第1章

仕様検討やアーキテクチャ設計を重視する開発プロセスへシフト

―― なぜASIC開発にUMLやSystemCが必要なのか

宮原忠義，萩原 篤，守田直也

ここでは，ASIC開発やシステムLSI開発にUML（unified modeling language），SystemCを導入する必要性について議論する．歴史的に，LSI設計においてチップに付加価値を付ける工程は下流から上流へと移行している．かつてはレイアウト設計や論理回路設計が，現在はおもにRTL（register transfer level）の機能設計が付加価値の源泉となっている．今後は，アーキテクチャ設計，アルゴリズム設計，仕様設計がASIC設計者の重要なしごとになる．一方，組み込みソフトウェア開発との並行開発や協調作業も求められている．そのため，ASIC開発においても，ソフトウェア開発と同じように要求分析やテスト仕様設計が重要になる．こうした部分を強化していかないと，ASIC開発やシステムLSI開発の業務そのものを継続できなくなる可能性がある．（編集部）

筆者らが所属する部署では，コピーやプリンタ，ファクシミリなどの複合機に搭載する数百万ゲート規模の制御用ASICや画像処理用ASICを開発しています．ASIC開発におけるプロセス（設計工程）改革を目指した社内ワーキング・グループの結成をきっかけに，筆者らは現在，UMLやSystemCなどの導入・評価を行い，一部ではすでに活用しています．

UMLはオブジェクト指向のソフトウェア開発のために考案された表記法です（例えば，UML 2.0では12種類の図が定義されている）．UMLの規格に合わせて図面を作成し，それに沿ってプログラム（ソース・コード）を開発していきます．SystemCはC++をベースとするシステム・レベル言語です．クラスの機能を用いてC++を拡張し，ハードウェアの動作を記述しやすくしています．

社内ワーキング・グループのメンバは，電気・電子系出身のハードウェア技術者と情報系出身のソフトウェア技術者が半々で構成されており，ハードウェアとソフトウェアの文化や技術を融合しながら活動しています．

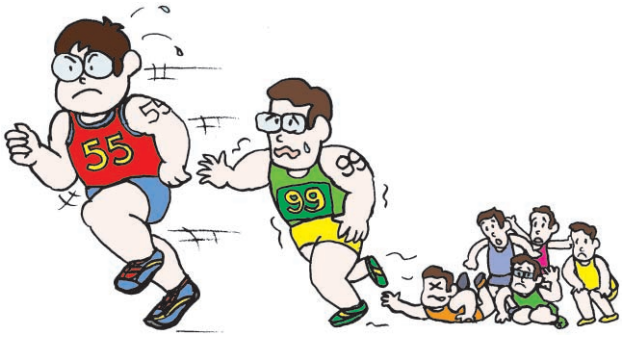

図1　ASIC開発は鉄人レース

タイミング収束，システム・レベル・モデリング，機能検証，低消費電力設計，製造容易化設計（design for manufacturability）など，ASIC開発における課題は年々増えている．新しく登場する手法や考えかたを自分の中で逐次消化していかないと，ASIC開発そのものを断念しなければならなくなるかもしれない．一度，遅れをとると，後から挽回することは難しい．

● 今傍観していて，果たして後から追いつけるのか？

ASIC設計の手段は，トランジスタ・レベル，回路図，HDL（ハードウェア記述言語）と10年単位で変わっています．現在は，C/C++をベースとした言語が注目を集めています．ほんとうにC/C++全盛の時代がくるのかどうかわかりませんが，徐々にSystemCなどが広がり，EDA（electronic design automation）ツールが整備されてきているのは事実です．読者のみなさんの中にも，時代の変化を予感されている方がいらっしゃるかもしれません．

筆者らの間でも，「設計言語はまだ変わらない」，「もっと広まってから活動しても遅くない」といった声が上がっています．その一方で，最新プロセス（半導体製造技術）やLSI開発手法の変化は速く，「今傍観していて，果たして後

から追いつけるのか？」という不安を抱えています（図1）．この不安を払拭するためにも，筆者らは新しい手法を試行して，良いものはどんどん吸収していくことに努めています．

● RTL設計者が要らなくなる!?

