SystemCによるLSI設計検証技術

内海　功朗

近年，システムLSIの設計はLSIの規模が拡大するにつれて設計と検証の工数が増大し，ハードウェア記述言語によるモデル設計と検証は限界に近づいている。その限界を打開するための設計手段としてC言語による設計が注目されている。

本稿では，LSI設計におけるSystemC*1) 導入の背景から，SystemCの特長および我々が実施したSystemCを用いたLSIの設計と検証手法について述べる。

Verilog-HDL on WS設計から SystemC on PC-Linux設計へ

1980年代前半から，ハードウェア設計が回路記述からVerilog*2) -HDL（Hardware Description Language）等のハードウェア記述言語によるモデル記述へ移り変わった。この時から汎用WS上のEDA（Electronic Design Automation）ツールを用いて①仕様設計，②RTL設計，③RTL検証，④論理合成，⑤レイアウトという設計フローに基づいたLSI設計がなされるようになった。近年，システムLSIの大規模化が進み，一昔前ではボードで構成されていたCPU＋メモリ＋周辺回路の構成が1チップ上に搭載されるようになった。このような状況に対応するためLSI開発工程においてエミュレータやプロトタイプボードを利用したハードウェアとソフトウェアの協調検証を実施することで，開発期間短縮化と高品質化が図られてきた。

しかし，市場からさらに多機能かつ性能の高いシステムLSIが要求されるようになると，**図1**のようにハードウェアとソフトウェアを統合したFPGA検証の段階で，①期待した性能が出ない，②ソフトウェア設計者とハードウェア設計者間で仕様解釈の食い違いが露呈する，③実際のシステム動作で不具合が発覚する等により，手戻りや開発期間の延長がしばしば見受けられるようになった。従来の設計フローでは，もはや顧客が求める短納期化に対応できなくなってきている。

このような状況を打開するため各社でC言語によるシステムLSIの設計が試行されている。以前までは数種の独自仕様のC言語ツールが提案されてきたがSystemCが事実上，業界標準に定まった[1)]。SystemCをLSI設計フローの初期段階で利用することで，①性能評価，②ハードウェアとソフトウェア仕様の早期整合，③実チップの動作に近い環境での設計と検証が可能になる，といったことにより潜在的な問題が設計フローの初期段階で解決される。

一方，PC-Linuxの普及により従来のWSの性能に制限されていたツールの性能がPC-Linuxへ移行することにより数倍向上することになった。また，PC-Linuxはオープンソースであり，SystemCからPC-LinuxのOS機能や多くのソフトウェアライブラリが利用できるので，より実際に近いシステム検証の環境を整えることが可能となる。

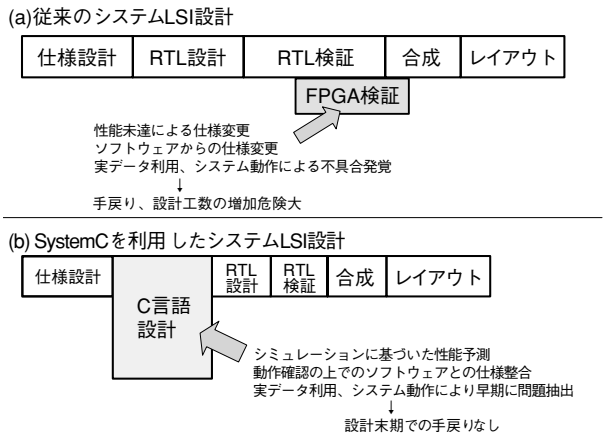

図1　SystemC導入によるLSI設計フロー

設計共通言語としてのSystemC

まず，SystemCの特長について簡単に述べる。

従来，ハードウェア設計者はハードウェア記述言語であるVerilog-HDLなどを利用し，アルゴリズム設計者やソフトウェア設計者は主にC言語を利用してきた。ハードウェア記述言語では主にレジスタの並列動作を記述するため，回路動作に慣れていないソフトウェア設計者には理解し難かった。一方，ソフトウェアのC言語ではタスクのシステムコールによる並行動作を記述しているため，シ

＊1）SystemCはOpen SystemC Initiativeの登録商標です。　＊2）VerilogはCadence Design Systemsの登録商標です。

ステムコールの動作を理解していないハードウェア設計者には理解が困難であった。SystemCはC++を拡張した言語であり，モジュール内のプロセスの並行動作を記述するため，双方の専門知識が要求されずハードウェア設計者にもソフトウェア設計者にも理解し易いものになっている。

