LB-4

組込みソフトウェア開発を対象とした高速ハードウェアシミュレーション手法

秋葉 剛史† 野々垣 直浩† 深谷 哲司†
†株式会社東芝 研究開発センター システム技術ラボラトリー

1 はじめに

システムLSI開発を伴う組込み機器開発の多くは、ハードウェア(HW)実機完成後の開発終盤に、ソフトウェア(SW)とHWを結合した検証を行っている。この時点で、HWの不具合が発見されても、HWを再設計することは時間的/コスト的に困難であるため、多くの場合、SWによる対応処置が行われる。近年、この問題を解決するため、HW実機が完成する前にSW/HW間のシミュレーション(コシミュレーション)を行い、SW/HW双方の不具合を早期に発見することが重要であるとの認識が広まってきている[1][2]。しかし、従来のHW設計で用いるHWシミュレータをコシミュレーションに利用する場合、十分なシミュレーション速度が得られないという問題がある。

本稿では、SW/HWコシミュレーション向けHWシミュレーション高速化手法を提案する。提案手法は、機能等価性を保持したまま、コシミュレーションで必要とされる精度でHWのシミュレーションを行うことにより、効果的なシミュレーションを提供する。

2 ハードウェアシミュレータの問題点

本稿で述べるSW/HWコシミュレーションは、SW側の要求する精度と速度が満たされるものであるとする。SW側の要求する精度と速度とは、HWへの入力列に対する、出力列の正当性が保証されている(機能等価性)精度と、SW実行の妨げにならない程度の速度である。

HW設計用のHWシミュレータには主に、ゲートレベルシミュレータ、イベントドリブンシミュレータ、サイクルベースシミュレータがある。最も高速なサイクルベースシミュレータは、クロック単位でのレジスタの値が保証される精度である。複数のペリフェラル(約200Kゲート数)を回路対象とした場合、サイクルベースシミュレータは約160クロック/秒以下の処理速度であり、SW側が要求する速度である約20Kクロック/秒には及ばない。

High-Speed Processing Method of Hardware Simulation for Enbeded Software Development
† Takashi AKIBA (takashi.akiba@toshiba.co.jp)
† Nobuhiro NONOGAKI (nobuhiro.nonogaki@toshiba.co.jp)
† Tetsuji FUKAYA (tetsuji.fukaya@toshiba.co.jp)
System Engineering Laboratory,Corporate Research & Development Center,TOSHIBA Corporation (†)

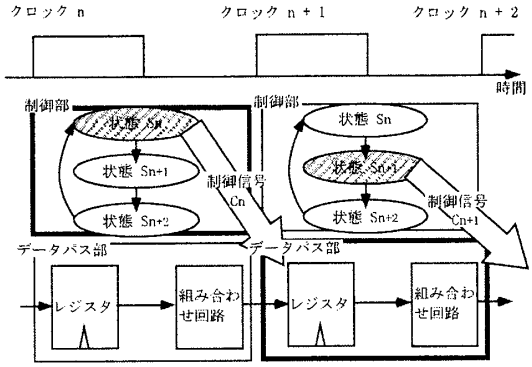

図1: HW回路の動作

3 提案ハードウェアシミュレーション高速化手法

提案手法は、HDL(Hardware Description Language)記述を入力として、機能等価性を保ったままC++に変換するものである。

提案手法では、HWシミュレーションの高速化のために、「データパス部の動作箇所特定」,「不要回路削除」を行い、C++に変換するために、「並行動作から逐次動作への変換」を行う。以下、それぞれを説明する。

3.1 データパス部の動作箇所特定

基本的なHDLは、回路の動作を制御する「制御部」と、実際に演算を行う「データパス部」で構成される。制御部は、時刻nのクロックで状態S_nと制御信号C_nを決定し、データパス部は時刻$n+1$のクロックで制御信号C_nに従った動作をする(図1)。そのため、各クロックごとの制御信号値をデータパス部に与えることで、その制御信号値で活性化するデータパス部の動作箇所を特定することが可能となる。

3.2 不要回路削除

データパス部の動作箇所を特定することで、図2に示すように、計算上不要な回路を検出できる。クロック単位で不要な回路の計算を省くことで、HWシミュレーションの高速化が可能となる。図2では、太線で囲まれた箇所のみの計算で、機能的に等価な演算結果を得ることができる。

