ハードウェア・ソフトウェア　コ・シミュレーション技術

Hardware-Software Co-Simulation

鈴木　政光*　　中村　勝*　　如澤　俊明*
Masamitsu SUZUKI　Masaru NAKAMURA　Toshiaki JOZAWA

要　　旨

機器あるいはチップの開発における新しい開発手法としてハードウェア・ソフトウェア　コ・シミュレーション技術を紹介する．プリンタなどの組み込み機器は年々高機能かつ複雑なシステムとなっており，半導体の集積度の向上によってシステムLSIなど大規模チップ開発にも対応が必要になった．ハードウェア・ソフトウェア　コ・シミュレーション技術の概略と，当社MFP機のパネル操作部を対象としたシミュレーション結果について報告する．

ABSTRACT

A new method for Hardware-software co-simulation for developing systems and chips are reported. Functions of　printers and other embedded systems are growing increasingly complex, and huge chips, called "System LSIs," are being developed by increasing the integration of gates in silicon. The introduction of　hardware-software co-simulation and the results of simulation on the Ricoh MFP operation panel are described.

* 画像システム事業本部　ソフトウェア研究所
Software Research Center, Imaging System Business Group

1. 背景と目的

プリンタ，MFPなどの組み込み機器は年々高機能かつ複雑なシステムとなっている．また，近年半導体の集積度が向上したことにより，マイクロプロセッサ，周辺論理回路，メモリ，各種IP(Intellectual Property)などを1チップ化したシステムLSIあるいはSOC(System On a Chip)と呼ばれる大規模チップ開発が行われるようになってきた．こうしたシステムやチップの設計・開発および検証に費やす工数は非常に増大しているが，一方では，製品の市場投入時期が早まり開発期間の短縮が求められている．ハードウェアとソフトウェア開発を別々に行っていた従来方法ではこうした課題を解決できない．

そこで，ハードウェアとソフトウェアを並行設計・開発・検証することにより開発期間を短縮したり，システム・レベルでの設計・検証の精度を上げたりすることによってハードウェアの作り直しをなくすことを目的としたハードウェア・ソフトウェア　コ・シミュレーション技術が注目されている[1)]．

ハードウェア・ソフトウェア　コ・シミュレーションは，組み込み機器のハードウェアとソフトウェアをWS(Work Station)あるいはPC(Personal Computer)上の疑似環境下で同時にシミュレーションする手法である．特に，開発したハードウェアおよびソフトウェアの検証を目的としたコ・シミュレーションをハードウェア･ソフトウェア　コ･ベリフィケーション(以下協調検証と呼ぶ)と呼び，EDA(Electronic Design Automation)ベンダから各種ツールが製品化されている．また最近脚光を浴びているハードウェア・ソフトウェア　コ・デザイン(以下協調設計と呼ぶ)は，主にシステムLSIの設計手法として，システム・レベルでの設計・検証，ハードウェアとソフトウェアの最適分割と並行開発を狙っている．

本稿では，まず協調検証技術について解説する．次に，当社MFP機のパネル操作部を対象とした協調検証ツールの機能・性能評価を行った結果を示す．また協調設計の技術動向についても紹介する．

2. 技術

2-1　協調検証

協調検証は，組み込まれるマイクロプロセッサあるいはDSP(Digital Signal Processor)のシミュレーション・モデル(以下MPUモデルと呼ぶ)とASIC，周辺論理回路などのハードウェアのシミュレーション・モデルを用意し，ハードウェアとソフトウェアを協調させてシミュレーションを行う．一般的な協調検証ツールでは，HDL(Hardware Description Language)記述されたハードウェアを既存のHDLシミュレータで動作させ，ソフトウェアをMPUモデルと既存のソフトウェア・デバッガを組み合わせて動作させる(Fig.1)．通常MPUモデルには命令セット・シミュレータ(ISS:Instruction Set Simulator)が使用される．シミュレータ間の協調動作は，HDLシミュレータのPLI(Programming Language Interface)を使用し，HDLシミュレータのクロック動作にMPUモデルの実行サイクルを同期させることにより行っている．

