Vol.2017-SLDM-180 No.2
2017/5/10

トレースバッファを使用した 電気的バグの発生箇所絞り込み手法

岩田 健太郎[1,a)]　ガラバギ アミル マスード[1,b)]　藤田 昌宏[2,c)]

概要：オンチップメモリの一種のトレースバッファで一定期間，値を記録するフリップフロップを適切に選択することで，電気的な要因で生じるエラーが起きたと推定される箇所を絞り込む手法について，従来手法と新手法の比較結果を示す．評価は，トレースされた値を回路に当てはめた後，サイクルとフリップフロップの両面から網羅的に発生したエラーの候補となるかどうかを順次調べることで行った．

キーワード：ポストシリコンデバッグ，電気的バグ，トレースバッファ

A Method on Narrowing Down Candidates of Electrical Bugs Using Trace Buffers

Iwata Kentaro[1,a)]　Gharehbaghi Amir Masoud[1,b)]　Fujita Masahiro[2,c)]

1. はじめに

大規模集積回路はトランジスタの微細化に従って大規模化が進行し，その設計は複雑さを増している．設計段階（プリシリコン）においてもその影響を受けており，主にシミュレーションによる検証を経て論理バグを修正していくが，複雑な設計を求められつつ市場に出すまでの時間的要求が厳しくなっている中ですべてのバグを検出するのは難しく，バグが残されたまま製造されてしまうことがある．

たとえ，プリシリコンでの検証を終えてバグが無いように見えても，数多くのサイクルを回していくと，見つけられなかったバグが動作に影響を及ぼすことがある．そのため，実行速度の遅いシミュレーションでは行えないようなサイクル数での検証を，実際に製造したチップを使用してリアルタイムの速度で行うことが必要になる．これはチップを製造した後の段階で行われるデバッグであることからポストシリコンデバッグと呼ばれ，機能的なバグがこの段階まで潜在することが増加しているため重要性を増してきている．ポストシリコンデバッグは，次のような大きく 4 つの段階に分けられる [1].

(1) 問題の検出
(2) 問題を引き起こす箇所の特定
(3) 問題の根本となる原因の特定
(4) 原因の修正

ポストシリコンデバッグでは，シミュレーションと異なり，そのままでは内部信号を観測することが困難である．そこで，内部信号を観測できるような仕組みが用いられているが，観測する信号が増えるほどオーバーヘッドも大きくなるので多くの内部信号を観測することは不可能であり，得られる内部信号の数が限られている．そのため，回路全体に対し少ない信号の情報を元にデバッグを始めることになり，デバッグは一般的に非常に困難である．その数少ない内部信号の値を観測するための方法には，前もってプリシリコンでデバッグに向けた設計（Desgin for Debug）を実装するというものがある．例えば，テストの容易化設

[1] 東京大学大学院工学系研究科電気系工学専攻
Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo
[2] 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
[a)] iwata@cad.t.u-tokyo.ac.jp
[b)] amir@cad.t.u-tokyo.ac.jp
[c)] fujita@ee.t.u-tokyo.ac.jp

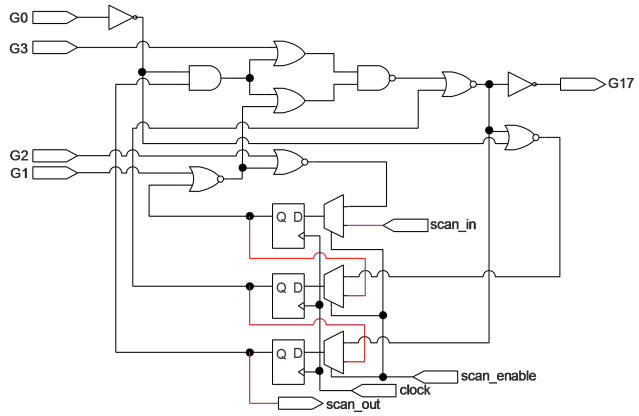

図 1　ISCAS'89 ベンチマーク [3] の s27 にスキャンチェーンを実装したもの.
赤い経路が，*scan_enable* が 1 のときのスキャンパスである.

