ISSN 1349-1121
JAXA-RM-04-023

宇宙航空研究開発機構研究開発資料
JAXA Research and Development Memorandum

宇宙用電子機器設計支援システムの基本設計結果

中川 敬三, 木下 常雄, 竹田 修, 辻 政信, 山本 徹也

2005年2月

宇宙航空研究開発機構
Japan Aerospace Exploration Agency

宇宙航空研究開発機構研究開発資料

JAXA Research and Development Memorandum

宇宙用電子機器設計支援システムの基本設計結果

Result of Preliminary design on Electric Design Guidance Tool for Space Use

中川　敬三、木下　常雄、竹田　修、辻　政信、山本　徹也

Keizo NAKAGAWA, Tsuneo KINOSHITA, Osamu TAKEDA,

Masanobu TSUJI, Tetsuya YAMAMOTO

総合技術研究本部 情報技術開発共同センター

Information Technology Center

Institute of Technology and Aeronautics

2005年2月

February 2005

宇宙航空研究開発機構

Japan Aerospace Exploration Agency

目　　次

宇宙用電子機器設計支援システムの基本設計結果[*1]

中川　敬三[*2]、木下　常雄[*2]、竹田　修[*2]、辻　政信[*2]、山本　徹也[*2]

Keizo NAKAGAWA, Tsuneo KINOSHITA, Osamu TAKEDA,
Masanobu TSUJI, Tetsuya YAMAMOTO

Abstract

Electric Design Guidance Tool for Space Use (ELEGANT) has the capability of top down design using hardware and software collaborating design methodology. It will realize high quality and rapid design of satellite onboard digital circuits.
In this paper, we will report the preliminary design result on the ELEGANT system.

Keywords: digital, hardware software co-design, SpecC, behavior synthesis, formal verification, refinement

概　　要

宇宙用電子機器設計支援システム(ELEGANT)は、ハードウェア・ソフトウェア協調設計によるデジタル回路のトップダウン設計を行なうツールである。ELEGANT を使用することにより、人工衛星搭載デジタル機器を高品質かつ短期間に設計することが可能となる。本報告書では、平成１６年度に実施した ELEGANT の全体システムの基本設計結果について報告する。

1. はじめに

従来、宇宙用電子機器の設計では、アルゴリズムを数値解析ソフトウェアである MATLAB や C プログラムで検証した後は、設計者の経験によって必要な機能をデジタル系およびアナログ系のハードウェアとソフトウェアに割振り、それぞれ別々に開発を進めて、ハードウェアとソフトウェアが完成した時点で動作試験が行なわれていた。この結果、①ハードウェアとソフトウェアの機能分担が最適化され難い、②ハードウェア設計が進むまで回路規模が確定しないので規模の大きなデバイスを選択しがちである、③ハードウェアとソフトウェアを組み合わせた動作試験で不具合が発生しやすく、設計作業の手戻りが大きいといった問題を抱えていた。これらの問題を解決し、宇宙用電子機器(基板数枚程度の規模)の設計期間の短縮、信頼性向上を図るため、我々のグループではハードウェアとソフトウェアを協調させたトップダウン設計を目指した宇宙用電子機器設計支援システム(ELEGANT)の試作を開始した。ELEGANT を使うことにより、設計工程の最も上流の仕様レベルから、RTL(レジスタートランスファーレベル)までの設計を合理化し、設計品質を向上させ、設計期間を短縮することができる。

この数年の間に、デジタル家電をはじめとした一般民生機器の製造会社においても、LSI の大規模化に伴い、設計を短期間に確実に開発するために、LSI 内部のハードウェアとソフトウェアを協調させたトップダウン設計の研究が開始されている。産業界全体で考えても、ハードウェアとソフトウェアを協調させたトップダウン設計は、デジタル機器設計手法に対する大きなブレークスルーと言うことができる。

また、副次的な効果として、高位レベルでは実装に依存しないため、宇宙開発産業と他の産業分野の間での設計資産(IP)の相互利用が期待でき、他の産業分野の優れた IP を取り込むことや、宇宙開発産業の IP のスピンオフが容易になる。

2. 宇宙用電子機器設計の現状に関する調査

今日、宇宙用デジタル電子機器は、人工衛星等の無線通信機器の変復調装置(MODEM)やフィルタバンク、画像処理関連装置の圧縮器や暗号化装置、メカトロニクスのロボットアームやアンテナ駆動系、姿勢制御装置などに広く使用されている。ELEGANT の基本設計を行う以前の平成 15 年度に、宇宙用デジタル電子機器の一例とし

*1　平成 17 年 2 月 10 日受付（received 10 February, 2005）

*2　総合技術研究本部　情報技術開発共同センター（Information Technology Center, Institute of Technology and Aeronautics）

て、制御系機器について、設計フローのヒアリングと分析を行い、現在の宇宙用電子機器の設計工程の問題点の抽出を行なった。図１に、制御系コンポーネントの設計フローの現状分析の結果を示す。

この結果、以下の問題点が明らかになった。

(1) 機器コンポーネントの開発仕様検討では、設計者が過去の経験に基づいて、主として机上検討により、各コンポーネントへの機能配分、ハードウェアとソフトウェアの機能配分を行なっている。このため、CPUの能力やハードウェアリソースに余裕を見込みすぎる傾向があり、設計の最適化が不十分であった。

(2) ハードウェア設計では、基板の回路構成の設計後に、外部回路構成設計および基板の回路設計と、個別のFPGA 設計が並行して行なわれている。FPGA 単体でのRTL 試験は行なわれているが、基板レベルの動作試験で不具合が発見され、設計変更を行なうケースが多く、工程上の手戻りが発生していた。また、動作テストを基板レベルで行なうため、基板試作の一回のサイクルに長期間を要するという問題もあった。

(3) ソフトウェア設計では、ソフトウェアの仕様設計とコーディングを行なっても、試作基板が完成するまでは、実際にソフトウェアを動作させて確認することが出来なかった。また、ソフトウェア試験の段階でハードウェアの不具合が発見されると、試作基板が修正されるまで待つことになり、工期が延びていた。

(4) 試作基板を使った試験では、発生頻度が極めて低い異常ケースを意図的に発生させることが難しい。また、異常ケースの試験を行なうことによって、試作基板を破壊する危険性のある試験は実施し難い。このため、異常ケースの試験が十分に出来なかった。

これらの問題は、制御系機器設計ばかりではなく、ミッション系、通信系等の電子機器設計に共通する問題であると考えられる。

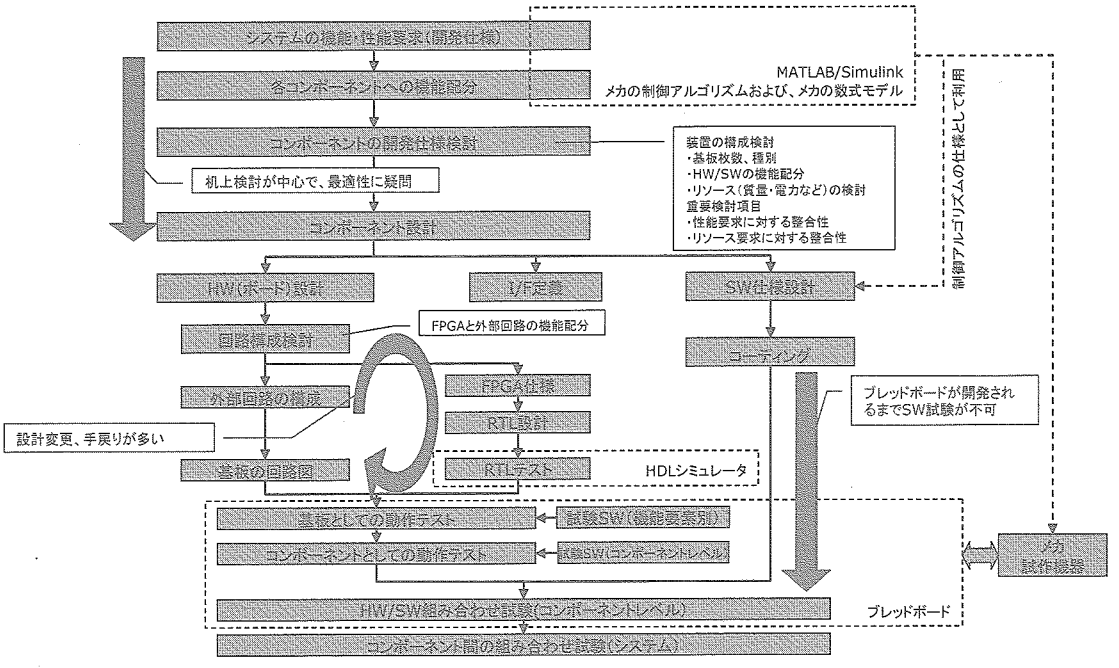

図１：設計フローの現状分析(制御機器コンポーネントの場合)

