招待論文

# 組込みシステム開発技術の現状と展望

高田広章†

システムLSIの適用分野の多くは組込みシステムであり，システムLSIの設計手法を議論する際には，ソフトウェアまで含めた組込みシステム全体の特性や要求事項を理解することが重要になる．本論文では，組込みシステムとそのソフトウェア開発の特性を整理し，組込みシステム開発を取り巻く最近の状況について述べる．また，組込みシステム開発が今後目指すべき姿を提示し，それを実現するために研究・開発が求められる技術課題について，ソフトウェア開発の面に重点を置いて議論する．

# The Recent Status and Future Trends of Embedded System Development Technology

Hiroaki Takada†

Because embedded systems are major application areas of system LSI, it is important to understand the characteristics of and requirements on embedded systems including their software in discussing the design methodology of system LSI. This paper summarizes the characteristics of embedded systems and their software development, then describes the recent circumstances of embedded system development. We also present the desirable development flow of embedded systems and discuss the technical issues which require further research and development to realize the presented flow focusing on their software development.

## 1. はじめに

近年のLSI技術やマイクロプロセッサ技術の発展により，システムLSIの適用分野が急速に広がりつつあるが，その多くは組込みシステム（embedded system）に分類される．

コンピュータシステム全体を1つのLSI上に集積するというシステムLSIの特性から，その設計手法を議論する際には，システム全体の特性や要求事項の理解が欠かせない．また，システム開発の効率化や設計されるシステムの高性能化のためには，ソフトウェアを含めたシステム全体の開発手法との関連でシステムLSI設計手法を議論すべきである．

このことから本論文では，システムLSI設計手法に関する今後の研究に資することを目的として，組込みシステムとそのソフトウェア開発の特性を整理し，その開発の特徴や要求事項について述べる．また，組込みシステム開発を取り巻く最近の状況について述べ，組込みシステム開発が今後目指すべき姿と，それを実現するために研究・開発が求められる技術課題について議論する．なお本論文では，組込みシステムとその開発技術に関して，特にソフトウェアの面に重点を置いて議論する．

† 豊橋技術科学大学情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology

## 2. 組込みシステムと組込みソフトウェア開発の特性

### 2.1 組込みシステムの多様性

組込みシステムとは，各種の機器に組み込まれてその制御を行うコンピュータシステムのことをいう．ただし，この定義には曖昧性があり，どこまでを組込みシステムであると考えるかは人によって異なる．たとえば，最近のプラント制御システムにはパーソナルコンピュータ（PC）ベースのハードウェアが用いられる場合も多いが，この場合，ハードウェア的にはPCそのものであり，組み込まれているとはいいにくい．しかし，プラントという「機器」に組み込まれてそれを制御しているコンピュータシステムと考えれば，組込みシステムに含めることができる．実際プラント制御システムは，次節に述べる組込みシステムの特性の多くに合致する．また，PDAやゲームマシンはコンピュータそのものであるとも考えられ，組み込まれて

いるといえるかどうか明確でない．

組込みシステムをこれらの境界例も含むものと広く定義すると，PCやワークステーションなどのいわゆる汎用システム以外の，ほとんどすべてのコンピュータシステムが組込みシステムであるということができる．言い換えると，ある目的に専用化されたコンピュータシステムを，組込みシステムと呼んでいることになる．このことから，dedicated systemという用語が，組込みシステムと同等の意味で使われる場合もある．

組込みシステムは，制御対象の機器の多様性を反映して，システムの規模においても，システムに対する要求事項においても，きわめて多様である．そのため，組込みシステム全体を対象として，その特性や開発技術を議論することは容易ではない．それにもかかわらず，組込みシステムを広くとらえて議論するのは，現時点で組込みシステムを分類する有効な指標が提案されていないためである．「自動車応用」などといった応用ドメインによって組込みシステムを分類する方法はあるが，自動車応用の中でも，システムによって（たとえば，エンジン制御，エアバッグ制御，カーナビゲーションシステムの3つのシステムは）その特性や要求事項は大きく異なる．そのため，システムの開発技術を議論する際の分類としては意義が低いと考えられる．

このように，開発技術を議論する上で組込みシステムをどのように分類すべきかは未解決の課題となっているが，開発手法を大きく左右する指標として，システム（ないしは，システムが組み込まれる対象の機器）の製造個数をあげることができる[1)]．組込みシステムの製造個数は，少ないものでは1つ（たとえば，プラント制御）から，多いものでは100万個を超えるもの（たとえば，携帯電話）まで広い範囲に分布するため，開発手法に与える影響が大きい．

