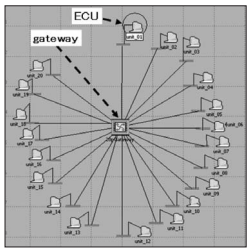


車載用通信プロトコル（スケーラブルCAN）開発

倉地 亮*・西村政信*・高田広章
手嶋茂晴・宮下之宏・堀端啓史
山本秀樹・夏目晃宏

Development of Scalable CAN Protocol —— by Ryo Kurachi, Masanobu Nishimura, Hiroaki Takada, Shigeharu Teshima, Yukihiro Miyashita, Satoshi Horihata, Hideki Yamamoto and Akihiro Natsume —— In today's automotive industry, FlexRay and other next generation protocols for automotive network communications are gaining attention. However, these protocols are unlikely to replace existing applications in the immediate future due to the cost and reliability problems caused by the replacement.

In this paper we propose Scalable CAN, a new automotive network protocol based on the existing CAN (controller area network). Having a new ACK (acknowledgement) information field, instead of an ACK slot, the Scalable CAN features 10 Mbps of transmission speed and a new collision resolution algorithm which guarantees delivery of a message within a given time period. Our analysis using a simulator indicates that the Scalable CAN protocol is superior to the conventional CAN in throughput performance such as maximum data transmission speed, scalability, and priority inversion.

This paper also includes considerations given for the implementation of the Scalable CAN.

Keywords: Scalable CAN, protocol, controller, delay

1. 緒　言

近年、自動車装備の電子化に伴い、車両一台あたりに搭載されるECU（Electronic Control Unit）は数十個にも及ぶ。各ECUは、車載ネットワークを介して、多くの情報を共有し、様々な機能や制御を実現している。車載ネットワークとして、低速ではLIN[1]（Local Interconnect Network）、中高速ではCAN[2]（Controller Area Network）が用いられることが一般的である。しかしながら、これらのネットワークには、既に多くのECUが接続されており、今後の拡張性は、GW（gateway）を介したネットワークの追加や、ECU統合による接続数の低減などに頼らざるを得ない。

一方で、高速・高信頼性を確保するネットワークとしてFlexRay[3]があり、既に欧州の高級車メーカなどから、ネットワークの一部にFlexRayを搭載した自動車が発売されている。しかしながら、CANからFlexRayへの移行は、既存のCAN向けソフトウェア資産（ミドルウェアやアプリケーション）を流用できず、ネットワークに接続可能なノード数がCANに比べて制限されるなど、高速化以外の課題が残る。

自動車の分野では、新たなプロトコルへの移行に伴い、システムや部品を一新することは、コストや品質確保の観点からも非常に難しい。そのため、既存プロトコルを使い続けるための技術開発も重要になる。

そこで本稿では、将来の高速化にも対応可能で、従来CAN向けに開発されたソフトウェア資産との互換性を保ちつつ、より高速で低遅延な通信を可能とする新たなプロトコル（スケーラブルCAN）について紹介する。また提案するプロトコルは、低速から高速まで、車載ネットワークとして様々な用途にスケーラブルに適用可能とすることを目指して開発した。

2. CANの課題

CANは、衝突メッセージを非破壊で調停可能とする特徴的なメカニズムを有するため、伝送路上の遅延など様々な制約を持つ。これらのメカニズムや制約は、伝送速度の高速化の妨げとなっており、仕様としては1Mbpsまで規定されているものの、車載環境下で使用されるのは500kbpsまでに留まっているのが現状である。また、500kbps以下のCANであっても、特定の条件下では、アプリケーション要求を満足できないなどの課題がある。

以下にて、CAN高速化へ向けて、本稿が解決しようとする課題の詳細を示す。

2－1　伝送路上の遅延　CANの高速化を妨げる要因として、以下の2つの特徴的な機能がある。

① 'bitwise'（non-destructive）arbitration

② ACK（Acknowledgement）field

CANは、CSMAによるコンテンション・ベースの調停メカニズムを持つが、SOFビットで開始されるフレームは、複数ノードからの送信によって衝突し得る。しかしながら、衝突した場合であっても、同期メカニズムによって同期された1bit毎にビット比較が行われ、優位ビット（ドミナント）を送信しているフレームは、劣位ビット（レセッシブ）を送信しているフレームによって破壊されることなく、調停フィールド以内に送信権限を獲得し、その後の送信を維持できる。