3. ELEGANT の目的と設計対象

この章では、ELEGANT の目的、設計対象について述べる。

3.1 ELEGANT の目的

2 章で述べた問題点を解決するために、ELEGANT では図 2 に示すトップダウン設計の手法に基づいた設計フローを採用し、以下の対応をとることとした。

(1) トップダウン設計により、上流工程の開発仕様設計の段階からシミュレーションによる仕様確認・分析・検証が出来るように、仕様モデルシミュレーションとアーキテクチャシミュレーションの機能を持たせる。

(2) 設計詳細化サブシステムおよび動作合成サブシステムツールにより、現在の RTL 設計よりも抽象度の高いモデルから設計を開始する。設計詳細化サブシステムおよび動作合成サブシステムにより、周辺回路設計と FPGA 設計をシームレスに行なう。

(3) 試作基板や実機が完成する前にソフトウェア検証を開始できるようにするための HW/SW 協調シミュレーション機能を備える。これにより、図 3 に示すように、宇宙用電子機器デジタル部の設計期間を半減することが可能になる。

(4) シミュレーションにより、試作ボードを使用することなく、異常ケースに関する試験を可能にする。

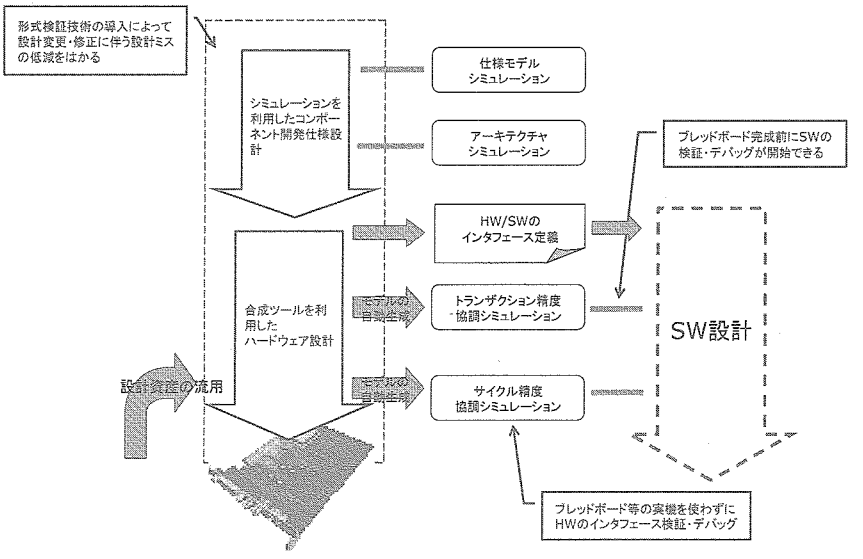

図 2　ELEGANT が目指す設計フロー

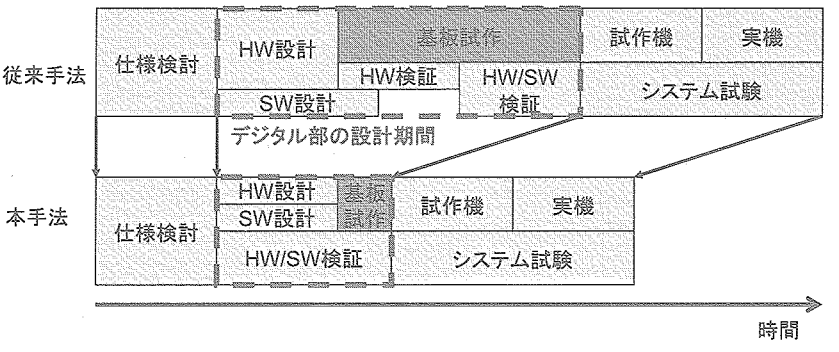

図 3　設計期間の短縮効果

3.2 ELEGANT の設計対象

ELEGANT が設計対象とする電子機器は、複数のボードからなる宇宙用電子機器の機器コンポーネントのデジタルハードウェアとソフトウェアである。典型的な規模としては、CPU1 個と FPGA 数個からなるデジタル回路を想定している。また、ELEGANT が対象とする工程は、機器コンポーネントの機能設計から、ハードウェアは RTL の設計コード作成まで、ソフトウェアは機器コンポーネントの試作基板(BBM)あるいは実機で稼動可能なコード作成までである。RTL コードから論理回路合成を行ない、配置設計には市販の論理合成ツールおよび配置設計ツールを利用することを想定している。典型的な宇宙用デジタル回路の構成を、図４に示す。

4. ELEGANT の構成

ELEGANT は、設計詳細化、シミュレーション、動作合成、形式的検証の４つのサブシステムから構成される。これらのサブシステムで取扱う設計モデル、シミュレーションの設定条件と結果、設計部品等に関する管理を構成管理ツールで行なう。構成を図５に示す。各サブシステムおよび構成管理ツールには、次の特徴がある。

- 設計詳細化サブシステム
  - ユーザは機能的な構成や演算アルゴリズムを仕様として SpecC 言語で記述する。
  - 実装に依存する通信部分（アーキテクチャ構成やバスのプロトコル）は、ユーザが GUI により段階的に指示を与えることにより、システムが詳細化する。
- シミュレーションサブシステム
  - 仕様モデルの入力支援を行なう。
  - 設計の各段階で、HW（SpecC 言語）と SW（SpecC/C 言語）を協調させたデジタル回路全体のシミュレーションを行って、動作検証やデバッグができる。
- 動作合成サブシステム
  - HW 演算部分は、動作合成により RTL（Verilog）コードまで短時間に生成できる。設計制約を与えることで速度や面積の異なる RTL を生成可能である。
- 形式的検証サブシステム
  - 設計詳細化前後（あるいはユーザによるコード修正前後）の二つのモデルが、すべての場合において等価な動作をすることを形式的に検証する。
- 構成管理ツール
  - 設計モデル、各ツールの環境設定、シミュレーション結果ログ、設計部品を管理する。
  - 手作業による修正、設計条件を変えたバリエーションとしての複数のリビジョンに対応する。設計の後戻りにも対応する。
  - ユーザ毎のファイルへのアクセス制御により、セキュリティを確保する。

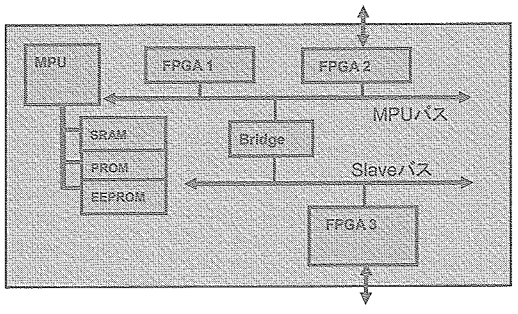

図４　典型的な宇宙用デジタル回路の構成

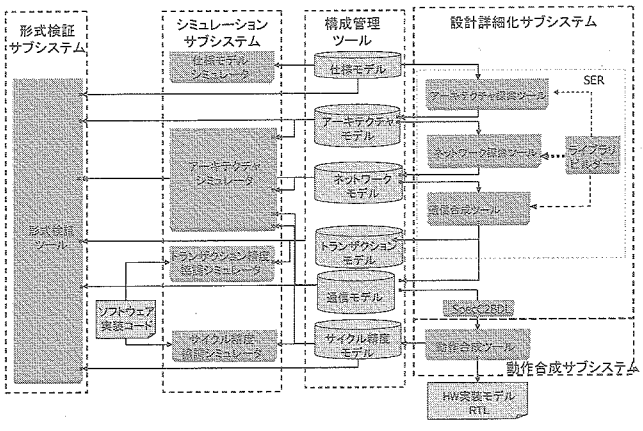

図５　ELEGANT システムの構成

5. ELEGANT の機能

5章では、ELEGANT の各機能について説明する。

5.1 仕様モデリング

仕様モデル作成機能では、設計対象が持つ実装に依存しない機能・アルゴリズムを、SpecC で記述する。機能の概略を図 6 に示す。この機能は、シミュレーションサブスステムで実現する。

(1) ビジュアルモデリング機能

SpecC 記述の設計モデルを作成・編集するための機能である。SpecC 言語のテキストコードを取り込み、ブロック図、FSM 図などの仕様モデルをビジュアルな形で入力、編集できる。また、これらの図形情報と属性情報を MS Word 形式で出力する機能も備える。

