SystemCプラットフォームへの NECエレクトロニクスの取り組み

NECエレクトロニクス株式会社
第４システムＬＳＩ事業本部　第１マイコン事業部
ＶＲプロジェクト　チームマネージャ
杉本　英樹

１．はじめに

システムＬＳＩの規模は年々拡大し、機能・性能の作り込みは加速度的に難易度を増しており、これに伴いより上流からのトップダウン設計が注目されています。その１つとしてSystemC/C++を中心にUML等の上位仕様記述を取り込んだ環境があり、キャッツ社様からもXModelink seriesを中心としたソリューションが提供されています。

しかし、実際の開発現場では過去の資産や利用する自社・他社ＩＰの制約により、全てを理想的な環境に揃えることが難しいのではないでしょうか。

NECエレクトロニクスは、ASICやISSP等のLSI 開発プラットフォームのプロバイダであると共に、多数のASSP開発を手がけるプラットフォームのユーザでもあります。ここでは、後者の立場でのSystemCソリューションへの取り組み例をその背景と共に紹介したいと思います。

２．システム検証の役割の変化

従来、LSIでのシステム検証の大部分は接続性の検証でした。これは、扱うデータに対しLSI内部のデータ転送能力を十分に大きく取ることが可能であり、転送能力の見積もりは机上検証が可能であったことによります。

しかし、現在では特に映像系などLSIの処理速度を上回るデータ量の増加と、同時に高速インタフェースの普及があり、トランザクションの処理能力の検証が大きなウェイトを占めるようになってきています。（図１）

３．シーケンスと応答性

次に、前頁のethernetの例を使用して、システム検証、特にトランザクション性能検証で何が必要な要素となるか、考えてみましょう。ethernetを扱ったことのある方にとっては当たり前の内容で申し訳ないのですが、説明用に分かりやすい例ということでご容赦ください。

ethernet自体はシーケンスベースのプロトコルであり、上位から見ればリアルタイム性が必須な処理ではありません。（ドロップ・再送を前提としています）

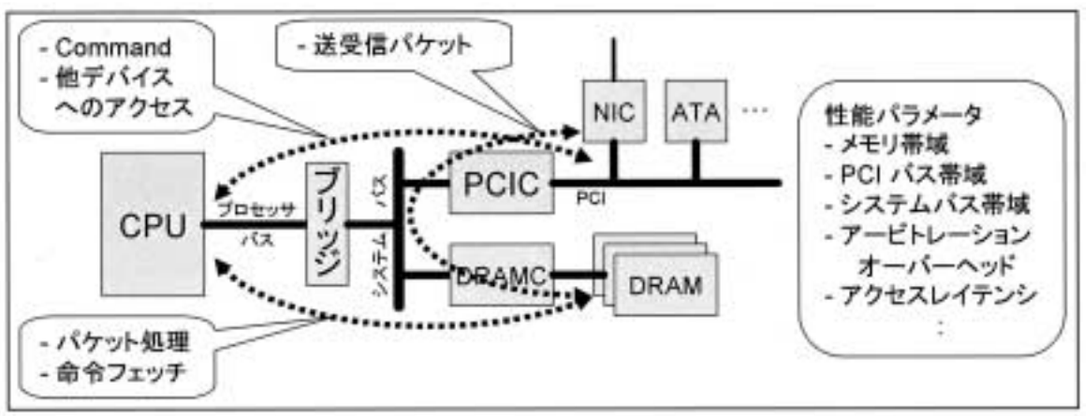

図１: ethernet のトランザクション処理と性能

しかし、プリミティブに見れば、最低１パケットを受け取れるためのハード・リアルタイム性が必要なことはすぐに分かります。但し、このことはシステム検証ではあまり重要ではありません。なぜならばパケット処理はethernetコントローラに閉じた処理であり、システムレベルの動作に影響されないためです。

システムレベルの検証で重要となるのはパケットの処理間隔すなわち応答性で、この検証方法を間違えると再送が発生して所望のパフォーマンスが出ないとか、負荷が集中したときに再送が多発し更に付加が増大してシステムダウンにつながる等の大きな問題になります。特に後者は単独の機能に着目している限り発生し得ない状況ですので、如何にこのようなケースをシステムレベルで再現するかがポイントとなります。

