ハードウェアトロイの検出および無効化に関する研究

Hardware Trojan Detection and Invalidation Methods

2019年2月

大屋　優

Masaru OYA

ハードウェアトロイの検出および無効化に関する研究

Hardware Trojan Detection and Invalidation Methods

2019年2月

早稲田大学大学院 基幹理工学研究科
情報理工・情報通信専攻 情報システム設計研究
大屋　優
Masaru OYA

目 次

第 1 章 序論 **3**
1.1 本論文の研究背景と目的 3
1.2 本論文の概要 8

第 2 章 ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法 **12**
2.1 本章の概要 12
2.2 トロイネット検出 12
2.2.1 トロイネットの特徴 15
2.2.2 スコア 16
2.2.3 一定値を出力する最大のクロックサイクル数 17
2.2.4 最大スコアネット数 18
2.2.5 強トロイネット 19
2.3 スコアに基づいたハードウェアトロイの有無を識別する手法 19
2.3.1 アルゴリズム 19
2.3.2 段階 1: 弱分類 20
2.3.3 段階 2: 強分類 20
2.3.4 段階 3: トロイ識別 21
2.4 実験結果 21
2.4.1 弱分類の結果 21
2.4.2 強分類の結果 24
2.4.3 トロイ識別の結果 24
2.4.4 提案手法と既存手法の性能比較 24
2.5 本章のまとめ 25

第 3 章 ゲートレベルネットリストの危険性を表現する指標 **28**
3.1 本章の概要 28
3.2 トロイネットの特徴に基づいた HT rank 28
3.2.1 HT rank の概要 29
3.2.2 キャラクタリスティックポイント (C-ポイント) 31
3.2.3 スケールポイント (S-ポイント) 35
3.2.4 ロケーションポイント (L-ポイント) 35
3.2.5 Hardware-Trojans rank 37
3.3 実験結果 37
3.3.1 FANCI vs. HT rank 37

3.3.2 Trust-HUB ベンチマークへの HT rank 適用結果 38
3.3.3 Trust-HUB，ISCAS85，ISCAS89，ITC99，OpenCores，ハードウェアトロイ有り/無し AES への HT rank 適用結果 39
3.4 本章のまとめ 42

第 4 章 回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法 43
4.1 本章の概要 43
4.2 ハードウェアトロイの分類 43
4.2.1 トリガによるハードウェアトロイの分類 43
4.2.2 ペイロードによるハードウェアトロイの分類 44
4.3 信号遷移 45
4.3.1 信号遷移ベクタ 45
4.3.2 定常信号遷移ベクタ 46
4.4 定常状態の学習に基づくハードウェアトロイ検出 46
4.4.1 Phase 1: 定常信号遷移ベクタの学習 47
4.4.2 Phase 2: ネットの振る舞いの監視 47
4.4.3 Phase 3: ハードウェアトロイの機能を推定する 50
4.5 実験結果 53
4.5.1 セットアップ 53
4.5.2 Phase 1: 定常信号遷移ベクタの学習結果 55
4.5.3 Phase 2: ネットの振る舞いの監視結果 55
4.5.4 Phase 3: ハードウェアトロイの機能の推定結果 56
4.5.5 既存の論理テストベース手法との比較 57
4.6 本章のまとめ 57

第 5 章 ハードウェアトロイ機能の無効化 62
5.1 本章の概要 62
5.2 内部トロイ無効化 62
5.2.1 無効化条件 64
5.2.2 無効化回路アーキテクチャ 65
5.2.3 マスキング回路アーキテクチャ 66
5.2.4 使用例 66
5.3 実験結果 70
5.3.1 無効化条件の最適化 70
5.3.2 ハードウェアトロイの挿入された 128 ビット AES への内部トロイ無効化回路実装 71
5.3.3 内部トロイ無効化 vs UCI 71
5.4 本章のまとめ 72

第 6 章 結論 73

謝辞 **77**

参考文献 **78**

研究業績 **83**

図 目 次

1.1 3PIP を含む IC. 4

2.1 ハードウェアトロイモデル. 13
2.2 弱トロイネット. 14
2.3 1 メガクロック間のシミュレーションを行った場合の一定値を出力する最大のクロックサイクル数. 16

3.1 ロケーションポイント. 30

4.1 トリガによるハードウェアトロイの分類. 44
4.2 ペイロードによるハードウェアトロイの分類. 44
4.3 ネット i の信号遷移が 2 つの 10 クロック間のセクションで構成される例. . 45
4.4 ハードウェアトロイの振る舞い. 52
4.5 Phase 1 の実験結果: EthernetMAC10GE-T710，s35932-T300 と s38584 の定常信号遷移ベクタ. 58
4.6 Phase 2 の実験結果: EthernetMAC10GE-T710，s35932-T300 と s38584 の信号遷移ベクタ. 59

5.1 内部トロイ無効化. 63
5.2 内部トロイ無効化回路. 63
5.3 トロイネットと疑われるネットが有効化される例（$(M, N) = (2, 3)$）. . . . 66
5.4 トロイネットと疑われるネットが有効化されない例（$(M, N) = (2, 3)$）. . . 67

表 目 次

2.1 ランダムに選んだゲートレベルのネットリスト. 16
2.2 弱トロイネットのスコア. 17
2.3 ランダムに選んだベンチマークの最大スコア. 17
2.4 最大スコアが 3 未満のトロイネットの一定値を出力する最大のクロックサイクル数. 18
2.5 一定値を出力する最大のクロックサイクル数が 999,996 以上のネット. . . . 18
2.6 強トロイネットによる識別結果. 20
2.7 ハードウェアトロイの挿入されているネットリスト. 22
2.8 ハードウェアトロイの挿入されていないネットリスト. 22
2.9 ケースネット数と最大スコアと最大スコアネット数. 23
2.10 トロイ識別の結果. 26
2.11 提案手法と既存手法の性能. 27

3.1 Trust-HUB のハードウェアトロイ有りゲートレベルネットリスト. 32
3.2 Trust-HUB のハードウェアトロイ無しゲートレベルネットリスト. 32
3.3 Trust-HUB ベンチマークの最大 C-ポイントと最大 C-ポイントネット数. . . 33
3.4 トロイポイントの詳細. 34
3.5 スケールポイントとロケーションポイントの内訳. 35
3.6 FANCI vs 提案手法. 38
3.7 各ベンチマークにおける HT rank. 40

4.1 1,000,000 クロックサイクルシミュレーションした際のハードウェアトロイの動作. 46
4.2 ベンチマークの説明. 54
4.3 実験で使用したパラメータ. 55
4.4 Phase 2 の実験結果. 60
4.5 Phase 3 の実験結果. 61
4.6 既存の論理テストベース手法との比較. 61

5.1 Trust-HUB のハードウェアトロイ有りと無しのゲートレベルネットリストに含まれるトロイネットと疑われるネット数. 68
5.2 トロイネットと疑われるネットの説明. 68
5.3 Trust-HUB ベンチマークの結果. 69
5.4 ハードウェアトロイの挿入された 128 ビット AES における結果 $((M, N) = (2, 5))$. 71
5.5 面積及び遅延オーバーヘッドの比較. 72
5.6 UCI との比較. 72

第1章　序論

1.1　本論文の研究背景と目的

近年，デジタル回路設計はコスト削減のために集積回路 (IC) の設計・製造工程を外部に委託するようになってきた．IC の設計・製造が自社内で完結しないようになり，第三者が関わるようになってきたことで，ハードウェアトロイと呼ばれる IC に悪意のある機能を持つ回路を挿入される可能性が出てきた．例えば，2008 年の IEEE Spectrum [12] によると，2007 年 9 月のシリアへの核関連施設へのイスラエルの空爆が成功したのは，遠隔操作による監視レーダーを停止する「キルスイッチ」が発動したからだと報告されている．他にも，匿名でアメリカ合衆国国防総省に接触した，あるヨーロッパのチップメーカによると，マイクロプロセッサにリモートで発動するキルスイッチを作成することに成功したと書かれている．IC にハードウェアトロイを挿入されることは，特に国防面においてセキュリティ上の懸念となっている．

アメリカでは，2007 年に採択された Defense Advanced Research Projects Agency の Trusted IC Project [10] が嚆矢となり，ハードウェアトロイ検出研究は目覚ましい勢いで成長している．ハードウェアトロイ対策のアプローチとして特に注力しているのがサプライヤーの信頼性の保証である [6]．Trusted Foundry Program [4] では，IC の設計・製造において信頼のおけるサプライヤーの選定を行い，実際に信頼性があると認定されたサプライヤーのリストを Defense MicroElectronics Activity の Web ページにて掲載している [3]．フランスでは，政府が Hardware trOjans : Menaces et robustEsse des ciRcuits intEgres. プロジェクト [5] で，積極的にハードウェアトロイ研究に対して援助を行っている．このプロジェクトの下で多くの研究が行われており，官と学とが一体となってハードウェアトロイへの対策をおこなっているのが窺える．

10 年前からハードウェアトロイへの脅威が認知され種々の取り組みが行われていたが，2018 年 10 月に The Big Hack [2] という記事が Bloomberg Businessweek によって掲載された．この記事はある国が Supermicro 社のサーバ用マザーボードにスパイチップを仕込んだという記事内容である．セキュリティの専門家の間では技術的な根拠にかけ信頼性に乏しいという見方が広まっているものの，当該記事の波紋は大きく，Apple，Amazon，Supermicro の CEO が即座にその内容を否定したり，米上院議員 2 人がスーパーマイクロにハードウェア部品改ざんの証拠が見つかったのかどうかなどを尋ねる書簡を送付 [11] させた程である．重要なことは記事の内容の真偽ではなくハードウェアインプラントは可能であり，コンピュータ機器

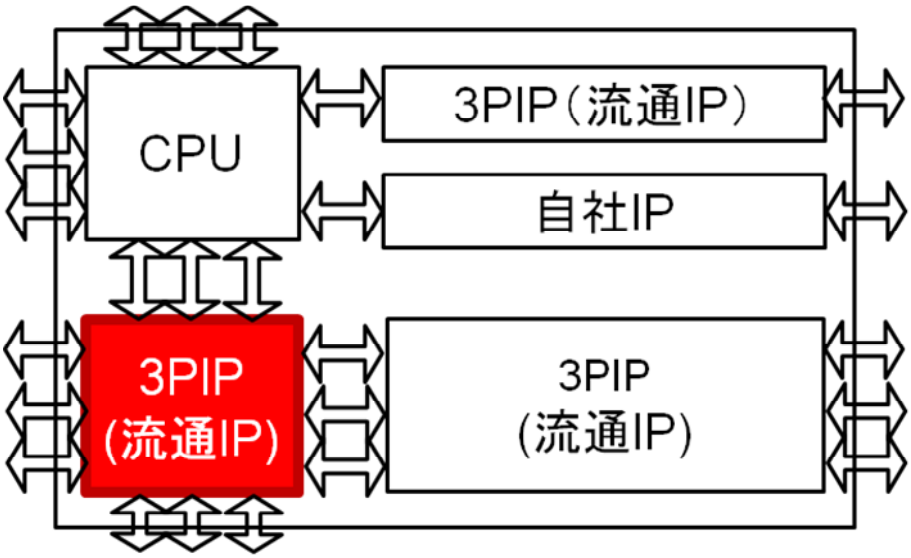


図 1.1: 3PIP を含む IC.

の製造は中国で行われており，そのような機器への安全性を検証に力を入れていないことは事実であるため，ハードウェアへのセキュリティ対策への危機意識が不十分なことである．

ここで大切なのは，サプライヤーの選定もハードウェアトロイ対策として有効ではあるものの，全てのサプライヤーを管理することは現実的には不可能であり，最終的には IC 自体の安全性を検証する技術によって安全性を担保する必要があるという認識である．IC は Intellectial Property (IP) と呼ばれる特定の機能を持った回路を利用して構成することが多い．IP は頻繁に利用される機能を容易に再利用できることを目的としており，特に基礎的な機能に関しては汎用 IP と呼ばれており，自社設計の IP との違いからサードパーティ IP (3PIP) と呼ばれたりもする．例えば，図 1.1 に 3PIP を含む IC を示す．赤色で塗りつぶされた 3PIP は，安全性が疑わしい IP だとする．IC 自体の安全性を検証するということは，赤色の 3PIP にハードウェアトロイが含まれているかを検査することを意味する．現在の IC では 3PIP が使用されることは常であるにも関わらず，IP・IC の安全性を検査する技術が未発達である．

ハードウェアトロイは IC の全ての設計・製造工程において挿入可能ではあるものの，製造工程よりも設計工程の方がハードウェアトロイの製作・挿入の技術的な障壁が低く，容易に挿入できることは知られている．なぜなら，デジタル回路設計の設計工程ではソフトウェアによって IC の設計が行われるためである．設計工程でハードウェアトロイを挿入する利点は，ソフトウェアの改竄で悪意のある機能を実現することができるため，チップとして残らず正規の IC の機能の一つとして紛れてしまい検出が困難になる点である．一方，製造工程でハードウェアトロイを挿入することを考えると，悪意のある機能を行うチップが形として残ってしまうという欠点がある．このような背景により，設計工程におけるハードウェアトロイに着目した研究開発の重要性と緊急性が高まっているのが現状である．

例えば，ハードウェアトロイの代表的な機能として，情報漏えいと機能低下が挙げられる．

秘密漏えいに関しては，暗号回路中の特定レジスタの値をトリガとして、秘密鍵やラウンド鍵等を I/O にバイパスする等が該当する．機能低下に関しては，演算回路部分の信号をトリガとして，演算結果に影響が出ないように元の回路に信号が伝搬しないような余分な回路を追加することで通常よりも演算に要する消費電力を増大させる等が該当する．どちらの機能も脅威だが，特に暗号回路を含む IC に対して情報漏えいの機能を持ったハードウェアトロイの挿入が深刻なセキュリティ上の懸念となっている．

設計工程における耐ハードウェアトロイの研究は，ハードウェアトロイ対策を施した設計とハードウェアトロイを検出する手法に大別できる．ハードウェアトロイ対策を施した設計を詳細に分類すると，耐リバースエンジニアリング [36, 43]，脆弱性解析 [40]，テストポイント挿入 [46]，論理撹拌 [34, 38]，ハードウェアトロイ挿入妨害 [48]，ハードウェアトロイ除去 [19] となる．耐リバースエンジニアリングはオリジナルの設計に対して特別なゲートを実装することでリバースエンジニア対策をする．脆弱性解析はハードウェアトロイが挿入されたら検出が困難な場所を特定する．ハードウェアトロイ検出技術は様々なハードウェアトロイの特徴に着目し，ハードウェアトロイの存在を明らかにする．テストポイント挿入はフリップフロップを挿入し，観測性および制御性を高めることでハードウェアトロイの検出を容易にする．論理撹拌は論理値を攪乱させるために余分な論理ゲートを実装することで論理値を撹拌させ，悪意のある第三者に正常な値が得られないようにする．ハードウェアトロイ挿入妨害は IC の使われていないスペースをフィルターセルで満たすことで，ハードウェアトロイが挿入できないようにする．ハードウェアトロイ除去は危険な箇所を特定し，それらを取り除くことで，ハードウェアトロイの脅威に対抗する．

ハードウェアトロイ検出手法 [13–25,30–33,35,41,42,44,47,49–52] は通常，サイドチャネルか論理テストに基づいている．サイドチャネルベースのアプローチ [13,14,17,21–25,35,49,50] は，ハードウェアトロイが挿入されることでオリジナルの回路の設計が破壊されることに着目し，サイドチャネル（面積，遅延時間，消費電力等) の変化を計測する．しかし，このアプローチはサイドチャネルの変化を計測するために，ハードウェアトロイの挿入されていないネットリスト（ゴールデンネットリスト）が必要であることとプロセスバラつきの影響を受けやすいという問題がある．加えて，[45] で設計されたサイドチャネルに影響を与えないようなハードウェアトロイを検出するのは困難であることが知られている．

論理テストベースのアプローチは，適切なテストパターンを生成してハードウェアトロイを動作させ，悪意のある値やロジックを検出することでハードウェアトロイを検出する．Wolff らの手法 [47] がハードウェアトロイを動作させるテストパターンを生成するアプローチの研究の嚆矢である．Rajendran らの手法 [37] は重要なレジスタを監視することで，悪意のある値の伝搬を検知する．加えて，Saha ら [39] の遺伝的アルゴリズムとブール充足可能性判定を組み合わせたテストパターン生成スキームである．これらの手法は，ハードウェアトロイの影響を出力に伝搬するテストパターンを効率よく生成することができる．しかし，これらの手法はハードウェアトロイを動作させる必要があり，ハードウェアトロイは通常のテスト工程で検出されないように設計されており，動作させるテストパターンの生成は厳し

い前提条件となる．

設計工程における様々なハードウェアトロイ対策手法を例示したが，本論文では設計工程におけるハードウェアトロイ検出手法に注目する．それは，ハードウェアトロイ検出手法が IC 自体の安全性を検証するという目的に沿った技術だからである．ハードウェアトロイ対策を施した設計は，ハードウェアトロイを挿入する技術的障壁を高くすることが目的であり，IC 自体が安全かどうかを検証する技術ではない．これもハードウェアトロイ対策として必要ではあるものの，最終的に求められる技術は IC にハードウェアトロイが挿入されていたら検出し，挿入されていない場合は検出しないという性質を持つはずである．本論文では設計工程におけるハードウェアトロイの検出に着目するため，Supermicro 社の事例のような製造工程での挿入の検出は想定していない．製造工程では製造工程でのハードウェアトロイ検出が必要であり，このようなハードウェアトロイは本論文の検出対象とするハードウェアトロイのスコープからは外れることを明記しておく．

加えて，本論文で対象としないハードウェアトロイとして，常に動作するハードウェアトロイが挙げられる．このようなトリガを持たない Always on 型のハードウェアトロイは，性能もしくは機能検証時で検出すべきだと考えているためである．ハードウェア設計の場合，サードパーティ IP を使用するとしても，性能が正しいのか機能が正しいのかということに対して厳密に検査する．Always on 型の場合だと，常に性能もしくは機能に対して影響を与えてしまい，注意深く検証することで多くは検出することが可能だと考えることができる．実際の脅威とは，IP として求められる性能と機能に影響を与えないハードウェアトロイであり，このようなハードウェアトロイを検出することは非常に困難であり，特別な検出手法が必要になる．これには内部状態をトリガとするハードウェアトロイが該当し，本論文ではこのハードウェアトロイを検出対象とする．

