Vol.2012-SE-176 No.8
Vol.2012-EMB-25 No.8
2012/5/21

組込みソフトウェアの系統的な設計と検証 —状態遷移設計と SPIN の適用—

望月祐希† 野口健一郎††

本研究では、セパレーション・カーネル方式の OS 上に、交通管理システムの信号制御機を行う組込みソフトウェアを状態遷移設計と SPIN を用いた系統的な設計と検証を通して試作した．これにより、信頼性の高い組込みソフトウェアを容易に作成できた．

Systematic Design and Verification of an Embedded Software —Application of State Transition Design and SPIN—

YUUKI MOCHIZUKI† KENICHIRO NOGUCHI††

We developed an embedded software, i. e. traffic signal control software, through systematic design and verification, applying state transition design and model check tool SPIN. We could obtain a reliable embedded software without trouble during implementation.

1. はじめに

本研究室では、安全な組込みシステムを実現するためにセパレーション・カーネル方式ベースのオペレーティングシステム OS-K を試作中である[1, 2, 3, 4]．OS-K が組込みシステムに適用できることを実証するために、OS-K を用いて交通管理システムの信号制御機を行う組込みシステムを試作した．この仕様設計を、状態遷移設計を適用して形式的に行い、またモデル検査ツール SPIN を用いて仕様検証を行った．この結果、実装が容易にでき、また実装時のバグ発生も無かった．

2. 背景

(1) セパレーション・カーネル方式

セパレーション・カーネル方式は、John Rushby によって提案された、高セキュリティを実現する OS 構成方式である[5]．複数の独立したパーティション空間とパーティション間の通信路を提供する OS 構成により、単一のプロセッサ上で仮想的な分散環境をシミュレートする．

この方式による OS は、特に強固なセキュリティが必要な組込みシステムへ適用される．米国 National Security Agency はセパレーション・カーネル方式の製品が高い頑健性を持つために満たすべき基準である Separation Kernel Protection Profile[6]を規定している．

(2) SPIN

SPIN は、Gerard J.Holzmann によって開発された、仕様のモデルの正しさを検証するためのツールである[7]．モデルは、Promela（Process Meta Language）言語によって記述する．

3. プラットフォーム：組込み用 OS「OS-K」

3.1 セパレーション・カーネル方式の採用

OS-K は、安全な組み込みシステムを実現するために試作中のセパレーション・カーネル方式をベースにしたオペレーティングシステムである[1, 2, 3, 4]．OS-K の概要を図 1 に示す．

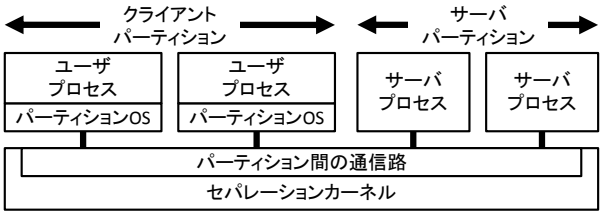

図 1 OS-K の概要

Figure 1 Abstract of OS-K.

3.2 セキュリティ機能

OS-K は以下のセキュリティ機能を持つ．

① 独立したパーティション空間

CPU の保護機能を用いて独立したパーティション空間を実現する．また、構成ベクタに記述した情報を基に、プロセスやメモリをパーティションに割り当てる．

② パーティション間の通信制御

構成ベクタに記述したメッセージ通信制御情報を基に、パーティション間のメッセージ通信制御を行う．

③ 全てのファイルの暗号化

IPL（Initial Program Loader）を除き、カーネルロードモジュールなどの OS に必要なファイルを全て暗号化する．パーティションが扱うファイルの暗号化は一つ一つ選択す

† 神奈川大学 (現在 日立情報通信エンジニアリング(株))
Kanagawa University (Hitachi Information & Communication Engineering, Ltd.)
†† 神奈川大学
Kanagawa University

Vol.2012-SE-176 No.8
Vol.2012-EMB-25 No.8
2012/5/21

ることができる.

④ ロードモジュールの完全性検証

OS 起動時に、暗号化した OS に必要なファイル全てについて完全性検証を行う.

