Vol.2015-SLDM-169 No.10
2015/1/29

マルチコアを用いた 高速並列論理シミュレーション手法

竹内勇矢　豊永昌彦　村岡道明
高知大学大学院　理学専攻　情報科学分野
〒780-8520 高知県高知市曙町 2-5-1
E-mail: { ytakeuch, toyonaga, muraoka } @ is.kochi-u.ac.jp

概要　本研究では, マルチコアプロセッサを使用した並列論理シミュレーションアルゴリズムを提案し，高速な並列シミュレータを開発した．本並列アルゴリズムは，論理回路をファンアウトコーンを用いた並列な回路に変換し, これらをマルチコアプロセッサにより並列処理を行なう．並列シミュレーションアルゴリズムの性能見積りを行なったところ，商用シミュレータと比較して，組み合わせ回路では 24 倍，順序回路では 27 倍以上の高速化率が得られる見通しを得た．

キーワード：マルチコアプロセッサ，論理シミュレーション，並列処理，並列アルゴリズム

A High-Speed parallel Logic Simulation Method Using Multi-core Processor

YUYA TAKEUCHI　MASAHIKO TOYONAGA
MICHIAKI MURAOKA
Information Science Division, Graduate School of Science, Kochi University
2-5-1 Akebono－Cho, Kochi, 780-8520 Japan

Abstract: In this paper, a parallel logic simulation algorithm using the multi-core processor is proposed, and a parallel logic simulator based on the algorithm is developed. The parallel algorithm converts the logic circuit to the fan-out cones, which operate in parallel and then are simulated in parallel on multi-core processors. The performance of the parallel simulation algorithm is estimated to be 24 times for combinational circuits and 27 times for sequential circuits faster than that of a high speed commercial simulator.

Keywords: Multi-core Processor, Logic Simulation, Parallel Processing, Parallel Algorithm

1　はじめに

近年，ソフトウェア/ハードウェアの協調シミュレータによる回路検証に膨大な時間を必要としている．現在，協調シミュレータの代わりとして FPGA を用いた論理エミュレータにより高速シミュレーションを行うが，大規模回路に対して FPGA の書き込みに膨大な時間がかかるため，デバック性が良くない．この問題を解決するための高速な論理回路検証法として, GP-GPU[7]や FPGA[8]を利用した並列処理による高速論理シミュレーションの研究が行われている．

本研究ではマルチコアプロセッサを用いたプログラムの並列化による高速論理シミュレーションを目指す．本研究室では関連する先行研究としてソフトウェアの実行時間を考慮したビヘイビア（SW）の並列化手法を提案している[4][5]．本稿では[4，5]の並列化手法に加え，効率よい並列シミュレーション手法と高速演算手法を提案する．また，本研究で提案する高速化手法を適応した高速論理シミュレーションの評価を行い，市販の高速シミュレータと比較した．

本稿の構成は以下の通りである．2 章では論理シミュレーションアルゴリズムについて説明する．3 章ではマルチコアプロセッサを用いた効率の良い並列シミュレーション手法について説明し, 4 章ではさらなる高速演算手法の提案を行う．5 章では並列シミュレーション時間の見積もり手法について述べ, 6 章では開発した高速論理シミュレーションの評価結果を示す．最後に本稿のまとめと今後の課題について述べる．

2　論理シミュレーションアルゴリズム

論理シミュレーションアルゴリズムには主にイベント・ドリブン法とレベルソート法の 2 通りが存在する．本章ではこの 2 つの手法について述べる．

2.1 イベント・ドリブン法

現在，広く普及している論理シミュレーションアルゴリズムはイベント・ドリブン法である．イベント・ドリブン法は信号の変化(イベント)のある論理素子に着目し，イベントが発生した論理素子と次段の論理素子を演算する手法でイベントが発生しない論理素子の演算を省くことができる．しかしながら，この手法はイベント管理や伝搬遅延時間を考慮する必要があるため，並列化には不向きな手法である．

