ソフトウェア・ハードウェア間
インタフェース生成ツールの開発
- ソフトウェア・ハードウェアコデザイン -

1　背景

近年の組込みシステム開発においては，製品の企画から市場への投入までの時間（Time-to-Market）の短縮化が重要となっている．Time-to-Market を短縮化するためには，設計生産性の向上による製品開発期間の短縮化が不可欠である．

組込みシステムはソフトウェアのみならずハードウェア（デバイス）もシステム毎に専用に設計される場合が多い．これまで，ハードウェアの設計は，HDL 言語による RTL レベルの設計が中心であった．しかしながら，RTL レベルの設計は設計抽象度が低く，設計生産性に問題がある．この問題に対して，設計対象を RTL レベルより抽象度の高い動作レベルにより記述することで設計生産性を向上させる試みが注目されている．動作レベルの記述には，主に C 言語にハードウェア記述のための拡張を加えたシステムレベル言語と呼ばれる言語が用いられる．

一方，ハードウェアを制御するソフトウェアであるデバイスドライバもシステム毎に開発する必要がある．そのため，システム全体の設計効率を向上させるためには，ハードウェアのみならずデバイスドライバの設計効率も重要となる．

しかしながら，デバイスドライバ開発はそのソフトウェア規模が小さいにも関わらず，大きな開発工数がかかっている．その主な原因としてハードウェアとのインターフェース部分の設計抽象度の低さを挙げることができる．設計者はデバイスレジスタの特定のビットへのアクセスやプロセッサの割込みといった低いレベルの設計を直接行わなければならない．

デバイスドライバの設計を効率化するためには，ハードウェア設計と同様の設計抽象度の引き上げが必要となる．そこで我々は，デバイスとそのデバイスドライバは非常に関連性の強いものであることに着目し，これらをシステムレベル言語により一体に記述し，その記述から実装記述（ソフトウェアは C 記述，ハードウェアは RTL レベルの HDL 記述）をツールにより自動生成する設計手法について研究を行っている．

デバイスとデバイスドライバを一体にシステムレベル言語により記述することで，デバイスとデバイスドライバ間のインターフェースは，システムレベル言語で用意されている同期・機構を用いた抽象度の高い記述が可能となる．そしてシステムレベル言語による記述から，デバイスドライバ，デバイス，そしてインターフェースをツールにより機械的に生成することにより，設計者は低レベルの記述を行う必要がなくなり，設計生産性の向上が実現できる．

ここで，生成するソフトウェアはリアルタイム OS を用いて構築することを前提とし，生成するデバイスドライバやインターフェースのソフトウェア側はリアルタイム OS の使用を前提としたものとする．

本設計手法ではシステムレベル言語として，SpecC を用いることを想定している．SpecC は，C 言語をベースとして，ハードウェア記述のための概念と構文が追加された言語であり，ソフトウェアとハードウェアが明確に区別されていないシステムレベルの記述から，それらが区別された実装記述の手前までをカバーする言語として提案されている．図 1に SpecC の記述例とその記述を図に表したものを示す．SpecC では，システムの機能要素をビヘイビアと呼ばれるオブジェクトとして記述する．設計者はシステムをビヘイビアの階層構造としてモデル化する．また，ビヘイビア間の複雑な同期・通信は，チャネルと呼ばれるオブジェクトとして記述する．チャネルは内部に変数やメソッドを持つ．ビヘイビアはチャネルのメソッドを呼び出すことにより同期・通信を行う．同期・通信の排他制御を実現するために，チャネルのメソッドは排他

的に呼び出される．

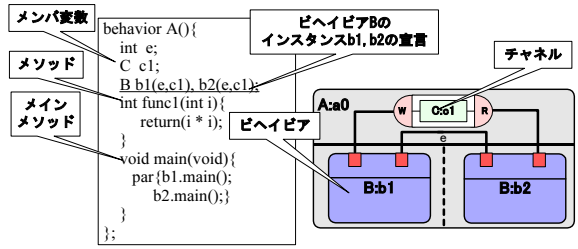

図 1: SpecC の記述例

2 目的

本プロジェクトでは，システムレベル言語による記述から実装記述への変換のうち，ハードウェアとソフトウェア間のインターフェースを自動生成するソフトウェア（インターフェース生成ツール）の開発を中心としたシステムレベル開発環境の構築を目的とする．

インターフェース生成ツールは，まず設計対象の SpecC 記述を読込み，その記述をハードウェア化する部分とソフトウェアする部分，その間のインターフェースの三つの部分に分類する．ハードウェア／ソフトウェア化の指定は，SpecC 記述でビヘイビア毎にその指定が可能になるよう言語仕様を拡張する．ソフトウェア化する部分は，SpecC 記述のまま，すでに開発したソフトウェア変換ツールに渡す．ハードウェア化する部分は高位合成ツールが処理可能な C 言語記述へ変換してから，高位合成ツールに渡す．この変換については，本プロジェクトの次の課題と位置づけている．

