

ICCAD 報告

松永 裕介†

†九州大学大学院システム情報科学研究院

〒819-0395 福岡市西区元岡 744

E-mail: †matsunaga@c.sce.kyushu-u.ac.jp

あらまし 2007年11月5日から8日まで米国カリフォルニアで開催された国際会議 ICCAD 2007 の概要と、おもに FPGA 関係のセッションの論文の紹介を行う。

キーワード ICCAD, 会議報告, FPGA

ICCAD summary report

Yusuke MATSUNAGA†

† Faculty of Information Science and Electrical Engineering, Graduate School of Kyushu University

744 Motooka, Nishi-ku, Fukuoka, 819-0395 Japan

E-mail: †matsunaga@c.sce.kyushu-u.ac.jp

Abstract This paper briefly summarizes the overview of ICCAD 2007, which was held in California, USA during Nov. 5 to Nov. 8. Also, some papers of sessions related FPGA design technology are introduced.

Key words ICCAD, conference report, FPGA

1. 概 要

ICCAD (International Conference on Computer-Aided Design) は LSI の設計自動化技術に関する国際会議で、2007 年 11 月 5 日から 8 日まで米国カリフォルニア州サンノゼ市の DoubleTree ホテルで開催された。ICCAD 2007 がちょうど 25 周年めに当たるが、DoubleTree ホテルでの開催はすでに 10 年以上変わっていない^(注1)。ICCAD は同じく米国で開催される DAC (Design Automation Conference)、ヨーロッパで開催される DATE (Design And Test in Europe)、アジアで開催される ASP-DAC (Asia South-Pacific Design Automation Conference) と並んで EDA 関係の 4 大会議と呼ばれている。これらの会議の運営は独立だが、それぞれ姉妹会議 (sister conference) として提携を行っており、チュートリアルや特別セッションのための情報交換や日程調整などを密に行っている。

会議の形式は 5 日がチュートリアル、6 日から 8 日が 4 つの並行トラックを持つテクニカルセッションで、6 日の午前中と 8 日の昼に基調講演があるという一般的なフォーマットである。

おなじく米国で開催される DAC とは異なり、大手 EDA ベンダの展示会は併設されていない。ただし Technology Fair という名称で 1 日だけ、小さなデモを行うスペースが用意されている。これは主に、スタートアップ企業の宣伝の目的で使用されている。また、大手 EDA ベンダは参加登録の e-mail 配信やカンファレンスバッグ、コーヒープレークなどの費用負担で会議の運営をサポートしている。

今回の ICCAD の参加者は約 640 人で、2006 年度より約 60 人減ったそうである。相変わらず論文採択率は 20% 台で厳しいが、論文投稿件数も横ばいからやや落ちてきているようで、会議の関係者はこの傾向を真剣に受け止めている。3 月に開催される DATE の投稿件数もこの数年の単調増加が止まり、全体では昨年度を下回っている。ASP-DAC は開催地が変わるので単純に比較はできないが、やはり 2008 年度は 2007 年度よりも投稿件数が減っている。ICCAD における日本からの発表はテスト関係が 2 件あったのみで、年々存在感が薄くなっている。対照的に、台湾からの発表は多く、特に Taiwan National University から 10 件程度発表があり、今回の ICCAD で単独の機関からの発表としては最多であった。日本のアクティビティが低迷している原因はいろいろと考えられるが、なるべく早い時期にこの分野における研究活動を活発にする対策を講じる必要があると思われる。

(注1)：当初の会場はサンノゼ市の隣のサンタクララの市の DoubleTree ホテルであった。その後サンノゼの現 DoubleTree ホテル (当時の名前は Red Lion ホテル) に会場が移り、数年前にサンノゼのホテル名が DoubleTree になっている。ちなみにサンタクララの DoubleTree ホテルは現在は別の名称になっている。

2. FPGA 設計技術関係の論文紹介

2.1 Session 3A: Resilient and Regular Circuits

3A.1

“Design, Synthesis and Evaluation of Heterogeneous FPGA with Mixed LUTs and Macro-Gates”

Yu Hu, Satyaki Das, Steve Trimberger and Lei He (UCLA) Research Laboratory (Xilinx)

