コンピュータ オン シリコン時代のハードウェア／ソフトウェア協調設計環境と協調検証環境

Hardware/Software Codesign/Coverification Environment for Computer on Silicon

中嶋 弘明
NAKAJIMA Hiroaki

半導体の微細化技術の進歩により，プロセッサ，周辺論理回路のハードウェア（HW）やその上で動作するアプリケーション ソフトウェア（SW）が，1 チップ LSI に搭載されるコンピュータ オン シリコンの時代になった。このような LSI 設計では，開発工程の後戻りをできるだけ少なくし設計期間を短縮することが必要である。そのためには，設計初期段階において，性能やコストの見積りが行え，HW の試作前に HW と SW を含めたシステム検証ができるようにすることが重要である。

当社では，RISC プロセッサ（TX System RISC）コアが組み込まれるシステム LSI の HW/SW 協調検証環境を構築・提供している。この結果，従来手法に比べて設計期間がほぼ 1/2 となる。また当社では，より高度な HW/SW 協調設計技術の研究・開発も進めている。

The progress in silicon device technologies has made it possible to create a computer on silicon (COS), which the application software and the processor and peripheral hardware are embedded on a single chip. In such LSI design, it is necessary to minimize redesign work and to shorten the design time. To achieve this, it is important in the initial stage of the LSI design process to estimate the performance and cost and to enable system verification cooperatively between the software and hardware before the engineering sample is created.

Toshiba is developing and supporting the hardware/software coverification environment for system LSIs having an embedded RISC processor (TX system RISC) core. Consequently, the design time is generally halved compared with the conventional technique. Toshiba is also engaged in research and development of more advanced hardware/software codesign technologies.

1 まえがき

コンピュータ オン シリコン（COS）の時代を迎え，1 チップ上に，プロセッサ，周辺論理回路やメモリなどの HW やその上で動作するアプリケーション SW も搭載することができるようになった。

このようなシステム LSI の設計では，従来，システム仕様確定後，設計者の勘と経験を頼りに，性能，チップ面積，消費電力，システム全体の仕様変更の自由度，開発費などを考えながら，HW と SW のトレード オフを考慮しつつ HW と SW に分割する作業を行っている。そして分割後は，HW と SW とが独立に開発され，HW 試作後に初めて SW も含めたシステム検証が行われる。この段階で仕様の誤りや設計ミスが見つかっても，HW 側へのフィードバックは難しい。もちろん，SW 側で対処したとしても，SW 修正によるコードサイズ（ソフトウェアの大きさ）の増加で，予定していたメモリ容量で不足する場合はチップを再設計をすることになり，設計期間や開発費が増大することになる。また HW 側の問題を SW 側で無理に対処するため，最適な設計がなされにくい。

プロセッサの高性能化が進むにつれ，従来，HW でしか性能が満たせなかった機能が，SW でも性能が満たせるようになってきており，HW と SW のトレード オフを定量的に見ながら HW と SW に分割する HW/SW 協調設計（コ デザイン）（図 1）が注目を集めている。

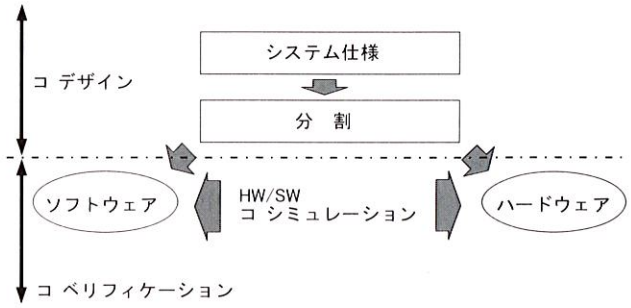

図 1．コ デザインとコ ベリフィケーション　システム仕様を HW と SW に分割するまでがコ デザイン，分割後に HW と SW を同時にシミュレーションし，システム検証することがコ ベリフィケーションである。