具体的には，システムの製造個数が多い場合には，システム1つあたりの製造コスト（部品のコストや組立てにかかるコストなど）を下げることが重要となるため，システムのハードウェアおよびソフトウェアの最適化設計を進めて製造コストが下がれば，開発コスト（システム設計のコストなど）が少々上がってもメリットがある．一方，製造個数が少ない場合には開発コストを下げることが重要となるため，製造コストが少々上がっても，標準的なハードウェアやソフトウェアを活用することが望ましい．

組込みシステムにシステムLSIが用いられるのは，(a) システムの製造個数が多く，開発コストが上がってもシステムの最適化設計を進める意味があるか，(b) きわめて高い計算性能が必要であるなど，システムに対する要求が特殊であるために標準的なハードウェアが適用できない場合である．日本においては，民生機器など (a) に該当する分野が興味の中心である☆．

### 2.2 組込みシステムの特性

この節では，多くの組込みシステムに共通する特性について述べるが，前述のとおり組込みシステムの特性はシステムごとに大きく異なり，これらがあてはまらないシステムも多いことに留意されたい．

(1) 専用化されたシステム

組込みシステムは，一般に，それが組み込まれる対象の機器を制御するという1つの目的に専用化して設計される．前述のとおり，これをもって組込みシステムの定義とする立場もある．

組込みシステムのソフトウェアは機器に固定されており，ソフトウェアが入れ換えられることは稀である．そのため，ハードウェアはソフトウェアが用いる必要最小限の機能を，基本ソフトウェアはアプリケーションソフトウェアが必要とする機能のみを備えていればよい．

(2) 厳しいリソース制約

多くの組込みシステムには，機器のコストダウン要請から，厳しいリソース制約が課せられている．特に大量生産される民生機器においては，わずかな製造コストの削減でも大きなトータルコストの削減につながるため，ハードウェアの低価格化（遅いプロセッサを使う，メモリを少なくする）に対する要求が厳しい．

また，動作温度条件や低消費電力化，EMC対策といった動作条件からくる制約や，小型・軽量化への要請により，ハードウェアリソースが厳しく制限される場合もある．たとえば，バッテリで駆動される携帯機器においては，充電なしに使用できる時間の延長に加えて小型・軽量化にもつながるなど，低消費電力化が商品価値を上げることに直結しており，低消費電力化に対する強い要求がある．また，自動車の制御システムにおいては，エンジンルームが高温になる状況や寒冷地においても動作しなければならないことから広い温度範囲（たとえば，零下40度〜125度）で動作することが求められ，その結果最先端の半導体プロセスが使えない場合もある．

(3) 高い信頼性

機器を制御するという性質から，組込みシステムに

☆ これに対して米国における組込みシステム研究は，軍事応用など (b) に該当する分野を主な対象としており，組込みシステムのイメージがかなり異なっていることに注意が必要である．

は高い信頼性が求められるのが通常である．プラントや自動車の制御といった応用の場合には，システムの誤動作が人命にかかわる事態につながることはいうまでもないが，家電製品などの民生機器の場合にも，システムの誤動作が重大な事態を引き起こす可能性を否定できない．

また，大量生産品の場合には，一度販売した機器を回収してソフトウェアを修正するには高いコストがかかる．そのため，機器の出荷前に徹底的にシステム検証を行うことが必要である．

さらに，技術的な要因ではないが，組込みシステムが高い信頼性を求められる理由の 1 つに，組込みシステム（ないしは，システムの制御対象機器）の信頼性に対するユーザの高い期待があることもあげられる．

高い信頼性を求められる結果として，システム開発コストの中で，検証のためのコストが占める割合が高くなる傾向がある．検証コストが高いことは，システム開発にも様々な影響を与える．たとえば，寿命の長いシステムの開発にあたっては，将来にわたって安定して供給される部品やサポートされるツールの利用が重視される．一例であるが，10 年前に開発されたシステムのソフトウェアに問題が発見されると，それを解決するために開発時に用いたコンパイラが必要とされる場合がある．同じコンパイラの新しいバージョンが存在していたとしても，新しいバージョンで再コンパイルすると生成されるコードが違ったものとなるために多くの部分の再検証が必要となり，余計なコストがかかる．

(4) リアルタイム性

リアルタイム性とは，単に計算処理速度が速いことやレスポンス時間が短いことをいうのではなく，システムが定められた時間制約に従って動作する性質をいう[2]．多くの組込みシステムは，やはり機器を制御するという性質から，制御対象の機器によって定まる時間制約に従って動作することが必要であり，リアルタイム性が求められるのが通常である．