また，ハードウェア，ソフトウェア共に制御記述のスタイルを状態遷移表（状態遷移図）に基づいた記述に従うことで，ハードウェアにもソフトウェアにも対応できる統一した記述が可能になる。

今後，SystemCは**表1**にあるようにソフトウェア／OSレベルを含んだ機能まで拡張される予定である。この拡張によってSystemCでOSの基本的なシステムコールを含んだ記述が可能になり，システム全体をハードウェア設計者とソフトウェア設計者が共通の理解のもと一元的に記述することが可能になる。

これによりソフトウェア設計者はソフトウェアのデバッグ段階で従来ブラックボックスになっていたハードウェア記述部分での問題点も見つけられるようになる。また，ハードウェア設計者はデバイスドライバの動作を理解し，適切なハードウェア仕様をインプリメントできるようになるため，LSI設計効率が加速度的に向上することが期待される。

表1　SystemCの記述レベルの変移

年	バージョン	記述レベル
2000	1.01	RTレベル
2003	2.01	トランザクションレベル
2005～	3	ソフトウェア／OSレベル

SystemC on PC-Linuxの特長を生かしたシステムLSIの設計と検証

システムLSIはその用途によって様々な構成のものが設計される。我々は対象とするシステムLSIの用途に応じて機能を拡張できるように市販のEDAツールをベースに，PC-Linuxの機能を組み合わせた設計検証環境の構築を検討している。以降，これらの技術について紹介する。

(1) ソフトウェア・ハードウェア協調シミュレーション

SystemCで記述した周辺回路モデルとCPUの命令セットシミュレータを同時に動作させる協調シミュレーションではPC-Linux上でソフトウェアがデバッグ可能なレベルにまで性能を上げつつある。**図2**にSynopsys社CoCentric System　Studio*3) 上に構築したARM946*4) と周辺回路を装備したSystemCモデルの例を示す。図中のモジュールはAHB-CLI（Advanced High performance Bus Cycle Level Interface）*5) のトランザクションレベルのポートとインタフェースで接続されている。モジュール間の情報は信号線ごとに伝達されるのではなく，一方のモジュールがポートからインタフェースを通してチャネルが提供するメソッドを直接呼び出すことで他方のモジュールへ伝達される[2]。この機能により，データの移動が少なく，高速なシミュレーションを可能にしている。ARM946のCモデルはサイクルごとに厳密に動作し，ハードウェアの動作と連携してシミュレーションするのでソフトウェアの検証ができる。現在はまだ50KHz程度のシミュレーション速度であるが，今後PC-Linuxおよびツール双方の性能向上で100KHzレベルにまで到達することが見込まれ，ソフトウェアがよりデバッグし易くなる。

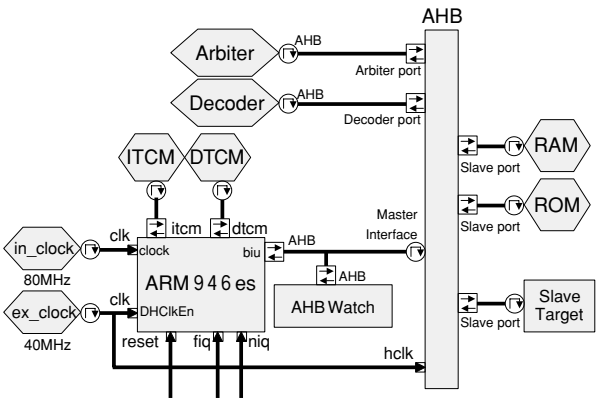

図2　ARM946と周辺回路を接続したSystemCモデル

(2) マルチプロセス検証環境

従来の単体RTLモデルの検証環境を**図3（a）**に示す。入力ファイルを用意し，そのパターンを検証モデルに入力させ，その結果をファイルにライトして期待値と比較するという手順で検証を行ってきた。このような方法は小規模モデルや定型化したデータしか処理しない場合には問題とならなかった。しかし，音声や画像等多量のデータを扱うLSIを検証する場合には，①出力の期待値の作成が困難で，波形や画像の表示による目視確認に限定される，②多量のデータを入力させなければ動作確認ができないといった課題が発生している。また，検証する段階に応じて検証環境モデルを拡張しなければならないため，検証環境の複雑化が問題になってきている。

SystemCはC++を拡張した言語であるので，通常のC言語と同じようにシステムコール等のシステムプログ

*3) CoCentric System StudioはSynopsys社の登録商標です。　*4) ARM946はARM Ltd.の登録商標です。　*5) AHB，AHB-CLIはARM Ltd.の登録商標です。