(2) C 言語から SpecC 言語への変換機能

ANSI-C で記述されたソフトウエアコードを、SpecC のビヘイビアにマッピングする。この時、ANSI-C の関数呼び出しおよび制御構造の解析を行い、SpecC のビヘイビアレベルと FSM を自動生成する。既存のソフトウェアがある場合に、それを仕様モデルに取り込むのに使用する。

(3) プログラムスライサ

ユーザがデバッグ時に、特定の変数や演算を指定することにより、その演算結果が影響を及ぼす範囲や、影響を受ける範囲を知ることが出来る。

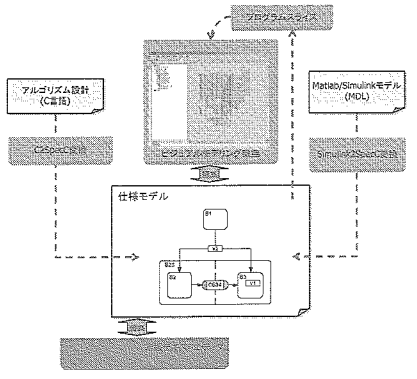

図 6 仕様モデル作成

5.2 設計詳細化と動作合成

設計詳細化サブシステムと動作合成サブシステムの機能・性能について説明する。

設計詳細化機能では、仕様モデルを、具体的なハードウェア／ソフトウェア・アーキテクチャおよび、それらを接続するバスインタフェースが定義された通信モデルに段階的に詳細化する。設計詳細化機能で、HW 部分と SW 部分の 2 つに分けられた後、HW 部分は動作合成機能、SW 部分は手動で設計を行う。動作合成機能では、SpecC で記述された通信モデルの HW 部分を RTL に変換する。設計詳細化、動作合成の各機能と各モデルの関係を図 7 に、設計詳細化機能を図 8 に、動作合成機能の概略を図 9 にそれぞれ示す。(1)～(5)は設計詳細化サブシステム、(6)と(7)は動作合成サブシステムで実現する。

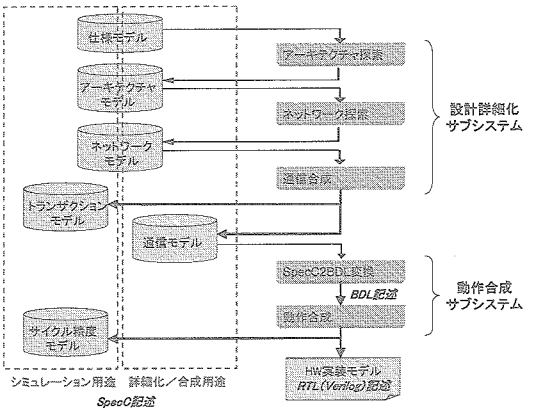

図 7 設計詳細化、動作合成サブシステムと各モデルの関係

(1) アーキテクチャ探索機能

仕様モデルを入力とし、ユーザの指示に従い、各タスクやタスク間共有変数を PE（Processing Element : MPU/DSP/メモリ/カスタム HW 等）にマッピングし、アーキテクチャ構造や SW/HW 配分を決定してアーキテクチャモデルを出力する。SpecC で記述された 10000 行未満で PE が 10 個未満かつビヘイビア、変数、チャネルが 100 個未満の仕様モデルを 5 分以内にアーキテクチャモデルに変換する。ライブラリとして、代表的な PE を用意する予定である。

(2) ネットワーク探索機能

アーキテクチャモデルを入力とし、ユーザの指示に従って、各 PE 間を接続するバス、およびバス間を接続する CE （Communication Element ：ブリッジ等）を決定し、ネットワークモデルとして出力する。この段階では、バスのプロトコルはまだ実装されない。SpecC で記述された 10000 行未満で PE 間のチャネルが 50 個未満かつ PE、CE、バスが各 20 個未満のアーキテクチャモデルを 5 分以内にネットワークモデルに変換する。宇宙用の代表的な CE およびバスを用意する予定である。

(3) 通信合成機能

ネットワークモデルを入力とし、個々のバスに接続される PE のバスインタフェース部分をライブラリに登録されたバスプロトコル部品を使って詳細化し、通信モデルおよびトランザクションレベルを出力する。SpecC で記述された 10000 行未満で PE 間のチャネルが 20 個未満かつ PE、CE、バスが各 20 個未満のネットワークモデルを 5 分以内に通信モデルに変換する。

(4) ライブラリ作成機能

PE および CE のデータベース作成支援とバスデータベースの作成支援を行なう。ユーザが設計詳細化サブシステムのモデリングスタイルやテンプレートに従ってプロファイルやモデルを作成するのを支援する。

(5) SpecC から BDL への変換機能

通信モデルの HW 設計部分を入力とし、SpecC 記述から既存動作合成ツールが動作合成可能な C 言語（BDL）記述に自動変換する。

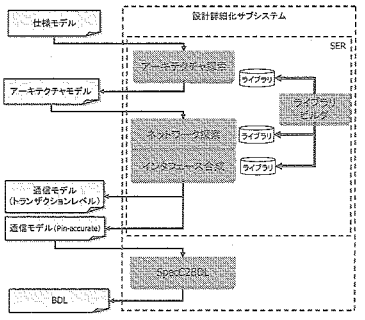

図 8　設計詳細化機能

(6) 動作合成機能

通信モデル HW 設計部分を BDL 記述に変換したものを入力とし、既存動作合成ツールが、論理合成が可能な RTL（Verilog）記述を生成する。動作合成にあたっては、ユーザが指定するクロックサイクル数や演算器の数などの制約条件により、同一の BDL 記述から、速度、面積の異なる RTL 設計モデルを生成でき、これらを比較評価することにより、HW 設計の最適化を図ることができる。動作合成機能には、タイミングが設計上重要な制御回路に対応した固定スケジューリングモードと、タイミングを意識しない信号処理回路等の設計を想定した自動スケジューリングモードの２つが用意される。

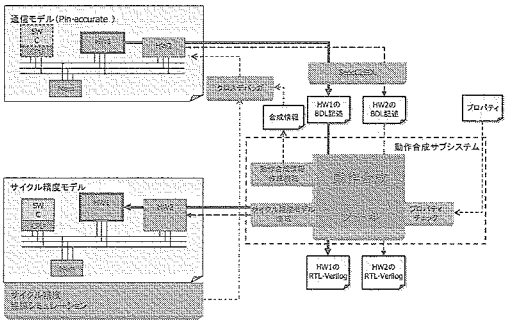

図 9　動作合成機能

(7) SpecC サイクル精度モデル生成機能

SW との協調シミュレーション用（サイクル精度 ISS を利用）に、RTL 記述とクロックサイクル精度で演算結果が等価な SpecC 記述サイクル精度モデルを出力する。500 行未満の BDL 記述を入力とした場合、SpecC サイクル精度モデルを 10 秒程度で出力するすることを目標としている。(Pentium4 2.4GHz マシン使用)

5.3 シミュレーションとデバッグ

ELEGANTには、シミュレーションとデバッグのための数多くのツールが用意されている。シミュレーション対象の各モデルに対する推奨ツールを表1に示す。ここでは、推奨ツールのそれぞれについて、機能・性能を説明する。

(1) SpecC2.0 シミュレータ

ELEGANTでは、SpecCのテキストコードをSpecCコンパイラ(SCC)でコンパイル、実行することによりシミュレーションを行なう。実行形式のEXEファイルを作る際に、SCCからVC++.netのコンパイラが呼び出され、SpecC言語の並列実行構文、例外処理実行構文、およびイベント変数などを実行するためのSpecC2.0シミュレータ実行ライブラリがリンクされる。(図10参照)

SpecC2.0 シミュレータは、SpecC で記述された全ての抽象度の SpecC コードを、C++コードに変換して実行する。SpecC から C++コードに変換するときに、並列プロセスの静的スケジューリング、スレッド管理の高速化等を行なって、シミュレーション速度の向上を図っている。性能としては、既存の OSCI 版 SystemC 標準シミュレータよりも、untimed な抽象度のレベルのモデルで 1.2 倍以上、サイクル精度の抽象度のレベルで5倍以上の高速化を図る。

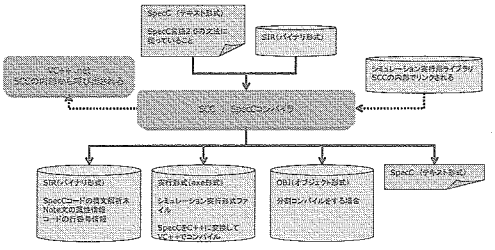

図 10 SpecC2.0 シミュレータの入出力

(2) システムモデルデバッグ機能(SpecC デバッガ)