LUT に簡単な AND/OR ゲートを付加した混合型 FPGA アーキテクチャと専用のツールチェーンの提案。LUT の入力数を大きくすると必要な段数と LUT 数を減らすことができるが、LUT の入力数を 1 つ増やすごとに LUT の面積は約 2 倍となる。さらに入力数が多い論理関数は、実際の回路中で均等に現れるわけではなく、ごく限られた関数のみが多く現れる傾向がある。そこで、ベンチマーク回路を解析して頻繁に現れる 6 入力関数を 4 つ選び出し、4 入力 LUT にいくつかの AND/OR ゲートを付加することでそれらの 6 入力関数を実現可能とした混合型 FPGA のアーキテクチャを提案している。さらに、このアーキテクチャに特化したテクノロジマッピング、パッキングなどのツールの提案もしている。それほど斬新なアイデアがあるわけではないが、ねらいとするポイント、そのためのアプローチと実際の提案手法、実験結果の提示と非常によくできた engineering paper であると言える。また、FPGA のアーキテクチャから論理合成、回路設計などいくつかの分野にまたがってソツのない仕事をしているのも米国の大学の優秀な研究グループの特徴と言える。

3A.2

“Device and Architecture Concurrent Optimization for FPGA Transient Soft Error Rate”

Yan Lin and Lei He (UCLA)

宇宙から降り注ぐ中性子線の影響で CMOS トランジスタの出力が反転してしまう“ソフトエラー”を考慮した FPGA のアーキテクチャ評価の話。従来はソフトエラーという SRAM やフリップフロップの値が反転する“permanent”ソフトエラーが主に考慮されてきたが、著者らの見積もりでは 25nm になると論理回路中の論理反転が回路全体の動作に影響を及ぼす“transient”ソフトエラーの影響が無視できなくなるとのこと。この“transient”ソフトエラーの影響がチップレベルでどの程度になるのかを見積もるシミュレーションの工夫がメインのコントリビューションとなっている。以下に簡単にこの論文におけるアプローチを説明する。

まず、論理回路中の論理反転はその全てが回路全体の動作に影響を及ぼす訳ではない。具体的には、論理反転を起こした信号線からフリップフロップまでエラーが伝搬して、かつ、誤った値が次のフリップフロップの値として取り込まれて始めて、回路動作としてのエラーとなる。そのため、ソフトエラーの影響を見積もるためには以下の 3 つの伝搬阻害要因を考える必要がある。

logic masking: 例えば AND ゲートの一つの入力が“0”であれば出力は他の入力の値に関わりなく“0”となる。つまり、他の入力の値はブロックされる。このようにソフトエラーの影響で論理反転が起こっても、それがフリップフロップに到達するまでにブロックされればエラーはそれ以上、伝搬しない。logic masking はいわゆる縮退故障検出用のテストパターン生成における故障伝搬と同様の概念である。

electrical masking: ソフトエラーによる論理反転は微小な時間しか持続しないため、CMOS 回路中では短いパルスが回路中に印加されたものとみなせる。通常、短いパルスはゲートを通して毎にだんだんと弱められてゆくので、しきい値電圧を越える時区間(実効パルス幅)が狭められてゆく。この減衰率はゲート遅延と入力のパルス幅によって決まる。

temporal(latch window) masking: フリップフロップの入力に論理反転の影響が伝搬してもその瞬間にクロックが来なければ誤った値が取り込まれることはない。ソフトエラーの起こるタイミングとクロックのタイミングには相関関係はないと考えられるので、誤った値がクロックに同期して取り込まれる確率は、クロック周期とパルス幅、およびセットアップ/ホールド時間によって決まる。

これらのうち、electrical masking と temporal masking の影響はトランジスタ回路を SPICE シミュレーションで解析することによって計算することができる。問題は logic masking である。logic masking の影響を厳密に計算するためには、全ての入力パターンと全てのエラー挿入箇所に対する論理シミュレーションを行って、どれだけの割合でエラーが伝搬しているかを数える必要がある。これは現実的には計算できないので、ランダムに生成したサンプルパターンを用いて、回路各部の信号が 0/1 になる確率と各ゲートが入力値を出力に伝搬させる確率を計算し、これらの値を用いて logic masking の影響を近似的に見積もる trace based simulation、という手法を提案している。実験結果では極めてよい見積もりということになっている^(注2)前の発表と同じグループの発表で、ともに着想からアイデアの実装、実験評価までしっかりとやっている点は高く評価できる。

2.2 Session 5C: Sequential Synthesis and FPGA Mapping

5C.1

“Exploiting Symmetry in SAT-Based Boolean Matching for Heterogeneous FPGA Technology Mapping”