⑤ 形式手法による仕様検証

これらの機能と Separation Kernel Protection Profile[6]が規定するセパレーション・カーネルへの要求との対応関係を表 1 に示す.

表 1　OS-K のセキュリティ機能と SKPP の主な要求

Table 1　Security features of OS-K and requirements of SKPP.

OS-K のセキュリティ機能	対応する SKPP の主な要求
独立したパーティション空間	資源の隔離
パーティション間の通信制御	情報の流れ制御
全てのファイルの暗号化	信頼された配達
ロードモジュールの完全性検証	信頼された初期化
形式手法による仕様検証	形式的な仕様記述

3.3 OS API

(1) セパレーション・カーネルの API

OS-K のセパレーション・カーネルは、次のセパレーション・カーネルコールをパーティション OS とサーバパーティションに提供する.

・ send

宛先パーティションにメッセージを送信し、応答メッセージを待つ.

・ receive

メッセージを受信する. メッセージが無い場合は、受信するまで待つ.

・ reply

受信したメッセージの送信元パーティションへ応答メッセージを送信する.

・ notify

宛先パーティションに通知メッセージを送信する.

・ sleep

指定した時間スリープする.

・ get_time

現在の日時情報を取得する.

(2) パーティション OS の API

今回の実験用の版は次の API を提供する.

・ send、receive、reply

セパレーション･カーネルの API と同等の機能を持つが、クライアントパーティション間のメッセージ通信に限られる.

・ sleep、get_time

セパレーション･カーネルの API をと同等の機能を持つ.

・ terminal_input

ターミナルエミュレータから入力する.

・ terminal_output

ターミナルエミュレータに文字列を出力する.

4. 組込みソフトウェア：信号制御機ソフトウェア

4.1 概要

信号制御機ソフトウェアの試作は、仕様設計、仕様検証、実装の順で行った. 流れを図 2 に示す.

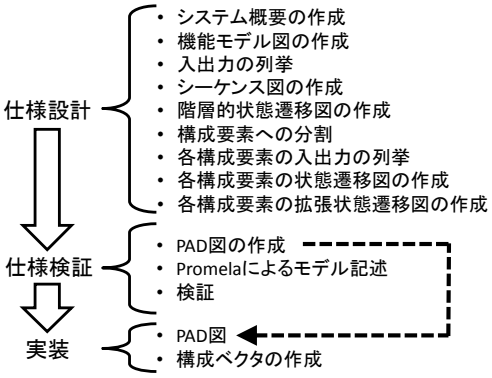

図 2　信号制御機ソフトウェアの試作の流れ

Figure 2　Flow at prototype of software of signal controller.

仕様設計には「ソフトウェアの論理的設計法」[8]に基づいて状態遷移設計を、仕様検証には SPIN を適用した.

4.2 状態遷移設計による仕様設計

(1) システム概要

交通管理システムは、1 つのサーバと複数の信号制御機から構成する. サーバは、管理者 A が操作し、全ての信号制御機を管理する. 信号制御機は、管理者 B が操作し、4 つ角交差点の 4 つの信号灯器（青、黄、赤を表示する）を制御する. 制御方法には、自動制御と手動制御がある.

(2) 機能モデル図

機能モデル図[9]により、インターフェースの位置を明確にした. 機能モデル図を図 3 に、信号灯器の位置を図 4 に示す.

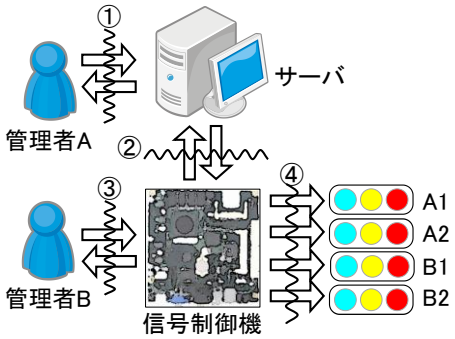

図 3　機能モデル図

Figure 3　Function model diagram

図 3 に示す 4 つのインターフェースは、以下である.