ラムが作成できる。SystemCプロセス上のプログラムからシステムコールができるので，マルチプロセスの検証環境が可能になる。**図3（b）**に示すようにモデルのシミュレーション段階で外部環境からの実データをリアルタイムでアクセスするプロセスと，シミュレーション後のデータを表示するプロセスを用意することで，実システムに近いリアルタイムの動作を設計・検証することができる。別プロセスをシステムプログラムの共有メモリやメッセージ通信を利用して，外部からの情報を入出力するので，検証環境を変えずに前処理，後処理を変更することができる。この方法は過重になってきた入出力の検証環境を独立のプロセスと見なせるのでブラックボックス化，部品化の促進にもなる。

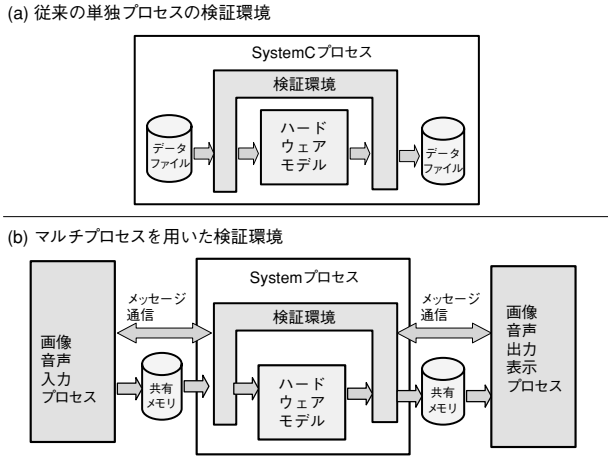

図3　マルチプロセスの検証環境

(3) FPGAアクセラレーション

既存のVerilog-HDLモデルや抽象度を低くしたRTレベルのモデルと他のトランザクションモデルを混在して検証する場合，Verilog-HDLモデルやRTレベルのシミュレーション速度に制限されて極端にシミュレーション性能が落ちる。

FPGAアクセラレーションはVerilog-HDLモデルやRTレベルのSystemCモデルをFPGAにマッピングして実クロックで動作させ，SystemCと協調シミュレーションすることでシミュレーションの高速化を図ることをいう。今回，我々が取ったアプローチは特別なボードを開発せずに市販のFPGA搭載PCI*6) ボードを用意し，デバイスドライバを組合せて実現する方法を採用した。従来のエミュレータに見られるような固定したデバイスドライバ制御を取らず，種々のデバイスに対応したデバイスドライバ構成を取ることで柔軟性および高速性を強化した。PCIボード上のモデルはPCIバス経由でPC上のSystemCプロセスと通信を取りながらシミュレーションされる。このようにシステムLSIに搭載する機能を，機能ごとにPCIボードのFPGAに割付けることで，多様なシステムLSIに対応できる検証環境の構築が可能になる。

従来，PCに装着した独自のボードを制御するデバイスドライバを開発することは困難であったが，PC-Linuxの場合は各種の情報が公開されており[3]，ソフトウェアプログラムの雛形が用意されれば，ハードウェア技術者でも容易にハードウェアに対応したデバイスドライバが開発できる。

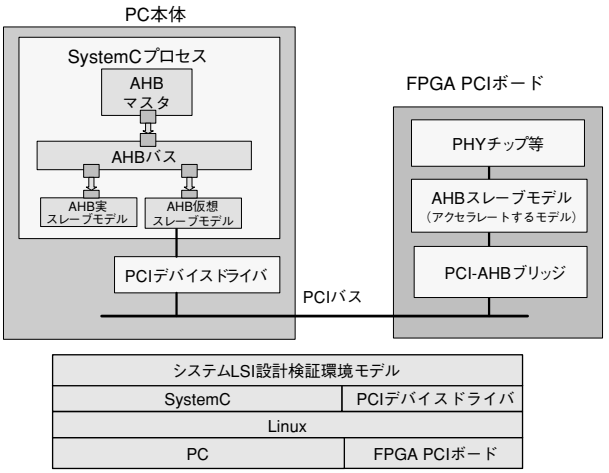

図4　FPGAアクセラレーションの基本構成

① 基本構成

FPGAアクセラレーションのハードウェアとソフトウェアの基本構成を**図4**に示す。**図4**では，アクセラレートするモデルはFPGAに置かれる。FPGA搭載PCIボードでは，モデル本体の他にモデルのインタフェースであるAHBとPCI間のブリッジ回路が置かれる。PC上で動作するSystemCプロセス上ではAHBスレーブモデルの代わりにデバイスドライバを呼び出す仮想スレーブモデルが置かれる。AHBマスタがAHB仮想スレーブモデルのレジスタをアクセスすると，AHB仮想スレーブモデルはレジスタアクセスに対応したPCIデバイスドライバを呼び出す。