計（Design for Test）のために長年使われているスキャンチェーンのように，フリップフロップに初期状態を与えたり，サイクルごとにそれらの値を取得したりできるスキャン技術や，トレースバッファのようにチップを動作させたまま内部信号をトレースする技術がよく用いられる [2].

スキャンチェーンは，**図 1** のようにフリップフロップの入力に 2 入力のマルチプレクサを接続し，一方の入力に元々の回路，もう一方の入力に異なるフリップフロップを繋げることで全てのフリップフロップを鎖状に繋げて一つのシフトレジスタを実現する．そして，マルチプレクサで通常動作モード（*scan_enable* が無効）とスキャンモード（*scan_enable* が有効）とを使い分けることで，フリップフロップの接続を制御し，デバッグのために値を取得することが出来る.

トレースバッファについては，第 1.2 節で述べる.

本研究では，製品化前のチップの，月単位の時間を要することもあるポストシリコンでの検証・デバッグにかかる時間を短縮することを，最終的な目標として進めている．現在は，先述したポストシリコンデバッグの 4 段階中の 2 に着目しており，今回の発表では 2 に関して既存手法の有効性を確認する実験を行った.

1.1　電気的バグ

電気的バグとは，一時的な電源電圧の降下・チップ内部の温度勾配・配線間のノイズなど，チップの電気的な動作環境が原因となるバグである [4]．このバグによって一時的にデータに誤りを生じさせるソフトエラーが引き起こされ [5]，その値が伝搬していくことで外部から観測可能な動作不良が引き起こされる.

論理バグは，動作設計・機能設計・論理設計といった設計段階でのミスが原因となって生じる機能的なバグで，その部分で処理される信号に対し常に同様の不具合を引き起こす．一方，電気的バグは論理バグとは違って先ほど述べた種々の電気的な環境が原因となるので，同じ環境を再現することが難しく，また，同じ環境を再現できたとしても不具合が引き起こされる場合とそうでない場合があるという点が特徴である．そのため，電気的バグは論理バグに比べて発生頻度と再現性が非常に低く，発生箇所の特定に非常に時間がかかる [1]．ポストシリコンデバッグには月単位での時間が必要なこともあり，製品化に向けて多くのリソースが電気的バグのデバッグに割かれることになる.

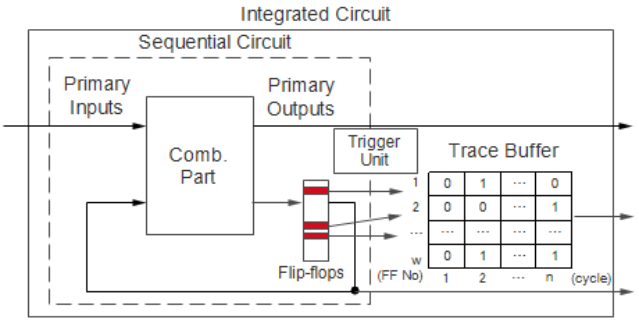

図 2　トレースバッファが実装された IC の概念図.
赤い部分で示したうに，一部のフリップフロップだけがトレースできる.

このような状況から，電気的バグのデバッグに要する期間を短縮することが重要で，効果的・効率的なデバッグ手法が必要とされている.

1.2　トレースバッファ

トレースバッファは，DfD の一つでオンチップメモリの一種である（**図 2**）．トレースバッファの役割は，順序回路の一部のフリップフロップの値を，一定期間記録することである．規模としては，1000 から 8000 サイクルの期間，8 から 32 個分のフリップフロップの値を記録できるものがよく用いられている [6]．一般的に，トレースバッファでの記録の開始および停止を制御するために，トリガーが設けられる．トレースバッファ自体は値を記録する機能を持つのみで，記録された値が正しいかどうかの判断は別に行う必要がある.