① サーバ管理者インターフェース

② 通信インターフェース

③ 信号制御機管理者インターフェース

④ 信号灯器インターフェース

Vol.2012-SE-176 No.8
Vol.2012-EMB-25 No.8
2012/5/21

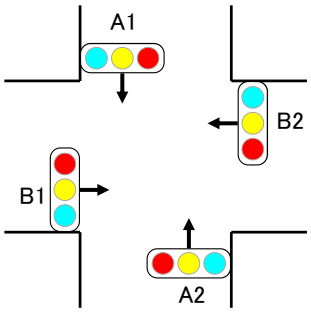

図 4　信号灯器の位置

Figure 4　Position of signals.

(3) **入出力**

図 3 の②③④のインターフェースを以下のように設計した.

②通信インターフェース

入力:制御方法

出力:サーバ操作可、サーバ操作不可、制御状態、信号状態

③信号制御機管理者インターフェース

入力:操作開始、操作終了、制御方法

出力:サーバ操作可、サーバ操作不可、制御状態、信号状態

④信号灯器インターフェース

入力:なし

出力:青、黄、赤

(4) **シーケンス図**

入出力の関係をシーケンス図により記述した. シーケンス図の例として、信号制御機の起動後に信号灯器を全て赤に初期化し、各状態をサーバと管理者 B に通知する流れを図 5 に示す.

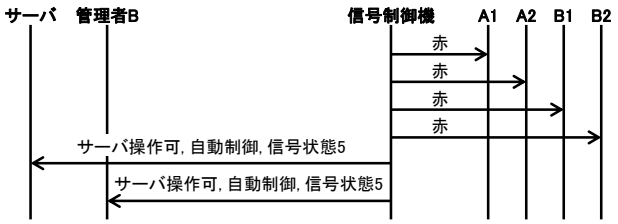

図 5　起動時のシーケンス図の例

Figure 5　Example of sequence diagram at start-up.

(5) **階層的状態遷移図**

シーケンス図を基に、階層的状態遷移図[10]を作成した. 一部を図 6 に示す. 内部状態数は 59 となった.

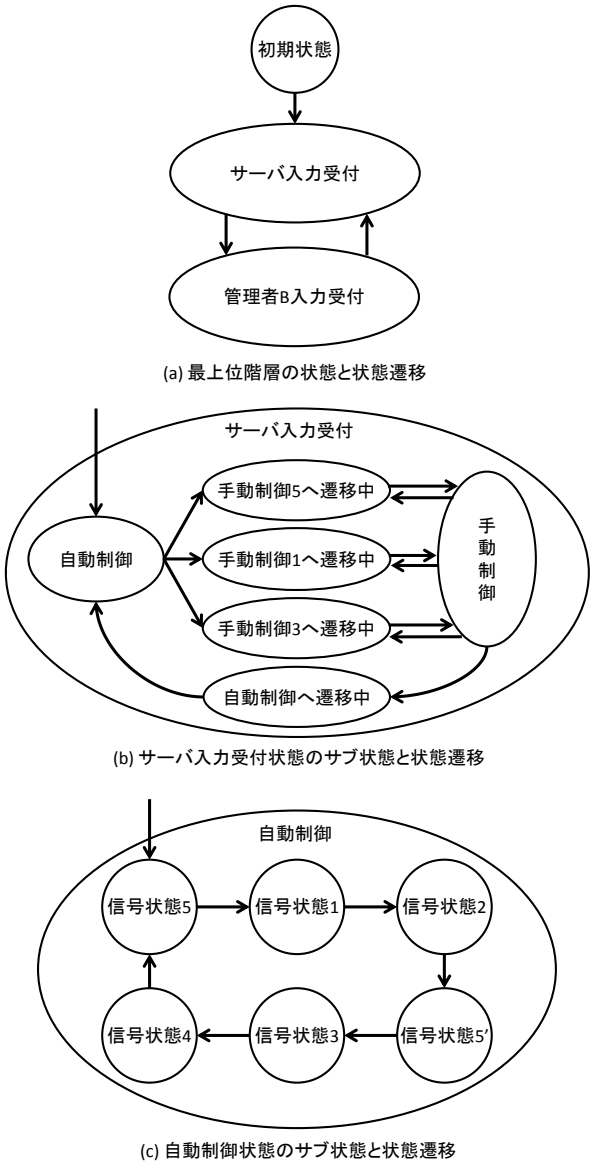

図 6　信号制御機の状態遷移図
（一部分のみ示す. なお入出力の記述は省略）

Figure 6　State transition diagram of signal controller (partly except I/O description.)