本論文は既存研究の抱える以下の 3 つの問題を克服することを目的とする．既存研究の 3 つの問題とは，ハードウェアトロイを検出するアプローチに偏りがあること，検出できるハードウェアトロイの種類が少ないこと，安全かそうでないかが曖昧なネットへの対策が不十分なことである．

ハードウェアトロイを検出するアプローチとして，半導体の自動設計の技術を利用した論理テストやサイドチャネル解析を採用した研究が数多くあった．しかし，ハードウェアトロイは既存の技術のみで解決できる問題ではないため，ハードウェアトロイに特化した新技術を研究開発する必要がある．そこで，本論文ではハードウェアトロイの持つ回路構造の特徴を明らかにするという既存の研究とは全く違ったアプローチをとることにした．このアプローチにより，ハードウェアトロイ検出において新しいパラメータを確立した．従来の悪意のある第三者は，既存技術に精通していれば検出が困難なハードウェアトロイを設計することができた．しかし，本論文は論理テストやサイドチャネル解析とは独立しているアプローチを確立したため，悪意のある第三者が検出されないハードウェアトロイを設計することが困難になった．

ハードウェアトロイにも発動条件や機能により様々な種類がある．基本的にハードウェア

トロイは，トリガ回路とペイロード回路で構成されている．トリガ回路は悪意のある機能が動作する条件を担当し，ペイロード回路は悪意のある機能の動作そのものを担当する．特にトリガ回路が組み合わせ回路ではなく，順序回路で構成されている場合には検出が困難だった．また，多くの検出手法がハードウェアトロイで生成された信号はオリジナルの回路に伝搬するというモデルに基づいていた．しかし，消費電力の増大等の機能を実現する場合，ペイロード回路が生成した信号がオリジナルの回路に伝搬せずに，ハードウェアトロイ回路内で閉じることでも実現することができる．そこで，本論文では回路の I/O の変化ではなく，ネットそのものの信号遷移に着目することで，ペイロードの回路が生成した信号がハードウェアトロイ回路内で閉じる場合でも検出することに成功した．

既存研究では，ハードウェアトロイだと疑われる危険なゲートやネットを検出してくれるが，その先を助けてくれない．実際には，危険なのか安全なのか明確に判断するのが困難だと思われるゲートやネットが存在し，それらに対しても対処をしなければならない．安全性が厳密に求められる場合は，設計ファイルの再設計や廃棄等ができるが，高コストであり多くの場合には難しいと思われる．そこで，本論文では怪しいネットを監視して，ハードウェアトロイかどうかを識別して悪意のある機能を無効化する回路を作成した．この無効化回路により，検出手法を適用した後の対策を提示することに成功した．

本論文で提案する手法により，設計工程における IC に挿入されたハードウェアトロイの検出および無効化を達成し，IC 自体の安全性検証に貢献する．

1.2 本論文の概要

本論文で提案する手法の目的と概要を示す.

2 章「ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法」では，ハードウェアトロイに含まれるネット（トロイネットと呼ぶ）の特徴に注目し，トロイネットを検出することでハードウェアトロイを検出する手法を提案する．

本手法の目的は，ハードウェアトロイを検出するアプローチに偏りがある問題を解決することである．この問題を解決するためには，論理テストやサイドチャネル解析とは違うアプローチでハードウェアトロイを検出する必要性がある．本手法では，Trust-HUB [9] ベンチマークと呼ばれるハードウェアトロイのベンチマークから，ハードウェアトロイの回路構造の特徴を抽出することで，ハードウェアトロイの検出を可能にした．本手法の成果により，全く新しいアプローチでハードウェアトロイ検出に成功したが，より詳しいハードウェアトロイの識別に論理テストを使用しているので，ハードウェアトロイを検出するアプローチに偏りがある問題を完全に解決したとは言えない．この問題の解決は，3 章「ゲートレベルネットリストの危険性を表現する指標」で試みている．

本手法の概要は以下である．まず，与えられたゲートレベルのネットリストに含まれる全てのネットにスコアを与える．スコアはハードウェアトロイの挿入されているベンチマークからトロイネットを抽出し，抽出したトロイネットの特徴に応じてスコアの重みづけを行う．次に，トロイ要素という 3 つのパラメータを最大のスコアを持つネットに設定する．最後に，3 つのトロイ要素の値とそれらの閾値を比較することで，トロイネットの可能性が最も高いトロイネットを検出する．強トロイネットが含まれていれば，ハードウェアトロイが挿入されていると判断する．提案手法は与えられたゲートレベルのネットリストから得られる情報のみを使用するため，挿入されているハードウェアトロイの情報を必要としない．それでいて，提案手法は与えられたネットリストにハードウェアトロイが挿入されているか否かを識別する．実際に，Trust-HUB の Abstraction Gate Level で公開されている全てのゲートレベルネットリストに対して，ハードウェアトロイの有無を識別することに成功した．提案手法を適用する際に要する時間は高々数時間程度である．

3 章「ゲートレベルネットリストの危険性を表現する指標」では，ゲートレベルネットリストの危険性を表現する指標として *HT rank* を提案する．

本手法では，2 章「ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法」で完全に解決できなかった，ハードウェアトロイを検出するアプローチに偏りがある問題を解決することである．本手法は，2 章「ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法」で明らかにしたハードウェアトロイの回路構造の特徴に加え，プライマリインプット・プライマリアウトプット・フリップフロップの位置等の情報を考慮することで，回路構造の特徴のみでハードウェアトロイを識別することに成功した．これにより，論理テストやサイドチャネル解析を利用しない，新規のハードウェアトロイ検出手法アプローチを確立した．したがって，本手法はハードウェアトロイ

を検出するアプローチに偏りがある問題を解決した．

本手法の概要は以下である．HT rank を設計するにあたり，全てのゲートレベルの Trust-HUB ベンチマーク [9] を解析し，いくつかのトロイネットの特徴を発見する．解析結果に基づいて，3 種類のトロイポイントを開発する．1 個目はキャラクタリスティックポイント（C-ポイント），2 個目はスケールポイント（S-ポイント），3 個目はロケーションポイント（L-ポイント）である．これらのポイントを全てのネットに割り当てることで，ゲートレベルネットリスト内の最大のトロイポイントを持つネット（最大トロイポイントネット）を検出する．最大トロイネットポイントはネットリスト内で最もトロイネットであると疑われるネット (疑トロイネット) であり，最大トロイポイントが HT rank となる．HT rank の値が高いほど，与えられたネットリストはハードウェアトロイが挿入されている可能性が高いことを示す．一方 HT rank の値が低いほど，与えられたネットリストはハードウェアトロイが挿入されていない可能性が高いことを示す．HT rank は他にも設計者にハードウェアトロイと疑われる部分の位置を提供する．HT rank はネットの特徴に基づいて計算されるため，シミュレーションやランダムテストを使用しない．それでいて，ゴールデンネットリストも使用しない．したがって，HT rank は厳密で一意な識別結果を得られる．HT rank はゲートレベルネットリストの危険性を表現することができ，悪意ある回路の位置に関するヒントを提供する．実際に HT rank は全ての Trust-HUB，ISCAS85，ISCAS89，ITC99 のゲートレベルネットリストに加え，いくつかの OpenCores ゲートレベルネットリスト, そしてハードウェアトロイの挿入されている AES と挿入されていない AES に対して，ハードウェアトロイの有無を分類することに成功した．

4 章「回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法」では，定常状態の学習を利用した論理テストベース手法を提案する．

本手法の目的は，検出できるハードウェアトロイの種類が少ない問題を解決することである．本手法は，トリガ回路が組み合わせ回路もしくは順序回路のどちらで構成されていても検出することができる．加えて，ペイロード回路で生成された信号がオリジナルの回路に伝搬する場合もしくはオリジナルの回路に伝搬せずにハードウェアトロイ回路内で閉じる場合のどちらでも検出することができる．したがって，本手法は検出できるハードウェアトロイの種類が少ない問題を解決した．

本手法の概要は以下である．本手法は短時間ランダムシミュレーションを通じ，回路の信号遷移の定常状態を学習することでハードウェアトロイを検出する．多くのハードウェアトロイは特別な条件を満たした時にのみ動作するように設計されており，短時間シミュレーションでランダムにテストパターンを与えてもハードウェアトロイは動作しないままである．この状態の信号遷移を定常状態として学習し，長時間ランダムシミュレーションによりハードウェアトロイを動作させることで，ハードウェアトロイを検出することができる．ハードウェアトロイは短時間ランダムシミュレーション中は動作しないで隠ぺいされたままである．これはほとんどのハードウェアトロイが特別な条件を満たした時にのみ動作するよう設計されているからである．したがって，各疑トロイネットにおける定常信号遷移状態を短時間ラ

ンダムシミュレーションによって得る．定常信号遷移状態は一定のクロックサイクル間の 4 次元ベクタによって定義される．4 次元ベクタとは，0 を保持し続けるクロックサイクル数の合計，1 を保持し続けるクロックサイクル数の合計，クロックの立ち上がり数の合計，クロックの立ち下がり数の合計である．ハードウェアトロイは短時間ランダムシミュレーションでは動作しないが，長時間のシミュレーションでは動作する可能性が極めて高い．それは，ハードウェアトロイは稀に動作するように設計されているが，長時間のシミュレーションを行うことでハードウェアトロイが動作することが期待できるからである．そこで，各疑トロイネットに対して長時間のシミュレーションを行い，信号遷移ベクタの集合を得る．定常信号遷移ベクタと長時間シミュレーションで得た信号遷移ベクタを比較し，定常信号遷移ベクタと異なる場合は異常状態が存在するため，ハードウェアトロイと判断する．加えて，異常状態が検出された場合は更に分析することで，ハードウェアトロイの機能を推測する．ハードウェアトロイのタイプはトリガとペイロードによってそれぞれ二種類のタイプに分類できる．トリガによる分類は，組み合わせ回路をトリガとするハードウェアトロイと順序回路をトリガとするハードウェアトロイである．ペイロードによる分類は，観測可型トロイと観測不可型トロイである．提案手法は組み合わせ回路・順序回路をトリガとしたハードウェアトロイと観測可型トロイ・観測不可型トロイの全てを検出することができる．実際に Trust-HUB ベンチマークに対して，組み合わせ回路をトリガとしたハードウェアトロイ，順序回路をトリガとしたハードウェアトロイ，観測可型トロイ，観測不可型トロイを検出し，ハードウェアトロイの機能を推測することに成功した．

5 章「ハードウェアトロイ機能の無効化」では，内部トロイ無効化技術を提案する．

本手法の目的は，安全かそうでないかが曖昧なネットへの対策が不十分な問題を解決することである．本手法では，ハードウェアトロイ検出手法では安全かどうかを判断できなかったネットに対して監視回路を取り付ける．一定クロックの間監視を続けた後，監視結果に基づいてネットの値をマスキングするかどうかを決めることにより，シミュレーションで見つけられないハードウェアトロイに対しても実回路段階で無効化することができる．エンジニアとしてもネットリストの再設計や破棄といったコストの高い対策を取らずに運用できる．以上により，本手法は安全かそうでないかが曖昧なネットへの対策が不十分な問題を解決する．

本手法の概要は以下である．本手法は，まずハードウェアトロイ検出手法を適用して，疑トロイネットを抽出する．抽出した疑トロイネットを監視して，信頼性の低いネットリストに挿入されたハードウェアトロイの機能を無効化する．提案手法のアプローチは，信頼性の低い IC を監視モードとノーマルモードで動作させる．監視モードでは，埋め込まれたトロイ無効化回路が疑トロイネットの一定クロック間のビットフリッピング回数を監視し，本当のトロイか否かを識別し，それらが本当にトロイであるか否かを判断する．もし無効化条件を満足したら，疑トロイネットはノーマルネットと判断するため有効化させる．一方，トロイネットと判断した場合は無効化し，トロイの機能をマスキングする．これにより，ハードウェアトロイが挿入されている信頼性の低いネットリストをノーマルモードでも安全に動作させることができる．頻繁に動作するハードウェアトロイは回路検証で容易に検出されてし

まうため，賢いハードウェアトロイは滅多に動作しないように設計されている．このハードウェアトロイの特徴に基づき，提案した内部トロイ無効化技術は条件により，ノーマルネットとトロイネットの相違を認識し，本当のトロイネットのみを無効化する．このアプローチにより信頼性の低いネットリストにハードウェアトロイが存在しても，安全に動作させることができる．いくつかの Trust-HUB ベンチマークをトレーニングセットとし，無効化条件を最適化する．適切な無効化条件を設定することにより，信頼性の低いネットリストにあるトロイネットのみを無効化することができる．一般的な 128 ビットの AES に内部トロイ無効化回路を埋め込み，例えハードウェアトロイがあろうとも安全かつ正常に動作することに成功した．面積オーバーヘッドは 0.79%だけであり，遅延オーバーヘッドは生じなかった．

6 章「結論」では，本論文を総括し，本論文の手法の妥当性について議論し，今後の課題を示す．

第2章　ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法

2.1　本章の概要

本章では，ハードウェアトロイに含まれるネット（トロイネットと呼ぶ）の特徴を明らかにし，トロイネットを検出するためのスコアリングアルゴリズムを示し，ゲートレベルのネットリストに対してスコアに基づいたハードウェアトロイの有無を識別する手法を提案する[1]．以下に本章の構成を示す．

第 2.2 節「トロイネット検出」では，ハードウェアトロイを検出するための導入として，トロイネットというアイデアを説明し，トロイネットの特徴を説明する．トロイネットの特徴である回路パターン 9 個と信号遷移を説明し，これらの特徴とトロイネットを検出するための重みづけを与えるスコアの関係性を示す．

第 2.3 節「スコアに基づいたハードウェアトロイの有無を識別する手法」では，ハードウェアトロイの有無を識別する手法を提案する．提案手法は 3 つのプロセスで構成されている．1 つ目のプロセスは，与えられたゲートレベルのネットリストに含まれる全てのネットにスコアを与える．2 つ目のプロセスは，トロイ要素という 3 つのパラメータを最大のスコアを持つネットに設定する．3 つ目のプロセスは，3 つのトロイ要素の値ととそれらの閾値を比較することで，トロイネットの可能性が最も高い強トロイネットを検出する．強トロイネットが含まれていれば，ハードウェアトロイが挿入されていると判断する．

第 2.4 節「実験結果」では，提案手法のソフトウェア実験の結果を示し，提案手法の有効性を評価する．

第 2.5 節「本章のまとめ」では，本章の内容をまとめる．

2.2　トロイネット検出

本節で扱うハードウェアトロイモデルを図 2.1 に示す．ハードウェアトロイはトリガ回路とペイロード回路で構成されている．トリガ回路はハードウェアトロイを動作させるトリガ

[1]本章の内容は，文献 [27,30,55–57] による．

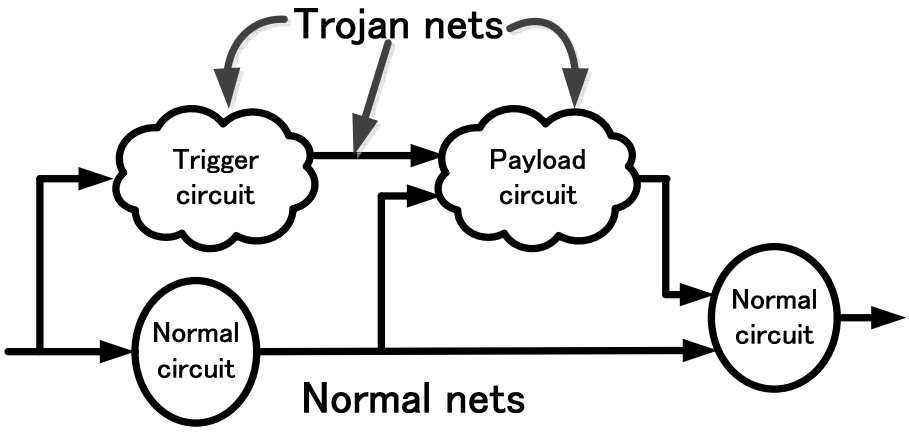
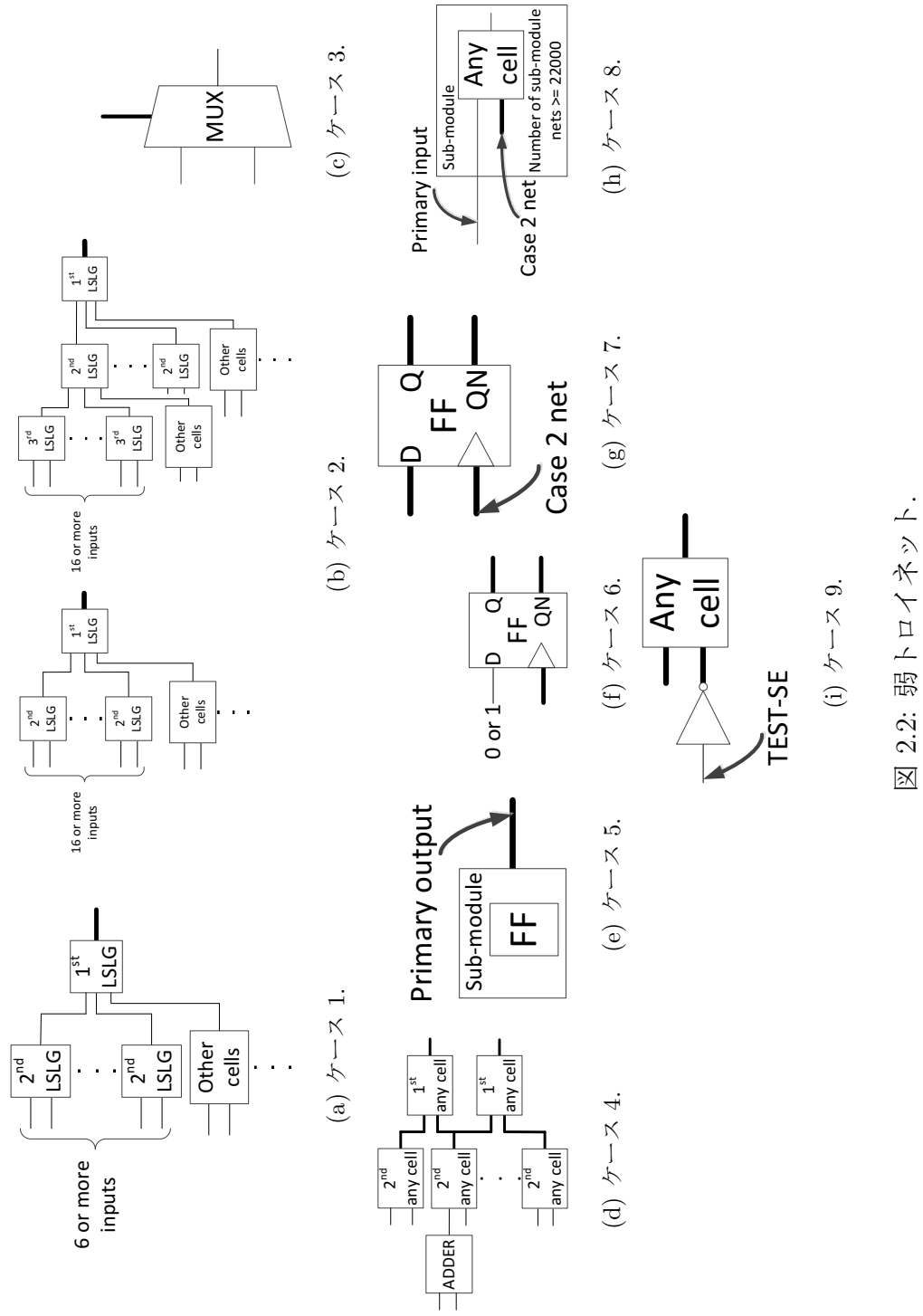


図 2.1: ハードウェアトロイモデル.

