．（課題３)を解決するため，またがりに対応したスロット配置方法を提案する．この際に制御ソフトウェアの実行は規定周期ごとに実行開始・完了するリアルタイム性を保持したスケジューリングが必要である．また，各制御ソフトウェアには実行順序が存在し，これを満たしたスロット配置が必要となる．そのため，制御ソフトウェアのリアルタイム性及び実行順序を満たしたスケジューリング方式を提案する．

実行順序及びリアルタイム性を満たすためのスロット割当およびスロット配置について説明する．図 6 にスロット割当の概要を示す．提案手法は各制御ソフトウェアの設計情報として「ソフトウェア名，最悪実行時間(WCET)，実行周期，前提ソフトウェア」を入力として扱う．前提ソフトウェアとは，制御ソフトウェアの実行開始前に実行完了している必要のある制御ソフトウェアを表す．スロット割当では実行順序の降順に，実行周期に応じてスロットサイズの決定を行う．入力された WCET にマージン時間を追加して親スロットを生成する．生成した親スロットが周期の最大公約数より大きい周期をもつスロットの場合，親スロットを「(スロット時間×(周期の最大公約数/周期))ms ×(周期/周期の最大公約数)個」にスロット分割する．(例：30ms 周期，スロット時間:6ms，最大公約数:10ms ⇒ (6ms × (10ms/30ms)) × (30ms/10ms) ⇒ 2ms スロット×3 個)．分割したスロットに番号付け及び実行順序を割り振る．実行順序は，分割したスロットの先頭スロットはもとの順序情

Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

報を保持し，後続スロットは分割前の順番に並ぶよう順序情報を設定する．

スロット配置の概要を図 7 を用いて説明する．本方式では配置するスロットの選択，配置箇所探索を全スロットが配置完了するまで繰り返す．前提スロットの配置が完了しているスロット，もしくは前提スロットの無いスロットを選択する．前提スロットとは，スロットが実行開始する前に実行完了している必要のあるスロットを指す．選択条件をクリアしているスロットが複数存在する場合，周期の短いスロットを優先して選択する．選択したスロットを前提スロットの実行完了以降かつスロット配置可能な空き時間の先頭からスロット配置する．配置したスロットの開始時刻を基準として，周期ごとにスロットを配置する．すでに他スロットが配置されている場合，空き時間を探索して配置する．本方式により，データ入力から処理して出力するまでの E2E(End-to-End)の処理時間を E2E の周期内に抑えたスロット配置を可能としている．

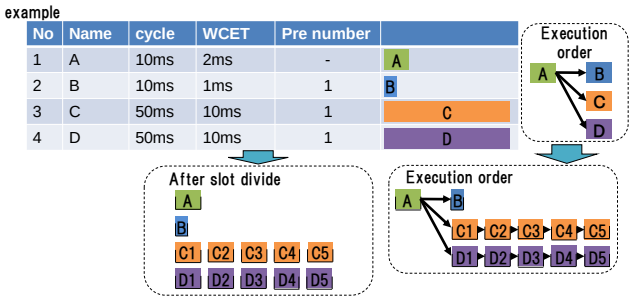


図 6　スロット割当

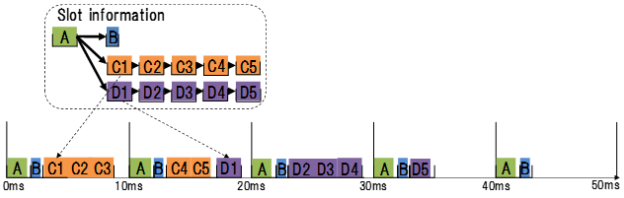


図 7　スロット配置

### 3.3 監視付き排他制御

#### 3.3.1 ハイブリッドスケジューリング適用時の課題

自動車制御用ソフトウェアは異なる制御間隔で動作する制御ソフトウェアが混在しており，共有リソースへの同時アクセスが発生する．そのため，ハイブリッドスケジューリングを適用する場合に，優先度コアの制御ソフトウェアと時間駆動コアの制御ソフトウェアが使用する共有リソースは排他制御によって管理される必要があり，排他を介して優先度コアの制御ソフトウェアの処理に干渉する可能性がある．干渉の例を図 8 に示す．非安全系制御ソフトウェアが排他を取得した状態でまたがりスロットの終端になった場合，次に実行開始する安全系制御ソフトウェアや優先度コアの制御ソフトウェアは排他を取得できず，共有リソースを使用した処理を実行できなくなる．これにより，処理に大幅な遅延が発生し，デッドラインオーバーとなる．

