高度交通システム　12－9
(2003. 3. 7)

SpecCを用いたハードウェア/ソフトウェア協調設計方式とITS安全運転支援システムによる評価

井上 聡† 遠藤 祐† 吉田 健† 飯田 庸介† 小泉 寿男† 清尾 克彦‡

† 東京電機大学大学院理工学研究科 〒350-0394 埼玉県比企郡鳩山町石坂
‡ 三菱電機株式会社 技術研究所 〒247-8510 神奈川県鎌倉市大船5丁目1番1号

E-mail: † {satoshi,yu,ken,iida}@itlab.k.dendai.ac.jp,koizumi@k.dendai.ac.jp, ‡ k-seo@hit.isl.melco.co.jp

あらまし ハードウェア/ソフトウェア協調設計においては，機能・性能を主条件としてハードウェアとソフトウェア間のトレードオフが行われている．また，莫大なソフトウェアを開発する組込みシステムでは，再利用の実現性が開発成否の鍵となる．本稿では，ハードウェア/ソフトウェア協調設計において、トレードオフの条件として，開発工数，処理時間，メモリ，ゲート数を取り入れ，再利用の手法として，再利用部品のリアルタイム評価を取り入れた協調設計方式を提案する．本方式を実現する方法として，システムレベル言語 SpecC を用いる．SpecC は C 言語にハードウェアを記述するための構文を追加した言語であり，ハードウェアとソフトウェアを単一の言語で記述ができる特徴を持つ．本方式を ITS(Intelligent Transport Systems)安全運転支援システムに適用し評価を行う．

キーワード ハードウェア/ソフトウェア協調設計，SpecC，トレードオフ，再利用，ITS，安全運転支援システム

A hardware/software co-design method with re-use of components using SpecC and its evaluation to ITS Safe driving support system

Satoshi INOUE† Yu ENDO†
Takeshi YOSHIDA† Yousuke IIDA† Hisao KOIZUMI† Katuhiko SEO‡

† Graduate School of Science and Engineering, Tokyo Denki University
Ishizaka, Hatoyama-mati, Hiki-gun, Saitama, 350-0394 Japan

‡ Mitsubishi Electric Corporation Technical training institute 5-1-1, Oohuna, Kamakura-City, Knagawa 247-8501, Japan

E-mail: {satoshi,yu,ken,iida}@itlab.k.dendai.ac.jp,koizumi@k.dendai.ac.jp, ‡ k-seo@hit.isl.melco.co.jp

Abstract In hardware/software co-design, the trade-off between hardware and software is performed by considering function and performance as the main conditions. In embedded systems which develop immense software, the implement ability of software re-use serves as a key of development success. In this paper, development man-hours, processing time, memory, and the number of gates are taken in as the trade-off conditions in hardware/software co-design, and as the technique of software re-use, a co-design method which incorporates real-time evaluation of re-use parts is proposed. The system level language SpecC is used for realizing the proposed method. SpecC is C language augmented with the syntax for describing hardware description and has the feature of being able to describe hardware and software in a single language. The proposed method is applied to ITS (Intelligent Transport Systems) safe driving support system and evaluated.

Keywords co-design, trade-off, re-use, SpecC, ITS, safe driving support system

1. はじめに

組込みシステムの適用分野は，半導体技術やマイクロプロセッサ技術の発展とともに，通信機器やオフィス機器などの業務機器の分野，情報家電や自動車，娯楽機器などのコンシューマ機器の分野へと急速に拡大してきた．それらの多くは高機能化・複合化しており，それを制御する組込みシステムも大規模化・複雑化している[1][2]．そうした中，開発期間の短縮，コストダウン，性能向上などの市場要求を満たすためには，ハードウェア/ソフトウェア協調設計が重要となってきている．

筆者らは，制御対象モデルと設計対象モデルを結合させたシミュレーションを行い，これをもとにして設計仕様をハードウェアとソフトウェアに機能分担させ，それぞれを段階的に詳細設計していく協調設計方式をすでに提案した[3]．この方式のITS(Intelligent Transport System)の画像処理系・制御系開発への適用可能性を見出した．さらに筆者らはこの本方式をITSシステムアーキテクチャにおける運転補助，自動運転の車両制御という動的モデルを組み込み，ITS画像処理および制御系を含めた開発に適用できるようにモデルを拡張した方式を提案した[4]．

しかしながら，これらの方式においては，目標性能を満足する設計条件下でハードウェアのゲート数を最小にすることにトレードオフの条件を限定している．さらに，これらの方式においては，ハードウェア，ソフトウェアのコンポーネントを再利用する手法を協調設計方式に取り入れていない．そこで，本方式において，トレードオフの条件として，工数，性能，ソフトウェアモジュールのメモリ，ハードウェアモジュールのゲート数の複合要因を加味し，この条件下でソフトウェア再利用を実現する方式を提案する．再利用においては，まず再利用可能なモジュールをリポジトリから選択し，ネットワーク上でモジュールの動作を評価・確認しる方法をとる．次に，再利用可能な複数のモジュールから複合要因を加味したトレードオフによって最適なモジュールを選択する方法を実現する．