Codesign and coverification

しかしコ デザインは，大学での研究レベルはもちろんのこと，やっと EDA（Electronic Design Automation）ベンダー

による支援ツールが発表され始めたところである。

一方，HW/SW 分割後に HW の仕様と SW の仕様を同時にシミュレーションし，システム全体を検証する HW/SW 協調検証（コ ベリフィケーション）(図 1) ツールが，EDA ベンダーやソフトウェア開発ツールのベンダーから製品として発表・発売されており，他社のプロセッサの多くがこれらの製品でサポートされている。

2 HW/SW コ デザイン環境

大学におけるコ デザインの研究では，より良い成果を出しやすくする目的もあり，適用分野を DSP システムに絞って開発している例が多い。U. C. Berkeley 校は，1990 年頃から，Ptolemy というシステムの研究・開発を行っている。これらの DSP システムに特化したツールは，データパスを作るという用途に限定されているため，HW と SW に分割した後の見積り精度が良い。そこで，大手 EDA ベンダーからも DSP システムに特化したツールが出されている (Signal Processing Worksystem (注1)，COSSAP (注2)，DSP Station (注3)) など。

一方，組込み用途向けマイクロ プロセッサ システムに対するコ デザインの研究も進められており，その研究分野も①既存のマイクロ プロセッサの使用を前提としたもの，②プロセッサの最適命令セットを選び，プロセッサを合成することを含めて最適化を図るもの，の 2 種類に大別される。

①の手法は米国の大学を中心に研究されており，U. C. Irvine 校が SpecSyn，U. C. Berkeley 校が Polis というシステムを研究・開発している。またわが国や欧州では，上記②の手法の研究が盛んである。組込み用途向けマイクロ プロセッサ システムに対するコ デザインの研究では，システム レベルの仕様記述から，HW の設計データや SW そのものを自動生成する HW/SW 協調合成が中心となっている。しかし，実用化にはまだ時間がかかる見込みである。

一方最近大手 EDA ベンダーからは，人手で分割するときの判断材料を提供する支援ツールが発表され始めている。どの EDA ベンダーも大学の研究を基にコンセプトを固めたもので，システムユーザー／半導体ベンダーを巻き込んだ実証実験をスタートしたところである。

どのツールも考えかたは同じで，まずシステムを機能ブロックとアーキテクチャに分け，おのおのグラフィカル ツールを使ってシステム仕様を記述する (図 2)。さらに各機能ブロックの中身は，状態遷移図や C 言語などを使って記述する。

(注 1) Signal Processing Worksystem は，米国 Cadence Design Systems 社の商標。
(注 2) COSSAP は，米国 Synopsys 社の商標。
(注 3) DSP Station は，米国 Frontier Design 社の商標。

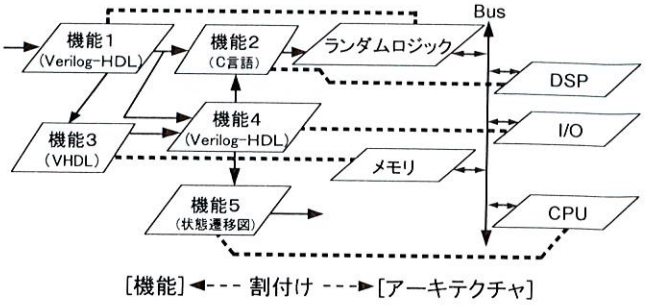

図 2．機能ブロックとアーキテクチャ　システム全体仕様を，機能ブロックとアーキテクチャを割付けすることによって記述する。
Function blocks and architectures

また各アーキテクチャに対しては，CPU なら処理速度を，ランダムロジックならタスク処理にかかるクロック数を，SW なら処理されるクロック数を，リアルタイム (RT) OS ならタスク優先度などを，ライブラリとして用意しておく必要がある。