自動運転システムはデータの信頼性を担保する必要があり，ロックフリーでの実装が困難であるため，排他制御を介した干渉を防止する必要がある(課題 4). これを解決するため，優先度ベーススケジューリングと時間駆動スケジューリングの共存に対応した排他制御を次項で提案する．

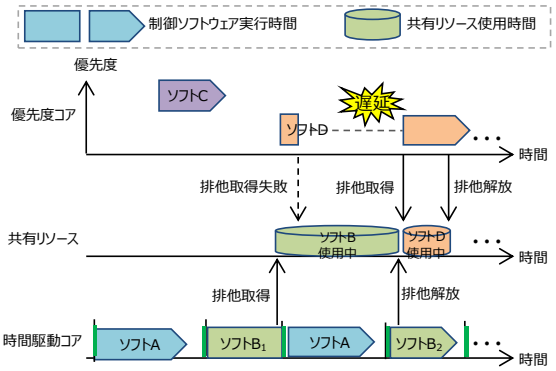


図 8　コア間排他制御の影響

#### 3.3.2 ハイブリッドスケジューリング対応排他制御

（課題３)および（課題４)を解決する，複数スケジューリングの共存に対応する監視付き排他制御の概要を図 9 および図 10 を用いて説明する．監視付き排他制御では，排他を介した共有リソースへアクセスするインタフェースの処理時間をあらかじめ測定しておく．共有リソースのアクセスをまたがりスロット（図 10 のソフト B1）の終端に共有リソースへのアクセスを禁止する排他取得禁止区間を設ける．排他取得禁止区間の長さはあらかじめ測定した共有リソースへのアクセス時間を満たすように設定する．各制御ソフトウェアが排他を介する共有リソースへアクセスする際に，アクセス開始タイミングが排他取得禁止区間外である場合，排他取得のうえ共有リソースへのアクセスを許可する．アクセス開始するタイミングが排他取得禁止区間内である場合，共有リソースへのアクセスを禁止する．共有リソースへのアクセスを禁止した際には禁止の旨を制御ソフトウェアに通知し，禁止区間を抜けるまで待機もしくは次の処理へ移行させる．

排他取得禁止区間はスロット切り替えのタイミングで解除されるため，制御ソフトウェアは次の割り当てられたスロットにて共有リソースへのアクセスが可能となる．排他取得禁止区間を共有リソースアクセス時間を満たすように設定することで，排他取得による他制御ソフトウェアへの干渉を防止可能となる．排他取得禁止区間内に共有リソースへアクセスを開始した場合，スロット内にアクセス処理が完了せず，排他を保持したまま処理が中断される可能性がある．これは他の制御ソフトウェアへの干渉につながる．そのため，排他取得禁止区間内での共有リソースアク

Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

セスを禁止している．排他取得禁止区間外で共有リソースへのアクセスを開始した場合，排他取得禁止区間がマージンの役割を果たし，割り当てられたスロット終端までに共有リソースアクセスを完了し，排他が解除される．これにより，排他を保持したまま次のスロットへ移行することを防ぐことができ，他の制御ソフトウェアへの干渉を防止可能となる．

また，同一コア上に非安全系制御ソフトウェアと安全系制御ソフトウェアがある場合，またがりスロット以外に非安全系制御ソフトウェアのスロット終端に排他取得禁止区間を配置してもよい．これによりまたがりに限らず他の制御ソフトウェアへの排他を介した干渉を防止可能となる．

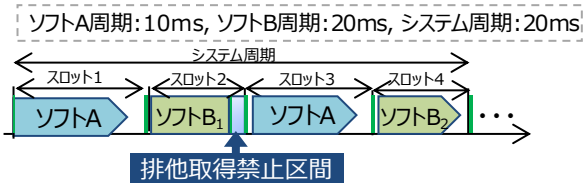


図 9　排他取得禁止区間

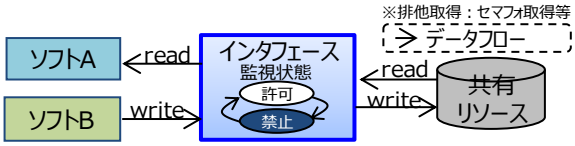


図 10　監視付き排他制御

### 3.3.3 排他制御の使用方法

各スケジューリング手法の組み合わせにおける排他制御の使用方法を表 1 に示す．時間駆動コア上の各制御ソフトウェアは共有リソースへのアクセスタイミングが分断されており，時間駆動コア上の制御ソフトウェア間は排他制御を必要としない．優先度コア上の制御ソフトウェアが共有リソースへアクセスを行う場合，応答性を重視するため，セマフォ等即時実行可能な排他制御手法を用いる．時間駆動コア上の制御ソフトウェアが優先度コア上の制御ソフトウェアと共有のリソースへアクセスする場合，優先度コア上の制御ソフトウェアへの干渉を防ぐため，監視付き排他制御を用いる．