(6) **構成要素への分割**

内部状態数を減らすために、入力処理部、主処理部、自動制御処理部、手動制御へ遷移処理部、手動制御処理部、および自動制御へ遷移処理部の 6 つの内部構成への分割を行った. 分割した内部構成を図 7 に示す.

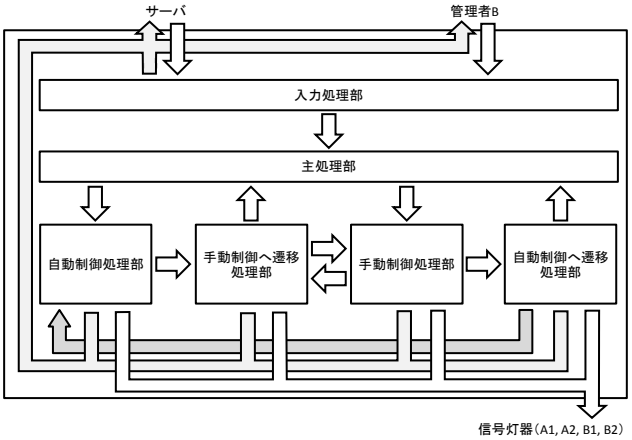

図 7　分割後の内部構成

Figure 7　Internal constitution after the division.

(7) **各構成要素の入出力**

内部構成を基に各構成要素の入出力を設計した．例として、自動制御処理部の入出力を次に示す.

① 主処理部とのインターフェース
入力：初期化、手動制御状態
出力：なし
② 自動制御へ遷移処理部とのインターフェース
入力：自動制御
出力：なし
③ サーバと管理者 B とのインターフェース
入力：なし
出力：自動制御、信号状態
④ 手動制御へ遷移処理部とのインターフェース
入力：なし
出力：手動制御状態と信号状態の組み合わせ
⑤ 信号灯器（A1, A2, B1, B2）とのインターフェース
入力：なし
出力：青、赤、黄

(8) **各構成要素の状態遷移図**

分割した構成要素ごとに状態遷移図を作成した．例として、自動制御処理部の状態遷移図を図 8 に示す.

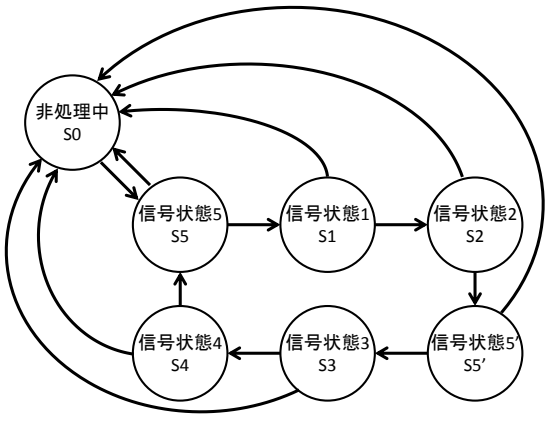

図 8　自動制御処理部の状態遷移図
（入出力の記述は省略）

Figure 8　State transition diagram of automatic control processing section (except I/O description.)

(9) **各構成要素の拡張状態遷移図**

更に状態数を減らすために、自動制御処理部、自動制御へ遷移処理部、および手動制御へ遷移処理部の状態遷移図に変数を導入して拡張状態遷移図[9]を作成した．例として、自動制御処理部の拡張状態遷移図を図 9 に示す.