厳しいリアルタイム性を持ったシステムの設計にあたっては，設計の早い段階から，時間制約を満たすように考慮して設計を進めることが必要である．すなわち，システム設計手法の中に，時間制約を満たすための手順が組み込まれていることが必要である．

以上の中で，(2) と (4) の特性は，汎用システムで研究・開発されてきた技術を組込みシステムに適用する際の阻害要因になる場合が多い．たとえば，オブジェクト指向プログラミング言語は，リソースの使用量が増えるために，組込みシステムの分野においては適用が進んでいない．また，プログラミングの抽象度を上げると，プログラマがシステムの時間的な振舞いを理解することが難しくなり，リアルタイム性確保を困難にするという問題もあげられる．

(2) と (4) の特性がシステム設計上の制約条件になるのに対して，(1) の特性は，これらの制約条件を満たすために活用することができる．すなわち，ハードウェアや基本ソフトウェアをシステムに専用化することで，必要な処理能力やメモリ容量を大幅に減らしたり，リアルタイム性の達成に必要な予測可能性を向上させたりする可能性がある．

### 2.3 組込みソフトウェア開発の特性

次に，組込みシステムのソフトウェア開発が持つ，汎用システムには見られない特性について議論する．これらの特性の多くは，前節で議論した組込みシステムの特性から導かれるものである．繰返しになるが，これらの特性があてはまらないシステムも多い．

(1) ハードウェアに密着したプログラミング

組込みシステムは，システムごとにハードウェア構成や周辺デバイスが異なるため，ソフトウェア開発の中でデバイスドライバなどハードウェアを直接扱う部分の比率が大きい．ハードウェアを直接扱うプログラミングは，ハードウェアに関する知識が必要であることに加えて，ノウハウに負う部分が大きく，開発が難しいといわれている．

(2) 開発環境とターゲットシステムの分離

組込みシステムのソフトウェア開発においては，ターゲットシステムがソフトウェア開発に必要な機能を持っているとは限らず，開発環境とターゲットシステムが分離しているのが通常である．そのため，クロスコンパイラやリモートデバッガ，ICE（In-Circuit Emulator）などの組込みソフトウェア開発に独特な開発支援ツールが用いられる[3]．

また，リアルタイム性を必要とするシステムの中には，システムの検証・デバッグ中といえどもシステムを停止させることができないものがある．上記のツールの中で ICE は，このようなシステムの検証・デバッグを支援するためのツールとしても重要である．バスがチップ外に出ていないシステムで ICE を利用できるようにするためのエバチップやオンチップデバッグ機能も，組込みシステムに独特のものである．

(3) ハードウェアとの並行開発とコデザインの可能性

組込みシステムのハードウェアは，そのシステム専用に開発されることが多く，そのような場合には，

ハードウェアの完成前にソフトウェア開発を始める並行開発が行われるのが一般的である．さらに，ハードウェアの完成前にソフトウェアの検証を開始するために，ソフトウェアの検証をシミュレータ上で行う場合も多い．ハードウェアが特殊なものである場合には，コシミュレーション環境を用いてのコベリフィケーションも行われる．

さらに，ソフトウェアを効率的に実行するために求められるハードウェアの要件を明確にし，それに従ってハードウェアを設計するコデザイン技法も広く適用されている．組込みシステム開発におけるコデザインの意義については，3.4 節で述べる．

(4) 多様なプロセッサや基本ソフトウェア

組込みシステムの規模や要求事項が多様であることから，用いられるプロセッサや基本ソフトウェアもきわめて多様である．そのため，ソフトウェアの再利用性を上げるためには，プロセッサや基本ソフトウェアに依存しないようにすることが不可欠である．たとえば，組込みシステム用のリアルタイムカーネルの多くは，プロセッサの違いを隠蔽するためのレイヤの上にプロセッサに依存しない部分を載せる形で実現されているし，ミドルウェアはリアルタイムカーネルの違いを隠蔽するためのレイヤを持っていることが多い．

さらに，DSP（Digital Signal Processor）やメディアプロセッサなどの特殊なプロセッサも用いられるが，このようなプロセッサにはコンパイラがないか，あったとしても生成されるコードの効率が悪い場合が多く，いまだにアセンブリ言語でのソフトウェア開発が必要となっている．

(5) システム内のソフトウェアは信頼できるという前提

組込みソフトウェアは，制御対象の機器に固定されて提供され，それを制御することのみを目的に開発されるのが通常である．そのため開発者は，システム内でどのようなソフトウェアが動作しているかを把握している☆．このことから，システムの動作を妨害するようなソフトウェアが組み込まれるおそれはなく，ソフトウェアの検証が完了すれば，システム内のソフトウェアは信頼できるという前提が成り立つ．組込みシステム用のリアルタイムカーネルの多くが保護のための機構を持っていないのは，保護機構を実現するためのオーバヘッドが多いうえに，この前提があるためである．

