社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

チップ内プログラマブル配線向け形式的検証手法

田川　貴聡†　　吉田　浩章††,†††　　藤田　昌宏††,†††

† 東京大学 工学部 電子工学科
†† 東京大学 大規模集積システム設計教育研究センター (VDEC)
〒 113–8656 東京都文京区本郷 7–3–1
††† 科学技術振興機構 戦略的創造研究推進事業 CREST
E-mail: †{tagawa,hiroaki}@cad.t.u-tokyo.ac.jp, ††fujita@ee.t.u-tokyo.ac.jp

あらまし　近年の開発コストの増大に伴い、FPGA に代表されるプログラマブルなデバイスの重要性が増している。このようなデバイスの重要な構成要素としてプログラマブル配線があげられるが、主に人手で設計されるため誤りが混入しやすい。本論文ではトランジスタレベルにおけるプログラマブル配線向けの形式的検証手法を提案する。提案手法では仕様と設計を論理式で表現し、充足可能性判定問題を解くことにより検証を行う。一般的な FPGA アーキテクチャを例題として提案手法の適用可能規模の評価を行い、また実設計に対して提案手法を適用することで検証した例を示す。

キーワード　プログラマブル配線、形式的検証、充足可能性判定問題、FPGA

A Formal Verification Method for On-Chip Programmable Interconnect

Takaaki TAGAWA†, Hiroaki YOSHIDA††,†††, and Masahiro FUJITA††,†††

† Dept. of Electronic Engineering, University of Tokyo
†† VLSI Design and Education Center(VDEC), University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656, Japan
††† CREST, Japan Science and Technology Agency
E-mail: †{tagawa,hiroaki}@cad.t.u-tokyo.ac.jp, ††fujita@ee.t.u-tokyo.ac.jp

Abstract　As the development cost increases, programmable devices such as FPGAs are becoming critically important. A key component of such programmable devices is a programmable interconnect. Typically they are designed with full-custom design methodology and hence it is likely to have design errors. In this paper, we propose a formal verification method for on-chip programmable interconnect at the transistor level. We present a scalability analysis of the proposed method and also demonstrate that the proposed method successfully proves the correctness of an actual VLSI design.

Keywords　Programmable interconenct, formal verification, satisfiability problem, FPGA

1. はじめに

半導体製造技術の進歩によって LSI 製造プロセスの微細化が進み、1 チップ上により大規模な回路を製造することが可能になっている。これに伴い、製造・開発にかかるコストは増大しており、設計の誤りにより製造したチップが動作しない場合、チップを製造しなおすために莫大なコストがかかってしまう。このような背景の下、FPGA (Field Programmable Gate Array) に代表される再構成可能な素子の重要性が増している。FPGA では、論理機能・配線構造を再構成して論理回路を実装することができ、実装した設計に誤りがあっても書き換えが可能なため、設計者はコストを気にせずに試行錯誤しながら開発を行うことができる。製造後のチップにおいて、電気的な接続関係を変更できるプログラマブル配線は、プログラマブル素子の重要な要素のひとつであり、FPGA [1] や動的再構成可能プロセッサ [2]〜[4] など様々なプログラマブル素子に使われている。今後さらなる製造・開発コストの増大が予想されるため、プログラマブル配線はさらに重要になると考えられる。