ソフトウェアとハードウェアの分割指定はビヘイビア単位とするため，その間を接続するチャネルの記述をソフトウェアとハードウェア間のインターフェースに変換する．変換はチャネルのメソッドの排他性，メソッド内での操作，内部変数，イベントの送信／受信などの機能を，ハードウェアとソフトウェア間で動作する記述へ変換する．

チャネルからソフトウェアとハードウェア間のインタフェースへの変換については，サンプルシステムを用いてその実現可能性があることを確認している．具体的には，サンプルシステムを SpecC により記述しその記述を手作業によりハードウェア，ソフトウェア，そしてその間のインタフェースへ変換し，実際にソフトウェア部を組込みプロセッサ上でハードウェア部を FPGA 上に実装して動作することを確認している．図 2にサンプルシステムの SpecC 記述とその変換結果を示す．点線で囲まれている部分がハードウェアとソフトウェアのインタフェースであり，本プロジェクトではこの部分の変換を行うソフトウェアを開発する．

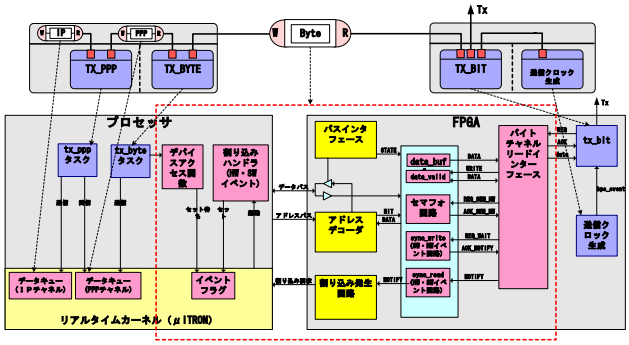

図 2: サンプルシステムの変換

インタフェース生成以外にも，コシミュレーション機能や FPGA への実装機能も開発の目的とする．

3 開発の内容

インタフェース生成の開発内容としては以下が挙げられる．

3.1 生成規則の検討

（a） インタフェースのソフトウェア側の生成規則の決定

インターフェースのソフトウェア側の生成規則を定める．生成するソフトウェアはリアルタイム OS を使用するものとし，リアルタイム OS としては μITRON を前提とする．そのため，変換はチャネルの排他性やハードウェア側からイベントの通知を μITRON の同期機構や割込みハンドラを用いて実現する．

（b） ハードウェア高位合成ツールの評価と接続方法の検討

一体設計手法では，ハードウェア化する部分は高位合成ツールにより生成することを想定している．そのため，インタフェースのハードウェア側は高位合成ツールにより生成されるハードウェアと通信を行わなければならない．現在入手可能な高

位合成ツールは，ツール毎に異なる特徴を持ったコードを生成する．変換規則はどうような高位合成ツールの出力もサポートするのが理想であるが，これは非常に困難であると考えられる．そのため，まず第一段階として，特定の高位合成ツールをサポートする変換規則を定める．

(c) インタフェースのハードウェア側の生成規則の決定

高位合成ツールの接続方法を検討した上で，インターフェースのハードウェア側の生成規則を決定する．変換はチャネルの排他性やソフトウェアからハードウェアのイベントの通知を専用回路により実現する．ハードウェアからソフトウェアへのイベントの通知はプロセッサの割込みを利用する．この変換についても基本的な予備検討は終了している．また，高位合成により生成したハードウェアからのハードウェア側のインタフェースの呼出し方法についても検討し生成規則として定める．具体的にはSpecC記述でのビヘイビアからのチャネル呼出しをFSM回路としてハードウェアに変換する．

3.2 実装

定めた変換規則をインタフェース生成ツールとして実装する．インタフェース生成ツールは，SpecC記述の読込みと解析をまず行わなければならないが，これらの機能については，既に開発したソフトウェア実装変換ツールに実装されている．そのため，この機能を流用することにより，主に本質的な変換規則の実装に注力して開発を行うことが可能である．また，コシミュレーション機能やFPGAへの実装機能も同時に実装する．

4 従来の技術との相違

従来の技術との相違をまとめると次のようになる．

SpecC言語からのシステム合成のサポート

SpecC言語によるデザインの記述はC言語による入力と比較して，言語レベルで並列記述や通信記述をサポートしているため，構造の変更が容易である．しかしながら，記述の解析が必要なことから，SpecC言語を入力してシステム合成を行うツールは研究レベルでしか存在しなかった．