信号を生成する．ペイロード回路は実際のトロイの動作である．例えば，秘密鍵をプライマリアウトプットに出力する．トロイネットはハードウェアトロイのネットを指す．ノーマルネットは通常回路のネットを指す．検出対象のハードウェアトロイは，検出が困難な頻繁に動作しないハードウェアトロイとする．これは頻繁に動作するハードウェアトロイは，回路検証で容易に検出されるからである．

ハードウェアトロイの危険を完全に無くす方法として，回路からハードウェアトロイのみを取り除くことが理想である．しかし，ハードウェアトロイと通常回路を完全に区別することは難しい．そこで，検出対象をハードウェアトロイ回路全体ではなくトロイネットだけに注目する．トロイネットを検出することは，ハードウェアトロイ全体を検出することよりも容易である．そのため提案手法は一部のトロイネットを検出することで，ハードウェアトロイの有無を識別する．

本節では，トロイネットの特徴を抽出し，特徴と一致するネットにスコアを与えることで，トロイネットの可能性が高いネットを検出する方法を提案する．



(a) ケース 1. (b) ケース 2. (c) ケース 3.

(d) ケース 4. (e) ケース 5. (f) ケース 6. (g) ケース 7. (h) ケース 8.

(i) ケース 9.

図 2.2: 弱トロイネット.

2.2.1 トロイネットの特徴

トロイネットの特徴を抽出するために Trust-HUB [9] の Abstraction Gate Level で公開されているハードウェアトロイの挿入されているゲートレベルネットリストからランダムに7個選んだ．表 2.1 にランダムに選んだゲートレベルのネットリストを示す．ベンチマークは1つもしくは複数のモジュールで構成されており，論理ゲートや，フリップフロップ，アダーなどのセルで構成されている．

図 2.2 にトロイネットの特徴を示す．図 2.2 の特徴を持つネットを弱トロイネットと呼ぶ．図 2.2 において太線が弱トロイネットを示す．LSLG (low-switching logic gate) は AND，NAND，OR，NOR ゲートを意味する．LSLGs は低スイッチング確率のゲートであり，トリガ回路を主に構成していると考えられる．FF はフリップフロップ，Any cell はセル全てを指す．トロイネットの特徴を 9 つに分類したものを以下に示す．

ケース 1 (図 2.2(a)): 1st LSLG のインプットに 2nd LSLG のアウトプットが含まれており 2nd LSLG のインプットが 6 以上の場合，1st LSLG のアウトプットは弱トロイネットである．

ケース 2 (図 2.2(b)): 1st LSLG のインプットに 2nd LSLG のアウトプットが含まれており 2nd LSLG のインプットが 16 以上もしくは，3rd LSLG のインプットが 16 以上の場合，1st LSLG のアウトプットは弱トロイネットである．ケース 2 はケース 1 の特別な場合であり，ケース 1 とケース 2 の両方を満たすネットは Trojan Net である可能性が高い．

ケース 3 (図 2.2(c)): MUX の選択信号は弱トロイネットである．

ケース 4 (図 2.2(d)): ADDER のアウトプットをインプットに持つセルを 2nd any cell とし，2nd Cell のアウトプットをインプットに持つセルを 1st any cell とする．1st any cell のインプットとアウトプットは弱トロイネットである．ADDER は，ハーフアダーとフルアダーを指す．

ケース 5 (図 2.2(e)): FF を含むモジュールのプライマリアウトプットは弱トロイネットである．

ケース 6 (図 2.2(f)): 直接 ’0’ or ‘1’ が論理値として入力される FF のインプット，アウトプット，クロックは弱トロイネットである．

ケース 7 (図 2.2(g)): クロックがケース 2 のネットである FF のインプット，アウトプット，クロックは弱トロイネットである．

ケース 8 (図 2.2(h)): サブモジュールのネット数が 22000 以上であり，同じセルのインプットにプライマリインプットとケース 2 のネットがある場合，ケース 2 のネットはより強い弱トロイネットとなる．

ケース 9 (図 2.2(i)): test se 信号の否定信号を入力に持つセルのインプットとアウトプットは弱トロイネットである．

表 2.1: ランダムに選んだゲートレベルのネットリスト.

ベンチマーク	種類	ネット数
b19	トロイなし	108,332
EthernetMAC10GE	トロイなし	103,206
s35932	トロイなし	6,423
EthernetMAC10GE-T700	トロイあり	103,220
RS232-T1000	トロイあり	311
s15850-T100	トロイあり	2,456
s38417-T100	トロイあり	5,819
s38584-T200	トロイあり	7,580
vga_lcd-T100	トロイあり	70,162
wb_conmax-T100	トロイあり	22,197

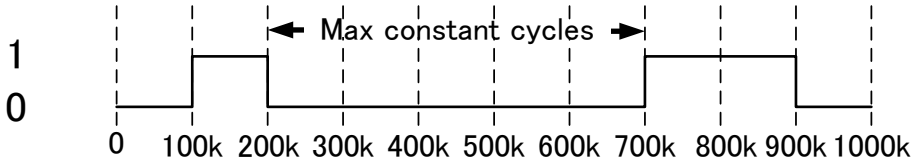


図 2.3: 1メガクロック間のシミュレーションを行った場合の一定値を出力する最大のクロックサイクル数.

2.2.2 スコア

トロイネットの特徴は図 2.2 に示した．しかし，ケース 1–9 に一致するネットは数多くある．そのため，より正確にトロイネットを検出するためにスコアによる重みづけを行う．

スコアを設定するために表 2.1 のベンチマークを注意深く解析したところ，弱トロイネットは2種類に分けられることが分かった．1つはノーマルネットとトロイネットの両方にみられる特徴であり，もう1つはトロイネットのみにみられる特徴である．前者はケース 1–5 であり，これらの弱トロイネットと一致するネットにはスコアを1点加算する．後者はケース 6–9 であり，これらの弱トロイネットと一致するネットにはスコアを2点加算する．表 2.2 に弱トロイネットのスコアを示す.

弱トロイネットのスコアに基づき，各ベンチマークの全てのネットに対してスコアを計算した．あるネットが複数の弱トロイネットと一致した場合，スコアをさらに加算し総和を取った．表 2.3 に各ベンチマークの最大のスコアを示す．表 2.3 によると，最大スコアが 3 以上のネットはトロイネットである.

しかし，最大スコアが 3 未満の場合でもネットリストはトロイネットを含む．したがって，ノーマルネットとトロイネットを分ける別の閾値が必要である.

表 2.2: 弱トロイネットのスコア.

弱トロイネット	スコア
ケース 1	1
ケース 2	1
ケース 3	1
ケース 4	1
ケース 5	1
ケース 6	2
ケース 7	2
ケース 8	2
ケース 9	2

表 2.3: ランダムに選んだベンチマークの最大スコア.

ベンチマーク	最大スコア	最大スコアネット数	最大スコアネットの中身
b19	2	77	ノーマルネットのみ
EthernetMAC10GE	2	55	ノーマルネットのみ
s35932	1	420	ノーマルネットのみ
EthernetMAC10GE-T700	6	1	トロイネットのみ
RS232-T1000	2	2	トロイネットのみ
s15850-T100	6	1	トロイネットのみ
s38417-T100	2	5	ノーマルネットとトロイネット
s38584-T200	3	1	トロイネットのみ
vga_lcd-T100	2	2	トロイネットのみ
wb_conmax-T100	4	1	トロイネットのみ

2.2.3 一定値を出力する最大のクロックサイクル数

多くのハードウェアトロイは特別な条件下で動作するように設計されているため，トロイネットは長時間同じ値を保持し続けていると予想される．そこで Synopsys 社の論理シミュレータである VCS を使用し，ランダムデータを 1 メガクロック間入力して，最大スコアネットの値を観測した．Max constatnt cycles は一定値を出力する最大のクロックサイクル数を示す．例えば，1 メガクロック間のシミュレーションを行った場合の図 2.3 の一定値を出力する最大のクロックサイクル数は 5000 サイクルである．

表 2.4 に表 2.3 に示した最大スコアが 3 未満のトロイネットの一定値を出力する最大のクロックサイクル数を示す．表 2.4 によると，Tj_OUT5678 の一定値を出力する最大のクロックサイクル数が最小値である．そのため，一定値を出力する最大のクロックサイクル数が 999,996 cycles 以上の場合，トロイネットであると予想される．

そこで，表 2.3 の最大スコアが 3 未満のネットに対して一定値を出力する最大のクロック

表 2.4: 最大スコアが3未満のトロイネットの一定値を出力する最大のクロックサイクル数.

ベンチマーク	トロイネットの名前	一定値を出力する 最大のクロックサイクル数
RS232-T1000	iRECEIVER_CTRL	999,999
	iCTRL	999,999
s38417-T100	Tj_Trigger	999,997
	Tj_OUT1234	999,997
	Tj_OUT5678	**999,996**
vga_lcd-T100	Tj_Trigger	999,999
	Tj_OUT1	999,999

表 2.5: 一定値を出力する最大のクロックサイクル数が999,996以上のネット.

ベンチマーク	最大 スコア ネット	一定値を出力する最大の クロックサイクル数が 999,996以上のネット数	中身
b19	2	59	ノーマルネットのみ
EthernetMAC10GE	2	10	ノーマルネットのみ
s35932	1	0	N/A
RS232-T1000	2	2	トロイネットのみ
s38417-T100	2	3	トロイネットのみ
vga_lcd-T100	2	2	トロイネットのみ

サイクル数を測定した．表2.5に一定値を出力する最大のクロックサイクル数が999,996以上のネットを示す.

表2.5によると，b19には59個，EthernetMAC10GEには10個のノーマルネットが一定値を出力する最大のクロックサイクル数が999,996以上であることがわかる．特に，b19には1メガクロックサイクル間，一定値を保持し続けるノーマルネットが存在した．これはb19とEthernetMAC10GEが他のベンチマークと比べて大規模なネットリストことが原因である．一定値を出力する最大のクロックサイクル数は小規模なネットリストに対しては有効であるが，大規模なネットリストに対しては誤検出を含む原因となってしまう．したがって，ノーマルネットとトロイネットを分ける別の閾値が必要である.

2.2.4 最大スコアネット数

表2.3によると，トロイネットを含む最大スコアネットの数は比較的少なく，ノーマルネットのみを含む最大スコアネットの数は比較的多い.

スコアによる重みづけの結果から，最大スコアが高ければ高いほど最大スコアネット数は減少し，最大スコアが低ければ低いほど最大スコアネット数は増加することがわかる．したがって，最大スコアネット数が十分に多い場合，ゲートレベルのネットリストにハードウェアトロイは挿入されていないと予想される．表 2.3 によると，トロイネットを含む最大スコアネット数の最大数は s38417-T100 の 5 である．したがって，最大スコアネット数が 6 以上の場合，そのゲートレベルのネットリストはハードウェアトロイが挿入されていないとする．

最大スコアネット数に上記の閾値を設定することで，遂に表 2.3 のベンチマークに対してハードウェアトロイが挿入されているか否かを完全に識別することができる．

2.2.5 強トロイネット

表 2.3 に示した各ベンチマークの各最大スコアネットを n とおく．n が以下の条件を満たした場合，n はトロイネットである．

1. n のスコアが 3 以上の場合, n はトロイネットである．
2. さもなければ，n の一定値を出力する最大のクロックサイクル数が 999,996 以上であり，
   ネットリストに含まれる最大スコアネット数が 5 以下の場合，n はトロイネットである．

上記の条件を満たしたネットを特に強トロイネットと呼ぶ．

表 2.6 に強トロイネットによる識別結果を示す．ハードウェアトロイの挿入されていないネットリストは，強トロイネットを 1 つも含まないことがわかる．一方，ハードウェアトロイの挿入されているネットリストは，強トロイネットを含むことがわかる．上記のスコアと閾値を用いることで強トロイネットを検出しているため，ゴールデンネットリストを使用せずハードウェアトロイに関する情報を一切必要としないで，ハードウェアトロイの有無を識別することに成功している．

2.3 スコアに基づいたハードウェアトロイの有無を識別する手法

2.3.1 アルゴリズム

本節では，スコアに基づいたハードウェアトロイの有無を識別する手法を提案する．提案手法は大まかに分類と識別の 2 つの処理で構成される．分類は弱分類と強分類の 2 つで構成される．提案手法を要約すると以下になる．

段階 1: 弱分類
段階 2: 強分類
段階 3: トロイ識別

表 2.6: 強トロイネットによる識別結果.

ベンチマーク	強トロイネット数	中身	識別
b19	0	N/A	トロイなし
EthernetMAC10GE	0	N/A	トロイなし
s35932	0	N/A	トロイなし
EthernetMAC10GE-T700	1	トロイネットのみ	トロイあり
RS232-T1000	2	トロイネットのみ	トロイあり
s15850-T100	1	トロイネットのみ	トロイあり
s38417-T100	3	トロイネットのみ	トロイあり
s38584-T200	1	トロイネットのみ	トロイあり
vga_lcd-T100	2	トロイネットのみ	トロイあり
wb_conmax-T100	1	トロイネットのみ	トロイあり

弱分類は，与えられたネットリストから最大スコアネットを検出する．最大スコアネットはまだノーマルネットとトロイネットを含む．次に，強分類でトロイ要素と呼ぶ3つのパラメータを各最大スコアネットに設定する．その後，トロイ識別により最大スコアネットから強トロイネットを抽出し，ネットリストにハードウェアトロイが挿入されているか否かを識別する．

2.3.2 段階 1: 弱分類

弱分類はスコアの最大値を持つネットを検出する．最大スコアネットはネットリスト中で最もトロイネットである可能性が高いネットを意味する．

初めは全てのネットのスコアは0である．各ネットに対して弱トロイネット (図2.2) と一致するかを判定し，表2.2に基づいてスコアを加算する．最大のスコアを持つネットが最大スコアネットとなる．

最大スコアの計算および最大スコアネットの検出にネットリストに含まれているハードウェアトロイの情報は必要ない．ただネットリストに含まれているネットが弱トロイネットと一致するかを判定するだけである．したがって，弱分類は前提1と前提2を前提に置くことができる．

2.3.3 段階 2: 強分類

強分類は3つのトロイ要素を各最大スコアネットに設定する．

トロイ要素 1: 最大スコア

トロイ要素 2: 一定値を出力する最大のクロックサイクル数

トロイ要素 3: 最大スコアネット数

トロイ要素は前提 1 と前提 2 を前提に置いて計算できることは明確である．

2.3.4 段階 3: トロイ識別

トロイ識別は最終的に与えられたネットリストにハードウェアトロイが挿入されているか否かを識別する．トロイ要素の閾値は以下である．

トロイ要素 1 の閾値: 3
トロイ要素 2 の閾値: 999,996 cycles
トロイ要素 3 の閾値: 5

強トロイネットが検出された場合，トロイ識別はネットリストにハードウェアトロイが挿入されていると識別する．さもなければ，ハードウェアトロイが挿入されていないと識別する．強トロイネット数で識別するだけであるため，提案手法が問題 1 と問題 2 を前提 1 と前提 2 を前提に置いて解決できることがわかる．

2.4 実験結果

Trust-HUB [9] の Abstraction Gate Level で公開されている全てのベンチマークに対して提案手法を適用した．表 3.1 はハードウェアトロイの挿入されているネットリストを示す．表 3.2 はハードウェアトロイの挿入されていないネットリストを示す．Trust-HUB はハードウェアトロイの挿入されていない RS232 を公開していないため，RS232-T1000 から注意深くハードウェアトロイを取り除いて作成した．

2.4.1 弱分類の結果

表 3.1 の全てのベンチマークに対して，ケースネットに一致するネット数を測定した．検出するプログラムは C 言語で記述し，Core i7 を搭載した PC を使用して各ベンチマークに対して高々数十分の時間で実験を終えた．2–10 行目は弱トロイネットのケース 1–9 に一致したネット数を示す．

表 2.7: ハードウェアトロイの挿入されているネットリスト.

ベンチマーク	ネット数
b19-T100	95,502
b19-T200	95,502
EthernetMAC10GE-T700	103,220
EthernetMAC10GE-T710	103,220
EthernetMAC10GE-T720	103,220
EthernetMAC10GE-T730	103,220
RS232-T1000	311
RS232-T1100	312
RS232-T1200	315
RS232-T1300	307
RS232-T1400	311
RS232-T1500	314
RS232-T1600	307
s15850-T100	2,456
s35932-T100	6,438
s35932-T200	6,435
s35932-T300	6,460
s38417-T100	5,819
s38417-T200	5,822
s38417-T300	5,851
s38584-T100	7,399
s38584-T200	7,580
s38584-T300	9,110
vga_lcd-T100	70,162
wb_conmax-T100	22,197

表 2.8: ハードウェアトロイの挿入されていないネットリスト.

ベンチマーク	ネット数
b19	108,332
EthernetMAC10GE	103,206
RS232	298
s15850	2,429
s35932	6,423
s38417	5,807
s38584	7,380
vga_lcd	70,157
wb_conmax	22,182

表 2.9: ケースネット数と最大スコアと最大スコアネット数.