各モデルの SpecC コードをソースコードレベルでデバッグする機能である。この機能への入力は、SpecC 言語のテキストファイル、SpecC コンパイラを使ってコンパイルした実行形式ファイル、SpecC コンパイラから出力されるスタティックプログラム DB ファイルである。この機能の画面例を図 11 に示す。ビヘイビア・チャネルの階層ツールとソースコード表示の2種類を使ってモデル記述を見ることができる。モデル記述の任意の場所にブレイクポイントを設定できる。ユーザは、プログラムの実行・ステップ実行などの実行管理、ソース表示上での実行停止位置の確認、変数・ポートの現在値のモニターを行なうことができる。

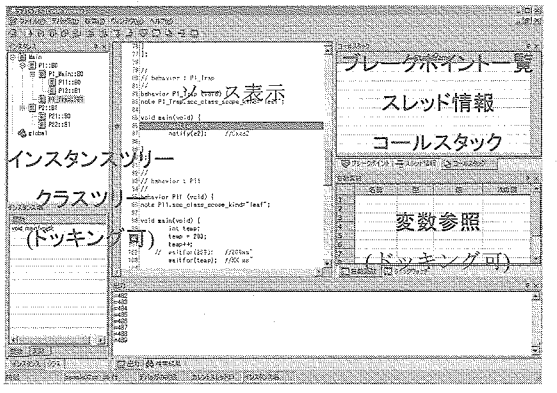

図 11 システムデバッグ機能 GUI

(3) MATLAB/Simulink 連携機能

SpecC2.0シミュレータとMATLAB/Simulinkのシミュレーションを協調実行するためのブリッジによる方法と、SpecC2.0 シミュレータと MATLAB /Simulink の RTW (Real Time Workshop) が出力した C コードを連動させて協調実行する方法の2つの方法を用意する。

表1 シミュレーション対象モデルと推奨ツール

シミュレーション対象のモデル ＼ 推奨ツール		SpecC2.0 シミュレータ	システムモデルデバッグ機能(SpecCデバッガ)	MATLAB/Simulink連携機能	モニターツール				アーキテクチャプロファイラ			動作合成							
					BSW	Strip Chart	Waveform Viewer	Memory Viewer	メモリアクセス解析	命令セット解析	バス解析	プロパティチェック	トランザクションレベルISSとの連携機能(Simmips ブリッジ)	トランザクションモデルビルド機能	時間精度付きSWモデル生成機能	ソフトウェアデバッガ連携機能	サイクル精度ISSとの連携機能(ASVP ブリッジ)	クロスデバッガ機能	Verilog-HDL/SpecC言語の協調シミュレーション
仕様モデル	機能、アルゴリズムを記述	○	○	○	○	○													
アーキテクチャモデル	仕様モデルの計算部分をPE(SW、MPU、カスタムHW、メモリ)に実装	○	○	○	○	○	○	○	○	○									
ネットワークモデル	仕様モデルの通信部分をCE(バスブリッジ、メモリコントローラ)に実装	○	○	○	○	○	○	○	○	○	○		ISS有りの方式		ISSなしの方式				
トランザクションモデル	命令単位の精度をもつ	○	○	○	○	○	○	○	○	○	○		○	○	○	○			
通信モデル	ピン、ワイヤー単位でバスを実装	○	○	○	○	○	○	○	○	○	○								
サイクル精度モデル	サイクル単位の精度をもつ	○	○	○	○	○	○	○	○	○	○	○					○	○	○

(4) モニターツール

(a) BSW (Behavior State Watcher)

ビヘイビアの状態遷移のログおよび並列に実行されるビヘイビア間での通信や同期をモニターするツールを用意する。(図 12 参照)

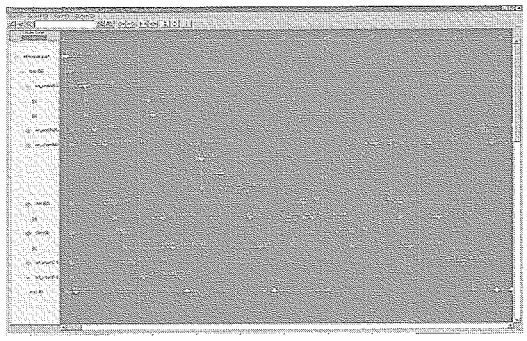

図 12 BSW 画面

(b) Strip Chart ビュア

SpecC 記述の中でユーザが指定した変数および信号値の時間的な変化をアナロググラフで表示するツールを用意する。(図 13 参照)

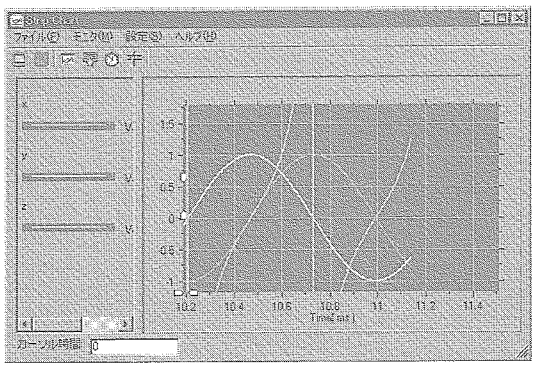

図 13 Strip Chart ビュア画面

(c) Waveform ビュア

SpecC 記述の中でユーザが指定した変数および信号値の時間的な変化を、デジタル波形で表示するツールを用意する。(図 14 参照)

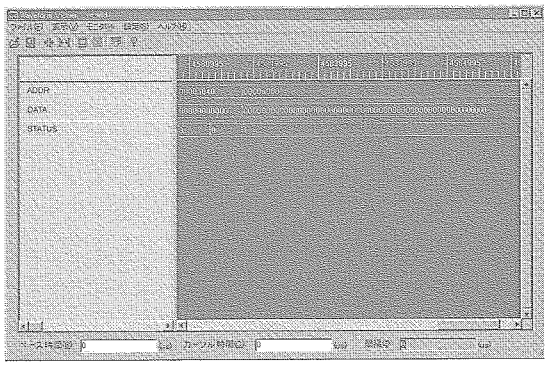

図 14 Waveform ビュア画面

(d) メモリビュア

SpecC 記述の中でユーザが指定した変数および信号値に対応するメモリの現在値をモニターするツールを用意する。(図 15 参照)

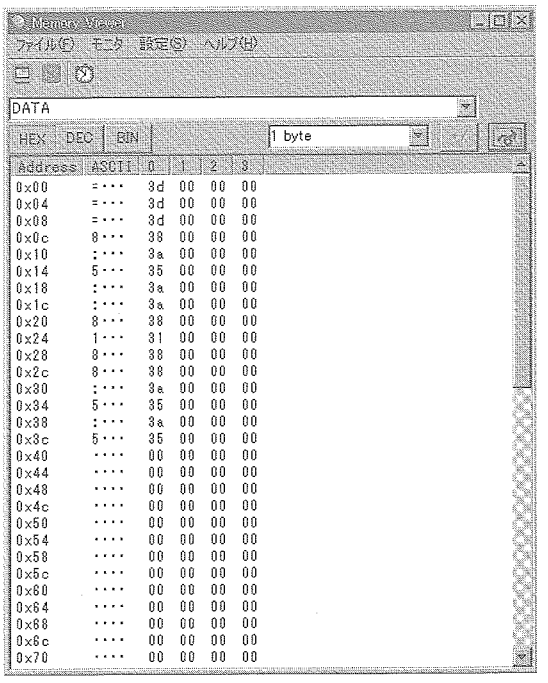

図 15 メモリービュア画面

(5) アーキテクチャプロファイラ

(a) メモリアクセス解析

メモリへのアクセスに関するログや統計を取る機能を用意する。

(b) 命令セット解析

時間精度付きソフトウェアモデルに関して、CPU の演算命令毎の呼出し回数や実行レイテンシの集計を行なう機能を用意する。この機能を使ってビヘイビア毎の性能を計測できる。

(c) バス解析

バスのアクセスとバスに流れるデータの解析を行なう機能を用意する。この機能を使って、バストランザクションの発生頻度、レイテンシ、バス利用率の時間的な変化を計測できる。

(6) 動作合成のプロパティチェック

BDL 記述に対して、配列変数のインデックスの境界違反や条件分岐の可到達性といったプロパティのチェックを行なう機能を用意する。プロパティが満足されない場合には、反例のデータ出力と波形表示を行う。

(7) トランザクションモデルビルド機能

設計詳細化サブシステムが生成したトランザクションモデルと、ISS(Simmips)のモデルや時間精度付きソフトウェア生成機能が出力するソフトウェア部分のモデルの連携部分のコードを自動接続する機能を用意する。