インタフェースの自動生成

開発ソフトウェアでは，ソフトウェアとハードウェアのインタフェースを自動生成する．インタフェースのソフトウェア側（デバイスドライバ）はリアルタイムOSを機能を用いたものを生成する．これまでこのようにリアルタイムOS用のコードを生成するツールは存在しない．

コシミュレーション機能

一般にHDL記述のハードウェアのシミュレーションには同じくHDLで記述されたシミュレーションのためのデータを生成するテストベンチを記述する必要があった．本ツールでは，C言語で記述を動作させるOSシミュレータとHDLシミュレータを連携させ，その間のインタフェースを自動生成することにより，テストベンチとして分割前のソフトウェア側の記述がそのまま使用可能となる．

FPGAへの実装機能

開発ソフトウェアでは，システム合成後のソフトウェアとハードウェアをFPGAへに実装する機能を持つ．FPGAへの実装にはXilinx社のソフトコアであるMicroblazeプロセッサを中心としたシステムを用いている．FPGAへの実装は，高速シミュレーションという側面も持つ．システム合成後のハードウェアは，クロックレベルの抽象でHDL言語により記述されているため，シミュレーションが低速である．さらにC言語から動作合成によって生成したハードウェアは手設計のものと比較して回路規模が大きくなる傾向があるため，シミュレーション速度が低い．これらをFPGAに実装することにより，高速な検証が可能となる．

5 期待される効果

本ツールの最大の特徴は，ソフトウェア・ハードウェア間のインタフェースを自動生成することであり，これにより，システム設計全体の設計抽象度の引き上げが可能となった．

また，これまでのC言語を用いてシステムレベル設計を行うためには，数多くのツールを組み合わせ，それらのツール間を人手で連携させなくてはならなかった．そのため，設

計者は多くのツールに精通していなければならず，一人の設計者でその全てを扱うのは困難であった．本ツールでは，インタフェース生成を核として外部のツールを組み合わせて連携させることにより，SpecC 言語もしくは C 言語によりシステム全体を記述し，その記述に対してソフトウェア化，ハードウェア化の指定をした後の機能シミュレーション，システム合成，コシミュレーション，FPGA への実装の自動化を実現した．外部ツールのためのプロジェクトファイルやその呼出しは可能な限り自動化しているため，設計者は最初のデザイン記述にのみ専念することが可能となる．

ソフトウェア・ハードウェアのコデザインの観点から見ると，コデザインでは，ソフトウェアとハードウェアの最適な分割方法を探索することが目的の一つである．そのためには，システムの機能様々分割方法実装してでの性能を見積もる必要があり，図3の (2)〜(4) のループを幾度となく繰り返すため，この間を短縮することが重要となる．本ツールを用いると，SpecC 言語もしくは C 言語により設計対象を記述したのち，ツールに分割方法の指定をするだけで性能見積もりに必要なソフトウェアとハードウェアの記述を自動的に生成することが可能である．そのため，図3の (2)〜(4) のループを短時間で実現でき，様々な分割方法の検討が短時間で可能となる．

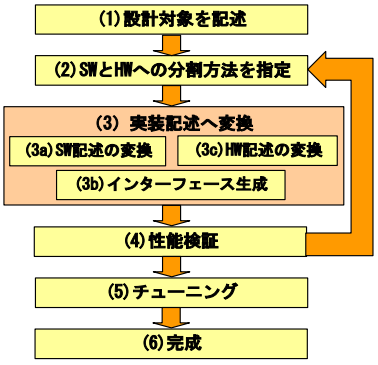

図 3: 一般的なコデザインの設計フロー

6 普及の見通し

今後の普及や実用化に関する取り組みとしては以下が挙げられる．

共同研究での使用

在共同研究としてトヨタ自動車 (株) とコデザインの研究を行っており，この共同研究で使用する予定である．

ソリトンシステムズと共同による事業展開

ソリトンシステムズと事業展開について検討しており，1 月に開催された EDS Fair 2004（http://www.edsfair.com/） ではソリトンシステムズと名古屋大学の両方のブースで本ツールを用いた デモの出展を行った．その結果幾つかのメーカーからの問い合わせを頂いた．

YXI 社との協力

eXCite を開発している YXI 社と，ツールの連携について協力していくつもりである．

Altera 社の Nios のサポート

FPGA 向けのソフトプロセッサとしては，今回使用した Microblaze の他に，Altera 社の Nios というプロセッサがある．現在，市場では Microblaze より Nios が普及している．そのため，市場での普及を考えると Nios のサポートは必要不可欠なため，今後サポートしていく予定である．

7 開発者名

本田 晋也 （橋技術科学大学大学院 工学研究科 電子情報工学専攻 honda@ertl.jp）

（参考）

- **開発者 URL**
  http://www.ertl.jp/ honda
- **ツール URL**
  http://www.ertl.jp/ honda/codesign/