応答性が必要な制御ソフトウェアへ干渉の可能性を持つ共有リソースアクセスに監視付き排他制御を適用することで，タイミングテストのパターン数を削減可能となる．

表 1　排他制御方法

| 排他制御方法 | | 共有相手 | |
|---|---|---|---|
| | | 時間駆動 | 優先度 |
| アクセス者 | 時間駆動 | 必要無し | 提案手法 |
| | 優先度 | セマフォ | セマフォ |

## 4. 実験評価

### 4.1 実験環境

提案したスケジューリング方式のオーバーヘッドを評価するため，提案スケジューリング方式および排他制御方式をミドルウェアとして実装し，性能評価を実施した．図 11 に示すように，ミドルウェアがスロットの始端にて割り当てられた制御ソフトウェアを起動することで時間駆動スケジューリングを実現している．性能評価にはルネサスエレクトロニクス社の評価ボード(R-CarV2H)を用いた．評価環境を表 2 に示す．R-CarV2H は自動運転向けのマイコンであり，ARM Cortex-A15 processor が搭載されている．自動運転ソフトウェアの設計情報を入力としてスロット割当・配置を行い，配置結果に沿って各制御ソフトウェアを実行した．

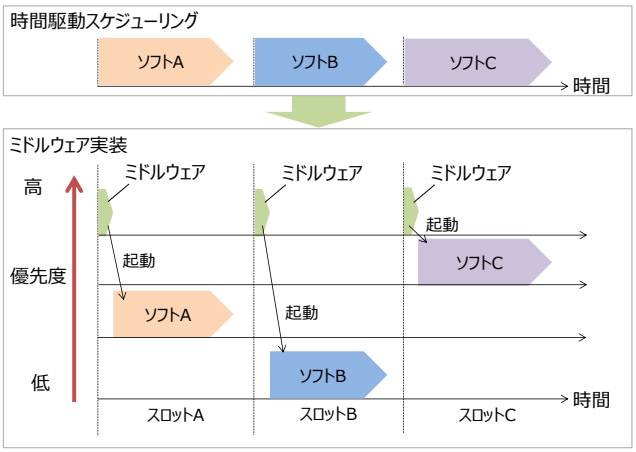


図 11　制御ソフトウェアの時間管理

表 2　評価環境

| マイクロコントローラ | 実行タスク | 個数 | 50ms中の切替回数 |
|---|---|---|---|
| ルネサス社製 R-CarV2H | 10ms | 1 | 10 |
| | 50ms | 3 | |

### 4.2 実験結果

提案スケジューリング方式を実装したミドルウェアの処理時間を図 12 に示す．処理時間は最小値，平均値，最大値および，揺れを考慮してマージン時間を設定した場合の時間をそれぞれ示している．提案スケジューリング方式の処理内容は，制御ソフトウェア状態制御(Ttt)，タイマ制御(Tcmt)，共有リソースの開放と OS 割込みによる遅延を考慮したマージン時間(Tbuf)，そしてコンテキストスイッチ(Tsw)から構成される．制御ソフトウェア状態制御(Ttt)の最大時間は 65 $\mu$ s であるが，これには OS 割込みが含まれている．OS 割込みはマージン時間(Tbuf)にて補うため，制御