本方式においては，システム機能仕様をハードウェアとソフトウェアの区別なしに同じ文法によって統一的にSpecCで記述する方法を活用する．この方式をITS安全運転支援システムへ適用し，評価を行う．

本稿では、再利用を考慮したハードウェア/ソフトウェア協調設計方式を提案し，ITS 安全運転支援システムの一部に適用した．

2. ハードウェア/ソフトウェア協調設計方式

2.1. ハードウェア/ソフトウェア協調設計フロー

筆者らが提案したハードウェア/ソフトウェア協調設計方式のフロー図を図1に示す[3][4]．以下，その要旨を述べる．

(1) 目標設定と全体モデリング

作成しようとするシステムの設計目標を決定する．これは，後に行うハードウェアとソフトウェアの機能分担の際の判断要素としても用いる．これを基にシステムが導入される側の状態を表す制御対象モデルとシステムそのものである設計対象モデルをC言語ベースによって記述する．制御対象モデル，設計対象モデル両者を連動させ，その動作を確認する．制御処理部分についてはMATLABのSIMULINKにてシミュレーションし動作確認を行う．

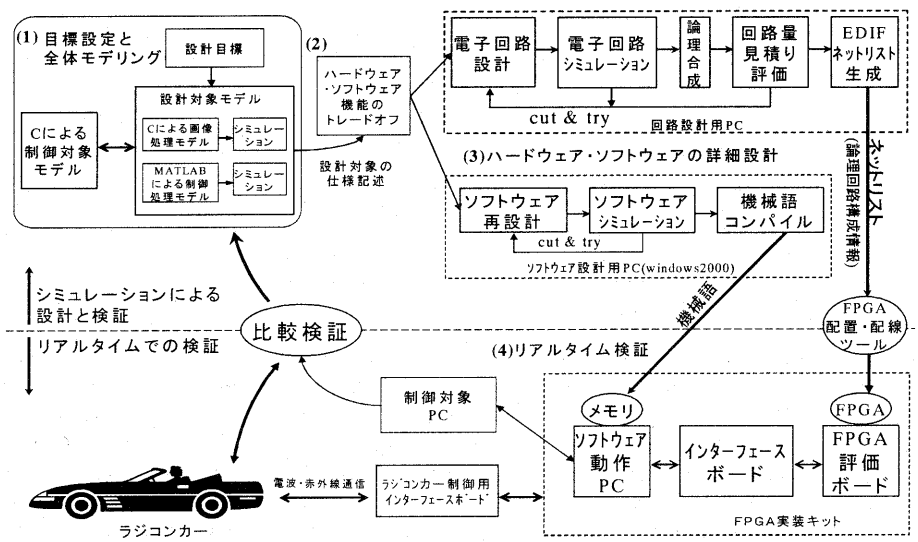


図1 ハードウェア/ソフトウェア協調設計フロー

(2) ハードウェア/ソフトウェアの機能分担

ハードウェア・ソフトウェア機能のトレードオフでは，設計対象モデルの仕様記述をもとにシステム性能（処理速度）を満足する範囲でのコストミニマムを目標とする．ここでは，LSI 化する電子回路のゲート数ミニマムをトレードオフの条件とする．このため，処理性能上ソフトウェアで実現可能な部分はソフトウェアで機能分担し，それ以外をハードウェアで機能分担させる．まず大分類として，C言語で仕様記述されている設計対象モデルをいくつかの処理コンポーネントに分割し，各処理コンポーネントに対してソフトウェア速度の見積りを行う．このとき,制御系への拡張のため,SIMULINK でモデリングした制御処理部分は,あらかじめC言語にコンパイルし,設計対象モデルの全ての処理をC言語の仕様記述にする．見積りの結果，設計目標・設計環境条件からみてソフトウェアの処理速度では遅すぎて実現困難と考えられる処理はハードウェアに機能分担し，処理速度をそんなに必要としない処理をソフトウェアに機能分担する．

次に同系列処理コンポーネントのグルーピングを行う．本プロセスでは同じ系列の処理は1つにグルーピングし，トレードオフ対象のコンポーネント数を減らして次に行う詳細分類の各処理に対する全通りのハードウェア，ソフトウェア割り当て作業回数を減らす．

詳細分類では，ハードウェア，ソフトウェア全通りの組み合わせとそのシステム全体の処理時間をシミュレーションによって算出する．シミュレーションの結果，設計目標・設計環境条件の処理時間を満たしている組み合わせを全てピックアップし，その中でも最小のハードウェア化作業ですむものを選び機能分担する．設計目標・設計環境条件を満たしていなければ，システムのアルゴリズムの改善や設計目標・設計環境条件の見直しを行う．