Yu Hu, Victor Shih, Rupak Majumdar and Lei He (UCLA)

LUT といくつかのゲートを組み合わせた heterogeneous FPGA の場合、ある論理関数がその FPGA のひとつの論理素

(注2): 詳細は不明だが、そもそも比較相手が trace を作るのに用いているサンプルパターンでシミュレーションのような気がするので本当にこの手法でよいかは疑問。根本的には ATPG で用いられたテスト指標と同様のものなので、再収斂パスがある場合の相関の問題などがあると思われる。

子で実現可能かどうかは入力数を調べただけではわからない。そのため、Boolean matching と呼ばれる手法が必要となる。汎用の Boolean matching としては Boolean matching の条件を CNF に変換して SAT solver で解くという手法が知られているが、テクノロジマッピング中で用いるには計算時間がかかりすぎる欠点がある。この計算時間を増大させているひとつの要因は入力順序の入れ替えを考慮しなければならないことであるが、実際には多くの変数の間に対称性がある場合が多く、対称性のある変数は区別する必要がない。そこでこの論文では変数の対称性を考慮して考慮すべき入力順序を制約することによって SAT の実行時間を大幅に削減している。

個人的にはこのテーマは昔考えたことがあるが、最終的にテクノロジマッピングで Boolean matching を用いることの最大の欠点は多入力のカットを列挙することであって、たとえ Boolean matching が高速化されても試さなければならないカットの候補が莫大になるのでそのままでは使い物にならないと思われる。(5C.2 参照)

5C.2

“Combinational and Sequential Mapping with Priority Cuts”

Alan Mishchenko, Sungmin Cho, Satrajit Chatterjee, and Robert Brayton (UCB)

LUT 型 FPGA 用テクノロジマッピングでカットサイズが大きくなると考慮しなければならないカットが多くなるので、カットに適当な指標をつけて、上位 C 個のカットのみを保持することで、必要な計算時間とメモリ量を削減する手法の提案。ほかにもいろいろと細かな工夫がしてある。もちろん、カットを捨てることで解の最適性は損なわれるが、もともと最適解が求まるわけではないし、そもそも大きなサイズのカットをすべて列挙してそれをすべて保持すること自体不可能な場合があるので実用的なヒューリスティックではある。

ただし、この手法では LUT 段数最小解すら見逃すことがある。FlowMap [1] で述べられているように、ただの LUT 段数最小解はカットの列挙を行わずに求めることができるのでこの点に関しては工夫が必要であるように感じた。

5C.3

“A general model for performance optimization of sequential systems”

Dmitry Buftov, Jordi Cortadella (Univ. Polit'ecnica de Catalunya)

Mike Kishinevsky (Intel Corp.)

Sachin Sapatnekar (University of Minnesota)

retiming, c-slow retiming, recycling という順序回路に対する 3 つの異なる回路変換の解を求めるための統合的モデルの提案。retiming はフリップフロップの位置を前後に動かすことでクロック時間を最適化する手法。c-slow retiming は内部クロックに対して外部入力の印加タイミングを c 倍遅くするとい

うもの。recycling はバッファ(フリップフロップ)を挿入することでクロック時間を最適化するもの。回路の実現方法を FPGA と考えた場合、多くの FPGA では LUT の出力にフリップフロップが用意されているのでこのような retiming や recycling は簡単に実装できる。この論文では、従来は別々に考えられてきた問題を一般化して一つの枠組みで書き表したところに意義がある。

3. 論理合成、検証関係の論文紹介

3.1 Session 1D: Synthesis and Verification of Quantum Circuits

1D.1

“Fast Exact Toffoli Network Synthesis of Reversible Logic”

Robert Wille, Daniel Grosse (University of Bremen)

n 入力 n 出力関数で、入力と出力の対応が 1 対 1 のものを reversible function と呼ぶ。reversible function はエントロピーを変化させないので、回路を工夫すれば消費エネルギーを低減できるとか、量子計算モデルに適用できるとか言われている(実際に今それらが実現可能かというところでもないが)。

この論文と次の論文はこの reversible function の論理合成に関するもので、ともに与えられた論理関数を実現する Toffli Network という形式の回路を生成する。Toffli Network とは Toffli gate とよばれる基本素子をかスケード接続するもので図 1 のような形で表せる。図中の破線の長方形が Toffli gate に対応している。