CANのACKメカニズムは、ビット同期メカニズムによって同期された複数のノードが、受信したフレームに対するACK/NAK判定結果を、同時に同じビット位置（ACKスロット）へ送信し合うことで、CANバス上の全てのノードによる判定を実現している。

これらのメカニズムは、CANを特徴付ける上で重要な仕様であるが、CANバス上の各ノード間での伝送遅延に対して厳しい制約を生む要因でもある。

CANをより高速で使用可能とするためには、上記2つの特徴的な機能を置換し、伝送遅延に対する制約を排除できる工夫が必要となる。

その他にも、配線長の増加に伴う信号の減衰や、CANバスの分岐点の数、支線長の影響など、物理的な信号特性の劣化要因も高速化を妨げるが、これらの要因は、後述する提案手法を採用することで、バス・リピータの導入など一般的な手法での改善が期待されるため、ここでは議論しない。

2－2　メッセージの遅延保障　車載システムでは、しばしば一定時間内にECUから別のECUへメッセージを受け渡すことが求められる。CANバスに対するメッセージの送信は、送信ノードの内部イベントに応じて開始される。CANバスに対し、複数のノードから同時にフレームが送信される場合、そのフレームは、各送信ノードによって同期され、調停フィールド内で最終的な送信権限が決定されるため、低い優先度のIDを持つフレームは、調停負けする。この調停負けは、再送信の場合であっても同様に複数回連続して起こり得る。このことは、イベント発生に伴い送信開始されたメッセージが、他のノードに受信されるまでの最悪時間を保障できないことを意味する。

一般的に、連続して調停負けすることへの対策としては、OSのタスクスケジューリング等と同様の考えにより、高優先度のフレーム送信を連続して行わないなど、運用上の工夫によって一定の改善が期待できるが、このような対策は、問題を根本的に解決するものではない。

3.　CAN高速化への提案手法

本稿では、伝送路遅延の影響を最小にし、メッセージ遅延の最悪値を保障可能とするために、プロトコル上の工夫によって実現する手法を提案する。

3－1　アプローチの概要　既存CANに対する伝送遅延要求を緩和し、最悪遅延の予測を可能にするため、従来CANに対し、下記の変更を加える。

① ACKメカニズムの変更（ACK/NAK情報のメッセージ化）

② 調停手法の変更（スロットによる送信権限の管理）

(1) 遅延要求の緩和

①②の変更により、従来、厳しい伝播遅延要求を伴い、1bit時間内で実施されていたバス調停およびACKメカニズムが廃止され、既存の各種シリアル通信プロトコルと同様に、ノード間の伝播遅延に対する制約は大きく緩和できる。さらに、受信したフレームに対するACK/NAK情報は、フレーム中の新たな情報ビットとして送信することで、従来機能を代替する。

(2) 最悪遅延の保障

②の変更により、バスに対する送信権限を時間間隔（スロット）毎に管理し、各スロット毎に、1つの送信ノードを割付けることで、送信時の衝突を回避し、従来CANにおいて「バス調停で負け続ける可能性」によって予測困難とされていたアプリケーション間の最悪遅延時間を、一定時間内に収めることが可能となる。

以降にて、これら提案手法を実現する手段について、その詳細を示す。

3－2　フレームフォーマット　提案手法におけるフレームフォーマットと従来CANとの比較を**図1**に示し、その変更点について以下に示す。

「ACK情報フィールド（図中、AckInfo）」は、自身の送信以前に受信した他者の送信に対するACK情報を格納する。従来CANにおいてACKスロットで提示されていた情報をメッセージとして通知するためのものである。

「CRC（cyclic redundancy check）フィールド」は、17ビットに変更され、CRC（15bit）とCRCデリミタ（2bit）で構成される。ACKビットの廃止に伴い、CRCデリミタを2ビットとし、1ビットのレセッシブと1ビットのドミナントで構成する仕様とした。ACKビットの廃止に伴い、