トレースされた値を当該フリップフロップの前後の信号に伝搬していくことで，トレースされていないフリップフロップの値を修復することができる．これらの修復は，論理ゲートの全ての入力値が既知である場合や，入力値の一部だけが既知である場合でもその値が制御値であったり，特定の出力値が既知であったりする場合にも適用できる（**図 3**）.

2.　既存のトレース信号選択手法

2.1　SRR 基準のトレース信号選択手法

第 1 節で述べた通り，DfD や DfT という技術を用いても，ポストシリコンデバッグでは観測できる内部信号の

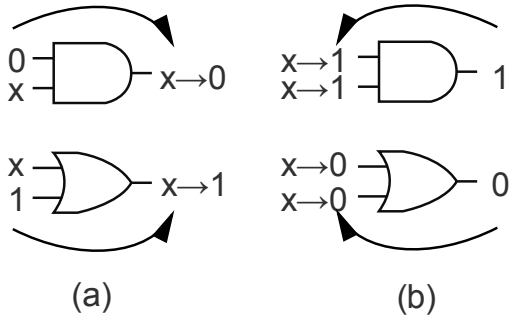

図 3　トレースした値の伝搬の例.
（a）順方向の場合.　（b）逆方向の場合.

数が限られる．その限られた数を有効に利用するため，トレースしたフリップフロップの値を元にしてトレースしていないフリップフロップの値をどれだけ修復できるかを示す割合（Signal Restoration Ratio）を基準として，トレースするフリップフロップを選択する手法がこれまでに考案されている．

SRR は，次の式で算出される．

$$SRR = \frac{\text{トレースされた信号数+修復された信号数}}{\text{トレースされた信号数}}$$

これまでに数多くの SRR を指標としたトレース信号の選択手法が提案されてきた．[7] では，シミュレーションに加えて，それまでに選んだフリップフロップをトレースしたときにそれぞれの信号が修復できる確率や，あるフリップフロップをトレースすると仮定した時にどれぐらいの他のフリップフロップにを修復できるかなどの 4 つの判断基準を導入することで，速い実行速度と高い正確性を両立したトレース信号選択アルゴリズムが提案されている．

それぞれの論理ゲートの入出力の値の変化によって値が変化する確率を導き，それを回路全体に波及させて考えることで効率的にトレース信号を選択する手法も提案されている [8]．

Rahmani ら [9] の手法では，シミュレーションに基づいてトレースすべき信号を選ぶ．空集合 $S = \phi$ を準備し，S の要素数がトレースすべきフリップフロップの数 w に達するまで，S に含まれないフリップフロップの中からその時点での SRR が最大になるようなフリップフロップを 1 つずつ選んでいくアルゴリズムを考案した．このアルゴリズムの中ではランダムな入力ベクトルを使用するが，ランダム性の影響を軽減するために，繰り返しアルゴリズムを適用し出来た複数のトレース信号の集合の和集合 A から w だけ選択する，ということを整数線形計画法を利用して行った．

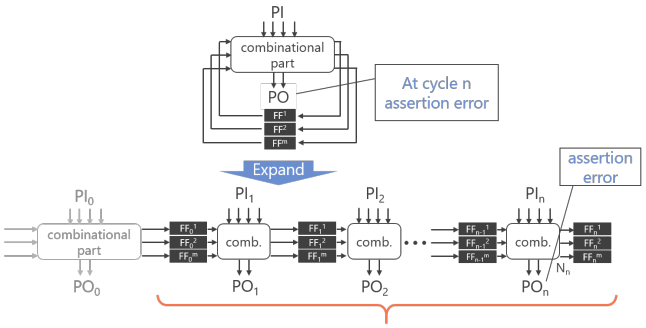

図 4　n サイクル分の時間展開.

