Vol.2010-SLDM-147 No.10
2010/11/29

情報処理学会研究報告
IPSJ SIG Technical Report

上位設計記述の解析を利用した製造後機能テストの効率化

松 本 剛 史[†1]　藤 田 昌 宏[†1,†2]

製造後の VLSI チップの機能テストでは、故障カバレッジを向上させるために、スキャンチェインを利用して構造テスト用のテストパタンが利用されることが多い。しかし、スキャン FF の値を任意に設定できる場合、実際の回路動作では起こり得ないテストパタンが含まれ、機能的に問題のないチップを不良品としてしまうオーバーテストの問題がある。本研究では、この問題を解決するために、上位設計において成り立つ制約を抽出し、それをテストパタン生成の際の制約として与える手法を提案する。提案手法によって、本来はチップの機能に影響を与えない故障のテストを除くことができる。いくつかの制約について、テストパタン生成の実験を行い、制約を与えない場合に比べて、故障カバレッジが減少することを示す。

Improving the Efficiency of Functional Test Based on Analysis of High-Level Design Descriptions

Takeshi Matsumoto[†1] and Masahiro Fujita[†1,†2]

In production test for manifactured VLSI chips, test patterns for sturctural test using scan chains are usually applied in ordet to improve the fault covarage. However, since arbitrary values can be set to scan flip-flops in scan based test, the test patterns include ones that are not functionally infeasible in the chip. It results in yield loss, and this problem is called over testing. In this work, to solve this over testing problem, we propose a method to add constraints in test pattern generation that are extracted from high-level designs. By considering those constraints in test pattern generation, we can avoid functionally infeasible patterns, that are generated for structural test, from the test set. Applying the proposed method to several constraints, the fault coverage reductions are shown, compared to that without considering high-level constraints.

†1 東京大学大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo
†2 科学技術振興機構 戦略的創造研究推進事業 CREST
CREST, Japan Science and Technology Agency

1. はじめに

半導体集積回路 (VLSI) の製造過程の微細化・集積化に伴い、製造過程に発生する欠陥を検出するための製造テストは必要不可欠な作業となっている。VLSI の製造テストでは、与えられたテストパタンに対して、実際に製造された個々の VLSI チップが所望の動作をするかどうかを調べる。得られたテスト結果は、欠陥箇所の特定を通して製造過程の改善を行ったり、不良品の出荷を防いだりすることに利用される。

製造過程の改良を目的としたテストでは、その VLSI チップ中にある欠陥をできるだけ多く検出することが求められる。そのため、回路内の信号値に対する可制御性と可観測性を高めるため、スキャンチェインを挿入し、スキャンフリップフロップによって区切られた組合せ回路部分に対してテストを行う。このとき、外部入力から到達可能な状態 (つまり、その状態における各フリップフロップの値) をテストパタンにするのではなく、スキャンフリップフロップに外部からテストパタンの値をセットすることができるため、故障検出率の高いテストが可能である。これは、スキャンを用いたテストでは、実際にその VLSI チップが動作している際には到達しないような状態において故障の有無を調べていると言うこともできる。

一方、テストのもう 1 つの目的は、製造された VLSI チップが機能的に正しく動作するかどうかを調べることである。これは機能テストと呼ばれている。一方、前述のようなできるだけ多くの欠陥を検出するためのテストは、回路の構造を解析してテストを生成するため、構造テストと呼ばれている。機能テストでは、製造過程で発生した欠陥によって、その VLSI チップが仕様とは異なる動作をすることがないかどうかを確認する。理想的には、回路の初期状態から到達可能な状態において外部出力に影響を与えるような故障のみが検出されるようなテストパタンが用いられるべきである。しかし、実際には、ランダムに、または、人手で書かれた機能テストパタンは故障カバレッジが不十分な場合が多いため、部分的にスキャンテストが用いられている。そのため、到達可能な状態では回路の動作に影響を与えないような故障が検出される可能性があり、回路が正しく動作するにも関わらず、不良品とされてしまう場合がある。これはオーバーテスト問題と呼ばれている[1)]。

本研究では、このオーバーテストの問題を解決するための 1 つの手法として、上位設計を解析して得られた設計の動作が満たすべき条件によって、テストパタンを制約する手法を提

