Vol.2015-SLDM-169 No.11
2015/1/29

FPGA を用いた高速論理シミュレーションエンジン

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

概要　本報告では，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とその高速化手法およびそれに基づき試作したシミュレーションエンジンについて述べる．アルゴリズムの高速化手法は，高速アクセス可能な FPGA 向きのデータ構造，論理ゲートの演算部の並列化手法，および，FPGA 内部の高速メモリの効率的使用手法により構成される．本手法に基づくアルゴリズムを FPGA に実装し，シミュレーションエンジンを試作し，性能を評価した結果，高速市販論理シミュレータよりも 20 倍以上の高速化ができる見通しを得た．

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 High Speed Logic Simulation Engine 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, an acceleration method of a hardware algorithm of the logic simulation using the leveled method is proposed, and a prototype simulation engine based on it is developed. The acceleration algorithm consists of the high speed accessible data structure for FPGA, the parallel logic evaluation parts of the hardware algorithm and the efficient and high-speed memory access method in FPGA. The parallel logic simulation algorithm was implemented on the FPGA, and the performance of the prototype simulation engine was evaluated. As the result of the evaluation, the speed of the proposed hardware algorithm is estimated to achieve approximately 20 times faster than one of the fastest commercial logic simulator.

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

1. はじめに

大規模回路やシステムの論理検証には，ソフトウェアとハードウェアの協調シミュレーションが行われているが，通常は非常に長い時間を要し，高速化が望まれる．ハードウェア部のシミュレーションの高速化方法として論理シミュレーションのハードウェア化[1]もかつては行われていたが，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，大規模回路では，FPGA の再構成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協調シミュレーションに使用されることが多い．この数年来，論理シミュレーションの高速化方法として，マルチコア[2][3]や GP-GPU[4][5]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，先行研究[6]を基に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，及び本アルゴリズムを FPGA に実装し論理シミュレーションエンジンの試作を行った．また，アルゴリズムの高速化方法として，高速アクセス可能な FPGA 向きのデータ構造，論理ゲートの演算部の並列化手法，および FPGA 内部の高速メモリの効率的使用手法を検討した．

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述べ，3 章で提案するハードウェアアルゴリズム及び論理ゲートの演算部の並列化，FPGA 内部の高速メモリの効率的使用手法による高速化、並列演算用のデータ構造について述べる．4 章で FPGA に実装するための制約について述べる．5 章で論理演算の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 に実装した場合の実行時間をタイミングシミュレーションによって評価した．また．論理演算を並列化した場合の 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

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アルゴリズム

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が，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．

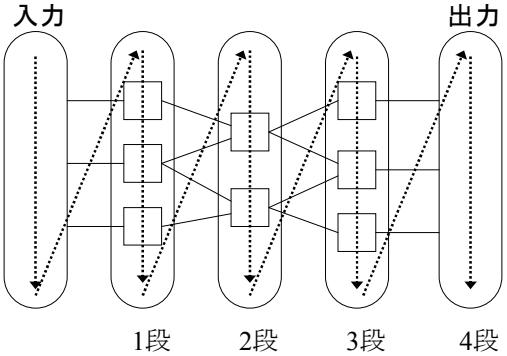

図 1 レベルソート法の処理手順

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，論理段内の論理ゲートは並列に演算することができるため，並列が容易に可能なハードウェア化に向くと考えられる．

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 FPGA を使用し，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なった．ソフトウェアでは難しい並列化をハードウェアでは容易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 へ転送する.

(2) ネットリストテーブルは onchip RAM(BRAM)，テストベクタ及び出力値は offchip RAM 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 offchip RAM に格納する.

(8) 出力結果を offchip RAM から読み込み，ホスト(PC)に転送する.

(9) (3) ～ (8)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)論理ゲートの演算を並列に行うことで高速化を図る．(6)の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 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 1，2，3 の値をレジスタに格納する.

(iii) 並列論理演算行い，演算結果をレジスタに格納する.

(iv) レジスタから onchip RAM(BRAM)に値を格納する.

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.

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.