2.2 レベルソート法

レベルソート法とはイベントに関係なく全論理素子を順次に演算する手法である．イベント・ドリブン法と異なり並列化に不向きなイベント管理や伝搬遅延時間を考慮する必要がないため，並列化向きの手法である．そのため，本研究ではアルゴリズムが簡単で並列化に向くレベルソート法を対象とした．

図 1 はレベルソート法の処理手順を示す．四角が論理素子，線が配線，太線が論理素子の演算順序をそれぞれ示す．レベルソート法は以下の 5 ステップにより構成される．

(1) はじめに論理素子の論理段数を求める．
(2) 外部入力端子にテストベクタを設定する．
(3) 2 段目の論理素子の演算を行う．
(4) 次段の論理素子の演算を行う．
(5) (4)の処理を論理段数分行うことで外部出力端子の信号値を得る．

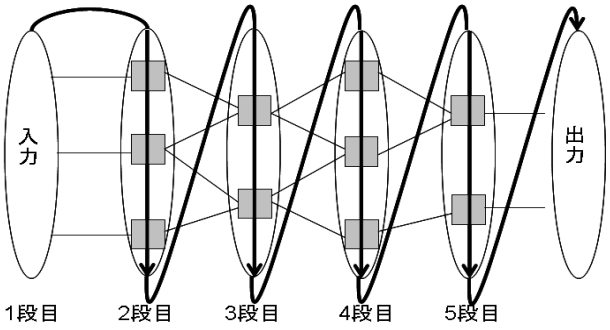

図 1 レベルソート法の演算手順

3 並列シミュレーション手法

本研究では[4, 5]で提案されている並列化手法を論理シミュレーションに適応し，マルチコアプロセッサを用いて論理演算プログラムを並列化して高速な並列論理シミュレーションを目指した．論理演算の並列化には並列演算プログラムに回路データを割り当てる必要がある．しかしながら，回路データを共有または単純に分割して割り当てただけではデータ通信が発生し，並列論理演算による高速化は難しい．よりよい並列化の効果を得るにはデータ通信が発生しない回路表現が必要であり，データ通信が発生しない回路表現としてファンアウトコーンや AND/OR[9]プレーンが考えられる．本章ではファンアウトコーンを用いた並列シミュレーション手法を提案する．

3.1 ファンアウトコーン

ファンアウトコーンとは論理回路において，外部出力端子やフリップフロップ(FF)を始点とし，始点となる論理素子の信号値に影響する論理素子を切り出した部分回路のことである．外部出力端子と FF を始点とし，外部入力端子と FF を終点とする．図 2 ではファンアウトコーンの作成例を示す．

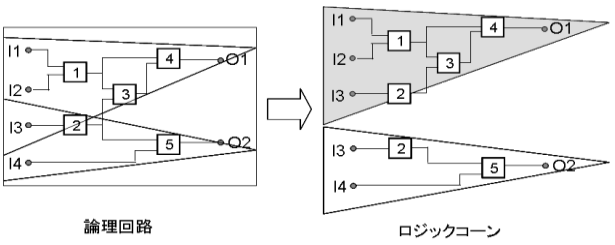

図 2　ファンアウトコーンの作成例

図 2 で示す論理回路は外部出力端子 O1，O2 を始点とする 2 つのファンアウトコーンが作成される．O1 のコーンでは O1 から外部入力端子までにたどる論理素子 1，2，3，4 と終点である外部入力端子 I1, I2, I3 が含まれる．これらのファンアウトコーンはコーン間でデータ通信が発生しないため，各ファンアウトコーンをデータ通信が発生せず並列に演算することが可能である．

作成したファンアウトコーンを並列演算プログラムに割り当て並列に演算を行うが，一般的な回路で生成されるファンアウトコーンの数はマルチコアに含まれるプロセッサの個数よりも多くなる．すべてのファンアウトコーンを並列に演算するには，ファンアウトコーンのグループ化を行いプロセッサ数以下のコーングループを作成する必要がある．次節にてコーングループの作成方法について説明する．