(8) トランザクションレベル ISS との
連携機能(simmips ブリッジ)
ターゲット CPU(TX49)の命令をホストマシン上で擬似実行するトランザクションレベルの ISS(命令セットシミュレータ)と SpecC2.0 シミュレータを連携実行する機能を用意する。トランザクションレベル ISS として、Green Hills Software 社の MULTI に含まれる simmips を使用する。Simmips と SpecC シミュレータの連携シミュレーションを 50k 命令/秒以上(目標)の速度で実行できる。

(9) 時間精度付きソフトウェアモデル生成機能
ANSI-C または SpecC で記述されたソフトウェアモデルを入力とし、クロスコンパイラを用いてレイテンシ解析を行い、ターゲット CPU (TX-49)でソフトウェアを実行した場合の実行時間を推定する。実行時間の推定結果を入力となったソフトウェアモデルに挿入し、SpecC に変換して出力する機能を用意する。(図 16 参照)

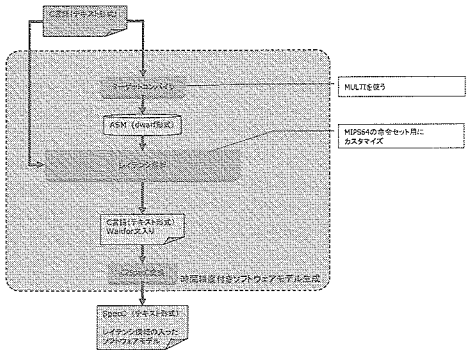

図 16 時間精度付きソフトウェアモデル
生成機能

(10) ソフトウェアデバッガ連携機能
C ソースコードから SpecC 言語に変換して作成したモデルをシミュレーションする際に、元の C コードをソース表示しながらシミュレーションのデバッグを行なう機能を用意する。

(11) サイクル精度 ISS との
連携機能(ASVP ブリッジ)
ターゲット CPU(TX49)の動作をサイクル精度で擬似実行する CPU シミュレータと SpecC2.0 シミュレータを連携実行する機能を用意する。サイクル精度 ISS として、アドバンスドデータコントロール社の ASVP を使用する。ASVP と SpecC シミュレータの連携シミュレーションを 5k 命令/秒以上(目標)の速度で実行できる。

(12) クロスデバッガ機能
動作合成サブシステムが生成するサイクル精度モデルを使ってシミュレーションを行なう際に、合成前の通信モデルのコードを参照しながらシンボリックデバッグを行なう機能を用意する。この機能により、動作合成前のコードのデバッグや性能のチューニングが容易になる。

(13) Verilog-HDL と SpecC 言語の協調シミュレーション
機能
Verilog-HDL で記述されたコードを、これと等価な SpecC 言語のコードに自動変換する機能を用意する。この機能により、既存の RTL 資産の利用や動作合成サブシステムが生成した実装モデル(RTL)と、SpecC 言語の協調シミュレーションが可能になる。

5.4 形式的検証

形式的検証は、ユーザによるモデルの修正や、ELEGANT システムによる詳細化の前後で、モデルが等価であることを確認するために用いる。機能の概要を図17に示す。

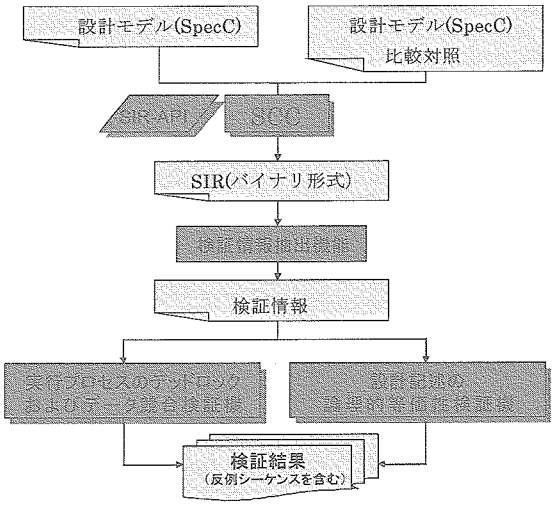

図17　形式的検証

形式的検証サブシステムは、SpecC で記述された仕様モデル、アーキテクチャモデル、ネットワークモデル、トランザクションモデル、通信モデルの各モデルのデッドロックおよびデータ競合の検証と、隣接するレベル間の等価性および同一レベルでの修正前後でのモデルの等価性の検証を行なう。

最初に、全てのタスクがイベント待ち状態に入って処理が進まなくなるデッドロックの検証と、２つのタスクのどちらが先に共有変数にアクセスしたかによって実行結果が異なるデータ競合の検証を行なう。設計モデル中にデッドロックやデータ競合を発見した場合には、それを再現するシーケンスを内部信号の波形表示が可能な形式で出力する。

次に、比較したい２つの設計モデルの等価性検証を行なう。入力は、シミュレーションサブシステムの SpecC コンパイラによりコンパイルされた SIR ファイルである。等価性検証は処理に時間がかかる(1GHz クロックの PC で、SpecC 記述 500 行の部分回路の検証が１０分程度、500 行の部分回路 50 個の設計モデルで１０時間程度)ため、問題をビヘイビア単位に分割した後、２つのビヘイビアの間で等価性を検証したほうがよい。等価でない場合には、2 つのビヘイビア間で出力変数を異ならせるシーケンスを内部信号の波形表示が可能な形式で出力する。

5.5 構成管理

構成管理機能は、ELEGANT で取扱う設計モデル、シミュレーション用のデータ、評価結果、設計部品などの情報のリビジョン管理、工程作業履歴、アクセス制御を行なうためのものである。図 18 に構成管理データベース上のオブジェクトの階層を示す。

ELEGANT では、設計詳細化/動作合成の各レベルで出力されるモデルの種類が多く、ユーザが設計制約条件をパラメータにして、同じレベルのモデルを何種類も作成するため、設計モデルやその制約条件、シミュレーション結果の管理が重要になる。

ユーザが作業を実施するごとに、作業工程、作業を行なった日時、利用したファイルやツール、作業によって作成・更新されたファイル等に関する工程作業履歴を記録・管理する。

また、多人数のユーザが ELEGANT を使用する場合に、セキュリティを保持しながら設計データにアクセスして設計作業を進められるように、ユーザ毎にデータベースの閲覧、書込み、追加、削除のそれぞれに対してアクセス制御を行なう。

ELEGANT の構成管理データベースには、市販ソフトを利用し、サーバ PC に置かれた設計情報に、各クライアント PC から構成管理ツールのクライアントソフトウェアを介して、アクセスできるようにする。

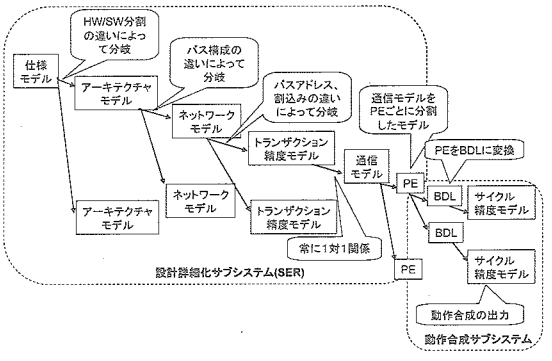

図18　構成管理データベース上のオブジェクトの階層

6. ELEGANTによる設計方法

6章では、5章で説明したELEGANTの各機能を使った設計方法について説明する。ELEGANTを用いた設計には、図19に示すように、仕様モデリング、設計詳細化および動作合成、シミュレーションおよびデバッグ、HW/SW協調検証、形式的検証の各設計工程がある。

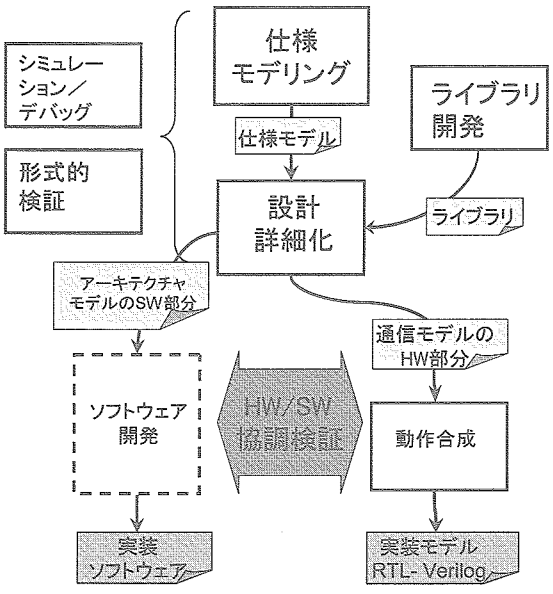

図19 ELEGANTによる設計の流れ

6.1 仕様モデリング

仕様モデリング工程を図20に示す。まず最初に、ユーザは設計対象の機能・アルゴリズムを表現した「仕様モデル」を作成する。仕様モデルは、ELEGANTプロジェクトで準備するスタイルガイドに沿って、SpecCまたはGUIを使って下記の内容を記述する。

