Vol.2021-SLDM-195 No.8
2021/10/15

ファジングと高位合成を用いた近似コンピューティング回路のタイミング検証手法

熊谷 僚太[1]　増田 豊[2]　石原 亨[2]

概要：本研究では，近似コンピューティング (Approximate Computing; AC) 回路のタイミング検証手法を提案する．AC 回路のタイミング検証では，「どのようなテストパターンが，計算結果 (例. 画像処理の画質) を許容できない範囲まで低下させるか」調査する．本研究では，変異ベースファジング (Coverage-based Greybox Fuzzing; CGF)，高位合成，及び Fault Injection (FI) を利用した AC 回路のタイミング検証手法を提案する．CGF によりテストパターンを生成することで， AC 回路の計算品質を低下させるパターンを効率的に発見する狙いがある．提案手法は，既存の CGF ツールの移植性を考慮して，SystemC などの高水準言語で記述されたハードウェアを対象とし，上記のハードウェアに対して高位合成を利用してタイミング特性を抽出する．タイミングクリティカルな Flip-Flop (FF) に FI を行う実装を追加したのち，ファジングを実行することで，「遅延故障の発生条件を考慮しつつ，どのようなテストパターンが計算結果の品質を低下させるか」自動探索する．

1. 序論

集積システムの省電力化と高性能化を推進するポストムーアコンピューティング技術として，近似コンピューティング (Approximate Computing; AC) に注目が集まっている [1], [2], [3]．AC は，集積回路内で実行される計算を，重要な計算と重要ではない計算に分別し，重要な計算を正確に，重要ではない計算を近似的に実行する設計技術である．機械学習，ディジタル信号処理，エッジコンピューティングなどの多様な分野において活躍が期待されており，ソフトウェアからハードウェアまでを横断する様々な近似方法が数多く提案されている．

設計した AC 回路の信頼性を保証するためには，多様なテストパターンの元で正しく動作するか検証することが不可欠である．AC 回路の検証は，機能的検証とタイミング検証に大別される．機能的検証では，設計した回路が論理的に正しく振る舞うか確認する．一方，タイミング検証では，論理ゲートや配線の遅延を考慮したうえで，設計後回路がタイミング制約を満足して正しく動作するか評価する．本研究では，AC 回路のタイミング検証に焦点を絞る．

ここで，従来回路のタイミング検証では，全てのパスが遅延故障を起こさないことを制約とする．一方，AC 回路の検証では，計算結果の品質 (例. 画像処理の画質) を制約とし，制約を満足する範囲であれば，故障の発生を許容する．この観点から，AC 回路の検証においては，タイミング故障の影響を適切に見積もることが極めて重要である．換言すれば，AC 回路の検証では，従来回路と異なり，計算品質の制約を脅かすテストパターンを明らかにする必要がある．しかし，前述の通り，AC 回路と従来回路においては，故障の取り扱いが異なるため，従来検証技術を AC 回路に適用することは容易ではない．以上の問題から， AC 回路のタイミング検証手法は依然として確立されておらず，新たな検証技術の開拓が強く望まれている．

他方，ソフトウェアテストの研究領域では，ファジング [4], [5], [6], [7] と呼ばれる手法が盛んに研究されている．ファジングは，テストパターンの生成と実行を繰り返し行う手法であり，未知のバグに対する高い発見能力が数多く報告されている．ファジングの中でも，Coverage-based Greybox Fuzzing (CGF) と呼ばれる手法が，最も有望な手法の一つとして，注目を集めている．

このような背景から，近年，ソフトウェアのみならずハードウェアの領域において，CGF を活用した研究が開拓されつつある [8], [9]．例えば，文献 [8] では，AC 回路の品質検証に CGF を適用するために，ハードウェアレベルの論理値の組み合わせに対して，検証網羅性を評価する手法を提案し，ランダムテストと比較して 3 倍の検証高速化を実験的に確認している．また，[9] では，クリティカルパスを活性化させるテストパターンを生成するために，CGF