４．開発プラットフォームの実装レイヤ

ethernetの例でシステム検証に必要な１つの特性を説明しましたが、それでは現在システム検証でよく使用されている開発プラットフォームの実装にはどのような特徴があるでしょうか。その実装レイヤ毎に簡単に表にまとめてみました。（表１）

表１　各プラットフォームの特徴

実装レイヤ	実装例	特徴
実システム		実時間精度だが、完成フェーズが遅い
モックアップ	FPGA H/W エミュレータ	高速で一般的にはサイクル精度 変更に対するオーバーヘッドが大きい
Co-Sim		速度は非ハードウェア部分に依存 接続 I/F の作りこみコストが高い
ハードウェア記述言語	Verilog VHDL	低速、一般的にはサイクル精度 高位記述には制約が多い
システム記述言語	SystemC	中速、必要に応じてサイクル精度可 ソフトウェアとの連携がやりやすい
ソフトウェア	C/C++	サイクル精度を扱うのは困難

実装レイヤ毎に特性に差があり、万能の環境は無いことが分かります。

そこで、これらの中から行おうとしているシステム検証に必要な特性を持つものがどれであるかを判断する必要があります。

例えば、先のethernet例のような応答性のシステム検証にはどのような特性が必要でしょうか。それは、以下のようになると思います。

１）検証対象に影響するシステム構成要素が全て含まれること
２）検証対象の構成要素がクロックサイクル精度で動作可能なこと
３）検証対象以外の構成要素の動作制御が容易なこと
４）初期検証では、構成の変更が容易であること
５）検証量に見合ったシミュレーション速度であること

そこで、これらの特性で先の各レイヤでの環境を評価すると以下のようになります。（表２）

SystemCを含むサイクル精度トランザクションベースのシステム検証環境がバランスの取れたソリューションであることがお分かりいただけると思います。特に、高位記述による検証対象関連構成要素のトランザクション発生と詳細レベルでの検証対象の動作検証の混載による検証シナリオの自由度の高さは、ハードウェアベースの高速シミュレータやＣモデルに対するシミュレーション速度の弱点を補って余りあるものです。

表2　必要な特性とプラットフォームの特性

レイヤ	実装例	1)	2)	3)	4)	5)
実システム		◎	◎	×	×	◎
モックアップ	FPGA H/W エミュレータ	◎	○	×	△	◎
Co-Sim		◎	○	△	○	○
ハードウェア記述言語	Verilog VHDL	○	○	△	○	△
システム記述言語	SystemC	○	○	◎	◎	○
ソフトウェア	C/C++	△	×	△	◎	◎

問題は、冒頭でも書きましたとおり、必ずしも全ての要素がこの開発プラットフォームにすぐに乗せられるわけではない点ですが、この点もプラットフォームをサポートするツールの進化により、大幅に少ないコストでの乗換えが可能になろうとしています。

次章では、この点を中心にNECエレクトロニクスでの取り組みを紹介させて頂きます。

５．SystemCへの取り組み

ここでは、マイクロプロセッサの応用システム検証における、NECエレクトロニクスでのSystemCへの取り組み例を紹介します。

マイクロプロセッサの開発と聞くと、システムLSIの開発と接点がないようなイメージを持つ方も多いかもしれませんが、最近では弊社VRシリーズなどGHzクラスのプロセッサであっても大部分は機能合成・論理合成を使用して開発されています。

もちろん、一部で非常に低位な記述を使用することはありますが、ことシステム検証に関しては、後述の理由から上位記述のモデルを使用しており、システムLSIの開発と非常に近い環境となっています。したがって、マイクロプロセッサと関係のない開発であっても参考にしていただけるのではないかと思います。

さて、プロセッサのシステム検証の特徴は何でしょうか。いろいろとありますが、最も重要なのは（プロセッサに限りませんが）「汎用」であることによる動作組み合わせの膨大さにあります。更に最近では「あたりまえ」に接続される高速インタフェース（Gbit etherやS-ATA, PCI-Express等）とのパフォーマンスも含む接続性までもが検証の対象になります。当然、これらの動作にはソフトウェアが必要ですから、システム検証環境は結局実際のシステムに近いものを用意しないといけません。

先に、システムLSIの開発と非常に近い環境と書きましたが、実はほぼ同じものを使用しているのです。（プロセッサ以外の部分は実装を前提にしない点が実は異なりますが）

従来は、これをVerilog/VHDL+PLIによるものと、C++をベースにしたモデルの２つを使い分けることでシステム検証を行ってきました。しかし、上位では先に書いたような高速インタフェースとの接続パフォーマンスなどサイクル精度の要求、下位ではマルチスレッド・マルチプロセッサ化によるソフトウェアとの連帯強化の要求が強くなっています。