FPGA に代表されるプログラマブル素子は汎用的な用途であるという性質上大量生産されるため、可能な限り高密度なレイアウトをするためにフルカスタム設計が用いられる。現在 FPGA は大きいものでは数億トランジスタ規模のものも使われており、設計は大規模なものとなっている。一般的な ASIC 設計では抽象度の高いハードウェア記述言語を用いて設計を記述し自動設計ツールを用いて設計を行うが、一方でフルカスタム設計ではトランジスタ等の素子レベルで人手で設計を行うため、その設計は非常に複雑となり、結果的に誤りが混入しやすくなる。現在フルカスタム設計の検証は大部分をシミュレーションに頼っているが、大規模化・複雑化とともにシミュレーションベース検証は限界に来ており、形式的検証の重要性が増している。フルカスタム設計においても、論理回路部分であれば、トランジスタレベルネットリストから論理回路を抽出し、形式的検証を適用することは可能である。フルカスタム設計に対する形式的検証手法は主に、トランジスタレベル回路から論理レベルの回路に抽象化を行い、従来の論理回路に対する形式的検証手法を適用する手法が一般的となっている [5]。FPGA の設計に対する形式的検証手法として、トランジスタレベル設計から数学的モデルを構築してレジスタ転送レベル記述に抽象化する手法が提案されている [6]。この手法は主に FPGA の論理部分の検証を対象としており、プログラマブル配線の検証は考慮していない。プログラマブル配線は何らかの計算ではなく接続関係のみを変更するものであるため、従来の論理回路検証手法ではあまり考慮されてこなかった。また、プログラマブル配線にはクロスバ方式 [2]、マルチプレクサ方式 [3]、バス方式 [4] など様々な機構が存在し、これらをどう扱うかという問題も存在する。このような背景の下、プログラマブル配線の検証手法の研究開発が重要となっている。

本研究では、プログラマブル配線の形式的検証を行うことを目的とする。プログラマブル配線とは外部入出力端子を複数持ち、その間の接続関係が与えられた構成情報によって決められるものである。構成情報はある決められた範囲を持つ値で与えられ、その範囲内における各値はある接続関係に対応する。この対応関係をプログラマブル配線の仕様と呼ぶ。仕様において取りうる構成情報の集合が可能な全ての接続関係の集合を表している。ここで、必ずしも任意の接続関係が実現可能であるわけではないことに注意する。本論文では構成情報は複数ビットからなる 2 進数値によって与えられるものとする。プログラマブル配線の検証とは、仕様上可能な全ての構成情報に対して、設計が仕様通りの接続関係を満たしていることを証明することである。今回の研究では、問題を簡単化するため 2 端子間のみを接続する構成情報の集合について検証を行うことを目的とする。

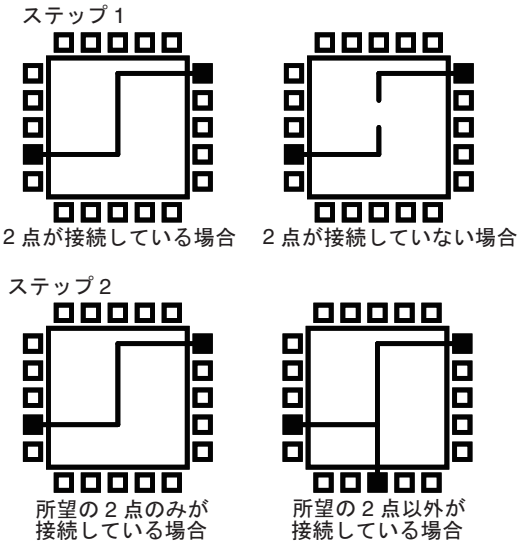


図 1　スイッチブロックで可能な配線

2. 充足可能性判定に基づくプログラマブル配線の形式的検証手法

2.1 提案手法の概要

プログラマブル配線とは複数の外部入出力端子を持ち、構成情報によって外部入出力端子間の接続が決められるものである。構成情報はある決められた範囲を持つ値であり、各値はある接続関係に対応する。この対応関係をプログラマブル配線の仕様と呼ぶ。本提案手法では、仕様上可能な接続関係の集合の中でも 2 端子間のみを接続するような部分集合について、設計が仕様通りの接続関係を満たしていることを証明する。

本提案手法では、この検証問題を充足可能性判定問題 (SAT 問題) として定式化する。SAT 問題は、和積標準型 (CNF) で表現されたある論理関数の値を 1 にする変数の割り当てが存在するかを判定する問題である。例えば、$f = (\overline{a} + b)(a + \overline{b})$ (+ は論理和) では、$a = 1, b = 1$ のときに $f = 1$ となるので、この論理式は充足可能であると判定され、$f = a\overline{b}(\overline{a} + b)(a + \overline{b})$ では、$f = 1$ となるような変数の割り当ては存在しないので、充足不可能であると判定される。SAT の研究は盛んに行われており、Chaff [8] や MiniSat [7] など SAT 問題を解くためのツールである SAT ソルバの研究も進んでいる。現在最も高速なソルバのひとつである MiniSat では、100 万変数程度の問題も解くことができる。