案し、その評価結果を示す。提案手法では、上位設計 (動作設計) から動作合成を用いる設計フローを前提として、回路の入力に関する制約、実行中の変数間で成り立つべき条件に基づく制約、上位設計中のフォールスパスに基づく制約を抽出し、既存のテストパタン生成に対する制約として利用する。本研究では、パス遅延故障を対象としている。これは、単一縮退故障のような多くのパタンによって検出が可能な故障に比べて、オーバーテストの問題がより顕著に現れるためである。提案手法の評価として、制約を追加した上で得られたテストパタンと制約を考慮せずに生成されたテストパタンの間で、パタン数と故障カバレッジを比較する。この評価を通して、制約を追加することにより、機能に影響を与えない故障を検出するパタンを除くことにより、テストパタン数を減少させることができることを示す。

本稿の構成は、以下の通りである。第 2 節では、関連研究として、擬似機能テストの基本的な考え方と上位設計記述における制約の抽出手法を概観する。続いて、第 3 節では、提案する上位設計制約を利用した擬似機能テストを紹介する。第 4 節でいくつかの種類の制約を用いてパス遅延故障のテストパタンを生成し、どの程度のパタン数が削減できるかを示す。最後に、第 5 節で本研究のまとめと今後の課題を述べる。

2. 関 連 研 究

2.1 擬似機能テスト

擬似機能テスト (Pseudo-functional test) とは、構造テスト用のテストパタンを生成する際に、回路制約を考慮することによって、より機能テストに近いテストパタンを生成しようとする手法である。文献[1] では、論理回路中の到達不可能状態とフォールスパス条件を 2 パタンテストによるパス遅延故障のテストパタン生成の制約として加える手法が提案されている。パス遅延故障のテストは、ある条件下で生成された単一縮退故障のテストパタンを 2 パターン用いて行うことができる[2]。文献[1] では、回路から抽出した前述の制約を与えてテストパタン生成を行い、故障カバレッジが減少することを示している。これは、回路の機能には影響を与えないパス遅延故障のテストを省くことができたためである。

本研究では、構造テスト用のテストパタンに回路制約を与えて、機能テスト不可能な故障に対するテストパタンを取り除く点は同じである。しかし、提案手法では、回路制約を上位設計記述、または、動作合成の合成情報から得ることにより、より効率的に制約を抽出することを目指す。本研究では、そのための準備として、いくつかの種類の制約について、どの程度、構造テスト可能で機能テスト不可能な故障に対するテストパタンを除くことができるかを評価する。

2.2 上位設計記述における制約抽出

上位設計の動作に関する制約としては、フォールスパス条件やループ不変式が挙げられる。フォールスパスおよびフォールスパス条件の検出は、大規模な設計記述では扱うパス数が指数的に増大するため、効率的に条件式を保持することが重要となる[3]。加えて、多くのフォールスパスがフォールスになる要因は局所的ないくつかの分岐条件であることが多いことが経験的に知られているため、文献[4] で提案されているような設計記述の分割による効率化も有効である。一方、ループ不変式を求める研究については、従来より盛んに研究が行われており、設計記述のシミュレーション結果からループ不変式を求める動的な手法[5] や、設計記述の静的な解析による手法が存在する。本研究では、上位設計記述において成り立つ制約については、以上のような手法を利用して得ることを前提としている。

3. 提案する上位設計における制約を利用した擬似機能テスト

3.1 基本的な考え方

提案手法の概要を図 1 に示す。本研究では、上位設計記述は C 言語または C ベース設計記述言語で記述された動作設計であると仮定する。まず、この上位設計記述を解析し、設計中で成り立っている制約を抽出する。この制約の抽出は、人手で行うことも可能であるが、第 2.2 節で述べたようなフォールスパス条件やループ不変式を抽出する手法を利用することも可能である。上位設計記述は動作合成によって RTL 回路へと合成される。この RTL 回路は、図に示すように、一般的にデータパス部分とそれを制御する有限状態機械 (FSM: Finite State Machine) で構成されている。このとき、本研究では、上位設計から得られた制約を RTL 回路上での制約に変換する必要があるため、動作合成ツールから得られるスケジューリング・バインディング情報を利用する。また、動作合成時に決定される制御 FSM の状態エンコーディング情報からも制約を抽出する。その後、得られた制約の成否を判定する回路を設計回路に付加する。この回路は、制約を与えたテストパタン生成を行う際にのみ用いるもので、最終的な VLSI チップ内には存在しない。制約判定回路が付加された回路は、通常のテストパタン生成ツールによってテストパタンの生成を行う。このとき、制約を与えることにより、オーバーテストの原因となる「構造テスト可能で機能テスト不可能」なテストパタンを除くことができる。ただし、どの程度のテストパタンを除くことができるかは、与えられた制約に依存する。