ベンチマーク	ケース 1	ケース 2	ケース 3	ケース 4	ケース 5	ケース 6	ケース 7	ケース 8	ケース 9	最大スコアネット数	最大スコア
b19-T100	822	43	84	16701	131	0	0	0	0	1	3
b19-T200	822	43	84	16701	131	0	0	0	0	1	3
EthernetMAC10GE-T700	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T710	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T720	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T730	347	56	2595	417	193	3	3	0	0	1	6
RS232-T1000	11	2	2	0	12	0	0	0	0	2	2
RS232-T1100	11	2	2	0	12	0	0	0	0	2	2
RS232-T1200	10	2	2	0	12	0	0	0	0	2	2
RS232-T1300	9	2	2	0	12	0	0	0	0	2	2
RS232-T1400	12	3	2	0	12	0	0	0	0	3	2
RS232-T1500	11	2	2	0	12	0	0	0	0	2	2
RS232-T1600	9	2	2	0	12	0	0	0	0	2	2
s15850-T100	80	11	1	0	160	2	4	0	3	1	6
s35932-T100	3	3	1	0	420	2	2	0	4	1	8
s35932-T200	3	3	0	0	420	0	0	0	4	3	4
s35932-T300	3	3	1	0	420	0	0	0	4	1	5
s38417-T100	206	4	0	0	206	0	0	0	0	5	2
s38417-T200	203	3	0	0	206	0	0	0	0	4	2
s38417-T300	206	4	0	0	206	4	4	0	0	1	6
s38584-T100	112	3	2	0	404	0	0	0	4	2	3
s38584-T200	112	4	4	87	405	0	0	0	0	1	3
s38584-T300	123	16	3	783	405	0	0	0	0	1	3
vga_lcd-T100	19	0	1954	168	209	0	0	0	0	2	2
wb_conmax-T100	466	55	0	0	1516	0	0	1	0	1	4
b19	598	38	1108	16586	130	0	0	0	0	77	2
EthernetMAC10GE	344	55	2595	417	193	0	0	0	0	55	2
RS232	5	0	2	0	12	0	0	0	0	19	1
s15850	68	5	0	0	160	0	0	0	0	5	2
s35932	0	0	0	0	420	0	0	0	0	420	1
s38417	199	1	0	0	206	0	0	0	0	2	2
s38584	110	3	2	0	404	0	0	0	0	9	2
vga_lcd	16	0	1954	161	209	0	0	0	0	2340	1
wb_conmax	455	52	0	0	1516	0	0	0	0	48	2

2.4.2 強分類の結果

強分類は3つのトロイ要素を各最大スコアネットに設定する．一定値を出力する最大のクロックサイクル数を計算するために論理シミュレータとしてVCSを使用した．Xeon E7-4870を使用し各ベンチマークに対して高々2時間ほどで，1メガクロックサイクルシミュレーションを終えた．

2.4.3 トロイ識別の結果

3つのトロイ要素に閾値を設定することで，提案手法は強トロイネットを検出する．

表2.10にトロイ識別の結果を示す．提案手法は全てのハードウェアトロイが挿入されているネットリスト(b19-T100–wb_conmax-T100)に対して，最低でも1つは強トロイネットを検出している．したがって，これらのネットリストはハードウェアトロイが挿入されていると識別する．更に，提案手法はハードウェアトロイの挿入されていないネットリスト(b19–wb_conmax)に対して，強トロイネットを1つも検出していない．したがって，これらのネットリストはハードウェアトロイが挿入されていないと識別する．

表2.10の“強トロイネットの中身”は強トロイネットがトロイネットなのかノーマルネットなのかを示す．全ての強トロイネットはトロイネットのみを含むため，提案手法は誤検出無しで正しく識別をできていることがわかる．

以上より，提案手法は与えられたネットリストにハードウェアトロイが挿入されているか否かを3時間以内で識別することに成功した．

2.4.4 提案手法と既存手法の性能比較

提案手法と既存手法の性能を比較した．[51]の比較方法にならい，ハードウェアトロイと疑われる信号やネットを列挙し，その中にハードウェアトロイの信号やネットが含まれていた場合，検出に成功したと評価する．この比較方法にて，表3.1の全てのベンチマークに対して，提案手法と既存手法がどれだけハードウェアトロイを検出できたかを比較する．

表2.11に提案手法と既存手法の性能を示す．表2.11のチェックマークは，各ベンチマークに対してハードウェアトロイが検出できたか否かを示す．UCIとVeriTrustの検出結果は[51]より引用した．UCIが検出できたハードウェアトロイは4個，VeriTrustが検出できたハードウェアトロイは8個検出できている．しかし，UCIとVeriTrustはハードウェアトロイの情報を事前に知っている必要がある．一方，提案手法はハードウェアトロイの挿入されている全てのベンチマークに対してハードウェアトロイを検出することに成功した．加えて，提案手法は誤検出がなくゴールデンネットリストもハードウェアトロイに関する一切の情報を必要としない．以上より，提案手法はゲートレベルネットリストに対するハードウェアトロイ検出手法の中で最も現実的な使用に堪えうる手法であるといえる．

2.5 本章のまとめ

本章では，ゲートレベルのネットリストに対してスコアに基づいたハードウェアトロイの有無を識別する手法である．提案手法は，ゴールデンネットリストもハードウェアトロイに関する一切の情報も必要とせずに，ネットリストにハードウェアトロイが挿入されているか否かを識別できる．提案手法は，Trust-HUB で公開されている Abstraction Gate Level のベンチマーク全てに対して，ハードウェアトロイが挿入されているか否かを識別することに成功した．

表 2.10: トロイ識別の結果.

ベンチマーク	強トロイネット数	強トロイネットの中身	判定
b19-T100	1	トロイネットのみ	トロイあり
b19-T200	1	トロイネットのみ	トロイあり
EthernetMAC10GE-T700	1	トロイネットのみ	トロイあり
EthernetMAC10GE-T710	1	トロイネットのみ	トロイあり
EthernetMAC10GE-T720	1	トロイネットのみ	トロイあり
EthernetMAC10GE-T730	1	トロイネットのみ	トロイあり
RS232-T1000	2	トロイネットのみ	トロイあり
RS232-T1100	2	トロイネットのみ	トロイあり
RS232-T1200	1	トロイネットのみ	トロイあり
RS232-T1300	1	トロイネットのみ	トロイあり
RS232-T1400	2	トロイネットのみ	トロイあり
RS232-T1500	2	トロイネットのみ	トロイあり
RS232-T1600	1	トロイネットのみ	トロイあり
s15850-T100	1	トロイネットのみ	トロイあり
s35932-T100	1	トロイネットのみ	トロイあり
s35932-T200	3	トロイネットのみ	トロイあり
s35932-T300	1	トロイネットのみ	トロイあり
s38417-T100	3	トロイネットのみ	トロイあり
s38417-T200	2	トロイネットのみ	トロイあり
s38417-T300	1	トロイネットのみ	トロイあり
s38584-T100	2	トロイネットのみ	トロイあり
s38584-T200	1	トロイネットのみ	トロイあり
s38584-T300	1	トロイネットのみ	トロイあり
vga_lcd-T100	2	トロイネットのみ	トロイあり
wb_conmax-T100	1	トロイネットのみ	トロイあり
b19	0	N/A	トロイなし
EthernetMAC10GE	0	N/A	トロイなし
RS232	0	N/A	トロイなし
s15850	0	N/A	トロイなし
s35932	0	N/A	トロイなし
s38417	0	N/A	トロイなし
s38584	0	N/A	トロイなし
vga_lcd	0	N/A	トロイなし
wb_conmax	0	N/A	トロイなし

表 2.11: 提案手法と既存手法の性能.

ベンチマーク	UCI [19]	VeriTrust [51]	提案手法
b19-T100			✓
b19-T200			✓
EthernetMAC10GE-T700			✓
EthernetMAC10GE-T710			✓
EthernetMAC10GE-T720			✓
EthernetMAC10GE-T730			✓
RS232-T1000			✓
RS232-T1100			✓
RS232-T1200			✓
RS232-T1300			✓
RS232-T1400			✓
RS232-T1500			✓
RS232-T1600			✓
s15850-T100	✓	✓	✓
s35932-T100		✓	✓
s35932-T200	✓	✓	✓
s35932-T300		✓	✓
s38417-T100	✓	✓	✓
s38417-T200		✓	✓
s38417-T300	✓	✓	✓
s38584-T100			✓
s38584-T200		✓	✓
s38584-T300			✓
vga_lcd-T100			✓
wb_conmax-T100			✓

第3章　ゲートレベルネットリストの危険性を表現する指標

3.1　本章の概要

本章では，トロイネットが持つ回路パターンの詳細な特徴を明らかにし，トロイネットを検出するためのスコアリングアルゴリズムを示し，ゲートレベルネットリストの危険視を表現する指標を提案する[1]．以下に本章の構成を示す．

第 3.2 節「トロイネットの特徴に基づいた HT rank」では，ゲートレベルネットリストの危険視を表現するための HT rank という指標を提案する．トロイネットの持つ特徴をキャラクタリスティックポイント・スケールポイント・ロケーションポイントの 3 つに分類する．これらのポイントとトロイネットを検出するための重みづけを与えるスコアを組み合わせることで，ゲートレベルネットリストの危険視を表現する．

第 3.3 節「実験結果」では，提案手法のソフトウェア実験の結果を示し，提案手法の有効性を評価する．

第 3.4 節「本章のまとめ」では，本章の内容をまとめる．

3.2　トロイネットの特徴に基づいた HT rank

信頼性の低いゲートレベルネットリストにハードウェアトロイが含まれているかどうかをハードウェアトロイを識別する指標を用いて識別することを考える．理想的な指標以下の三つの条件を満足する必要がある．

(C1) 理想的な指標はハードウェアトロイが挿入されているネットリストに対して，高い値を示さなければならない．

(C2) 理想的な指標はハードウェアトロイが挿入されていないネットリストに対して，低い値を示さなければならない．

(C3) 理想的な指標はランダムテスト，ロジックシミュレーション，ゴールデンネットリストに依存してはならない．

このような理想的なハードウェアトロイを識別する指標を開発することができた時，ゲートレベルネットリストの安全性を定量的に評価することができるようになる．

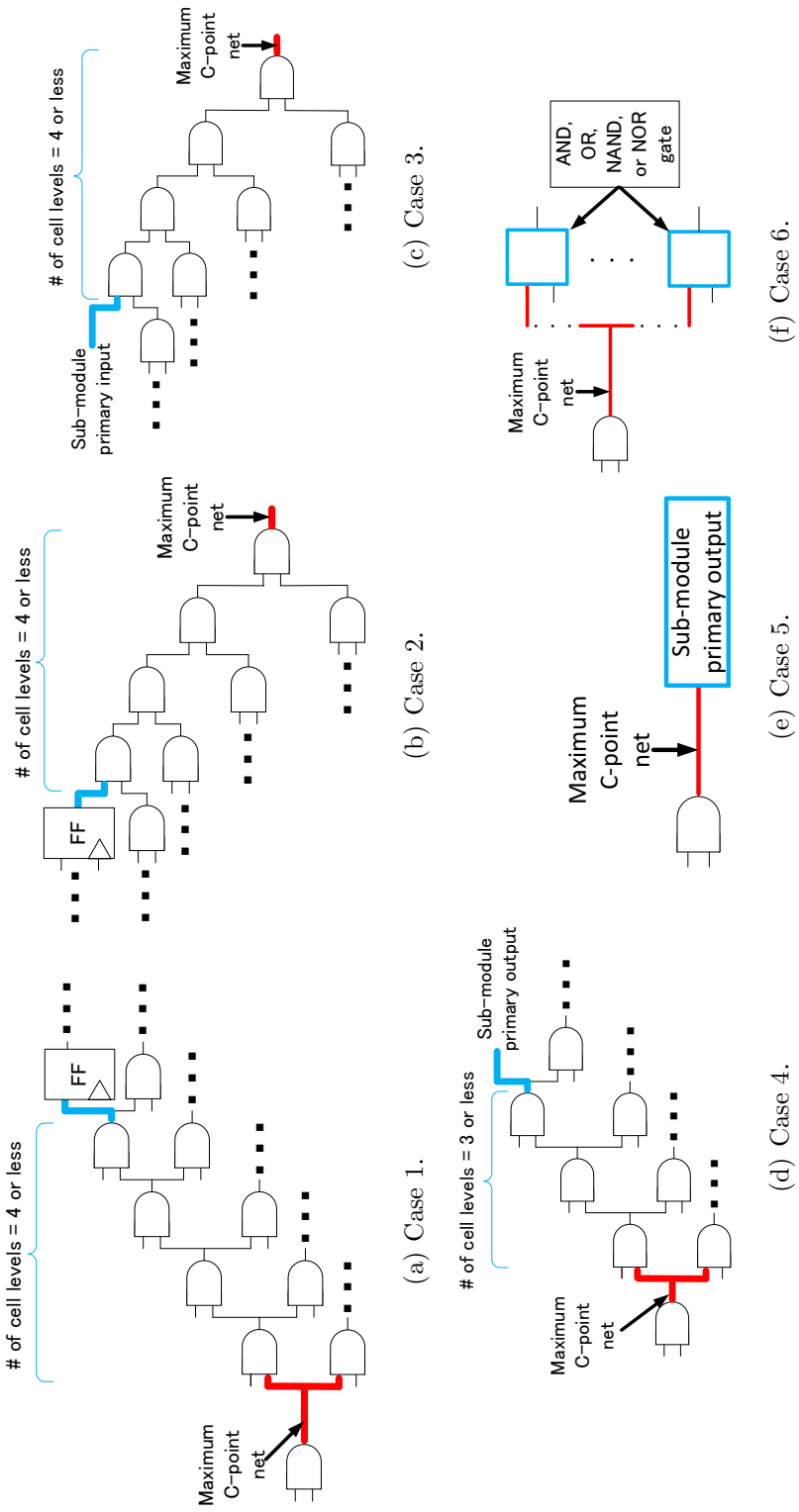
[1]本章の内容は，文献 [28,53] による．

3.2.1 HT rank の概要

理想的なハードウェアトロイを識別する指標を開発するために，トレーニングデータからトロイネットらしいネットを識別することを試みる．今回使用したトレーニングデータは Trust-HUB [9] で手に入るゲートレベルネットリストであり，これらのネットリストの情報を表 3.1 と表 3.2 に示す．各ベンチマークは 1 個もしくは複数のサブモジュールで構成されており，さらにゲート，フリップフロップ，アダーのようなセルで構成されている．

全ての Trust-HUB のゲートレベルネットリストのハードウェアトロイを検査し，トロイネットの持つ特別な特徴を発見した．トロイネットのいくつかの特徴を Trust-HUB のゲートレベルネットリストから抽出し，それらに対してどの程度トロイネットであるかということを意味するトロイポイントを与える．適切にトロイポイントを与えることで，Trust-HUB のネットリストに対してハードウェアトロイの有無を分類することができる．

後に議論するように，ネットリストに最大トロイポイント（HT rank と呼ぶ）を与えハードウェアトロイを識別する指標にすることで，悪意ある回路の位置を特定する強力なヒントを提供する．



(a) Case 1. (b) Case 2. (c) Case 3. (d) Case 4. (e) Case 5. (f) Case 6.

図 3.1: ロケーションポイント.

3.2.2 キャラクタリスティックポイント (C-ポイント)

最初に，[30] のハードウェアトロイ検出手法は世界初であり，近年提案された全ての Trust-HUB のゲートレベルネットリストのハードウェアトロイを検出する．第一段階目では, 与えられた信頼性の低いネットリスト中の全てのネットにスコアを与える．スコアはに関しては, もしネット n が 9 個の特徴のいずれかに一致したら，1 もしくは 2 ポイント n に与える．スコアの計算にロジックシミュレーションは必要ではない．各ネットのスコアはトロイネットとノーマルネットを完全に区別することはできない．したがって，次の段階でシミュレーションを使用してハードウェアトロイだけを検出している．

このアプローチ [30] は信頼性の低いネットリストに対して有効なハードウェアトロイ検出手法だが，致命的な欠陥がある．

1. ロジックシミュレーションが必要なため，ハードウェアトロイの結果が入力するテストパターンに依存する．
2. 与えられた信頼性の低いネットリストにハードウェアトロイが入っているかどうかしかわからない．どの程度安全でなのかという尺度を与えてはくれない．

この手法は上記の深刻な欠点はあるが，第一段階目で与えられたスコアはトロイネットと疑われるネットを識別するためにとても有用である．したがって，ネットリストの各ネットにこれらのスコアが与えられた状態を考える．提案する HT rank は最初の [30] で提案されたネットの特徴に基づいて計算する方法を採用し，与えられた信頼性の低いネットリストの各ネットのポイントを計算する．

このスコアをキャラクタリスティックポイントもしくは C-ポイントと呼ぶ．最大の C-ポイントを持つネットを最大 C-ポイントネットと呼ぶ．最大 C-ポイントネットは一度だけ全てのネットをスキャンするだけで得ることができる．

信頼性の低いネットリスト中にある最大 C-ポイントネットは最もトロイネットと疑われるもしくはトロイネットらしいネットであるため，最大 C-ポイントネットに注目していく．

表 3.1: Trust-HUB のハードウェアトロイ有りゲートレベルネットリスト.

ベンチマーク	ネット数
b19-T100	95,502
b19-T200	95,502
EthernetMAC10GE-T700	103,220
EthernetMAC10GE-T710	103,220
EthernetMAC10GE-T720	103,220
EthernetMAC10GE-T730	103,220
RS232-T1000	311
RS232-T1100	312
RS232-T1200	315
RS232-T1300	307
RS232-T1400	311
RS232-T1500	314
RS232-T1600	307
s15850-T100	2,456
s35932-T100	6,438
s35932-T200	6,435
s35932-T300	6,460
s38417-T100	5,819
s38417-T200	5,822
s38417-T300	5,851
s38584-T100	7,399
s38584-T200	7,580
s38584-T300	9,110
vga_lcd-T100	70,162
wb_conmax-T100	22,197

表 3.2: Trust-HUB のハードウェアトロイ無しゲートレベルネットリスト.

ベンチマーク	ネット数
b19	108,332
EthernetMAC10GE	103,206
RS232	298
s15850	2,429
s35932	6,423
s38417	5,807
s38584	7,380
vga_lcd	70,157
wb_conmax	22,182

表 3.3: Trust-HUB ベンチマークの最大 C-ポイントと最大 C-ポイントネット数.