☆ 最近はこの前提が成り立たない組込みシステムも増えつつある．

## 3. 組込みシステムの IP ベース設計

### 3.1 組込みシステム開発の変化

最近，組込みシステムの制御対象となる機器の高機能化や複合化が急速に進行している．この背景には，機器メーカが，デジタル化やネットワーク化により機器の付加価値を高めようとしていることがある．また，機器メーカのシェア競争の激化から，機器の開発期間の短縮や低コスト化に対する要求も強くなっている．

このような制御対象機器の変化に対応して，組込みシステム開発にも変化が生じている．機器の高機能化や複合化は，組込みシステムの大規模化・複雑化をもたらす．特に，機器のデジタル化はメカやアナログ回路で実現されていた機構がコンピュータシステムで置き換わることであり，機器のネットワーク化は組込みシステムのネットワーク化にほかならない．つまり，機器に追加される付加価値の多くは，それを制御する組込みシステムによって実現されており，システムの大規模化・複雑化は必然的な結果である．また，機器に対する開発期間短縮や低コスト化の要求は，組込みシステムに対する要求に直結する．

さらに，近年の LSI 技術とソフトウェア技術の発展は，ハードウェアとソフトウェアの境界を流動化させている．従来は，性能的にハードウェアで実現するしかなかった機能が，ソフトウェアで実現できるようになっている．一方で，ハードウェアで実現した方がトータルには低コスト化できる場合もあるし，低消費電力の面からはハードウェアで実現した方が有利な場合もあるなど，ハードウェアで実現するかソフトウェアで実現するか選択可能な場面が増えている．

### 3.2 組込みソフトウェア開発のオープン化

従来の組込みソフトウェア開発においては，機器メーカがすべてのソフトウェアを自社開発することで，ソフトウェアの品質や信頼性を確保するという考え方が広くとられてきた．システムを構成するすべてのソフトウェアモジュールを自社開発することで，何らかの障害があった場合にもその原因を確実に突き止め，それを取り除くことができる．また，各ソフトウェアモジュールの振舞いが予想できるため，予期できない要因により時間制約が満たせなくなるといった障害を最小限にできる．

ところが，前節で述べたような組込みシステム開発の変化は，このような従来型のソフトウェア開発を困難にし，他社で開発されたソフトウェア IP（ソフトウェア部品）の導入を余儀なくしている．これが，組込みソフトウェア開発のオープン化である[4]．ソフト

ウェア開発のオープン化は，汎用システムの分野では1980年代に始まったものであり，組込みシステム分野にもオープン化の流れが到達したととらえることができる．

他社からのソフトウェアIPの導入が余儀なくされる最大の理由は，制御対象機器の複合化により，すべてのソフトウェアを自社開発するには従来とは違う種類の技術が求められることである．典型的には，ネットワーク接続される機器のソフトウェアをすべて自社開発するには，それまでネットワークとは無縁の機器を開発していた機器メーカが，新たにネットワーク技術を習得しなければならない．しかも，それらの技術は進歩の速度が速いために陳腐化も速く，進歩に追従するために大きな開発投資を必要とする．そのため，他社で開発されたソフトウェアIP（ネットワーク接続する場合にはプロトコルスタックなど）の導入を選択することになる．

同様のことは，ハードウェアとソフトウェアの境界の流動化によっても起こっている．これまで外部で開発されたハードウェア部品によって実現してきた機能を，性能的にはソフトウェアで実現可能となっても，その実現技術が自社内にあるとは限らない．

もう1つの理由として，すべてを自社開発するアプローチでは，ソフトウェアの開発期間が長くなることが避けられないことがあげられる．ソフトウェアが大規模化・複雑化する中で，開発期間短縮の要求に応えるためには，他社のソフトウェアIPの導入が不可避となっている．

### 3.3 IPベース設計とリアルタイムカーネル

これと類似の状況は，システムLSIの設計にも見られる．具体的には，システムを構成するすべてのハードウェア要素を1つのLSI上に集積できるようになっても，そのすべての設計技術が1つの会社内にあるとは限らない．その結果，他社の開発したハードウェアIP（ハードウェア設計資産）を導入せざるをえない状況にある．技術の進歩の速度が速いために陳腐化も速く，進歩に追従するために大きな開発投資を必要とするのも，ソフトウェアの状況と同様である．また，大規模化・複雑化するシステムLSIを短時間で設計するためにも，他社の設計したハードウェアIPの導入が避けられない．

