重点企画 システム・レベル設計言語入門


# 無償配布ツールを使ってSystemCを体験しよう！

Dundar Dumlugol, Pete Hardee, Abhijit Ghosh, Martin Wang

**システム・レベル設計言語の一つである「SystemC」の概要を紹介する．SystemCはC/C++ベースの言語である．ハードウェアのモデリングに必要となる機能（コンカレント処理，クロック，4値論理，固定小数点型など）をクラス・ライブラリとして提供する．ソフトウェア・プログラムとハードウェア・モデルをリンクして実行できるので，システム・オン・チップを開発する際のハードウェア・ソフトウェア協調設計に有効であるという．SystemCのシミュレーション環境は，無償で入手できる．ここではSystemCの特徴，SystemCの文法概要について解説した後，無償で手に入るツールとサンプル記述を用いたチュートリアルを紹介する．必要となるサンプル記述は，Design Wave Magazineのホームページhttp://www.cqpub.co.jp/dwm/からダウンロードできる．また，7月号の本誌付属CD-ROMにも収録する予定である．（編集部）**

この記事の目的は，SystemC言語とモデリング・プラットホーム（モデリングのための環境）を読者のみなさんに紹介することです．この記事に目をとおすだけで終わってはいけません．ここで紹介するSystemCのコーディングやサンプル記述の実行に，ぜひ挑戦してみてください．そして，あなた自身の手で独自のSystemCモデルを作成してみましょう．

もしあなたが，すでにSystemCの環境をダウンロードした数千人のうちの1人でないのなら，http://www.systemc.org/にアクセスしてください．そこで，あなたはSystemCの環境を無償で手に入れることができます．あなたがC++のエキスパートである必要はありません．SystemCのクラス・ライブラリには，あなたが必要とするほとんどのC++コードが用意されています．そのほかに必要となるものについては，この記事のなかで説明します．なお，ここでは，あなたがC言語に対するある程度の知識をもっており，Verilog-HDLやVHDLなどのハードウェア記述言語に慣れ親しんでいることを前提に，話を進めます．

いったんSystemCの環境をダウンロードすれば，あなたはもう立派なOpen SystemC Initiative（SystemC言語の普及推進団体）の一員です．なにか問題があったときや詳しい情報が必要なときには，http://www.systemc.org/にアクセスし，オンラインのフォーラム（会議）に参加できます．ここでは，ほかのメンバに支援を求めたり，あなたの意見を他のメンバに伝えることができます．さらに，ほかのメンバが直面している問題を知ることもできます．SystemCの初心者を卒業したら，今度はあなたがほかのメンバを助けてあげましょう．

SystemCのコミュニティへようこそ！

## 1．なぜSystemCが必要なのか？

システム・オン・チップ（SOC），システム・チップ，システム・レベル・インテグレーション（SLI）…．さまざまな呼び方があります．あなたがそれをどのように呼んでいようともかまいません．それはいまここにあり，そして使われています．

こうしたチップは，IPコア（少なくとも1個のマイクロプロセッサ，複数のバス，メモリ）とカスタムASICで構成されています．さらに，こうしたチップにはソフトウェアが統合されていなければなりません．そして，そのソフトウェアはだんだん大きく，複雑になってきています．

このようなチップは，世界でもっとも著名で，かつ最大の売り上げ実績を誇っている民生用電子機器のなかでよく見つかります．消費者の要求レベルは非常に高く，日本の設計技術者のみなさんは，これまで以上に複雑な電子機器をこれまで以上に短い期間で市場に送り出さなければならない，というプレッシャが存在することを十分に認識されていることと思います．

### ●システム・オン・チップの開発ならSystemC

IPコア，カスタムASIC，ソフトウェアを開発し，統合する過程では，検証作業が大きな問題になります．ここで，十分に抽象度の高いIPコアのモデルが存在しなければ，検証速度は非常に遅いものになります．同じことは，カスタムASICの設計についても言えます．もっともよい方法は，これらの

モデルを，ソフトウェアと同じ言語で作成することです．こうすれば，最大の効率でハードウェアとソフトウェアを一体化できます．

システム・レベルのIPコア，カスタムASIC，ソフトウェアを含むシステム・レベルの設計データを，一つの言語で完全に記述する標準的な方法はないのでしょうか．あります．それこそがSystemCなのです．