その後システム設計者が，各アーキテクチャに対し各機能ブロックの対応付けを行い（つまり，人手による HW/SW 分割），システム性能の制約条件を入力すると，性能見積りツールがボトルネックとなる部分を教えてくれる。得られた性能が制約を満たさなかった場合は，再度，HW/SW 分割を人手によりやり直して性能見積りを行う。システム設計者は，性能が満たされるまで，この作業を繰り返す必要がある。

そして性能が満たされると，HW 側は RTL (Register Transfer Level) の HDL（ハードウェア記述言語）を，SW 側は C 言語を出力する。しかし出力されたものについて，HW 側もターゲット テクノロジーを使った論理合成がしやすいように RTL を設計者が介在して直す必要があり，SW 側もターゲット CPU に適したコードになるように，C 言語のソースを設計者が見直す必要がある。

ある程度の性能見積りを支援するツールが，大手 EDA ベンダーからコ デザイン ツールとして発表されているが，より機能拡張・改良を必要としている。

当社としても，各 EDA ベンダーのコ デザイン ツールを評価して，システム LSI 設計環境への組込みを検討する予定である。

3 HW/SW コ ベリフィケーション環境

最近の EDA 業界の動向は，HW/SW コ ベリフィケーションを行う HW/SW コ シミュレータが主体である。コ シミュレータは，CPU と周辺論理（ハードワイヤド論理）回路のシミュレーション モデルを用意し，これらと CPU 上で実行される SW を入力して，システム全体でシミュレーションするものである。

このコシミュレータは既存のHDLシミュレータや既存のSW開発環境ツールを組み合わせて構成していることが多いため，HW，SWそれぞれの設計者にとってはあらためてツールの操作法を覚える必要がない。また，現在使用しているツールや過去の設計資産をそのまま利用できる場合もある。

シミュレーション方式から見ると，HW/SWコシミュレータは，3とおりに大別される（図3）。

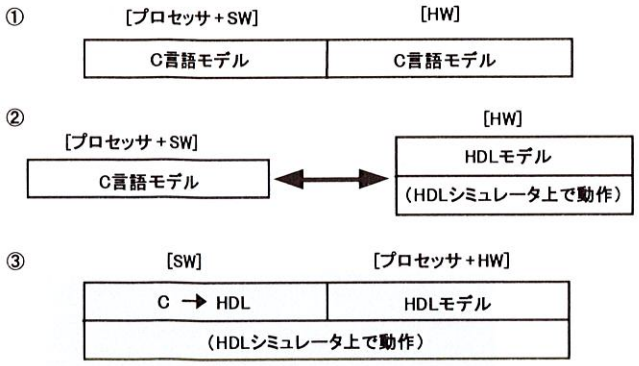

図3．HW/SWコシミュレーション手法　HW/SWコシミュレータを，シミュレーション方式で大別すると3とおりに分類できる。
Hardware/software cosimulation method

シミュレーション方式①は，CPU，周辺論理回路そしてアプリケーションSWまでのすべてが，C言語でモデル化されている。方式②は，CPUとアプリケーションSWはC言語でモデル化されているが，周辺論理回路はHDLモデルを用いHDLシミュレータ上で動作させ，CPU部とHDLシミュレータが通信および同期をとりながらシミュレーションが進む。方式③は，CPU，周辺論理回路ともHDLでモデル化されており，アプリケーションSWもHDLシミュレータが読めるフォーマット（実際の中身は，ターゲットCPUのマシン語の羅列）に変換して，すべてをHDLでシミュレーションするものである。一概には言えないが，方式③より②，①になるほどシミュレーションの処理速度は速くなる（表1）。

一方，CPUのシミュレーション モデルだけを考えても，単に命令を逐次実行する命令セット モデル，パイプラインまで考慮しサイクル精度を上げたサイクル アキュレート モデルがある。当然，精度とシミュレーション速度はトレード オフの関係にあるため，精度が上がればシミュレーションの処理速度は低下する。