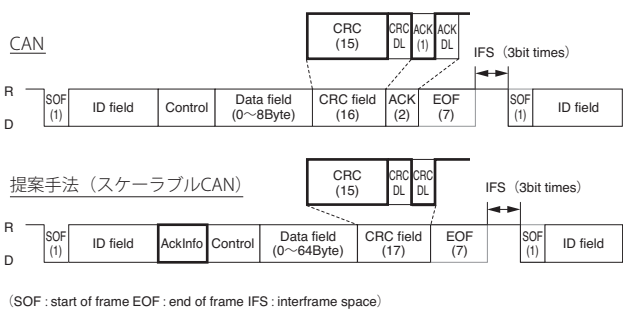


図1　フレームフォーマット

フレーム終端を示すEOF（end of frame、レセッシブ7bit）以前にビット再同期処理が行われる立ち下がりエッジは、15ビットのCRC中にて発生するドミナントビットとなる。従来CANの仕様であれば、EOF以前の立ち下がりエッジは、CRCデリミタに続くACKビットであり、その後にビット再同期されないレセッシブ・ビットの連続で累積しうるノード間の同期ズレ量は、このエッジを基点に推定可能であった。今回、ACKビットの廃止に伴い、従来ACKビットが担っていた同期メカニズム上の意味合いは、新しいCRCデリミタ（2bit）のビットパターンによって代替することとした。

3－3　通信シーケンス　　通信シーケンスについて、簡略化のため、**図2**に示すように、GWとECUが1対1で接続された場合を例にして説明する。

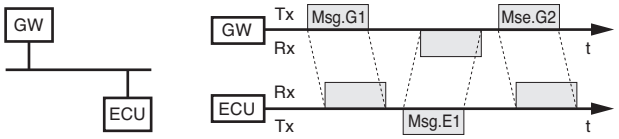


図2　接続構成と通信シーケンス

（1）基本シーケンス

図2に示すように、GWが先に送信を開始し、送信されたフレームが衝突しない場合を例に通信シーケンスを示す。GWは、一定時間バス上にフレームが送信されていないことを確認した後、自身のフレーム（Msg.G1）を送信する。バス上にフレームが送信されたことを検出したECUは、そのフレーム（Msg.G1）を受信する。

フレーム（Msg.G1）を送信完了したGWは、ECUからの応答を期待し、受信待ち状態となる。GWからのフレーム（Msg.G1）を受信したECUは、受信したフレームの異常の有無に応じて、ACK情報フィールドにACKまたはNAKを格納したフレーム（Msg.E1）を送信する。このときECUは、GWから受信したメッセージに対する応答や、自身が送信したいイベントデータなどがあれば、データフィールドに格納してフレームを送信する。

ECUからの応答フレームを待っていたGWは、フレーム（Msg.E1）を受信し、このフレームに含まれるACK情報を参照することによって、先の送信の成否を判定する。同様に、GWは、受信したフレーム（Msg.E1）に対するACKまたはNAKを格納した次のフレーム（Msg.G2）を送信する。

このように、GWとECUが順にACK情報を含むフレームを送信し合うことで、衝突を回避し、各ノードが、一定時間内に、送信権限を獲得することで、調停に負け続ける可能性を排除し、アプリケーション間における最悪遅延の保障を可能にしている。

（2）送信待ち時間

前述（1）にて示すとおり、フレームを送信する順番にあるノードが、自身のフレームを送信するまでの送信待ち時間は、自身の送信前に受信したフレームのCRCデリミタ（2bit）を検出後から、$t_{backoff}$ 時間経過後とする。

$$t_{backoff} = \left(t_{EOF} + t_{IFS}\right)$$

（3）期待応答受信時間

前述（1）の通信シーケンスに示すとおり、一旦、送受信が開始された後は、現在の送信者と次の送信者との間で、フレームの衝突が回避できることを担保する必要がある。このために必要な、時間条件に対する考え方を以下に示す。

図3は、GWがフレームを送信後、ECUからのフレーム受信を待つ時の、期待時間を示すものである。メッセージの送信中、CRC デリミタの送信が完了した時点を $t = 0$ とし、応答メッセージのSOFを検出するためにビットサンプリングを開始する。SOFを検出した時の時刻を t とする。