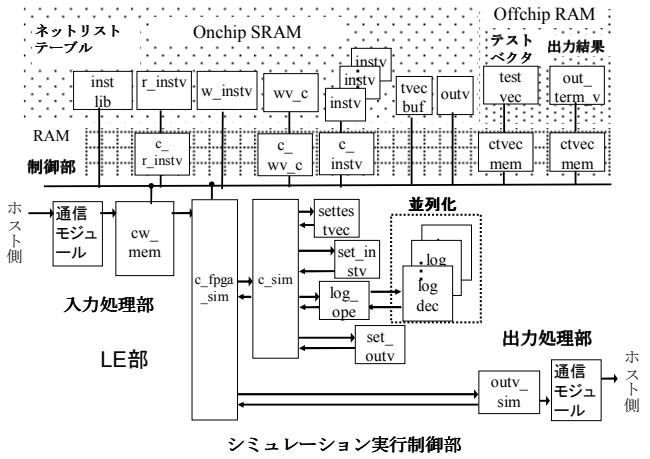

図 2 論理シミュレーションの全体ブロック図

入力処理部でホスト側(PC)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 settestvec モジュールでテストベクタを入力ピンへの設定を行う. 次に log_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. log_dec モジュールを並列に用意することで論理ゲートの並列演算を行う. set_instv では, 演算を行なった論理ゲートの状態値を onchip RAM に書き込む．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ホスト側(PC)へ送信する．ネットリストテーブルは，アクセスが頻繁に行われるため FPGA の onchip RAM(BRAM)を用いる．テストベクタ，出力結果は，容量が大きな offchip RAM を用いる.

3.3 回路データの転送

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，入力ピン値となる論理ゲートの状態値が格納されているアドレス，論理機能，F/O 元となる論理ゲートの状態値の選択信号，演算段ごとの演算結果の書込み回数を生成する.

上記手順で作成した実行用ネットリストテーブルをハードウェアに送信し，FPGA の BRAM や offcnip RAM に格納する.

図 2 の r_instv は入力ピン値となる論理ゲートの状態値が格納されているアドレス，instlib は論理機能，w_data は F/O 元となる論理ゲートの状態値の選択信号，wv_c は演算段数ごとの演算結果の書込み回数と対応する.

3.4 並列演算手法

3．1 節で説明したハードウェアアルゴリズムの(6)論理ゲートの演算を限られた容量の BRAM を効率的に使用して並列化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が，FPGA の制約によって並列に演算できる論理ゲートの数（以降，並列演算数とする）が決まる．並列演算数と FPGA の制約については 4 章で述べる.

以下の図 3 の回路図で並列演算手法について述べる．例として，図中の並列演算数は 5 とする.

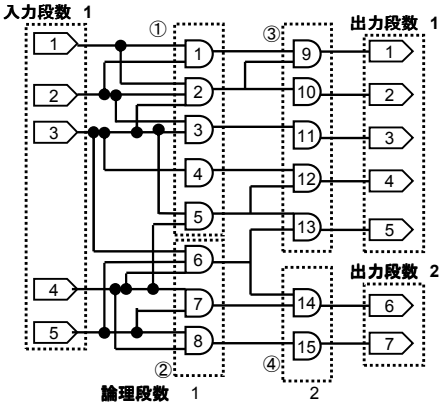

図 3 回路図例

図 3 の外部入力端子，論理ゲート，外部出力端子それぞれの内部の数字は外部入力端子番号, 論理ゲート番号, 外部出力端子番号を表す．点線内の論理ゲートが並列に演算され、丸数字は並列に演算する順番を表し，以降演算段数と呼ぶ．また，外部入出力端子も論理ゲートと同様に並列演算数で分割し, 入力段数, 出力段数を求める.

図 4 に図 3 の回路例を元に演算段数でソートしたものを表す.

図 4 の左図は図 3 の回路図例を表形式で表したものである．これを入力段数，演算段数，出力段数でソートしたものが右図である．入力段数，演算段数，出力段数内の番号の低い順に 1 列目の論理ゲート群（外部入出力を含む）とし，5 列の論理ゲート群に分ける．並列演算数が N の場合 N 列の論理ゲート群に分けられる.