これらのことから，組込みシステムの開発は，ハードウェアとソフトウェアのいずれにおいても，既存のIPを利用したIPベースの設計へと転換しつつある．すなわち，システムを構成する要素の中で，既存のIPが利用できるものについてはそれを利用し，新しく設計する部分を極力減らすようにする．既存のIPが，一部の要求に合致しないためにそのままでは利用できない場合には，一部修正を加えて利用することになる．

ハードウェアIPの中で，プロセッサコアは，ソフトウェアIPを実行するための機構であるという意味で，特別な位置付けにある．同様にリアルタイムカーネルは，複数のソフトウェアIPを1つのプロセッサ上で実行するための機構であるという意味で，特別な位置付けにある．最近，小規模な組込みシステムにおいてもリアルタイムカーネルが利用される割合が増加しているが[5]，これは，自社内でそれまで開発してきたソフトウェアに，プロトコルスタックなどのソフトウェアIPを導入するケースが増えているためである．

一般に各ハードウェアIPには，システム設計を効率化するために，それを操作するための最低限のソフトウェアであるデバイスドライバが付属していることが望ましい．デバイスドライバには，同等の機能を持ったハードウェアIPの間の小さな差違を隠蔽するという役割もある．リアルタイムカーネルは，見方を変えると，プロセッサコアのデバイスドライバであるととらえることもできる．

### 3.4 コデザインの意義

組込みシステムにおけるハードウェアとソフトウェアの並行開発とコデザインの可能性については，2.3節で述べた．ところが現状の組込みシステム開発では，ハードウェア設計が完了してからソフトウェアの開発が行われることが多く，ソフトウェア開発側からの要求がハードウェア設計に反映されにくいという問題がある．また，ハードウェアの実装が終わるまでソフトウェアの検証を行うことが難しく，開発期間が長くなる原因の1つとなっている．

コデザインの主な目的は，ハードウェアとソフトウェアを一体で設計することにより，システム全体としてより最適な設計を可能にすることである．また，コシミュレーションなどの技術により，ハードウェアの完成前にソフトウェア検証を開始することを可能にし，並行開発の促進による開発期間の短縮も狙っている[6]．

コデザイン技術のもう1つの重要な目的として，ハードウェアとソフトウェアの境界が流動化する中で，システムのある機能要素をハードウェアで実現するかソフトウェアで実現するかの決定をできる限り遅らせたいという要求がある．これは，要求性能を達成するために鍵となる機能要素を，ハードウェアで実現することが必要なのか，ソフトウェアによる実装でも要求を達成できるのかを，システム設計の後の段階になるほど正確に分析・評価でき，システムの最適な設計につ

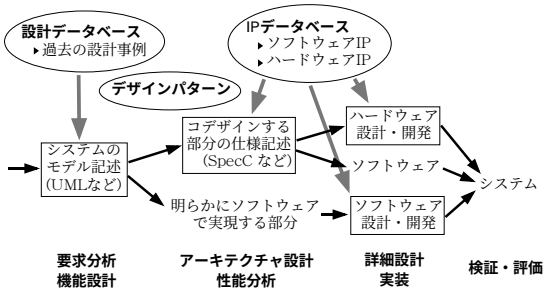


**図 1** 組込みシステム開発の目指すべき姿
Fig. 1 Desirable development flow of embedded systems.

ながるためである．また，制御対象機器がシリーズ化して開発されることも重要な動機の 1 つである．具体的には，上位機種では性能を上げるためにハードウェアで実現した機能要素を，下位機種ではコストダウンのためにソフトウェアで実現する場合がある．また，前世代の機種のハードウェア設計時には仕様が確定できなかったためにソフトウェアで実現した機能要素を，次世代の機種ではハードウェアで実現する場合もある．

### 3.5 組込みシステム開発の今後の姿

前節までの議論より導かれる組込みシステムの開発が今後目指すべき姿を**図 1** に示す．この図は，システム LSI を用いた組込みシステム開発の流れを示したものであるが，ここでは図に示した開発の流れについて解説し，それを実現するために研究・開発が必要な技術課題については 4 章で議論する．

開発の最初の段階として，システム（ないしは，システムの制御対象機器）全体に対する要求事項を分析し，それを基に機能設計を行う．この段階では，過去の設計事例も参考にして，UML[7]などを用いてシステムのモデル記述を行う．また，時間制約や消費電力などのシステム設計に対する制約条件も，この段階で記述する．モデル記述は，システムの振舞いを完全に記述するものではないが，例外的な条件であっても，信頼性やリアルタイム性を確保するために重要となる部分は記述されている必要がある．