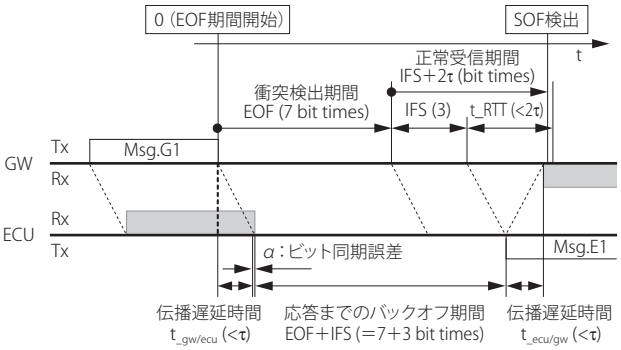


図3　通信タイミングの詳細

GWがフレーム（Msg.G1）を送信後、ECUの応答送信フレーム（Msg.E1）を受信するまでの時間は、理想的には、t_{response} 時間である。ビット同期誤差 α は、正負の値を取りうる。

$$t_{\mathrm{response}} = \left(t_{EOF} + t_{IFS} + t_{gw/ecu} \pm |\alpha| + t_{ecu/gw}\right)$$

ここで、GWとECU間の往復に要する伝播遅延時間（$t_{gw/ecu} + t_{ecu/gw}$）を t_{RTT}（Round Trip Time）と表わすと、GWがECUからの応答を期待できる待ち時間は、（EOF＋IFS＋$t_{RTT} \pm |\alpha|$）時間となり、（$t_{RTT} > \alpha$）の関係が成立しない限り、GWが認識するIFS（3bit times）中に、ECUの送信したフレームの先頭ビット（SOF）が検出される可能性が生じる。

t_{RTT}は、1ビットの時間よりも十分短い場合もあり得るだけでなく、αは、ボーレートが低い場合や、ビットに対するサンプリング周期が長い程、t_{RTT}に対し相対的に大きくなる（$t_{RTT} < \alpha$）。
本稿で提案する方式では、下式が成立する範囲内の伝送遅延と同期誤差を許容することとした。

$$\left(t_{IFS} + t_{RTT} - \alpha\right) > 0$$

(4) 許容伝播遅延時間

前述（3）では、**図3**で例示したGWとECU間の伝播遅延時間にて期待応答時間を示した。実際の車両では、この伝播遅延時間は、ECUの配置やハーネスの形態などによって、様々な値を取り得る。また、複数ノードが接続されたバスに対し、本方式を用いる場合には、任意の2ノード間の伝播遅延の内、最悪の値（$t_{RTT\max}$）が考慮されるべきである。

ここでτを、システム内の全てのノードで同じ値をもち、伝送路遅延の許容値に基づいて設定される値として定義する。当然のことながら、τは設計上、伝送路遅延の許容値よりも大きな値で設定される。

$$\tau > \left(\frac{t_{RTT_{\max}} + |\alpha|}{2} \right)$$

(5) 衝突判定アルゴリズム

前述（3）（4）より、応答待ち状態においてSOF検出した時間tに対する判定ルールを**表1**の通り定義する。

表1　衝突判定条件

No.	条件式	判定結果
①	t < 7 [bit times]	衝突あり
②	7 [bit times] ≦ t < *rtout*	正常受信
③	*rtout* ≦ t	応答受信なし

※ *rtout* = EOF + IFS + 2τ

(6) 再送信待ち時間

表1の条件式に基づいて、フレームの衝突が検出された場合、GWからフレーム（Msg.G1）の再送信を行う必要がある。この時、次の送信は衝突することなく、必ず送信を終了できることを保障するために、ノード毎に異なる再送信待ち時間を設定する。

衝突回避の手法として、衝突発生後、ランダムな待ち時間を設けて再送信を試みる手法が一般的に用いられる。しかしながら、この手法では、2度目も衝突してしまう可能性が残るため、最悪遅延時間の保障が求められる車載用プロトコルとしては適さない。