- 機能の構成
- 個々のタスクを駆動するためのアルゴリズム
- 並列タスク間の同期／非同期通信
- 外部システム(テストベンチ)の動作モデル

また、必要に応じて、ユーザがC言語ソースコードやMATLAB/Simulinkモデルを変換して、仕様モデルに取り込むことが出来るようにする。

次に、仕様モデルをコンパイルして、「仕様シミュレーション」を実行し、モデルの挙動の実測値を計測する。必要に応じて、波形ビュアーやストリップチャートを使って、仕様モデルの各ブロックの状態遷移や各変数の変化を観測することができる。次にユーザは、得られた実測値と、あらかじめ用意しておいた期待値を比較することにより、仕様モデルシミュレーション結果の評価を行なう。評価の観点は、以下の通りである。

- 必要な機能が備わっているか
- 外部システムとやりとりするデータの妥当性
- アルゴリズムが正しく設計されているか
- 外部からの入力データに対して期待される出力が出ているか

この段階で仕様モデルの記述間違い(バグ)が見つかった場合にはデバッグを行なう。

仕様モデルの修正の前後で、ユーザが指定した範囲の仕様モデルが等価であることを形式的検証サブシステムにより確認する。また、並列に駆動するプロセス間の同期通信に関する設計誤りにより実行がブロックされないことをデッドロック解析で確認する。

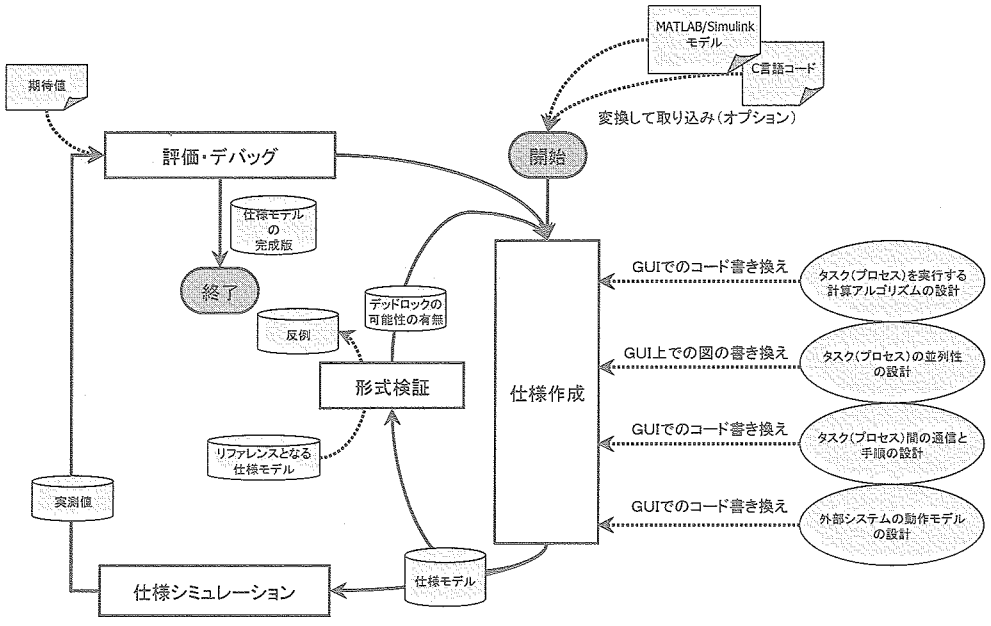

図20 仕様モデリング工程

6.2 設計詳細化

設計詳細化工程を図21に示す。この工程では、アーキテクチャ探索、ネットワーク探索、通信合成、シミュレーションを行なう。

アーキテクチャ探索では、仕様モデルを入力として、各タスクやタスク間共有変数を PE にマッピングし、アーキテクチャ構造や SW/HW 配分を決定する。ユーザは、GUI を使って PE データベースから PE を選択し、各タスクに PE をマッピングする。

ネットワーク探索では、「アーキテクチャモデル」を入力とし、各 PE 間を接続するバス、およびバス間を接続する CE を決定する。ただし、この段階ではバスのプロトコルはまだ実装されない。ユーザの操作は、GUI を使って CE を選択し、PE 及び CE 間にバスを配置することである。

通信合成では、「ネットワークモデル」を入力とし、各バスのプロトコルやバス接続、バスのアドレスを決定する。バス通信に関してピン精度の「通信モデル」として出力する。(PE にマッピングされた各タスクの演算部分は仕様モデルから変わっていない。) また、SW との協調シミュレーション用(トランザクション精度 ISS を利用)に、バスプロトコルの詳細を簡略化した「トランザクションモデル」を別途出力する。

アーキテクチャ探索、ネットワーク探索、通信合成の入力となったモデルと、生成されたモデルの間の等価性は、形式検証により確認する。

アーキテクチャモデル、ネットワークモデル、通信モデルを入力として、シミュレーションを行うことにより、ターゲット CPU でソフトウェア部分を実行した場合の演算量とレイテンシを解析し、アルゴリズムあるいは機能の中で処理が重い部分を探すことができる。入力として、どのモデルを使用するかは、ユーザが解析したい性能の内容と精度に応じて選択する。

シミュレーションにより、ハードウェア／ソフトウェア間の通信量を分析し、ハードウェア／ソフトウェア間の通信量の分布や時間的推移を知ることができる。これらの結果から、ハードウェア化が必要な部分やハードウェア/ソフトウェア分割の検討を行なうことが出来る。

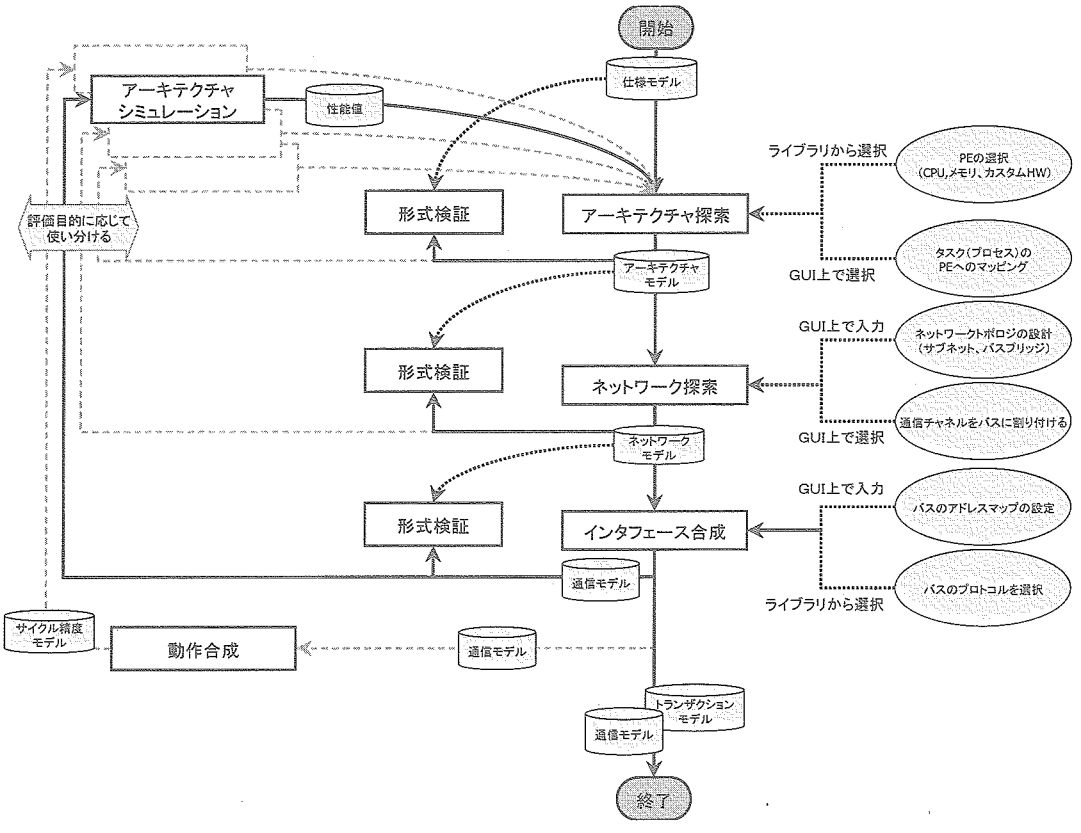

図21 設計詳細化工程

6.3 動作合成

動作合成工程を図 22 に示す。この工程では、SpecC で記述された HW 部分の「通信モデル」を動作合成可能な BDL 記述に書き換え、BDL からユーザが GUI により設定した制約条件のもとで動作合成を行ない、HW 実装モデル(RTL-Verilog)とサイクル精度モデルを出力する。ユーザは、設定する制約条件を変更することにより、同一のモデルから速度や面積の異なる RTL 設計モデルを生成することができる。