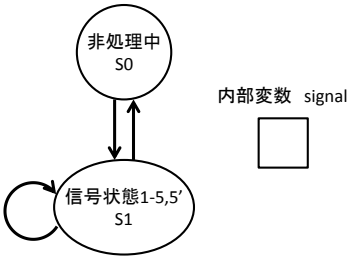

図 9　自動制御処理部の拡張状態遷移図
（入出力の記述は省略）

Figure 9　Extended state transition diagram of automatic control processing section (except I/O description.)

4.3 SPIN を用いた仕様検証

(1) **PAD（Problem Analysis Diagram）による記述**

仕様設計で作成した、状態遷移図と拡張状態遷移図に基づいて、各内部構成に対応して PAD 図を作成した.

(2) **Promela 言語によるモデル記述**

PAD 図に基づいて、Promela 言語によるモデルの記述を行った．なお仕様検証のために、サーバ、管理者 B、および信号灯器をそれぞれプロセスとして記述した．サーバと管理者 B のプロセスは、それぞれが入力処理部へ行う全ての入力を非決定的に行う. 信号灯器のプロセスは、A1、A2、B1、B2 の信号を受付け、標準出力に出力する.

(3) **検証と結果**

次の条件が満たされることについて検証を行った.

① 全てのプロセスがデッドロックに陥らない
② 各信号灯器が「青」から「黄」、「黄」から「赤」、「赤」から「青」へのみ遷移する
③ 表示方向が直行する信号灯器どうしが同時に「青」または「黄」にならない

ランダムシミュレーションによる検証では、モデルが無限に動作するため、ステップ数が 5000 のケースのみ行い、条件①②③が満たされることを確認できた.

検証器による網羅検証では、条件②と③を検証するために LTL（Linear Temporal Logic）式を用いた．例として、条件③を検証するための LTL 式を次に示す.

```
![]((a1_b || a1_y || a2_b || a2_y) -> (b1_r && b2_r))
```

網羅検証を行ったが、サーバと管理者 B のプロセスが無限にランダムな入力を行うモデルの場合、状態爆発が起こり検証することが出来なかった．そのため、ランダムな入力をそれぞれ 2 回行うモデルに変更することで、条件①②③が満たされることを確認できた.

図 10 と図 11 に、ランダムシミュレーションと網羅検証の結果例をそれぞれ示す.

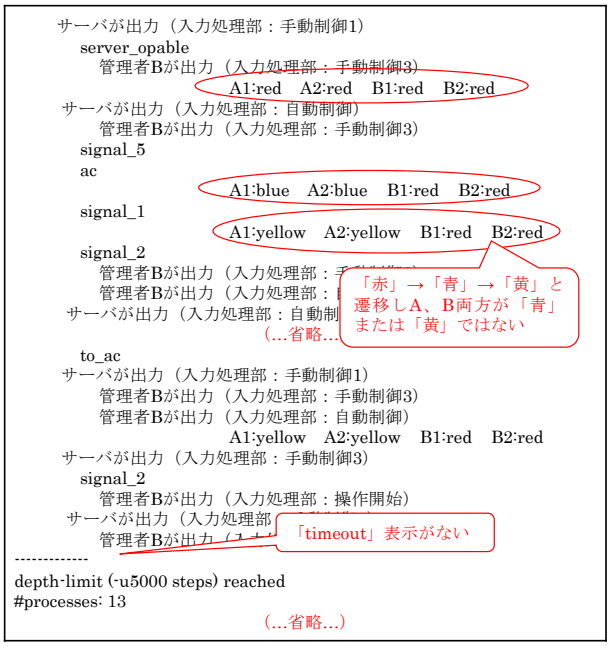

図 10 信号制御機のランダムシミュレーションの結果

Figure 10 Random simulation result of signal controller.

