システムＬＳＩ設計技術　108－5
(2003. 1. 28)

FPGA と PC の連携による System On a Chip の検証手法

中村 祐一　細川 晃平

NEC マルチメディア研究所 〒216-8555 神奈川県川崎市宮前区宮崎４－１－１
E-mail: yuichi@az.jp.nec.com, k-hosokawa@da.jp.nec.com

あらまし

本稿では、System On a Chip(SoC)のシステム検証手法を提案する。提案手法は FPGA を使った HW ベースのモデリング手法と PC 上のソフトウエアによるモデリング手段を組み合わせることによって、全体を FPGA でモデリング化するフルチップエミュレータより少ない FPGA の個数で構成できるため安価で、全体を計算機上のソフトウエアシミュレーションでモデル化する場合と比べて高速に実行することができる。FPGA と PC を通信する回路は単純な直接接続になっており、FPGA と PC にモデル化した回路構造に依存しない。また、この通信回路の構成を変更することによってさまざまな FPGA と PC の協調手法が実現できる。３つの LSI 設計においてこの検証手法を適用した結果、３個の FPGA で 0.5MHz から 16MHz の動作周波数でのシステム検証を行なうことができた。

キーワード　SoC, システム検証、FPGA

A Verification Method for System On a Chip by using Combined Approach of FPGA Emulation and PC Software.

Yuichi Nakamura and Kouhei Hosokawa

Multimedia Laboratories, NEC Corporation,

4-1-1, Miyazaki, Miyamae-ku, Kawasaki-shi, Kanagawa, 216-8555 JAPAN
E-mail: yuichi@az.jp.nec.com , h-hosokawa@da.jp.nec.com

Abstract

This paper describes a new verification method for Systems On a Chip. The proposed method integrates FPGA based modeling using an emulation engine and software based modeling on a PC. The communication between the FPGA emulation and the PC is implemented by simple bus architecture. The proposed method achieves low cost, easy debugging, rich portability, and high verification speed. For a cheap verification environment, it must be implemented by at least the number of FPGAs. The method was applied to 3 verification projects involving the design of real chips. In these projects, this verification methodology completes system verification at 0.5-16MHz using at most 3 FPGAs and a Windows PC.

Keyword : SoC, System Verification, FPGA

1. はじめに

SoC は、CPU や DSP のような IP(Intellectual Property)ブロックや、各種インターフェースブロック、ユーザが設計するブロック、そしてそれらの上で動作するソフトウエアから構成されている LSI である。近年の LSI の大規模化と、LSI と動作ソフトウエアを含めたシステムの複雑化により、LSI 製造前における LSI 全体のシステム検証の重要性が増している。特に、LSI 全体のシステム検証を高速、かつ低コストに実行することが重要である。

従来、LSI の論理・機能設計検証には Verilog-HDL[1]や VHDL[2]のレジスタトランスファレベルの言語を使ってモデリング化し、それらの言語のシミュレータを使って検証作業やソフトウエア開発作業が行なわれてきた。RTL シミュレータはソフトウエアで構成されており、対象回路の設計とテストベンチを準備するだけで、簡単に検証を実行することができる。また、LSI 内部の動作情報の取得が容易で、かつ、波形表示などのデバッグ容易化ツールが完備されており効率のよい検証作業が実行可能である。

しかし、RTL シミュレータは低速であり、100Kgate 程度の大きさをもつ SoC の１ブロックの単体検証でも 100Hz 程度、数 MGate クラスの SoC 全体検証の場合は、1Hz 程度の動作速度が限界である。このことはすなわ

ち、RTLシミュレータの動作速度を100Hzとした場合、100MHz動作の実SoCでたった1秒の検証パタンであっても、その実行に10日以上が必要であることを意味し、実際動作と等価なパタンによる検証は不可能である。実際の設計検証においては、検証実用上の時間に短縮・概略化した検証パタンによる検証を行なうか、あるいは、RTLシミュレータで高速動作可能な10KGate程度の細かいブロックにSoCを分割して検証を行っている。この結果、検証漏れが頻発してSoCや動作するソフトウエア、あるいはSoCとソフトウエア間の接合部分にバグが発生する。その結果、SoCの再設計や、SoCを搭載した製品の回収といったコストが発生する。