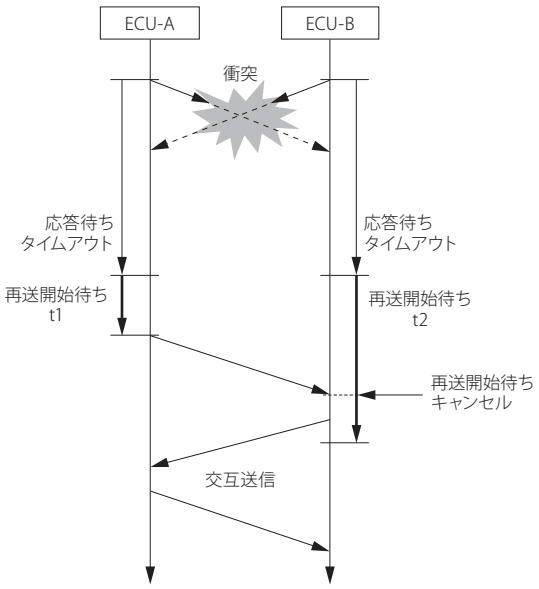


図4　非対称な再送待ち時間

提案する手法では、ノード毎に異なる再送信待ち時間（**図4**中、t1, t2）を設定し、2度目の衝突を起こすことなく、再送信が開始されるようにする。

最小の再送待ち時間の設定は、ノード毎にユニークなIDなど、車載ECUが静的に持つ情報に基き算出する手法とした。但し、再送待ち時間の最小単位は、応答待ち時間と同様に、ノード間で生じ得る同期誤差を考慮して決定されるべきであり、同一バス上の各ノードで、統一する必要がある。

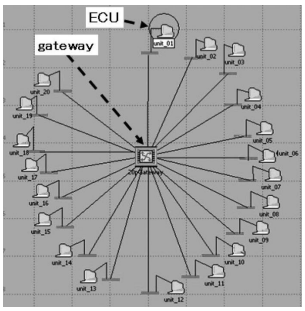
4.　提案手法の評価

本稿で提案する手法に対しては、その仕様の妥当性・十分性を検証するために、①モデル検査、②シミュレーション、③FPGAによる試作、の3つのアプローチによる評価を行った。

4－1　モデル検査　本提案方式の振る舞いは、鵜飼ら[4]により、モデル検査[5]、[6]の手法を用いて網羅的に解析する試みがなされた。仕様の振る舞いは、伝送路におけるビットエラーの発生も考慮したモデルがSMV言語で記述され、モデル検査ツールNuSMV[7]を用いて検査が行われた。

ここから得られた反例を解析することで、仕様の不具合・不足などの有益な情報を得ることができ、開発プロセスの早期段階において、仕様の完成度を高めることができた。

4－2　シミュレーションによる評価　本提案方式の倉地ら[8]による最悪遅延に関するシミュレーションの一例として、GWを介して20ノードが接続された状態におけるシミュレーション結果を紹介し、提案手法を用いることで、ノード間の送信遅延が、一定レベル以内に収まることを示す。



Simulation parameters of 10Mbps CAN.

Parameters	Status
Number of ECUs	20
Number of gateways	1
Size of message queue	200 messages
Data rate	10 Mbps
Simulation time	10 seconds

図5　提案手法のシミュレータ構成

(1) シミュレーション環境

提案手法のシミュレーションは、OPNET Modeler[9]を用いて、**図5**に示す構成で実施された。

このシミュレーションでは、提案手法を既存のCANバスに順次導入・展開する場合を想定し、GWを介して提案手法のバスを拡張する構成を想定している。シミュレーション時に送信するフレームは、実際の車両で使われるメッセージセット（ID、データ長、送信周期）を使い評価を実施した。また、シミュレーションモデルは、1ビット毎の振る舞いを表現できるようモデル化されており、ほぼ実際のネットワークで起こり得る現象を再現可能な環境となっている。

(2) シミュレーション・シナリオ

最悪遅延時間を評価するために、シミュレーション開始と同時に、各ノードからGWに対する送信が開始され、あるECUに対する出力が最も遅滞するシナリオとしている。シミュレーションからは、GWに蓄積されたフレームが、どれだけ効率よくポートから出力可能であるかという点が評価できる。