2.2　フリップフロップ間の繋がりを基準にしたトレース信号選択

Kumar ら [10] は，得られる信号を最大化する方法ではエラー発生箇所の特定は難しいとして，回路のトポロジーに着目して，フリップフロップ同士の接続の組み合わせの多さをスコア化し，そのスコアを基準にトレース信号を選択する手法を提案した．

3.　電気的バグのデバッグへのトレースバッファの利用

第 2 節で述べたような，SRR を最大化する手法と，フリップフロップ間の接続の組み合わせが多いものを選ぶ手法では，どちらが電気的バグの発生した位置を特定する（絞り込む）のに有効なのかを確かめたい．

そこで，電気的バグをシミュレーションして引き起こされる不具合について，2 つの基準で選ばれたトレース信号が電気的バグにおいてそれぞれどのように機能するかについて実験する．対象とする回路は ISCAS'89 ベンチマーク [3] の順序回路で，n サイクル分だけを考える．フリップフロップを n サイクル分だけ時間展開すると**図 4** の下段のように組み合わせ回路として考えられる．回路設計は正しい（論理バグは無い）ものとし，電気的バグが n サイクル内に一度だけ生じると仮定する．つまり，時間展開した回路全体で一度だけ値の反転が発生するという仮定である．

一般に，電気的バグは回路内部の任意の信号でエラーを引き起こす可能性があるが，ここでは簡単のため，発生した際には一つのフリップフロップにのみエラーが伝搬する電気的バグを考える．そこからそのエラーがサイクルごとに他のフリップフロップに伝搬していき，外部出力にまで伝搬し不具合が表面化するとアサーションエラーが出て回路の動作が即座に止められるものとする．初めての外部出力に不具合が生じたときを n サイクル目とする．

Procedure1 に実験の手順を示す．はじめに，準備段階として，電気的バグが起こった状態を作る．デバッグ対象のチップ ckt_r を複製したもの ckt_g の，それぞれの外部入力 $PI_1, \ldots, PI_n$ と初期状態信号 FF_0 を共通化する．次に，電気的バグが 1 サイクル動作した後に発生し，n サイクル目

の動作で外部出力に初めて不具合が表面化するという状況を人工的にシミュレーションする．そこで，1 サイクル動作したときの状態信号 FF_1 において，いずれか 1 つのフリップフロップのみに電気的バグが伝搬してきたとして，そのフリップフロップの値を反転させる．そして，$n-1$ サイクルまでの外部出力は ckt_r と ckt_g で同じ値となり，かつ，最後の n サイクル目においてのみ 2 つの回路の外部出力の値が少なくとも 1 組異なるという条件（以下，条件 A とする）を充足可能性判定問題（SAT 問題）に変換して解き，条件を満たす（SAT となる）ものを 1 つ選び，電気的バグが起こった状況を作り出す．これを，FF_1 のフリップフロップそれぞれについて行う．つまり，フリップフロップの数を m とすると，i サイクル目の状態信号は $FF_i^1, \ldots, FF_i^m$（$i = 1, \ldots, n$）と表せるので，FF_1^1 から FF_1^m までのフリップフロップの値をそれぞれ一度ずつ反転させ，m 種類の電気的バグを作成しようとしている．なお，あるフリップフロップについて，いかなる外部入力と初期状態を考えても条件 A を満たす電気的バグを作れない（UNSAT となる）場合には，そのフリップフロップについての電気的バグは考慮しない．ここまでが準備段階である．

このようにして人工的に作成した電気的バグによって不具合が発生した状況を元に，展開された回路内の 1 つの状態信号値を反転したとき，電気的バグと等価な振る舞いをするものがどれだけあるかをシミュレーションで調べる．この段階で使える値は，準備段階で電気的バグを作成したときの信号値のうち，1 から n サイクルまでの全ての外部入力・外部出力，そして，トレースしたフリップフロップの値および最終 n サイクルにおける全てのフリップフロップ $FF_n^1, \ldots, FF_n^m$ の値である．$FF_n^1, \ldots, FF_n^m$ の値が使えるのは，n サイクル目でチップの動作が止まるためスキャンチェーンを利用して値を読み出せるからである．それらの信号には，電気的バグが生じた ckt_r のものを割り当てる．

