システム仕様記述言語 SpecC と組み込みシステム設計ツール VisualSpec™

(株)東芝 研究開発センター システム技術ラボラトリー
荒木 大

1. はじめに

LSI の微細化が進んだ結果、ハードウェア（HW）とソフトウェア（SW）からなる大規模なシステムが一つの LSI に載るようになりました。また、通信分野、携帯端末、マルチメディア分野といった組み込みシステム製品は、仕様の複雑さが急激に増大するとともに製品のライフサイクルの短縮化も相まって、今まで以上に開発期間短縮とコスト削減が求められています。しかしながら、このような仕様の膨張と性能の向上に対して、組み込みシステム製品の設計技術が追いついてゆけないプロダクティビティ・ギャップの心配が大きくクローズアップされてきました。SW の開発者にとっても、LSI 設計者にとっても、HW の集積度の向上、あるいは、近年の組み込みシステム製品に求められる仕様の巨大化と複雑化に立ち向かえるほど設計技術の進歩が追いついてゆけないからです。

組み込み SW の設計技術の近年の動向は、ようやくオブジェクト指向設計技術が組み込みシステムの世界でも導入されつつあるが、リアルタイムな性能あるいは HW の制約がクリティカルな組み込みの世界においてオブジェクト指向設計がどこまで寄与できるかはいまだ未知数ではないでしょうか。一方で、システム LSI の設計技術の世界においては、ここ数年の一つの変化として、C 言語あるいは C++言語を使った仕様設計が、システム LSI 設計の最上流において行なわれ始めています。このことは大きな意味を持つと考えます。なぜならば、組み込みシステム設計の最上流においては、言語の上では、SW も HW も区別がなく設計を行えるようになるからです。そこで、21 世紀に向けた組み込みシステム製品の設計力向上の鍵は、① HW/SW コデザインの導入、② 概念設計（仕様設計）から詳細設計への連続性の確立、③設計事例の再利用、の三つにあるのではないでしょうか。

HW/SW コデザインとは、HW の設計と SW の設計を別々に行うのではなく、互いを考慮しながら同時に行うことを意味します。HW/SW コデザインの一般的な手順では、システム全体をまず統一的にモデル化してから、HW として実装する部分と SW として実装する部分に分割してゆきます。さらに、HW と SW で実現される機能との間のインタフェースと同期をとるメカニズムを設計します。最終的には、HW 部分は既存の HW 合成ツールを用いて ASIC、FPGA 等の専用回路で実現され、SW 部分はプロセッサ上で実行可能なコードにコンパイルされることになりますが、コデザインの途中段階では、形式的検証あるいはシミュレーション、あるいはプロトタイピングの技術を用いて性能およびコスト面での検証を行うことで、HW と SW のトレードオフを考慮した設計が、HW と SW の詳細設計前の段階で行えます。

次に重要な点が、「概念設計から詳細設計への連続性の確立」です。概念設計のレベルにおいては、HW や SW といった具体的な実装方法は未検討の状態で、製品のユーザ・インタフェースといった機能的な側面を中心に仕様を作り上げます。次に、ここが重要なのですが、概

念設計で作成した仕様記述を、そのまま加工してゆく形で、具体的なアーキテクチャの検討が行なえるようにすることが必要です。つまり、概念設計と詳細設計の両者が、設計プロセスとして見たときにも、仕様記述のデータとして見たときにも、別々にあるのではなくシームレスに結合しておれば、各設計フェーズで行なうべき作業と成果物が明確になり、上流での設計内容と下流での設計内容の整合性を確実に保証することができます。

また、再利用型あるいは部品組み立て型の設計手法が、組み込みシステム製品の設計においてますます重要度を持ってきています。すでにLSI設計のEDA業界においては、設計資産であるIPの流通が始まっており、システムオンチップの設計にはIPの利用が必要不可欠となっています。組み込みSWの世界においても、TCP/IPのような通信用APIといったレベルのIPから、ブラウザといったアプリケーションレベルに近い領域でのIPが広く使われつつあります。こういった外部からのIPの調達の形のほかに、自部門で蓄積された過去の設計資産を効率よく再利用できる仕組みを作り上げることも重要です。

2. SpecC言語による組み込みシステム設計手法

SpecC言語はUCI（カリフォルニア大学アーバイン校）のD.ガイスキー教授を中心にして開発されたシステム仕様記述言語で、HW/SW混在型システムの仕様記述を行なえるプログラミング言語です。また、このSpecC言語を中心にした組み込みシステムの設計方法論が提案されています。この設計方法論がカバーする範囲は、組み込みシステム製品の初期設計段階である機能設計から、システムアーキテクチャの割り当て、HW/SWへの機能分割、各モジュール間を接続するインタフェース仕様の作成を経て実装レベルの設計仕様に至るまでの上位設計プロセスを対象としています。この方法論は、HWだけで製品が構成される場合あるいは組み込みSWだけで構成される場合にも適用可能な枠組みとして考えることができます。