本手法ではまず、プログラマブル配線の構成情報や信号値に対応する二値変数を用意し、仕様と設計に対する制約式をそれぞれ論理式で表現する。次に、仕様に対応する論理式は構成情報と入出力接続関係の対応が正しいときに真となり、また設計に対応する論理式は仕様における接続関係を満たさないときに真となるように構築する。これら 2 つの制約式を合わせて SAT 問題として解く

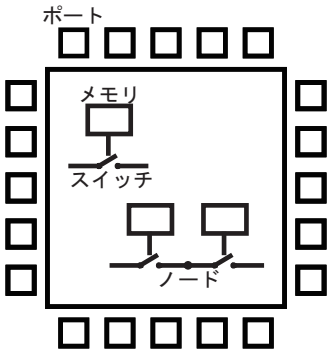


図 2　仕様モデルの概要

ことにより検証を行う。設計誤りが存在する場合、つまり仕様における接続関係を設計が満たさない場合にはこの SAT 問題は充足可能となる。充足可能となる変数の割り当ては設計誤りとなる構成情報を表している。充足不可能であれば、設計は仕様を満たすことが証明される。また、検証問題は図 1 に示すように 2 つのステップに分かれており、まず最初のステップでは所望の 2 点が接続していることを、また次のステップではそれ以外の点と接続していないことを証明する。

2.2　仕様に対する制約式の構築

検証対象とするプログラマブル配線の仕様は以下のような仕様モデルによって与えられるものとする。仕様モデルは、図 2 のように、外部入出力端子、スイッチ、配線、メモリで構成する。スイッチは 3 端子で、1 つは制御信号を入力する端子であり、制御信号によって他の 2 端子の接続関係が決まる。制御信号が 1 のとき、スイッチは ON になり、制御信号が 0 のとき、スイッチは OFF になる。スイッチ同士は配線で接続される。メモリは 1 ビットの値を記憶するもので、これが制御信号の値を決める。メモリの集合が構成情報を表す。

2.1 節で説明した通り、ここでは仕様における構成情報と入出力接続関係の対応が正しいときに真となる論理式を構築する。まず、各配線および各メモリに変数を用意する。配線に対応する変数はその配線が接続に使用されるかどうかを表し、メモリに対応する変数はそのメモリが保持している 1 ビットの値を表す。最初の制約式は、スイッチが ON ならば両側の配線が接続に使用されるため、対応する変数が 1 になるという制約式である。あるスイッチの制御信号が c 、接続する配線が w_1,w_2 のときこの制約式は次のようになる。

$$S_{sw} = (\overline{c} + w_1)(\overline{c} + w_2) \tag{1}$$

$c = 1$ ならば、$w_1 = 1$,$w_2 = 1$ のときのみ $S_sw = 1$ になる。2 つ目は、配線が 1 のとき、接続するスイッチのうち 2 つが ON になるという制約式である。ある配線が w、接続するスイッチの制御信号が $c_1, c_2, \cdots, c_n$ のとき、2 つ以下のスイッチが ON になる制約式は、次のようになる。

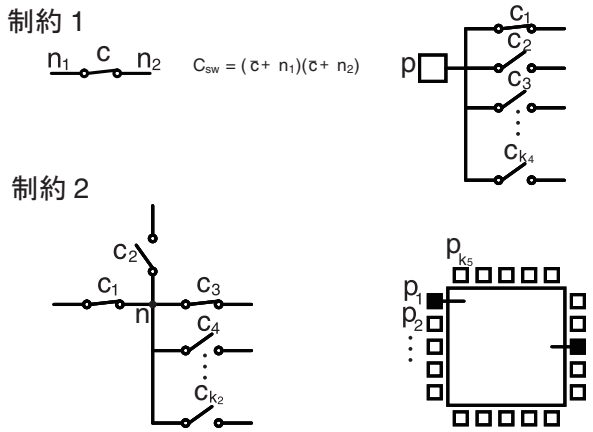


図 3　仕様に対する制約式の構築

$$S_{wl} = \prod_{i,j,k \in S \mid i \neq j, j \neq k, k \neq i} (\overline{w} + \overline{c_i} + \overline{c_j} + \overline{c_k}) \tag{2}$$

