4A-5

SCE-MI を利用した協調エミュレーション/シミュレーション環境の共通化手法

大山 将城 † 名野 響 † 近藤 信行 † 清水 尚彦 † 星野 民夫 ††

東海大学 †　　株式会社 アプリスター ††

1　はじめに

協調エミュレーション環境の一般的な問題として, ベンダツール独自の API(Application Programming Interface) を利用することによる移植性の低さがある.

SCE-MI は DUT(Device or Design Under Test) に対するテストベンチの設計の標準化を目的に, テストベンチ-DUT 間のインタフェース仕様として accellera 社より提案されている [1].

SCE-MI を利用することで,

- テストベンチの開発側は SCE-MI 定義の 6 つの API 仕様を理解するのみで良い
- DUT, テストベンチ間インタフェースは自動生成するのでユーザが開発する必要が無い

という利点がある.

我々は SCE-MI 仕様をもとに協調エミュレーション環境を独自の実装仕様で開発し，Linux と FPGA ボードを用いて実装した．さらに同一のテストベンチを協調シミュレーション環境で使用する手法について検討した. 本稿では開発した検証環境の詳細と実装による評価, 検証環境の共通化手法について述べる.

2　SCE-MI 概要

SCE-MI はユーザが設計したテストベンチと DUT とのインタフェース仕様である. SCE-MI API,Transactor と呼ばれるモジュール,PCI 等の物理インタフェースとそのドライバを総称して SCE-MI Infrastructure と呼ぶ. 我々は PCI インタフェースを物理インタフェースとして SCE-MI Infrastructure を開発した*.

テストベンチは SCE-MI Infrastructure を介して DUT にアクセスする．SCE-MI の概念図を図 1 に示す.

また,DUT のソースコードを解析し Transactor とテストベンチの初期化用パラメータファイルを出力するプログラムを Infrastructure Linker と呼ぶ.

3　Transactor の設計

送信用に 1024 ビットのレジスタを 1 本, 受信用に 1024 ビットのレジスタを 1 本, クロックとリセットの供給用モジュールとコントローラにより構成する (図 2)．送受信用の内部レジスタを 32 ビット毎に PCI のメモリ空間にマップし, レジスタを Transactor の入出力端子に接続する．これにより PCI のあるアドレス空間にアクセスすることで DUT の入出力ピンのアクセスが可能となる (図 3)．これを SCE-MI では MessagePort による Message の送受信という概念で扱う．Transactor の生成と PCI インタフェース,DUT との接続, マッピングに関しては後述する Infrastructure Linker が全て自動で行う.

表 1　ソフトウェアサイドのクラス仕様概要

クラス名	機能概要
SceMiEC	エラーハンドリング用クラス.
SceMi	SCE-MI Software Side の主要な処理を行うクラス
SceMiMessageInPortProxy	MessageInPort にアクセスする MessageInPortProxy を提供するクラス
SceMiMessageOutPortProxy	MessageOutPort にアクセスする MessageOutPortProxy を提供するクラス
SceMiParameters	パラメータファイルから初期化に使用するパラメータセットを生成するクラス
SceMiMessageData	送信 Message を保存するクラス

4　SCE-MI API の設計

設計したクラスの仕様を表 1 に示す．SCE-MI Software Side は 6 つのクラスによって, テストベンチに対してハードウェアサイドにアクセスするためのインタフェースを提供する.

5　Infrastructure Linker の設計

Infrastructure Linker は Transactor とソフトウェアサイドの初期化用パラメータファイルを DUT のソースコードより生成するプログラムである．Infrastructure Linker により SCE-MI API を使用したテストベンチと DUT のソースコードのみでエミュレーションが可能となる.

Infrastructure Linker は DUT の Verilog ソースコードを読み込み, クラスファイルで使用するパラメータファイルと DUT,PCI インタフェースを合成可能な Verilog コードとして生成する. 前処理部, 解析部, コード生成部の 3 部で構成する (図 4).

前処理部ではコメント部の削除, フィールドの分割など解析部が処理しやすい形にソースコードを変形する. 解析部ではユーザが指定した情報を元に DUT インタ

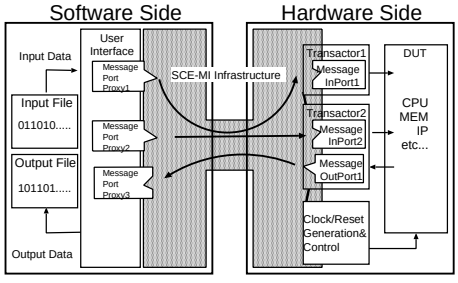

図 1　SCE-MI の概念図

The Sharing Method of Co-Emulaiton/Co-Simulation Environment featuring SCE-MI

Masashiro Ohyama, Hibiki Nano, Nobuyuki Kondoh, Naohiko Shimizu, Tamio Hoshino

†Tokai University

††Applistar Corporation

*PCI インタフェース/ドライバは研究室所有 IP を使用 [2]