SpecC言語による組み込みシステムの設計手法を図1に示します。システム設計の第一段階は機能仕様を記述することです。機能仕様には実装の詳細までは含まれませんが、状態遷移図とアルゴリズム記述（C言語のスタイルの記述）で仕様を表現しますので、シミュレータを使ってそのまま実行することが可能です。このステップの目的は、製品設計の初期段階におけるラピッド・プロトタイプを開発するです。つまり、製品企画あるいは機能仕様を検討している段階で、機器の動作、製品の見た目、使い勝手などをユーザと同じ視点で検証するためのバーチャル・モックアップを開発します。このようなステップを設けることは、仕様の誤認や解釈の違いによる下流での設計変更・修正の削減が図れるなどの効果が期待できます。

SpecC言語を利用した設計手法の特徴は、このラピッド・プロトタイピングで作成した機能仕様記述から、アーキテクチャの選定、機能分割、スケジューリング、コミュニケーション合成に至る各設計作業を、それぞれの前段階で作成した設計モデルに対してなんらかの置き換えあるいは洗練を行う形で定義されており、それぞれを一貫的にかつ連続的に実行できる点です。したがって、アブストラクトな機能仕様の設計から、アーキテクチャレベルの詳細仕様設計までをシームレスにつなぐことができます。

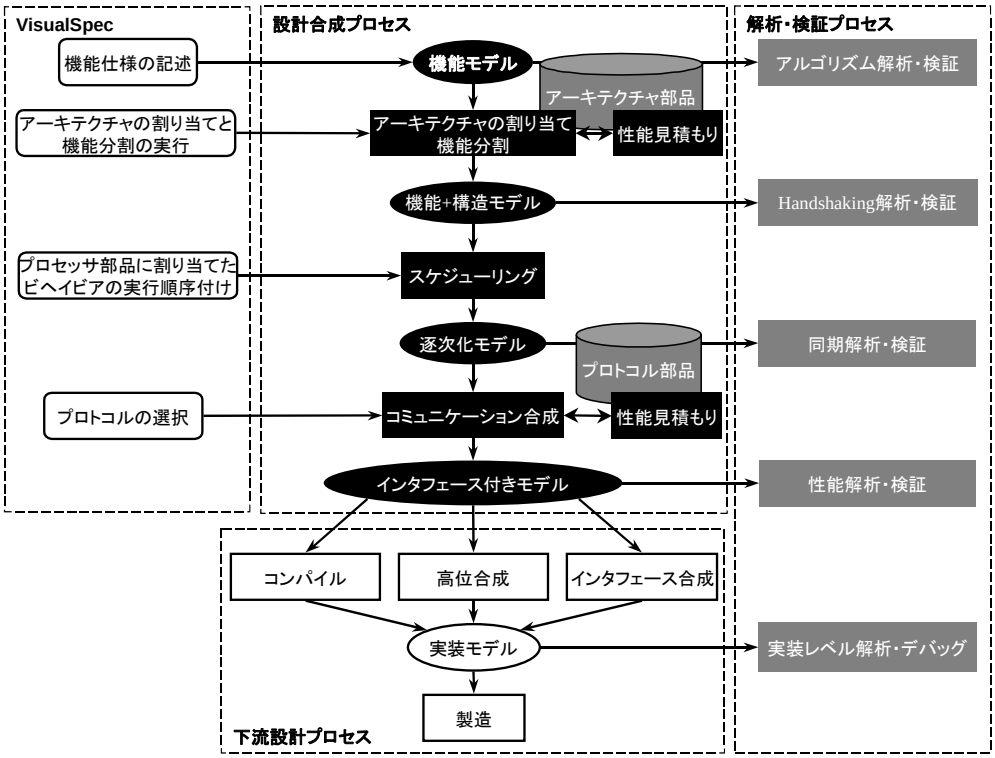

図 1　SpecC 言語を利用した組み込みシステム設計の流れ

具体的には、設計者は、図 1 の設計ステップにしたがって、仕様記述の各機能をターゲットシステムのアーキテクチャにマッピングしていくことで、設計をすすめることができます。設計合成プロセスには「アーキテクチャ割り当て(allocation)」「機能分割(partitioning)」「スケジューリング(scheduling)」「コミュニケーション合成」の 4 つのステップがあります。「アーキテクチャの割り当て」とは、システムを構成するコンポーネントのタイプとその数を決定することです。ここでいうコンポーネントとは、システムのビヘイビアを実装するために使用する、たとえばプロセッサ、ASIC、バスといった部品です、あるいは、全体を RTOS 上で動く一つの組み込み SW として構成するのであれば、RTOS のタスクがコンポーネントに対応します。「機能分割」とは、機能仕様記述で定義した各ビヘイビアを、システムコンポーネントに割り振ることを意味します。「スケジューリング」とは、個々のコンポーネントごとに割り振られたサブ・ビヘイビアの並べ替えを行って実行順序を決定することを意味します。「コミュニケーション合成」とは、ビヘイビア間の通信を実装するために、プロトコルとリソースの選択を行うことを意味します。

これらの設計合成ステップを経て最終的に出力される設計仕様(インタフェース付きモデル) は、図 1 の下部に示す下流設計ツールにそのまま渡すことができます。つまり、インタフェース付きモデルの設計仕様に含まれる SW 部分は C/C++言語のソースコードの形になり、HW 部分は VHDL のビヘイビアコードの形になります。下流設計ツールとは、SW の設計については既存のコンパイラを意味し、HW の設計については高位合成ツール、インタフェース合成ツール等が相当します。