2 つ以上のスイッチが ON になる制約式は、次のようになる。

$$S_{wg} = \prod_{i}^{n} (\overline{w} + c_1 + c_2 + \cdots + c_{i-1} + c_{i+1} + \cdots + c_n) \tag{3}$$

したがってちょうど 2 つが ON になる制約式は、

$$S_w = S_{wl} \cdot S_{wg} \tag{4}$$

3 つ目は、外部入出力端子が 1 のとき、それに接続するスイッチが 1 つ ON になるという制約式である。ある外部入出力端子を t、接続するスイッチの制御信号が $c_1, c_2, \cdots, c_n$ のとき、1 つ以下のスイッチが ON になる制約式は、

$$S_{tl,1} = \prod_{i,j \in S \mid i \neq j} (\overline{t} + \overline{c_i} + \overline{c_j}) \tag{5}$$

1 つ以上のスイッチが ON になる制約式は、

$$S_{tg,1} = (c_1 + c_2 + \cdots + c_n) \tag{6}$$

したがってちょうど 1 つのスイッチが ON になる制約式は、

$$S_{t,1} = C_{tl} \cdot C_{tg} \tag{7}$$

4 つ目は、2 つの外部入出力端子が 1 になり、他の外部入出力端子が 0 になるという制約式である。これで 2 つの外部端子のみが接続していることになる。端子を $t_1, t_2, \cdots, t_n$ とすると、2 つ以下の端子が 1 になる制約式は、

$$S_{tl,2} = \prod_{i,j,k \in S \mid i \neq j, j \neq k, k \neq i} (\overline{t_i} + \overline{t_j} + \overline{t_k}) \tag{8}$$

2 つ以上の端子が 1 になる制約式は、

$$S_{tg,2} = \prod_{i}^{n} (t_1 + t_2 + \cdots + t_{i-1} + t_{i+1} + \cdots + t_n) \tag{9}$$

2 つの端子だけが 1 になる制約式は、

$$S_{t,2} = S_{tl,2} \cdot S_{tg,2} \tag{10}$$

となる。これら 4 つの制約式を同時に満たす、つまり 4 つの制約式の論理積を充足する変数の集合が、すべての可能構成情報や接続関係の集合となる。

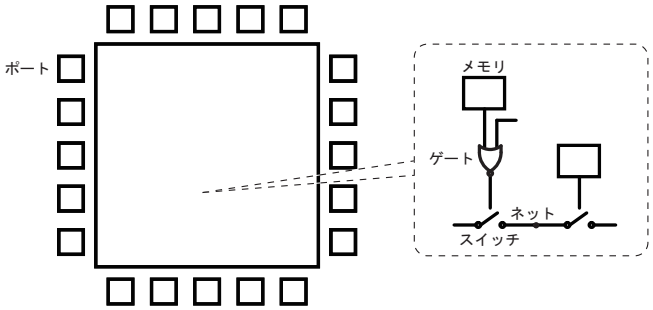


図 4　設計モデルの全体像

2.3　設計に対する制約式の構築

設計はマスクパタンから抽出したトランジスタレベル回路として与えられるとする。まず最初にこの回路の抽象化を行い設計モデルを構築する。ここで、設計モデルは図 4 に示すようにメモリとスイッチと論理ゲートからなるものとする。トランジスタレベル回路から SRAM セルや論理ゲート、スイッチとしてのトランジスタを検出し、設計モデルを構築する。トランジスタレベル回路からこれらの構成要素を検出する手法としては、既存手法 [5], [6] を利用する。

2 端子を枝分かれすることなく接続するという仕様を満たさない場合、1 となり、設計誤りとして判定される制約式を考える。設計誤りは 2 つのケースに分けて考える。1 つは 2 端子が接続されない場合、もう 1 つは 2 端子は接続されるが、他の端子とも接続される場合である。まず、2 端子が接続されない場合を考える。各ネットと端子に信号値を表す変数を割り当てる。メモリとスイッチの間の配線にも信号値を表す変数を割り当てる。メモリには、仕様で対応するメモリと同じ変数を割り当てる。