Vol.2019-ARC-235 No.6
Vol.2019-SLDM-187 No.6
Vol.2019-EMB-50 No.6
2019/3/17

ソフトウェア状態制御(Ttt)に割り当てる時間からは除外する. OS 割込みによる時間増加が無い場合には 35μs が最大となっており，制御ソフトウェア状態制御に割り当てる時間(Ttt)は実行時間の揺れを考慮して 40μs とした．タイマ制御(Tcmt)は最大実行時間が 1.4μs となっており，揺れを考慮して 5μs を割り当てた．共有リソースの開放には 17μs，頻度の高い OS 割込みによる遅延が 30μs となっている．すべての OS 割込みが重なった場合に生じる遅延は 50μs となっており, 揺れを考慮してマージン時間(Tbuf)は 80μs を割り当てた. コンテキストスイッチの時間は最大 12.8μs となっており，揺れを考慮して 1 回あたり 15μs 割り当てた．制御ソフトウェアの管理にはコンテキストスイッチが最大で 6 回必要となり 90μs 必要である．以上から提案スケジューリング方式をミドルウェア実装した際の制御ソフトウェア切り替えの最大時間は 220μs, CPU 使用率にして全体の 3.49%と，自動運転システムにおいて十分小さいことを確認した．これにより，ミドルウェアを用いた実装により，時間駆動専用 OS を用いずに，優先度ベーススケジューリングにてタスク管理を行う OS においても提案手法を適用可能なことを確認した.

また，提案スケジュール方式によるオーバーヘッドは割込み処理の遅延時間増加量から算出される．応答性重視の優先度ベーススケジューリングのみと比べた場合のオーバーヘッドを図 13 に示す．提案方式を適用することで発生する処理増加は排他取得禁止区間の制御時間である 2.8us となり, 共有リソースアクセス処理(アクセス時間+OS 割込み)の 67us に比べ 4.2%のオーバーヘッドで実現可能となる.

各スケジューリング方式における割込み処理応答時間のジッタの机上計算を行った．結果を図 14 に示す．優先度ベーススケジューリングにおける割込み処理のジッタは他制御ソフトウェアによる共有リソースアクセス時間のみの 67[us]となる．時間駆動スケジューリングで割込み処理を扱う場合，排他制御による干渉は無い一方で，割込み処理を行うスロットを待つ必要があり，最大で 9530[us]必要となった．排他制御による干渉対策を行わずに優先度スケジューリングと時間駆動スケジューリングを混在させた場合，時間駆動コアにおけるまたがりを経由した排他保持に影響を受ける．そのため，ジッタは 470[us]となった．ハイブリッドスケジューリングおよび監視付き排他制御を適用した提案手法では，ジッタは共有リソースアクセス時間に排他取得禁止区間の制御時間を加算した 69.8[us]となる．以上のように，提案手法は優先度ベーススケジューリングの応答性と時間駆動スケジューリングの統合容易性を両立可能な見込みを得た.

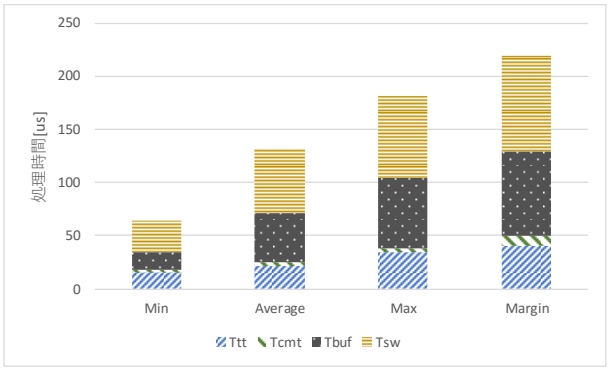


図 12　ミドルウェア実装時の処理時間

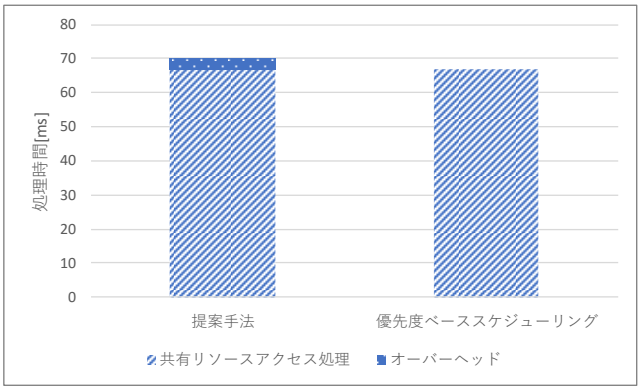


図 13　提案スケジューリング方式のオーバーヘッド

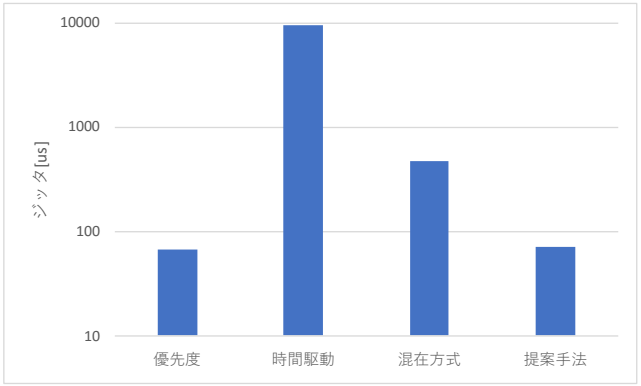


図 14　割込み応答時間のジッタ

## 5. 関連研究

本章では，本論文で提案したスケジューリング方式に関連する既存研究を紹介する.