(3) ハードウェア/ソフトウェアの詳細設計

機能分担後，ハードウェア，ソフトウェアに分担されたものをそれぞれ電子回路設計，ソフトウェア設計を行う．ハードウェアに分担されたものは，ハードウェア記述言語である VHDL(VHSIC Hardware Description Language)にて記述し，電子回路に置き換え，シミュレーションや回路量を繰り返しながら詳細設計を行う．設計結果を論理合成し，ネットリストを生成する．生成されたネットリストを用いて，プログラム可能なハードウェアである FPGA（Field Programmable Gate Array）にてハードウェア化する．ソフトウェアに分担されたものは再設計した後，コンパイルし，機械語に変換しておく．

(4) リアルタイム検証

ハードウェアに機能分担され FPGA でハードウェア化されたものと，ソフトウェアに機能分担されコンパイルされ PC 上で動作させているものをインターフェースボード経由にて連動させ，動作結果を制御対象 PC に出力する．出力結果とモデル作成時のシミュレーション結果とを比較し，ハードウェア・ソフトウェア機能のトレードオフやアルゴリズムの改善を繰り返すことにより，目標を満たすシステムへと近づけていく．

2.2. 課題

以上に示した協調設計フローでは下記のような課題が挙げられる．

(1) 現在のハードウェア・ソフトウェア機能のトレードオフでは処理性能を満たす範囲でのコストミニマムを目標としていたが，さらにトレードオフ条件として，開発工数，メモリ，ゲート数を考慮する必要がある．

(2) 開発効率向上には部品の再利用が不可欠になってきているので，再利用を考慮した設計フローに必要がある．

(3) 現在，C言語，MATLAB を用いてモデリング,シミュレーションを行っているが，C 言語では並列処理などのハードウェア記述ができない．ソフトウェア，ハードウェアを含めたシステム全体を効率よくシミュレーションする必要がある．

3. 再利用を考慮したハードウェア/ソフトウェア協調設計方式の提案

3.1. 課題への対応

課題(1)～(3)に対し，本稿では次の対応の実現を図る．

(1) トレードオフのパラメータとして開発工数，処速度，メモリ，ゲート数を用意し，各々に制約条件式を設けることにより，より最適なソフトウェアとハードウェアの組み合わせを求める．また，各パラメータを一つの単位系に変換することで，トレードオフ作業の自動化とさらなる高速化を目指す．

(2) リアルタイムに再利用部品を活用できる再利用環境を加えた設計フローにすることにより，ネットワーク上で再利用部品の動作を評価・確認できるようにし、工数とコストを削減する．

(3) C 言語にハードウェアを記述するための構文を追加した SpecC を用いることにより，システム全体を効率よくシミュレーションできるようにする．

3.2 再利用を考慮したドウェア/ソフトウェア協調設計フロー

提案する再利用を考慮したハードウェア/ソフトウェア協調設計方式のフローを図2に示す．再利用環境下における制御対象モデルと設計対象モデルを結合さ

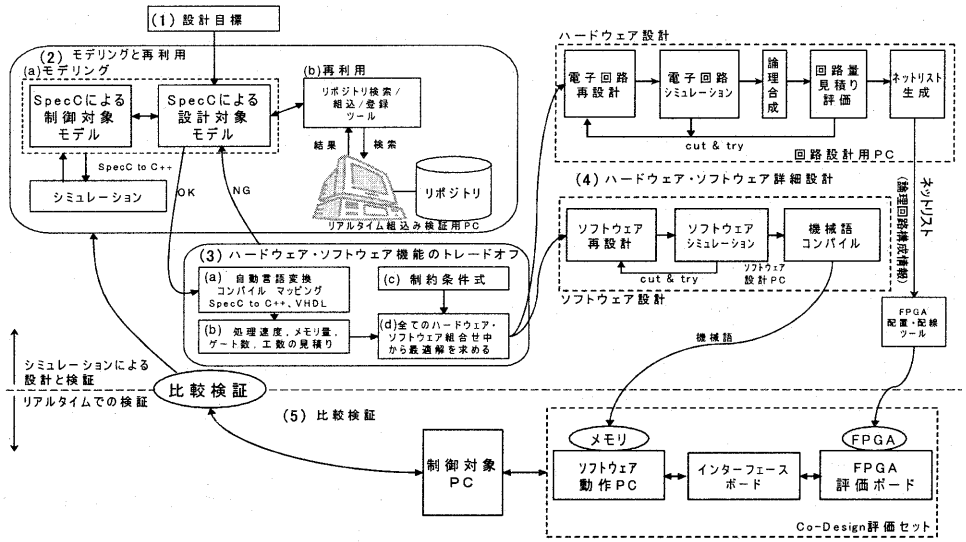


図2 再利用を考慮したハードウェア/ソフトウェア協調設計フロー

せたSpecCシミュレーションを行い，ソフトウェアとハードウェアを含めたシステム全体のシミュレーションを行う．これをもとにして見積もりを行い，制約条件による設計仕様をハードウェアとソフトウェアに機能分担させ，それぞれを段階的に詳細設計していく．以下，具体的な方法について述べる．