通信モデルの SpecC から BDL への書き換えは基本的には ELEGANT システムが自動で行なうが、ユーザは必要に応じて、動作合成できない記述を修正したり、HW の性能を高めるための変更を行なう。この記述修正によってデッドロックが発生するような潜在的な問題を生じさせていないことを形式的検証サブシステムによって確認することができる。

ユーザは制約条件として、クロックサイクル数や演算器の数などを設定し、動作合成を実行する。動作合成では入力が同一のモデルであっても、ユーザが与える制約条件により、速度や面積の異なる RTL モデルが生成される。ここでは、プロパティチェック機能により、配列インデックスの境界違反や条件分岐の可到達性といった設計対象によらない一般的なチェックや、ユーザによって記述された設計対象固有のチェックを行うことができる。

動作合成サブシステムが出力するサイクル精度モデルと実装ソフトウェアあるいはハードウェア試験用コードを使って、サイクル精度協調シミュレーションを行う。サイクル精度協調シミュレーションにより、ハードウェアロジックの検証、通信路のトラフィックの同期タイミングの検証、サイクル精度でのシステムのレイテンシー解析を行なう。

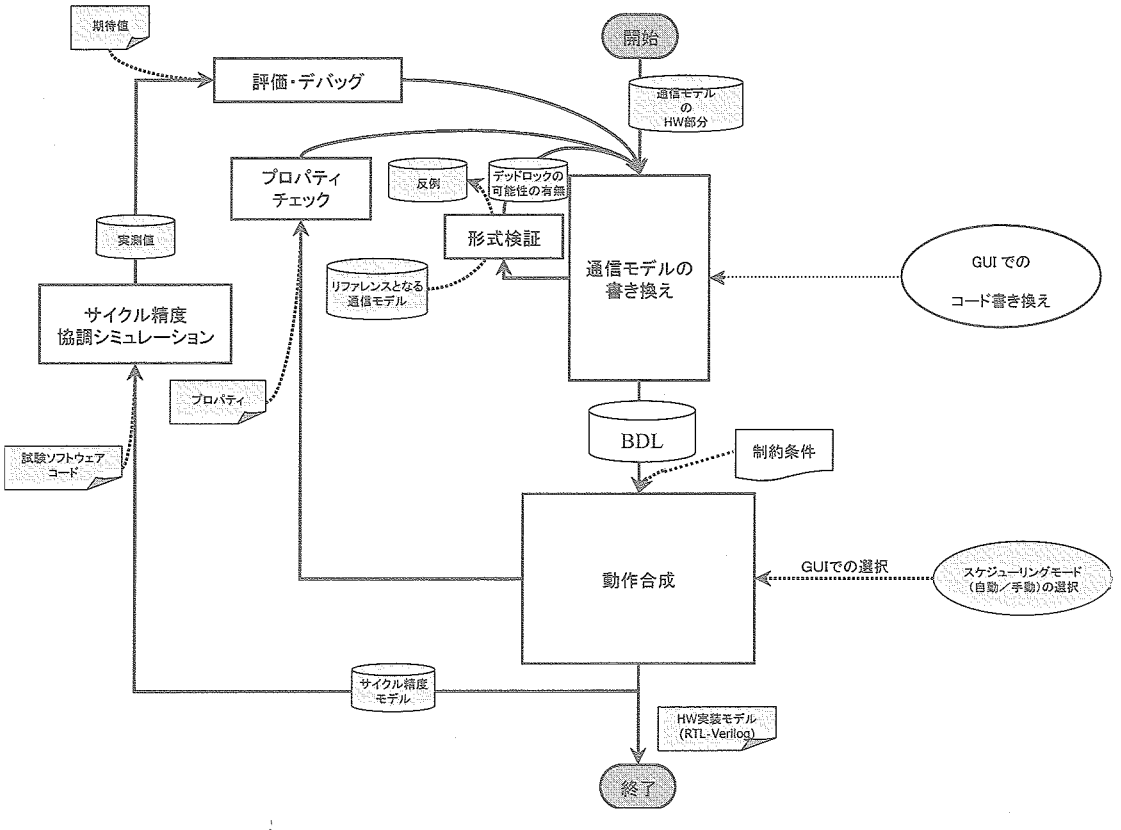

図 22　動作合成工程

6.4 ソフトウェア開発

ソフトウェア開発工程を図23に示す。

ELEGANTは、システムとしてソフトウェア開発に対する直接の設計ツールの機能は持たないが、以下に示す方法で、ELEGANTによるハードウェア設計と並行して、ソフトウェア設計を進めることができる。

ソフトウェア開発は、詳細設計サブシステムによって出力される通信モデルのうち、ソフトウェアにマッピングした部分が、その仕様書となる。この仕様書をもとに、ソフトウェアの設計を行なう。

ソフトウェアのアルゴリズム部分の設計に関しては、通信モデルの中でソフトウェアによる実行として割り付けたタスクのアルゴリズムが、そのままソフトウェアのアルゴリズム設計となる。この部分は基本的に、仕様モデルで記述したアルゴリズムがそのまま踏襲される。

ソフトウェア開発では、以下の作業を主に(1)→(2) →(4)の順で繰り返す。また、必要に応じて(1) →(3) →(4)のサイクルを実行し、評価結果が満足できるものとなったところで完了する。

(1) ソフトウェアコード修正

アルゴリズム部分とインタフェース部分の設計が終わったソフトウエアに、RTOSを利用する場合のタスク制御、スタートアップルーチンを追加する。また、必要に応じて、HW/SWインタフェース仕様を逸脱しない範囲で、ソフトウェアアルゴリズムの修正を行なう。

(2) トランザクション精度協調シミュレーション

実装ソフトウェアとHWであるトランザクションモデルの協調シミュレーションを行って、実装ソフトウェアを試験する。トランザクション協調シミュレーションには、命令セットシミュレータ(ISS)を利用する方式と、ソフトウェア実行部分のモデルを時間精度付きモデルに変換してISS無しで行なう方式の２つがあり、ユーザが選択する。

(3) サイクル精度シミュレーション

SWがHWと通信するタイミングを検証するため、ターゲットCPUのサイクル精度シミュレータとハードウェアのサイクル精度モデルを連動させて、設計対象のシステム全体の挙動をクロックサイクルの精度で試験する。このシミュレーションは、実行速度が非常に遅いため、試験用の小さなソフトウェアで検証を行なうことが望ましい。

(4) 評価・デバッグ

ソフトウェアデバッガを使って、各変数、レジスタ、メモリ等の値を監視し、ソフトウェアの機能確認を行なう。

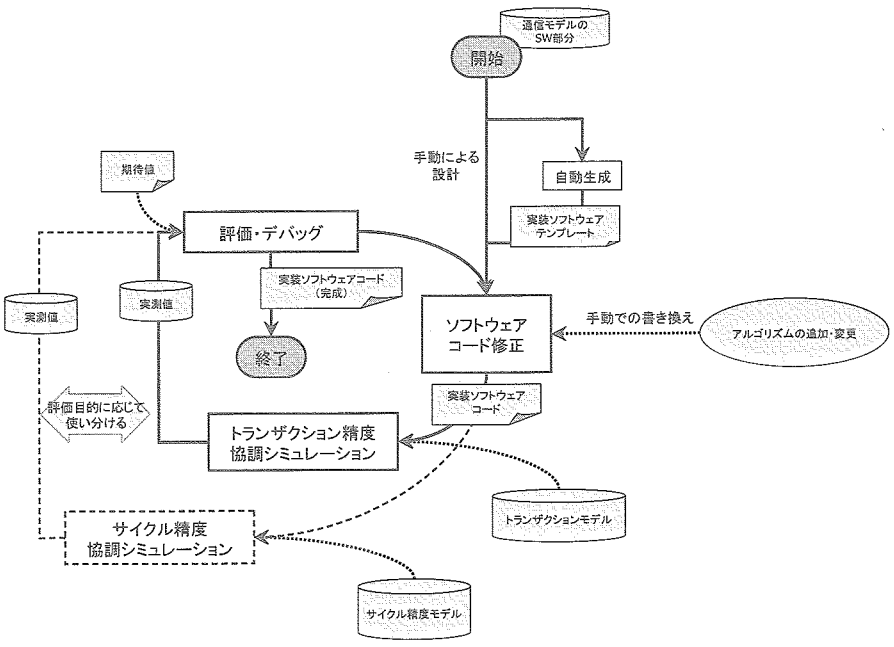

図23 ソフトウェア開発工程

7. ELEGANTの動作環境

7.1 ハードウェア環境

ELEGANTシステムを利用するのに必要なハードウェア環境を以下に示す。

(1) ホストマシンの稼動環境

- ホストマシン：インテル互換x86 PC (Pentium 1GHz以上)
- 1GB以上のRAM
- 1GB以上のハードディスク空き容量
- OS：MicroSoft Windows XP