[1] 名古屋大学情報学部コンピュータ科学科
[2] 名古屋大学大学院情報学研究科

を活用している．しかしながら，AC 回路のタイミング検証に CGF を応用する方法は依然として未確立である．

本稿では，CGF を用いた AC 回路のタイミング検証法を提案する．提案手法の肝は，高位合成 と Fault Injection (FI) を用いた遅延故障の表現，および，CGF を用いた自動検証機構にある．高位合成では，合成時に豊富な制約を与えて，多様なアーキテクチャにおけるタイミング特性を抽出する．次に，FI を応用し，抽出したタイミングクリティカルな Flip-Flop (FF) に対して，ビット反転を行う機構を実装する．

本研究の主な貢献は，CGF を用いた AC 回路のタイミング検証法にある．AC 回路のタイミング検証に対して CGF を応用した研究は，著者らの知る限り，本研究が初である．提案手法では，AC 回路のタイミング特性を CGF に取り入れるために，高位合成と FI を活用する．その後，CGF を用いて，計算結果の品質を低下させる遅延故障，および，それらの故障を誘発するテストパターンを自動探索する狙いがある．

本稿の以降の構成は以下の通りである．2 章では，提案手法の核である CGF について説明する． 3 章では，提案手法の全体像を述べ，高位合成，FI，CGF の役割について述べる．4 章では，高位合成を用いたアーキテクチャ探索に向けた予備実験を行う．最後に，5 章で結論と今後の展望を述べる．

2. ファジングと AC 回路のタイミング検証

本章では，まず，2.1 節において，ファジングを説明する．次に，2.2 節において，AC 回路のタイミング検証について述べる．

2.1 ファジング

ファジングは，テストパターンの変形と実行 (Program Under Test; PUT) を繰り返し行う手法であり，現在では，ソフトウェアの品質テストにおける最重要技術の一つとされている．ファジングの手法は，ブラックボックス，ホワイトボックス，グレイボックスの三種類に大別される．ブラックボックス型は，ランダムテストの派生形であり，テストパターン生成の工数が小さいという利点を持つ．ホワイトボックス型は，既知のテスト対象ソースコードに対して，解析を行い効率的なテストパターンを生成する手法である．グレイボックス型は，上記の二種のハイブリッド型であり，効率的なテストパターンを素早く生成することを狙った手法である．

グレイボックス・ファジングにおいて，コードカバレッジを入力の変異に取り入れる手法を，特に Coverage-based Greybox Fuzzing (CGF) [10] と呼ぶ．図 1 を用いて，CGF の動作を説明する．CGF では，「テストパターンの変異，PUT の実行，カバレッジの集計とテストパターンへの

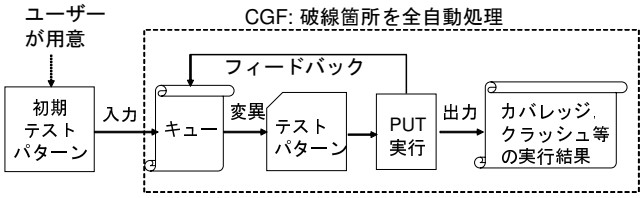

図 1: CGF (Coverage-based Greybox Fuzzing) の概要

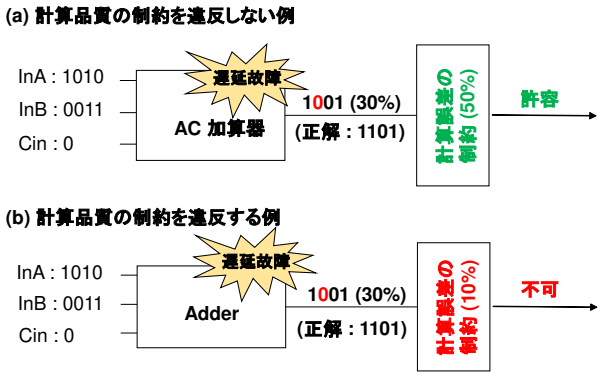

図 2: AC 回路のタイミング検証: (a) 遅延故障を許容できる例，(b) 許容できない例

フィードバック」を繰り返し実施することで，テスト対象空間を自動探索する．PUT 時に実行された命令を観測し，新たな分岐 (例. if，else if) が実行された場合には，分岐の網羅性を向上した入力パターンとしてキューに追加し，次の変異に利用する．CGF は，変異とカバレッジの集計が軽量であり，未知のバグに対する発見能力も高い [4]．本研究では，CGF の高速性と高い探索能力に着眼し，AC 回路のタイミング検証への応用法を提案する．