3.2 コーングループの作成方法

コーングループは複数のファンアウトコーンを 1 つのグループにまとめたものである．グループ化の際にファンアウトコーンの演算時間を考慮せず作成した場合，コーングループの演算時間にばらつきが生じ，演算時間が長いコーングループが作成されてしまう．並列演算プログラムの実行時間は演算時間が最も長いコーングループに依存するため，各コーングループにかかる演算時間が均等になるように作成する必要がある．演算時間はコーングループに含まれる論理素子数に依存するので論理素子数が均等になるようにファンアウトコーンをグループ化することにより演算時間が均等になる．

コーングループの作成手順を以下に説明する．図 3

ではコーングループの作成手順を示す.
(1) ファンアウトコーンを解析し，各ファンアウトコーンに含まれる論理素子数を求める.
(2) 求めた論理素子数を対象にファンアウトコーンを降順にソートする.
(3) 論理素子が多いファンアウトコーンからコーングループに割り付ける．割り当てるコーングループは最も論理素子数が少ないものを選択する.
(4) (3)の処理を生成されたファンアウトコーン個数回行う.

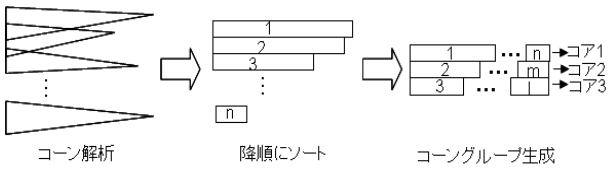

図 3 コーングループ作成

作成したコーングループは並列演算プログラムに割り当て演算される．次節では提案する並列論理シミュレーションアルゴリズムの構成について述べる.

3.3 並列論理シミュレーションの構成

図 4 では本研究で提案する並列論理シミュレーションの構成を示す．点線は回路データと論理演算の対応を示し，太線はプロセッサと論理演算の対応を示す．並列アルゴリズム部分は並列論理シミュレーションアルゴリズム，ハードウェア部分はマルチコアプロセッサ, データ構造部分は 3.2 節で説明したコーングループをそれぞれ示す.

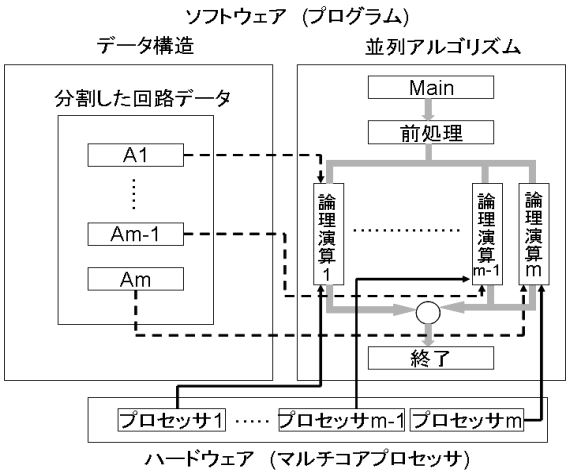

図 4 並列論理シミュレーションの構成図

並列論理シミュレーションの処理手順を説明する．はじめに論理演算を行うための前処理を行い，次に分割した回路を並列に演算され，演算終了後に演算結果が同期されてシミュレーションが終了する．この並列論理演算の並列数はマルチコアに含まれるプロセッサ数に合わせ, 1 対 1 の割り当てになるように構成する．そのため，コーングループの個数も並列す分用意する．作成した回路データは論理演算に割り当て，論理演算はプロセッサにより並列に処理を行い割り当てられた回路データの演算を行う．以上が本研究で提案する並列論理シミュレーションの構成となる．次章でさらなる高速化を目指した高速演算手法について説明する.

4 並列シミュレーションの高速化手法

5 章で説明した並列シミュレーション手法により効率の良い並列論理シミュレーションを開発できた．本章では，さらに高速な演算が可能な論理素子の複合化について述べる.