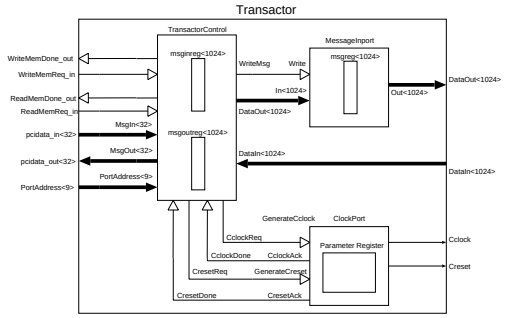

図 2　Transactor のブロック図

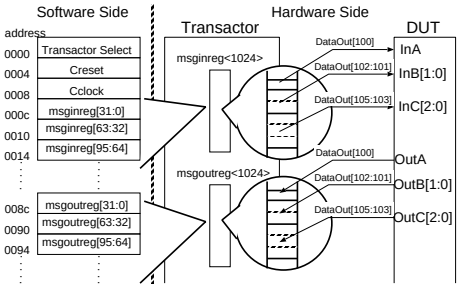

図 3　Transactor 内部のレジスタを PCI のメモリ空間,DUT の入出力ピンにマッピングした様子

フェース記述の解析を行う. コード生成部では解析部の出力したパラメータファイルを元に Transactor モジュールの生成を行う.

6　エミュレーションテスト

設計した SCE-MI に DUT とテストベンチを実装し Red Hat Linux 7.2 上でエミュレーションテストを行った. ターゲットデバイスを EP1S10F780C7ES, 論理合成ツールを QuartusII ver4.1 を用いた実装結果を表 2 に示す.

エミュレーションテストではテストベンチから DUT に対して 1 クロック毎にデータを書き込み，その時の FPGA ボード上の 7 セグ LED により動作を確認した. (図 5)

7　テストベンチの共通化手法

現在,SCE-MI API を使用したテストベンチの協調シミュレーション環境への共通化について検討している. これは論理合成可能な Verilog ソースコードを C++もしくは SystemC へと変換するプログラム Verilator [3] と連係することにより可能となる予定である.

この方法を図 6 に示す. 協調シミュレーションの場合は DUT ソースコードを Verilator に入力し,C++ソースコードに変換する. そして,DUT の C++ソースコードとテストベンチをコンパイルすることによりシミュレーション用の実行ファイルを生成できる. ただしこの場合, 図 6 の斜線で示す Infrastructure Linker からの SW SCE-MI Infrastructure 出力が必要である. SW SCE-MI Infrastructure は今回の実装では PCI インタフェースであった物理インタフェース部をソフトウェア的に実現するものである.

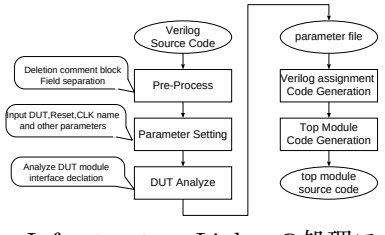

図 4　Infrastructure Linker の処理フロー

表 2　合成結果

Module Name	Logic Element
PCI-IF	395
Transactor	271
DUT	32

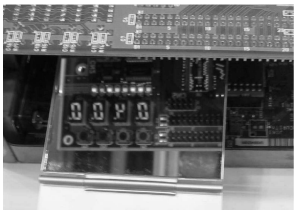

図 5　FPGA ボード上の 7 セグ LED 表示

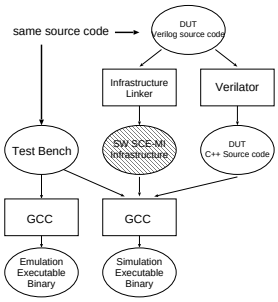

図 6　協調エミュレーション/シミュレーションにおけるテストベンチの共通化

8　まとめ

SCE-MI 仕様に沿ったハードウェア論理 (Transactor) の開発, クラスライブラリの開発を行った. また,Transactor モジュール, ソフトウェアサイド初期化用パラメータファイルを自動生成する Infrastructure Linker を開発した. そして, これらを使用して実際に FPGA でエミュレーションを試行し正常な動作を確認した.

今後は複雑な DUT/テストベンチを用いての動作検証,Infrastructure Linker の機能を拡張し同一の DUT/テストベンチで協調シミュレーションを実現する予定である.

参考文献

[1] accellera:”SCE-MI Reference Manual DRAFT”,2003

[2] 早坂, 志水, 横山, 孕石：”第 9 回 ASIC デザインコンテスト 規定課題 B PCI バスインタフェース” 第 22 回パルテノン研究会,2003

[3] 大山, 田中, 清水：”HDL コンバータ sfl2vl,Verilator を用いた C 言語ベースハードウェア・ソフトウェア協調シミュレーション環境” 組込みソフトウェアシンポジウム 2004 論文集,pp34-40,2004