ハード／ソフト協調シミュレーションによるシステム検証

System Verification through Hardware/Software Co-Simulation

中　野　邦　夫＊
Nakano, Kunio

Generally, SOC (system on a chip) integrates CPU, memory, IPs (Intellectual Property), and user designed logic. Therefore, We must verify a SOC on both the hardware and software. In a traditional way, this verification have been done at software debugging after hardware debugging. So, the verification needs a long time to complete.

Co-simulation approaches to verify software and hardware concurrently.

This report describes the outline of Co-simulation and the assignments by a case study.

1　はじめに

一般的にSOC（System On a Chip）はCPU、メモリ、IP（Intellectual Property）とユーザ設計の論理回路を半導体チップ上に搭載する。このため、SOCの検証ではハードウェアとソフトウェアの両面から検証を行わねばならない。　従来この検証はハードウェア、ソフトウェアの順に行われていたため、非常に長い期間を必要としていた。　ハード／ソフト協調シミュレーション（Co-Simulation、以降Co-Sim.と略す）はこの２つの検証を同時に行おうとするものである。

本報告ではCo-Sim.の概要とケーススタディから得られた課題について述べる。

2　Co-Sim.の概要

Co-Sim.はワークステーション上に仮想的にSOCを構築し、このSOCに搭載している仮想CPU上でソフトウェアを動作させることで実機と同じ状態を具現化し、SOCの実チップを作成する前にソフトウェアとハードウェアの検証を行おうとするものである。この両面からの検証を行うための環境はISS（Instruction Set Simulator)、LS（Logic Simulator)、Co-Sim.マネージャの３つから構成される。(Fig.２.１参照)

①ISS（Instruction Set Simulator）

Co-Sim.におけるCPUの実体。ソフトウェアをワークステーション上で仮想的に実行する。ＧＵＩ（Graphic User Interface）としてはソースコードウィンド、メモリウィンド、レジスタウィンドなどがある。

