M_026

制御ネットワークの段階的詳細化設計のUML記述
UML Description of Stepwise Refinement Design of Control Networks

白石 崇† 小林 憲貴† 山崎 亮介† 吉田 紀彦†
Takashi Shiraishi Kazutaka Kobayashi Ryosuke Yamasaki Norihiko Yoshida

1. はじめに

制御に用いられるネットワークは，高い信頼性，リアルタイム性，堅牢性が求められるため，高度なプロトコルを持ち，その設計は困難である．そこで，その生産性向上に向けて，本研究ではシステムLSIの分野で提唱されているシステムレベル設計手法の適用を試みた．

システムレベル設計とはSoC（System-on-a-Chip）を高い抽象度で，ハードウェアとソフトウェアを協調して設計する設計方法である．高い抽象度によって，無関係の詳細事項をモデルから取り除いて設計判断をすることによって生産性を上げる．システムレベル設計の手法としてSpecC方法論がある．SpecC方法論とは，モデルを仕様モデルから実装モデルに段階的に変換していくシステムレベル設計方法論である．

本研究では，車載ネットワーク規格のFlexRay[4][5]のプロトコルを使った通信仕様を例として用いた[3]．近年の自動車の高機能化，インテリジェント化にともなって，70以上のECU（Electoronic Control Unit）を搭載した車種が市場に出てきている．その多くのECUを分散制御するために，車載ネットワーク技術が発達し，CAN, LINといったプロトコルが使われている．FlexRayはそのCANの置き換えやX-by-wireをターゲットとして作られた先端的なプロトコルである．

また，本研究では表現に，OMG[6]が定めた統一された図式の言語である，UML2.0[7]を用いた

本論文では，2.で，SpecC方法論について，3.でFlexRayについて説明し，4.でUML2.0を用いてFlexRayの段階的詳細化の例を示し，最後に5.で本論文のまとめと今後の課題を述べる．

2. SpecC方法論

SpecC方法論とは，システムの仕様モデルから，ソフトウェアとハードウェアからなる実装モデルに段階的に変換していくシステムレベル設計方法論である．SpecC方法論の設計フローには，仕様モデル，アーキテクチャモデル，通信モデル，実装モデルの4つのモデルと，そのモデルを変換するアーキテクチャ探索，通信合成，バックエンドプロセスの3つの変換がある．

特に，本研究に関連する通信合成について述べると，通信合成では，アーキテクチャモデルの部品間の抽象的な通信をプロトコルの挿入，トランスジューサの挿入，プロトコルの埋め込みによって洗練し，結線の上の実際の実装と，システムバスのプロトコルに変換する．

3. FlexRay

3.1 プロトコル概要

FlexRayのプロトコルは遅延の生じにくいタイム・トリガである．設計者によって，あらかじめメッセージ送信についてスケジューリングされ，受信するノードも決められている．

FlexRayのメディア・アクセス方式は，上位からコミュニケーション・サイクル・レベル，アービトレーション・グリッド・レベル，マクロティック・レベル，マイクロティック・レベルの4階層で構成される．本論文では，アービトレーション・グリッドレベルまでの上位2レベルを対象とした．

3.2 ノード・アーキテクチャ

FlexRayの1つのノードは，ハードウェア的にコントローラ部とドライバ部から構成される．コントローラ側には，アプリケーションが実行されユニット全体を制御するホスト，通信を制御する通信コントローラが存在する．ドライバ部には，送信検出異常と通信遮断を実施するバス・ガーディアン（搭載しない場合もある）と，FlexRayバスと通信コントローラの間で物理的信号と論理的信号の変換を行うバス・ドライバが存在する．バス・ドライバはチャネル一つに対して一つ存在する．

4. FlexRayの段階的詳細化のUML記述

本研究では，SDL(Specification and Description Language)[8]のように，構造的な面と動作的な面からモデリングをおこなった．構造的な面はシステムの構成要素の関係を表し，システムレベル設計における動作とチャネルのつながり方に対応する．モデルの構造的な面を表すために，本研究ではコンポジット構造図を用いる．一方、動作的な面はシステムの活動や反応を表し，システムレベル設計における動作とチャネルがどのように相互作用するかに対応する．モデルの動作的な面を表すために，本研究ではアクティビティ図を用いる．以下で4段階からなるFlexRayの段階的詳細化設計の概要を述べる．今回は簡単のため静的セグメントのみを考える．

4.1 第1段階

第1段階では，アプリケーションが必要なメッセージを考え，メッセージの受け渡しをするだけのモデルを作る．

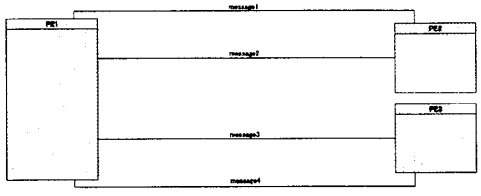

図1: 始めのモデル

図1はモデルの構造を表したコンポジット構造図である．「PE1」のような全体クラスは，プロセスエレメント(PE)を表し，PEとPEと結ぶコネクタはその間にメッセージの通信があることを表すこととする．