各設計ステップでは、シミュレーションによって動作検証を行うことができます。たとえば、機能仕様の段階では、記述されている機能が設計意図に合致するかをシミュレーションで検証できます。機能分割とスケジューリングが終わった後では、異なるプロセッサ部品に振り分けられたビヘイビア間が仕様どおりに同期するかをシミュレーションで検証できます。コミュニケーション合成の後では、設計モデルを使ったシミュレーションで、演算部と通信部を含めたシステムの性能まで検証することができます。

3. SpecC 言語の特徴

SpecC 言語の文法的な特徴について簡単に説明します。SpecC 言語は C 言語をベースとしており、HW/SW 混在型システムの仕様記述を行なうために、並列性、割り込み処理、状態遷移、通信といった動作仕様を記述するための各種の構文が C 言語に追加されています。

図 2 に SpecC プログラムの基本構造を示します。SpecC 言語は、Program State Machine(PSM)という記述様式に基づいています。PSM とは、階層型並列有限状態機械と手続き型プログラミング言語を組み合わせたモデルです。システム全体は「ビヘイビア」と呼ぶブロック（図の X、Y、Z）の階層によって表現します。ビヘイビア階層のリーフに位置するビヘイビアでは、C 言語の形式でプログラムを埋め込むことができます。一方、ビヘイビア間の通信はポートとポート間の接続関係によって表現し、通信プロトコルは「チャネル」と呼ぶブロック（図の c1、c2）の中で手続きを定義することによって表現します。これらのビヘイビアやチャネルの記述には、オブジェクト指向プログラミングのスタイルを用います。

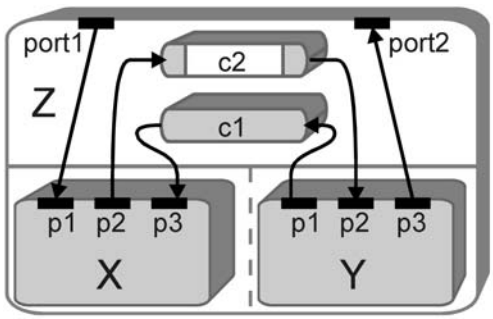

図 2　SpecC プログラムの基本構造

SpecC 言語の特長は、同一言語で、さまざまな抽象レベルのモデルが書ける点です。たとえば、システム仕様記述、システム構造記述、システム構築部品、各部品の設計仕様までを統一的に記述でき、設計の各段階でシミュレーション等により設計検証が行えます。SpecC 言語を利用することによって、仕様から設計へとシステム開発を進める際に、連続的に開発の基幹データを共有することができるとともに、IP あるいは設計データベースの構築による設計再利用も容易に実現できるようになります。また、SpecC 言語で記述された設計仕様は、HW と SW の部分に文法上の区別がありません。実装の詳細をニュートラルな状態のままで、各々の機能を HW あるいは SW として実装した場合のトレードオフを容易に分析・検討することができるのです。

SpecC 言語は、システムレベル仕様記述言語として広く言語仕様が公開されており、SpecC テクノロジ・オープン・コンソーシアム(http://www.specc.org)において標準化と普及活動が行なわれています。

