様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ６ 月 １９ 日現在

機関番号：１２６０１
研究種目：基盤研究(B)
研究期間：2012～2014
課題番号：２４３０００１５

研究課題名（和文）システム間の差異に着目した検証・デバッグ手法

研究課題名（英文）Logic verification and synthesis based on difference analysis

研究代表者
藤田　昌宏（Fujita, Masahiro）

東京大学・大規模集積システム設計教育研究センター・教授

研究者番号：７０３２３５２４

交付決定額（研究期間全体）：（直接経費）　　12,500,000円

研究成果の概要（和文）：ハードウェア・ソフトウェア開発では旧設計を効率よく、かつ正しく再利用することが重要である。本研究では論理回路とソフトウェアにおいて、旧設計と新仕様の差異を部分回路や部分プログラムの変換として定式化する新規手法を考案し、評価した。提案手法により、ユーザが指定した範囲の修正で旧設計や旧プログラムを新仕様に合わせることができる。企業と共同でサーバー計算機の設計に適用し、実設計においても有効であることを実証している。また、解析はごく少数のテストパタンで検査すれば完全な検証になることも示し、従来不可能であった複数の故障や誤りを同時かつ完全に検査するテストパタンの自動生成にも成功している。

研究成果の概要（英文）：　When developing new hardware/software, it is essential to utilize existing designs efficiently and correctly. We have developed new methods which define and analyze the difference between old and new designs as transformations on sub-circuits and sub-programs. With the proposed methods, users can specify how much modifications/additions are allowed on the old designs in order to meet new specifications. Appropriate transformations are automatically identified with very small numbers of test patterns. With joint efforts with industry, practical usefulness of the proposed methods have been shown by applying them to the real server machines designs. Moreover, the proposed methods can be used as automatic test patterns generation methods for various types of multiple faults or bugs in the design, and for the first time, complete sets of test patterns for multiple faults have successfully been generated on the circuits having more than 10,000 gates.

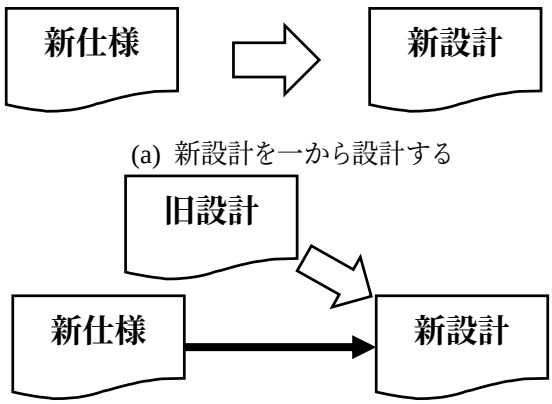
研究分野：ハードウェア・ソフトウェア自動設計技術

キーワード：形式的解析　設計自動合成　設計デバッグ　差異抽出　論理関数解析　プログラム解析

１．研究開始当初の背景

半導体技術の飛躍的な進歩は続いており、1 チップ内に 1 億ゲート以上を集積する技術が実用化されている。一方、企業間の競争から、ハードウェアシステム開発に許される設計期間と人的労力を増やすことが難しいため、従来より数倍大きい設計を従来より早く設計する必要に迫られている。このため、図 1(a)に示すような、新規設計を一から新しく行っていく設計手法は、現実的ではなくなっている。そこで、同図(b)に示すように、旧設計を一部修正・拡張して新設計を構築する、設計再利用技術が非常に重要になっている。一般に、現在の大規模半導体（System on Chip, SoC）では、設計の 80%以上は、既設計の再利用であると言われている。

設計再利用技術は、今までから種々の形で研究され、また取り入れられて来ているが、その多くは、既設計を IP（Intellectual Property）としてデータベース化し、そのまま新設計で一部利用するという形を基本として行われている。その際、新設計で旧設計を一部修正する必要がある場合には、その修正は人手で行われており、手間がかかるとともに、人手による間違いも生じやすく、設計効率向上がなかなか進まない。修正時における設計バグの混入により、設計期間が大幅に延びてしまうことも少なくない。設計を一部修正する場合でも、効率よく、かつ正しく再利用できる技術や設計手法が強く望まれている。



(a) 新設計を一から設計する

(b) 旧設計を修正しながら再利用することで新設計を設計する

図１　設計再利用

２．研究の目的

設計再利用を効率よく、かつ正しく行えるようにするための設計手法に関し、旧設計をどのように修正・追加して新仕様に合わせるかという立場から、新規設計手法を考案し評価する。具体的には図 2 に示すように、旧設計と新仕様間の差異に注目し、その差異を解消するように、旧設計に対してユーザが指定した変換の組み合わせを適用することで、新仕様に対する設計を自動的に生成することを目標とする。変換量や範囲はユーザが指定できるため、ユーザの意図通りの設計再利用が可能となり、様々な設計条件を満たす設計が可能となる。また、多くの場合、人手介入なしに自動的に再利用できるようになり、設計効率が大幅に改善されるとともに、自動合成された新設計の正しさも形式的に（100%）保障される。