(3) 評価指標

シミュレーション結果の評価は、遅延率を用いて行った。遅延率は、各送信メッセージに設定された送信周期に対する遅延時間で表わされる。

$$遅延率=\left(\frac{メッセージの遅延時間}{メッセージの送信周期}\right)\times 100\%$$

(4) 評価結果

評価結果を**表2**に示す。

シミュレーションの結果、最も中継が混雑するノード宛のGW出力ポートにおいて、最大181メッセージがキューイングされた。シナリオ内での結果に限定されるが、遅延率の最悪値は、ボーレート10Mbpsにおいて1.75％と、良好な効果が確認された。更にメッセージ量を10倍に増やしたシミュレーションにおいても、16.76％と十分な遅延率に抑えられることが確認できた。

表2　遅延率評価結果（最悪値）

No.	方式	ボーレート	トラフィック量	遅延率
1	スケーラブルCAN	10Mbps	×1	1.75％
2	スケーラブルCAN	10Mbps	×10	16.76％

4－3　FPGAによる試作

(1) 試作の概要

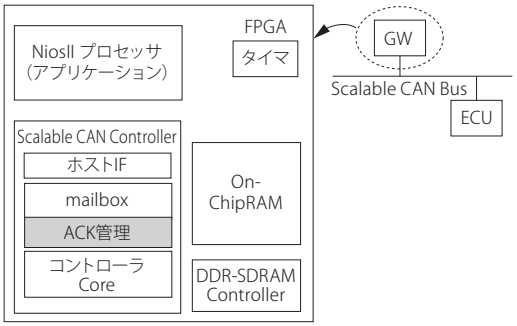
本提案方式は、倉地ら[8]によってFPGA上への実装が行われ、10Mbps設定での評価が行われた。FPGA上への実装仕様を**図6**に、提案手法を実装したコントローラのリソース消費の概要を**表3**に、評価の様子を**図7**に示す。

このデザインの主要なコンポーネントであるNiosII プロ

表3　テストボードにおける合成結果

	ECU	GW
Total logic elements	4,251	5,200
Total pins	81	85
Total memory bits	48,384	48,512

（内、Scalable CAN Controllerは949［LE］）



評価環境
NiosII Development Kit, Cyclone II Edition

コンポーネント
NiosII プロセッサ, タイマ, オンチップRAM,
DDR-SDRAMコントローラ, Scalable CANコントローラ

図6　FPGAによる試作環境

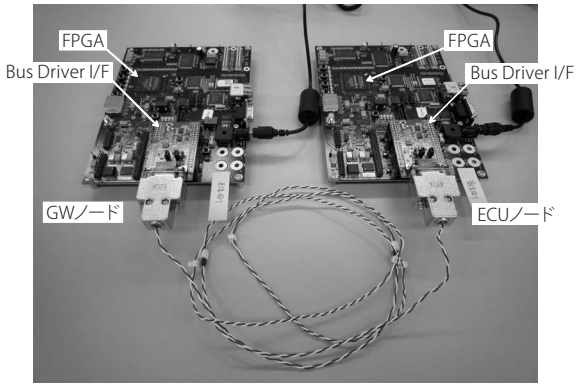


図7　評価の様子

セッサとスケーラブルCANコントローラはそれぞれ、スケーラブルCANコントローラのホストプロセッサ、提案手法を実現するネットワークコントローラとして実装されている。スケーラブルCANコントローラの機能は主にフレームのデコードやエンコード、ビットスタッフなどを行うことであり、このコンポーネントは**図6**に示すような内部モジュールで構成される。また、コントローラ実装における従来CANとの違いは少なく、新しいACKシーケンスや提案する衝突解決アルゴリズムを、プロトコルのステートマシーンとして実装している点などである。

(2) 試作結果

本実装に対して、①従来CAN コントローラとの互換性と、②コントローラの処理能力を評価した。

一つ目の評価項目である従来CANとの互換性は、その制御レジスタ構成を比較することで実施された。その結果、送信待ちのためのバックオフ時間の優先度設定ビットを追加している以外は従来CAN コントローラと同等であり、従来CANとはコントローラの初期化処理のみの変更で互換性を確保できていることを確認した。