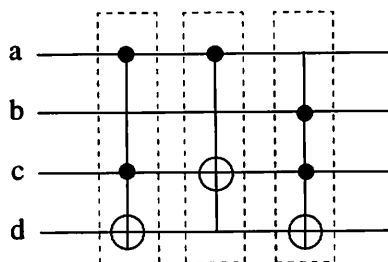


図 1 Toffli Network

もっとも左の Toffli gate では $ac \oplus d$ という計算を行う。このうち、 a, c はコントロール線と呼ばれ、このゲートでは値を参照するだけに用いられ、Toffli gate の前後で値は変わらない。 d はターゲット線と呼ばれ、このゲートの後で d の値が $ac \oplus d$ で置き換えられる。generalized Toffli gate ではコントロール線の数に制限はない。

この論文ではゲート数最小の Toffli network を求める厳密アルゴリズムの高速化について述べている。簡単にいうと、従来はゲート数 k の Toffli network が存在するかどうかを CNF の形で記述して、それを SAT solver で解いていたものを、専用の分枝限定法で解くようにしたので速くなった、というもの。

とはいえ、扱える入力数が 6 入力程度で、ゲート数が 9 程度

なので、通常の組み合わせ回路の論理合成でいうと 70 年代にあったゲート数最小の NOR ネットワークを求める問題を ILP で解く、という話を彷彿させる。

1D.2

"A Novel Synthesis Algorithm for Reversible Circuits"
Mehdi Saeedi, Mehdi Sedighi, Morteza Saheb Zamani
(Amirkabir University of Technology)

こちらは前の論文のような search-based アプローチではなく、逐次的に回路を作っていくヒューリスティックの話。search-based に比べて計算複雑度的に高速といっているが、そもそもできた解が最小解になっていないような気がする。会場からの質問で search-based のほうも実際の計算時間は結構速いがあるので、計算複雑度だけでなく実際の時間でも比較したほうがよいとコメントがあった。

1D.3

"Checking Equivalence of Quantum Circuits and States"
George F. Viamontes (Lockheed Martin ATL), Igor L. Markov, and John P. Hayes (University of Michigan)

タイトルの通り量子計算回路の等価検証を行うアルゴリズムの話。論理関数に対する 2 分決定グラフのようなものの量子計算版 (QuIDD と呼んでいる) を提案して、それを用いて等価検証を行っている。会場からの質問もなく、たぶん、中身がわかっていて人があまり多くない感じの発表。

3.2 Session 2C: Logic Synthesis

2C.1

"Incremental Component Implementation Selection:
Enabling ECO in Compositional System Synthesis"
Soheil Ghiasi (UC Davis)

Component Implementation Selection Problem (CISP) と呼ばれる問題をインクリメンタルに解くアルゴリズムの提案。CISP の例として、論理回路のネットワークを現す DAG の各ノードを、論理的には等価だが遅延や消費電力の異なるライブラリセルの中から最適なものを選ぶ問題などがある (離散的なゲートサイジング、いわゆる gate selection 問題)。この問題に対してはさまざまな手法が提案されているが、この論文はそれをインクリメンタルに行うときに効率のよいアルゴリズムを提案している。

2C.4

"Enhancing Design Robustness with Reliability-aware Resynthesis and Logic Simulation"
Smita Krishnaswamy, Stephen M. Plaza, Igor L. Markov, and John P. Hayes (University of Michigan)

ソフトエラーに対する耐性を持った回路を合成するための

手法とソフトエラーの論理レベルでのモデル化とシミュレーション手法の話。基本的な着眼点は前述の 3A.2 と同じである。ただし、この論文では logic masking のみを考慮しており、electrical masking と temporal masking は一定の割合で確率を下げることで近似するとしている。前述のように logic masking を厳密に計算するためには、 2^n のすべての入力ベクトルに対する単一縮退故障シミュレーションが必要となる。さすがにこれは実用的ではないので、ランダムシミュレーションに基づいて確率を推定する手法を用いている (いわゆるモンテカルロシミュレーション)。さらに故障シミュレーションにも少し工夫がしてある。通常の故障シミュレーションの場合は、すべての故障箇所 (厳密にはすべてのファンアウトフリー領域の出力) に関して、その場所の故障の影響が出力まで伝播するかを調べるため、回路サイズの 2 乗に比例した手間がかかる。ここでは、外部出力側から入力方向に向かって、与えられたボタンで各ゲートの入力値がブロックされるか否かを計算することで回路サイズに比例した手間で回路各部における logic masking の影響を計算するというものである。もちろん、この計算手法には穴があって、回路中に再収斂パスがある場合にはこの計算結果は正しくない。このことは論文にも触れられているが、そもそも確率的なシミュレーションなので計算時間を優先して誤差に関しては目をつぶっている。実はこの手法は目新しいものではなく、多段論理最適化手法の一つである transduction 法 [2] で提案されている MSPF (maximal set of permissible functions) と CSPF (compatible set of permissible functions) の計算法に非常に似通っている。具体的には回路全体の計算方法としては CSPF と同様の手法を用いて、ゲートの出力からゲートの入力へ至るパスの可観測性 (ブロックされるか否か) の計算を MSPF と同様に行っている。