4.1 論理素子の複合化

論理素子の複合化とは複数の論理素子を 1 つのマクロとして表現し，ルックアップテーブル(LUT)方式に置き換える．複数の論理素子を 1 つのマクロで表現するため，論理素子数と論理演算回数が削減され，演算時間の短縮が可能である．本節では論理素子の複合化方法としてレベル間の複合化とレベル上の複合化の 2 種類を提案する.

1) レベル間の複合化

レベル間の複合化はレベルが異なる論理素子同士を複合化する手法であり，対象となる論理素子とその F/O 先の論理素子を複合化する．生成したマクロの番号及びレベルは F/O 先の論理素子と同様とする．なお, F/O 先の論理素子数が 2 つ以上のとき複合化を行わない.

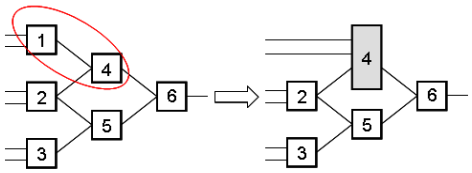

図 5 レベル間の複合化例

図 5 はレベル間の複合化の例を示す． 論理素子 1 とその F/O 先である論理素子 4 の複合化する．複合化して作成されたマクロは入力数は 3 となり，マクロの番号とレベルは論理素子 4 と同様になる. また，論理素子 2 の複合化は F/O 先の論理素子が 2 つ存在するため，複合化ができない．論理素子 2，4 を複合化した場合，論理素子 5 の F/I 先である論理素子 2 の

が複合化され，信号値が求まらないため演算することができない．

2) レベル上の複合化

レベル上の複合化はレベルが同じ論理素子同士を複合化する手法である．図 6 ではレベル上の複合化の例を示す．

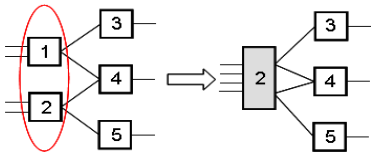

図 6 レベル上の複合華麗

論理素子 1，2 は同じレベルなため，複合化が可能である．複合化により生成されるマクロの入力数は 4 本，出力数は 2 本となる．

4.2 最適入出力数の検討

4.1 節で述べた論理素子の複合化は入出力数が多いほど論理素子の数を削減できるが，マクロ 1 つのアクセス回数が増加する．本節では入出力数を 5～10 本の 6 パターンを評価し，実行時間が最も短縮可能な最適な入出力数を求めた．表 1 では複合化により作成されるマクロの総数を示し，表 2 ではマクロを用いた実行時間を示す．GATE は複合化を行っていない回路を示す．評価に使用する回路は順序回路の cpu と組み合わせ回路の ldpc_encoder あり，基本情報は 6.2 節に記載する．

表 1 複合化により生成されたマクロ数

回路名	論理素子数						
	GATE	LUT					
		入出力数=5	入出力数=6	入出力数=7	入出力数=8	入出力数=9	入出力数=10
cpu	2,111	573	509	465	312	286	266
ldpc_encoder	75,035	23,823	18,614	15,903	13,098	12,056	11,032

表 2 複合化を適応した実行時間

回路名	実行時間[sec]						
	GATE	LUT					
		入出力数=5	入出力数=6	入出力数=7	入出力数=8	入出力数=9	入出力数=10
cpu	1.76	1.05	1.05	1.05	0.84	0.84	0.97
ldpc_encoder	64.13	61.31	48.58	44.26	39.48	38.67	39.97

表 1 から入出力本数が多いほど論理素子数を削減できていることがわかる．最も削減できているのが入出力数 10 本のときであり，順序回路である cpu が 87.4%，組み合わせ回路である ldpc_encoder が 85.3% 削減できている．表 2 では入力本数が 9 本のとき，最もシミュレーション時間が速くなり，順序回路である cpu と組み合わせ回路である ldpc_encoder はともに約 2 倍の高速化となった．本研究では入出力本数を 9 本に設定して複合化を行った．