2.2 AC 回路におけるタイミング検証

1 章で前述した通り，AC 回路のタイミング検証では，計算品質の制約を違反しない範囲で，遅延故障の発生を許容する．**図 2** を用いて，さらに詳述する．図 2 は，4 ビットの近似加算器において，遅延故障が発生した例である．入力は，InA = 4'b1010，InB = 4'b0011，Cin = 1'b0 である．また，図 2(a) と (b) において，近似加算結果の計算誤差に対して，それぞれ 50% と 10% の制約が付与されている．近似加算器は 4'b1001 を出力しており，この計算誤差を 1 - $\frac{9}{13}$ = 30% と見なす．

図 2 より，同様の遅延故障であっても，計算誤差の品質に応じて，許容できるか否かが変動することが分かる．例えば，図 2(a) では，計算誤差に対して 50% の制約を与えているため，30% の誤差は許容できる．一方，図 2(b) では，制約は 10% であるため許容できない．なお，従来回路の検証では，全ての遅延故障を許容しないため，図 2(a) と (b) の両方が異常動作と見なされる．以上より，AC 回路では，計算品質の制約を満足する範囲であれば，遅延故障の発生が許容され，この検証方針は従来回路と全く異な

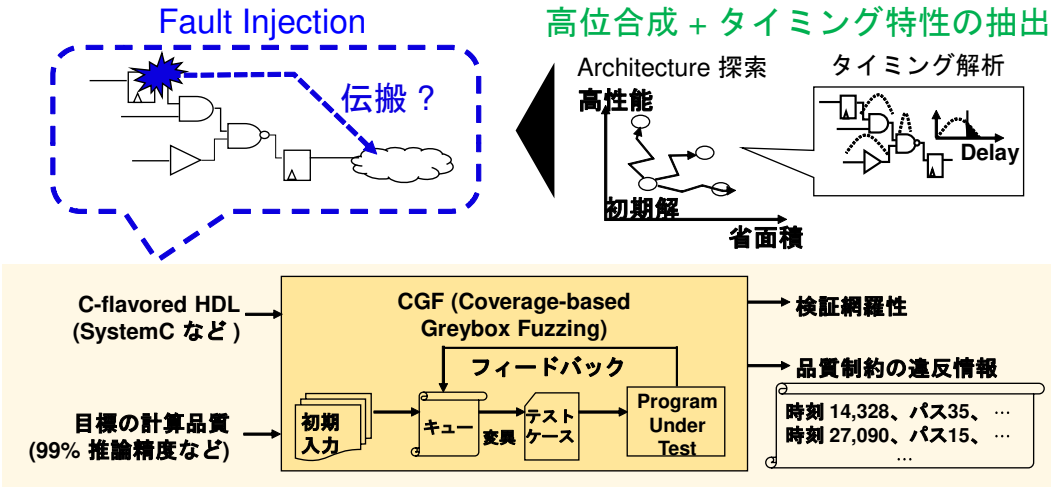

図 3: 提案手法の全体像: 高位合成，FI，および，CGF を用いた AC 回路のタイミング検証法

る．すなわち，AC 回路のタイミング検証では，品質制約を違反する遅延故障，および，その故障を引き起こすテストパターンを発見することが求められる．次章では，AC 回路のタイミング検証手法を提案する．

3. 提案手法

本章では，CGF を用いた AC 回路のタイミング検証法を提案する．**図 3** に，提案手法の全体像を示す．提案手法は，既存の CGF ツールの移植性を考慮して，SystemC などの高水準言語で記述された AC 回路を対象とする．タイミング特性を考慮したファジングテストを実行するために，高位合成と FI を利用する．3.1 節，3.2 節，3.3 節において，提案手法を構成するキーアイデアである高位合成，FI，および CGF について，それぞれ説明する．