まず、スイッチに対して次のような制約を考える。あるスイッチについて、その制御信号を c 、両端のネットの信号値を n_1、n_2 とすると、

$$C_{D1-1} = (\overline{c} + \overline{n_1} + n_2)(\overline{c} + n_1 + \overline{n_2}) \tag{11}$$

C_{D1-1} はスイッチが ON、つまり制御信号 c が 1 で n_1、n_2 が同じ値のときと、スイッチが OFF, つまり制御信号 c が 0 のとき 1 になる。c が 1 で、n_1 と n_2 の値が異なるとき、0 になる。スイッチを k 個含む設計全体に対する制約は、

$$C_{D1} = \prod_{k} C_{D1-1} \tag{12}$$

になる。配線が接続しているということは、接続されている 2 点では信号値が同じになるということである。そのため、配線が接続していないという設計誤りを SAT で見つけるためには、接続される 2 端子の信号値が異なるという制約式を 1 にする変数の割り当てが 1 つでもあれば設計誤りということになる。仕様で割り当てた端子 $p_{S1}, p_{S2}, p_{S3}, \cdots, p_{Sk_5}$ に対応する設計側の端子の変数をそれぞれ $p_{D1}, p_{D2}, p_{D3}, \cdots, p_{Dk_5}$ とする。次のような制約が

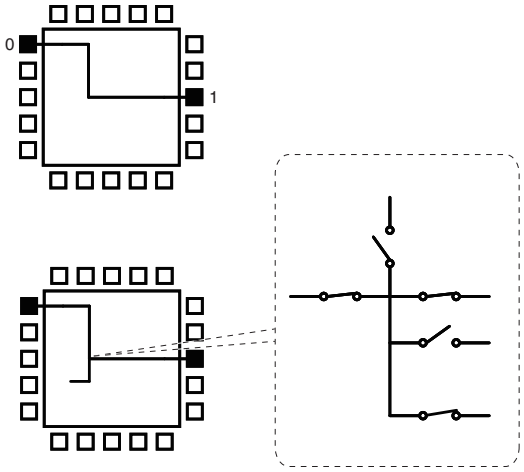


図 5　設計誤りの例

立つ。

$$C_{D2} = \prod_{kC_2} (\overline{p_{Si}} + \overline{p_{Sj}} + p_{Di} + p_{Dj})(\overline{p_{Si}} + \overline{p_{Sj}} + \overline{p_{Di}} + \overline{p_{Dj}}) \tag{13}$$

$(\overline{p_{Si}} + \overline{p_{Sj}} + p_{Di} + p_{Dj})(\overline{p_{Si}} + \overline{p_{Sj}} + \overline{p_{Di}} + \overline{p_{Dj}})$ の 2 つのクローズについて、仕様側で、p_{Si} と p_{Sj} が接続されていない場合、どちらか 1 つは 0 なので、このクローズは 1 になる。仕様側で、p_{Si} と p_{Sj} が接続されていない場合、このクローズは $(p_{Di} + p_{Dj})(\overline{p_{Di}} + \overline{p_{Dj}})$ となり、設計の端子の信号値 p_{Di} と p_{Dj} が異なる値のとき 1 になる。$C_S C_{D1} C_{D2}$ が充足可能であれば、仕様で表されるプログラムに対して、2 端子間が接続されないことがあるという設計誤りを検出できる。$C_S C_{DI}$ が充足不可能であれば、2 端子間の接続は保証され、この仕様に対して設計は正しいと言える。

次に、2 端子は接続されるが、他の端子とも接続される場合の設計誤りを考える。前述の方法で 2 端子間の接続は保証されているものとする。配線が枝分かれした場合を設計誤りとして検出することにする。メモリとスイッチの間の配線に信号値を表す変数を割り当てる。あるネットが 1 のとき、そのネットに接続されるスイッチが 3 つ以上ある場合を設計誤りとして検出する。各ネットに、変数を割り当てる。この変数が 1 になることは、そのネットに接続しているスイッチの制御信号が 3 つ以上 1 になることと同値であるようにする。あるネットに割り当てる変数を t_i 、そのネットに接続しているスイッチの制御信号の変数を $c_1, c_2, c_3, \cdots, c_k$ とすると、制御信号が 3 つ以上 1 になるならば、$t = 1$ であるという制約は、

$$C_{D3-1} = \prod_{kC_3} (t + \overline{c_i} + \overline{c_j} + \overline{c_k}) \tag{14}$$