ソフトエラーに対する合成手法としては 2 つの方法を提案している。ひとつは covering-based resynthesis と呼ばれるもので、エラーの影響を伝播しやすいゲートに対して冗長な入力を新たに加えることで、エラーが伝播する確率を下げている。冗長な入力の追加には SAT ベースの等価検証アルゴリズムを用いている。ふたつめは local rewriting と呼ばれるもので、例えば 4 入力以下の部分回路 (複数のゲートを含む) を列挙して (4LUT に対するテクノロジマッピングのときに用いられるカット列挙と同様のアルゴリズムを用いる)、その部分回路と等価で、エラー確率の低い部分回路に置き換える、という処理を繰り返す。置き換える候補の 4 入力の論理関数はあらかじめ用意しておく。

組み合わせ論理回路に対するソフトエラー耐性を考えた論理合成というテーマは目新しいものと思われ、ここで提案されている個々の手法以外にもさまざまなアイデアが考えられるのではないと思われる。ただし、単一のソフトエラーに対する絶対的な耐性が必要であるならば、回路を三重化して多数決論理をとる (Triple Modular Redundancy) などの工夫が必要であると思われ、エラー確率を下げてでもゼロにはできない合成手法がどれだけ実用上の意味を持つかは疑問である。

3.3 Session 4A: High Quality Test Cases for Verification

4A.1

“An Incremental Learning Framework for Estimating Signal Controllability in Unit-Level Verification”

Charles H.-P. Wen (National Chiao-Tung University),
Li-C. Wang (UC Santa Barbara), Jayanta Bhadra
(Freescale Semiconductor Inc.)

論理関数に対する入出力応答のサンプルからもとの関数を模倣するモデルを作るといういわゆる機械学習の話。題名にある learning はテストパターン生成や SAT で用いられているものではない。参考文献をみてもどちらかというと EDA が専門というより機械学習が専門の人が論理設計を応用にしているように見受けられる。

k 個のサンプルからもとの関数を推測するということは 2^n 個の最小項のうち、 k 個の最小項のみ定義された不完全定義論理関数からそこに含まれる論理関数をひとつ選ぶということであるが、論理数学の世界で言えば、その 2^{2^n-k} 個の論理関数の間に何の区別もないので、どの関数が適切でどの関数が不適切という基準はそもそもないのである (積和形論理の最小化などの指標を持ち込まない限り)。そのため、この研究テーマは個人的にはあまり興味を覚えない。通常のデータマイニングはもとになるモデルを厳密に定義することができないのでモデル化するのであって、論理回路という厳密なモデルが存在するものを学習して近似モデルを作るという発想が研究のための研究になっているように思える。

4A.2

“Stimulus Generation for Constrained Random Simulation”

Nathan Kitchen and Andreas Kuehlmann (UCB, Cadence Research Labs)

制約つきランダムパターン生成手法の話。ただ単にランダムにパターンを生成して、それが制約を満たしていないときに捨てるのではひとつのパターンを生成する時間が最悪の場合、指数爆発する (いわゆる NP 問題)。制約を満たすように乱数の系列を調節しても、生成したパターンの分布が偏っている場合には、検証に必要なパターンの数が指数爆発する。従来手法のほとんどが生成されるパターンの分布の一様性に問題があった。この論文では統計科学の分野ではよく知られたマルコフ過程モンテカルロ法を応用している。具体的には Gibbs sampler を少し改良して、いったん制約外への動きを許すようにしている。そのために制約を満たす解がないときでも、次の move で制約充足解に到達できるようにサンプル生成のための確率分布関数を修正している。0/1 の離散数学の世界だった論理検証がだんだんと統計科学の世界になってきた感じ。ただ、私の理解不足でよくわからないが、このような修正を加えても出力分布の一様性がちゃんと得られているのかは疑問。