ベンチマーク	最大 C-ポイント	最大 C-ポイントネット数
b19-T100	3	1
b19-T200	3	1
EthernetMAC10GE-T700	6	1
EthernetMAC10GE-T710	6	1
EthernetMAC10GE-T720	6	1
EthernetMAC10GE-T730	6	1
RS232-T1000	2	2
RS232-T1100	2	2
RS232-T1200	2	2
RS232-T1300	2	2
RS232-T1400	2	3
RS232-T1500	2	2
RS232-T1600	2	2
s15850-T100	6	1
s35932-T100	8	1
s35932-T200	4	3
s35932-T300	5	1
s38417-T100	2	5
s38417-T200	2	4
s38417-T300	6	1
s38584-T100	3	2
s38584-T200	3	1
s38584-T300	3	1
vga_lcd-T100	2	2
wb_conmax-T100	4	1
b19	2	77
EthernetMAC10GE	2	55
RS232	1	19
s15850	2	5
s35932	1	420
s38417	2	2
s38584	2	9
vga_lcd	1	2340
wb_conmax	2	48

表 3.4: トロイポイントの詳細.

ベンチマーク	HT-inserted/free	最大 C-ポイント	最大 C-ポイント数	S-ポイント	ネット名	ロケーション ケース 1	ロケーション ケース 2	ロケーション ケース 3	ロケーション ケース 4	ロケーション ケース 5	ロケーション ケース 6	L-ポイント	トロイ ポイント	種類
b19-T100	HT-inserted	3	1	3	Trigger_out	0	1	0	1	2	0	4	**10**	Trojan net
b19-T200	HT-inserted	3	1	3	Trigger_out	0	1	0	1	2	0	4	**10**	Trojan net
EthernetMAC10GE-T700	HT-inserted	6	1	3	Tj_OUTClock	1	1	0	0	0	0	2	**11**	Trojan net
EthernetMAC10GE-T710	HT-inserted	6	1	3	Tj_OUTClock	1	1	0	0	0	0	2	**11**	Trojan net
EthernetMAC10GE-T720	HT-inserted	6	1	3	Tj_OUTClock	1	0	0	0	0	0	1	**10**	Trojan net
EthernetMAC10GE-T730	HT-inserted	6	1	3	Tj_OUTClock	1	1	0	0	0	0	2	**11**	Trojan net
RS232-T1000	HT-inserted	2	2	3	iCTRL	0	1	0	1	0	3	5	10	
					iRECEIVER_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1100	HT-inserted	2	2	3	iCTRL	0	1	0	1	0	3	5	10	
					iRECEIVER_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1200	HT-inserted	2	2	3	iCTRL	0	1	0	1	0	3	5	10	
					iXMIT_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1300	HT-inserted	2	2	3	iCTRL	0	1	1	1	0	3	6	**11**	Trojan net
					iXMIT_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1400	HT-inserted	2	3	3	iCTRL	0	1	0	1	0	3	5	10	
					iRECEIVER_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
					iXMIT_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1500	HT-inserted	2	2	3	iCTRL	0	1	0	1	0	3	5	10	
					iRECEIVER_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
RS232-T1600	HT-inserted	2	2	3	iCTRL	0	1	0	1	0	3	5	10	
					iXMIT_CTRL	0	1	1	1	0	3	6	**11**	Trojan net
s15850-T100	HT-inserted	6	1	3	Tg1_Trigger1	1	1	0	0	0	0	2	**11**	Trojan net
s35932-T100	HT-inserted	8	1	3	Tj_Trigger	1	1	1	1	0	0	4	**15**	Trojan net
s35932-T200	HT-inserted	4	3	3	Tj_OUT1234	1	1	0	1	0	3	6	13	
					Tj_OUT5678	1	1	0	1	0	3	6	13	
					Tj_Trigger	1	1	1	1	0	3	7	**14**	Trojan net
s35932-T300	HT-inserted	5	1	3	Tj_Trigger	0	1	1	0	0	0	2	**10**	Trojan net
s38417-T100	HT-inserted	2	5	3	g25489	0	1	0	1	2	0	4	9	
					n2401	1	1	0	0	0	0	2	7	
					Tj_OUT1234	1	1	0	0	0	3	5	**10**	Trojan net
					Tj_OUT5678	1	1	0	0	0	3	5	**10**	Trojan net
					Tj_Trigger	1	1	0	0	0	3	5	**10**	Trojan net
s38417-T200	HT-inserted	2	4	3	g25489	0	1	0	1	2	0	4	9	
					n2401	1	1	0	0	0	0	2	7	
					Tj_OUT1234	1	1	0	0	0	3	5	**10**	Trojan net
					Tj_Trigger	1	0	0	0	0	3	4	9	
s38417-T300	HT-inserted	6	1	3	Tj_Trigger	1	1	0	0	0	0	2	**11**	Trojan net
s38584-T100	HT-inserted	3	2	3	Tj_OUT1234	1	1	0	0	0	3	5	11	
					Tj_Trigger	1	1	1	0	0	3	6	**12**	Trojan net
s38584-T200	HT-inserted	3	1	3	Trigger_out	0	1	0	1	2	0	4	**10**	Trojan net
s38584-T300	HT-inserted	3	1	3	Trigger_out	0	1	0	1	2	0	4	**10**	Trojan net
vga_lcd-T100	HT-inserted	2	2	3	Tj_OUT1	1	1	0	0	0	3	5	10	
					Tj_Trigger	1	1	0	1	0	3	6	**11**	Trojan net
wb_conmax-T100	HT-inserted	4	1	3	Tj_Trigger	1	0	0	0	0	3	4	**11**	Trojan net
s15850	HT-free	2	5	3	n1132	1	1	0	0	0	0	2	7	
					n1440	0	1	0	0	0	3	4	**9**	Normal net
					n1491	1	0	1	0	0	0	2	7	
					n1509	1	0	0	0	0	0	1	6	
					n1546	1	0	0	0	0	3	4	**9**	Normal net
s38417	HT-free	2	2	3	g25489	0	1	0	1	2	0	4	**9**	Normal net
					n2401	1	1	0	0	0	0	2	7	

表 3.5: スケールポイントとロケーションポイントの内訳.

スケールポイントとロケーションポイントの内訳	ポイント
スケールポイント	3
ロケーションケース 1	1
ロケーションケース 2	1
ロケーションケース 3	1
ロケーションケース 4	1
ロケーションケース 5	2
ロケーションケース 6	3

3.2.3 スケールポイント (S-ポイント)

各 Trust-HUB ベンチマークの C-ポイントを表 3.3 に示す．表 3.3 が示すように最大 C-ポイントではトロイネットとノーマルネットを完全に区別することができないが，最大 C-ポイントはトロイネットを識別する強力なヒントになる．

表 3.1 と表 3.3 によると，全てのハードウェアトロイの挿入されているネットリストはネット数が 300 以上であるが，それらの最大 C-ポイントネットは比較的小さい．これはハードウェアトロイを識別する指標を開発する際の強力なヒントになる．表 3.1 と表 3.3 により，全てのハードウェアトロイの挿入されているネットリストの最大 C-ポイントネット数は 5 以下であることがわかる．したがって，これらのハードウェアトロイの挿入されているネットリストに対してポイントを追加する．

表 3.3 を得た時，[30] が示す 9 つの特徴に一致したら 1 もしくは 2 ポイントをネットに与えた．そこで，ネットリスとの総ネット数が 300 以上かつ最大 C-ポイントネットが 5 以下の場合に最大 C-ポイントネットに 3 ポイント追加する．この追加ポイントはスケールポイントもしくは S-ポイントと呼ぶ．新しいポイントを S-ポイントとして追加することで，よりトロイネットらしいネットのポイントを増やすことができる．

しかし，ハードウェアトロイの挿入されていないベンチマークである s15850 と s38417 も S-ポイントに該当する．S-ポイントを上のベンチマークに追加するのは望ましくない．したがって，ハードウェアトロイの挿入されていないネットリストを挿入されているネットリストと区別するためにさらなる指標を開発する．

3.2.4 ロケーションポイント (L-ポイント)

最大 C-ポイントネット数が 5 以下の Trust-HUB ベンチマークに着目し，より強力なトロイネットの特徴を発見するために最大 C-ポイントネットからトロイネットの特徴を抽出する．

悪意ある攻撃者は初めにハードウェアトロイを動作させるために，トロイネットが高い制御性を持つように設計する．トロイネットは制御性を得るためにフリップフロップやプライ

マリインプットの近くに挿入される（ロケーションケース 1–3 に該当する）．ハードウェアトロイは秘密情報をサブモジュールのプライマリアウトプットから漏洩する可能性があるため，トロイネットはプライマリアウトプットの近くに挿入される（ロケーションケース 4 に該当する）．特に，トロイネットが直接サブモジュールのプライマリアウトプットに接続されている可能性がある（ロケーションケース 5 に該当する）．更に，Trust-HUB ベンチマークを注意深く解析した結果，いくつかのトロイネットは複数のファンアウトを持ち，それらの全てが AND，OR，NAND，NOR ゲートに接続されていることが分かった（ロケーションケース 6 に該当する）．

ロケーションケースを以下に要約する．

ロケーションケース 1 (図 3.1(a)): 最大 C-ポイントネットがフリップフロップのインプットの近くに位置するならば，L-ポイントを追加する．

ロケーションケース 2 (図 3.1(b)): 最大 C-ポイントネットがフリップフロップのアウトプットの近くに位置するならば，L-ポイントを追加する．

ロケーションケース 3 (図 3.1(c)): 最大 C-ポイントネットがサブモジュールのプライマリインプットの近くに位置するならば，L-ポイントを追加する．

ロケーションケース 4 (図 3.1(d)): 最大 C-ポイントネットがサブモジュールのプライマリアウトプットの近くに位置するならば，L-ポイントを追加する．

ロケーションケース 5 (図 3.1(e)): 最大 C-ポイントネットがサブモジュールのプライマリアウトプットのみに接続しているならば，L-ポイントを追加する．

ロケーションケース 6 (図 3.1(f)): 最大 C-ポイントネットが複数のファンアウトを持ち，その全てが AND，OR，NAND，NOR ゲートのインプットに接続されているならば，L-ポイントを追加する．

ロケーションケース 1–4 において，トロイネットがどの程度フリップフロップもしくはプライマリインプット・アウトプットに近いのかを解析した．その結果，トロイネットはフリップフロップのインプットもしくはアウトプットから，4 段以内にあることが分かった.. プライマリインプットも同様に 4 段以内であり，プライマリアウトプットは 3 段以内である（図 3.1(a)–(d) 参照）．

最大 C-ポイントネット n がロケーションケース 1–ロケーションケース 4（図 3.1 参照）のいずれかに一致した時，n に 1 ポイントを追加する．n がロケーションケース 5 に一致した時は特別なケースとして，n に 2 ポイントを追加する．n がロケーションケース 6 に一致した時は更に特別なケースとして，n に 3 ポイントを追加する．この追加ポイントをロケーションポイントもしくは L-ポイントと呼ぶ．

最後に，各最大 C-ポイントネットの C-ポイント，S-ポイント，L-ポイントの総和を計算し，トロイポイントを算出する．最大 C-ポイントネットが持つ最も高いトロイポイントを最

大トロイポイントと呼び，最大トロイポイントを持つネットを最大トロイポイントネットと呼ぶ．

表 3.4 に L-ポイントとトロイポイントを示す．上記の様に適切に L-ポイントを設定することで，全てのハードウェアトロイの挿入されているネットリストは少なくとも 1 個は最大トロイポイントが 10 以上のネットを含む（太字の数字が “Trojan point”）．全ての最大トロイポイントネットは実際にトロイネットである．二つのハードウェアトロイの挿入されていないネットリスト（s15850 と s38417）は共に最大トロイポイントが 9 以下である．これは最大トロイポイントによって，Trust-HUB におけるハードウェアトロイの挿入されているネットリストと挿入されていないネットリストを完全に区別できることを示す（表 3.7 参照)．

最大トロイポイントが 10 がハードウェアトロイの挿入されているネットリストと挿入されていないネットリストを区別する極めて有効な閾値であると結論付ける．

3.2.5 Hardware-Trojans rank

HT rank は各ベンチマークの最大トロイポイントと定義する．最大トロイポイントは，C-ポイント，S-ポイント，L-ポイントの総和で得られる．表 3.5 はトロイポイントを計算するための S-ポイントと L-ポイントを示す．

要約すると，HT rank は以下で計算される．

1. **C-ポイント:** 与えられたネットリストの各ネットの C-ポイントを計算する [30]．もし最大 C-ポイントが 0 ならば，処理を終了して HT rank を 0 とする．
2. **S-ポイント:** 最大 C-ポイントネットに対して，もしネットリストの総ネット数が 300 以上かつ最大 C-ポイントネット数が 5 以下ならば，S-ポイントを追加する．
3. **L-ポイント:** 各最大 C-ポイントネットに対して，L-ポイントを追加する．
4. **Maximum Trojan Point:** ネットリスト中のトロイポイントを計算し，最大トロイポイントが HT rank として与えられる．

3.3 実験結果

3.3.1 FANCI vs. HT rank

FANCI [44] と提案手法の結果を比較する．

HT rank が 10 以上のネットリストはハードウェアトロイが挿入されている可能性が高いことを既に示した．表 3.6 は全てのハードウェアトロイの挿入されているゲートレベル Trust-HUB ベンチマークにおける FANCI と HT rank の結果を示す．HT rank における “HT detected” は HT rank が 10 以上の場合に記述している．

HT rank 全てのハードウェアトロイの挿入されているゲートレベル Trust-HUB ベンチマークの検出に False positive 無しで検出することに成功している（図 3.7 参照）．FANCI の結

表 3.6: FANCI vs 提案手法.

HT-inserted ベンチマーク (Trust-HUB)	FANCI [44]	提案手法	
		HT rank	検出結果
b19-T100		10	HT detected
b19-T200		10	HT detected
EthernetMAC10GE-T700		11	HT detected
EthernetMAC10GE-T710		11	HT detected
EthernetMAC10GE-T720		10	HT detected
EthernetMAC10GE-T730		11	HT detected
RS232-T1000	HT detected	11	HT detected
RS232-T1100	HT detected	11	HT detected
RS232-T1200	HT detected	11	HT detected
RS232-T1300	HT detected	11	HT detected
RS232-T1400	HT detected	11	HT detected
RS232-T1500	HT detected	11	HT detected
RS232-T1600	HT detected	11	HT detected
s15850-T100	HT detected	11	HT detected
s35932-T100	HT detected	15	HT detected
s35932-T200	HT detected	14	HT detected
s35932-T300	HT detected	10	HT detected
s38417-T100	HT detected	10	HT detected
s38417-T200	HT detected	10	HT detected
s38417-T300	HT detected	11	HT detected
s38584-T100		12	HT detected
s38584-T200		10	HT detected
s38584-T300		10	HT detected
vga_lcd-T100		11	HT detected
wb_conmax-T100		11	HT detected

果は [44] から引用した．FANCI はいくつかのベンチマークに対してハードウェアトロイが検出できない.

3.3.2 Trust-HUB ベンチマークへの HT rank 適用結果

HT rank を Trust-HUB [9] で提供されている全てのゲートレベルネットリストに対して適用した．Trust-HUB はハードウェアトロイの挿入されていない RS232 を提供していないので，RS232-T1000 からハードウェアトロイを除去して “RS232” を作成した．Trust-HUB ベンチマークは論理合成後のゲートレベルネットリストとして提供されている．表 3.7 の下線は Trust-HUB ベンチマークを示す．HT rank は Xeon E7-4870 で計算し，計算時間はネット

リストの大きさにより数分から一日である．

HT rank は Trust-HUB ベンチマークに対して，ハードウェアトロイが挿入されているベンチマークは HT rank が 10 以上，挿入されていないベンチマークは HT rank が 9 以下 と分類することに成功している．

3.3.3 Trust-HUB，ISCAS85，ISCAS89，ITC99，OpenCores，ハードウェアトロイ有り/無し AES への HT rank 適用結果

HT rank の性能を確認するため，より多くのベンチマークに対して適用する．ハードウェアトロイの挿入されているベンチマークに対して高い HT rank，挿入されていないベンチマークに対して低い HT rank が与えられれば成功である．ISCAS85，ISCAS89 ベンチマークは [1]，ITC99 ベンチマークは [8] で提供されている．これらのベンチマークは [1,8] で論理合成後のゲートレベルネットリストとして提供されている．ハードウェアトロイの挿入されている AES と挿入されていない AES は (AES-T1，AES-T2，AES) は Synopsys 社の Design Compiler で TSMC 90nm ライブラリを使用して論理合成を行い，ゲートレベルネットリストを得た．全ての ISCAS85，ISCAS89，ITC99 ベンチマークはハードウェアトロイが挿入されていない．AES はハードウェアトロイの挿入されていない 128 ビットの AES ゲートレベルネットリストである．AES-T1 と AES-T2 はハードウェアトロイの挿入されているネットリストである．このネットリストに挿入されているハードウェアトロイは，特定の平文が入力されたら動作するようになっている．

表 3.7 は各ベンチマークにおける HT rank を示す．下線の引かれていないネットリストは ISCAS85，ISCAS89，ITC99，AES のベンチマークである．HT rank はこれらのベンチマークに対しても，ハードウェアトロイが挿入されているベンチマークは HT rank が 10 以上，挿入されていないベンチマークは HT rank が 9 以下 と分類することに成功している．HT rank は Xeon E7-4870 で計算し，計算時間はネットリストの大きさにより数分から一日である．

表 3.7 の “ネット数” は各ベンチマークのネット数を示す．“最大トロイポイントネット数” は各ベンチマークの最大トロイネット数を示す．ほとんどのベンチマークは最大トロイネット数が 10 未満である．特にハードウェアトロイの挿入されているベンチマークは 3 以下である．以上より，HT rank はトロイネットらしきネットを検出するのに有効である．

表 3.7 の “最大トロイポイントネット数 [%]” は各ベンチマークにおける最大トロイネットが占める割合を示す．平均値は 0.774%未満であり，HT rank がトロイネットらしきネットの候補を大幅に削減し，設計者がこれらのネットを検査するコストを削減している．“誤検出ネット数 [%]” はハードウェアトロイの挿入されている各ベンチマークにおける最大トロイネットにノーマルネットが含まれる割合を示す．HT rank は誤検出がなくトロイネットのみを検出している．

以上より，ネットリストは HT rank が 10 以上の場合はハードウェアトロイが挿入されている可能性が高く，HT rank が 9 以下の場合はハードウェアトロイが挿入されていない可能

性が高い.