$t_i = 1$ ならば、制御信号が 3 つ以上 1 になるという制約は、

$$C_{D3-2} = \prod (\overline{t_i} + c_1 + c_2 + \cdots) \tag{15}$$

k 個のネットに対して、

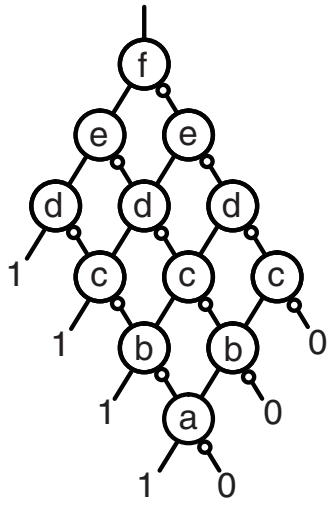


図 6　最大配線長制約に対応する二分決定グラフの例

$$C_{D3-3} = \sum_k t_i \tag{16}$$

$$C_{D3} = C_{D3-3} \prod_k (C_{D3-1} C_{D3-2}) \tag{17}$$

となる。$C_S C_{D3}$ が充足可能であれば、仕様で表されるプログラムのとき、配線が枝分かれすることがあるという設計誤りを検出できる。$C_S C_{D3}$ が充足不可能であれば、設計は正しいと判定される。

2.4　検証効率化のための制約

ここでは検証を効率化することを目的に、実際に使用される頻度の低いであろう配線パタンを検証対象から除外して探索空間を制限する 2 種類の制約式を提案する。

1 つ目の制約は、最大配線長の制約である。仕様モデル中の全ての配線を $n_1, n_2, n_3, \cdots, n_k$ とすると、最大配線長を l 以下とする制約式は、

$$n_1 + n_2 + n_3 + \cdots + n_k \leqq l \tag{18}$$

である。この式を CNF で効率的に表現するため、図 6 に示すような $(k-l+2) \times (l+1)$ の節を持つに二分決定グラフ (BDD) を構築する。この BDD では、$n_1, n_2, n_3, \cdots, n_k$ のうち、1 が l 個以下のとき 0 になり、$(l+1)$ 個以上のとき 1 になる。BDD の各節は ITE 演算に対応し、制御変数を s、then 入力を t、else 入力を f、出力を x とする。これらの関係を CNF で表現すると

$$\begin{aligned}&(\overline{s} \wedge \overline{t} \wedge x) \vee (\overline{s} \wedge t \wedge \overline{x}) \vee (s \wedge \overline{f} \wedge x) \vee \\ &\quad (s \wedge f \wedge \overline{x}) \vee (\overline{t} \wedge \overline{f} \wedge x) \vee (t \wedge f \wedge \overline{x})\end{aligned} \tag{19}$$

となる。最後の 2 つのクローズは冗長だが、SAT を解く際の効率化のために入れてある。

2 つ目は、カットの大きさの制約である。カットとは図 7 のように各行または各列に対して、垂直または水平な線上を配線が何回またいでよいかという制約である。ある垂直または水平な線上にある配線に対応する変数を

