オペレーティングシステムインターフェースを使ったプロセッサモデルのRTLシミュレーション手法

1K－4

伊藤義行＊　加藤哲＊＊　丸田善彦＊＊　本村真人＊　小長谷明彦＊
＊NEC　＊＊NEC情報システムズ

1　はじめに

複雑化するプロセッサの設計において、新規設計の過程や、設計のフィードバックの際などに、仕様策定用の機能(behavior)レベルの上位設計からネットリストレベルの下位設計まで、複数レベルの設計が混在した状態での動作検証/性能評価が必要となる場合がある。

また、プロセッサ設計の場合、装置全体としての動作/性能を検証するためのシミュレータへの入力パタンを与えるには周辺機器操作を含めたOSの動作が必要となる。

しかし、一般的には各レベルにおいて個別に仕様と入力パタンの設計を行って、シミュレーションするに留まる場合が多い。

本稿では、プロセッサ設計におけるシミュレーションにおいて、異なったレベルの混在と、OSのインターフェースをシミュレータ内に持たせることによって、パタン設計不要な統一的性能評価/検証を行なう手法を提案し、RTL記述によるSpecベンチマークの評価を行った。

2　マルチレベルシミュレーション

システムのレベルから回路記述のレベルまでが混在する場合、周辺機器やOSとのインターフェースを統一的環境で実現するためには、同一シミュレーション環境での、マルチレベルなシミュレーションが必要である[1]。

しかし、一般的な、記述レベル毎に分離した検証/評価の環境では、命令レベルの評価が終了してからでなければ、ハードウェアを意識した統合的な検証を行うことができず、未実装部分の混在を許さない。また、周辺機器のバスレベル仕様の設計が完了しなければ、外部からの入力パタンを与えることができない。

そこで、設計記述言語VHDL[2]が言語仕様上本来持っている、configuration節による記述の切り替え機能を記述レベル切り替えに生かし、実プログラム動作によるパタン生成と、設計レベル混在のプロセッサの統一的設計環境を構築する手法を提案する。

2.1　シミュレーションレベルの拡張

本手法のマルチレベルシミュレーションでは、プロセッサ記述のシミュレーション環境に、プロセッサの未実装部分を代行するソフトウェアと、部分的に先行してハードウェア化する混在させる(図1)。

ソフトウェアによる代行では、プロセッサ記述中の未実装部分の代わりにシミュレータとのインターフェースを設け、プロセッサ記述外部の代行ソフトウェアと協調した動作を行う。

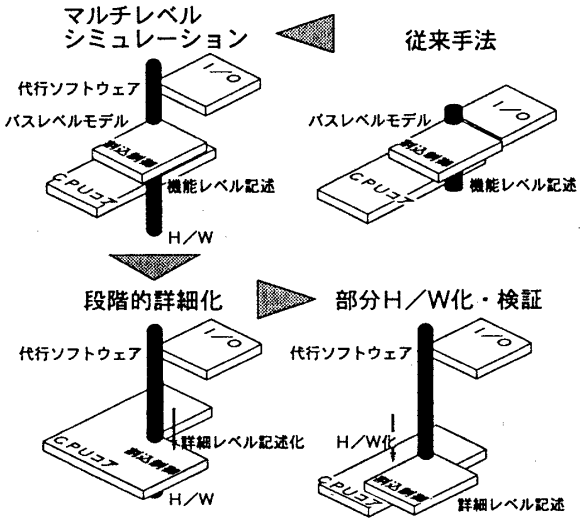

図1: 段階的詳細化に応じたシミュレーション

2.2　OS実行サポート

未実装部分を代行するソフトウェアとして、画面入出力やネットワークおよびディスク等の周辺機器操作を行う、割り込み処理をも含めたI/O処理を実現するOSインターフェースを実装した。

これにより、OSインターフェースを経由したシステムリソースの利用が実現され、Specベンチマークに代表される一般プログラムの実行を可能にした。

3　実現手法

OSインターフェースは、プロセッサハードウェア記述内には最小限のH/W→S/Wインターフェースを配置し、シミュレータ内部に置かれる代行ソフトウェア側でシステムコール等を実現する手法を取っている(図2)。

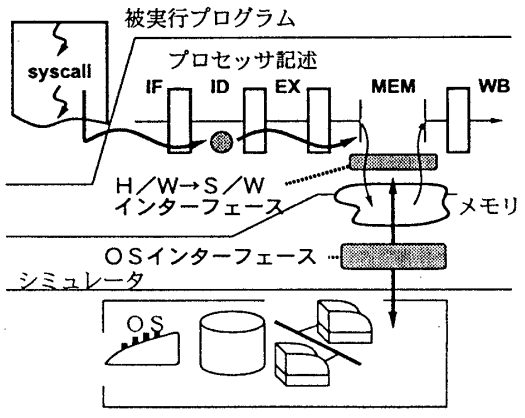

図2: システムコール実行環境

3.1　H/W→S/Wインターフェース

ハードウェア部分とOSは、レジスタやメモリを介して受け渡しが行われるため、H/W→S/Wインターフェー

A RTL simulation method of processor models by using operating system interface.
Y. Ito*, T. Katoh**, Y. Maruta**, M. Motomura*, A. Konagaya*
* NEC Corporation, ** NEC information systems inc.