4. 仕様オーサリングツール VisualSpec™

VisualSpec は、SpecC 言語を利用してシステムの仕様記述を作成し、シミュレーションを行う、あるいは、機能仕様をアーキテクチャレ

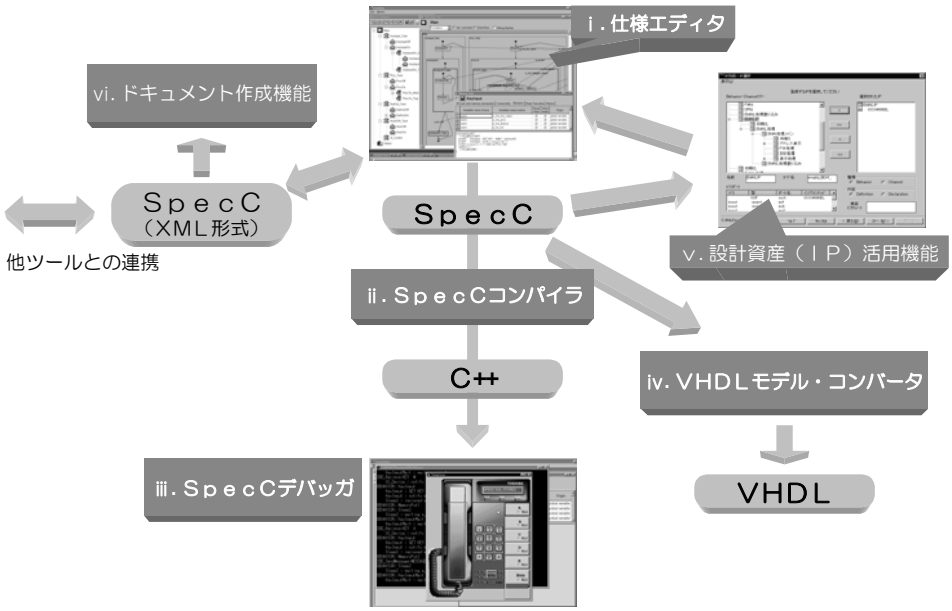

図 3　VisualSpec の機能構成

ベルの設計仕様に合成する、といった SpecC 言語を使用して組み込みシステム設計の作業を行なうためのツールです。

図 3 は VisualSpec の機能構成です。仕様エディタ(図 3-i)は SpecC 言語を使った仕様記述をエントリーするためのエディタであり SpecC 言語のソースコードを出力します。この仕様エディタは、SpecC 言語の持つ各構文に合わせて、プログラム状態間の階層関係や有限状態機械、通信、例外処理の構造と動作をビジュアルに編集することができ、仕様の記述・詳細化を支援します。仕様エディタから、コンパイラを始めとした各ツールが実行できます。仕様エディタの画面例を図 4 に示します。

SpecC コンパイラ(図 3-ii)は SpecC コードを C++コードに変換します。これを、C++コンパイラを利用すれば SW の実行モジュールができあがります。現在対応している C++コンパイラは、Microsoft® VisualC++6.0 と、Red Hat 社の GNUPro です。SpecC デバッガ(図 3-iii)は SpecC コードレベルで仕様記述のデバッガが行なえるデバッガで、仕様エディタと同様の GUI を持っています。

VHDL モデル・コンバータ(図 3-iv)は、SpecC コードの一部あるいは全部を VHDL コードに変換するツールです。IP 活用機能(図 3-vi)は、SpecC コードを切り出して部品化することによって、再利用が容易な形に変換するツールです。ドキュメント作成機能(図 3-v)は Microsoft® Word の形式で、仕様記述のドキュメントを自動作成します。

VisualSpec を使って作成した SpecC プログラムは、Microsoft® VisualBasic などの他のソフトを開発したプログラムと連動して実行させることが簡単に行なえます。たとえば、組み込み製品の GUI のモックアップを