4A.3

“Probabilistic Decision Diagrams for Exact Probabilistic Analysis”

Afshin Abdollahi (UC Riverside)

ゲートの出力が一定の確率で正しくない (否定の値をとる) と仮定した回路の出力の 1 になる確率を計算するという話。その計算のために、2 分決定グラフを改良した probabilistic decision diagram (PDD) を提案している。この PDD の肝は 2 分決定グラフで用いられている否定枝を確率 p で値を否定する枝に取り替えていること。この確率的論理回路モデルは前述のソフトウェアが多重に起こると仮定したときのモデルとなるので、新しい研究分野を切り開くものになる可能性はある。たしかにこの PDD を作ることができれば評価にかかる計算時間はとても短くてすむが、すくなくともこの PDD は BDD よりも共有しにくい (枝に確率がついているので同形になりにくい) のでメモリ爆発の危険がより大きく、スケーラビリティの問題があると思われる。

4A.4

“Computation of Minimal Counterexamples by Using Black Box Techniques and Symbolic Methods”

Tobias Nopper and Christoph Scholl and Bernd Becker
(Albert-Ludwigs-University)

モデルチェッキングでプロパティを満たしていないとわかったときに、エラーとなる状態に導く反例 (counterexamples) を生成することはデバッグのために必要だが、この論文では回路の一部がブラックボックスで中の挙動がわからない (ブラックボックスの出力値が非決定的に決まる) と仮定したときの反例を求める手法を提案している。ブラックボックスは実際にその回路の中身がわからないというよりも、その部分を非決定的なモジュールに置き換えることでモデルを簡単にする目的で用いられる。そのため、どの部分をブラックボックスにするかは設計者や検証技術者が指定する必要がある。

4. CADathlon

ICCAD の特徴的なイベントとして CADathlon (<http://sigda.org/programs/cadathlon/>) と呼ばれるものがある。これは EDA 分野におけるいわゆるプログラミングコンテストであり、概要は以下のとおりとなっている。

- 対象は大学院の学生で、2 名 1 組のチームを作って参加する。2 名が同じ所属機関である必要はない。
- プログラミングは Linux が走る Intel ベースの PC 上で行う。ハードウェアは会場で提供されているものを用いる。ノート PC を持ち込んだり、インターネット接続をすることは禁止されている。
- コンテストの 1 週間程前に、対象となる CAD エリア、関連論文、EDA 用の基盤ソフトウェアなどが提供される。WEB では明示していないが OA-Gear (<http://openedatools.si2.org/oagear>) のことだと思う

れる。

- コンテスト当日 (11 月 5 日) に問題とテストデータが与えられ、参加者はその問題を解くプログラムを開発する。ただし、最終判定に用いるデータは提示されない。

- より多くの問題を少ない計算時間で解いたチームが勝者となる。ただし、プログラムが誤った答を出した場合には減点される。

今回の問題は以下の分野から出題されている。

- Circuit Design & Analysis
- Physical Design
- Logic & High-Level Synthesis
- System Design & Analysis
- Functional Verification
- Timing, Test, and Manufacturing

今回は台湾からの参加者が多く、優勝者も National Taiwan University の学生であった。ここでも最近の台湾の躍進ぶりがうかがえる。聞くところによると、台湾大では国内で参加者のスクリーニングを行って優秀な学生を旅費補助をつけて送り出しているようである。

CADathlon は前もって準備したプログラムを投稿するのではなく、コンテスト当日にアルゴリズムを考えて実装しなければならない。優秀なプログラムを作るためには、関連分野の広い知識、論理的思考能力、プログラミング技術などが総合的に問われることとなる。

日本における EDA 研究のアクティビティを活性化させるための具体的な手段の一つとして、この CADathlon へ参加するための事前の調査や情報交換、チュートリアルなどを希望者を募って行えばそれなりに工科あるのではないと思われる。日本からもぜひこのコンテストに参加して優秀な成績を納める若手が将来現れることを期待したい。

文 献

- [1] J. Cong and Y. Ding: "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA design", IEEE Transactions on Computer-Aided Design, 13, 1, pp. 1-12 (1994).
- [2] S. Muroga, Y. Kambayashi, H. C. Lai and J. N. Culliney: "The Transduction Method — Design of Logic Networks Based on Permissible Functions", IEEE Trans. Comput., C-38, no. 10, pp. 1404-1424 (1989).