(1) 設計目標

ここではまず，目標システム構築における設計目標および設計環境条件を決める．設計環境条件は目標システム実現のために利用可能な設備，部品，開発ツールの制限・制約から生じる条件の記述である．設計目標では，工数，処理時間，使用メモリ，ゲート数の上限をそれぞれ，K_{MAX}，T_{MAX}，M_{MAX}，G_{MAX} とする．ハードウェア/ソフトウェア機能の作成の工数をそれぞれ k_{HW}，k_{SW}，ハードウェアとソフトウェアの処理時間をそれぞれ t_{HW}，t_{SW}，新規作成機能と再利用部品を用いた機能の使用メモリをそれぞれ m_{SW}，m_{SWIP}，新規作成機能と再利用部品を用いた機能のゲート数を g_{HW}，g_{HWIP} とし，制約条件を次のように示すことができる．

$$\text{工数} \quad : \quad K_{MAX} \geq k_{HW} + k_{SW} = k$$
$$\text{処理時間} \quad : \quad T_{MAX} \geq t_{HW} + t_{SW} = t$$
$$\text{使用メモリ} : \quad M_{MAX} \geq m_{SW} + m_{SWIP} = m$$
$$\text{ゲート数} \quad : \quad G_{MAX} \geq g_{HW} + g_{HWIP} = g$$

式1　制約条件式

(2) モデリングと再利用

(a) モデリング

続いて，設計対象モデルと制御対象モデルの両者をSpecCで作成，結合し、プログラミングを行う．ここでは，SpecCがハードウェアを記述するための構文を備えていることから，ソフトウェア，ハードウェアを含めたシステム全体の記述を行う．そして，シミュレーションによって機能動作確認をし，設計対象モデルを確定する．また，モデルをはじめとする各段階のモデルの作成の際には，その機能が設計資産にあるかどうかを，逐次再利用部品が格納してあるリポジトリ内を検索する．

(b) 再利用部品の選択

設計者は必要となるSpecCの再利用部品（IP：Intellectual Property）をWebアプリケーションサーバーから処理名・キーワード検索し，その検索結果を受け取る．そして，該当IPをクライアントのSpecCプログラムと連動させて，リアルタイム機能検証を行う．IPをクライアントPCに持ってこなくとも，Webアプリケーションサーバーに問い合わせるだけで，クライアントのSpecCプログラムにIPを組み込んだシミュレーションができる．構成図を図3に示し，詳細手順を下記に示す．

① 必要とするIPをWebアプリケーションサーバーから検索する．処理名・キーワードによる検索を行う．
② 検索結果をクライアントPCに表示をする．
③ 該当した全てのIPをクライアントPC上で動作している再利用しない部分のプログラムと組み合わせて，設計目標を満足する処理結果が得られるか否かのシミュレーションを行う．この時点では，実際にIPが機能するか，満足する結果を返すかなどの確認を行う．ここで満足する

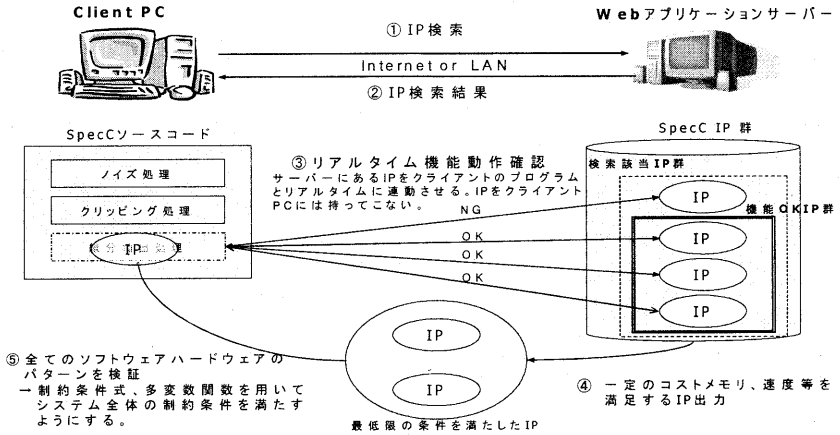
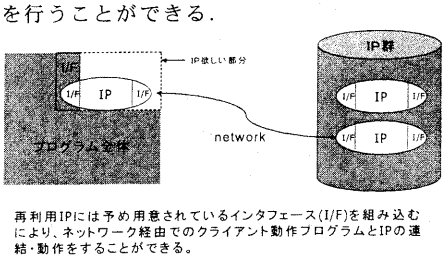