HW/SWコシミュレータの処理速度は，実デバイスの速度に比べてけた違いに遅い。そのため試作したHWを用いたICE（In-Circuit Emulator）を使って，SWのデバッグに慣れているSW設計者にとっては，結果が出るまでのTAT（Turn Around Time）が待てないのが実状である。一方，HW設計者においては，HDLシミュレータが終了するまで相当な時間を費やしている。

したがって，現在のHW/SWコシミュレータはHW設計者向けのものと言える。では，何がHW設計者にとって有効であるかというと，従来では対CPUとの信号を想定して設計者がテストベクタ(注4)を作成していたが，実際のアプリケーションSWを実行させることで，その結果として，テストベクタが自動生成されるため，テストベクタ開発リソースの軽減ができるところである。また，実アプリケーションSWを用いているため，よりシステムに近いHWのデバッグもできるようになる。

一方，SW設計者にとってみると，結果が出るまでの時間を考えるならば，マシン語レベルで約200～1,000ステップ程度のデバイス ドライバSWのデバッグには有用である。

当社では，方式①を取っているASVP（Application Specific Virtual Prototype）Lab(注5)用のTX39プロセッサ モデルを用意し，SWデバッガMULTI(注6)とつながるHW/SWコ ベリフィケーション環境を構築した（図4(a)，図5）。引き続き，他のTXシリーズにも移植を進める計画である。

方式①としては，ASVP Labの他にArchGen(注7)があり，プロセッサはもちろんのこと周辺論理回路のHWも，状態遷移図を書くことでC言語モデルの自動生成が可能である（図4(b)）。この自動生成されたC言語モデルはRTL-C(注8)と呼ばれており，サイクル アキュレートなモデル（精度を上げたシミュレーション モデル）になっている。そのため，表1の①の値よりシミュレーション速度は1けた程度遅くなる。

一方，実際LSIを開発するためにはHW側をHDL（RTL）で記述する必要があり，部分的にHDLモデルが現れるトップダウン設計を行うのが至極当然である。またHW-IPにC言語モデルが付いていないことも多い。

そのため当社としては，これらの場合も想定し，方式②

表1．HW/SWコシミュレーション処理速度の比較
Hardware/software cosimulation speed ratio

項　目	方式③	方式②	方式①	CPUデバイス
処理速度（サイクル/s）	10	200～2 k	200 k	50 M
比	$\frac{1}{10^7}$	$\frac{1}{10^6}$～$\frac{1}{10^5}$	$\frac{1}{10^3}$	1

(注4)　LSIの機能・論理シミュレーションのときに，各入力端子に対して印加する論理0や1の入力値を羅列したもの。
(注5)　ASVP Labは，米国CAE Plus社の商標。
(注6)　MULTIは，米国Green Hills Software社の登録商標。
(注7)　ArchGenは，米国CAE Plus社の商標。
(注8)　RTL-Cは，米国CAE Plus社の商標。

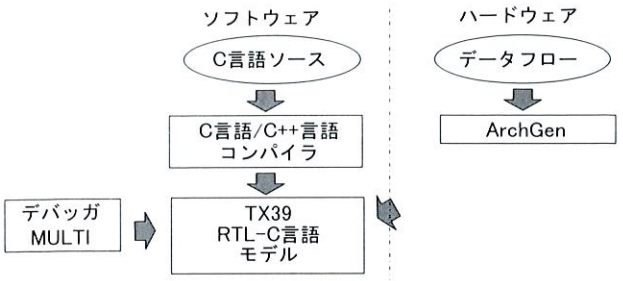

図4．TX39を用いたASVP LabとArchGenによるコベリフィケーション環境　(a)は当社が構築した方式①によるHW/SWコベリフィケーション環境ASVP Labを，(b)はC言語モデル作成ツールArchGenを示す。

Hardware/software coverification environment for ASVP Lab and ArchGen with TX39 C model