†埼玉大学 Saitama University

ここでは，送信側のPE1と受信側のPE2，PE3の3つのプロセスエレメントが4種類のメッセージをやりとりすることを想定する．簡単のため，全ての通信においてPE1は送信側とし，PE2はmessage1とmessage2において，PE3はmessage3とmessage4においてそれぞれ受信側とした．この段階のモデルは，PE1からPE2，PE3に対して送るべきメッセージをただ渡すだけである．

4.2 第2段階

第2段階では，第1段階のモデルを洗練して，フレーム形式でメッセージを送るモデルを作る．FlexRayの静的セグメントにおいて，設計者が決めた一定の大きさのフレームで通信が行われる．1フレームに収まらないメッセージを通信する場合はメッセージをフレームサイズに分割し，一方，1度に多種の小さなメッセージの通信を行う場合は1フレームに多種のメッセージをまとめて送信し，受信側はそのフレームをメッセージに再構成する必要がある．

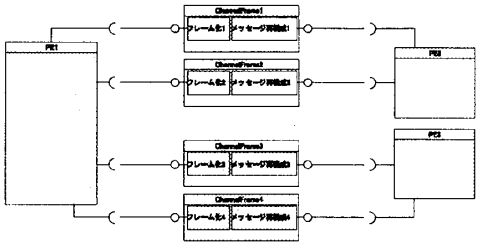

図2: フレームによる通信のモデル

図2は第1段階のモデルを洗練したフレームで通信をおこなうモデルのコンポジット構造図である．第1段階で全体クラスはPEを表すこととしたが，提供インターフェースの繋がった全体クラスはチャネルを，提供インターフェース自体はチャネルのインターフェースを表す．また，提供インターフェースに繋がった要求インターフェースはそのインターフェースを使用するプロセスエレメントを表すこととする．ここでは「ChannelFrame1」などのチャネルがフレームの通信を行う．また，全体クラス内のパートはその全体クラスが表す動作のもつ処理やチャネルの持つ処理を表すこととする．モデルではチャネル内にメッセージをフレーム化する処理と，そのフレームからメッセージを再構成する処理を持つことを示している．

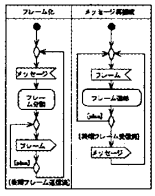

図3: フレームによる通信のチャネルの振る舞い

図3はモデル中のチャネルの振る舞いを表すアクティビティ図である．図はパーティションに分かれているが，「フレーム化」のパーティションがフレームを送信する側のチャネルの振る舞いを，「メッセージ再構成」のパーティションがフレームを受信する側のチャネルの振る舞いを表す．このモデルでは，フレーム化では，メッセージを受け取ると分割して，フレームを順に送っていき，メッセージ再構成では，受け取ったフレームを連結しメッセージにして送り出していることがわかる．また，フレームを送る時間間隔などは示されていない．

最後に，フレームのチャネルをプロセスエレメントに埋め込むために，フレーム化の処理をPE1に，メッセージ再構成の処理をPE2とPE3に埋め込む．

4.3 第3段階

第3段階では，後の段階で時分割の通信にするために通信のスケジューリングを行うモデルを作る．ここで，通信のスケジューリングは各ノードのフレーム送受信の時間を決めることである．このモデルでは，コミュニケーション・サイクルのたびに1度フレームを送受信する．この段階でコミュニケーション・サイクルの長さが決まる．これによって，スロットごとの送信周期が決まるので，時間制約が満たせるかどうかもおおよそ見積もれる．

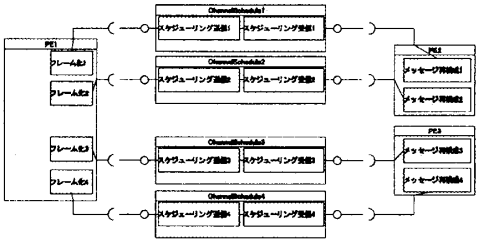

図4: スケジューリングする通信のモデル

図4は，通信のスケジューリングを行ったモデルを表す．この図から，スケジューリングの機能をもったチャネルが間に挿入されることがわかる．

図5は，スケジューリングされた通信を行うチャネルの振る舞いを表すアクティビティ図である．「スケジューリング送信」のパーティションがフレーム送信側の，「スケジューリング受信」のパーティションがフレーム受信側のチャネルの振る舞いを表す．スケジューリング送信には，サイクルカウンタがあり，時間イベントアクションによってスロットごとに一定時間を待つ．これによって，スケジューリング送信はコミュニケーション・サイクルごとに1度フレームを送信していることを表現する．それに対応してスケジューリング受信もコミュニケーション・サイクルごとに1度フレームを受信している．

4.4 第4段階

第4段階では，スケジューリングをもとに，時分割でアービトレーションをする通信をするモデルを作る．FlexRayでは，決められた時間にフレームを送信し，送信されたフレームは自分を含めた全てのノードで受信され，そのフレームを必要とするノードのみがフレームを受け取り，メッセージを再構成する．

このモデルでは，スロットごとに分かれていたチャネルが1つにまとめられる．送信側では，チャネル内でスロットごとに分けた通信をするために送信畳み込みの処理でフレームを送信する順序を決める．また，受信側で