それぞれ、サイクルベースのCモデルや、Verilog-C混載シミュレータなどのソリューションもあり実際に導入していますが、プロセッサのような複雑でかつロードマップ開発が必要とされるものでは検証プラットフォームの流用性が開発TATに与える影響は非常に大きく、より応用範囲の広いプラットフォームの追求が重要となります。

そこで、我々が着目し取り組んでいるものの１つがSystemCプラットフォームです。次頁に従来のVerilog/VHDLベースのプラットフォームとSystemCでターゲットとしているプラットフォームの図を示しますが、２つの図を比較すると用途に応じて各資源の記述レベルを柔軟に混載することが出来るSystemCの特徴が良く分かると思います。また、元々プロセッサのシステム検証のシナリオはC言語で記述されている場合が多く、更にOS Kernelやruntime library等も同様ですので、プロセッサを搭載するシステムのプラットフォームとしては理想に近い形といえます。

しかし、ここで忘れてはならないのが何度も書きました過去の資産の継承と既存ＩＰの扱い

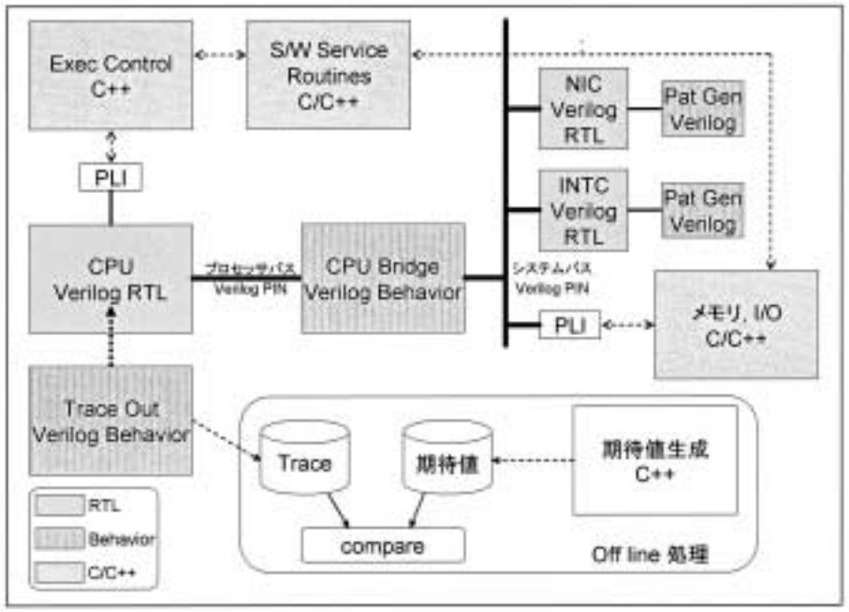

図２　Verilog/VHDLベースのプラットフォーム

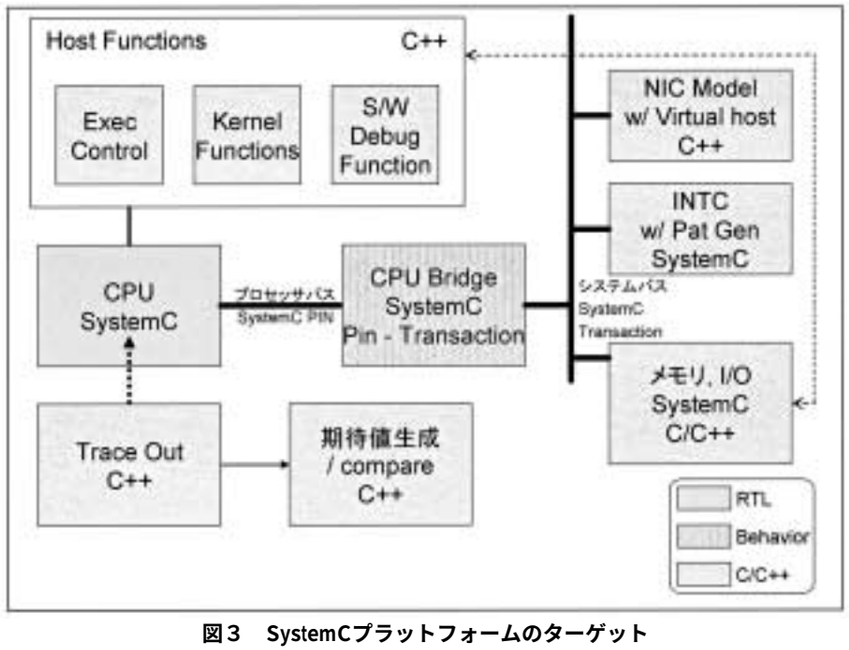

図３　SystemCプラットフォームのターゲット

ZIPC WATCHERS Vol.8

です。我々の場合検証対象となるプロセッサ自体、Verilog記述と非サイクル精度のCモデルのみしかありませんでしたので、SystemCの良さは理解しながらもそのプラットフォームを作ることは不可能にさえ思えました。

ところが、今年に入りキャッツ社様からVerilog to SystemCのトランスレータでありますV2Sの紹介を頂いたことにより状況が大きく変化しました。

６．V2S

V2Sはキャッツ社様が販売代理店（開発元：礎デザインオートメーション社）を務めるVerilogのRTL記述をSystemCにトランスレートできるツールです。　NECエレクトロニクスではこのV2Sの評価の１つとして弊社64bit RISC CPUでありますVR5500のプロセッサ・コア部のVerilog RTLから種々の記述スタイルのサンプルを選択しました。そして、キャッツ様の協力により実際に変換を試行させて頂きました。