以上の理解のもと、本研究では、(1)ハードウェア論理回路設計段階での支援技術と、(2)ハードウェアの動作の設計やソフトウェアの開発支援技術の 2 つの互いに補完的な技術を研究開発する。

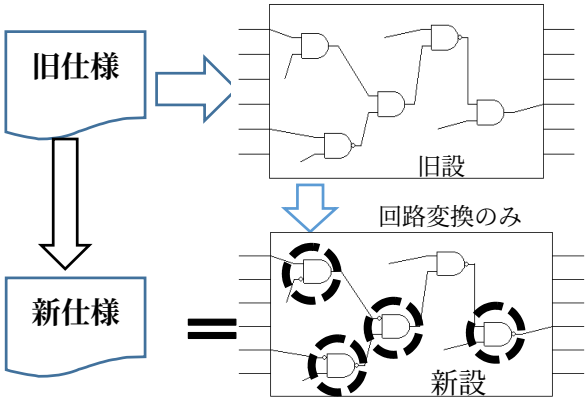


図 2　旧設計の回路変換で新仕様を実現

３．研究の方法

旧設計と新仕様間の差異を抽出し、回路変換によって両者を等価にする技術に関し、次の 2 つについて研究開発を行う。

(1) ハードウェア論理設計（論理回路レベルの設計）において、旧設計の論理回路に対する回路変換をユーザが部分回路（あるいは個々のゲート）の変換の集合として定義する。新仕様と等価となる変換の組み合わせを自動的に生成する新規手法

(2) ハードウェアの動作を設計する段階（ハードウェアの動作を C 言語などプログラミング言語で記述する設計段階）、あるいはソフトウェア開発において、旧仕様に対応するプログラムやハードウェア設計と、新仕様に対応するプログラム間のマッチングを取り、差異部分を生成する手法

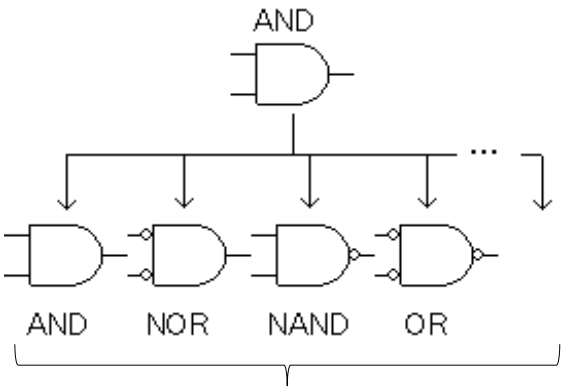
４．研究成果

一般に設計再利用問題は、数学的には QBF（Quantified Boolean Formula）として定式化できることが分かっている。本研究で取り組む、複数の回路変換を施すことで旧設計を

新仕様に合わせる問題に対しては、「適切なある回路変換を行うことで、すべての入力に対して新仕様と同じ出力値をなるようにできる」という定義となり、「ある回路変換」と「すべての入力」という、Existential Quantifier と Universal Quantifier の両方が現れる問題となる。これは QBF 問題と呼ばれている。従来の QBF 問題に解くプログラム（ソルバーと呼ばれる）は、Existential Quantifier のみが現れる SAT 問題を解く SAT ソルバーを比較して、極めて小さな問題しか解けなかった。SAT ソルバーは一般に数百万変数の問題でも解ける場合が多いが、QBF ソルバーは数千変数でも解けないことが多い。本研究に関する成果の１つとして、QBF 問題を複数の SAT 問題をインクリメンタルに解く新規手法を考案し、設計再利用に適用した。この新手法により、扱える回路やソフトウェアの規模が１桁から２桁以上、大幅に向上した。結果として、企業から頂いた例題なども問題無く処理できるようになった。

以下、図を用いて提案手法の流れを説明する。

① 旧設計と新仕様の差異を表現するための部分回路変換を決定する。たとえば、変換が各ゲート単位で、各ゲートの入出力を反転する変換の場合には、図 3 に示すように 2 入力ゲートの場合には、8 種類定義される。回路変換は。ゲート単位でも、ゲートが複数集まった部分回路単位でもよく、ユーザが目的によって自身で自由に決められる。また、使い勝手を向上されるため、いくつかのよく利用される変換は、ライブラリとして予め定義されており、ユーザをそれをそのまま、あるいは追加・修正して利用することができる。