このような後半の処理を FF_1^1 から FF_1^m で発生したそれぞれの電気的バグについて行い，各サイクルにおいてそれぞれの状態信号の値を 1 つずつ反転していき，電気的バグと等価な値の反転を見つける（**図 7**）．

トレース信号を選択する手法として，第 2 節で述べた Rahmani ら [9] の SRR 基準の選択手法と，Kumar ら [10] の手法を比較した．

さらに，$n = 3$ サイクルの時に電気的バグを人工的にシミュレーションできるフリップフロップのみを対象として，Kumar らの手法でトレース信号を選択したものも比較に加えた．

実験した回路では，トレースするフリップフロップの数 w は 4 とした．

Procedure 1 Experiment(ckt_r, n, res_data)

Input: ckt_r, n ▷ let ckt_r be a real chip and n be the number of its cycles
Output: res_data
  $ckt_g = ckt_r$ ▷ Let ckt_g be golden use common pis and initial state signals between ckt_r and ckt_g
  for each $FF_1^j \in FF_1$ **do** ▷ An electrical error injected at cycle 1 and simulated artificially
    $ckt_r{}' =$ flip FF_1^j value of ckt_r
    $res, trace\_values =$ CreateBuggySituation($ckt_r{}', ckt_g$)
    if $res ==$ UNSAT **then**
      continue
    end if
    $ckt_s = ckt_r$
    assign $trace\_values$ to ckt_s ▷ All primary inputs/outputs and traced signals are assigned.
    for $i = 1$ **to** n **do**
      for each $FF_i^k \in FF_i$ **do**
        $ckt_s{}' =$ flip FF_i^k of ckt_s
        $res\_data[FF_1^j][FF_i^k] =$ CheckFeasibility($ckt_s{}'$)
      end for
    end for
  end for

  function CreateBuggySituation(ckt_1, ckt_2)
    for each combination of signal values **do**
      if $ckt_1(1) == ckt_2(1)$ **and** ...**and** $ckt_1(n-1) == ckt_2(n-1)$ **and** $ckt_1(n) \neq ckt_2(n)$ **then** ▷ $ckt_x(i)$ returns values of primary outputs at cycle i
        return SAT, $signal\_values$ of ckt_1
      end if
    end for
    return UNSAT, None
  end function

  function CheckFeasibility(ckt_1)
    if there exists a pattern of signals which statisfies conditions of ckt_1 **then**
      return SAT
    end if
    return UNSAT
  end function

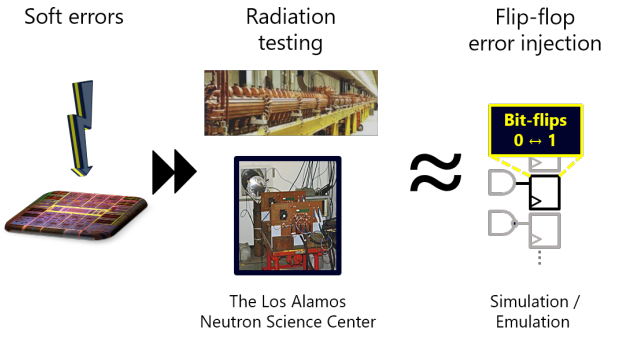

図 5　ソフトエラーは，フリップフロップの値が反転することに近似できる．[5].

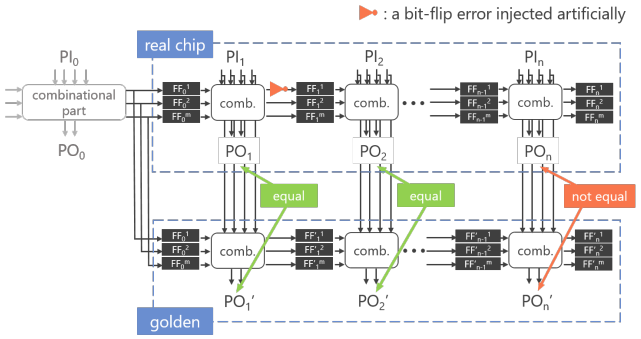

図 6　電気的バグを人工的にシミュレーションする．
FF_1^1 から FF_1^m までの状態信号を 1 つずつ反転し，それぞれの場合で $n-1$ サイクル目までは正しく，n サイクル目に不具合が表面化する状況を探す．