4.3 複合化方法

論理素子の複合化は複数の論理素子を 1 つのマクロで表現するため，複合化を行う前の回路と行った後の回路は異なり，同様に作成されるファンアウトコーンも異なる．本節では複合化を行う前にコーンを作成したものと複合化を行った後にコーンを作成したものを評価し，比較を行った．表 3 では前述した 2 つのアルゴリズムにより作成されるマクロの総数を示す．評価に用いる回路は 4.2 節のものと同様である．生成されるマクロ数が少ないほど高速に演算が可能である．

表 3 からファンアウトコーンを作成したから複合化を行った方が生成されるマクロの数が少ない結果となっている．本アルゴリズムでは，論理回路をファンアウトコーンに変換した後に複合化を適応する．

表 3 論理素子総数の比較

回路名	マクロの総数	
	コーン→複合化	複合化→コーン
cpu	2,688	5,094
ldpc_encoder	28,926	143,374

5 並列シミュレーション時間の見積もり手法

本研究では，対象となる並列アルゴリズムの CPU 上の実行時間を正確に見積もるために JAXA(Japan Aerospace Exploration Agency:宇宙航空研究開発機構)が開発した ELEGANT システム[1]の中のシミュレータ Visual Spec を使用する．このツールにより生成される時間精度付きモデル[3]をプロファイリングすることで CPU 上の実行時間を見積もることができる．時間精度付きモデルとは対象となるアルゴリズムに基本ブロック(分岐を含まない手続き的な記述部分)の実行時間を付与したものである．C 言語では時間を表現することができないため，SpecC 記述言語[2]を用いて記述される．

PC に搭載されるマルチコアを用いて並列シミュレーション時間を測定することは可能だが，アプリケーションから並列数と使用するコアを制御することができない．Visual Spec では並列数と使用するコアを制御でき，正確な並列実行時間を見積もることができる．

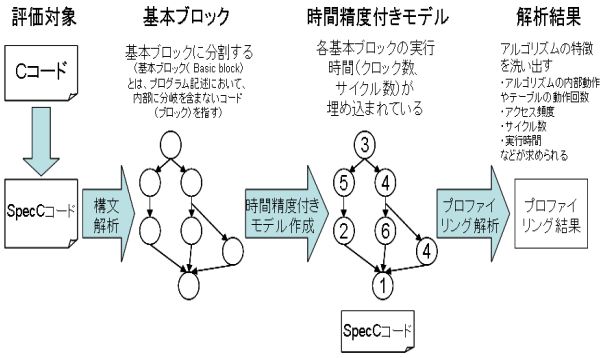

図 6 実行時間の見積もり手法

図 6 は Visual Spec を用いた実行時間の見積もり手法を示す．実行時間の見積もり手法は以下の 4 ステップにより構成される．

(1) 対象となる C 言語で記述されたアルゴリズムを SpecC 記述言語に変換する．

(2) SpecC 記述言語に変換したアルゴリズムを Visual Spec を用いて構文解析を行い，基本ブロックに分割する．

(3) 各基本ブロックの実行時間を算出し，アルゴリズムに付与する．(時間精度付きモデルの生成)

(4) 生成した時間精度付きモデルのプロファイリングを行い，アルゴリズムの実行時間を算出する．

本研究では Visual Spec を用いて並列アルゴリズムの解析を行い，並列シミュレーション時間を見積もり評価する．

6 評価

本章では，本研究で提案した並列化アルゴリズムと高速演算手法を適応した並列論理シミュレーションの評価を行う．

6.1 評価環境

並列論理シミュレーションの並列実行時間と CPU 実行時間を比較した．並列実行時間の計測には 5 章で述べた見積もり手法を用いる．実験環境は以下の通りである．

1) 並列実行

シミュレータ：JAXA-Elegant/Visual Spec(version4.1.6)
ターゲット CPU：ARM946E-S
コンパイラ：arm-elf-gcc
クロック周波数：200MHz
テストパターン：10,000 サイクル