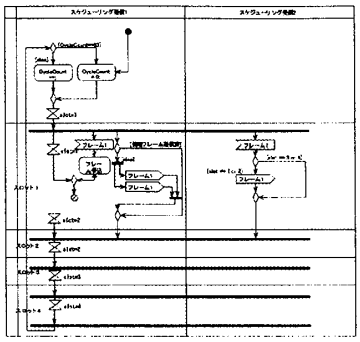

図 5: スケジューリングするチャネルの振る舞い

は，受信選択の処理でそのスロットに対応するフレームのみを受信するように選択する．

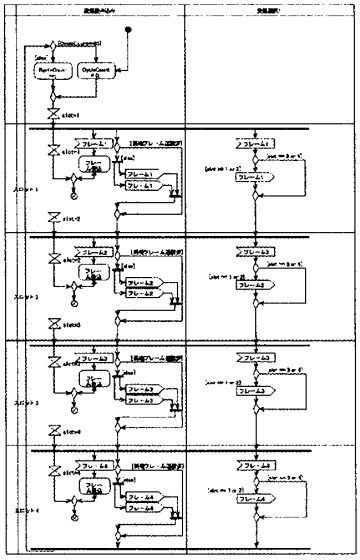

図 6: アービトレーションするチャネルの振る舞い

図 6 は，アービトレーションするチャネルの振る舞いを表す．スロットごとに同期して送受信を行い，フレームを 2 つのノードに送っていることがわかる．FlexRay では自身を含めたすべてのノードにフレームが送られるが自身への送信はここでは省略している．

最後に，アービトレーションするチャネルをプロセスエレメントに埋め込む．図 7 は，埋め込み後のモデルを表す．ここでは，既存のプロセスエレメントに埋め込むのではなく，新たに作成した通信コントローラを想定したプロセスエレメントにチャネル内の機能を割り当てる．

このようにして，初めはアプリケーションのメッセージの送受信という形だったモデルを，フレームの送受信という形にした後，コミュニケーション・サイクルにもとづいた時間でのフレームの送受信という形にして，最終的に，スケジューリングをもとに時分割してフレームをマルチキャストするモデルにする．

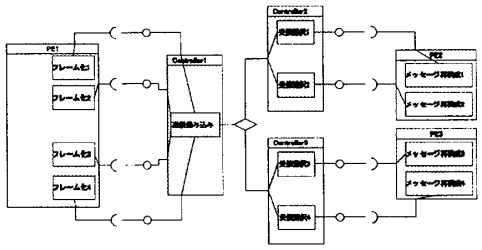

図 7: アービトレーションを行うチャネルの埋め込み

5. おわりに

本研究では，UML2.0 を用いて FlexRay を 4 段階に分けて詳細化した．第 1 段階でアプリケーションが必要なメッセージを通信するモデルを考え，第 2 段階でフレームで通信するモデルを考え，第 3 段階でスケジューリングに基づいて通信するモデルを考え，第 4 段階で時間でアービトレーションを行って通信するモデルを考えた．

段階的に考える場合，その段階ごとのモデルが成り立たなければならない．フレーム化とメッセージ再構成の処理が対応しているように，通信は送信側と受信側が対応しなければならないので，通信の処理は送信側と受信側に分かれてそれぞれ決まった側の部品に割り当てられる必要があった．また，分散システムでは間のネットワークを占有しないためにフレーム化（パケット化）することが必要である．さらに，本研究では，タイムトリガの FlexRay を取り上げたが，イベントドリブンの CAN などを取り上げる場合でもイベントをトリガとしたアービトレーションや，時間制約を考えるためのスケジューリングが必要である．以上のような理由で，本研究のような段階分けは他の通信にも適用可能である．

今後の課題としては Action Semantics を利用した詳細な UML モデルを考案することが挙げられる．また，FlexRay についてエラー処理や，クロック同期の仕組みや，スタートアップなど，より細かな設計について考えることと，他のプロトコルについても段階的に詳細化できるかを確かめることが挙げられる．

参考文献

[1] Andreas Gerstlauer, Rainer Domer, Junyu Peng, Daniel D. Gajski，(木下 常雄 訳)，“システム設計 SpecC による実現”，SpecC Technology Open Consortium，2002.

[2] Daniel D. Gajski, Jianwen Zhu, Rainer Domer, Andreas Gerstlauer, Shuqing Zhao,(木下 常雄，富山 宏之 訳)，“SpecC 仕様記述言語と方法論”，CQ 出版社，2000.

[3] 白石崇，“システムレベル設計における通信仕様の段階的詳細化”，埼玉大学卒業論文，2006.

[4] FlexRay Consortium，“FlexRay Communication System Protocol Specification Version 2.1”，http://www.flexray.com/

[5] 佐藤道夫，“DesignWave12 月増刊号 車載ネットワーク・システム徹底解説”，CQ 出版社，2005.

[6] Object Management Group, http://www.omg.org/

[7] Object Management Group，“Unified Modeling Language: Superstructure version 2.0”，2005.

[8] 若原恭，長谷川晴朗，“仕様記述言語 SDL”，カットシステム，1996