VR5500は既に量産中の製品であり、そのプロセッサ・コア部のVerilog RTL記述は、現在の設計メソドロジからすると多少癖のある低レベルの記述やセルのインスタンシエートを多く含むものですが、V2Sでのトランスレートはあらかじめ現在のバージョンではトランスレートに対応していないセルフアサーション用のdisplayタスクを削除するのみで可能であり、出力結果の内容を見ても十分納得のいくものでした。また、モジュール構造は維持されますので、SystemCのプラットフォームの豊富なデバッグ環境を駆使して、クリティカルなモジュールをより高速なシミュレーションの可能な高位記述に修正することも容易です。

もちろん、高位記述自体はVerilogでも可能ですが、元々どちらかといえばリソース／イベント指向であったVerilogとオブジェクト指向のSystemCとでは、その記述性およびシミュレーション高速化の方向性には大きな差があります。

このようにして、V2Sを使用することにより、一時は断念も考えたプロセッサ・システム検証環境のSystemCプラットフォームへの搭載に道筋が出来ました。前頁の図にも示しましたように、検証対象のプロセッサ部はピンレベルで、その他のサイクル精度資源はトランザクションベースで、更にそれ以外の環境および実際のソフトウェア実装部はC++での記述が可能になり、プロセッサ搭載システムのシステム検証としては理想に近いプラットフォームとなっています。

プロセッサ開発の場合、検証でシステムに接続する資源は既存のものがほとんどでありSystemC/C++より上位の記述はあまり使用しませんが、実際のシステム開発においては同様のプラットフォーム上でキャッツ社様のXModelink seriesを始めとする充実した仕様合成ツールを利用可能であり、更に大きな効果が期待できるのではないかと思います。

７．おわりに

以上、NECエレクトロニクスの製品開発でのSystemCの取り組み例を紹介しました。

開発環境の抽象度という観点からのみみれば、今回ご紹介させて頂いたプラットフォームは、最近の上流設計を意識した新規開発案件に比べ実装レイヤの低い部分が多い感があったと思います。

実際、プロセッサ開発という限られた範囲をターゲットにしていることもあり、弊社が提供を実施・計画しておりますASICやISSP等向けの開発プラットフォームに比べても、十分にシステム記述言語のメリットを使い切っているとはいえない部分もあります。

これらの先端プラットフォームはまた別の機会にご紹介したいと思いますが、まだまだ数的には多い従来からの延長線上での開発や流用設計の多いシステムでの適用という意味では、今回の例はその１実装として参考になる部分もあったのではないかと期待しております。

最後に、V2Sを始めとするSystemCプラットフォーム関連ソリューションを提供頂くと共に本誌への寄稿の機会を頂きましたキャッツ社様に、感謝の意とさらなるSystemCソリューションの発展の期待を述べ本稿での紹介を終わらせて頂きたいと思います。