2) PC

CPU：CPU:Intel®core™i7
RAM：2.99GB
クロック周波数：3.07GHz
シミュレータ：ModelSim SE 10.1c(VDEC 提供)
テストパターン：100,000 サイクル

6.2 実験内容及び評価用回路

シーケンシャルな論理シミュレーションと並列論理シミュレーションを見積もり手法を用いて実行時間を算出し，比較を行って並列化による高速化率を求める．また，見積もり手法で求めた実行時間を用いて市販の高速シミュレータとの比較も行った．

評価用回路として組み合わせ回路と順序回路の 2 種類を用いる．組み合わせ回路では低密度パリティビットエンコーダである ldpc_encoder を，順序回路では 8bit-CPU を 1，16，32 個を並列に並べた 3 種類の回路を用いた．表 4 は評価用回路の基本情報を記載している．

テストベクタは組み合わせ回路がランダムで生成されたものを，順序回路である 8bit-CPU はロード，ストア，加算の命令を行ったものを用いた．

表 4 評価用回路の一覧

回路名	インスタンス数	入力数	出力数	FF数	ゲート数	論理段数
cpu	2,148	19	18	173	2,111	56
cpux8	17,051	19	144	1,384	16,888	56
cpux16	34,083	19	288	2,768	33,776	56
ldpc_encoder	78,806	1,723	2,048	0	75,035	12

6.3 評価結果

本節で評価を行うシミュレータを以下に示す．
hostsim(a)：シーケンシャルな論理シミュレーション
並列 sim(b)：並列論理シミュレーション(16 並列)
商用 sim(c)：高速な市販シミュレータ

1) 並列化による高速化

表 5 は Visual Spec を用いて見積もった実行時間をを示す．シーケンシャルな論理シミュレーションである hostsim(a)と開発した並列論理シミュレーションである並列 sim(b)の比較結果を示し，各回路の実行時間[sec]と速度比率(a/b)を示す．

表 5 からシーケンシャルな hostsim(a)と比較して並列 sim(b)は順序回路で約 2～10 倍，組み合わせ回路で 9.42 倍の高速化率を得られた．また，中でも cpux16 の高速化率が 10.16 倍となり，一桁以上の高速化率を得られた．この結果から，本並列シミュレーション手法と高速演算手法により順序回路，組み合わせ回路ともに実行時間を高速化することがわかった．

表 5 並列シミュレータとの比較

サイクル数:10k	実行時間[sec]		速度比(a/b)
回路名	hostsim(a)	並列sim(b)	
cpu	2.72	1.28	2.13
cpux8	21.57	2.67	8.08
cpux16	43.16	4.25	10.16
ldpc_encoder	96.29	10.22	9.42

2) 商用シミュレータとの比較

表 6 は並列論理シミュレーションを PC 環境で実行したときの推定実行時間を示す．PC 環境では並列論理シミュレーションの正確な実行時間を測定できないため，PC 環境でシーケンシャルな論理シミュレーションの時間(d)を測定し，表 5 の速度比(e)を考慮して算出した値(d/e)となる. 表 7 では市販の高速シミュレータである商用 sim(c)と並列 sim(b)の比較結果を示し，各回路の実行時間[sec]と速度比率(c/b)を示す.

表 7 から商用 sim(c)と比較して並列 sim(b)は順序回路で約 1～27 倍, 組み合わせ回路で 24.62 倍の高速化率を得られた．並列比較と同様に cpux16 において高速化率が 27.81 倍と最も高い高速化率が得られた．この結果から，並列比較と同様に順序回路，組み合わせ回路ともに市販の高速なシミュレータよりも高速であることがわかった.