次の段階へ進む前に，モデル記述から，ソフトウェアで実現すべきであると明らかに判断できる部分の切り分けを行う．切り分けた部分は，通常のソフトウェア開発と同様の流れとなる．残った部分は，明らかにハードウェアで実現すべき部分と，諸々の条件を考慮してハードウェアとソフトウェアのいずれで実現するかを決定すべき部分である．この残った部分を，ここでは，コデザインする部分と呼ぶことにする．

コデザインする部分については，まず SpecC[8]などのシステム記述言語を用いて厳密な仕様記述を行う．次に，この仕様記述をベースにアーキテクチャ設計や性能分析を行い，設計制約を満たすことができるハードウェアとソフトウェアの機能分割を決定する．ハードウェアとソフトウェアの機能分割については，数多くの研究がなされているものの，設計のこの段階で最終的な性能やコストを正確に見積もることは難しく，最適な機能分割を自動的に決定することは困難と考えられる．また，仕様変更の可能性がある部分はソフトウェアで実装する方が望ましいといった定量化しにくい尺度もあり，当面は人手（すなわち，設計者の判断）に委ねることになるだろう．ここで，設計者が正しい判断を下すことを支援するために，設計者が示した機能分割で設計を進めた場合に得られる性能やコストを，なるべく正確に予測する技術が重要になる．

このようにシステムが構成要素に分割されていく過程で，最初に記述したシステム設計の制約条件は，それぞれの機能要素の設計制約にブレークダウンしていく．このブレークダウンの際にも，機能要素の性能予測が重要な役割を果たす．

機能分割が決まると，ソフトウェア部分は仕様記述からほぼ自動的に生成できる．また，ハードウェアとソフトウェアをつなぐ部分も，機械的な生成が可能になると予想される．それに対してハードウェア部分については，十分な品質のハードウェアを自動生成する技術が開発されるまでにはまだ時間を要すると考えられ，人手によってアーキテクチャ設計やタイミング設計を進める必要がある．設計を進める間にも，そのまま設計を進めた場合に得られる性能やコストを随時予測し，設計制約を満たせないことが分かった場合には，すぐに前の段階に戻って再設計する必要がある．

システム記述言語による仕様記述は，そこからハードウェアを自動生成することはできなくても，設計すべきハードウェアの仕様を厳密に定めることで，ソフトウェア設計者との間で仕様の誤解を防ぐために有効である．また，ハードウェアの実装が完了する前に，開発したソフトウェアを検証するためのコシミュレーションに用いることもできる．

IP データベースは，モデル記述より後の各段階において参照される．IP データベースには，通常のハードウェア IP やソフトウェア IP に加えて，システム記述言語で記述された仕様記述と，その設計結果であるハードウェア IP およびソフトウェア IP も登録されるべきである．また，IP データベースには，各 IP の性能見積り（処理時間，消費電力，面積など）やテストベクタも登録しておくことが必要である．特に各 IP の性能見積りは，設計の早い段階で正確な性能予測を

行うために重要である．

## 4. 研究・開発が求められる技術課題

組込みシステム開発にかかわる問題の多くは，汎用システムのソフトウェアにおいては古くから存在し，多くの学術的な取り組みがなされてきたものである．しかし，それらの成果の中で，組込みシステム分野に適用できるものもあれば，そのままでは適用できないものもある．組込みシステムにおける厳しいリソース制約やリアルタイム性が適用にあたっての阻害要因となるのは，2.2 節で述べたとおりである．

以下では，3.5 節で解説した組込みシステム開発の流れを実現するために必要なものを中心に，組込みシステム開発に特有の技術課題について述べる．

(1) ハードウェア化しやすいモデル記述

図 1 の最初の段階で重要となる課題は，どのようなモデル記述を行うとハードウェア化しやすいか（逆に，どのようなモデル記述を行うとハードウェア化しにくいか）を明らかにすることである．UML を用いたシステムのモデル化には様々なアプローチがあるが，最終的にハードウェアで実現する可能性を考えると，素直にハードウェアに対応するモデル化を行っておくのが望ましい．ないしは，ハードウェア化しにくいモデル記述を，ハードウェア化しやすい形に変換する技術を開発するアプローチも考えられる．

(2) システム記述言語

図 1 に示したコデザインする部分の仕様記述は，ハードウェアにもソフトウェアにも自然に落とせる記述とすることが重要である．たとえば並列性に対する考え方は，ハードウェアで実現する場合とソフトウェアで実現する場合はかなり異なるが，この段階での記述は，実現方法に依存しないことが望ましい．