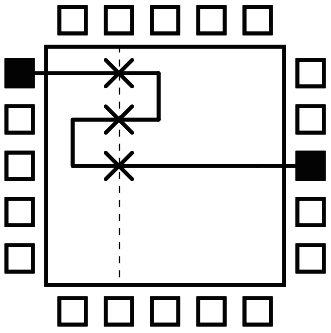


図 7　最大カット制約の例

$n_1, n_2, \ldots, n_t$ とし最大カット数を c とすると

$$n_1 + n_2 + n_3 + \cdots + n_t \leqq c \tag{20}$$

となる。この制約に関しても最大配線長制約と同様に BDD を構築してから CNF を構築する。

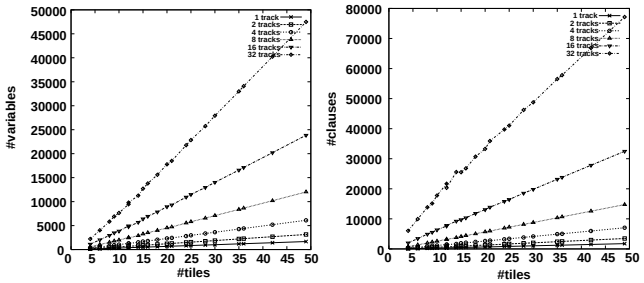
3.　計算機実験結果

提案手法で検証可能な回路の規模を評価することと、実設計へ手法を適用したときの有効性を実証することを目的として計算機を用いた実験を行った。実験環境は OS が CentOS 5, CPU は Intel Xeon 3.33GHz, メモリ容量 16GB である。検証可能規模を評価するため、指定した規模のモデルに対して提案手法による検証を行うツールを作成した。内部では SAT ソルバに MiniSat [7] を使用している。モデルは、一般的な FPGA アーキテクチャの仕様モデル・設計モデルである。また、実設計へ手法を適用するため、SPICE ネットリストから SRAM とトランジスタ部分を認識し、メモリとスイッチからなる設計モデルを構築し、それに対する制約の CNF を生成するツールを作成した。

FPGA は規則的な配列構造になっているが、その基本構成要素をタイルと呼び、論理ブロック間を通っている配線の数をトラック数と呼ぶ。検証可能規模の評価のために作成したツールは、タイル数とトラック数で検証する回路規模を指定できる。

このツールによって生成された CNF 中の変数・クローズの数とタイル数・トラック数の関係を図 8(a)(b) に示す。変数の数、クローズ数共に、タイル数とトラック数に比例して増加している。タイル数・トラック数と検証時間の関係を図 9 に示す。トラック数 8、タイル数 49 のとき検証時間は 3 時間程度となっている。

次にカット数と最大配線長に制約を加えて、検証を効率化した場合の検証時間を評価する。まず、最大配線長に制約を加えたときのタイル数と検証時間の関係を図 10 に示す。最大配線長が 5 や 10 の場合にはタイル数 1000 以上の例題を検証可能であった。しかしながら、最大配線長を 20 以上にした場合にはこの制約を入れない場合と同等の結果となった。例えば、11x10 タイルの対角線を



(a) 変数の総数　　(b) クローズ数

図 8　タイル数と制約式の大きさの関係

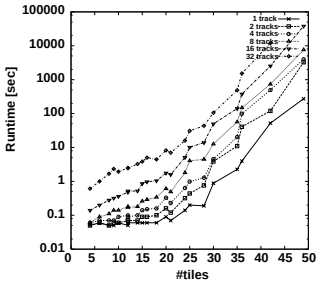
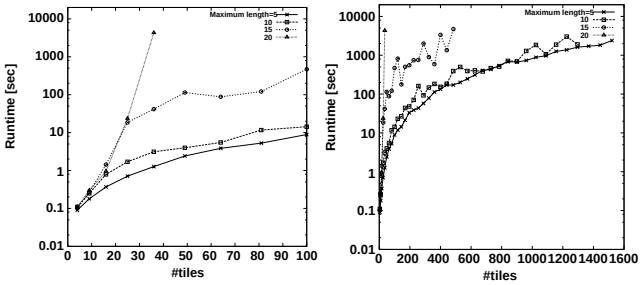


図 9　タイル数と検証時間の関係

接続するためには配線長は最低でも 21 必要であり、これは最大長制約が 20 の時には任意の 2 点間の接続を網羅的に検証できないことを意味する。よって、ある程度大きな規模の例題に対してはこの制約は実用上問題があると思われる。