この問題の解決のために、以下の2つの手法が提案されている。1つは、RTLよりより高度の記述、すなわち、Spec-C[3]やSystem-C[4]などのC/C++ベースの上位記述言語と、そのシミュレータを使った検証環境である。これらの記述では、必ずしもクロックを記述する必要がなく、処理の流れをシーケンシャルに記述できるため非常に高速に実行できる。その動作速度は、CPUなどの単体モデルで数100KHz、SoC全体のシステムモデルでは10KHz程度である。これにより、100MHz動作のSoCにおける1秒は数時間に削減されて、システム検証として十分実用に耐える。しかし、高速化を達成した副作用として、クロックやレジスタの記述が残っておらず、クロック同期なレベルでの検証という意味からは精度が低い。

もう1つの検証手法がFPGA(Field Programmable Gate Array)などのハードウエアエンジンを使ったエミュレーションとアクセラレーションである。Ultra Sparc Iの検証プロジェクト[5]はFPGAエミュレータを使ったシステムLSIの検証として非常に有名である。このプロジェクトでは、1000個以上のFPGAを使った巨大なハードウエアエミュレータを使ってRTL記述のCPUをモデリング化し、OSを起動することによってCPUの検証を行なった。動作速度は1MHz以下であったが、必要なレジスタはロジックアナライザで確認が可能であり、実際動作に使われる検証パタンを用いて検証を行なうことができた。この結果、エミュレーション検証において25-50%のバグをLSI製造前に発見できたと報告されている。このUltra Sparc Iの検証後、LSIの製造前検証、あるいはLSI製造前のファームウエア開発にエミュレータの適用が盛んになり、いくつかのプロセッサ開発において、テストベンチの入出力方法や信号の観測手法を工夫して、各種のエミュレータの適用が行われている[6][7][8]。

しかし、一般にフルチップエミュレータを使った検証手法に関しては以下にような問題点がある。

1) コストが高い。すなわち、SoC全体をFPGAの利用によりモデリングしようとすると、多数のFPGAを必要とするため、エミュレーション装置の規模が増大し、検証コストが高くなる。

2) FPGA内の信号観測に手間が必要である。すなわちロジックアナライザの利用による信号観測は非常に面倒であり、またFPGAの端子から遠い位置にあるFPGA内部信号線の観測が難しい。

3) 取り扱いが困難である。大きいサイズのハードウエアは取り扱いが困難になり、検証環境構築のために長時間の時間を必要とする。また、安定動作や保守のための仕組み作りが必要である。

そこで、高速動作、低コストの検証手法がいくつか提案されている。これらの手法は、少ない個数のFPGAを利用してコストを下げ、運用容易性を確保しながらも、高速検証性を維持しようという手法である。

本稿ではSoC全体をFPGA化する代わりに、一部をPC上のソフトウエアを使ってモデル化して補完することによって、低コストで扱いが容易なシステム検証手法を提案する。この手法はSoCの特徴である、再利用ブロックやCPUやDSPなどのIPブロックの存在をうまく利用する。本稿では、2章においてFPGAの個数を削減するための従来手法を説明し、3章において本提案検証手法の基本手法に関して述べる。4章においては、提案手法を使った実際の検証事例について述べ、5章でまとめを述べる。

2. システム検証の高速手法

クロックレベルの検証精度を保ったまま、SoC全体のシステムの高速な検証速度を実現するためには、FPGAなどのなんらかのハードウエアを使った加速機構が不可欠である。しかし、SoC全体のシステム検証を行なうためにSoC全体をハードウエア利用のモデル化を行なった場合、システムのコストと検証の準備にかかるコストの両方が問題となる。そこで利用するFPGAなどハードウエアの加速機構を最小化してSoCのシステム検証を実行するための手法がいくつか提案されている