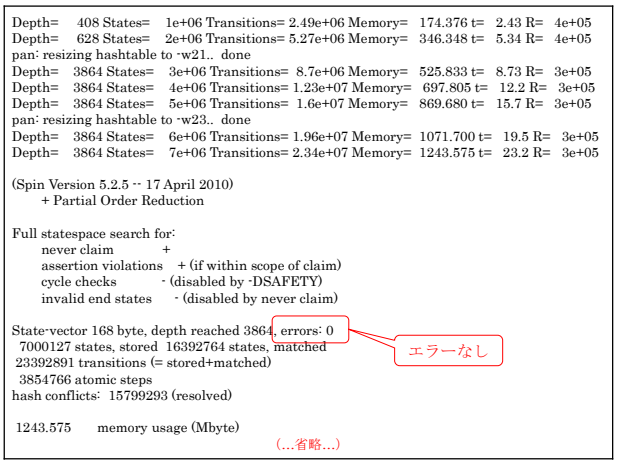

図 11 信号制御機の網羅検証結果

Figure 11 Coverall verification result of signal controller.

4.4 実装

(1) 信号制御機の構成

モデル記述のために作成した PAD 図を基に OS-K 上に図 12 に示す構成で実装を行った.

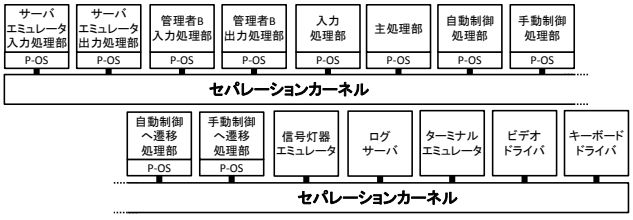

図 12 信号制御機の構成

Figure 12 Configuration of signal controller.

管理者 B のユーザインターフェース処理をクライアントパーティションで実装した. また、本研究の OS-K はネットワーク通信機能と信号灯器を持たないため、サーバと信号灯器をエミュレータとして実装した.

(2) 構成ベクタ

信号制御機の構成を基に人可読の構成ベクタを作成した. 人可読の構成ベクタの一部を図 13 に示す.

```
partitions {
    video_driver {
        max_quotas      : 0x00100000;
        type            : server;
        subject         : 6;
    }
                        (…省略…)
    main_proc {
        max_quotas      : 0x00100000;
        flow_to         : log_server, ac_proc, mc_proc;
        type            : user;
        subject         : 16;
    }
    ac_proc {
        max_quotas      : 0x00100000;
        flow_to         : log_server, signal, server_receive,
                          admin_b_receive, to_mc_proc;
        type            : user;
        subject         : 17;
    }
                        (…省略…)
}
resources {
    video_ram {
        partition       : video_driver;
        io_port         : 0x03D4, 0x03D5;
        memory          : {0xB8000, 0xFA0, rw};
    }
                        (…省略…)
}
```

図 13 人可読の構成ベクタ

Figure 13 Human-readable configuration vector.

(3) 処理

サーバエミュレータは、サーバに送られてくる制御状態をサーバエミュレータ出力処理部によりターミナルエミュレータに出力し、サーバから入力処理部へ送る制御方法をサーバエミュレータ入力処理部によりターミナルエミュレータから入力する.

管理者 B は、管理者 B に送られてくる制御状態を管理者 B 出力処理部によりターミナルエミュレータに出力し、管理者 B から入力処理部へ送る制御方法を管理者 B 入力処理部によりターミナルエミュレータから入力する.

入力処理部から手動制御へ遷移処理部までの内部機械は、仕様検証のために作成した PAD 図を基に実装した.

信号灯器エミュレータは、信号灯器の状態をターミナルエミュレータに出力する.

(4) 実装結果

信号制御機ソフトウェアを実装した OS-K を動作させた結果、正しく仕様通りに動作していることを確認できた.

信号灯器エミュレータの画面を図 14 に示す.

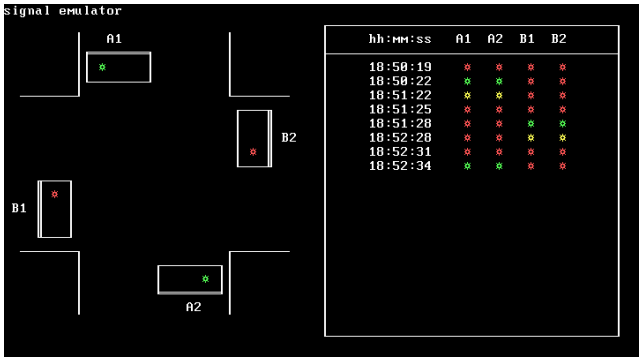

図 14　信号灯器エミュレータの画面

Figure 14　Screen of signal emulator.