3.1 高位合成

高位合成は，別名，動作合成とも呼び，SystemC などの高水準言語で記述されたハードウェア設計アルゴリズム (動作記述) を，RTL 記述に変換する処理のことである．高位合成は，RTL 設計作業の効率化，アーキテクチャの最適化に伴う高性能化と省電力化を提供する．一方，本研究では，タイミングクリティカルなパスを探索するために，高位合成を使用する (図 3 右上部)．具体的には，面積，スループット等に対して，多様な制約を与えて，広い範囲でアーキテクチャ探索を行う．各アーキテクチャにおいて，タイミング解析を行い，タイミングクリティカルな FF 群を推定する．

3.2 Fault Injection

Fault Injection (FI) は，回路システム上に恣意的に誤りを注入して，動作確認を行う手法である．誤り挿入後，回路が誤動作を起こすか確認することにより，故障に脆弱な回路機構を診断できる．FI は，ソフトウェアシステムの脆弱性検査，宇宙線に起因するソフトエラーの影響評価など，多様な用途に利用される．

本研究では，AC 回路内の Flip-Flop (FF) に FI を適用し，AC 回路内での遅延故障発生を模擬する (図 3 左上部)．具体的には，3.1 節において，高位合成とタイミング解析でタイミングクリティカルな FF を抽出した後，各 FF に対応する信号名を，高水準言語で記述された AC 回路から取得する．その後，対応する信号群に対して，論理値を恣意的に反転させる記述をハードウェア記述言語 (Hardware Description Language; HDL) 上に追加実装する．以上の実装により，遅延故障の発生を疑似したシミュレーション環境を提供する．

3.3 CGF を用いたタイミング検証

3.1 節と 3.2 節において，CGF 実行時に遅延故障を模擬的に発生させる手法を構築した．提案手法では，この環境下で CGF を実行する (図 3 下部)．タイミングクリティカルな FF に FI を行う実装を追加したのち，CGF を実行することで，「遅延故障発生の有無を考慮しつつ，どのようなテストパターンが計算結果の制約を違反させ得るか」自動探索させる狙いがある．すなわち，CGF の変異と PUT 実行において，遅延故障の有無と計算品質を考慮したフィードバックループを構築できると期待される．以上より，3.1 節から 3.3 節までを統合することにより，AC 回路のタイミング検証を実現する．

4. 予備実験

本章では，3 章で提案したタイミング検証法の実現に向けて，予備実験として，高位合成ツールを用いたクリティカルパスの抽出と解析を行う．

本予備実験では，高位合成ツールとして，NEC 社製の商

用ツールである CyberWorkBench (CWB) 8.3 Enterprise を使用した．また，CWB で高位合成するベンチマークプログラムとして，S2CBench の FIR (Finite Impulse Response; フィルタ回路) を使用した．本実験には，Ubuntu 16.04 LTS と AMD Ryzen Threadripper 3990X 64-Core Processor を搭載した計算機を用いた．

本実験では，高位合成時のクロック周期として，1 ns から 15 ns まで 1 ns 刻みで 15 種類用意した．各クロック周期を用いて，FIR フィルタ回路の高位合成を実施した．その後，高位合成後のレポートファイルを元に，各合成におけるクリティカルパス遅延を取得した．

評価結果を図 4 に示す．この図では，横軸が高位合成時に指定したクロック周期，縦軸が合成後に得られたクリティカルパスの遅延を表す．なお，クロック周期を 1 ns と 2 ns に設定した際には，セットアップ制約を満足できず，高位合成が正常に終了しなかったため，図 4 には記載していない．図 4 より，合成時に指定したクロック周期によって，クリティカルパスの遅延が大きく変動していることが読み取れる．例えば，クロック周期を 5 ns に指定した際のクリティカルパス遅延は 2.90 ns である一方，クロック周期を 6 ns に指定した際の遅延は 4.68 ns であった．また，一般に合成時のクロック周期を長くした際には，セットアップスラックを高性能化や小面積化に活用してクリティカルパスの遅延も長くなると予想されるが，その予想に従わない結果も観測した．例えば，クロック周期を 8 ns に指定した際の遅延は 2.21 ns であった．このような動作は，高位合成ツールの最適化アルゴリズムにおける，一種のヒューリスティック性によるものだと推察される．従って，タイミングクリティカルな FF を適切に抽出して，3 章で提案したタイミング検証法を実現するためには，ヒューリスティック性を考慮したアーキテクチャ探索法が重要であると考えられる．アーキテクチャ探索法の検討については，今後の課題の一つである．