1つの手法は、検証ターゲットの抽象度を上げてFPGA上のモデル化を施す手法[9][10]であり、RTLシミュレータとC/C++レベルシミュレータとの中間レベルのモデルをFPGAにモデリングする手法である。この手法ではFPGAにモデル化される回路はSoCとピンレベルで等価であるが、クロックに対して完全には等価な回路としてはモデル化されない。たとえば、[10]の手法はSoCを細かいブロックに分割し、それらを小さな多数のプロセッサと、その上で実行される命令セットの実行結果でモデリング化する。SoC上のレジス

タも命令セットの実行結果として参照可能である。この手法では、たった1個のFPGAを利用して、100KgateのLSIの検証が50-100KHzで検証が可能であった。しかし、大規模回路に対応するために、プロセッサの命令コードのスタックを大きく確保する必要があるため、プロセッサの付近に高速で大容量なメモリが必要となり、大規模回路の対応が困難となる。現在市販の最大規模のFPGAを利用しても、500Kgate程度がその限界である。

もう1つの手法は、ソフトウエアシミュレーションとハードウエアエミュレーションの連携手法である。この手法はFPGAを詳細検証部分と概略検証部分に分割し、詳細検証ターゲットブロックはFPGAに、その他の部分はPCにモデリング化する手法である。たとえば、図1に示すように、[11]に提案されている手法の場合は、FPGAに一部回路を直接に論理回路としてモデリング化し、この一部回路のテストベンチとなるアプリケーションプログラムをPC上で発生させ、FPGAに与えてシステム検証を行なうというものである。この手法の場合は、1.5MgateのSoCに対して700KHz程度の動作周波数で検証可能であったが、PCはあくまでもテストベンチの生成機でありFPGAの個数はフルチップエミュレータと比較してあまり削減されていない。また、PCとFPGA間の接続は、テストベンチの大量送信を仮定しているために FIFO(First In First Out)のメモリになっており、PCとFPGA間を同期が困難であり、厳密な意味でのクロックレベルのSoC全体のシステムモデリングができているとは言えない。

本提案手法では、この[11]の問題点である、FPGAの個数の問題とPCとFPGA間の同期の問題を改善した。提案手法では、高々3個のFPGAのみを使い、SoCの一部を、その設計に応じたソフトウエアモデリングをすることで、PCとFPGA間の効率のよい協調によるシステムモデリングを実現した。

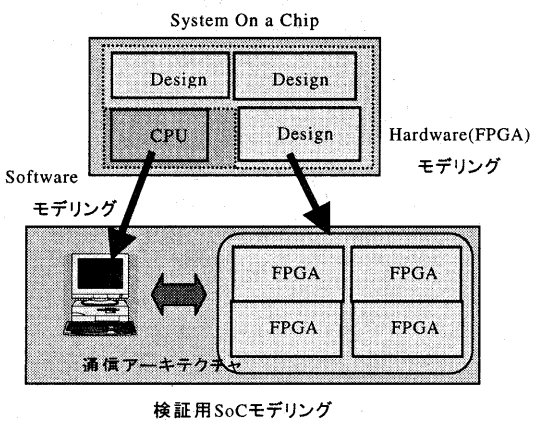

図1 FPGA/PC協調モデリング手法

3. PCとFPGAの協調モデリング手法

SoCの一部FPGA化による高速性を保ったまま、低コスト検証を行なうためにはソフトウエアとFPGAの通信機構の改善が不可欠である。本稿ではSoC全体のシステム検証の速度と、FPGAの利用個数を削減するために、多様なソフトウエアモデリングが実現可能なPCとFPGAの通信機構を持ったSoC検証手法を提案する。

3.1 PCとFPGAの接続手法

PCとFPGA間を接続する接続機構は[12]の手法ではメモリを使ったFIFOで実現されていたが、提案手法では図2のようにPCがPCIを経由してFPGA上の回路を読み書きする構造を採用した。この構造の場合、FIFOが有利な大量なデータ転送には処理時間が必要となってしまうが、PC上でソフトウエアモデリングされたSoCの一部分と、FPGA上の回路の同時間性を保つことができる。この直接通信の利点は以下の通りである。

1) PC上のソフトウエアがFPGA上の信号の読み書きが可能
2) 上記の機能を利用して、データとアドレスを渡すことによりメモリの読み書きが可能
3) PCからFPGAの回路への命令転送が、PCと同時刻で行なうことが可能