あなたが機器メーカのASIC設計者ならば，ASIC開発の5年後，10年後を考えてみてください．果たしてRTL設計の工程を自社で持ち続けていると思いますか？ 仕様を渡して，RTL設計はASICベンダや設計委託会社にお任せ，となっている可能性があることは否定できないと思います．

また，ビヘイビア合成技術が進歩すれば，C言語で書かれたアルゴリズム記述をわざわざVerilog HDLやVHDLに変換する必要がなくなるかもしれません．「将来，RTL設計者は要らなくなるのでは…」，筆者らはこんな危機感を感じ始めています．このような状況では，自社に残すものは適切な機能仕様やアーキテクチャ仕様を模索し，決定する業務，いわゆる上流設計のみになると思われます．少なくとも，仕様やアルゴリズム，アーキテクチャなどを取り扱う工程の比重が高まることはまちがいないでしょう．

LSI設計の技術革新は急速に進んでいます．そのため，ASIC設計者はつねに開発フローの将来像をイメージしておく必要があると考えます．例えば筆者らの場合，オブジェクト指向モデリングやUMLの表記法，SystemCを採用した場合の開発フローを図2のようにイメージしています．前述の上流設計は，要求（項目）からアーキテクチャ設計までの工程に相当します．要求分析の工程では，UMLを用いてモデリングを行い，モデルを洗練していくことを思い描いています．

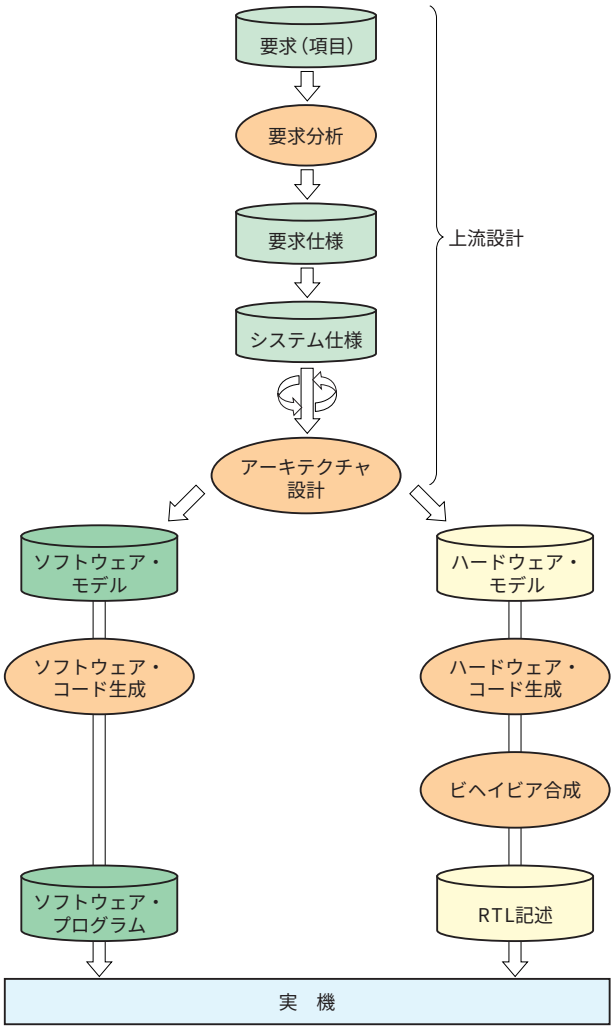

図2　将来の設計フローの例

オブジェクト指向モデリング，UML，SystemCを意識した設計フロー．システム仕様はおもに機能決定と機能分割を，アーキテクチャ設計はハードウェア・ソフトウェア分割，CPU決定などをイメージしている．本設計フローはあくまでも一つの例である．

● UMLは難しいのか？

社内の設計技術者とUMLについて話していると，「UMLは難しい」という答えがよく返ってきます．しかしUMLは単なる表記法であり，UMLを書くことそのものはそれほど難しいはずがありません．おそらく，ほんとうに敷居が高いのは，UMLではなく，要求分析やモデリングの作業なのだと思います（図3）．

この敷居を低くするために筆者らが取り組んできたことは，まずは「クリティカル・シンキング」です．新聞の社説を読み，これをフリー・フォーマットの図で表すことを実施しました．なぜ，これが要求分析に生きてくるのかは後で述べます．

図3　UMLは難しくない

UMLは表記法であり，深追いしなければ習得は難しくない．これを使った要求分析やモデリングが難しい．習得にも時間がかかる．