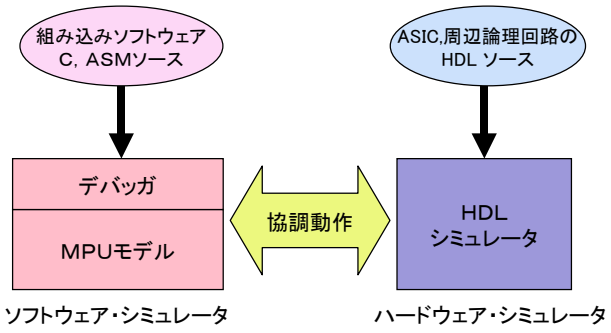

Fig.1　Hardware-Software Co-Verification

協調検証による開発フローをFig.2に示す．従来はハードウェア開発が完了し試作ボードが出来てからソフトウェアの検証を行っていたが，協調検証ではこれらを並行して進められるようになり，その結果，開発期間を短縮することができる．協調検証環境では，ASICなどのハードウェアの検証はテスト・パターンの代わりに実際に動作させるソフトウェアで行うことができるため，実機と同じ動作で検証を行うことができる．またハードウェア・モデルを仮想ハードウェアとして行うことができるため，試作ボードができあがる前にソフトウェアの検証を行うことができる．

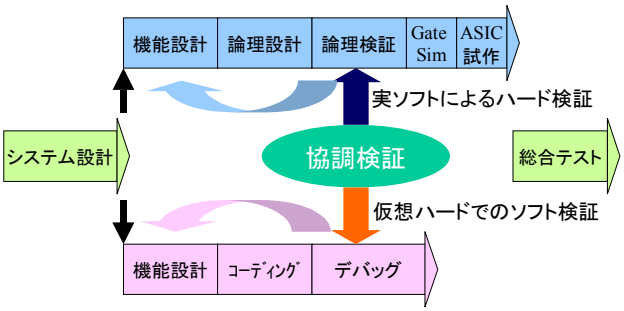

Fig.2　Development Process using Co-Verification

協調検証で用いるHDLシミュレータは，年々検証速度の向上が図られているものの，検証するASICのゲート規模の増大に追いつかない状況である．したがって，協調検証においてもHDLシミュレータの検証速度がボトルネックとなることが多い．また，検証速度はハードウェアのシミュレーション・モデルやMPUモデルの検証精度とトレードオフの関係にあるため，使用する目的に応じた検証精度を確保することが必要である．たとえば，ハードウェアの検証を目的とした場合は検証精度を優先し，ソフトウェアの検証を目的とした場合は検証速度を優先する．

MPUモデルの検証精度は，どれだけマイクロプロセッサの動作を精度よくシミュレーションできるか，ということになる．マイクロプロセッサをRTL記述しゲートレベルの精度でシミュレーションできるモデル，マイクロプロセッサのパイプライン動作などを模擬するサイクル精度のモデル，実行する命令精度のモデル(命令セット・シミュレータはこれに当たる)，シミュレーションを実行するマシンの命令セットに置き換えてしまうモデル(通常ネイティブ・モードと表現される)などがある．上述したこれらのモデルでは記述した順に精度が高いが，その分検証速度が遅くなる．

ハードウェア側もその記述の方法によって検証精度が異なる．RTL(Register Transfer Level)あるいはゲートレベル記述では，開発したリソースをそのまま使用してクロック精度の検証を行うことができる．ASIC以外の周辺回路モデルを作成する場合やクロック精度まで精度が必要ない場合は，HDLあるいはC言語によるビヘイビア・レベル記述を用いてモデルを作成する．ソフトウェアの検証では，ハードウェアの検証精度は必ずしもクロック精度まで必要としないため，ビヘイビア・レベルで記述し検証速度を速くすることができる．