二つ目の評価項目である処理能力は、実際のコントローラ動作を確認するために、AUTOSAR[10] COMスタック及びその上で動作するデモアプリを作成し、その処理能力を確認した。ここで作成されたAUTOSAR COMスタックはCANの仕様で定義されるCAN Driver、CAN Interface、COM、PDU Router、および、デモアプリで構成され、PDU Routerはゼロコストオペレーションで実装された。その結果、10Mbps設定において、約7Mbpsの伝送が可能であることを確認した。

5. 提案手法の課題

提案手法を、実際の車両環境に適用可能とするまでには、複数ノードがバス型に接続された形態を実現可能とすることが必要であり、①フレームフォーマットの見直し、②送信順序管理手法の規定、③複数ノード間でのACK/NAK情報を管理する方式、などに対する検討・開発が必要である。

6. 結　　言

本稿では、従来CANにおける様々な制約を緩和し、高速化を可能とするためのプロトコル的な対策を提案した。その上で、これら提案手法が、アプリケーション間の最悪遅延を一定のレベル以内に抑えることが可能であることを確認し、FPGAによる試作によって、ボーレート10Mbpsにおいて、約7Mbpsの伝送能力を発揮できることを確認した。また、開発過程においては、仕様策定フェーズにおいて、モデル検査とネットワークシミュレータによる検証を行い、開発前工程における品質確保手法として、その有効性を確認できた。

今後は、より車載環境に近い条件において、仕様の検証、シミュレーションと、試作による動作検証を行うとともに、車両走行環境下における信号特性の評価を進めていく。

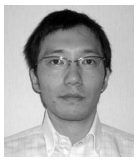
なお、本研究の成果は、名古屋大学と株式会社オートネットワーク技術研究所との共同研究で得られたものである。

参　考　文　献

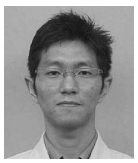
(1) LIN Consortium, http://www.lin-subbus.de/
(2) International Organization for Standardization, Road vehicles . Controller area network (CAN). Part1: Data link layer and physical signaling, ISO IS11898-1 (2003)
(3) FlexRay Consortium, http://www.flexray.com/
(4) 鵜飼謙児、坂部俊樹、高田広章、倉地亮、酒井正彦、草刈圭一朗、西田直樹、「電子情報通信学会技術研究報告. SS」、ソフトウェアサイエンス108 (242)、61-66、20081009
(5) Huth,M. and Ryan,M : Logic in Computer Science : Modelling and Reasoning about Systems, Cambridge University Press, 2nd ed., 2004, Chapter 3, pp.172-255.
(6) Clarke,E.M., Grumberg,O and Peled,D : Model Checking, MIT Press, (2000)
(7) NuSMV home page, http://nusmv.irst.itc.it/
(8) 倉地亮、高田広章、手嶋茂晴、宮下之宏、「10MbpsCAN プロトコルの設計と評価」、情報処理学会論文誌、Vol.50 No.11、pp. 2643-2653 (2009年11月)
(9) OPNET Modeler, http://www.opnet.com/
(10) AUTOSAR. http://www.autosar.org/

執　筆　者

倉地　亮*：名古屋大学　大学院情報科学研究科
附属組込みシステム研究センター
研究員
車載ネットワークシステムに関する研究に従事



西村　政信*：㈱オートネットワーク技術研究所
ネットワーク研究部
車載ネットワークシステムの開発に従事



高田　広章：名古屋大学　大学院情報科学研究科
附属組込みシステム研究センター　センター長　博士（理学）
手嶋　茂晴：名古屋大学　大学院情報科学研究科
附属組込みシステム研究センター　特任教授　博士（工学）
宮下　之宏：㈱オートネットワーク技術研究所　ネットワーク研究部
堀端　啓史：㈱オートネットワーク技術研究所　ネットワーク研究部
主任研究員
山本　秀樹：㈱オートネットワーク技術研究所　ネットワーク研究部　室長
夏目　晃宏：㈱オートネットワーク技術研究所　ネットワーク研究部　部長

＊主執筆者