Vol.2013-SLDM-163 No.11
2013/11/27

FPGA を用いた論理シミュレーション手法

松本　夏樹　　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: {matsumot, muraoka}@is.kochi-u.ac.jp

あらまし　本稿では，レベルソート法を用いた論理回路シミュレーションのハードウェアアルゴリズムを提案する．アルゴリズムを高速化するために，そのハードウェアアルゴリズムの論理演算部の並列化を行う．これにより，論理回路内の論理段（レベル）上の論理ゲートの論理演算の並列化が行われ高速化される．本並列論理シミュレーションアルゴリズムを FPGA へ実装した場合についてタイミングシミュレーションで性能評価を行った結果，商用論理シミュレータと比較して 3 万ゲートで約 10 倍の高速性を達成できる見通しを得た．また，大規模回路に適応した場合の高速性について推定を行うと，100 万ゲートで商用シミュレータと比較して約 20 倍の高速性が達成可能であると見込まれる．

キーワード　FPGA，論理シミュレーション，並列処理，並列アルゴリズム, LSI

A Logic Simulation Method using FPGA

Natsuki Matsumoto　　Michiaki Muraoka

Information Science Division, Graduate School of Science, Kochi University
2-5-1 Akebono-cho, Kochi 780-8520 Japan
E-mail: {matsumot, muraoka}@is. kochi-u. ac. jp

Abstract　In this paper, a hardware algorithm of the logic simulation using the leveled method is proposed. To speed up the algorithm, the logic evaluation part of the hardware algorithm is parallelized. Then this accelerates the evaluation speed of the logic gates on the logic level of the logic circuit by the parallelization. The parallel logic simulation algorithm was implemented in the FPGA, and the performance of the FPGA was evaluated. As the result of the evaluation of the timing simulation, the speed of the proposed hardware algorithm achieved approximately 10 times faster than that of a fast commercial logic simulator using 30,000 gates. In addition, when it is applied to larger scale circuits such as one million gate circuits, it is estimated to be 20 times faster respectively.

Keyword　FPGA,　Logic Simulation, Parallel Processing, Parallel Algorithm, LSI

1. はじめに

論理回路の論理シミュレーションは，従来ではソフトウェアシミュレータが用いられてきた．しかし，ソフトウェアシミュレータでは大規模論理回路の論理シミュレーションにおいて，膨大なシミュレーション時間が必要とされることより，シミュレーション時間の短縮が望まれている．そのため，ソフトウェアシミュレータの並列化による高速化が考えられているが，シミュレーションアルゴリズムの並列化は難しいという問題があった．また論理シミュレータのハードウェア化[1]も行われていたが，拡張性やバージョンアップが難しいため広く普及しなかった．現在では，FPGA を用いた論理エミュレータによる高速シミュレーションが行われているが，大規模回路については FPGA への書込みに膨大な時間がかかり，デバッグ性がよくない．そのため，回路のバグがほぼなくなった最終段階で，ソフトウェアを含めた大規模なシミュレーションに使用されることが多い．それらの問題を解決する方法として，最近では，マルチコア[2][3]や GP-GPU[4][5][6]を利用した並列処理による論理シミュレーションの高速化などの研究も行われている．

本研究では，レベルソート法を用いたデバッグ性の高い高速な論理回路シミュレーションのハードウェアアルゴリズムの提案，及びそのハードウェアアルゴリズムの FPGA への実装を検討した．また，アルゴリズムの高速化方法として，そのハードウェアアルゴリズムの論理演算部の並列化を行った．

以後の章では，2 章で論理シミュレーションアルゴリズムについて説明し，3 章で提案するハードウェアアルゴリ

ズム及び論理演算部の並列化による高速化手法について説明する. 4 章で FPGA に実装するための制約について説明する. 5 章で並列化なしと並列化した論理シミュレーションアルゴリズムを FPGA に実装した場合のタイミングシミュレーションで性能評価を行った. また, 並列化した場合の FPGA と商用論理シミュレータと比較した結果を示す. 最後に 6 章で本研究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

2.1 論理シミュレーションについて

論理シミュレータとは, 論理回路が正しく動作するかを検証するためのツールである. 検証を行う論理回路に入力パターン（テストベクタ）を与えて, 回路の論理ゲートの動作に基づいて論理演算を行い, 出力を得る. 設計時に想定していた結果とシミュレーションの出力結果を比較し, 回路が正しく動作しているかを検証する.