図3 IPの検索と利用

結果が得られない,組込めないなどといった場合は，再利用候補からはずす．IP はクライアントPC に持ってこなくとも，ネットワーク経由でクライアント・サーバー間を通信するI/F コードをクライアントの SpecC プログラムに組み込むことで，リアルタイムに IP を組み込んだシミュレーションを行うことができる.このネットワーク接続型シミュレーションの IP 評価法を用いることにより,ネットワークに接続された環境であれば，短時間で多くの IP の組込み，評価，動作確認を行うことができる．



再利用IPには予め用意されているインタフェース(I/F)を組み込むにより、ネットワーク経由でのクライアント動作プログラムとIPの連結・動作をすることができる。

図4 ＩＰの通信

④ 満足するシミュレーション結果を得ることができた IP 群は，次にコスト，メモリ，処理速度によるフィルタリングが行われる．クライアントによる一定の条件を満たさない IP は除去，絞込みを行う．

この作業により,膨大な IP 群から本当にシステムにあっているものを見つけることができ，ただ結果を満足するだけではなく，コスト・処理速度を満たすＩＰの候補を見つけることができる．ＩＰが見つからなかった時には，新規に作成，もしくは検索キーワードを変更し，参考となるＩＰを探し，それを元に作成をする.また，ＩＰにはソースコードだけでなく，モデル，使用方法等を記載したドキュメントが必ず付属させるなどといったルールに基づいた方法により格納されている．

(3) ハードウェア/ソフトウェア機能のトレードオフ

(a) 言語変換 SpecC to C++,VHDL

SpecC で記述された設計対象モデルを「VisualSpec2002」ツールを用いて SpecC から VHDL，C++に言語変換する.VHDL は FPGA にマッピングし，ハードウェア化をする．C++は，コンパイラで機械語に変換しＰＣ上で実際に動作させる．また同時に，また，3.2 で絞り込まれた IP を，ここでクライアント PC に移動し,絞り込まれた全ての IP 一つずつをソフトウェア，ハードウェア化をする（図５参照）.

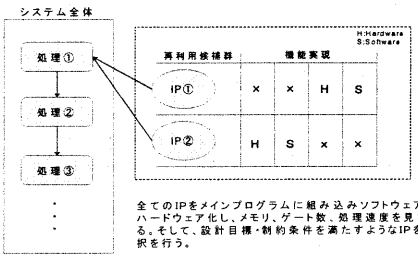


図5 IPのソフトウェア，ハードウェア化

しかし，IP ベンダーなどの IP を購入する際には，IP を購入してから初めてソースコードを得ることができるので，IP の組込み評価ができない．その場合，各 IP にはメモリ，ゲート数，処理速度など，問い合わせ可能なメソッドが各 IP に装備されているとする．

(b) 見積もり

次に下記項目を見積もる．

① システム全体の処理時間
② ビヘイビアの処理時間
③ ビヘイビアの実行回数
④ ビヘイビア同士の通信頻度
⑤ チャネルの処理時間
⑥ チャネルの実行回数

⑦ メモリ(ソフトウェアの場合)
⑧ ゲート数(ハードウェアの場合)

上記の項目①～⑥は，ハードウェアの場合とソフトウェアの場合の２通り見積もる．ビヘイビアとは，SpecCにおいて，目標システムの機能的な動作そのものである．同様に，チャネルとは，動作と動作の間でやり取りされるデータやイベント，プロトコルなど，やり取り手順を示す通信記述のことである。

(c) 制約条件

式１に示した制約条件式を満たし，かつ設計者の意図を組み込むことを可能とするため，設計者自身が工数，処理速度，メモリ，ゲート数の４要素に対して，重み付けを行う．

その際，次に示す方法により４要素の異なる単位をあわせる．まず，設計者に対して図６に示す項目からなる質問に回答してもらう．

(例)
・工数(日/人)を 1point とします．(K_{user} = 1)
・処理速度[　]ms で 1point 相当します．＝ T_{USER}
・メモリ[　]Kbyte で 1point 相当します．＝ M_{USER}
・ゲート数[　]K ゲートで 1point 相当とします．
＝ G_{USER}

図６　設計者に対する質問シート

この回答を基に，４項目に対して平等な重み付けが可能となるように単位統一関数を P_K，P_T，P_M，P_G とし，以下の式を導く．

$$P_K = (K_{MAX} - k)/K_{USER}$$
$$P_T = (T_{MAX} - t)/T_{USER}$$
$$P_M = (M_{MAX} - m)/M_{USER}$$
$$P_G = (G_{MAX} - g)/G_{USER}$$