ニーズは非常に明確なのですが，システム・レベル言語について，現在，かなりの混乱が見受けられます．それは，「システム・レベル」という言葉がなにを指すのか，そしてシステム・レベル言語を一つに絞り込むことが本当に可能なのか，という点での混乱です．

記述の抽象度が上がるにつれて，設計言語が分化し，特定のドメイン（応用）が対象となっていくのはたしかです．さらにソフトウェア開発者にとってC/C++は，設計の出発点ではなく，インプリメンテーション（設計の最終段階）の道具であるということも認識しています．ですから，わたしたちはここで，SystemCが「すべての人にとって最適な言語である」と主張する気はありません．

しかし，システム・オン・チップを開発するために，システム・レベルのIPコア，カスタムASIC，ソフトウェアを統合する局面では，C/C++ベースの設計言語こそ成功を収める唯一の言語であると，わたしたちは信じています．

### ●ハードウェアをモデリングするためにC/C++を拡張

システム・レベルのIPコア，カスタムASIC，ソフトウェアのすべてのモデリングに対応するためには，C/C++の言語仕様を拡張する必要があります．SystemCのバージョン1.1には，システム・レベルのIPコアとカスタムASIC（ハードウェア）の記述に必要となるすべての拡張機能が備わっています．拡張機能は，クラスを利用して純粋なC++で作成されています．ここでクラスとは，ユーザ定義のデータ型のことです．

わたしたちはこれらのクラスを一つのライブラリのなかに収めました．さらに，あなたが作成したコードを正しく実行できる軽快なシミュレーション・カーネルを用意しています．これらのC++ソース・コードのすべてが，わたしたちが供給するSystemCの環境に組み込まれています．SystemCを利用する際に必要となるのは，C++コンパイラだけです．

現在，提唱されているほかのいくつかのソリューション（システム・レベル言語）と異なり，SystemCの環境ではプリプロセッサ（前処理用の変換ツール）は使いません．つまり，SystemCは純粋なC++であり，またC++はCを包含しています．したがって，ANSI CとANSI C++の記述をそのまま実行できます．そのため，ソフトウェア開発者はなんの問題もなくSystemsCのシステム・モデルを利用できるのです．

さらに，C/C++のソフトウェアとSystemCのシステム・モデルをソース・コードのレベルで，いっしょにデバッグすることも可能です．プリプロセッサ・ベースのソリューションには，このようなソース・レベルのデバッグを実現しにくいという欠点があります．

このほか，SystemCのロードマップでは，リアルタイム動作をともなう組み込みソフトウェア（タスク，時間制約，リアルタイムOS）のための機能拡張が予定されています．この点で，SystemCは，C++による「Verilog-HDLのリプレース」などというレベルをはるかに超えた言語なのです．

## 2．SystemCによる設計フロー

次に，従来の設計フローとSystemCの設計フローの違いについて，説明します．

### ●アーキテクト，ASIC設計者，ソフト開発者の間の壁

HDLをベースとした従来型のシステム設計フローでは，システム・アーキテクトとASIC設計者が別々に作業を進める傾向があります．

システム・アーキテクトは，システムのコンセプトを考えたり，システムの一部，あるいはシステム全体をモデリングし，C/C++によるシミュレーションを行いながら，最適なアーキテクチャを模索します．設計仕様をインプリメンテーション・チームに引き渡す段階では，昔ながらの紙に書かれた仕様書が利用されています．ASIC設計チームは仕様書に書かれた内容を理解し，システムのモデルをビヘイビア・レベルのHDLで記述し直します．ハードウェアで実現する部分については，さらにRTL（register transfer level）のHDLで記述し直し，検証し，合成し，さらに再検証します．

しかし，紙に書かれた仕様書は一般に完全ではなく，しかも整合性がとれていないこともあります．さらに，HDLを使って記述し直す過程でミスが発生することがあります．システム・アーキテクトとASIC設計者が共通に利用できる実行可能な言語が存在していないため，実際，両者の間には壁が存在しています．その結果，開発期間が伸びたり，経費のかさむ設計の繰り返し作業が発生したり，当初の目標と合わない製品ができあがることになります．

さらに困ったことに，システム・オン・チップの設計の場合，ASICチームとソフトウェア・チームの間にも，巨大な壁が存在します．ソフトウェアとハードウェアは，ASICのプロトタイプ（試作チップ）ができあがるまで統合されません．もっとも条件のよい場合でも，せいぜいRTL設計が完了した