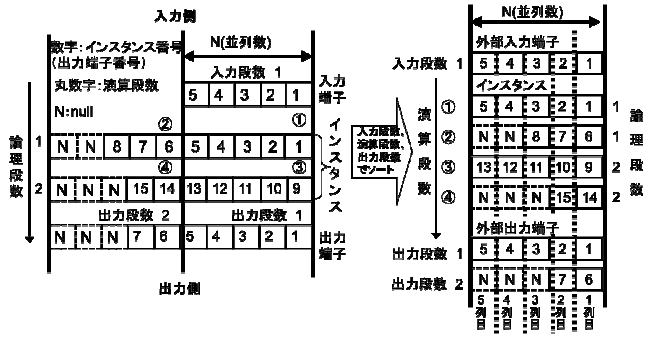

図 4 回路の演算段数でのソート

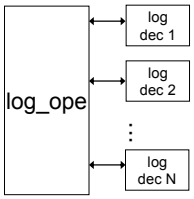

図 5 論理ゲートの演算モジュールの並列化

図 5 は論理ゲートの演算を行なうモジュールを N 並列化した場合のブロック図である．log_ope モジュールは，演算を行なう論理ゲートの論理機能，入力ピンの状態値を読みこみ，log_dec モジュールで論理機能，入力ピンの状態値を基に論理演算を行なう．log_dec モジュールを並列演算数分用意することで並列演算を実現する．log_dec1 モジュールでは，1 列目の論理ゲート群が演算，log_dec2 モジュールでは，2 列目の論理ゲートが演算され，以降の列の論理ゲートも同様に演算される．

論理ゲートの演算段数①から順に最終演算段数まで並列演算することで，出力結果が確定する．

3.5 並列演算用データ構造

ネットリストテーブルは高速な論理演算を行うため，必要最小限なテーブル部分のみ FPGA 内部の高速な BRAM に書込み使用する．並列演算を行うためには，演算を行う論理ゲートの論理機能等のデータを並列演算数分同時に用意する必要がある．そこで，本データ構造は，演算段数に着目し，演算段数を BRAM のアドレスと置き換える．1 アドレスに並列演算に必要なデータを保持することで，並列演算が実現できる．

論理ゲートの状態値を格納するメモリは論理ゲートの入力ピンの値読込みと結果の書込みで頻繁にアクセスが行われるため，論理ゲートの状態値は高速にアクセスできる FPGA の BRAM に書込み使用する．また，並列演算を行う場合，論理ゲートの入力ピン値となる論理ゲートの状態値の読込みが並列演算数同時に行われる．

そこで，本データ構造では状態値を格納する BRAM を並列数分用意し，演算を行なう論理ゲートの入力ピン値となる F/O 元の演算結果だけを格納することで，並列演算の実現と BRAM に格納する状態値の削減を行う．

図 4 のように論理ゲートを 1 から 5 列目の論理ゲート群に分けることで，1 列目の論理ゲート群の入力ピン値となる F/O 元の状態値を 1 個目の BRAM に格納する．以降ついても同様に対応している．

4. 論理シミュレーションアルゴリズムの FPGA 実装

4.1 実装対象とする FPGA ボード

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，本研究で実験用に使用した FPGA の詳細を以下に示す．

搭載 FPGA： ALTERA Cyclone Ⅲ 3C16

- LE (ロジックエレメント)数 : 15,408 個
- M9K メモリブロック(BRAM)数 : 56 個
- M9K メモリ総 bit 数 : 516,096bit

今回は FPGA シミュレーションエンジンの性能評価のための試作として，安価な FPGA を使用したが，大規模高性能 FPGA への展開も可能である．

4.2 ハードウェアリソースについて

ネットリストテーブルは BRAM に格納し，テストベクタ，出力結果は offchip RAM に格納する．論理ゲートの状態値は BRAM を使用する．また，テストベクタと出力結果を一時的に保持するバッファに BRAM を使用する．

次に今回使用した FPGA へ実装するための BRAM の見積りを述べる．以下の表 1 にそれぞれのデータ幅，アドレス数，BRAM 使用個数の見積りを示す．

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	64 (4bit×16)	16,384	2
r_instv	256	480 (10bit×16×3)	122,800	14
wv_c	256	64 (4bit×16)	16,384	2
w_instv	2,048	4	131,072 (w_instv×16)	16
instv	4,096	1	32,768 (instv×16/2)	8
tvecbuf	1,024	16	16,384	2
outv	1,024	16	16,384	2
合計			352,176	46