2.2 論理シミュレーションについて

現在, 広く普及している論理シミュレータはイベント・ドリブン法が用いられている. イベント・ドリブン法とは, 入力信号の変化（イベント）のある論理ゲートに着目し, イベントが発生した論理ゲート及びそれが伝搬する論理ゲートを演算する手法である. この手法は正確なタイミングに基づく検証を実行することができ, 一般の論理回路においてイベント発生率は 10%以下であるため論理演算回数を最小限にすることができる. しかし, イベント管理や遅延時間を考慮する必要があるため, 並列アルゴリズムの実現は容易ではない. そのため, 本研究ではアルゴリズムが簡単で並列化が容易であるレベルソート法（レベライズド法とも呼ぶ）を採用することとした. 図 1 にて, レベルソート法の処理手順を示す. 四角が論理ゲート, 点線が処理手順を示している.

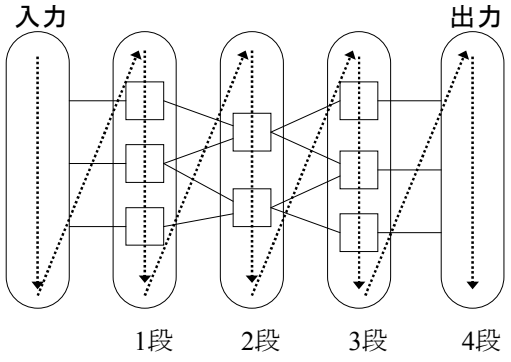

図 1 レベルソート法の処理手順

レベルソート法とは, 論理回路を入力端子から順に論理段に分け, 入力端子にテストベクタを設定し, 入力端子から順に各段上の論理ゲートの演算を行い, 出力端子の信号値を決定する方法である. 図 1 では, イベントの発生に関わらず 1 段目から出力まで段数順にすべての論理ゲートの演算を行うことにより出力端子の信号値を決定する. 本演算手法は, イベント管理や遅延時間を考慮せず演算することができるため, 並列化が容易である.

3. 論理シミュレーションアルゴリズムのハードウェア化

3.1 ハードウェアアルゴリズム

今回レベルソート法を用いた論理シミュレーションアルゴリズムのハードウェア化を行った. ソフトウェアでは難しい並列化をハードウェアでは用意に実現することができるためハードウェア化による高速化が期待できる.

今回提案する論理シミュレーションのハードウェアアルゴリズムを以下に示す.

(1) ホスト(PC)側で作成されたハードウェア用のネットリストテーブル, テストベクタ等のデータを FPGA で受信する.
(2) ネットリストテーブルは onchip SRAM(BRAM), テストベクタは offchip SRAM に格納する.
(3) テストベクタを最初のテストパターンから順に読み込み, 入力端子に設定する.
(4) すべての FF の値のアップデートを行う.
(5) 入力端子から順に各段の論理演算を出力に至るまで行う.
(6) 各段上のすべての論理ゲートを演算する.
(7) 出力端子の値を PC 側に送信する.
(8) (3) ～ (7)をテストベクタ長の回数分繰り返し行う.

上記のハードウェアアルゴリズムの(6)論理演算部について詳細な処理手順を以下に示す.