(2) SER(詳細設計化サブシステム)の稼動環境

- ホストマシン：インテル互換x86 PC (Pentium 500MHz以上)
- 128MB以上のRAM
- 200MB以上のハードディスク空き容量
- OS：RedHat Enterprise Linux WS,Version 3

同一のホストマシンでSERと他のELEGANTシステムのソフトを利用する場合は、Windows XPをホストとして、VMwareの中にゲストOSとしてLinuxをインストールして運用する。その際には、Pentium 2GHz程度の計算機環境が望ましい。

7.2 ソフトウェア環境

ELEGANTシステムの設計環境のホストマシンに必要な第3者ソフトウェアを以下に示す。

(1) Microsoft VisualStudio.NET2003 (MicroSoft社)

シミュレーションサブシステム、形式的検証サブシステムを実行するために必要である。
SpecCコードのコンパイル、SpecCコードからSIRへの変換に使用する。

(2) VMware Workstation (VMware社)

設計詳細化サブシステムのSERと他のサブシステムを、同一のホストマシンで使用する場合に必要。
Windows XPをホストとしてVMWareの中にゲストOSとしてLinux (RedHat Enterprise)をインストールしてSERを運用する。

(3) MULTI及びSimIO (Green Hills Software社)

シミュレーションサブシステムにおいて、トランザクション精度協調シミュレーション、サイクル精度協調シミュレーション、時間精度付ソフトウェアモデル生成機能を実行する場合に使用する。

(4) TX49ASVP for VisualSpec (アドバンスドデータコントロールズ社)

シミュレーションサブシステムにおいて、TX49をCPUとするアーキテクチャでサイクル精度シミュレーションを行う場合に使用する。

(5) MATLAB/Simulink (Mathworks社)

シミュレーションサブシステムにおいて、MATLAB/SimulinkとSpecC言語の協調シミュレーションを実行する場合に使用する。Real Time Workshopが出力したCコードとSpecC言語の協調シミュレーションを行う場合には、Real Time Workshopも必要になる。

(6) Enabler (FUJITSU Enabling Software Technology社)

構成管理機能を利用する場合に使用する。

(7) CodeSurfer (GrammaTech社)

シミュレーションサブシステムにおいて、プログラムスライサを利用する場合に使用する。

(8) gcc, glibc, libxml2 (Free Software Foundation)

シミュレーションサブシステムにおいて、プログラムスライサを利用する場合に使用する。

8. 今後のスケジュールと展望

ELEGANT の試作・研究スケジュールを表 2 に示す。平成 16 年度に、全体システムの基本設計、詳細設計の大半を実施している。平成 17 年度に、詳細設計の一部、各サブシステムの試験、システムインテグレーション、総合試験を実施する予定である。実際に ELEGANT を使って設計を行なう場合、設計部品(IP)が揃っていると使い勝手が良く、参照できる設計例があると設計方法を理解する助けになるので、設計部品や設計例のデータベースを平成 16 年度と 17 年度に整備する。また、現在は対応する MPU が宇宙用として広く使われている TX49 のみであるが、宇宙用電子部品として開発が進められている次期 200MIPS 級 MPU にも対応すべく、検討を進めている。

ELEGANT の開発作業と並行して、形式的検証サブシステムの評価を行なうために東京大学藤田研究室と「形式的検証ツールの評価方法の研究」を、将来のアナログ回路とデジタル回路の協調シミュレーションを目指して静岡大学浅井研究室と「アナログ回路のモデル化に関する研究」をそれぞれ実施している。

平成 18 年度と 19 年度には、ELEGANT の評価・教育/訓練を行なうことを検討している。

9. おわりに

本資料では、平成 16 年度と 17 年度の 2 年間で試作を行なっている ELEGANT の、平成 16 年度に実施した基本設計の結果について述べた。ELEGANT の詳細設計以降の作業は現在、平成 17 年度末の完成を目指して順調に進んでいる。

今後、ELEGANT が宇宙開発分野で広く使われることにより、宇宙用電子機器の設計品質の向上、開発期間の短縮、異常ケースの十分な検証が可能となり、衛星全体の信頼性向上に貢献できるものと考える。

また、ELEGANT の主要機能部分は、開発を担当した企業から販売が計画されており、宇宙開発分野以外の産業分野においても、広く使用されることが期待される。部品レベルにおいては、使用時の環境条件や要求される信頼度の違いから、宇宙用半導体と民生用半導体を同列に扱うことは出来ないが、デジタル回路の高位設計では両者に共通する部分は多く、ELEGANT が産業界で広く使われるようになれば、複数の産業分野に跨った設計資産(IP)の流通が期待できるものと考える。

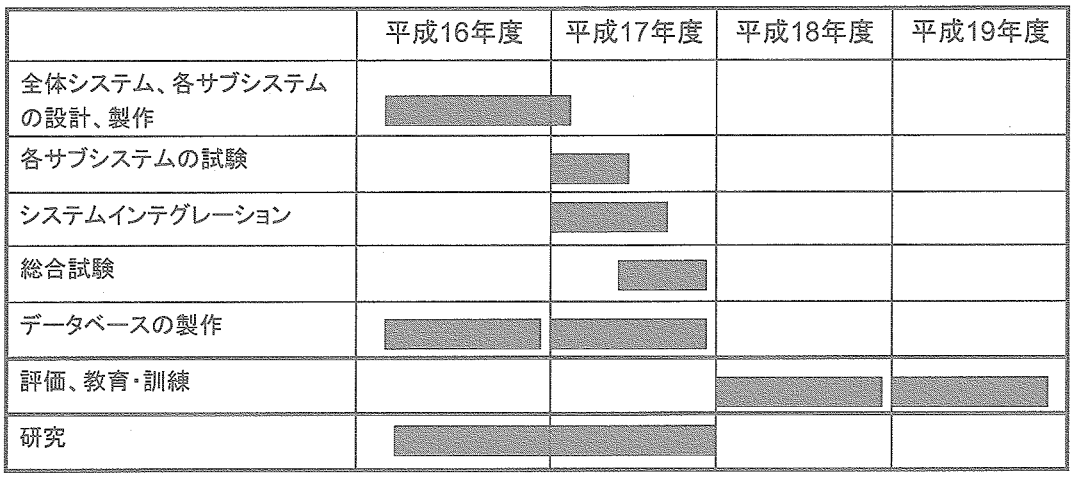

表 2 試作・研究スケジュール

略語表

ANSI-C	American National Standards Institute C
API	Application Program Interface
ASVP	Application Specific Virtual Prototype
BDL	Behavioral Description Language
BBM	Bread Board Model
BSW	Behavior State Watcher
CE	Communication Element
CPU	Central Processing Unit
DSP	Digital Signal Processor
ELEGANT	Electric Design Guidance Tool for Space Use
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GUI	Graphical User Interface
HDL	Hardware Description Language
HW	Hardware
IP	Intellectual Property
ISS	Instruction Set Simulator
LSI	Large Scale Integrated circuit
MIPS	Mega Instruction Per Second
MODEM	MOdulation and DEModulation equipment
MPU	Micro Processing Unit
OSCI	Open SystemC Initiative
PC	Personal Computer
PE	Processing Element
PROM	Programmable Read Only Memory
RTL	Register Transfer Level
RTW	Real Time Workshop
RTOS	Real Time Operating System
SIR	SpecC Internal Representation
SRAM	Static Random Access Memory
SW	Software

宇宙航空研究開発機構研究開発資料　JAXA－RM－04－023

発　行　日　2005年2月28日
編集・発行　独立行政法人宇宙航空研究開発機構
〒182－8522
東京都調布市深大寺東町七丁目44番地1
TEL　0422－40－3000（代表）
印刷所　有限会社　ノースアイランド
東京都西東京市ひばりヶ丘北4－1－9

＜本資料に関するお問い合わせ先＞
独立行政法人宇宙航空研究開発機構 情報化推進部 宇宙航空文献資料センター

宇宙航空研究開発機構
Japan Aerospace Exploration Agency