本研究で提案する上位設計制約を用いた擬似機能テストでは以下のような利点を得ることができる。

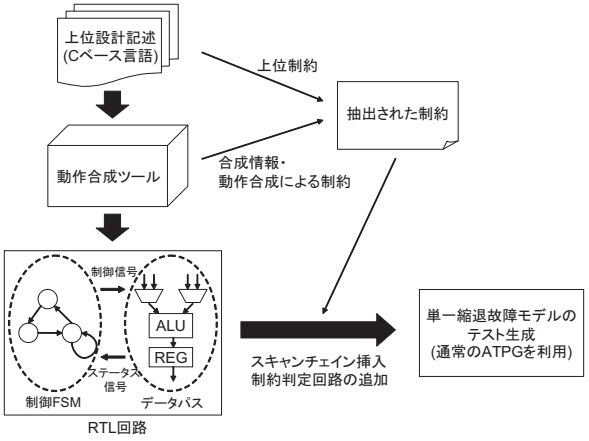


図 1 提案手法の概要

- RTL 回路・論理回路における論理的な回路制約を抽象度の高い上位設計記述において、比較的少数の変数間のより単純な論理式として表すことができる。ただし、提案手法では、上位設計記述しか参照しないため、文献[1] で含まれているような、回路中のパスにおける伝搬を考慮する必要のある制約は扱うことができない。
- RTL 回路・論理回路での制約抽出に比べて、対象としている設計に関する設計者の知識に基づく制約が与えやすい。これも、前項と同様に、上位設計の抽象度の高さによるものである。抽象度の低い設計記述では、非常に複雑になり、設計者が与えることが困難な制約であっても、上位設計では直感的に分かりやすい制約として表現することができる場合がある。
- 上位設計記述中にアサーションや入力仕様がある場合、それを制約として用いることができる。
- RTL 回路・論理回路における複数クロックサイクルにまたがる制約を与えることができる。通常、上位設計では時間の概念がない (または、大まかな経過時間の記述しかない) ため、実際の回路では複数クロックサイクルにまたがる制約 (sequential constraint) を時相論理ではなく、通常の論理式で与えることができる。

以上より、本研究で利用する上位制約は RTL 回路・論理回路での制約に比べて、比較的容易に抽出することが可能であると考えられる。また、設計者にとっても、論理回路上では複数クロックサイクルの複数フリップフロップに対する制約を上位設計で直感的に表すことができると言える。本研究では、上位設計における制約を付加することによって、どの程度のテストパタンを構造テストから除くことができるかを評価し、それらの有効性について論じる。

```
 1. input IN;
 2. output A, B;
 3. variable X, Y;
 4. X = Get(IN);  //get X from input
 5. Y = Get(IN);  //get Y from input
 6. A = 0;
 7. B = X;
 8. while(B >= Y) {
 9.   B = B – Y;
10.   A = A + 1;
11. }
```

図 2 例題 (上位設計記述)

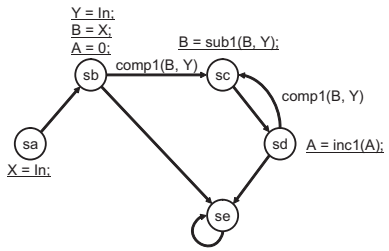


図 3 例題 (RTL 設計の制御 FSM)

3.2 上位設計における制約の抽出

本節では、上位設計記述から得ることができる制約の例のいくつかを述べる。

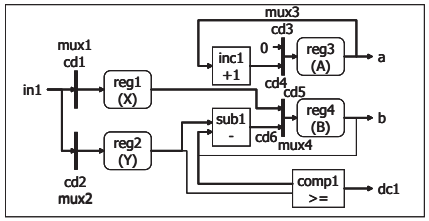
説明のための例題として、図 2 に示す減算によって整数の除算を行い、商と余りを出力する回路を考える。この動作記述を動作合成によって合成して得られる RTL 回路の例を図 4 に示す。この回路では、制御 FSM から $cd1, ..., cd6$ の 6 ビットの制御信号がデータパスを制御しており、比較器における大小比較の結果が $dc1$ として制御 FSM に入力されている。この回路の制御 FSM の状態遷移図と各状態における制御信号の値を図 3 に示す。以降では、この例を用いて、上位設計から抽出可能な制約を述べる。