5. 結論と今後の課題

本研究では，AC 回路のタイミング検証法を提案した．提案手法の肝は，高位合成を用いたタイミング特性の抽出，FI を応用した遅延故障の模擬，及び，ファジングを用いた効率的なテストパターン生成にある．高位合成と FI を活用し，CGF 実行時に遅延故障を疑似的に発生させることで，遅延故障発生と計算品質違反の有無を考慮可能なファジングテストを構築する狙いがある．予備実験として，高位合成ツールを用いたクリティカルパスの抽出と解析を行った．評価結果より，合成時のクロック周期を長くした際に，クリティカルパスの遅延が短くなるケースを観測した．

今後の展望としては，演算器，メモリ等の回路資源制約も含めた多様な条件でのアーキテクチャ探索，抽出したタイミング情報に基づく FI 箇所と時刻の決定，FI 機構を実装した HDL に対する CGF の実行が挙げられる．また，CGF を用いたタイミング検証結果を，タイミング設計最適化に還元する方法の検討も，今後の課題の一部である．

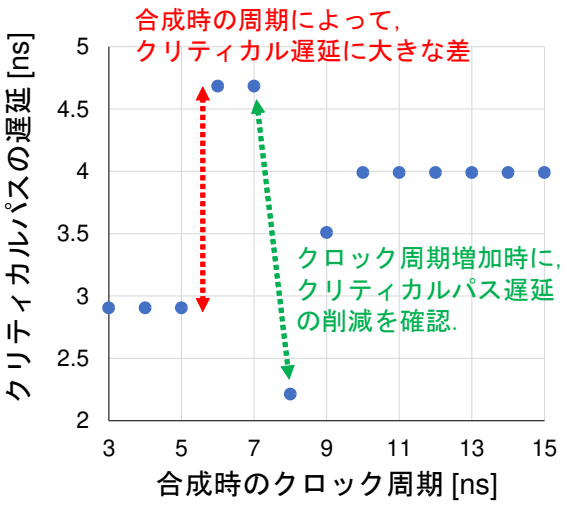

図 4: 高位合成時のクロック周期とクリティカルパス遅延の関係

謝辞 本研究の一部は JSPS 科研費 (20K19767) および JST，さきがけ，JPMJPR20M9 の助成を受けたものである．

参考文献

[1] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” *Proc. ETS*, pp. 1-6, 2013.
[2] Q. Xu, T. Mytkowicz, and N. S. Kim, “Approximate computing: A survey,” *IEEE Design & Test*, vol. 33, no. 1, pp. 8-22, 2016.
[3] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, “A survey of testing techniques for approximate integrated circuits,” *Proc. IEEE*, pp. 1-17, 2020.
[4] V. J. M. Manès et al., “The art, science, and engineering of fuzzing: A survey,” *IEEE TSE*, Oct. 2019.
[5] B. P. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of UNIX utilities,” *Communications of the ACM*, vol. 33, no. 12, pp. 32–44, 1990.
[6] J. E. Forrester and B. P. Miller, “An empirical study of the robustness of Windows NT applications using random testing,” *Proc. USEC*, Aug. 2000.
[7] P. Godefroid, M. Y. Levin, and D. A. Molnar, “Automated whitebox fuzz testing,” *Proc. NDSS*, pp. 151–166, 2008.
[8] K. Yoshisue, Y. Masuda and T. Ishihara, “Dynamic verification of approximate computing circuits using coverage-based grey-box fuzzing,” *Proc. IOLTS*, 2021.
[9] D. Ma, X. Zhang, K. Huang, Y. Jiang, W. Chang, and X. Jiao, “DEVoT: Dynamic delay modeling of functional units under voltage and temperature variations,” *IEEE TCAD*, 2021.
[10] M. Zalewski, “American Fuzzy Lop,” http://lcamtuf.coredump.cx/afl/.