これらの処理を実現するためには、FPGA上にインプリメントされる検証ターゲットブロックに、最低限のPCからのデータ転送を受け付けるための回路を付与する必要がある。この付与回路は検証ターゲットの構成により自由に変更することができ、4章における検証例の項において詳しく説明する。

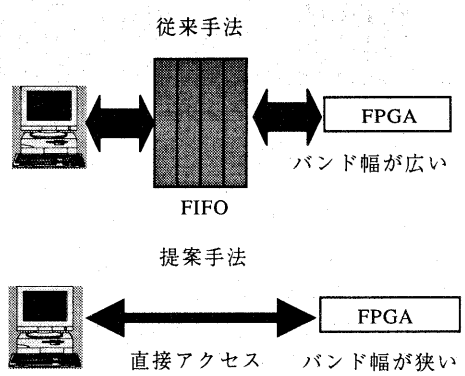

図2 PC-FPGA連結手法

3.2 FPGA ボードの構成

本提案手法では、FPGA を数個のみを利用するというメリットに着目した。FPGA の利用数を 3 個までと限定した場合、コスト以外にも以下のような利点がある。

1) FPGA 間の接続をデザイン依存で変更するためのルーティングチップが不要となり、動作速度を高く設定でき、環境構築も容易。
2) FPGA が多数ある場合は、FPGA 間の接続本数が少なくなり、FPGA の容量が大きくても使用効率が悪くなるが、FPGA の個数を少なくすることで FPGA 間の配線数が増加。
3) FPGA の自動分割ソフトウエアが不要。3 個程度なら手分割でも可能。
4) 接触不良などの確率が下がり安定動作

このような利点を生かすために、本提案手法を実現するために専用の FPGA ボードを開発した。このボードは図 2 のような概観を持ち、Xilinx[13]の VirtexE シリーズの 2000、あるいは 3200 の FPGA を 3 個搭載している。

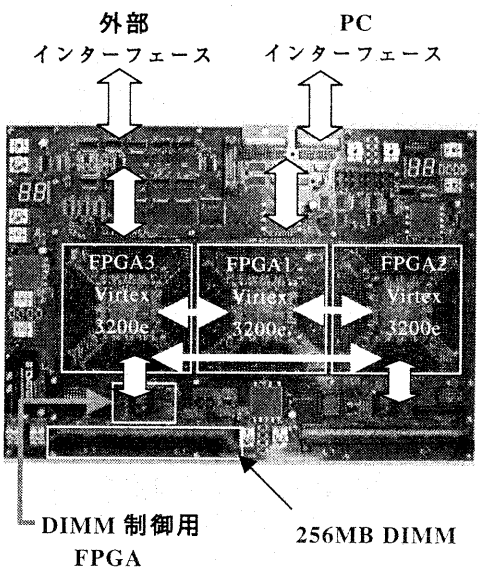

図 3 FPGA ボードの構成

このボードには、直接通信を行なうことにより[11]の手法と比較して劣る大量のデータ転送を不要とするために、大容量のメモリを搭載した。この大容量メモリは PC 用の DIMM であるが、制御回路を検証デザインインプリメント用の FPGA と DIMM の間に挿入し、制御回路が DRAM の制御に必要なリフレッシュなどの定期的な動作や、Read/Write の前後に必要な手続きをこの制御回路が受け持つことによって、アドレス、データ、および Enable 信号だけで読み書きが可能なメモリを構成している。この制御回路の動作は検証ターゲット FPGA と非同期に実行され、図 4 に示すように動作クロック差を利用して、DRAM を SRAM のような単純なメモリとして利用可能にしている。制御用 FPGA の動作周波数は 90MHz であり、1 度の Read/Write に必要なクロック数は 17T であるため、このメモリを利用する場合の検証ターゲットの最大動作周波数は約 5MHz となる。