各テーブルについては 3.3 節で述べた実行用ネットリストテーブルと対応する．

並列演算数は FPGA の BRAM の個数および LE 数，処理速度により決定される．今回対象とする FPGA では，16 並列演算まで可能であるため，並列演算数を 16 として論理シミュレーションエンジンを試作した．また，格納できる最大論理ゲート数は，4,096 までとなる．

今回試作した並列演算数 16 の論理シミュレーション

エンジンを 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	リソース量	消費量	消費率
LE [個数]	15,408	3,357	19%
BRAM [個数]	56	46	82%
BRAM [bit数]	516,096	385,024	75%

論理シミュレーションエンジンのハードウェアリソースの消費率は，LE は 19%，BRAM 個数は 82%，BRAM 使用 bit 数は 75%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

今回提案する論理シミュレーションアルゴリズムの論理演算の並列化なし(FPGA_SIM1)と 16 並列化した(FPGA_SIM16)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, 4bit-Adder を 80, 160 個並列に並べた 2 種類の回路(adder4 x 80，adder4 x 160)を，順序回路として，8bit-CPU の回路（cpu x 1）を用いた．表 3 に評価回路を示す．テストベクタ長は 100,000 テストサイクルである.

表 3 評価回路

	論理ゲート数	FF数	論理段数
adder4 x 80	1,600	0	10
adder4 x 160	3,200	0	10
cpu x 1	2,111	173	56

商用シミュレータを使用して，FPGA_SIM のシミュレーション時間をタイミングシミュレーションにより評価した結果を表 4 に示す．なお，FPGA_SIM の周波数は共に 100MHZ で動作可能であることが分かった.

表 4 評価回路での FPGA_SIM の評価結果

評価回路	シミュレーション時間[sec]		比率
	FPGA_SIM1(a)	FPGA_SIM16(b)	(a/b)
adder4 x 80	11.2	1.7	6.5
adder4 x 160	22.4	3.3	6.8
cpu x 1	14.8	2.5	5.9

表 4 から並列数を増加させることで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，論理段ごとの論理ゲート数が並列演算数よりも多くなり，並列演算が効率よく行われるためである.

並列演算なしと 16 並列演算での理想的な比率は, 並列演算数分の 16 倍であるが，16 並列演算では並列演算化のための処理を追加しているため処理クロック数が増加し理想的な比率とならなかった.

以上の結果より，並列演算数を増加させることにより論理段内の論理ゲートを並列に演算し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_SIM16 と商用シミュレータのシミュレーション時間を以下の評価環境と評価回路で比較した.

・評価環境

商用シミュレータ(C_SIM)

- ModelSim SE 10．2c(VDEC 提供)
- PC 環境 : Windows XP SP3, Intel Core i7-950 3.07GHz

・評価回路

- 組合せ回路　　adder4 x 80，　adder4 x 160
- 順序回路　　cpu x 1

テストベクタ長は 100,000 テストサイクルとし，表 5 にそれぞれのシミュレーション時間を示す.

表 5 評価回路での評価結果

評価回路	シミュレーション時間[sec]		比率
	C_SIM(a)	FPGA_SIM16(b)	(a/b)
adder4 x 80	3.5	1.7	1.9
adder4 x 160	7.6	3.3	2.3
cpu x 1	1.2	2.5	0.5

表 5 から FPGA_SIM16 は商用シミュレータと比較して組合せ回路では adder4 x 80 で約 1.9 倍，adder4 x 160 で約 2.3 倍，順序回路 cpu x 1 で約 0.5 倍の高速性となった.

順序回路は組合せ回路と比較してイベントの発生率が低いため、商用シミュレータでは adder4 x 160 と cpu x 1 を比較すると cpu x 1 のほうがシミュレーション時間は短い．FPGA_SIM16 は，演算段数（論理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ため，商用シミュレータと比較して順序回路 cpu x 1 では，組合せ回路よりも高速化率は下がっている.

対象とした FPGA では，16 並列が限界であったが，BRAM の量が多い FPGA を用いることで並列数は増加させることができる．次節で，大規模回路対応と並列数を増加させた場合の高速化率について推定する.

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 が 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は 4,096 までとなった. 今後 BRAM の容量が多い 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，LE(論理ブロック)量はほぼ変わらない.