システム記述言語として，最近，SpecC[8]や SystemC[9]といったプログラミング言語をベースとしたものが注目されている．これらが注目される理由の 1 つは，ソフトウェア記述に向いた言語をハードウェア記述ができるように拡張することで，ハードウェアとソフトウェアの分割前の記述が可能になると期待されることである．しかし現時点では，分割前の記述がエレガントにできるとは限らず，今後の研究・開発が必要な課題となっている．

(3) 構成可能なハードウェアと基本ソフトウェア

専用化されたシステムを効率良く開発するためには，ハードウェアや基本ソフトウェアが，アプリケーションソフトウェアを効率良く実行できるように構成可能（configurable）であることが望まれる．実際，構成可能なプロセッサ[10),11)]や構成可能なオペレーティングシステム[12)]に関する研究・開発が行われている．ハードウェアと基本ソフトウェアを構成可能とした場合の問題として，設計検証が困難になることがあげられる[13)]．また，構成可能となるようソフトウェアを記述した場合，構成によっては無駄なコードが生じるのは避けられず，それを取り除くためにコンパイラによるいっそうの最適化が求められる．

(4) リアルタイム性を考慮した上流設計手法と CASE ツール

ソフトウェアの上流開発工程を支援する環境として，オブジェクト指向分析・設計手法や，それらに基づいた CASE ツール，状態遷移図や状態遷移表をベースにした CASE ツール，制御システム用の CAD ツールの利用が有力である．これらの中には，組込みシステムへの適用を狙ったものもあるが[14)]，現状ではそれらの手法やツールにおけるリアルタイム性の扱いが不十分である．具体的には，時間制約や各処理にかかる時間を記述する機能が十分でない場合が多い．

(5) 処理時間・性能の予測・見積り・解析技術

ある機能をハードウェアまたはソフトウェアで実現した場合にかかる処理時間は，組込みシステム開発の上流から下流までの各段階で必要とされる情報である．コデザインにおいて，ハードウェアとソフトウェアの機能分割を最適化するためには，ある機能をハードウェアで実現した場合と，ソフトウェアで実現した場合の性能を，設計が固まらない段階で予測することが必要である．また，システムの設計が詳細化されていく段階でも，システム設計の妥当性を検証するために，処理時間の見積りが望まれる．さらに，システムのリアルタイム性を検証するためには，各処理にかかる最大時間が正確に分かることが必要である．

このように，システム設計の各段階において，段階に応じた精度で処理時間を知りたいという要求があるが，この分野の研究はあまり進んでいないのが現状である．リアルタイムシステム分野において，プログラムの最大実行時間を解析する技術については研究されているが，どうしても悲観的な評価結果になってしまう[15)]．

(6) 専用プロセッサやマルチプロセッサのためのソフトウェア技術

DSP やメディアプロセッサなどの専用プロセッサ上のソフトウェア技術，具体的には最適化コンパイ

ラ技術やオペレーティングシステム技術の研究・開発が望まれる．

また，1つのチップ上に複数のプロセッサを載せる状況も増えつつおり，複数のパイプラインを持ったプロセッサも開発されている．これらに対応するためには，新たなコンパイラ技術やオペレーティングシステム技術が必要になると予想される．さらに，汎用プロセッサと専用プロセッサを組み合わせたヘテロジニアスなマルチプロセッサのためのソフトウェア技術も重要性な課題となっている．

(7) システムの信頼性確保のための技術

2.2 節で述べたとおり，組込みシステムには高い信頼性が求められる一方で，より複雑なシステムを短期間で開発することも求められており，システムの信頼性確保のための技術が重要性を増している．

以上であげた以外にも，リアルタイム性を考慮した組込みシステム向けのソフトウェアのコンポーネント化技術，各システム階層における低消費電力設計技術，ハードウェアに密着したソフトウェアの開発効率向上，組込みシステムに向いたソフトウェアデバッグ環境，組込みシステムの特性を考慮したアーキテクチャなど，取り組むべき技術課題は多い．

## 5. おわりに

本論文では，組込みシステムとそのソフトウェア開発の特性を整理し，組込みシステム開発の IP ベース設計の流れとコデザインの意義，組込みシステム開発が目指すべき姿について議論した．また，組込みシステム分野，特に組込みソフトウェア分野において，今後研究・開発が望まれる技術について述べた．

システム LSI を用いた組込みシステムの開発において，ハードウェアとソフトウェアのコデザインは最も重要な技術の1つである．コデザイン技術，特に 4 章の (1) や (2) の技術課題に取り組むためには，ハードウェア設計技術の研究者とソフトウェア開発技術の研究者の協調が重要となる．ところが，ハードウェア設計側とソフトウェア設計側は，お互いに相手の状況を理解しておらず，議論がかみあわない場面が多い．相互理解を深めようとしても，まず用語の違いが障害になる．両者の相互理解を進めることも重要な課題の1つである[16]．