(i) 演算を行う論理ゲートの論理機能, 入力ピン数, 入力ピン 1, 2, 3 の値を読み込む.
(ii) (i)で読み込みを行った論理機能, 入力ピン数, 入力ピン 1, 2, 3 の値をレジスタに格納する.
(iii) 論理演算行い, 演算結果をレジスタに格納する.
(iv) 演算結果から次以降に演算を行う論理ゲートに必要な値を選択し, ファンアウト先の論理ゲートの入力ピンの値に格納する.

本アルゴリズムにおいて, 論理ゲートの入力ピン数は最大 3 本までとしたが, それ以上についてもアルゴリズムは同様に考えられる.

Vol.2013-SLDM-163 No.11
2013/11/27

3.2 ハードウェアアーキテクチャ

以下の図 2 は 3. 1 節で説明したハードウェアアルゴリズムをハードウェア化した際の論理シミュレーションの全体のブロック図を示す.

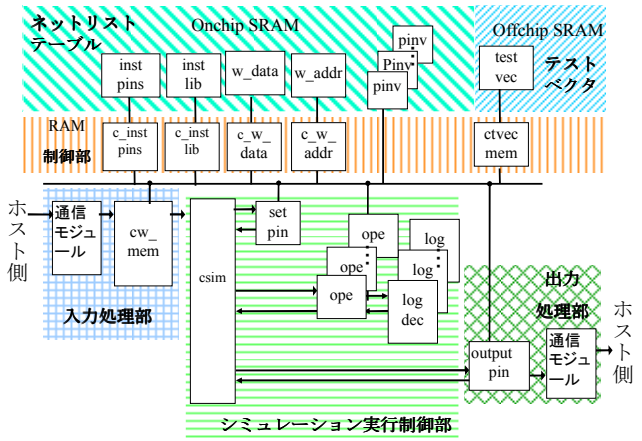

図 2 論理シミュレーションの全体ブロック図

入力処理部でホスト側から送られてくるネットリストテーブル及びテストベクタを受信し, RAM へ書き込む. シミュレーション実行制御部の setpin モジュールでテストベクタを入力端子への設定を行う. 次に ope モジュールで論理ゲートの論理機能や入力ピン等のネットリストテーブルを読み込み, logdec モジュールで論理機能に従って演算が行われる. c_sim モジュールではシミュレーション回数等の制御を行う. 出力処理部で出力となる値を PC 側へ送信する.

3.3 回路データの送信

ホスト側のプログラムで, ハードウェアに送信するネットリストテーブルを作成する. 本ネットリストテーブルの作成手順を以下に示す.

(1) HDL で記述された回路からネットリストコンパイラを用いて, ネットリストテーブルを作成する.
(2) (1)で作成したネットリストテーブル読み込む.
(3) 組合せ回路の論理段及びクリティカルパスを求める.
(4) (3)で求めた論理段数を用いてネットリストテーブルを段数の低い順にソートする.
(5) 入力端子数, 出力端子数, 論理演算回数, テストベクタ長, 論理ゲートの入力ピン数, 論理機能, 論理演算結果のデータ選択番号, 書き込みアドレスを生成する.

上記手順で作成したネットリストデータをハードウェアに送信する.

図 2 の instlib は論理機能, instpins は入力ピン数, w_data は演算結果書き込みデータ選択番号, w_addr は演算結果書き込みアドレスと対応している.

3.4 並列化手法

3. 1 節で説明したハードウェアアルゴリズムの論理演算部を並列にすることで高速化を図る. 同一論理段内の各論理ゲートは, 同段内の演算結果が影響しないため並列に演算が行える. それを可能とするための回路状態値の保持方法及び並列演算方法について次に述べる.

(1) 回路状態値の保持方法

論理演算部の並列化を行う際に問題となるのが論理ゲートの状態値の保持方法である. 論理ゲートの状態値は次段以降の論理ゲートの演算に影響を与える.
また, 並列演算を行うためには, 演算に必要なデータが同時にそろっている必要がある. これらの課題を解決するために 3 つの方法を検討した.