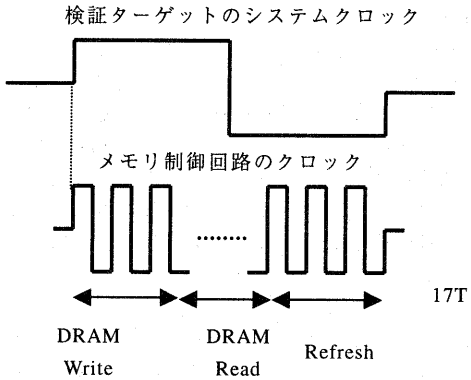

図 4 メモリ制御回路

3.3 PC の構成

PC には、FPGA ボードと接続するための PCI[12]カードが挿入され、PCI カードを経由して FPGA ボード上の通信回路と通信を行なうためのデバイスドライバが実装されている。デバイスドライバは Microsoft の Windows 上のシステムとして実装されており、デバイスドライバに通信回路、または検証ターゲットの位置情報とデータ、読み書きの種別をデバイスドライバに付属するライブラリの引数として与えるだけで、FPGA 上の回路の読み書きができるように構成した。検証システム全体の機能構成は図 5 のようになる。

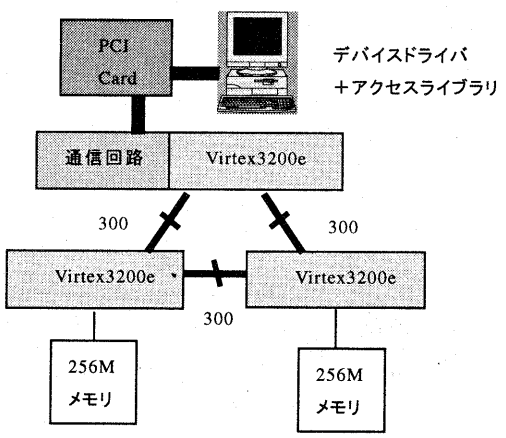

図 5 システム構成

4. 適用結果

このFPGAとPCの連携によるSoC検証手法を、実際のLSI設計に対して適用した場合の適用結果について述べる。実際に適用ではSoC中のCPUやDSP,などのハードマクロ、あるいは再利用ブロックなどの内部検証不要なブロックをソフトウエア側に、それ以外をFPGA側でモデル化することによって検証用FPGAの個数を削減している。また、すべての場合において数日から1,2週間で検証環境の構築を完了している。

4.1 画像処理用SoC

このSoCは画像の拡大や縮小などの変換を行なうためのLSIであり、全体は1.5Mgateである。SoCを構成するブロックはCPUと画像変換部本体、同期回路、メモリI/Fで構成されている。画像変換部はCPUからの命令に従って、外部バスから入力される画像データを拡大あるいは縮小などの変換を施し、メモリI/Fに出力する。これを以下のようなモデル化を行った。

1) CPU部分はハードマクロなので詳細検証が不要であり、単純なCプログラムに置き換えてソフトウエアとしてモデリング化した。画像変換部、同期回路に対する命令をFPGA上に書き込む。
2) 画像変換部、同期回路はRTLをFPGAに実装してFPGAでモデリングした。
3) 外部バスにより入力される画像データはPCからボード上の大規模メモリにあらかじめ書き込こみ、画像変換後の画像データは大規模メモリに書き込む。
4) 処理終了後、変換後の画像データをPCに移動し、変換結果をPC上のビューワで確認する。

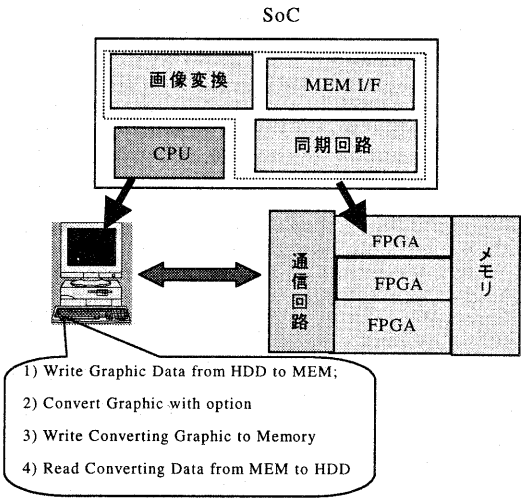

図6 画像変換SoC検証環境

このモデル化において、FPGA実装した画像変換部と同期回路の容量は約650Kgateであり、Virtex2000eのFPGA3個で実現され、動作周波数は1MHzを達成した。通信回路は、動的に変化するCPUからの命令の伝達用に実装され、画像変換ブロックと同時にFPGA合成された。また、画像データをPCのディスクに格納するため、画像表示装置が不要なメリットもあった。