組込みシステム技術は，日本の基幹産業を支える重要な基盤技術となりつつあるにもかかわらず，特にそのソフトウェア技術に関しては，大学や公的研究機関において取り組んでいる研究者はきわめて少ない[17]．本論文が，組込みシステム分野の研究・開発活動の活性化の一助となれば幸いである．

**謝辞** 本論文の議論は，組込みシステム技術に関するサマーワークショップ（SWEST）[18]およびソフトウェア技術者協会（SEA）名古屋支部のリアルタイムシステムに関するメーリングリストにおける議論に示唆を得た部分が多い．これらの議論への参加者に感謝します．また，本論文を発表する機会を与えていただき，また論文に対して貴重なご意見をいただいた本特集号ゲストエディタの若林真一先生に感謝します．

## 参考文献

1) 中本幸一，高田広章，田丸喜一郎：組込みシステム技術の現状と動向，情報処理，Vol.38, No.10, pp.871–878 (1997).
2) Stankovic, J.A.: Misconceptions About Real-Time Computing, *IEEE Comput.*, Vol.21, No.10, pp.10–19 (1988).
3) 宿口雅弘：組込みシステムのデバッグ手法，情報処理，Vol.38, No.10, pp.886–891 (1997).
4) 青山幹雄：組込みソフトウェアの転機，情報処理，Vol.39, No.7, pp.689–692 (1998).
5) トロン協会 ITRON 部会：リアルタイム OS の利用動向と ITRON に関するアンケート調査．http://www.itron.gr.jp/survey-j.html.
6) 今井正治，松永裕介（編）：特集「ハードウェア/ソフトウェア・コデザイン」，情報処理，Vol.36, No.7, pp.604–632 (1995).
7) Booch, G., Rumbaugh, J. and Jacobson, I.: *The Unified Modeling Language User Guide*, Addison-Wesley (1999).
8) Gajski, D.D., Zhu, J., Dömer, R., Gerstlauer, A. and Zhao, S.: *Spec C: Specification Language and Design Methodology*, Kluwer Academic Publishers (2000).
9) Arnout, G.: SystemC Standard, *Proc. Asia and South Pacific Design Automation Conference 2000*, pp.573–577 (2000).
10) 伊藤真紀子，檜垣茂明，塩見彰睦，佐藤 淳，武内良典，今井正治：パイプライン・ハザードを考慮した合成可能なプロセッサの HDL 記述生成手法の提案，DA シンポジウム '99 論文集，pp.201–206 (1999).
11) Puig-Medina, M., Ezer, G. and Konas, P.: Verification of Configurable Processor Cores, *Proc. 37th Design Automation Conference*, pp.426–431 (2000).
12) Red Hat Inc.: *eCos User's Guide* (2000). http://sources.redhat.com/ecos/.
13) 高田広章：構成可能な RTOS と $\mu$ITRON 仕様におけるアプローチ，ソフトウェアシンポジウム 2000 論文集，pp.49–54 (2000).
14) Douglass, B.P.: *Real-Time UML – Developing*

*Efficient Objects for Embedded Systems*, 2nd edition, Addison-Wesley (1999).
15) Puschner, P. and Burns, A.: A Review of Worst-Case Execution-Time Analysis, *Real-Time Systems*, Vol.18, No.2/3, pp.115–127 (2000).
16) 藤懸英昭：ハード・ソフトの協調設計を考える—意外に深いハード屋とソフト屋の溝をどう埋めるか，*Design Wave Magazine*, pp.63–70 (2001).
17) 高田広章ほか：組込みソフトウェア分野における産学間のギャップを埋めよう，情報処理，Vol.40, No.5, pp.536–540 (1999).
18) 高野裕之，藤懸英昭，宿口雅弘：SWEST 報告—組み込みシステム技術に産学の協調を！，*bit*, Vol.32, No.2, pp.30–35 (2000).

(平成 13 年 1 月 31 日受付)
(平成 13 年 2 月 1 日採録)

**高田　広章**（正会員）

豊橋技術科学大学情報工学系講師．1988 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同学科の助手等を経て，1997 年 12 月より現職．リアルタイム OS，リアルタイムスケジューリング理論，組込みシステム開発技術等の研究に従事．ITRON 仕様の標準化活動に，中心的メンバとして参加．博士（理学）．IEEE，ACM，電子情報通信学会，日本ソフトウェア科学会各会員．