ハードウェアとソフトウェアそれぞれにおいて，上述のような検証精度と検証速度を考慮し，目的にあったモデルを選択することが肝要である．

2-2　MFPパネル操作部への適応事例

協調検証ツールの機能・性能評価のため，当社MFP機に搭載されるパネル操作部を対象として協調検証を行った．ハードウェアのモデル化にあたっては，検証精度の異なるモデルを用意しそれぞれのモデルによる検証速度の比較を行った．

今回使用した協調検証ツールは，ガイオ・テクノロジー社のG-SIMplusである．このツールでは，ハードウェアはケイデンス社のNC-Verilogで，ソフトウェアは使用する32ビット1チップ・マイコンのMPUモデル上でそれぞれシミュレーションされ，PLIにより連動して動作する．本ツールではPLIの上位バージョンであるPLI2.0を使用することで高速化が図られている．MPUモデルは内部動作を模擬することでサイクル精度をもった命令セット・シミュレータ，デバッガはガイオ・テクノロジー社製のXDEB-Vデバッガを使用した．G-SIMplusでの操作画面の様子をFig.3に示す．

Fig.3　GUI of Co-Verification Tool

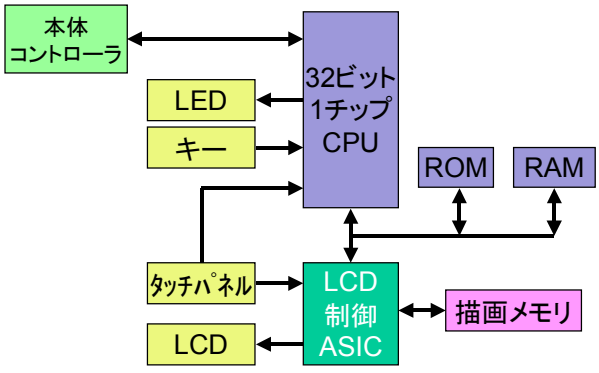

Fig.4　Block Diagram of Operation Panel

パネル操作部の機能ブロック図をFig.4に示す．操作部の制御を行う32ビット1チップ・マイコン，メモリ(ROM/RAM)，主にLCD描画を行うASIC，描画データを記憶しておくための描画メモリ，操作のためのタッチパネル，表示を行うLCD(液晶ディスプレイ)，LED，キーから構成されている．このうち，ユーザ・インターフェース部にあたるタッチパネル，LCD，LED，キーにはC言語記述モデルをもちいた．これらのモデルはWSのディスプレイ上にその形状を表示し，タッチパネルとキーはマウスで操作することができる．また，本来はMFP機本体からシリアル・インターフェースを介してコマンドを発行することにより操作部の制御が行われるが，今回はコマンド・インタプリタによってこの部分をモデル化している．操作部を動作させるソフトウェアはC言語で書かれ，リアルタイムOSも含んでいる．このソフトウェアの主な機能はMFP機本体からのコマンドにより必要な画面をLCDに表示させ，タッチパネル入力を解析してMFP機本体に通知することである．

今回の評価では，ハードウェア・モデルの検証精度が以下のように異なる3種類のモデルを作成した．

【RTLモデル】

ASICおよび描画メモリについてはRTL記述モデルを使用した．ASICはマイコンのデータ・バス，アドレス・バス，制御信号に接続されている．したがって，ASICを動作させるためにマイコンのバス信号を生成するバス・インターフェース・モデルをRTLで作成している．高速化のためROM，RAMはツール内部モデルとして動作させた．

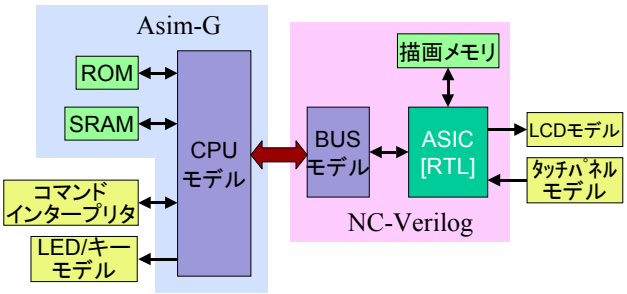

Fig.5　RTL Simulation Model

【描画ビヘイビア・モデル】

本システムにおいてシミュレーション時間がかかるのは描画メモリとASICによる画面描画であると思われるため，描画メモリをG-SIMplusの内部メモリとし，ASICの描画機能を含むLCDモデルをC言語で作成した．