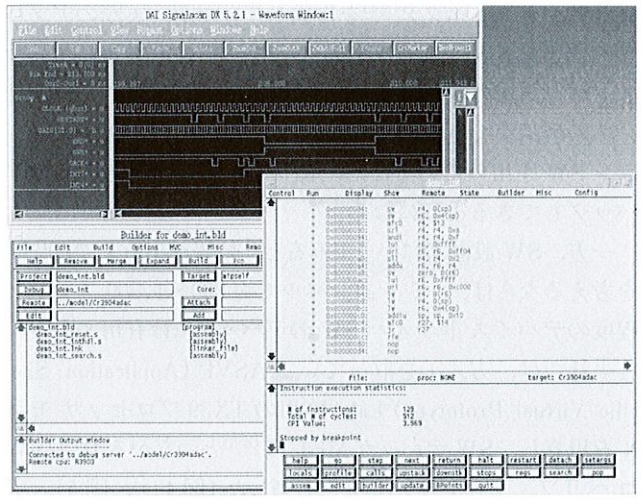

図5．TX39を用いたASVP Labの実行例　TX39を用いたASVP Labの実行画面を示す。

Example of execution of ASVP Lab with TX39 C model

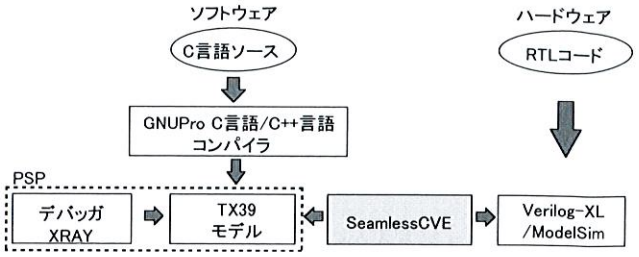

図6．TX39を用いたSeamlessCVEによるHW/SWコベリフィケーション環境　当社が構築した方式②によるHW/SWコベリフィケーション環境SeamlessCVEを示す。

Hardware/software coverification environment for SeamlessCVE with TX39 C model

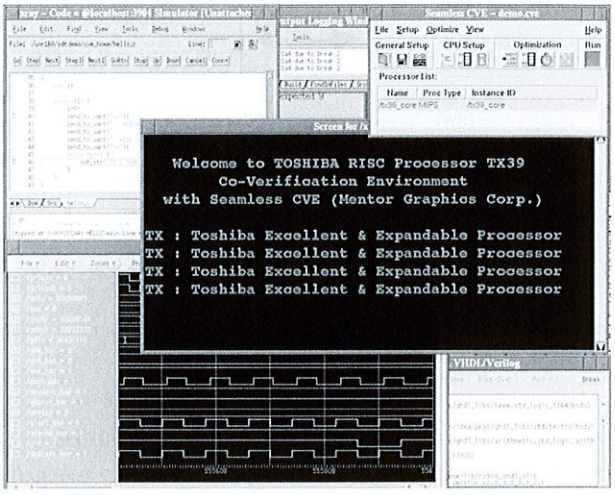

図7．TX39を用いたSeamlessCVEの実行例　TX39を用いたSeamlessCVEの実行画面を示す。

Example of execution of SeamlessCVE with TX39 C model

をとっているSeamlessCVE(注9)用のTX39モデルを用意している。同じくSWデバッガXRAY(注10)やGNUPro(注11)に付随するCコンパイラ(注12)とつながり，HW側のシミュレータであるVerilog-XL(注13)(for Verilog-HDL(RTL))やModelSim(注14)(for VHDL(RTL))と通信および同期をとりながらシミュレーションが進むHW/SWコベリフィケーション環境も構築した(図6，図7)。

現在提供中のTX39プロセッサモデルは命令セットレベルであるが，サイクルアキュレートモデルも開発を進めている。その場合シミュレーション速度は1けた程度遅くなると想定している。