3.3 あるクロックサイクルにおける制約

あるクロックサイクルにおける制約とは、ある演算が実行される時点で満たされるべき変数間の関係を表した論理式である。このとき、論理式に含まれる全ての変数は、動作合成においてその演算が行われる時点でライフタイム内である必要がある。これは、上位設計記述中では参照可能であっても、ライフタイム以降 (その変数の値が最後に使われた時点以降) は、回路中でその変数値が保持されているとは限らないためである。

満たされるべき変数間の関係としては、以下の 3 通りが考えられる。

- 該当する演算が含まれている代入文が実行されるための条件 (分岐条件)
- ループ不変式



コントローラの状態	cd1	cd2	cd3	cd4	cd5	cd6	次状態
sa(初期状態)	1	0	0	0	0	0	sb
sb	0	1	1	0	1	0	dc1 ? sb : se
sc	0	0	0	0	0	1	sd
sd	0	0	0	1	0	0	dc1 ? sc : se
se(終了状態)	0	0	0	0	0	0	se

図 4 例題 (RTL 回路)

- 設計記述中のアサーションから得られる制約

第 1 項の例として、図 2 の while ループ内で成り立つべき関係が挙げられる。ループ内を実行するためには、条件 $B \geq Y$ が真でなければならない。そこで、代入文 $B = B - Y$ が実行される時点では $B \geq Y$ が成り立っていることになる。合成情報から、$B = B - Y$ が実行されるのは、図 3 の状態 sc であり、そのとき、変数 B, Y はそれぞれ $reg4, reg2$ に割り当てられていることが分かるため、以下の条件式が得られる。

$$(state = sc) \rightarrow (reg4 \geq reg2)$$

なお、$B = B - Y$ において変数 B の値が更新されるため、それ以降では条件が成り立つとは限らないことに注意が必要である。

また、第 2 項の例として、同様に例題の while ループ内で常に成り立つ条件 $A \geq 0$ を挙げることができる。ループ内の 2 つの代入文に対応する制御 FSM の状態は sc, sd であるため、以下の条件式を得ることができる。

$$(state = sc)||(state = sd) \rightarrow (reg3 \geq 0)$$

このように、ある演算を実行しているときに成り立つ変数間の関係から得られる制約は、制御 FSM の該当する状態と、その時点での変数値が保持されているレジスタ値を用いて表すことができる。

第 3 項のアサーションについては、アサーション中で使われている全ての変数のライフタイム内において、アサーション条件が成り立つという制約を得ることができる。この制約は、前述の 2 つと同様に、その時点における制御 FSM の状態と変数値が保持されているレジスタによって記述可能である。

3.4 複数のクロックサイクルを含む制約

順序回路におけるある時点での状態 (フリップフロップ値) は、それまでに与えられた入力シーケンスによって決定するため、複数のクロックサイクルにまたがるフリップフロップ値の制約が存在する。これらの制約は、上位設計記述においては、ある 2 つの代入文にまたがる制約として表すことができる。また、複数のクロックサイクルを含む制約は、RTL 回路中で論理式として表現する場合には、時相論理式を用いて表す必要がある。

本研究では、上位設計記述中のフォールスパス条件から、複数のクロックサイクルを含む制約を得る。上位設計におけるフォールスパスとは、パス条件が偽になる実行パスのことであり、どのような入力値が与えられても実行されないパスである。例として、図 5 に示すような設計記述を考える。この例では、2 つの分岐条件において、$A > 0$ と $B = 0$ が共に真となることはあり得ない。ここで、条件 $A > 0$ の評価が行われる時刻を $T1$、条件 $B = 0$ の評価が行われる時刻を $T2$ とすると、以下の条件式が成り立つ。

$$\neg((A > 0) \wedge (state = s_A) \wedge T(T2 - T1, (B = 0) \wedge (state = s_B)))$$

ただし、ここで $T(t, C)$ は時刻 t において論理式 C が成り立つことを表す時相演算子であり、s_A, s_B はそれぞれの分岐条件が評価される時刻での制御 FSM の状態を表す。この条件式は、$A > 0$ を評価した結果と、その $(T2 - T1)$ サイクル後に $B = 0$ を評価した結果が共に真になることがないことを表している。このように、上位設計記述中で得られたフォールスパス条件は、動作合成のスケジューリング情報から分岐条件が評価される時刻を決定し、時相論理式を用いて表現することができる。