この検証モデルにおいて、数千の画像と画像変換のオプションを使ってこのSoCのシステム検証を行ったところ、1枚の画像処理に数時間必要なソフトウエアのRTLシミュレーションでは検出が不可能だった画像変換部のバグ、ソフトウエアとハードウエアの仕様不一致などを検出することができ、LSI製造後の実際の動作においては1件の問題なく動作した。

4.2 通信用LSI

このLSIは2個のセルベースLSIとハードマクロDSPからなる3LSIのチップセットうち、1LSIを構成しており、その入出力は他の1個のLSI(Other)とDSPから与えられる。このLSIの内部ブロックはさらに5分割されており、それらが役割に応じて独立に動作し、5ブロックが順々に設計されていった。最初は設計完了した1ブロックのみをFPGAに実装し、設計未完の4ブロック、OtherとDSPは、その仕様から与えられる境界の切り口の端子をソフトウエアにより動的にモデリングすることにより実装した。この最初のブロックの検証後、他の4ブロックを設計順に次々にFPGA化していき、最終的には5ブロック全部をFPGA化した。また、DSPとOtherも実チップとしてFPGAボードに接続し、16MHzで動作させることに成功した。最終段階ではPCはFPGAボードの初期設定とデバッグ用の信号を定期的にFPGAから受信する役割を実装した。この検証環境ではRTLシミュレータでは検出できなかったバグを複数発見し、製造された何の問題もなく動作した。この次々とFPGA化していく過程における、動作周波数とFPGAの回路規模の関係を表1に記す。

検証フェース	境界端子モデリング	FPGAモデリング	Gate Size	MHz
1	4ブロック DSP, Other	1ブロック	200K	0.5MHz
2	3ブロック DSP, Other	2ブロック	350K	1MHz
3	1ブロック DSP, Other	4ブロック	800K	10MHz
4	DSP, Other	全ブロック	850K	11MHz
5	実チップ接続	全ブロック	850K	16MHz

表1 段階的検証

4.3 CPU

乗除算器の回路は配線数が多く、FPGA で実現するとゲート数以上の多くのリソースを消費する。また CPU の設計上、乗除算器はハードマクロとしてカスタム設計する場合が多いので、FPGA 化して検証する必要性があまり高くない。また、プロセッサにおいては、入出力などを担当するシステムインターフェースが不可欠であり、FPGA などで実現する場合、このシステムインターフェースの実現のために専用モニタや入出力装置を必要とする。

そこで CPU の検証のための本提案手法を使った環境では、乗除算器とシステムインターフェースをソフトウエアでモデリングし、残りを FPGA で実現する手法を実現した。すなわち、ファームウエア中に乗除算命令が発生した場合、FPGA 上の回路を止めて PC にアクセスし、PC のプロセッサに乗除算を実行させてその結果を FPGA に戻すように通信回路を実装した。また、システムインターフェースも同様で、ファイルへの Write 命令が発行された場合、PC のハードディスクに Write を行なうように通信回路を構成した。ファームウエアはボード上のメモリにあらかじめ格納される。この環境において、FPGA は 10MHz で動作するが、ソフトエウアの実行を含むので、SpecInt92 の実行の場合で 3MHz、SpecFp92 の実行の場合では、500KHz で動作した。さらに、動作中の FPGA のレジスタなども、乗算など値の転送機構を利用して RTL シミュレータと同様に PC 上で表示可能とした