ここで評価関数をＦとし，以下の式を導く．

$$F = U_K P_K + U_T P_T + U_M P_M + U_G P_G$$

U_K;ユーザが決定する工数の重み付け係数
U_T;ユーザが決定する処理時間の重み付け係数
U_M;ユーザが決定するSWメモリの重み付け係数
U_G;ユーザが決定するゲート数の重み付け係数

全ての組み合わせにおいて評価関数Ｆを求め，その値が一番大きい組み合わせをハードウェアとソフトウェアの機能分担結果とする．

(d) 機能分割

ハードウェアで実現する部分とソフトウェアで実現する部分を決める．まず，絞り込まれたIPから，見積もられた情報を元に，設計目標，制約条件を満たすIPを選択する．これらの見積もり結果および制約条件により，システム全体としての設計目標，制約条件を満たすハードウェアとソフトウェアに機能分担を行う．

機能分担後，システム全体のシミュレーションを行う．性能が不十分であればハードウェアとソフトウェアの切り分けを見直し，修正し，再度シミュレーションを行う．

(4) ハードウェア・ソフトウェア設計

分担されたハードウェア，ソフトウェアそれぞれに対し，電子回路設計とソフトウェア設計を行う．ハードウェアは，SpecCからのVHDL変換結果を，構成しなおし，シミュレーションや回路量見積もりを繰り返しながら詳細設計を行う．次に設計結果を論理合成しネットリストを生成する．ネットリストの結線情報を用い，FPGAによってハードウェア化する．

ソフトウェア部分はSpecCからのC++変換結果をコンパイラで機械語に変換しＰＣ上で実際に動作させ，その結果がモデル実行時の動作結果と同じかどうかを確認する．さらに処理速度が設計目標・設計環境条件を満たしているかどうかを繰り返し評価，検証しながらソフトウェアのプログラムを作成していく．

(5) 比較検証

FPGA 評価ボードとソフトウェア動作ＰＣを，インターフェースボード経由で連動させる．その動作結果を，ＰＣ画面上に入力をし，その結果をシミュレーションの結果と目視にて比較検証を行う．

4. ITS 安全運転支援システムへの適用

ITS は９つの開発分野で構成されているが[6]，本研究では安全運転の支援分野を対象とする．本研究で作成する開発コンセプトを表１に示し，運転支援システムの概要を図７に示す．

表１　開発コンセプト

支援形態	支援範囲	内容
情報提供	認知	画面にて注意喚起
警報	認知＋判断	衝突の危険性 ⇒ 3つのレベルにて警告 運転操作遅れ、不足 ⇒ 操作補助
制御	認知＋判断＋操作	警告後、運転操作遅れ等 ⇒ 回避補助

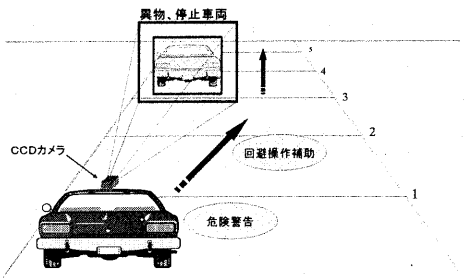


図７　ＩＴＳ安全運転システム概要

画像処理による前方の障害物や停止車両等を検出しドライバーに危険警告を行い，同時に車両に回避操

作補助を行うＩＴＳ安全運転システム（前方障害物衝突防止支援システム）に適用をする． CCD カメラにて前方画像を入力し，画像処理を行い危険検出する．処理結果をドライバーに危険警告，車両に回避操作補助を行う．

(1) 設計目標

作成する安全運転の支援システムの設計目標を表２,設計環境条件を表３に示す.表２の共通項目とは，危険警告，運転補助の２つの利用者サービスでの目標と条件の事である．制御処理の項目には，運転補助・自動運転での目標と条件を示している

表２　設計目標

共通	自動車専用の高速道路を対象とする
	専用レーン内を自律走行するシステムとする
	一般車両との混合交通とする
	システムの具体化にあたっては，車載自律型システムを基本とし，インフラの支援は必要最小限とする
	160m先の異物を検出できなくてはならない
	入力画像は256*256画素24bitカラーとする
	画像処理は単眼式ビジョンシステムとする
制御処理（運転補助，自動運転）	高速道路で時速100km走行時160m先の異物を回避できなくてはならない
	レーン移動にて回避するのか停車して回避するのか判断できなくてはならない
	異物が静止体なのか移動体なのか判断できなければならない
	移動体であればその速度と方向がわからねばならない
	異物に衝突しない停車制御目標を立てられねばならない
	障害物手前10mで停車する
	異物に衝突しないレーン移動制御目標を立てられねばならない
	停車制御目標にそった停車制御が出来なくてはならない
	レーン移動制御目標にそったレーン移動制御が出来なくてはならない
	常に道路状態を把握し，制御目標を逐次更新できなくてはならない．
	通常走行時は車速・車間・レーンを維持して走行できなくてはならない