3.5 動作合成における制御 FSM の状態エンコーディングに関する制約

設計によっては、動作合成後に得られる RTL 回路の段階で制御 FSM の状態エンコーディングが決定される場合がある。例えば、明示的に、ワンホット (または、ワンコールド) に状態をエンコーディングしたり、制御 FSM で生じる全ての状態遷移についてその状態を表す値の間の距離を一定 (以上) にしたりすることが行われる。そのような場合、これらは上位設計記述から得られる制約ではないが、動作合成ツールから取得可能な情報であるため、本研究で擬似機能テストに与える制約として扱うことができる。

3.6 テスト生成における制約の付加

これまで例を用いて説明したように、上位設計で得られた制約は、動作合成ツールから得られる合成情報に基づいて、RTL 回路中でのレジスタ値や制御 FSM の状態 (制御 FSM の

```
if(A>0) {
  B = 1;
 } else {
  B = 0;
 }
…
//no assignment to B
…
if(B == 0) {
 …
}
```

図 5 フォールスパスの例

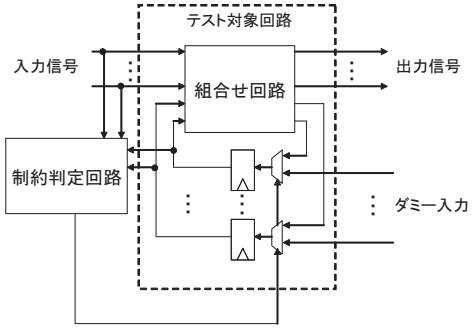


図 6 テスト対象回路への制約の付加

状態を表すフリップフロップの値) で表された制約に変換する必要がある。上位設計記述から得られる制約は、設計記述中のある時点での変数値を用いて記述される。そのため、上位設計記述中のある変数値が、どの時刻でどのレジスタに保持されているかが分かれば良い。このとき、レジスタで値が保持されている時刻は、制御 FSM の状態を用いて表すことができる。これは、どのレジスタにどの計算結果が入るかは制御 FSM によって制御されているためである。

例えば、図 2, 4, 3 の例では、上位設計記述の 7 行目の代入文 $B = X$ の変数 B に値が代入されるのは制御 FSM の状態が sb のときである。また、代入された値は $reg4$ に保持される。これらの情報は、動作合成の過程で決定されるため、原理的には合成ツールから得ることができる。

上位設計から得られた制約を制御 FSM の状態とデータパスのレジスタ値で表した後に、与えられたテストパタン (入力信号とフリップフロップへの値の代入) がそれらの制約を満たしているかどうかを判定する回路をテスト対象回路に付加する。制約がある 1 時刻の状態・レジスタ値で表現されている場合には、制約判定回路は組合せ回路として実現できる。一方、制約が複数の時刻の状態・レジスタ値で表されている場合 (時相論理式で表されている場合) には、その制約を順序回路に変換する必要がある。LTL(Linear Time Logic) 式から FSM への変換については、文献[6),7)] で提案されている手法を利用することができる。

図 6 は、回路として実装された制約がどのようにテスト対象回路に付加されるのかを示したものである。制約判定回路は、与えれたテストパタン (順序回路テストの場合は、テストシーケンス) が制約を満たす場合には 1 を出力し、満たさない場合には 0 を出力する。この制約判定回路からの出力は、各フリップフロップの直前に挿入されたマルチプレクサの制御信号へ接続されている。制約が満たされている場合、回路中のフリップフロップには、本来の回路で計算された値が接続される。一方、制約が満たされない場合には、テスト対象回路とは別に用意された入力信号が各フリップフロップに接続される。そのため、テストパタンを作るべき対象の回路中の故障がフリップフロップまで伝搬しないため、制約が満たされない場合は、それらの故障はテスト不可能になる。このようにして、上位設計記述で得られた制約を満たさないテストパタンを除くことができる。

4. 予備実験結果

提案手法による上位設計記述から得られた制約によって、どの程度のテストパタン数削減が可能か、また、どの程度の故障カバレッジの減少が生じるか、を評価するために、ある設計例題に対して、いくつかの制約を与えてパス遅延故障のテストパタン生成を行った。

例題として、加減算と条件分岐からなる 3 入力 1 出力の上位設計を用いた。この例題の C 言語記述は 15 行であり、3 つの入力変数は正整数であることが仕様として与えられている。設計中の各整数変数は 4 ビットとして動作合成を行い、6 状態の制御 FSM と、減算器 1 つ、加算器 1 つ、比較器 1 つから成るデータパスを得た。その後、論理合成を行い、論理合成・スキャンチェイン挿入を行い、パス遅延故障のテスト生成を行った。制約判定回路は、あくまでもテストパタン生成に制約を与えるための部分であるため、テスト対象からは除いている。論理合成は Synopsys 社 Design Compiler、スキャンチェイン挿入は Mentor 社 DFTAdvisor、テスト生成は Mentor 社 FastScan をそれぞれ用いた。論理合成後のテスト対象回路は 2 入力 NAND ゲート換算で約 320 ゲートであった。