5. 考察

構成ベクタにより、容易にパーティションの構成やパーティション間のメッセージ通信制御情報などを設定でき、OS-K に信号制御機ソフトウェアを構築することができた.

状態遷移図に階層的状態遷移図を用いることで、内部構成への分割や各内部構成の状態遷移図を容易に作成することができた.

ランダムシミュレーションによる検証で、仕様を修正することなく仕様が正しいことを確認した．しかし、検証器による網羅検証は、ゆるい条件のもとでは状態爆発が起こり、行えなかった．そのため、ある程度まで網羅検証を行えるように条件を付けることで、その条件の下では仕様が正しいことを確認した.

検証のために作成した Promela 言語によるモデル記述と、実際に実装した C 言語のコードの行数を比較すると、同程度のコード行数となった．比較を表 2 に示す.

表 2　Promela と C のコード行数

Table 2　Lines of Promela and C codes.

項目	行数（コメントを含む）	
	Promela	C
入力処理部	91	101
主処理部	81	87
自動制御処理部	172	172
手動制御処理部	125	139
自動制御へ遷移処理部	161	105
手動制御へ遷移処理部	102	169
合計	732	773

以上から、以下の事が言える.

・検証を行う前に形式的に設計を行うことが重要である
・モデルを完全に網羅検証することは難しい
・モデル記述に、実装と同等のコード行数が必要である
・モデル記述や LTL 式などによる検証する仕様の設定および網羅検証を行うための高度な技術が必要である

6. 今後の課題

今後の課題は、次のものなどがある.

・先行研究の OS-K のネットワーク機能[3]を統合し、サーバとのやり取りを実際のネットワークで行う
・実装の体系的なテスト
・より制約条件の少ない網羅検証

7. 結言

本研究では、セパレーション・カーネル方式の OS 上に、交通管理システムの信号制御機を行う組込みソフトウェアを状態遷移設計と SPIN を用いた系統的な設計と検証を通して試作した．これにより、信頼性の高い組込みソフトウェアを容易に作成できた．しかし、実装の体系的なテストを今後行う必要がある.

参考文献

[1] Kei Kawamorita, Ryouta Kasahara, Yuuki Mochizuki, and Kenichiro Noguchi, "Application of Formal Methods for Designing a Separation Kernel for Embedded Systems," *World Academy of Science, Engineering and Technology Issue 68*, pp. 506-514, July 2010.

[2] 望月 祐希, 野口 健一郎, “組込み OS 用暗号化ロードモジュール機能及びセキュアファイルサーバの試作,” *第 9 回情報科学技術フォーラム（FIT2010）*, 2010.

[3] 笠原 良太, 望月 祐希, 野口 健一郎, “組込み用 OS のネットワーク機能の設計と形式手法 SPIN による検証,” *第 10 回情報科学技術フォーラム (FIT2011)*, 2011.

[4] 望月 祐希, 野口 健一郎, “安全な組込み用 OS の試作―セパレーションカーネルに基づくセキュリティ機能の検討―,” *第 10 回情報科学技術フォーラム（FIT2011）*, 2011.

[5] John Rushby, “Design and Verification of Secure Systems,” *8th ACM Symposium on Operating System Principles*, pp. 12-21, 1981.

[6] Information Assurance Directorate, “U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness,” Version 1.03, Jun 2007.

[7] Gerard J. Holzmann, “The Model Checker SPIN,” *IEEE Transactions on Software Engineering*, 第 巻 23, 第 5, 5 1997.

[8] 野口 健一郎, ソフトウェアの論理的設計法, 共立出版株式会社, 1990.

[9] 野口 健一郎, 川守田 慶, “拡張状態遷移技法を用いた仕様検証の実験 ―組込み用 OS 試作への適用―,” *第 7 回情報科学技術フォーラム（FIT2008）*, 2008.

[10] 野口 健一郎, “計算機オペレーティング・システムの設計法,” *東京大学学位論文*, 1969.