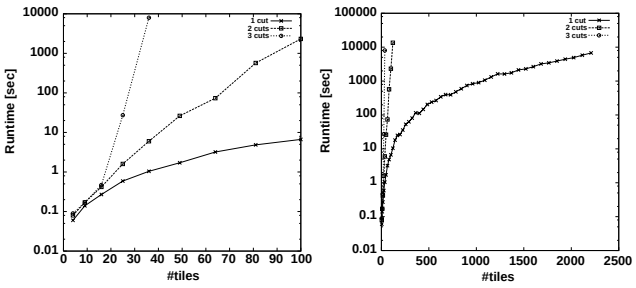
最後に図 11 に最大カット数に制約を加えたときのタイル数と検証時間を示す。最大カットが 1 の場合にはタイル数 2000 程度の場合でも一時間程度で検証可能であった。最大カットが 3 の場合には 5x5 タイル程度に対しても検証に数時間要した。しかしながら、最大カットが 1 の場合でも任意の 2 点間の接続を網羅しており、この意味で最大長制約よりも実用性が高いと思われる。

次に提案手法を実設計に適用し、その有効性を確認する。実設計の例題として、今回は VDD-FPGA [9] のプログラマブル配線部分を対象とした。この VDD-FPGA は準形式的検証と呼ばれる手続きを高速化するための専用 FPGA であり、プログラマブル配線も論理ゲートを含む複雑な構造となっている。このチップは ROHM0.18um プロセスを用いてフルカスタムで設計されており、タイル数は 12 × 13、トラック数は 8 本となっている。まず Synopsys 社の Hercules, Star-RCXT を用いて VDD-FPGA のマスクパタンからトランジスタレベルの SPICE ネットリストを抽出した。この SPICE ネットリストから SRAM とトランジスタ部分を認識して、メモリとスイッチからなる設計モデルを構築し、本提案手法を適用した。前述の考察に基づき最大カット数を 1 とする制約を併用することで、提案手法はこの設計を 19.7 秒で検証終了し、プログラマブル配線部分が正しいことを証明した。



(a) タイル数 100 以下　　(a) タイル数 1600 以下

図 10　最大配線長制約下におけるタイル数と検証時間の関係



(a) タイル数 100 以下　　(a) タイル数 2500 以下

図 11　最大カット数制約下におけるタイル数と検証時間の関係

4. ま と め

本論文では、トランジスタレベル回路でのプログラマブル配線向けの形式的検証手法を提案した。提案手法では仕様と設計を論理式で表現し、充足可能性判定問題を解くことにより検証を行う。計算機実験では、一般的な FPGA アーキテクチャを例題として検証可能規模の評価を行い、また提案手法を用いて実設計を検証することが可能であることを示した。

文　献

[1] I. Kuon, R. Tessier and J. Rose, “FPGA Architecture: Survey and Challenges,” *Foundations and Trends in Electronic Design Automation*, vol. 2, no 2, pp. 135–253.

[2] D. C. Chen, J. M. Rabaey, “A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths,” *IEEE J. Solid-State Circuits*, vol. 27, no. 12, Dec. 1992.

[3] T. Stansfield, “Using multiplexers for control and data in D-fabrix,” in *Proc. Field Programmable Logic*, pp.416–425, Sep. 2003.

[4] T. Sugawara, K. Ide, and T. Sato, “Dynamically reconfigurable processor implemented with IPFlex’s DAPDNA technology,” *IEICE Trans. Inf. & Syst.*, vol.E87-D, no.8, pp.1997–2003, Aug. 2004.

[5] A. Lester, P. Bazargan-Sabet, A. Greiner, “YAGLE, a Second Generation Functional Abstractor for CMOS VLSI Circuits,” *International Conf. on Microelectronics*, 1998.

[6] G. Dupenloup, T. Lemeunier, and R. Mayr, “Transistor Abstraction for the Functional Verification of FPGAs,” in *Proc. of Design Automation Conference*, pp. 1069–1072, Jun. 2006.

[7] N. Eén and N. Sörensson, “An extensible SAT solver,” in *Proc. SAT*, LNCS 2919, pp. 502–518, 2003.

[8] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik, “Chaff: Engineering an Efficient SAT Solver,” in *Proc. Design Automation Conference*, pp. 530–535, Jun. 2001.

[9] H. Yoshida, S. Morishita, and M. Fujita, “Hardware-Accelerated Formal Verification,” *International Workshop on Logic and Synthesis*, Jun. 2008.