変換後、8 種類の論理関数

図 3 回路変換の例

② 図 4 に示すように、新仕様と一致しない回路変換を検出できる入力値 in_1（反例と呼ぶ）を求める。これは、従来の SAT 問題として定式化できるため、非常に高性能な SAT ソルバーを利用して求めることができ、100 万変数以上でも処理可能である。次に求まった反例 in_1 では正しい出力が得られるという条件で、他の反例 in_2を求める。これも同様に SAT 問題である。以下、これを反例が見つからなくなるまで繰り返し、得られた反例の集合を $\{in_1, in_2, \cdots, in_N\}$とする。実験により、N は大規模回路でも数百程度で済むことが分かっている。

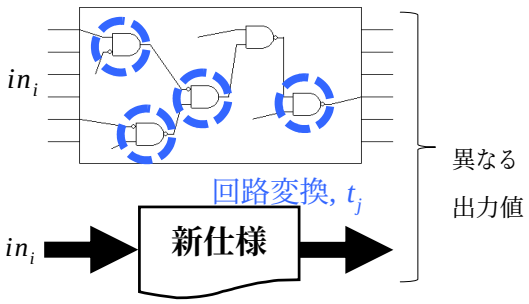


図 4 反例を生成する

③ 図 5 に示すように、得られた反例の集合を$\{in_1, in_2, \cdots, in_N\}$すべてについて、正しい出力が得られる回路変換を対応する SAT 問題を解くことによって求める。もし解がない場合には、新仕様を満たす回路変換は存在しないことになるので、新たな回路変換を定義する必要がある。一方、得られたすべての解は、正しい回路変換を示しているので、実装上の条件などから、解を必用に応じて取捨選択して利用する。

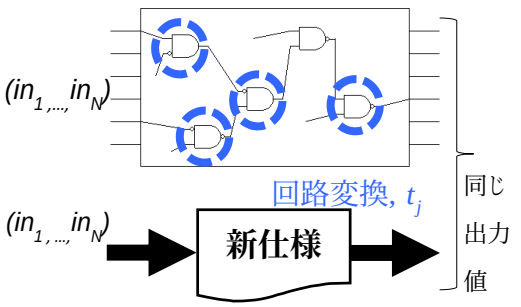


図 5 反例をすべて満たす解を探す

また、(2)のハードウェアの動作のプログラミング言語による記述や、一般のソフトウェアの差異を自動的に抽出する手法も新規に考案し、実装・評価した。これは、プログラム中のデータや制御依存をグラフの形で表現し、2 つの比較すべきプログラムから得られる 2 つのグラフのトポロジーを比較することで差異を求めるものであり、変数名や実行順序などが異なっても同じ動作をする場合には、自動的に差異はないと認識できる。プログラミング言語 C に対するものを実装し、主に組込みソフトウェアに対して評価を行った。

以上を評価した結果、以下の成果が得られた。

(1) 設計中の少数（数十から数百程度）の部分回路の内容が不明、あるいは間違っているような設計に対して、別途与えられた新仕様を満たすように、部分回路の内容を自動的に決定できることを、企業から得た実用回路を用いて実証した。その際、種々の回路変換を適用してみることで、ユーザの意図に沿った回路を生成できることも示した。

(2) (1)において、部分回路を完全に決定するために必用な入力値の総数（解析の回数、上のNに相当）は、大規模回路でも数十から数百で十分であることを実験で示した。このため、提案手法は大規模回路にも適用できることになり、実用的価値は高い。

(3) 提案手法を応用することで、与えられた設計中に現れる内部変数間の論理的な関係を自動的に抽出できることを示した。これら関係は設計に対する一種のコメントであり、それらから、設計者は自身の設計の動作の確認を行うことができる。また、設計が満たすべきアサーションとしても利用することができ、設計がさらに最適化のために修正された際の、設計検証を効率化することに利用できる。

(4) 部分回路に設計バグがあるのではなく、故障しているため動作が意図とおりで無い場合も、数学的には同様に定式化でき、かつ提案手法が応用できることを示した。そして回路中に複数の故障がある場合に対する、故障の有無を検出する完全なテストパタンの自動生成が可能であることを示した。従来は数十ゲート程度とごく小規模な回路以外は全く不可能であったが、本手法により 10,000 ゲート規模でも自動生成できることを実証した。また評価の結果、複数故障がある場合でも、その多くは、故障が１つしか存在しないと仮定して生成したテストパタンで検出できることが示された。さらに解析した結果、故障が１つしか存在しないとして生成したテストパタンに一定の規則でパタンを追加することで、さらに多くの複数の故障を検出できることを示した。故障が１つのみ存在するとしてテストパタン生成技術は非常に進歩しており、数百万ゲート規模でも処理可能となっている。その手法を一部拡張することで、複数の故障に対応できることなり、今後の発展がさらに期待できる技術である。