② 割込み処理

一般にデバイスは起動レジスタにライトされて動作を開始し，割り込みをアサートすることで動作の終了をCPUに通知する仕組みになっている。一方，SystemCのトランザクションモデルでは非同期の処理としてイベントを利用することが多い。ここでは，以下の手順でFPGAからの割り込みをSystemCへのイベントに変換している。

*6) PCIはPCI-SIGの登録商標です。

1. デバイスの割込み発生
2. PCI割込みへ通知
3. デバイスドライバ内で割込みハンドラを起動
4. 割込みハンドラからシグナル呼び出し
5. SystemCプロセスでのシグナルを受信
6. シグナル起動関数でデバイスドライバから割込み情報の取り出し
7. その割込み情報に従ったイベントの生成

③ アクセスの高速化

AHB仮想スレーブモデルへアクセスする度にデバイスドライバを呼び出すと，頻繁にPC-Linux上のプロセスが切り替わるため，そのオーバーヘッドがシミュレーション速度の低下につながる。より高速化するためには，デバイスドライバの呼び出し回数を削減するようにすれば良い。そこで，起動／停止レジスタがアクセスされるまではデバイスドライバを呼び出さず，FIFO（First In First Out）に蓄積しておき，起動／停止レジスへのアクセスがあった場合に，デバイスドライバ上で連続してFPGA搭載PCIボードのAHBスレーブモデルにアクセスする。デバイスドライバの呼び出し回数を最低限に控える手法を取ることでSystemCのシミュレーション速度低下を抑える。

SystemC設計の現状の課題

実際にSystemCを利用した場合の主要な課題を2点述べる。

（1）SystemCモデルの検証

図1（b）のように初期段階でC言語設計を実施する場合，Verilog-HDLのRTLモデルと比較してライン数は1/3～1/4に削減され，共通トランザクションインタフェースを用いることで信号レベルの検証問題は解消されるが，プロセス内部やプロセス間の動作確認は残る。この検証課題に関してはSCV（SystemC Verification Library）[*7)]をどのように利用するかが期間短縮の鍵になると考えられる。

（2）デバッグにおける課題

Verilog-HDLによるモデルのデバッグでは，シミュレーションによる波形表示によりデバッグを行っていた。しかし，SystemCで書かれたトランザクションレベルでは，いつモジュール内のプロセスが動作したかを理解する必要があり，波形表示と共にデバッガを連動させてモデルの動作を追いながら検証する必要がある。従来のデバッガではクラスライブラリの中で必要とする情報のみを表示させることが困難なこと，ステップ実行した場合，シミュレータのカーネルやライブラリまでトレースされてしまうこと等デバッグ効率上の課題が多い。EDAベンダから効果的なデバッガの提案が望まれる。

最　後　に

SystemCを用いた設計と検証に関して，C言語導入の背景と我々が実施した市販のEDAツールをベースにして用途に応じて拡張できるPC-Linuxの機能を組み合わせた設計検証環境の構築技術について紹介した。

今後は，大規模なシステムLSIの設計検証に十分対応できるよう以下の2点に関してさらに検討を加えていく所存である。

①μPLAT[*8)] のC言語インタフェースに対応したFPGAアクセラレータの検討および実チップ規模での評価。

②SCVを用いた効果的なSystemCモデルの検証。 ◆◆

■参考文献

1）大石基之：ハードーソフト協調検証　短期開発SoCの必需品へ，日経エレクトロニクス，2004.12.16，p.65-72，2004年

2）Thorseten Grotker他：SystemCによるシステム設計，1版，丸善株式会社，p.11-41，2003年

3）Alessandro Bubini他：Linuxデバイスドライバ，2版，オライリージャパン，2002年

●筆者紹介

内海功朗：Isao Utsumi. 株式会社沖ネットワークエルエスアイ SiSC開発支援本部 カスタムデザインチーム

*7）SCVはOpen SystemC Initiativeの登録商標です。　*8）μPLATは沖電気工業（株）の登録商標です。