方法 1 は, 全論理ゲートの状態値を容量の大きな offchip RAM に保持し, 論理演算部にそれぞれキャッシュを持たせる. キャッシュとは onchip SRAM を用いたメモリで図 2 のモジュール pinv と対応する. この方法では, キャッシュミスが発生した場合処理に時間がかかる. また, 複雑なメモリコントローラや制御用の CPU が必要となり, ハードウェア量多くなり, 並列化数が減少する.

方法 2 は, 論理演算部にそれぞれキャッシュを持たせ, 全論理ゲートの状態値を保持させる. この方法ではキャッシュの容量が大きくなり, さらに参照されない値が多く存在する.

方法 3 は, 論理演算部にそれぞれキャッシュを持たせ, 各論理演算部の演算に必要な値だけを保持させる. この方法では, キャッシュの容量を小さくできるが, 状態値の格納先をあらかじめ計算しておく必要がある.

本研究では, 並列化数を優先し方法 3 を採用することとし, 演算結果をすべての論理演算部で共有し, 演算結果から必要な値だけを選択する. 選択した値を入力ピン番号と対応したアドレスに格納する.

(2)並列演算方法

図 3 は, 論理演算モジュールを n 並列にした場合のブロック図である.

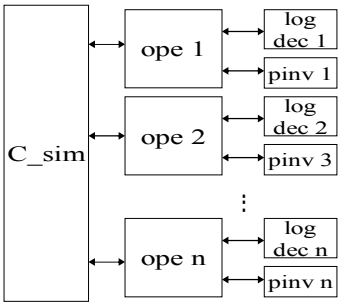

図 3 論理演算モジュールの n 並列化のブロック図

ope, logdec モジュール及びキャッシュとして利用される pinv を並列化個数分用意することで並列演算を実現する. ope, logdec モジュール及び pinv を合わせて論理演算モジュールと呼ぶ. 論理演算モジュールを n 並列化した場合, 同段内の論理ゲートを n 個並列に演算を行えるため並列化しなかった場合と n 並列化した場合では約 n 倍の高速化が期待できる.

3.5 ハードウェア化の設計フロー

論理シミュレーションアルゴリズムの C プログラムをもとにハードウェアアルゴリズムの検討を行い, HDL で記述しファンクションシミュレーションを行う. 次に FPGA へ実装した場合のタイミングシミュレーションで性能評価を行い, FPGA へ実装し FPGA 上での性能評価を行う. FPGA への実装を対象とした場合のハードウェア化の手順を図 4 に示す.

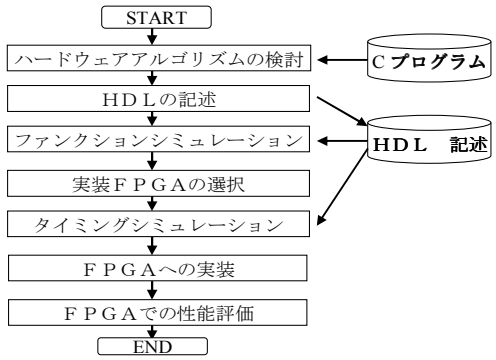

図 4　ハードウェア化の設計フロー

今回は, FPGA への実装までは行わず, FPGA 上でのタイミングシミュレーションまで行い, FPGA シミュレーションの性能評価を行った.

4. 論理シミュレーションアルゴリズムの FPGA 化について

4.1 実装対象とする FPGA

FPGA(Field Programmable Gate Array)は, 回路構成を変更できる（プログラマブル）デバイスであり, 本研究で対象とする FPGA の詳細を以下に示す.

ファミリ ： Cyclone Ⅲ LS
デバイス ： EP3CLS200F780C7
- LE (ロジックエレメント)数 : 198,464 個
- M9K メモリブロック数 : 891 個
- M9K メモリ bit 数 : 8,211,456 bit

4.2 ハードウェアリソースの見積り

ネットリストテーブルは BRAM に格納し, テストベクタは offchip SRAM に格納する. 論理ゲートの状態値を保持するキャッシュは BRAM を用いることとする.