(5) 提案しているソフトウェア同士を比較し差異を自動抽出する新規手法を組込みソフトウェア間の差異抽出に適用し、具体的にどのように変換すれば互いに等価になるかを自動的に示せることを実証した。これにより、ユーザは、差異の内容を把握して、プログラムの拡張を進めていくことができ、結果として著しい信頼性の向上が期待できる。従来の等価か不等価かという二者択一の理解ではなく、どのように差異があり、結果としてどのように不等価となっているかを一定の形式で示すことができ、応用範囲は広い。

(6) 提案手法をマイクロプロセッサのアーキテクチャにおける自動設計バグ修正に適用し、最先端アーキテクチャである、out-of-order, super scalar プロセッサが自動修正できることを実証した。これにより、先端プロセッサをさらに拡張する場合などでは、自動的に設計を生成できるようになる。

(7) 制御は比較的簡単であるが、複雑な計算を行うハードウェアやソフトウェアに提案手法を適用し、自動的にバグ修正ができることを実証した。従来、このような設計を効率的に扱う手法はなく、本手法はその第一歩であると言える。

(8) 提案手法が、ハードウェアやソフトウェアの遅延時間や計算時間のテストにも適用できることを実証した。遅延故障の扱いは重要となってきており、今後発展が期待できる。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 2 件）

[1] Amir Masoud Gharehbaghi, Masahiro Fujita: Automatic Rectification of Processor Design Bugs Using a Scalable and General Correction Model. IEICE Trans. Inf. & Syst. E97.D(4), pp.852-863, 2014.

[2] Satoshi Jo, Takeshi Matsumoto, Masahiro Fujita: SAT-based Automatic Rectification and Debugging of Combinational Circuits with LUT Insertions. IPSJ T. on System LSI Design Methodology, Vol.7, pp.45-55, 2014.

〔学会発表〕（計 9 件）

[3] Satoshi Jo, Takeshi Matsumoto, Masahiro Fujita: SAT-Based Automatic Rectification and Debugging of Combinational Circuits with LUT Insertions. Asian Test Symposium, Niigata, Japan, Nov. 2012.

[4] Kosuke Oshima, Takeshi Matsumoto, Masahiro Fujita: A debugging method for gate level circuit designs by introducing programmability. IFIP/IEEE International Conference

on Very Large Scale Integration, Istanbul, Turkey, Oct. 2013.
[5] Masahiro Fujita, Takeshi Matsumoto, Satoshi Jo: FOF: Functionally Observable Fault and its ATPG techniques. IFIP/IEEE International Conference on Very Large Scale Integration, Istanbul, Turkey, Oct. 2013.
[6] Masahiro Fujita, Satoshi Jo, Shohei Ono, Takeshi Matsumoto: Partial synthesis through sampling with and without specification. International Conference on Computer-Aided Design, San Jose, USA, Nov. 2013.
[7] Somayeh Sadeghi Kohan, Payman Behnam, Bijan Alizadeh, Masahiro Fujita, Zainalabedin Navabi: Improving polynomial datapath debugging with HEDs. European Test Symposium, Paderborn, Germany, May 2014.
[8] Masahiro Fujita: Variation-Aware Analysis and Test Pattern Generation Based on Functional Faults. IEEE Computer Society Annual Symposium on VLSI, Tampa, USA, July 2014.
[9] Masahiro Fujita, Alan Mishchenko: Efficient SAT-based ATPG techniques for all multiple stuck-at faults. International Test Conference, Seattle, USA, Oct. 2014.
[10] Masahiro Fujita, Alan Mishchenko: Logic synthesis and verification on fixed topology. 22nd IFIP/IEEE International Conference on Very Large Scale Integration, Playa del Carmen, Mexico, Oct. 2014.
[11] Masahiro Fujita, Naoki Taguchi, Kentaro Iwata, Alan Mishchenko: Incremental ATPG methods for multiple faults under multiple fault models. 16th International Symposium on Quality Electronic Design, Santa Clara, USA, March 2015.

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計 0 件）

〔その他〕
ホームページ等
www.cad.t.u-tokyo.ac.jp

６．研究組織

(1)研究代表者

藤田　昌宏（FUJITA, Masahiro）
東京大学大規模集積システム設計教育研究センター　教授
研究者番号：70323524

(2)研究分担者

松本　剛史（MATSUMOTO, Takeshi）
東京大学大規模集積システム設計教育研究センター　助教
研究者番号： 40536140

(3)連携研究者

なし