割込みへの高い応答性を確保しつつ不干渉による統合容易性を実現する手法が提案されているが，割込みの頻度が不明な場合への対応については言及されていない[5]．本研究では割込みの頻度が不明なシステムにおける高い応答性と統合容易性の両立を実現している.

制御ソフトウェアごとに適用するスケジューリングアルゴリズムを変更する手法が提案されているが，マルチコアシステムへの対応及び時間駆動スケジューリング混在時

の排他制御による干渉への対応については言及されていない[6]．本研究ではマルチコアシステムにおいて時間駆動スケジューリングと他スケジューリングアルゴリズムを適用した際の課題を解決している．

さらに，制御ソフトウェアの依存関係やデッドラインを制約式として定め，探索的に制約式を満たす制御ソフトウェアのスケジューリングを導く方法が提案されている[7]．しかしながら，自動運転システムのような大規模かつ複雑なシステムにおいて短時間で解法を行うことは困難である．一方，提案手法は解を導くアルゴリズムによって導き出す手法であり，短時間でのスケジューリングが可能である．

共有リソースにおける排他制御を介した干渉を防ぐ方法は存在するが，ハードウェアリソースの節約が必要な場合での実装は言及されていない[8]．本研究では単一リソースでの干渉防止により，ハードウェアリソースを節約可能とした．

以上のように，高応答性や統合容易性それぞれを実現する手法は提案されているが，大規模複雑な自動運転システムを想定した，高応答性と統合容易性の両立は実現されていなかった．

## 6. まとめ

本論文では，割込みの予測が困難な自動運転システムにおいてマルチコアシステム上で高応答性，統合容易性を両立するスケジューリング方式を提案した．提案手法はマルチコアシステムを利用して複数のスケジューリングを同一マイコン上に実装，高応答性と統合容易性の両立を可能とした．さらに，時間駆動スケジューリングに対応した監視付き排他制御によりコア間の共有リソースへのアクセス競合による遅延を防止した．提案手法をミドルウェアとして実装評価し，自動運転システム向けソフトウェア開発における高応答性と統合容易性を両立可能な見込みを得た．

今後は,スロット割当結果とデータ依存性を解析により，ミドルウェア実装時のスロット切り替え時間の最小化方法を研究する計画である

## 参考文献

[1] 一般社団法人 JASPAR, “機能安全対応のための解説書（ソフトウェア・パーティショニング編） Ver.1.0,” H-FSS-07-0002, 2013.
[2] ISO, “ISO 26262 : Road vehicles - Functional safety Part 1~9”, 2011.
[3] Florian Kluge, Chenglong Yu, Jörg Mische, Sascha Uhrig, and Theo Ungerer, “Implementing AUTOSAR scheduling and resource management on an embedded SMT processor”, Proceedings of th 12th International Workshop on Software and Compilers for Embedded Systems. ACM, p. 33-42, 2009.
[4] H. Kopetz, “Should Responsive Systems be Event-Triggered or Time-Triggerd, IEICE Trans. Information and Systems”, Vol.E76-D, No.1325-1332, 1993.
[5] 佐藤祐一, 松原豊, 本田晋也, 高田広章, "高応答性を要求するリアルタイムアプリケーション統合のための ARINC653 拡張スケジューリングアルゴリズム." 研究報告組込みシステム (EMB) 2014.3 (2014): 1-6.
[6] 松原豊, 本田晋也, 冨山宏之, 高田広章, "リアルタイムアプリケーション統合のための柔軟なスケジューリングフレームワーク." 情報処理学会論文誌 49.10 (2008): 3508-3519.
[7] Silviu S. Craciunas, ” SMT-based Task- and Network-level Static Schedule Generation for Time-Triggered Networked Systems”, Proceedings of the 22nd international conference on real-time networks and systems. ACM, 2014
[8] G. Buttazzo, "Achieving scalability in real-time systems", IEEE Computer., vol. 39, no. 5, pp. 54-59, May 2006.