また，BRAM の容量が多い FPGA へ実装することによって並列演算数を増加させ，高速化できる．並列演算数を 256，512，1,024 と増加させた場合のそれぞれの

FPGA_SIM(FPGA_SIM256，512，1024）のシミュレーション時間の見積りを行った．見積もりは、(1 つの演算段にかかる処理クロック数の演算段数分の総和)×テストベクタ長÷周波数，で求めた．

テストベクタ長は 100,000 テストサイクルとし，表 6 に見積もりに用いた評価回路を示す．一般的な大規模回路は順序回路であるため、順序回路である 8bit-CPU を並列に並べた場合についてのみ評価対象とした。表 7 に表 6 の回路を対象とした FPGA_SIM のシミュレーション時間の見積もり結果を示す．FPGA_SIM の周波数は 100MHz で見積もりを行った．

表 6 見積もりに用いた評価回路

	論理ゲート数	FF数	論理段数
cpu x 1	2,111	173	10
cpu x 20	42,220	3460	10
cpu x 40	84,440	6920	56

表 7 順序回路(8bit-CPU)で FPGA_SIM の評価見積り

	シミュレーション時間[sec]				比率		
	FPGA_SIM			C_SIM (d)	d/a	d/b	d/c
	256(a)	512(b)	1024(c)				
cpu x 1	0.9	0.8	0.8	1.2	1.3	1.5	1.5
cpu x 20	3.0	1.7	1.1	20.1	6.7	11.8	18.3
cpu x 40	5.5	3.0	1.7	38.3	7.0	12.8	22.5

商用シミュレータと比較して順序回路 cpu x 40 (約 8 万ゲート)で FPGA_SIM256 約 7.0 倍，FPGA_SIM512 は約 12.8 倍，FPGA_SIM1024 は約 22.5 倍の高速性が達成できる見通しが得られた．

回路の論理段内の論理ゲート数が増加すれば，並列演算を最大限活用できるため，大規模回路では更なる高速化率が期待できる．

並列演算数が 1,024 の FPGA_SIM1024 は，BRAM 個数が最低 2,000 個程度あれば実装できるため，ALTERA Stratix Ⅴ等の最新 FPGA ならば実装できると考えられる．

6. まとめと今後の課題

6.1 まとめ

本研究では，先行研究を基に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，論理シミュレーションエンジンの試作を行った．高速化手法として論理ゲートの並列演算および内部メモリの効率的手法，FPGA 向きデータ構造の検討を行った．今回実装対象とした FPGA では並列演算数は 16 となった．今回提案する論理シミュレーションアルゴリズムの実行時間をタイミングシミュレーションにより評価した結果，FPGA_SIM16 は商用論理シミュレータと比較して組合せ回路で約 2.3 倍，順序回路で約 0.5 倍となった．しかし，大規模 FPGA を対象とした場合に並列演算数を増加させると，商用シミュレータと比較して順序回路約 8 万ゲートで FPGA_SIM256 では約 7.0 倍， FPGA_SIM512 では約 12.8 倍，FPGA_SIM1024 では約 22.5 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 へ実装した場合の実行時間をタイミングシミュレーションによって評価を行った．今回実装対象とした FPGA では 16 並列演算でシミュレーション回路は 4,096 ゲートまでとなったが，BRAM の容量が大きい FPGA へ実装することで大規模回路への対応，並列演算数の増加による更なる高速化が期待できる．

本シミュレーションエンジンは，FPGA を用いた論理エミュレータよりも低速ではあるが，回路データの書込みを BRAM に書き込むため書込み時間は高速である．今後通信等も含め，大規模論理回路での総合的な比較を行なう必要がある．今回は論理シミュレーションの高速化に着目し通信に関しては触れていないが，PCI express 等の高速な IF を用いることでネットリストテーブルや出力結果のホストとの高速通信についても検討していく．

本ハードウェアアルゴリズムの LSI 化を行うと，ネットリストテーブルを保持している BRAM を SRAM に置き換えられるため，容量や配置が自由に設定でき大規模化や並列演算数の増加，また周波数の向上等により FPGA と比較して 10 倍以上(商用シミュレータの数百倍)の高速化も期待でき，論理エミュレータよりも効率のよいデバック性と高速性の実現も見込まれる．

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 2013，2013 年 8 月

[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

[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 2013，2013 年 8 月

[6] 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