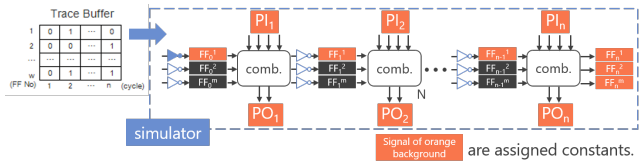

図 7　トレース情報を割り当てて電気的バグと等価故障となる値の反転を探す．
値の反転（青いインバータ）は，フリップフロップに順に挿入され，それぞれについて条件を満たすかどうかを判断する．

3.1　実験結果

実験は，Xeon E5-2690 2.90GHz CPU，128GB RAM，CentOS 6.6 の計算サーバ上で行った．SAT ソルバには Minisat 2.2.0[11] を用いた．

$n=3$ サイクルでの実験結果を表 1 に示す．左から，回路名，フリップフロップの数およびそのうち $n=3$ において電気的バグが生じた状況を作れたフリップフロップの数，値を反転したサイクルと実際に電気的バグを挿入したサイクルの差で，それぞれの結果の欄は左の列がトレースした信号数およびそのうち電気的バグが生じうるフリップフロップをトレースした数，右の列がそれぞれのサイクルでの電気的バグの発生箇所の候補（電気的バグと等価故障）の数である．

なお，実験結果はランダム性を軽減するため，電気的バグを 1 つのフリップフロップあたり 100 パターン生成し，それぞれの結果を平均したものである．

実験の結果，s444 と s510 については Kumar らの手法の方が，電気的バグの発生箇所の候補が少なく，良い結果を示した．一方，s526 ついては，Rahmani らの手法のほうが，優れた結果となった．s832 については，同じパフォーマンスを示した．

電気的バグを生じうるフリップフロップのみをトレースした場合は，いずれの回路の場合も他の手法より優れた結果または同等の結果になった．

3.2　考察と課題点

エラーの検出に指向した Kumar らの手法でも，回路によっては SRR を指標とした手法に劣ることがある．これは回路の性質によっては，単に多くのフリップフロップの値を復元した方が結果が良い場合もあるということを意味する．

電気的バグが起こりうるフリップフロップのみをトレースした場合が，他の手法より劣ることがなく，元の Kumar らの手法より数段良い結果になったことは，電気的バグが起こりえないところをトレースしたとしても，電気的バグの情報はあまり得られず絞り込みに寄与しないということを示唆する．

フリップフロップによっては，あるサイクル数で今回の条件を満たす電気的バグがにくいものもあるので，その特性を利用できればより効果的なトレース信号を選択できると考える．

課題点には以下の様なものがあげられる．

- 現在の手法では，候補が数十箇所にまでしか絞られないので，さらなる効果的な手法が必要である．
- 回路の解析により，エラー伝わりやすいフリップフロップがわかれば，有効な手法となると考えられる．
- 実験の対象とした回路が少ないので，より一般性を高めるために多種多様な回路に対し実験を行う必要がある．
- 実験は $n=3$ サイクルで行ったが，実際のポストシリコンデバッグでは数千から数万サイクルで起こるバグもあるので，サイクル数をさらに増やす必要がある．
- 今回は 100 個の電気的バグのパターンを用い，それらの結果の平均を元に評価を行ったが，パターン総数は外部入力数とフリップフロップ数に対して指数関数的に増加し莫大な数となるため，さらにカバレッジを上げる必要がある．
- 網羅的に候補を挿入して調べるという操作を行うと SAT 問題を解く時間が長くなるため，より効率的な評価アルゴリズムを検討する必要がある．これについては，並列実行が 1 つの有用な手法であると考えられる