次に対象とする FPGA へ実装するための BRAM の見積りを説明する. 以下の表 1 にそれぞれのデータ幅, アドレス数, BRAM 使用個数を示す.

表 1　BRAM 見積り

テーブル名	格納最大数 (アドレス)	データ幅 (bit)	合計bit数 (RAM容量)	RAM個数
instlib	256	1,024 (4bit x 256)	262,144	57
instpins	256	512 (2bit x 256)	131,072	29
w_data	1,024	2,048 (8bit x 256)	2,097,152	228
w_addr	1,024	2,560 (10bit x 256)	2,621,440	285
pinv	9,216	1	2,359,296 (pinv x 256)	256
合計			7,471,104	855

論理演算部の並列化数は BRAM の個数および LE 数, 処理速度より 256 並列とする. 演算結果の書き込み回数を論理ゲート 1 処理内 6 回とするため 1 論理ゲート当たり 6 個の w_addr, w_data を持つ.
今回提案するハードウェアアルゴリズムでは, 格納できる最大論理ゲート数は 43,250 までとなる.

提案する論理シミュレーションアルゴリズムを対象とする FPGA に実装した場合のリソース消費量と消費率を以下の表 2 に示す.

表 2 リソース消費量と消費率

リソース	消費量	消費率
Logic Elements	141,681 LE	71%
M9K Memorys	855個	96%
M9K Memory bits	5,373,953 bit	65%

論理シミュレーションアルゴリズムを対象とする FPGA へ実装した場合, ハードウェアリソースの消費率は, LE は 71%, BRAM 個数は 96%, BRAM 使用 bit 数は 65%である.

5. FPGA シミュレータの性能評価

5.1 並列化による性能向上

本論理シミュレーションアルゴリズムの論理演算部の並列なし(FPGA_SIM1)と 256 並列化した(FPGA_SIM256)場合の論理シミュレーションを FPGA へ実装し, シミュレーション時間について検証を行う. 評価用の論理回路は, 組合せ回路として, 4bitt-Adder を 40, 80 個並列に並べた２種類の回路(adder4x40, adder4x80)を, 順序回路として, 8bit-CPU を 1, 4, 16 個並列に並べた３種類の回路(cpu x 1, cpu x 4, cpu x 16)を用いた. テストベクタ長は 10, 000 テストサイクルとし, 表 3 に評価回路を示す.

表 3 評価回路

	論理ゲート数	F F 数	論理段数
adder4x40	800	0	10
adder4x80	1,600	0	10
cpu x 1	2,111	173	56
cpu x 4	8,444	692	56
cpu x 16	33,776	2,768	56

ModelSim ALTERA STARTER EDITION 10.1d で FPGA_SIM のタイミングシミュレーション行った結果を表4, 表5に示す. なお, それぞれの周波数はFPGA_SIM1, FPGA_SIM256 ともに 100MHz となっている.

表4 組合せ回路(4bit-adder)での FPGA_SIM の評価結果

評価回路	シミュレーション時間 (msec)		高速化率
	FPGA_SIM1(a)	FPGA_SIM256(b)	(a/b)
adder4x40	591	9.1	64.9
adder4x80	1,123	10.7	104.9

表 5 順序回路(8bit-CPU)での FPGA_SIM の評価結果

評価回路	シミュレーション時間 (msec)		高速化率
	FPGA_SIM1(a)	FPGA_SIM256(b)	(a/b)
cpu x 1	1,050	46.7	22.5
cpu x 4	4,200	57.2	73.4
cpu x 16	16,800	136.3	123.2

表4, 表5から論理演算を並列化することによって組合せ回路及び順序回路ともに高速化していること分かる. これは, 段ごとの論理ゲート数が増加すると並列化した論理演算部の使用率が向上するためである.