表３　設計環境条件

シミュレーションによる検証	SpecCにてモデリングし、シミュレートする。
	前方画像はハードディスクに記録されている動画像情報をもちいる。
FPGA実装キットによる検証	前方画像は、ビデオデッキより再生したものを動画像入力インターフェースボード(A/D変換)に入力する。
	実証は、動作結果を制御対象PC上に出力し、自動車の制御状態や警告状態の確認を行う。
共通項目	ハードウェア動作には50Kゲートの FPGAを使う。
	ボイスアラームには、PC用スピーカーを使用する。
	ITS DisplayにはPC用モニターを使用する。
	自車ステータスはPC用シリアルポートを介して入力する。
	ソフトウェア設計・動作には汎用PC（CPU:1GHz,memory:512MByte）を使う。
	外部制御入出力はシリアルポートを介し9500BPSである。
	FPGAとソフトウェア動作PCとの通信はISAバス(4MBPS)を使う。
	実証は、入力画像に対する制御信号、警告信号の出力結果のログを取り合う。

この設計目標より，危険警告における条件処理時間は 0.68[sec]未満となる．また，制御処理は，車両が100[km/h]走行時，20[cm]進むごとに１回処理を行うようにする．よって，制御処理の条件処理時間は 7.2[ms]以内となる[4]．また，設計環境条件は各検証ごとに異なる利用可能な設備や部品，開発ツール，制約事項などの条件である．

上記の設計目標，設計環境条件より，制約条件式は式２のようになった． 開発期間は、５人のプロジェクトチームで４カ月間とする。

$K_{MAX} \leqq 24$[日/人]　（開発工数）

$T_{MAX} \leqq 0.68$[ms]　（処理時間）

$M_{MAX} \leqq 512$[Mbyte]　（メモリ）

$G_{MAX} \leqq 50$[KGate]　（ゲート数）

式２　制約条件式

(2) モデリングと再利用

SpecC では，通信と計算の分離し、計算の入力と出力を明確に記述する．ここでの計算の部分を「ビヘイビア」，通信の部分を「チャネル」と呼ぶ．つまり，モデルはビヘイビアとチャネルによって構成される．また，並列な動作も明確に記述し，関連する機能はグループ・階層化を行い，局所的な影響が上位の階層に及ぶことを避けなければならない[5]．

ITS 安全運転支援システムの全体モデルを図８に示し，各ビヘイビアの名称を表４に示す．

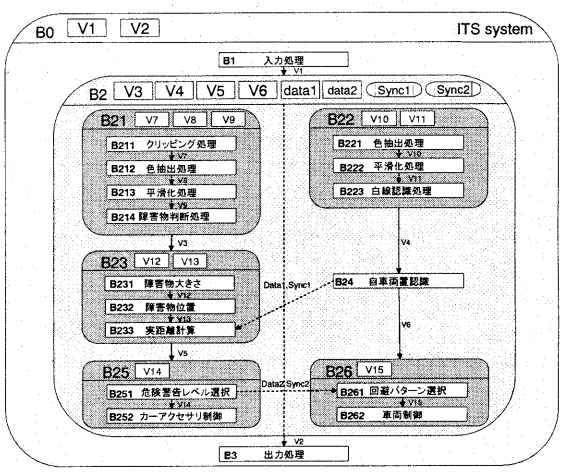


図８　全体モデル

表４　ビヘビア名称

ビヘイビア番号	処理名
B1	入力処理（A/D変換）
B2	ITS画像・制御処理
B21	障害物認識処理
B22	白線認識画像処理
B23	障害物位置特定処理
B24	自車両位置認識処理
B25	危険警告処理
B26	車両制御処理
B3	出力処理

ビヘビア B0 は三つの逐次実行するビヘイビア B1，B2，B3 からなる．B0 の実行は B1 から始まり、B1 実行が終了したら B2 の実行開始という流れの処理になる．さらに，ビヘイビア B2 は内部構造を持っており、B21 と B22,B25 と B26 は並列に動作するビヘイビアと逐次動作する B23，B24 を持つ．各ビヘイビアは変数 V1～V15 でコミュニケーションを行い，並列の動作するビヘイビアについては,チャネル Sync1,2 で同期をとり，共通変数 data1,2 を使いコミュニケーションをする[5]．これらの処理の流れを踏まえ，ITS 安全運転支援システムのモデルを説明する．

入力機器の CCD カメラから前方道路画像を取り込み，入力処理により A/D 変換された画像は，クリッピング処理により空などの画像を除き，必要な部分の道路画像を取得する．次に，色抽出処理により前方の障害物