スをプロセッサ記述のMEMステージに与える。その処理手順は以下のようになる。

データの引渡し プロセッサ処理を代行するソフトウェアへのデータ引渡しは、メモリへのデータ収納の記述を置き換える事で実現した。本来データバスとの入出力を担うMEMステージにH/W → S/Wインターフェースを設けた事で、プロセッサ記述への変更は、最小限に押えられている。

命令解釈 プロセッサ記述により実行がシミュレートされる被実行ソフトウェアプログラムに、システムコール実行命令が置かれていた場合、プロセッサ記述は命令をIDステージで認識し、MEMステージに存在するH/W → S/WインターフェースをOSインターフェースにシステムコールの実行を依頼する。

3.2 代行ソフトウェア

上記手法において、シミュレータ側でプロセッサ記述から依頼を受け、システムコールを実現するOSインターフェースは以下の2つからなる。

- 未実装処理代行ソフトウェアに汎用なインターフェース定義部
- インターフェース定義部から差し示されるアドレス変換等の個別処理

3.2.1 インターフェース定義部

ハードウェア記述言語によるプロセッサ記述側で実装されていない処理を代行する際、代行される処理と授受されるデータを、テーブルで定義する (図3)。

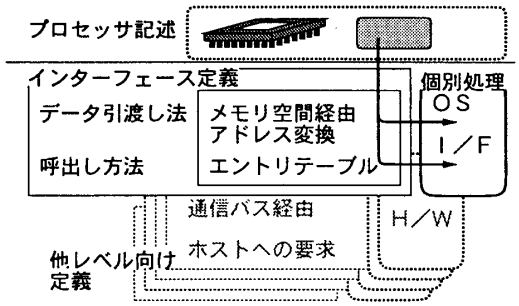

図 3: インターフェース定義部

3.2.2 データ変換

シミュレータが操作しているメモリアドレスは、OS側から与えられたメモリ空間であるため、シミュレートしているプロセッサ記述が想定しているメモリ空間と異なっており、アドレスやデータ型の変換規則を与える必要があるため、OSインターフェース内に変換関数を定義する (図4)。

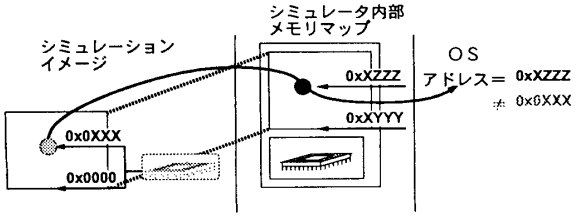

図 4: アドレス変換

以上の手法により、各設計レベルの混在と、アセンブラを直接入力できる、パタン設計不要な記述を得ることが可能となる。

4 評価結果

Specベンチマークの実行を目的として、MIPS社のR3000[3]をプロセッサコアとして、機能レベル、論理合成可能レベルでハードウェア設計を行い、前述の代行ソフトウエアとOSインターフェースを実現した。その仕様は以下のとおりである。

1. VHDL言語6K行の機能レベルプロセッサ記述
2. 浮動小数点およびバスアクセスはソフトウェアにより代行
3. R3000の全命令数71のうちbreakなどを除く58の命令を実現
4. 合成可能モデルとしてVHDL3K行を追加
5. 合成可能モデルの論理合成後の面積はゲートアレイで、49Kゲート、クロック周波数33MHz
6. 代行ソフトウエアはC言語で6K行

C言語インターフェースを持つVHDLシミュレータを使って、この記述にSpecInt92に含まれているアプリケーションのR3000用のプログラムを直接入力するシミュレーションを行った。MIPS R3000 CPU(30MHz)を用いているNEC EWS4800/220での実際の実行時間と比較した評価結果を以下に示す。

表 1: シミュレーション結果

	実機	シミュレーション			
		機能レベル		合成可能	
	sec	sec	実機比	sec	実機比
eqntott	0.04	952	24K	1585	40K
compress	0.04	316	8K	546	13K
espresso	0.09	10660	118K	17615	196K
li	0.21	43490	207K	71518	341K

このシミュレーションで、実装済命令のうち特殊な命令を除く50命令について正しく動作することを検証した。またシミュレーション速度の点では、実機に比較して、機能レベルでは平均で約90K倍、論理合成可能レベルでも機能レベルより1.7倍遅い程度で、平均約148K倍の時間で、入出力のシステムコールを含む一般プログラムのシミュレーションを実行できた。

5 おわりに

新規設計プロセッサの方式レベルでの検討段階で、システムの性能予測・検証用動作パタンを取得する方法として、マルチレベルシミュレーション手法を提案した。

その結果、OSシステムコール等に関わる部分をソフトウェアにより代行するOSインターフェースを開発し、R3000プロセッサコア相当の複数レベルによる記述でのSpecベンチマークシミュレーションをシステムレベルで実行し、評価検討を行った。

参考文献

[1] D.Patterson and J.Hennessy, "Computer Architecture — A Quantitative Approach", Morgan Kaufmann Publishers, 1990.

[2] R.Lipsett, C.Schaefer and C.Ussery, "VHDL: Hardware Description and Design", Kluwer Academic Publishers, 1989.

[3] G.Kane and J.Heinrich, "MIPS RISC Architecuture", Prentice Hall Inc., 1992.