表 1 比較結果

			Rahmani[9]		Kumar[10]		EBactive based on Kumar	
	FF	cycle diff	traced		traced		traced	
circuit	{EBactive}	(cand-real)	{EBactive}	cand	{EBactive}	cand	{EBactive}	cand
s444	21{7}	-1	4{1}	62.6	4{0}	59.7	4{4}	41.8
		0		44.9		47.5		31.0
		1		41.3		37.2		24.9
s510	6{5}	-1	4{3}	12.1	4{4}	11.9	4{4}	11.9
		0		7.0		5.0		5.0
		1		1.0		0.0		0.0
s526	21{7}	-1	4{1}	54.9	4{0}	84.4	4{4}	11.0
		0		38.1		63.4		13.6
		1		24.2		51.5		5.3
s832	5{5}	-1	4{4}	0.0	4{4}	0.0	4{4}	0.0
		0		5.0		5.0		5.0
		1		0.0		1.0		1.0

が、根本的に解決すべきである．

4. まとめ

本報告では，既存のトレース信号選択手法を電気的バグの発生箇所候補を絞り込むという観点で比較し評価した．エラーの検出に指向した手法でも，他の方法が効果的であることがあり，電気的バグの候補絞り込みという点では発展の余地があると言える．

サイクル数 n によって，条件を満たす電気的バグが生じうるフリップフロップとそうでないフリップフロップがあり，前者のみをトレース出来た場合，効果的な結果が得られた．そのため，各フリップフロップのサイクル数に対する電気的バグの生じる可能性を探り，それを元にトレース信号を選ぶという方向が今後の方針として考えられる．

参考文献

[1] Mitra, S., Seshia, S. A. and Nicolici, N.: Post-Silicon Validation Oppotunities, Challenges and Recent Advances, *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pp. 12–17 (2010).
[2] Nicolici, N. and Ko, H. F.: Design-for-debug for post-silicon validation: Can high-level descriptions help?, *2009 IEEE International High Level Design Validation and Test Workshop*, pp. 172–175 (2009).
[3] Brglez, F., Bryan, D. and Kozminski, K.: Combinational Profiles of Sequential Benchmark Circuits, *IEEE International Symposium on Circuits and Systems*, Vol. 3, No. 217, pp. 1929–1934 (1989).
[4] Gao, M., Lisherness, P. and Cheng, K.-T. T.: Post-silicon Bug Detection for Variation Induced Electrical Bugs, *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, IEEE, pp. 273–278 (2011).
[5] Jacob Abraham: Keynote Talk: Cross-Layer Resilience: Are High-Level Techniques Always Better?, IEEE International High-Level Design Validation and Test Workshop (2016).
[6] Ko, H. F. and Nicolici, N.: Automated Trace Signals Identification and State Restoration for Improving Observability in Post-Silicon Validation, *2008 Design, Automation and Test in Europe*, IEEE, pp. 1298–1303 (2008).
[7] Li, M. and Davoodi, A.: A Hybrid Approach for Fast and Accurate Trace Signal Selection for Post-Silicon Debug, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 33, No. 7, pp. 1081–1094 (2014).
[8] Liu, X. and Xu, Q.: On Signal Selection for Visibility Enhancement in Trace-Based Post-Silicon Validation, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 31, No. 8, pp. 1263–1274 (2012).
[9] Rahmani, K., Mishra, P. and Ray, S.: Efficient Trace Signal Selection using Augmentation and ILP Techniques, *International Symposium on Quality Electronic Design, ISQED*, pp. 148–155 (2014).
[10] Kumar, Binod and Jindal, Ankit and Fujita, Masahiro and Singh, Virendra: Post-silicon Observability Enhancement with Topology Based Trace Signal Selection, *IEEE Latin-American Test Symposium (LATS)* (2017).
[11] Eén, N. and Sörensson, N.: An extensible SAT-solver, *Theory and applications of satisfiability testing*, Springer, pp. 502–518 (2004).