評価を行った制約は以下の 3 つである。

- **制約 1** 条件分岐内の代入文を実行する間、評価された 2 つの変数の大小関係が保持される。これは、条件評価された変数への代入が条件分岐内で起こらないため、その分岐内の代入文が実行されるときに、条件 (2 変数の大小関係) が変化しないことを制約として用いている。
- **制約 2** 出力値が得られるまで入力は同じ値が保持されるという入力仕様があると仮定し、あるレジスタが入力値を保持している間は、そのレジスタ値がそのときの入力値と等価になることを制約として用いる。
- **制約 3** 動作合成後の RTL 回路において、制御 FSM の状態エンコーディングがワンホットであるため、これを制約とする。つまり、常に、状態を表すフリップフロップ値

表 1 実験結果

	テストパタン数	故障カバレッジ	テスト生成時間
制約なし	84	67.2%	0.9 秒
制約 1	73	59.5%	1.0 秒
制約 2	46	40.3%	1.2 秒
制約 3	56	57.8%	1.3 秒

のうち 1 つだけが 1 となっている。

実験結果を表 1 に示す。制約を加えた全ての場合において、制約を与えない場合に比べて、生成されるテストパタン数が減少し、故障カバレッジが減少する結果が得られた。一方、テストパタン数、故障カバレッジの減少の割合は与える制約によって大きく異なることが分かる。

5. まとめと今後の課題

本稿では、機能テストにおけるオーバーテストの問題を解決するための手法として、上位設計記述から得られた制約をテストパタン生成の際に与えることにより、機能テスト不可能な故障に対するテストを除く手法を述べた。提案手法によって、本来不良品とする必要のない VLSI チップが不良品となる可能性を減らすことが期待できる。また、従来の論理回路における制約抽出に比べて、上位設計の抽象度の高さを活かし、より直感的な論理式として制約を与えることが可能である。いくつかの制約に対する評価を通して、制約を与えることによって、テストパタン数、故障カバレッジが減少することを示した。ただし、減少の割合は与える制約に依存しており、どのような制約を与えるかが大きな問題となる。

本研究では、上位設計から得られた制約によって、テストパタン数と故障カバレッジの減少を実現することができたが、その減少幅には制約によって大きな違いがあることが分かった。今後の課題としては、どのような制約がテスト不可能な故障に対するテストをより多く取り除くことができるのかを、より多くの設計、制約について評価を行っていく予定である。この評価を通して、上位設計におけるどのような制約が、機能テストのテストパタンを生成する際の制約として有効であるかを明らかにしていくことを目指している。

参 考 文 献

1) Y.C. Lin, F. Lu, K.T. Cheng, “Pseudofunctional Testing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.25, No.8, pp.1535–1546, Aug. 2006.
2) S. Ohtake, K. Ohtani, and H. Fujiwara, “A Method of Test Generation for Path Delay Faults Using Stack-at Fault Test Generation Algorithms,” *Proc. of Design Automation and Test in Europe*, pp.310–315, Mar. 2003.
3) A. Koelbl and C. Pixley, “Constructing Efficient Formal Models from High-Level Descriptions Using Symbolic Simulation,” *International Journal of Parallel Programming*, Vol.33, No.6, pp.645–666, Dec. 2005.
4) T. Matsumoto, T. Nishihara, and M. Fujita, “Performance Estimation with Automatic False-Path Detection for System-Level Designs,” *IPSJ Trans. on System LSI Design Methodology*, Vol.3, pp.69–80, Feb. 2010.
5) M.D. Ernst, J.H. Perkins, P.J. Guo, S. McComant, M.S. Tschantz, and C. Xiao, “The Daikon system for dynamic detenction of likely invariants,” *Science of Computer Programming*, Vol.69, No.1–3, pp.35–45, Dec. 2007.
6) M. Danial, F. Giunchiglia, and M.Y. Vardi, “Improved Automata Generation for Linear Time Temporal Logic,” *Proc. of 11th Conference on Computer Aided Verification*, pp.249–260, 1999.
7) F. Somenzi and R. Bloem, “Efficient Buchi Automata from LTL Formulae,” *Proc. of 12th Conference on Computer Aided Verification*, pp.248–263, 2000.