並列なしと 256 並列での理想的な比率は, 並列化数分である 256 倍であるが, 今回評価した回路では段ごとの論理ゲート数が 256 個に達していない段が存在しているため論理演算部の使用率が低いことが理想的な比率とはならなかったと考えられる. また本アルゴリズムでは, 並列化による論理演算処理の同期のために並列なしに比べて処理クロック数が増加している.

以上の結果より, 論理演算部の並列化数を増加させることにより段内の論理ゲートを並列に演算ですることができるため高速化できることが確認できた.

5.2 商用論理シミュレータとの性能比較

FPGA_SIM256 と商用論理シミュレータ ModelSim のシミュレーション時間を比較した. 評価環境と評価回路は以下のとおりである.

・評価環境

商用シミュレータ ModelSim (VDEC 提供)

- ModelSim SE 6.2e
- PC 環境 : Windows XP SP3, Intel Core i7-950 3.07GHz

・評価回路

- 組合せ回路 adder4x40, adder4x80
- 順序回路 cpu x 1, cpu x 4, cpu x 16

表 6 は 4bit-adder を, 表 7 は 8bit-CPU を評価したシミュレーション時間を示す.

表 6 組合せ回路(4bit-adder)での評価結果

評価回路	シミュレーション時間 (msec)		比率
	ModelSim(a)	FPGA_SIM256(b)	(a/b)
adder4x40	191	9.1	21.0
adder4x80	347	10.7	32.4

表 7 順序回路(8bit-CPU)での評価結果

評価回路	シミュレーション時間 (msec)		比率
	ModelSim(a)	FPGA_SIM256(b)	(a/b)
cpu x 1	127	46.7	2.7
cpu x 4	361	57.2	6.3
cpu x 16	1,450	136.3	10.6

表 6 から FPGA_SIM256 は商用シミュレータと比較して adder4x40 で約 21 倍, adder4x80 で約 32 倍の高速性となる.

表 7 から FPGA_SIM256 は商用シミュレータと比較して cpu x 1 で 2 倍, cpu x 4 で約 6 倍, cpu x 16 で約 10 倍の高速性となる.

商用論理シミュレータは, 順序回路では組合せ回路と比較してイベントの発生率が低いため論理ゲート数が同程度の adder4x80 と cpu x 1 では cpu x 1 のほうが速くなっている.

FPGA_SIM256 は, レベルソート法を用いているため段数と段ごとの論理ゲート数によってシミュレーション時間は決定される. Adder4x80 と cpu x 1 では, cpu x 1 は段数が多く段ごとの論理ゲート数が少ないため論理演算部の使用率が少なくシミュレーション時間が長いと考えられる.

今回は実装する FPGA の BRAM の関係上 cpu x 16 までしかシミュレーションを行うことができなかったが, さらに大規模な回路については, cpu x 512(約 100 万ゲート)で FPGA_SIM256 は, 商用シミュレータと比較すると約 20 倍以上の高速性が達成可能であると推定できる.

以上のことから, 段内の論理ゲート数が増加すると, FPGA_SIM256 は商用シミュレータよりも高速にシミュレーションできる.

5.3 大規模回路対応及び高速化

本研究では, 対象とした FPGA の BRAM 数の関係より

シミュレーションできる最大論理ゲート数が 43,520 までとなった. 今後 BRAM の容量が大きな FPGA へ実装することによって大規模回路への対応が可能であると考える. また, 大規模回路に対応した場合, 回路に対応して BRAM は大きくしなければならないが, その他のハードウェアリソース量はほぼ変化しない.

次にキャッシュに割り当てられる BRAM の個数が増加できれば, 並列化数も増加することができる. 図 5 は, 並列化数を 512, 1024 と増加させた場合のそれぞれの FPGA_SIM(FPGA_SIM512, FPGA_SIM1024）のシミュレーション時間の見積りを行った結果を示す.