や停止車両等の異物色検出する．そして，平滑化処理により，道端に落ちている小さなゴミなどが障害物と認識しないように輪郭線の長さが一定以下のもの除去を行う．障害物があると判断をしたら障害物の位置，大きさと自車両と障害物間の距離を算出する．また，同時に白車線を認識する画像処理を行い，自車両の位置を認識する．これらの障害物情報、車線情報の結果から危険警告処理にて，カーアクセサリを用いた危険警告レベル選択を行うとともに，障害物を回避する車両制御信号を出力する．この処理結果を，出力処理によって運転者に警告、車両制御を行う．

モデル完成後は，リポジトリに再利用部品の有無の検索を行う．検索により、再利用部品が見つかった際には，IPをリポジトリから選択し，ネットワーク上でIPの動作を評価・確認を行う．リポジトリにはコンパイル済みの実行形式になっているIPがあり，設計者は自身のローカルPCでIPを含まない部分をコンパイルして実行できるようにしておく．そして，あらかじめクライアント・サーバー間を通信するI/Fコードをクライアントの SpecCプログラムに組み込んでおき，サーバー上で動作しているIPと設計者のローカルPC上で動作している設計対象をネットワークで接続し，実行させデータをやりとりする．その処理結果を満足する再利用可能な複数のIPからユーザの条件にあうIPを選択し，モデルのシミュレーションにより機能動作確認・評価を行っていく．

(3) ハードウェア/ソフトウェア機能のトレードオフ

モデルのシミュレーションによる見積もり結果，制約条件より，最適なIPを選択し，設計目標，制約条件を満たすハードウェアとソフトウェアに機能分担・評価を行う．

(4) ハードウェア・ソフトウェア設計

分担されたハードウェア，ソフトウェア機能はそれぞれに対し，ハードウェア設計とソフトウェア設計を行う．ハードウェア設計は，設計者が設計結果のシミュレーション画面を見ながら電子回路の詳細設計をCut&Try式に作り上げていく．この時，設計環境条件にある50Kゲートの回路量を満たすようにする．ソフトウェア設計は，windows搭載のPC上で動作確認を行い，制約条件であるメモリ量を満たすように設計を行っていく．

(5) 比較検証

両者設計後は，FPGA実装キットによる検証を行う．FPGA評価ボードとソフトウェア動作PC，そして両者をつなぐハードウェア/ソフトウェア・インターフェースボードからなる．ハードウェアの設計結果であるNetListはFPGAにダウンロードし，ソフトウェアの設計結果であるC++言語ソースは機械語にコンパイルしたものをソフトウェア動作PC上のメモリにロードし実行する．シミュレーションによる検証と同じ入力動画像で動作さ，モデルのシミュレーションと同等の結果が得られるかどうかを確認・評価を行う．

今後は，これらを実現するための再利用環境構築し，実際にIPをリアルタイムに組み込み，評価・確認を行う．再利用部品を考慮したハードウェア/ソフトウェア協調設計方式により，最適なハードウェアとソフトウェアの機能分担を実現するとともに，ITS 安全運転支援システムへの適用・検証を行っていく．

5. まとめ

筆者らは，協調設計方式をITS画像処理系・制御系開発に適用し，構築・評価を行ってきている[3][4]．本稿では，システム全体をSpecCでシミュレーションを行い，トレードオフの条件として，開発工数，処理時間，メモリ，ゲート数を取り入れ，リアルタイムに再利用部品を容易に評価・組込める環境を加えたハードウェア/ソフトウェア協調設計方式を提案した．今回は，その方式をITS安全運転支援システムの設計目標とモデリングの一部に適用した．

今後の課題は次の通りである．

(1) 本方式のモデリング部分に再利用環境を構築し，設計者の作業工数をさらに減少させる．

(2) 本方式をITS安全運転支援システムに適用し，比較・評価を行う．

参考文献

[1] 高田 広章“組み込みシステム開発技術の現状と展望”情報処理学会論文誌，Vol.42,No.4, pp.930-938(2001).

[2] Thomas D.E.,Adams J.K,and Schmit H.,"A Method and Methodology for Hardware-Software Codesign",IEEE Design and Test of Computers,Vol.10,No.3,pp6-15(1993)

[3] 遠藤祐，小泉寿男，清尾克彦：ハードウェア・ソフトウェア協調設計方式とITS画像処理開発への適用検証，電気学会論文誌 D,T.IEE Japan,Vol.120-D, No.10, pp.1118-1126 (2000).

[4] 遠藤 祐，吉田 健，井上 聡，飯田庸介，小泉寿男，”ITS 画像処理系・制御系開発におけるハードウェア・ソフトウェア協調設計方式”，情報処理学会論文誌，Vol.43,No.12,pp3745-3755(2002).

[5] DANIEL D.GAJSKI 他著，” SpecC 仕様記述言語と方法論”，CQ出版株式会社(2000)

[6] 高度道路交通システム(ITS)の開発分野：http://www.iijnet.or.jp/vertis/j-frame.html (2002)