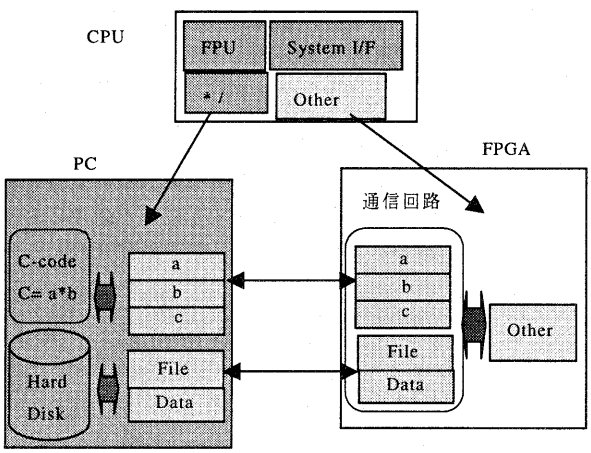

図 7　プロセッサ検証環境

5. おわりに

本稿では、PC 上のソフトウエアと FPGA の協調により少ない FPGA の個数で、SoC のシステム検証を行なう手法を提案した。提案手法は直接的な通信を行なう連携機構と、その欠点を補うハードウエア上のメモリ構造により、安価で環境構築が容易なシステム検証環境を実現した。特に FPGA と PC の通信回路の形態を変化させることで、PC と FPGA の役割をフレキシブルに対応させることができ、多様なソフトウエアモデリングの形態と検証環境を構築することができた。この提案手法を 3 つの SoC の設計検証プロジェクトに適用した結果、最大で 16MHz の動作周波数を最大 3 個の FPGA でシステム検証環境を実現することができ、RTL シミュレーションでは発見できなかったバグを発見するなどを実証した。今後は通信回路の汎用化と環境構築の容易化に関して研究を進める予定である。

文　献

[1] R. Lipsett, et al, "VHDL: Hardware Description and Design", Kluwer Academic Publishers, 1989.

[2] D. E. Thomas and P. Moorby, "The Verilog Hardware Description Language", Kluwer Academic Publishers, 1991.

[3] D. D. Gajski,,et al, "SPEC C: Specification Language and Methodology", Kluwer Academic Publishers, 2000.

[4] System C, http://www.systemc.org

[5] J. Gateley, M. Blatt, D. Chen, S. Cooke, P. Desai, M. Doreswamy, M. Elgood, G. Feierbach, T. Goldsbury, and D. Greenley, " UltarSparc-I Emulation", Proceedings of ACM/IEEE Design Automation Conference (DAC)95, pp.13-18, 1995.

[6] F. Casaubieilh, A. McIsaac, M. Benjamin, M. Bartley, F. Pogodalla, F. Rocheteau, M. Belhadj, J. Eggleton, G. Mas, G. Barrett, and C. Berthet, "Functional verification methodology of Chameleon processor", Proceedings of ACM/IEEE Design Automation Conference (DAC)96, pp.421-426, 1996.

[7] G. Ganapathy, R. Narayan, G. Jorden, D. Fernandez, M. Wang, and J. Nishimura, "Hardware emulation for functional verification of K5", Proceedings of ACM/IEEE Design Automation Conference (DAC)96, pp.315-318, 1996.

[8] J. Monaco, D. Holloway, and R. Raina, " Functional verification methodology for the PowerPC 604 microprocessor", Proceedings of ACM/IEEE Design Automation Conference (DAC)96, pp.319-324, 1996.

[9] N. Kim, H. Choi, S. Lee, S. Lee, I-C. Park, C-M. Kyung, "Virtual chip: making functional models work on real target systems", Proceedings of ACM/IEEE Design Automation Conference (DAC)98, pp.170-173, 1998.

[10] S. Cadambi, C. S. Mulpuri, P. N. Ashar "A Fast, Inexpensive and Scalable Hardware Acceleration Technique for Functional Simulation", Proceedings of ACM/IEEE Design Automation Conference (DAC), 2002

[11] M. Kudlugi, S. Hassoun, C. Selvidge, D. Pryor, "A Transaction-Based Unified Simulation/Emulation Architecture for Functional Verification", Proceedings of ACM/IEEE Design Automation Conference (DAC), pp. 623-628, 2001

[12] "PCI Local Bus Specification, Revision 2.1" PCISig, 1995

[13] Xilinx, http://www.xilinx.com

[14] Spec benchmarks, "http://www.spec.org"