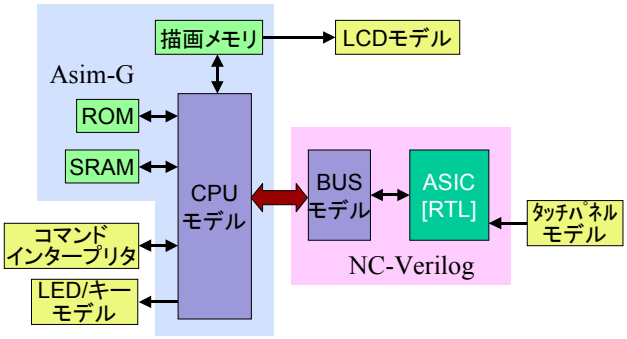

Fig.6　Behavioral Simulation Model for Drawing LCD

【ビヘイビア・モデル】

上記モデルからさらにASICのタッチパネル制御機能をRTL記述からビヘイビア記述にしたモデルを作成した．

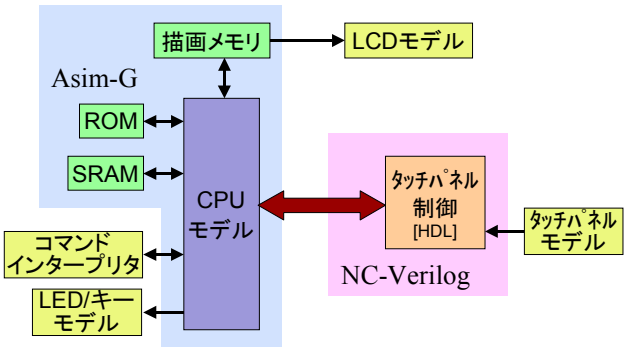

Fig.7　Behavioral Simulation Model

以上3つのモデルを使用し，パワーオン・リセットからMFP機本体(ここではコマンド・インタープリタ)との通信回線をオープンし，起動時のコマンド・シーケンスを行った後，初期画面描画を終了するまでのシミュレーション時間を測定した．命令数は約200万命令，測定に使用したマシンはUltra Sparc 450MHz(2CPU構成)である．操作パネルの初期画面を表示した様子をFig.8に示す．

Fig.8　GUI of Co-Verification for Operation Panel

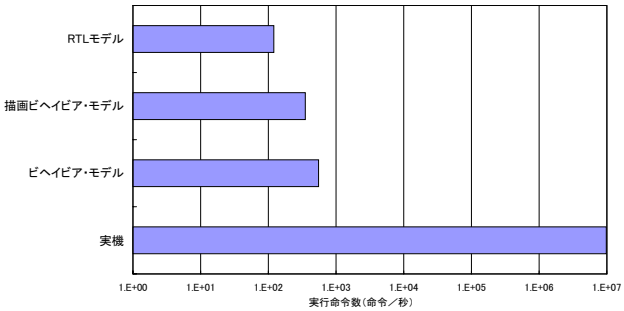

Fig.9　Comparison of Number of Executed Instruction

各モデルを使用した協調検証の単位時間あたりの実行命令数を実機も含めFig.9に示す．RTLモデルの120命令／秒に対して，描画ビヘイビア・モデルでは350命令／秒，ビヘイビア・モデルでは550命令／秒となった．ハードウェアの機能をビヘイビア・モデルにすることにより，Verilog上でのイベント数が削減され検証速度が大幅に向上することがわかった．