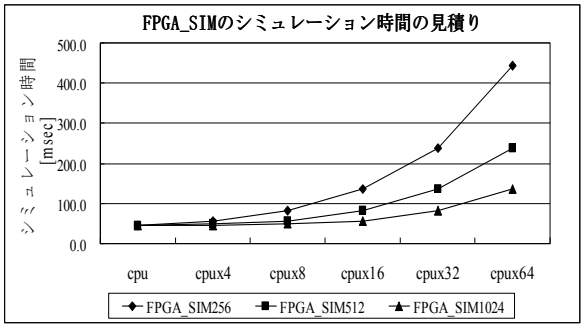

図 5　順序回路(8bit-CPU)で FPGA_SIM の評価見積り

図 5 より, FPGA_SIM256 に対して cpu x 64(13 万ゲート)で FPGA_SIM512 は 1.9 倍, FPGA_SIM1024 は 3.2 倍の高速化率となった. 段内の論理ゲート数が増加すれば並列数分の高速化率が期待できる. また, cpu x 64(13 万ゲート)で商用論理シミュレータと比較して FPGA_SIM512 は 28 倍, FPGA_SIM1024 で 50 倍の高速性が達成できる見通しが得られた.

6. まとめと今後の課題

6.1 まとめ

本研究では, 論理シミュレーションアルゴリズムのハードウェア化を行い, FPGA へ実装することによる高速化を目指した. 高速化手法として論理演算部の並列化を行った. 今回実装対象とした FPGA では 256 並列まで行った. 今回提案する論理シミュレーションアルゴリズムのタイミングシミュレーションを行ったところ, FPGA_SIM256 は商用論理シミュレータと比較して組合せ回路では 32 倍, 順序回路では 3 万ゲートで約 10 倍の高速性を達成できる見通しが得られた. また, 大規模回路での推定を行った結果, 100 万ゲートで 20 倍の高速性を達成できる推定を得た. また, FPGA_SIM512 では, 商用シミュレータと比較して 13 万ゲートで 28 倍, FPGA_SIM1024 では 50 倍の高速性が達成できる見通しが得られた.

6.2 今後の課題

本研究では, 論理シミュレーションアルゴリズムのハードウェア化を行い, FPGA へ実装した場合のタイミングシミュレーションを行いその高速化率の評価を行った. 今回実装対象とした FPGA では 256 並列化でシミュレーション可能最大回路は 43,520 ゲートまでとなったが, BRAM の規模が大きい FPGA へ実装することで大規模回路への対応, 並列化個数の増加による更なる高速化が期待できる. さらに, 論理演算部のパイプライン処理を追加することによる高速化も考えられる. FPGA を用いた論理エミュレータよりも低速ではあるが, 本ハードウェアシミュレータでは, 回路データの書込みをメモリに書込むため書込み時間は高速である. 今後大規模論理回路での総合的な比較を行なう必要がある. 本ハードウェアアルゴリズムの LSI 化を行うと, 周波数の向上により FPGA と比較して 1 桁以上の高速化も期待できる. また, 論理合成等での演算子の数を減らすことによる高速化なども検討していく. ネットリストテーブルを容量の大きな DRAM 等の offchip RAM への保持などによる大規模回路対応も検討していく. 今後はさらに大規模な順序回路による評価を行いさらなる高速化アルゴリズムの確立を図りたいと考える.

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] Gregory F. Pfister, "THE YORKTOWN SIMULATION ENGINE: INTRODUCTION", 19th Design Automation Conference, 1982
[2] 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
[3] トウブンチク, 竹内勇矢, 村岡道明, "マルチコアプロセッサ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
[4] Debapriya Chatterjee, Andrew DeOrio, Valeria Betracco, ”Event-Driven Gate-Level Simulation with GP-GPUs”, DAC’09, July 26-31, 2009
[5] 橋口拓哉, 豊永雅彦, 村岡道明, "GP-GPU を用いた並列論理シミュレーション手法", DA シンポジウム 2013 論文集, pp. 97-102, 2013 年 8 月 22 日
[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