表 3.7: 各ベンチマークにおける HT rank.

ベンチマーク	ネット数	最大トロイポイントネット数	最大トロイポイントネット数 [%]	誤検出ネット数 [%]	HT-inserted /free	HT rank
s35932-T100	6438	1	0.016	0%	HT-inserted	15
s35932-T200	6435	1	0.016	0%	HT-inserted	14
AES-T1	14157	1	0.007	0%	HT-inserted	12
s38584-T100	7399	1	0.014	0%	HT-inserted	12
AES-T2	14029	1	0.007	0%	HT-inserted	11
EthernetMAC10GE-T700	103220	1	0.001	0%	HT-inserted	11
EthernetMAC10GE-T710	103220	1	0.001	0%	HT-inserted	11
EthernetMAC10GE-T730	103220	1	0.001	0%	HT-inserted	11
RS232-T1000	311	1	0.322	0%	HT-inserted	11
RS232-T1100	312	1	0.321	0%	HT-inserted	11
RS232-T1200	315	1	0.317	0%	HT-inserted	11
RS232-T1300	307	2	0.651	0%	HT-inserted	11
RS232-T1400	311	2	0.643	0%	HT-inserted	11
RS232-T1500	314	1	0.318	0%	HT-inserted	11
RS232-T1600	307	1	0.326	0%	HT-inserted	11
s15850-T100	2456	1	0.041	0%	HT-inserted	11
s38417-T300	5851	1	0.017	0%	HT-inserted	11
vga_lcd-T100	70162	1	0.001	0%	HT-inserted	11
wb_conmax-T100	22197	1	0.005	0%	HT-inserted	11
b19-T100	95502	1	0.001	0%	HT-inserted	10
b19-T200	95502	1	0.001	0%	HT-inserted	10
EthernetMAC10GE-T720	103220	1	0.001	0%	HT-inserted	10
s35932-T300	6460	1	0.015	0%	HT-inserted	10
s38417-T100	5819	3	0.052	0%	HT-inserted	10
s38417-T200	5822	1	0.017	0%	HT-inserted	10
s38584-T200	7580	1	0.013	0%	HT-inserted	10
s38584-T300	9110	1	0.011	0%	HT-inserted	10
s820	315	2	0.635	N/A	HT-free	9
s832	313	2	0.639	N/A	HT-free	9
s5378	2996	2	0.067	N/A	HT-free	9
s15850	2429	2	0.082	N/A	HT-free	9
s38417	5807	1	0.017	N/A	HT-free	9
b10	206	3	4.082	N/A	HT-free	8
b15	8922	1	0.011	N/A	HT-free	8
b19	108332	3	0.003	N/A	HT-free	8
s953	443	1	0.226	N/A	HT-free	8
s1423	751	1	0.133	N/A	HT-free	8

次ページに続く

前ページからの続き

ベンチマーク	ネット数	最大トロイポイントネット数	最大トロイポイントネット数 [%]	誤検出ネット数 [%]	HT-inserted /free	HT rank
s35932	6423	2	0.031	N/A	HT-free	8
usb_funct	13215	2	0.015	N/A	HT-free	8
b03	160	4	2.5	N/A	HT-free	7
b07	441	8	1.814	N/A	HT-free	7
b08	183	1	0.546	N/A	HT-free	7
b09	170	1	0.588	N/A	HT-free	7
b11	770	3	0.39	N/A	HT-free	7
b12	1076	14	1.301	N/A	HT-free	7
b13	362	9	2.486	N/A	HT-free	7
b14	10098	53	0.525	N/A	HT-free	7
b18	6867	23	0.335	N/A	HT-free	7
c432	196	3	1.531	N/A	HT-free	7
c1908	913	1	0.11	N/A	HT-free	7
c3540	1719	1	0.058	N/A	HT-free	7
c5315	2485	10	0.402	N/A	HT-free	7
c7552	3720	22	0.591	N/A	HT-free	7
EthernetMAC10GE	103206	33	0.032	N/A	HT-free	7
pci_bridge32	17357	1	0.006	N/A	HT-free	7
RS232	298	11	3.691	N/A	HT-free	7
s344	194	1	0.515	N/A	HT-free	7
s349	195	4	2.051	N/A	HT-free	7
s1196	561	6	1.07	N/A	HT-free	7
s1238	543	6	1.105	N/A	HT-free	7
s1488	677	3	0.443	N/A	HT-free	7
s9234	5847	6	0.103	N/A	HT-free	7
s13207	8660	5	0.058	N/A	HT-free	7
vga_lcd	70157	6	0.009	N/A	HT-free	7
AES	13811	35	0.253	N/A	HT-free	6
b04	737	7	0.95	N/A	HT-free	6
b05	988	18	1.822	N/A	HT-free	6
b06	56	4	7.143	N/A	HT-free	6
b17	12783	66	0.516	N/A	HT-free	6
b20	4956	2	0.04	N/A	HT-free	6
b21	5080	2	0.039	N/A	HT-free	6
b22	7590	2	0.026	N/A	HT-free	6
c499	243	10	4.115	N/A	HT-free	6
c2670	1502	1	0.067	N/A	HT-free	6
des_perf	98873	37	0.037	N/A	HT-free	6
mem_ctrl	11825	1	0.008	N/A	HT-free	6
s298	146	7	4.795	N/A	HT-free	6

次ページに続く

前ページからの続き

ベンチマーク	ネット数	最大トロイポイントネット数	最大トロイポイントネット数 [%]	誤検出ネット数 [%]	HT-inserted /free	HT rank
s382	192	3	1.563	N/A	HT-free	6
s386	182	9	4.945	N/A	HT-free	6
s400	198	3	1.515	N/A	HT-free	6
s420	262	6	2.29	N/A	HT-free	6
s444	215	5	2.326	N/A	HT-free	6
s510	246	2	0.813	N/A	HT-free	6
s838	522	6	1.149	N/A	HT-free	6
s38584	7380	7	0.095	N/A	HT-free	6
wb_conmax	22182	7	0.032	N/A	HT-free	6
b01	49	2	4.082	N/A	HT-free	5
b02	28	1	3.571	N/A	HT-free	5
s27	27	1	3.704	N/A	HT-free	5
s641	443	1	0.226	N/A	HT-free	5
s713	457	1	0.219	N/A	HT-free	5
s526	227	2	0.881	N/A	HT-free	4
c17	11	0	0	N/A	HT-free	0
c880	1198	0	0	N/A	HT-free	0
c1355	587	0	0	N/A	HT-free	0
c6288	2448	0	0	N/A	HT-free	0

以上

3.4 本章のまとめ

本章ではゲートレベルネットリストの脆弱性を表現する指標として HT rank を提案した．HT rank はシミュレーションツールを使わずにトロイポイントを計算することで算出する．HT rank は全ての Trust-HUB，ISCAS85，ISCAS89，ITC99 のゲートレベルネットリストに加え，いくつかの OpenCores ゲートレベルネットリスト, そしてハードウェアトロイの挿入されている AES と挿入されていない AES に対して，ハードウェアトロイの有無を分類することに成功した．HT rank はゲートレベルネットリストを分類することのできる強力な指標である．

第4章　回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法

4.1　本章の概要

本章では，トロイネットの信号遷移の特徴を明らかにし，信号遷移をベクタ化して解析するアルゴリズムを示し，回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法を提案する[1]．以下に本章の構成を示す．

第 4.2 節「ハードウェアトロイの分類」では，トリガとペイロードによるハードウェアトロイの分類を説明する．

第 4.3 節「信号遷移」では，ネットの信号遷移をベクタ化する方法を説明する．

第 4.4 節「定常状態の学習に基づくハードウェアトロイ検出」では，回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法を提案する．短時間ランダムシミュレーションと長時間ランダムシミュレーションで得られる信号遷移を比較することで，ハードウェアトロイの検出および機能を推定する．

第 4.5 節「実験結果」では，提案手法のソフトウェア実験の結果を示し，提案手法の有効性を評価する．

第 4.6 節「本章のまとめ」では，本章の内容をまとめる．

4.2　ハードウェアトロイの分類

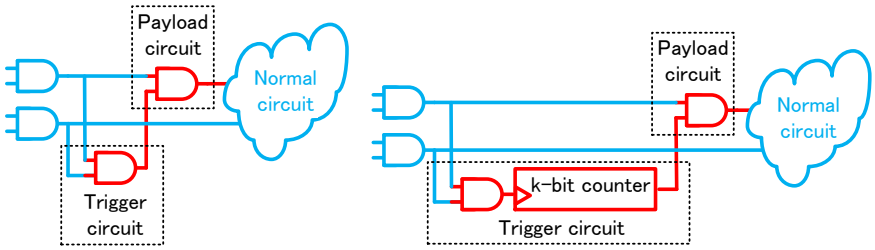
4.2.1　トリガによるハードウェアトロイの分類

トリガに着目した場合のハードウェアトロイのタイプを二種類に分類する (図 4.1 参照):

組み合わせ回路をトリガとするハードウェアトロイ: これらのハードウェアトロイは，組み合わせ回路でトリガが構成され，特定の値が入力された場合に起動する．

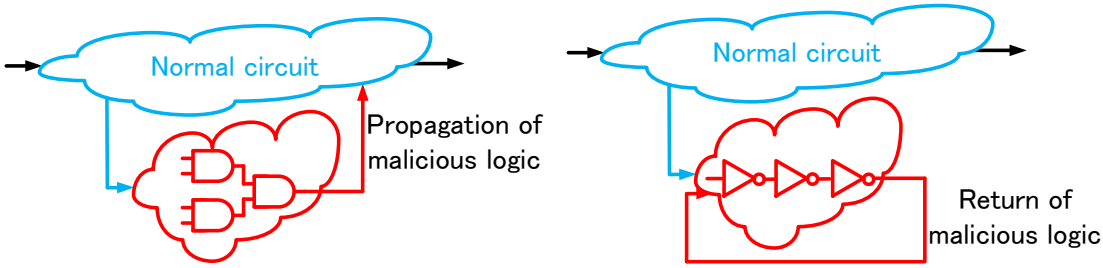
順序回路をトリガとするハードウェアトロイ: これらのハードウェアトロイは，順序回路でトリガが構成され，一定回数以上の状態遷移を経た場合に起動する．

[1]本章の内容は，文献 [29,32,54] による．



(a) 組み合わせ回路をトリガとするハードウェアトロイ. (b) 順序回路をトリガとするハードウェアトロイ.

図 4.1: トリガによるハードウェアトロイの分類.



(a) 観測可型トロイ. (b) 観測不可型トロイ.

図 4.2: ペイロードによるハードウェアトロイの分類.

組み合わせ回路をトリガとするハードウェアトロイは，既存の論理テストベースのハードウェアトロイ検出手法がハードウェアトロイを起動することに成功した場合に検出できる．しかし，順序回路をトリガとするハードウェアトロイは，組み合わせ回路をトリガとするハードウェアトロイを起動させることよりも困難である．4.5.1 節の表 4.2 に Trust-HUB [9] で提供されているハードウェアトロイが挿入されているベンチマークを分類してある．

4.2.2 ペイロードによるハードウェアトロイの分類

ペイロードに着目した場合のハードウェアトロイのタイプを二種類に分類する (図 4.2 参照).

観測可型トロイ: このハードウェアトロイは悪意のある信号をオリジナルの設計に伝搬させるため，プライマリ出力を観測することで検出できる．

観測不可型トロイ: このハードウェアトロイは悪意のある信号をオリジナルの設計に伝搬させずにハードウェアトロイ内で完結する．そのため，オリジナルの設計の内部状態やプライマリ出力に影響を与えない．

観測可型トロイは既存の論理テストベースの手法が対象としているハードウェアトロイである．しかし，観測不可型トロイはオリジナルの設計の内部状態やプライマリ出力を観測しても変化がないため，これらの手法では検出することができない．

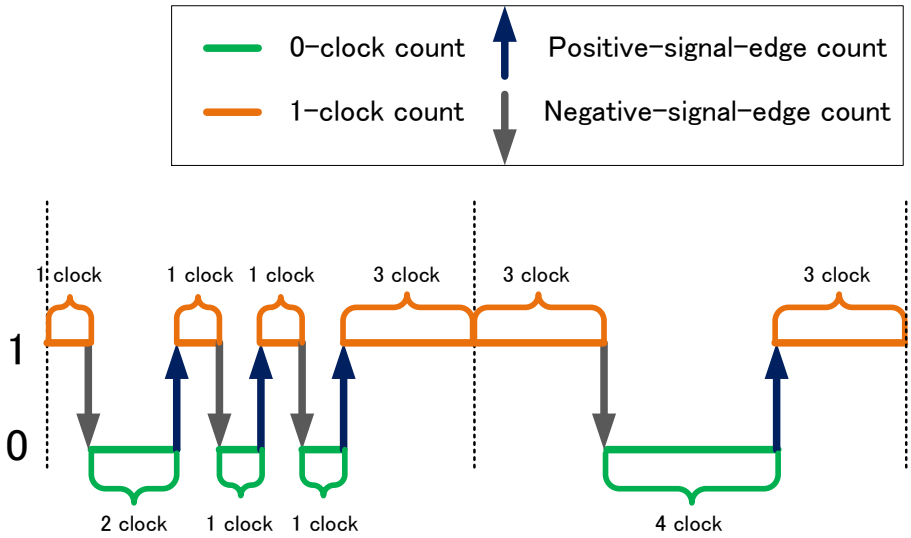


図 4.3: ネット i の信号遷移が 2 つの 10 クロック間のセクションで構成される例.

4.3 信号遷移

4.3.1 信号遷移ベクタ

論理シミュレーションにおける信号遷移を考える．ここでは3値（0, 1, X）論理シミュレーションを使うと仮定する．しかし，通常の信号は初期状態が設定されるまでXを保持し続けるため，ここではXを考慮しない．

ネット i の信号遷移状態を決定するために，一定クロック間のセクションを考える．例えば，図4.3のようにネット i を20クロックサイクルシミュレートし，10クロックずつ2つのセクションで分割する．セクション sc_j におけるネット i の信号遷移状態は以下によって特徴づけられる．

n_0: セクション sc_j における0を保持し続けるクロックサイクル数の合計.
n_1: セクション sc_j における1を保持し続けるクロックサイクル数の合計.
n_p: セクション sc_j におけるクロックの立ち上がり数の合計.
n_n: セクション sc_j におけるクロックの立ち下がり数の合計.

したがって，セクション sc_j におけるネット i の信号遷移状態は信号遷移ベクタ $s_{i,j} = (n_0, n_1, n_p, n_n)$ として表現される．

図4.3はネット i の信号遷移が2つの10クロック間のセクションで構成されていることを示す．1番目のセクションは $n_0 = 4$, $n_1 = 6$, $n_p = 3$, $n_n = 3$ である．この場合，信号遷移ベクタは信号遷移ベクタは $\mathbf{s_{i,1}} = (4, 6, 3, 3)$ となる．同様に2番目のセクションの信号遷移ベクタは $\mathbf{s_{i,2}} = (4, 6, 1, 1)$ となる．

表 4.1: 1,000,000 クロックサイクルシミュレーションした際のハードウェアトロイの動作.

ベンチマーク	ペイロードタイプ	トリガ名	HT activation?
AES-T500	Non-observable	Tj_Trig	No
s38584-T200	Observable	Trigger_out	No

4.3.2 定常信号遷移ベクタ

ここで，ハードウェアトロイに対して以下の仮定を置く．

仮定 1: 多くのハードウェアトロイは特別な条件を満たした時にのみ動作するように設計されている．そのため，短時間のシミュレーションでランダムにテストパターンを与えてもハードウェアトロイは隠れたままである．

この仮定が正しいか否かを調べるために，Trust-HUB のベンチマークから観測可型（Observable）トロイと観測不可（Non-observable）型トロイを 1 つずつ選択し，これらに対してて 1,000,000 クロックサイクルのランダムシミュレーションを Synopsys VCS で行った．表 4.1 にその結果を示す．表 4.1 中の “トリガ名” は，各ベンチマークのトリガ信号を意味しており，この信号が “1” になったらハードウェアトロイは動作する．[2] トリガ信号は 1,000,000 クロックサイクルのランダムシミュレーション中に “1” にならなかったため，ハードウェアトロイは動作しなかった．

短時間シミュレーションでの実験で使用したベンチマークは観測可型トロイと観測不可型トロイの両方が含まれている．したがって，仮定 1 は正しいといえる．

この仮定に基づき，短時間ランダムシミュレーションで得たネット i の信号遷移ベクタを定常信号遷移ベクタと呼ぶ．定常信号遷移ベクタはハードウェアトロイを動作させない特徴を持つ．この定常信号遷移ベクタがトロイネットとノーマルネットを分ける強力なヒントとなる．

4.4 定常状態の学習に基づくハードウェアトロイ検出

ゲートレベルネットリスト NL に挿入されているハードウェアトロイを検出することを考える．ネット i を疑トロイネットとする．これは，ネット i がトロイネットの可能性が高いと疑われるネットを意味する．定常信号遷移ベクタはネット i の定常的な振る舞いを示す．ネット i に対して長時間ランダムシミュレーションを行い，その振る舞いが定常信号遷移ベクタと似通っている場合，ネット i はトロイネットではないといえる．ネット i に対して長時間ランダムシミュレーションを行い，その振る舞いが時々定常信号遷移ベクタと著しく異なる場合，ネット i は本当にトロイネットであるといえる．この手続きにゴールデンテストパターン・レスポンスおよびプライマリ出力の観測は行っていない．

[2] トリガ信号は Trust-HUB によって全てのネットリストに与えられている．

このアイデアに基づき，定常状態の学習に基づくハードウェアトロイ検出手法を提案する．初めに，疑トロイネットを検出する．疑トロイネットは本当にトロイネットかどうかがわからない状態のネットである．疑トロイネットは設計者の注意深い検査もしくは既存のトロイネット検出手法によって発見される．既存のトロイネット検出手法はトロイネットを検出できるが，トロイネットのみを完全に検出することはできない．

提案手法は，既存のトロイネット検出手法によって得たトロイネットの集合 STN に対して疑トロイネットの振る舞いを監視する．疑トロイネットを取得後，各ネット $i \in STN$ の信号遷移ベクタを短時間ランダムシミュレーションによって得る（Phase 1）．次に，ネット i の振る舞いを長時間ランダムパターンを与えてシミュレーションで得る．この信号遷移ベクタと Phase 1 で得た定常信号遷移ベクタとを比較することで，異常セクションを検出する．異常セクションが 1 つでも存在する場合，ネット i はトロイネットである（Phase 2）．異常セクション中の信号遷移回数を数え上げることで，ハードウェアトロイの機能を推定する（Phase 3）．

4.4.1 Phase 1: 定常信号遷移ベクタの学習

Phase 1 は各トロイネット $i \in STN$ の定常状態を学習する．初めに M クロックサイクルだけランダムシミュレーションをネットリスト NL に行う．4.3.2 節で議論した仮定 1 に基づき，ここで行う短時間シミュレーションではハードウェアトロイは動作しないため，定常的な信号遷移状態を得ることができる．M クロックを c クロックサイクルのセクションに分割するため，各セクション数は (M/c) となる．セクションの集合 SSC は以下で与えられる．

$$SSC = \{ssc_1, \ldots, ssc_{M/c}\}. \tag{4.1}$$

各セクション $ssc_j \in SSC$ は c クロックサイクルとなる．

各疑トロイネット $i \in STN$ と各セクション $ssc_j \in SSC$ に対して，“0” と “1” を保持するクロックサイクル数の合計（n_0 と n_1），クロックの立ち上がりと立ち下がり数の合計（n_p と n_n）を数えることにより定常信号遷移ベクタ $\boldsymbol{s}_{i,j}$ を得る．各疑トロイネット i における定常信号遷移ベクタの集合 S_i は，以下で与えられる．

$$S_i = \{\boldsymbol{s}_{i,1}, \ldots, \boldsymbol{s}_{i,M/c}\}. \tag{4.2}$$

S_i はネット i の定常状態を示す．

Phase 1 を要約したアルゴリズムをアルゴリズム 4 に示す．

4.4.2 Phase 2: ネットの振る舞いの監視

Phase 2 ではネットリスト NL を長時間ランダムシミュレートし，各疑トロイネット $i \in STN$ が本当にトロイネットがどうかを調べる．ネットリスト NL を N クロックサイクルだけ監視

Algorithm 1 (Phase 1) 定常信号遷移ベクタの学習.