3.3 並行動作から逐次動作への変換

HDLの構成要素であるレジスタや組み合わせ回路は、互いに並行動作をする。これを、逐次動作であるC++に変換するために、並行動作から逐次動作への

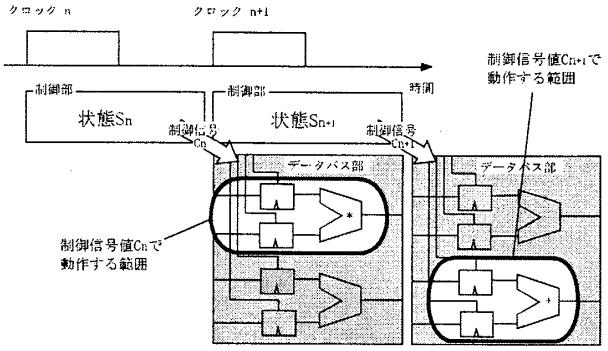

図 2: 不要回路削除

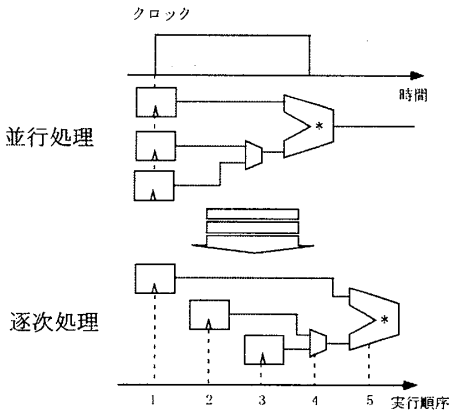

図 3: 並行動作を逐次動作に変換

変換を行う。レジスタは、他のレジスタとの入出力に依存関係が生じないように逐次的に並べる。組み合わせ回路は、他の組み合わせ回路と半順序関係が守られるように逐次的に並べる。図 3 には、演算結果が等価になるように、レジスタと組み合わせ回路を逐次的な実行順序に並べるイメージを示す。

4 高速化手法の評価と考察

提案手法を自動で行うツールを実装し、ツールが出力した C++によるシミュレーションと既存の HW シミュレータによるシミュレーションとの速度比較実験を、UltraSPRC-IIi(333MHz) ワークステーション上で行った。HW シミュレータは、サイクルベースシミュレータである、NC-Verilog ver.3.1(Cadence 社) を用いた。モチーフ回路として、検証済みの既設計回路である 4 入力 1 出力加乗算回路と 8 ビット CPU の Verilog-HDL を用いた。4 入力 1 出力加乗算回路への実験には 700,000 個の入力パタンを用い、8 ビット CPU への実験には 256 個のデータと、データをソートするアルゴリズムのメモリイメージを用いた。

提案手法を用いたシミュレーションの方法を以下に説明する。まず、ツールによって C++に変換された HW を、SW 側の C++から呼び出すように記述する。

表 1: シミュレーション速度比較実験結果

	NC-Verilog		Ours	
モチーフ回路	加減算	CPU	加減算	CPU
シミュレーション時間 [s]	38.62	64.66	0.93	1.50
NC-Verilog 比 [倍高速]	1	1	41.5	43.1

表 2: データパス部回路削除率

	加減算	CPU
平均削減率 [%]	88.6	81.7

次に、SW と HW の C++を共に C++コンパイラによってコンパイルする。最後に、コンパイル後に生成された実行ファイルを実行して、シミュレーションを行う。

シミュレーション速度の比較実験結果と、データパス部の回路削除率を、それぞれ表 1, 2 に示す。表 1 から、提案手法によって約 40 倍高速なシミュレーションが可能であることが分かる。複数のペリフェラルを回路対象にした場合、約 6.4K クロック/秒相当の処理速度が可能となり、SW 側が要求する速度である約 20K クロック/秒の約 1/3 を達成できたことが分かる。また表 2 から、不要回路削除が約 40 倍の高速化のうち約 5〜10 倍に寄与していることが分かる。また、表 1, 2 から、CPU の方が高速化率、削減率共に高いことが分かる。これは、CPU が制御部に状態数が多く、データパス部は状態ごとに大きく異なる動作をするため削減されやすいことが要因である。このことから提案手法は、回路の利用率が高く、不要となる回路が少ないコーデック系のデータ演算を中心とした回路よりも、通信プロトコル系などの制御系回路に対して、高い効果が得られると考えられる。

5 おわりに

提案した HW シミュレーション高速化手法により、SW 側が要求する速度の 1/3 の高速化が実現された。今後は、ビットレベルの動作箇所特定による、不要回路削除の性能向上などにより、さらなる高速化を目指す。また、実際の組み込みソフトウェア開発へのツール適用を行う。

参考文献

[1] K. Hines, and G. Borriello, "Optimizing Communication in Embedded System Co-simulation", Proc. Int. Workshop on Hardware-Software Codesign, pp. 121-125, Mar. 1997.

[2] K. Lahiri, A. Raghunathan, and S. Dey, "System level performance analysis for designing on-chip communication architectures", IEEE Trans. on Computer Aided-Design of Integrated Circuits and Systems, vol. 20, no. 6, pp. 768-783, Jun. 2001.