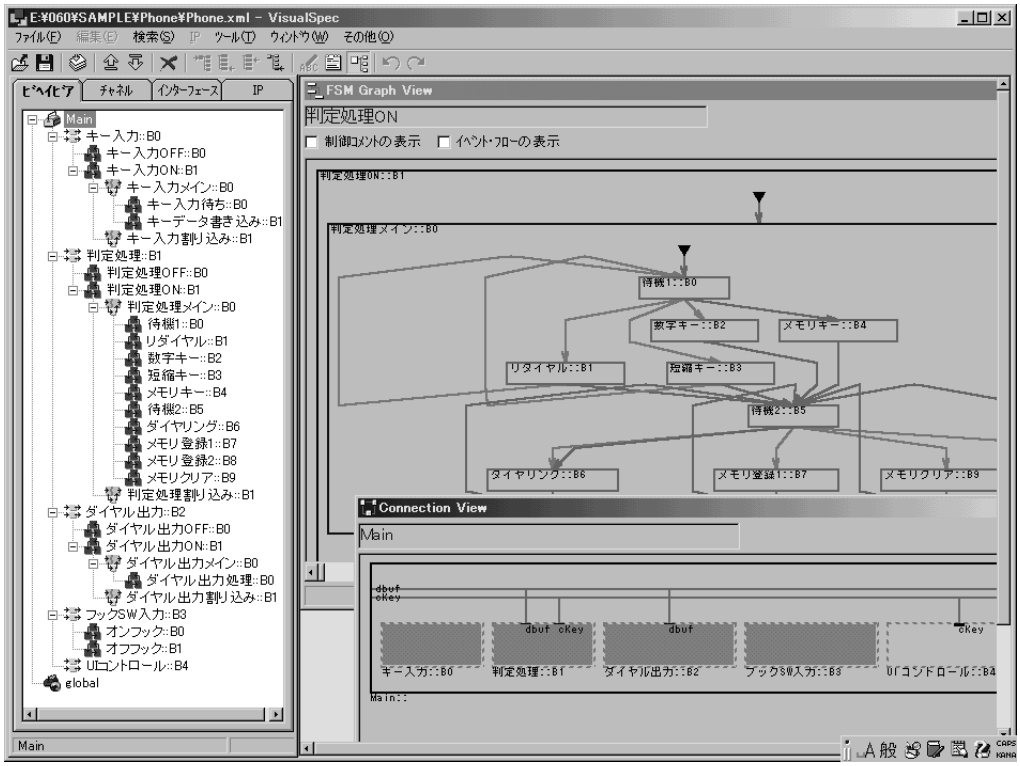

図 4　VisaulSpec の編集画面

Microsoft® VisualBasic で作成し、内部の機能仕様あるいは動作仕様は SpecC 言語を使って記述するといった使い分けを行ないます。このような設計環境を使うと、ターゲット製品のプロトタイプを非常に短期間に開発ができ、かつ内容の修正も非常に容易になります。このモックアップを使って機能あるいは製品の見た目や使い勝手のレビューが行なえるようになります。

5. VisualSpec と ZIPC の統合利用による組み込みソフトウェア開発

VisualSpec で作成した SpecC 言語のビヘイビアモデルをキャッツ(株)の ZIPC の状態遷移表(STM)の形式に変換する機能が開発されています。このデータ変換機能を使えば、VisualSpec と ZIPC を組みあわせて利用することが可能です。この組み合わせにより、組み込みシステム製品の仕様設計から SW としての実装までをシームレスに結合した設計環境が実現されます。

VisualSpec のレベルで設計する内容は、製品の仕様レベルでの設計であり、この段階では、ある程度の抽象度を持たせつつ、必要な機能・動作を明確に洗い出すことに焦点がおかれます。つぎに、VisualSpec で作成した SpecC 言語のビヘイビアモデルを ZIPC の状態遷移表(STM)の形式に変換します。ZIPC で行なうことは実装方法の明確化です。たとえば、SpecC 言語のレベルでは、タスク間の同期は SpecC 言語のイベントの概念を使って表現されています。それぞれのイベントを、RTOS のどの機能を使って実装するかといったレベルでの見直しは ZIPC 上で見直します。このように、組み込み SW あるいは RTOS 上での実装を意識した設計を ZIPC で行なうことができ、ターゲット OS 上で

の検証を行うことができます。

まとめ

SpecC 言語を利用した組み込みシステム製品の設計手法は、システムオンチップの時代に向けた大きな課題である、製品設計のプロダクティビティ・ギャップを解決する一つの道筋であると考えています。仕様の可視化とラピッドプロトタイプの活用は、顧客満足重視の商品開発が求められる中で、製品開発の最初の段階で正しく・早く・的確に製品仕様を把握・決定する手段として重要です。また、SpecC 言語を活用することで、製品の概念設計と詳細設計、あるいはSW設計とHW設計をシームレスに結合することができます。さらに、IP の活用を中心とした設計への転換も容易になると考えています。

SpecC 言語を利用するための設計ツールであるVisualSpec と ZIPC が結合されたように、今後、SpecC 言語を利用するための各種設計ツールの登場と、関連する設計ツールとの間での接続や連携が一層進むことが期待されています。SW 設計と HW 設計の垣根を越えた、組み込みシステム設計者にとって利用しやすい設計環境の整備が加速されることが望まれます。

（あらき だい）