(注9)　SeamlessCVEは，米国Mentor Graphics社の商標。
(注10)　XRAYは，米国Mentor Graphics社の登録商標。
(注11)　GNUProは，米国Cygnus Solutions社の商標。
(注12)　C言語で記述されたソフトウェアを実行するときに，CPUがわかる機械語に翻訳するソフトウェアのこと。
(注13)　Verilog-XLは，米国Cadence Design Systems社の商標。
(注14)　ModelSimは，米国Mentor Graphics社の商標。

4 HW/SWコベリフィケーション環境の効果

当社の主要IPの一つでもあるRISCプロセッサTX39の高速シミュレーションモデルを用意することで，従来よりも2～3けた速いシミュレーションができるようになった。この検証環境を用いると，HW試作前にSWを含めたシステム全体の検証が効率的に行え，テストベクタ開発のリソース軽減ができる。

この結果，設計初期段階で機能や性能の見積り精度が上がり，設計時の開発工程の後戻りによるムダも大幅に減少でき，従来手法に比べて設計期間がほぼ半減する(図8)。

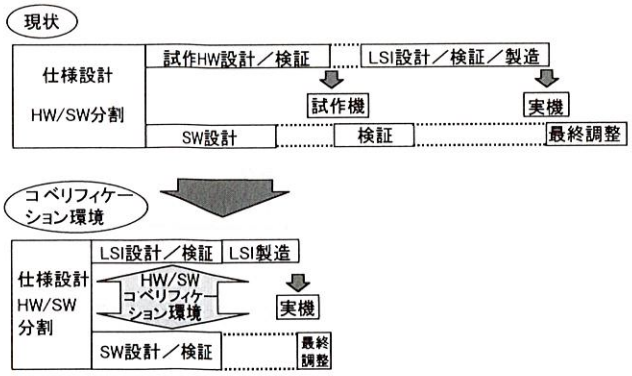

図8．HW/SW コベリフィケーション環境を使用することによって得られる効果　HW/SW コベリフィケーション環境を使用することにより，従来手法に比べて設計期間がほぼ半減する。

Effects obtained by using hardware/software coverification

5 あとがき

当社における HW/SW コデザイン環境とコベリフィケーション環境の取組みについて述べた。しかし，ツール自身がいくら良くなったとしても，それで良いシステム LSI が開発できるものではない。ツールを使用するのは設計者，すなわち人間である。

システム LSI のうちには CPU，メモリ，各種 HW-IP とともに CPU 上で実行されるアプリケーション SW なども実装されるため，多様の文化をもった設計者が介在してシステム LSI ができ上がっている。例えば，HW 設計者と SW 開発者を比べると，前の例にもあるように，許容できるシミュレーション実行時間も違えば，ツール購入の際の金銭感覚も異なっている（HW 設計用ツールは数千万円なのに対し，SW 設計用ツールは数十万円である）。

また CPU コアの設計者と ASIC プリミティブセル（AND，OR やフリップフロップのような基本セル）設計者を比べてみても Characterization (注15)の考えかたも違う。さらに ASIC のデザイン キット担当者は，顧客サポートのためのツールの横展開を念頭におくが，ASSP（Application Specific Standard Product）開発担当者は自分が開発できるツールさえそろっていればよいと考えている。

つまり，システム LSI のインテグレーションを言い換えるならば，人間どうしがもっている異文化のインテグレーションである。異文化をもつ多数の設計者が，互いの文化をわかり合い吸収できる環境を整えることこそが，コンピュータ オン シリコン時代の設計環境構築のために，真っ先にやらねばならないことである。

今後はこの連携をいっそう強め，市場動向に遅れることなくニーズに合った設計検証環境を提供していく所存である。

(注15)　セルのマスクパターンから回路シミュレーション用接続記述を抽出し，さらにその回路シミュレーション用接続記述から回路シミュレーションを通して論理シミュレーション用セルの遅延情報を抽出する一連の作業。

中嶋　弘明　NAKAJIMA Hiroaki

COS 事業推進部 COS 技術推進第三部主務。
RISC-ASIC の応用技術に従事。電子情報通信学会会員。
COS Div.