②LS（Logic Simulator）

ハードウェア（CPUのハード的な動作を含む）の動作をワークステーション上で仮想的に実行する。GUIとしては制御ウィンド、波形表示ウィンドなどがある。

③Co-Sim. マネージャ

ISS上で実行されるソフトウェアによるCPU動作とLS上で実行されるハードウェアの動作をクロックレベルで同期をとる。

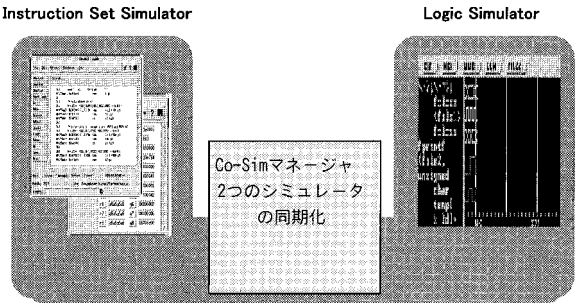

Fig.２.１　Co-Sim.の構成

3　Co-Sim.ケーススタディ

Co-Sim.のケーススタディとして200万画素デジカメのメイン回路部分を対象とした。

回路の構成をFig.３.１、ソフトウェアの処理フローをFig.３.２に示す。

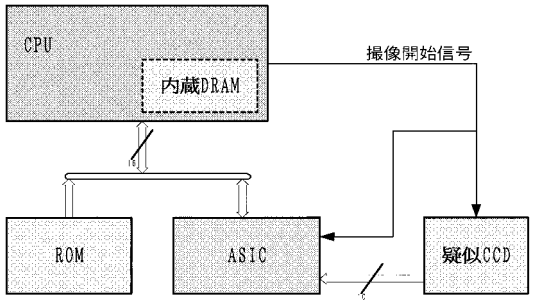

Fig.３.１　ケーススタディの回路

＊コーポレートラボラトリーG　中央研究所

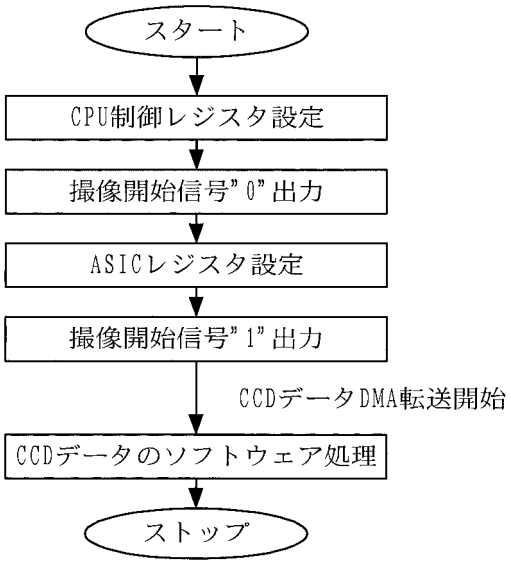

Fig.3.2　ソフトウェアの処理フロー

4　評価結果

ケーススタディから得られたCo-Sim.の評価結果を以下に示す。

①実機使用のソフトウェア実行オブジェクトを使用し、LS用テストベンチなしでソフトウェア／ハードウェアの両面から検証が可能である。

②ISSのメモリウィンド機能を使用して、CPUにメモリアクセスやASICのレジスタアクセスを実行させることができる。このためCPU用プログラムなしでもCPUアクセスについては単独でハードデバッグが可能である。

③LSがISSと同期して動作するCo-Sim.では、LSのシミュレーション結果から処理時間の見積りを正確に行うことができる。

5　課題

ケーススタディからの課題を以下に示す。

①高速なLSとそれに対応したCPUモデルが必要。
ISSに比べLSのシミュレーション速度が遅いため、Co-Sim.のシミュレーション速度はLSのシミュレーション速度に依存してしまう。このためCo-Sim.マネージャは高速なLSに対応している必要がある。またその高速なLSとそれに対応したCPUモデルが必要となる。

②pull-up、pull-downの処理（使用上の注意点）
CPUバスなどpull-up（pull-down）がされている場合、結線する線自体をアクティブロー（アクティブハイ）とする必要がある。（通常の接続ではドライブする値が衝突し、"X"になるため）具体的にはVerilog-HDLでは回路記述上で結線そのものをWAND(WOR)宣言する必要がある。

③CPUモデルの精度
CPUモデルの動作は当然ながら極力実チップと同じであることが望ましい。
CPU動作が実チップとかけ離れてしまうと本来目的とする検証が不可能となってしまう。

④クロスツールの一貫性
実機で使用するクロスツールとCo-Sim.で使用するクロスツールが異なってしまう場合、セットアップファイルなどのアセンブラコードを書き直さなければならなくなる。また、実行オブジェクトも再作成しなければならない。Co-Sim.用にコードを書き直したり、再コンパイルしていては非効率的であるし、バグが混入する恐れがある。Co-Sim.の実使用では実機で使用されるクロスツールが使用可能でなければならない。

⑤Co-Sim.使用者に必要な知識
Co-Sim.はハードウェア、ソフトウェアの境界部分を扱う。従って、ハードウェア、ソフトウェアの両方の知識が必要とされる。特に、CPUのセットアップファイルやリンカスクリプトについては熟知する必要がある。

6　まとめ

ケーススタディを通して、下記に示すCo-Sim.の効果、および注意点、課題を抽出できた。
また、今後のSOC開発には非常に有効なツールであることを確認した。
組込型CPUを搭載したシステムではハードウェアとソフトウェアの両分野、特に境界部分となるリンク、コンパイルについては熟知する必要性がある。

効果：

①従来はハードウェア開発を完了（SOCサンプル試作）し、サンプルを入手してからでないとソフトウェアを含めたシステム全体の検証はできなかった。Co-Sim.によって、ハードウェア、ソフトウェア双方のデバッグがワークステーション上の仮想的なSOCを用いて可能となり、SOCサンプルを作成することなく、システム全体の検証が可能になった。

②SOCサンプルを作成する前に実機に近いレベルでの検証ができるため、SOCの試作回数の削減とともに、開発期間の短縮が可能になった。