各モデルによる検証精度と検証速度をTable 1にまとめた．検証精度と検証速度がトレードオフの関係となることが確認された．今回の協調検証において，ハードウェアのモデルを精度によって3種類用意したが，操作部ソフトウェアについては何も変更していない．したがって，ソフトウェアの検証には検証速度が速いビヘイビア・モデルが有効であるといえる．

Table 1　Comparison of Co-Verification Tool Performance and Accuracy

	検証精度	検証速度（命令／秒）
RTLモデル	クロック精度	120
描画ビヘイビア・モデル	描画機能以外クロック精度	350
ビヘイビア・モデル	機能の検証	550

2-3　協調設計の技術動向

システム・レベル設計では，性能，コスト，消費電力などの配分を考え，ハードウェアとソフトウェアの最適分割，CPU，メモリ，バス，周辺論理回路などのアーキテクチャを決定する．従来はこれらを経験と勘で行っていたため，ハードウェアの不具合をソフトウェアで回避したり，ハードウェアの作り直しが生じたりして，その結果開発期間が長くなっていた．

こうした問題を解決するため，協調設計では，上流工程であるシステム設計段階で，経験と勘ではなくシミュレーションによる検証を行うことで最適なシステム設計を行う．さらにハードウェアとソフトウェアの分割をできるだけシステム設計の後半にまわすことによって，後戻りすることなく最小コストやパフォーマンスへの対応が可能になる．

協調設計ツールはこのようにシステム・レベルの検証，ハードウェアとソフトウェアの最適分割を行うことができるツールである．特に各種IPも含めたシステムLSIの設計ではこうした初期段階の設計がより重要になるため，協調設計ツールの必要性が叫ばれている．

このようなツールは以前から通信や画像処理のようなデータ・ドリブン型(データが各ブロックで順次処理されていく)のアプリケーションで使われており，最近では携帯電話のシステムLSIの設計事例もある．これまでプリンタなど複雑な制御系システムに適応できる協調設計ツールはなかったが，最近になってこうした製品も出始めている．

また，システム設計ではハードウェアとソフトウェアを区別することなく機能を記述することができるシステム記述言語が必要とされ，いくつかの推進団体によって開発が進められている．主な言語としては，シノプシス社，コ・ウェア社などが推進しているSystem C[2](C/C++を拡張した言語)，カリフォルニア大学アーバイン校が開発し日立，東芝など日本の主要メーカーが推進しているSpec C[3](Cをベースとした言語)，コ・デザイン・オートメーション社，デナリ社などが推進しているSUPERLOG[4](Verilog HDLをベースとした言語)などがある．これらの言語に対応した各種ツールの開発もすでに始まっている．今後システム記述言語を用いたシステム設計が主流となることも予想される．

3. 成果

我々はMFP用パネル操作部について協調検証環境を構築し，協調検証ツールの機能・性能評価を行った．ハードウェアをビヘイビア・モデル化することによって検証速度が向上すること，また検証精度と検証速度がトレードオフの関係にあることを示した．

さらに，新しい技術動向として協調設計とシステム記述言語について調査を行った．

4. 今後の展開

今後はOS(Operating System)やデバイス・ドライバを含むシステム・レベルでの検証，性能評価を行う環境構築を進めていく予定である．そのためには検証速度を格段に上げる必要があるため，C言語やシステム記述言語によるモデリングが必要であると考えている．

謝辞

本研究を行うにあたり，ご指導，ご協力いただきましたプリンタ事業部，C/MF事業部，IT/S推進センター，電子デバイス事業部の皆様に深く感謝いたします．

参考文献

1) 特集 システムASIC ハードとソフトは並行開発が常識に，日経エレクトロニクス，no.697 (1997) pp.67-85

2) Open SystemC Initiative, http://www.systemc.org/

3) SpecC Technology Open Consortium, http://www.SpecC.gr.jp/

4) Superlog2000 Partners Program, http://www.superlog.org/

注1) G-SIMplus, XDEB-Vはガイオ・テクノロジー社の商標です．

注2) NC-Verilogは米国ケイデンス・デザイン・システムズ社の商標です．