Require: Gate-level netlist NL and a set STN of suspicious Trojan nets
Ensure: A set S_i of steady signal-transition vectors for every net $i \in STN$
1: Simulate NL in M cycles by giving test patterns generated by an ATPG tool and obtain signal transitions for every net $i \in STN$.
2: **for** Every net $i \in STN$ **do**
3: Calculate a steady signal-transition vector $\boldsymbol{s}_{i,j}$ for every section ssc_j and generate S_i
4: **end for**

する．N と M の大小関係は $N >> M$ である．N クロックを c クロックサイクルのセクションに分割するため，各セクション数は (N/c) となる．セクションの集合は以下で与えられる．

$$SC = \{sc_1, \ldots, sc_{N/c}\}. \tag{4.3}$$

各セクション $sc_j \in SC$ も c クロックサイクルとなる．

各疑トロイネット $i \in STN$ と各セクション $sc_j \in SC$ に対してシミュレーションを行うことで，信号遷移ベクタ $\boldsymbol{t}_{i,j}$ を得る．各疑トロイネット i の信号遷移ベクタの集合 T_i は以下で与えられる．

$$T_i = \{\boldsymbol{t}_{i,1}, \ldots, \boldsymbol{t}_{i,N/c}\}. \tag{4.4}$$

定常信号遷移ベクタと判断する範囲を計算するために，δ_{01}, δ_{pn} を導入する．δ_{01} は，信号遷移ベクタ $\boldsymbol{t}_{i,j}$ の n_0, n_1 が定常状態かどうかを判断する閾値となる．δ_{pn} は，信号遷移ベクタ $\boldsymbol{t}_{i,j}$ の n_p, n_n が定常状態かどうかを判断する閾値となる．

二つの信号遷移ベクタ $\boldsymbol{s} = (n_0^s, n_1^s, n_p^s, n_n^s)$ と $\boldsymbol{t} = (n_0^t, n_1^t, n_p^t, n_n^t)$ のそれぞれに対して関数 $TH(\boldsymbol{s}, \boldsymbol{t})$ を以下で定義する．

$$TH(\boldsymbol{s}, \boldsymbol{t}) = \begin{cases} 0, & \text{if all of } \frac{|n_0^s - n_0^t|}{n_0^s} \leq \delta_{01}, \\ & \frac{|n_1^s - n_1^t|}{n_1^s} \leq \delta_{01}, \\ & \frac{|n_p^s - n_p^t|}{n_p^s} \leq \delta_{pn}, \text{and} \\ & \frac{|n_n^s - n_n^t|}{n_n^s} \leq \delta_{pn} \text{ hold} \\ 1, & \text{otherwise} \end{cases}$$

$TH(\boldsymbol{s}, \boldsymbol{t}) = 0$ の場合，$\boldsymbol{t}$ は $\boldsymbol{s}$ に似通っているといえる．$TH(\boldsymbol{s}, \boldsymbol{t}) = 1$ の場合，$\boldsymbol{t}$ は $\boldsymbol{s}$ と異なっているといえる．セクション $sc_j \in SC$ と疑トロイネット $i \in STN$ における信号遷移ベクタは $\boldsymbol{t}_{i,j}$ である.. 定常信号遷移ベクタ $\boldsymbol{s}_{i,k} \in S_i$ が $TH(\boldsymbol{s}_{i,k}, \boldsymbol{t}_{i,k}) = 1$ となる場合，sc_j は疑トロイネット i の異常セクションである．疑トロイネット i の異常セクションの集合 A_i は以下で与えられる．

$$A_i = \{sc_j \in SC \mid sc_j \text{ is an abnormal section for } i\}. \tag{4.5}$$

疑トロイネット i の異常セクションの集合 A_i が空集合以外 $A_i \neq \emptyset$ の場合，疑トロイネット i は本当にトロイネットである．異常セクションの集合 A_i が空集合 $A_i = \emptyset$ ならば，疑トロイネット i はノーマルネットである．1つでもトロイネットがネットリストに存在するなら，そのネットリストにはハードウェアトロイが挿入されている．

例え異常セクションが存在していなくても，順序回路をトリガとするハードウェアトロイは非常に長い時間動作しない．そのため，常に同じ値を保ち続けるセクションに注目し，特に一定セクションと呼ぶことにする．4.3.2 節で議論した仮定に基づき，ハードウェアトロイは動作していない間は常に同じ値を保ち続けると考えられる．そこで疑トロイネット i がほとんどすべてのセクションで常に同じ値を保ち続けた場合はトロイネットと判断する．i の

一定セクションの集合を C_i とすると，C_i は以下のように定義される．

$$C_i = \{sc_j \in SC \mid sc_j \text{ is a constant section for } i\}. \tag{4.6}$$

もし $|C_i|$ の割合が全てのセクションの 99%以上だった場合，疑トロイネット i を真のトロイネットと判断する．そうでない場合は，疑トロイ i とノーマルネットと判断する．

ネットリスト NL が一つ以上の真トロイネットを含む場合，NL にはハードウェアトロイが挿入されると判定する．ネットリスト NL が一つも真トロイネットを含まない場合，NL にはハードウェアトロイが挿入されていないと判定する．

Phase 2 を要約したアルゴリズムをアルゴリズム 2 に示す．

Algorithm 2 (Phase 2) ネットの振る舞いの監視．

Require: Gate-level netlist NL and a set S_i of steady signal-transition vectors for every suspicious net $i \in STN$

Ensure: Every net $i \in STN$ is a real Trojan net or not and the netlist NL is Trojan inserted or not

```
Simulate NL in N-clock cycles by giving random test patterns and obtain signal transitions for every net i ∈ STN;
for every net i ∈ STN do
    Obtain a set A_i of abnormal sections;
    if A_i ≠ ∅ then
        The net i is a real Trojan net;
    else if |C_i| ≥ 0.99 × |SC| then
        The net i is a real Trojan net;
    else
        The net i is a normal net;
    end if
end for
if NL includes Trojan nets then
    NL is Trojan-inserted;
else
    NL is Trojan-free;
end if
```

4.4.3 Phase 3: ハードウェアトロイの機能を推定する

Phase 3 は Phase 2 で異常セクションを検出した際に実行する．ハードウェアトロイの振る舞いは大まかに二つのグループに分類される．

(タンパ性の振る舞い) オリジナル回路の機能を変更する目的のハードウェアトロイの振る舞いである．例えば，秘密情報を漏洩させたり等である．

(機能低下性の振る舞い) オリジナル回路の機能を変更せずに機能低下を目的としたハードウェアトロイの振る舞いである．例えば，消費電力の増大等である．

タンパ性の振る舞いと機能低下性の振る舞いは，表 4.2 に示されているように 4.2 節で定義したハードウェアトロイのタイプとは無関係である．信号線の信号遷移は，ハードウェアトロイの振る舞いを分析するうえで有効であることを以下に概略する．

1. **タンパ性の振る舞いの信号遷移:** タンパ性の振る舞いは図 4.4(a) に示すように比較的少ない信号遷移である．例として，秘密漏洩機能の振る舞いの場合，回路が特定の条件を満足した時，ハードウェアトロイは短時間の間プライマリアウトプットを通じて特定のネットの値を漏洩する．このような場合，信号遷移回数は少ない．
2. **機能低下性の振る舞いの信号遷移:** 機能低下性の振る舞いは図 4.4(b) に示すように比較的多い信号遷移である．例として，オリジナルの回路よりも消費電力を増大させる場合，ハードウェアトロイは長い間隠れているが，一度起動すると消費電力を増大させるためにトロイネットは頻繁に遷移するようになる．

注意すべきだが，トロイネットとノーマルネットは時々同じような振る舞いをする．図 4.4 では，青色のクロックサイクルはトロイネットがノーマルネットのように振る舞うことを示し，赤色のクロックサイクルではトロイネットの様に振る舞うことを示している．これらの振る舞いはハードウェアトロイにおける重要な特徴である．

したがって，定常状態の信号遷移と異常セクションにおける信号遷移を比較することで，ハードウェアトロイの振る舞いから機能を推定することができる

疑トロイネットを i とし，i の定常信号遷移ベクタ集合を $S_i = \{s_{i,1}, \ldots, s_{i,M/c}\}$ とする．各信号遷移ベクタを $s = (n_0, n_1, n_p, n_n)$ とした時，信号遷移回数 $ST(s)$ は以下のように定義される．

$$ST(s) = n_p + n_n \tag{4.7}$$

i の定常信号遷移ベクタ中の最大信号遷移回数 $ST_{max}(S_i)$ は以下のように与えられる．

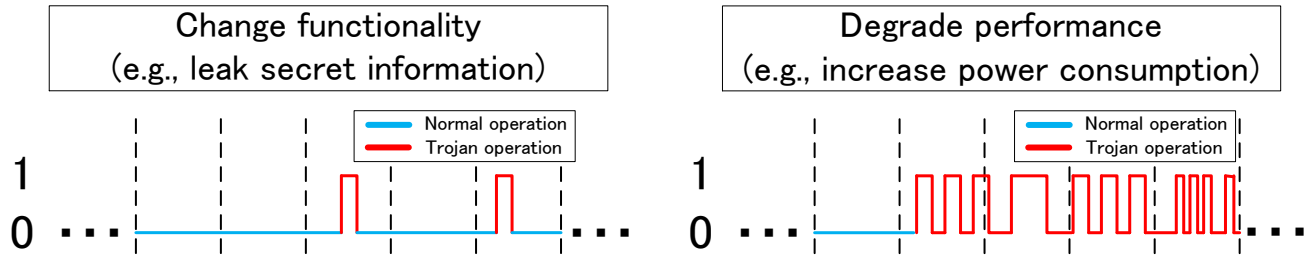
$$ST_{max}(S_i) = \max_{1 \le j \le M/c} ST(s_{i,j}) \tag{4.8}$$

Phase 2 で検出したハードウェアトロイと疑われるトロイネットの集合 STN とその集合中に含まれる真のトロイネットの集合 TN との関係は $TN \subseteq STN$ である．Phase 1 の定常信号遷移ベクタの集合 S_i と Phase 2 の異常セクションの集合 A_i における真のトロイネットは $i \in TN$ である．Phase 3 のプロセスを以下で説明する．

初めに，異常セクションの集合 A_i における A_i^1 and A_i^2 は各ネット $i \in TN$ を利用して以下の様に定義する．

$$A_i^1 = \{sc_j \in A_i | ST(t_{i,j}) < \delta_b \times ST_{max}(S_i)\} \tag{4.9}$$

$$A_i^2 = \{sc_j \in A_i | ST(t_{i,j}) \ge \delta_b \times ST_{max}(S_i)\} \tag{4.10}$$



(a) タンパ性の振る舞い. (b) 機能低下性の振る舞い.

図 4.4: ハードウェアトロイの振る舞い.

δ_b は閾値のパラメータである[3]. A_i^1 における各セクションは信号遷移回数が少なく，疑トロイネット i がタンパ性の振る舞いであることを示唆する. A_i^2 における各セクションは信号遷移回数が多く，疑トロイネット i が機能低下性の振る舞いであることを示唆する.

次に TN における全てのトロイネットに対して以下の計算をする.

$$A^1 = \sum_{i \in TN} |A_i^1| \tag{4.11}$$

$$A^2 = \sum_{i \in TN} |A_i^2| \tag{4.12}$$

もし $A^1 > A^2$ ならば，タンパ性のハードウェアトロイであると推定する. もし $A^1 < A^2$ ならば，機能低下性のハードウェアトロイであると推定する.

Phase 3 を要約したアルゴリズムをアルゴリズム 3 に示す.

Algorithm 3 (Phase 3) ハードウェアトロイの機能を推定する.

Require: Gate-level netlist NL and a set A_i of abnormal sections for every real Trojan net $i \in TN$ detected in Phase 2
Ensure: HT behavior; (Tamper-behavior) or (Degradation-behavior)
1: Partition A_i into A_i^1 and A_i^2 for each net $i \in STN$.
2: Calculate the A^1 and A^2 values
3: **if** $A^1 > A^2$ **then**
4: Output Tamper-behavior
5: **else if** $A^1 < A^2$ **then**
6: Output Degradation-behavior
7: **else**
8: We cannot guess HT behavior.
9: **end if**

ほとんどのハードウェアトロイは一つの機能を持つため，十分に小さいサイズで仕込むことができる. もしタンパ性と機能低下性の振る舞いを持つハードウェアトロイが存在する場

[3]実験結果を 4.5 で示している. 閾値のパラメータ $\delta_b = 2.0$ とすることで，タンパ性の振る舞いと機能低下性の振る舞いを区別することに成功している.

合，提案したアルゴリズムではそのようなハードウェアトロイを正しく判別することはできない．

4.5 実験結果

4.5.1 セットアップ

本節では，Trust-HUB [9] で提供されているいくつかのハードウェアトロイ入り RTL レベルとゲートレベルネットリスト，ハードウェアトロイの挿入されていないゲートレベルネットリストに対して提案手法を評価する．AES-T500 と AES-T1300 は TSMC 90nm ライブラリを使用して，Synopsys Design Compiler で論理合成を行った．信号遷移を得るために Phase 1 と Phase 2 で Synopsys VCS を使用した．

表 4.2 は Trust-HUB ベンチマークの詳細を示す．“Weak sequential” は小さいカウンタをトリガとしているベンチマークを示す．例えば，EthernetMAC10GE-T710 のトリガは 1-bit カウンタを持つため，トリガタイプを “Weak sequential” としている．一方， b19-T200 は 32-bit カウンタを持つため，トリガタイプを “Sequential” としている．“ペイロードタイプ” はベンチマークが観測可型トロイもしくは観測不可型トロイかを示している．“HT-behavior” はハードウェアトロイの振る舞いがタンパ性か機能低下性かを示している．各ベンチマークを注意深く分析して，“トリガタイプ”，“ペイロードタイプ” と “振る舞い” を分類した．“トリガタイプ” と “ペイロードタイプ” は RTL-level コードとゲートレベルネットリストをそれぞれ検査することで，“振る舞い” は Trust-HUB で提供されている仕様書から得た．

ハードウェアトロイの挿入されているベンチマークに加え，ハードウェアトロイの挿入されていないベンチマークとして RS232，s38417 と s38584 を Trust-HUB から用意した．Trust-HUB では，ハードウェアトロイの挿入されていない RS232 を提供していないため，RS232-T1000 から手動でハードウェアトロイを取り除くことで作成した．

スコアに基づく分類手法 [30] は最新の強力なパターンベースの疑トロイネット検出手法である．この手法は悪意のある回路パターンと一致するネットにスコアを与え，最大スコアを持つネットを疑トロイネットとして検出する．

表 4.3 は本実験で使用したパラメータである．各パラメータは経験的に得られたパラメータである．

表 4.2: ベンチマークの説明.

ベンチマーク	トリガタイプ	ペイロードタイプ	振る舞い	ハードウェアトロイの機能 [9]
AES-T500	Weak sequential	Non-observable	Degradation-behavior	The Trojan increases the power consumption.
AES-T1300	Combinational	Non-observable	Tamper-behavior	The Trojan has an additional dynamic power consumption [26].
b19-T200	Sequential	Observable	Tamper-behavior	The Trojan payload is one OR gate which restitches a design net through.
EthernetMAC10GE-T710	Weak sequential	Observable	Tamper-behavior	Whenever Trojan gets triggered, its payload gains control over an internal signals.
RS232-T1600	Combinational	Observable	Tamper-behavior	Whenever Trojan gets triggered, its payload gains control over two primary output signals.
s35932-T300	Combinational	Non-observable	Degradation-behavior	The Trojan payload is a ring oscillator along some signal path.
s38417-T100	Combinational	Observable	Tamper-behavior	The Trojan gains control over an internal signal after activation.
s38417-T200	Combinational	Observable	Tamper-behavior	The Trojan payload propagates erroneous values over four internal signals.
s38417-T300	Weak sequential	Non-observable	Tamper-behavior	The Trojan payload leaks the value of a specific net through side-channel signals.
s38584-T200	Weak sequential	Observable	Tamper-behavior	The Trojan payload leaks the value of one internal signal through a primary output.