表 6 並列論理シミュレーションの推定実行時間

サイクル数:100k	並列sim(b)		
回路名	実行時間[sec](d)	速度比(e)	b = d / e
cpu	0.42	2.13	0.20
cpux8	3.54	8.08	0.44
cpux16	7.13	10.16	0.70
ldpc_encoder	17.14	9.42	1.82

表 7 商用シミュレータとの比較

サイクル数:100k	実行時間時間[sec]		速度比(c / b)
回路名	商用sim(c)	並列sim(b)*1	
cpu	1.22	0.20	6.16
cpux8	9.76	0.44	22.27
cpux16	19.52	0.70	27.81
ldpc_encoder	44.8	1.82	24.62

*1: 並列 sim の実行時間は表 6 で求めた時間

本評価にて提案する並列理シミュレーションは市販の高速シミュレータと比較して高速にシミュレーション可能であることがわかった.

7 まとめと今後の課題

7.1 まとめ

本研究では，[4，5]で提案されたマルチコアプロセッサを用いた並列化アルゴリズムと本稿で提案する並列シミュレーション手法及び高速演算手法に適応した高速な並列論理シミュレーションを開発した.

開発した並列論理シミュレーションを Visual Spec を用いて評価を行いシーケンシャルな論理シミュレーションと比較した結果, 組み合わせ回路で約 9 倍, 順序回路で最大 10 倍となる高速化率を得られた．また，商用シミュレータとの比較も行った．組み合わせ回路では 24 倍，順序回路では最大 27 倍の高速化率を得られた．また，本評価から並列シミュレーション手法と高速演算手法により商用シミュレータよりも高速にシミュレーションすることがわかった.

7.2 今後の課題

今後の課題として，本並列シミュレーション手法及び高速演算手法や並列数の増加による高速化率の評価が必要である．将来の展開としては，大規模な回路やシステムの検証をスーパーコンピュータ(例. 京など)で高速で行うことができると考える.

謝辞

本研究は東京大学大規模集積システム設計教育研究センターを通し，メンター・グラフィックス・ジャパン株式会社の協力で行われたものである.

参考文献

[1] 荒木大，“ELEGANT の SpecC シミュレーションと設計詳細化について”, 第 3 回先端宇宙情報技術ワークショップ前刷講演集, pp.1-13, 2007 10 月

[2] Andreas Gerstlauer, Rainer Dömer, Junyu Peng, Daniel D. Gajski, “システム設計：SpecC による実現”, 2001 年

[3] Michiaki Muraoka, Noroyoshi Itoh, Rafael K. Morizawa, Hiroyuki Yamashita, Takao Shinsha, “Software Execution Time Back-annotation Method for High Speed Hardware-Software Co-simulation”, Proc. Of SASIMI2004, pp. 169-175, October 2004

[4] 松永惇弥, 村岡道明, “タイミングを考慮したハードウェア/ソフトウェア分割手法の評価”, デザインガイア 2009, pp31-36 2009 年 12 月

[5] とう文竹, 竹内勇矢, 村岡昌彦, 村岡道明, "マルチコアプロセッサを用いた並列論理シミュレーション手法", デザインガイア 2013, 2013 年 11 月

[6] 大菊祥子, 橋口拓哉, 豊永昌彦, 村岡道明, ”GP-GPU を用いた並列論理シミュレーションアルゴリズムの評価”, DA シンポジウム 2012 論文集, pp.109-114, 2012 年 8 月 29 日

[7] 橋口拓哉, 豊永雅彦, 村岡道明, "GP-GPU を用いた高速並列論理シミュレーション手法", ETNET2014, No.19 2014 年 3 月

[8] 松本夏樹, 村岡道明, "FPGA を用いた論理シミュレーション手法", デザインガイア 2013, 2013 年 11 月

[9] 竹内勇矢, トウブンチク, 村岡道明 ”並列化アルゴリズムによる論理シミュレーションの高速化手法の提案”, DA シンポジウム 2013 論文集, pp.91-96, 2013 年 8 月 22 日