表 4.3: 実験で使用したパラメータ.

c	M	N	δ_{01}	δ_{pn}	δ_b
10,000	1,000,000	100,000,000	1.7	1.0	2.0

4.5.2 Phase 1: 定常信号遷移ベクタの学習結果

スコアに基づく分類手法 [30] で検出した疑トロイネットに対して，短時間ランダムシミュレーションを行った．図 4.5 にハードウェアトロイの挿入されていないネットリストである s38584 とハードウェアトロイの挿入されているネットリストである EthernetMAC10GE-T710 と s35932-T300 の定常信号遷移ベクタの例を示す．本実験では $c = 10,000$ と $M = 1,000,000$ を設定し，信号遷移を 100 セクションに分割し，100 個の定常信号ベクタを得た．図の各棒グラフは 0 を保持し続けるクロックサイクル数の合計（0-clock count），1 を保持し続けるクロックサイクル数の合計（1-clock count），クロックの立ち上がり数の合計（positive-signal-edge count），クロックの立ち下がり数の合計（negative-signal-edge count）である．

各ネットがノーマルネットかトロイネットかは Trust-HUB から与えられている．図 4.5(a) と (b) の EthernetMAC10GE-T710 のネット Tj_OUTClock と s35932-T300 のネット Tj_OUT1234 は共にトロイネットである．図 Fig. 4.5(c) の s38584 のネット g28030 はノーマルネットである．

図 4.5(a) と図 4.5(b) は，Tj_OUTClock と Tj_OUT1234 が全てのセクションで長い間 1 の値を保ち続けていることを示す．これは，例えハードウェアトロイのトリガやペイロード，機能が異なったとしても動作していない時は同じような挙動になることを意味するこのデータは 4.3.2 で議論した仮定が正しかったことを意味し，提案手法がハードウェアトロイの定常状態を適切に学習することが可能であることを示す．一方，g28030 のセクションは同じ値を保ち続けておらず，図 4.5(c) に示すように一定セクションを持たないことを示す．これら 3 つの信号遷移を比較することで，短時間のランダムシミュレーションではノーマルネットを動作させることはできてもトロイネットを動作させることはできないことがわかる．

4.5.3 Phase 2: ネットの振る舞いの監視結果

次に，疑トロイネットに対して長時間ランダムシミュレーションを行った．表 4.4 に Phase 2 の結果を要約する．表 4.4 中の “ネット名” は [30] で検出した各ベンチマークの疑トロイネットを示す．もし各疑トロイネットに異常セクション数が 1 つ以上ある場合，疑トロイネットをトロイネットと判断する．表 4.4 に示されているように，全てのトロイネットをトロイネット，全てのノーマルネットをノーマルネットと識別することに成功している．この長時間シミュレーションにおいて，ハードウェアトロイが挿入されているネットリストは実際にハードウェアトロイが動作した．提案手法は異常セクションというアイデアを導入することで，ハードウェアトロイを正しく識別する．

図 4.6 は長時間ランダムシミュレーションで得た信号遷移ベクタの例を示す．特に EthernetMAC10GE-

T710とs35932-T300で検出した異常セクションを含む100個のセクションを選んだ．図4.6(a)によると，ネットTj_OUTClockは0-clock countsを含むセクションが2つあることが確認できるが，定常信号遷移ベクタでは図4.5(a)に示すように1-clock countsのセクションしか存在しない．そのためPhase 2でこの2つのセクションを異常セクションとして検出する．s35932-T300では図4.6(a)と図4.6(b)の違いが分かりやすい．s35932-T300は機能低下性の振る舞いを持つため，ネットTj_OUT1234は動作していると考えられる．s38584の図4.5(c)と図4.6(c)が似通っているのは，長時間ランダムシミュレーションで得られた信号遷移ベクタと定常信号遷移ベクタに違いがみられないからであり，したがってPhase 2はs38584に対して異常セクションを検出しない．

長時間ランダムシミュレーションでは，いくつかのハードウェアトロイが実際に動作した．これは異常セクションというアイデアの導入がハードウェアトロイの識別に効果的であることを示している．

AES-T1300，s38417-T100とs38417-T200のハードウェアトロイは長時間ランダムシミュレーションでも動作しなかった．しかし，提案手法はこれらのハードウェアトロイを検出することができ，なおかつノーマルネットとトロイネットを区別することにも成功した．これ一定セクションというアイデアの導入が効果的であることを示している．

提案した学習アルゴリズムは組み合わせ回路もしくは序回路をトリガとしたハードウェアトロイを検出することに成功した．例え，疑トロイネットに観測不可型トロイだったとしても，提案手法は正しく識別することができる．組み合わせ回路もしくは順序回路をトリガとしたハードウェアトロイ，観測可型もしくは観測不可型トロイをハードウェアトロイが動作するテストパターンやプライマリアウトプットのデータを必要としないというPhase 2の実験結果は，提案手法が既存の論理テストベースのハードウェアトロイ検出手法よりも強力であることを示している．

4.5.4 Phase 3: ハードウェアトロイの機能の推定結果

表4.5にPhase 3の実験結果を要約する．“推測の正誤”は提案手法がほとんどのケースにおいて正しくハードウェアトロイの機能を推測したことを示す．例えば，Phase 3は図4.6(b)に示したようにネットが頻繁に動作しているため，s35932-T300を機能低下性の振る舞いと正しく推測した．

Phase 2はAES-T1300，s38417-T100とs38417-T200において異常セクションを検出しなかった．そのため，Phase 3ではこれらのベンチマークに対してハードウェアトロイの振る舞いを推測することができなかった．したがって，実験結果では“Unknown”と“Incorrect”であると示した．しかし，Phase 3のアルゴリズムはPhase 2ので異常セクションを検出した場合は全てのベンチマークに対して正しく機能を推測している．以上の実験結果より，Phase 3は極めて強力なハードウェアトロイの機能を推測するアルゴリズムである．

4.5.5 既存の論理テストベース手法との比較

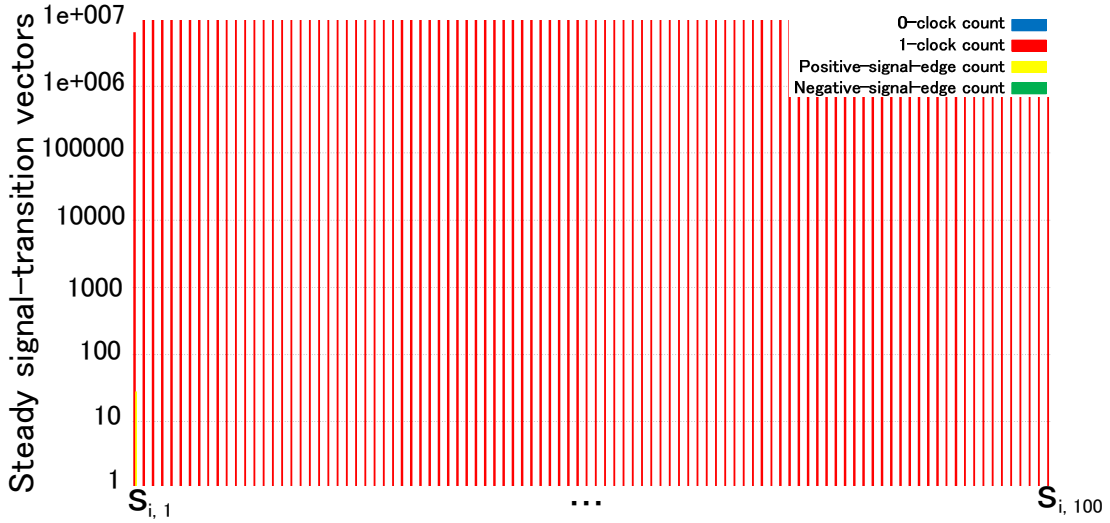
提案手法といくつかの既存手法とを比較する．表 4.6 に比較結果を示す．

[47] はゴールデンテストパターン・レスポンスとプライマリ出力に悪意ある論理値が伝搬する必要がある．[37] と [39] も同様に悪意ある論理値の伝搬が必要であるため，これらの手法は観測不可型トロイを検出することは極めて困難である．これらの手法は観測可型トロイを検出することはできる．表 4.6 中の “detectable” は，常にハードウェアトロイを検出するテストパターンを発見することができないかもしれないが，そのようなテストパターンをこれらの手法に与えれば検出できるという意味である．

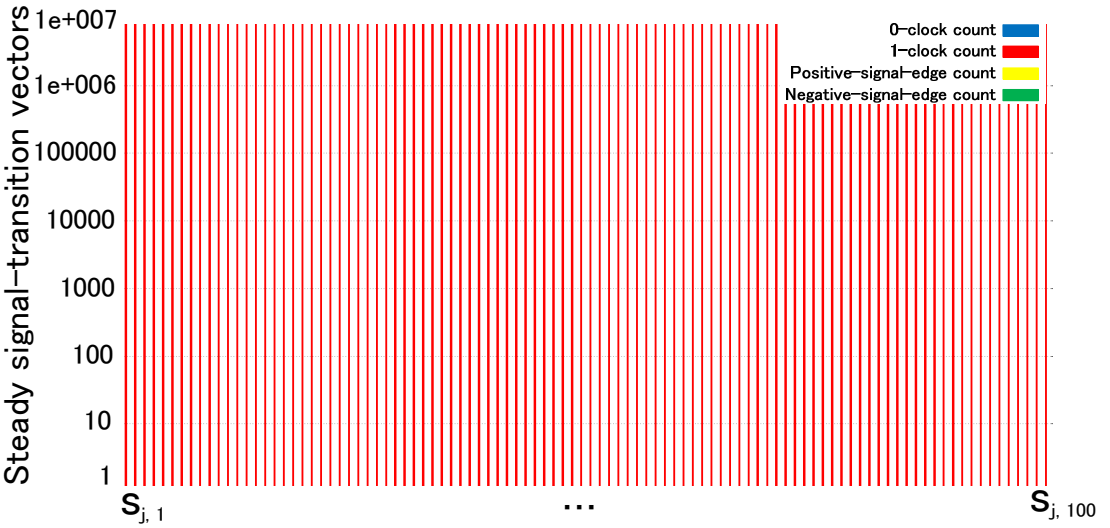
注目すべきは，提案手法がゴールデンテストパターン・レスポンスと悪意ある論理値の伝搬なしに，観測可型トロイと観測不可型トロイの検出に成功していることである．

4.6 本章のまとめ

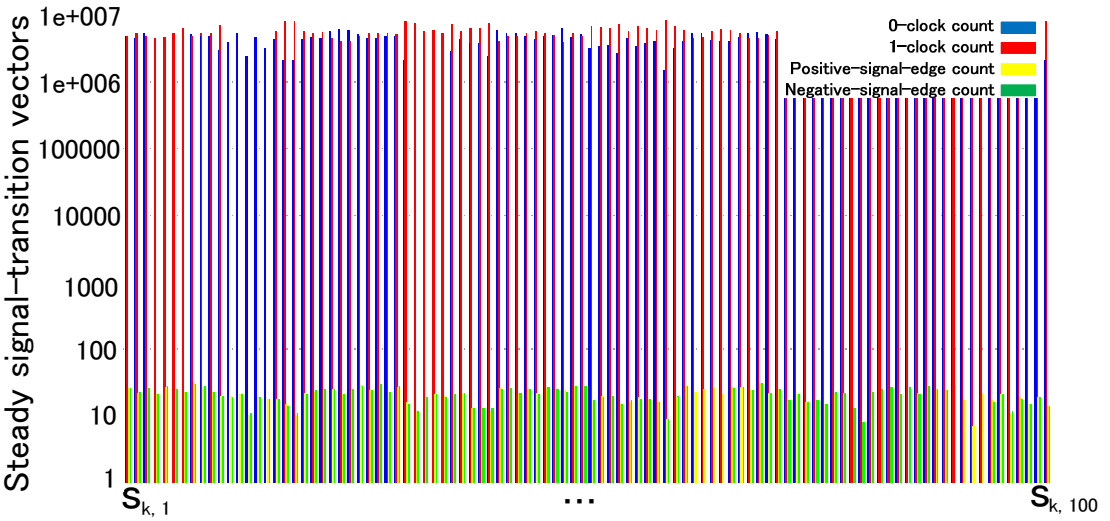
本章では，定常信号遷移ベクタを利用するハードウェアトロイ検出手法を提案した．本アプローチは短時間ランダムシミュレーションの信号遷移と長時間ランダムシミュレーションの信号遷移を比較することでハードウェアトロイを検出する．提案手法を Trust-HUB ベンチマークに適用した結果，ハードウェアトロイを検出することに成功した．



(a) EthernetMAC10GE-T710 のネット i = Tj_OUTClock における定常信号遷移ベクタ $(\boldsymbol{s}_{i,1}, \ldots, \boldsymbol{s}_{i,100})$.

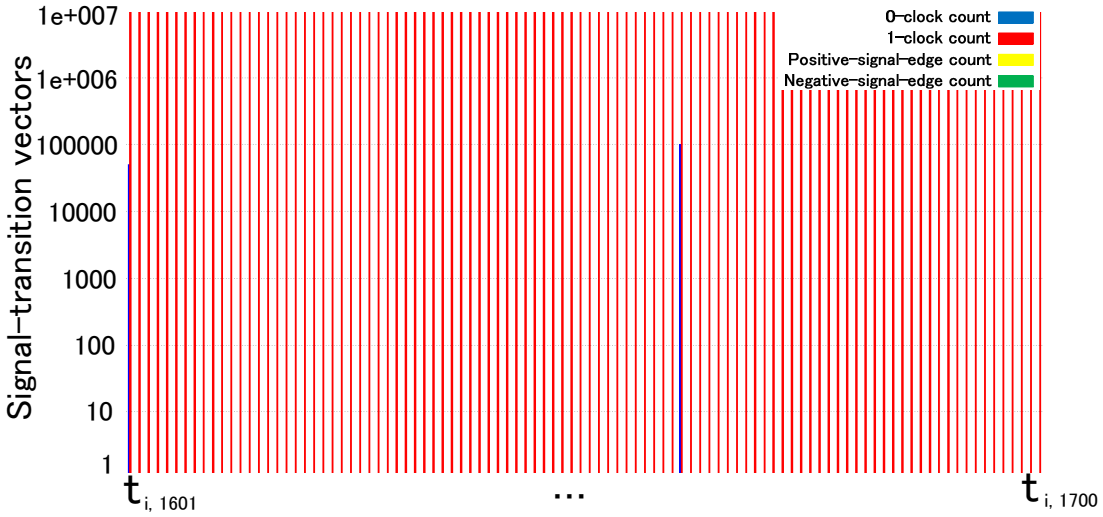


(b) s35932-T300 のネット j = Tj_OUT1234 における定常信号遷移ベクタ $(\boldsymbol{s}_{j,1}, \ldots, \boldsymbol{s}_{j,100})$.

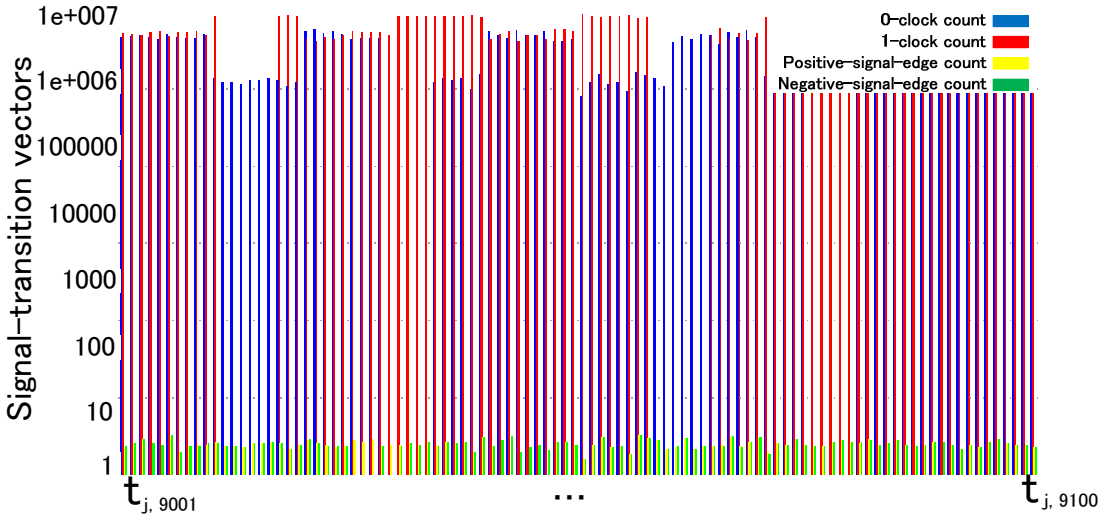


(c) s38584 のネット k = g28030 における定常信号遷移ベクタ $(\boldsymbol{s}_{k,1}, \ldots, \boldsymbol{s}_{k,100})$.

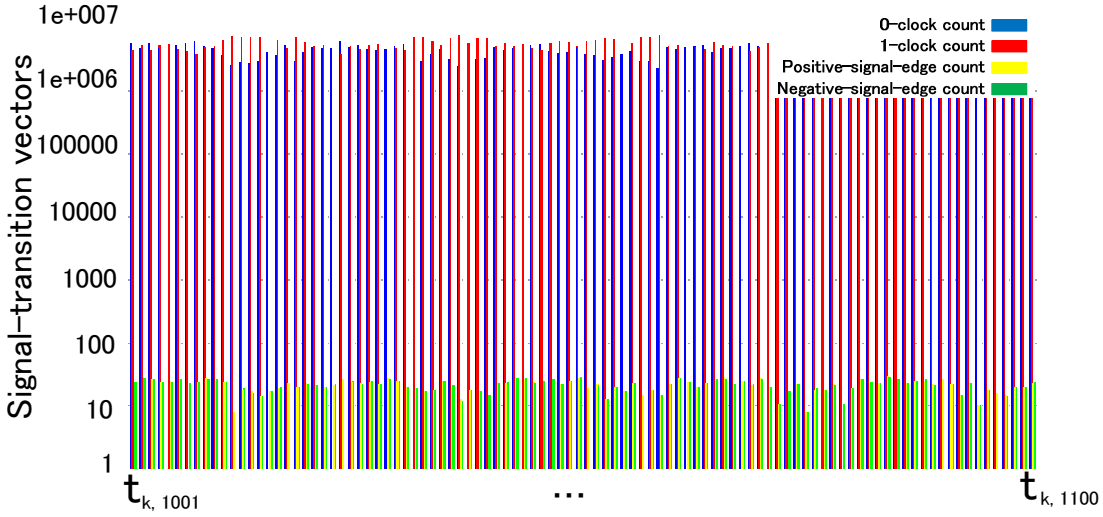
図 4.5: Phase 1 の実験結果: EthernetMAC10GE-T710, s35932-T300 と s38584 の定常信号遷移ベクタ.



(a) EthernetMAC10GE-T710 のネット i = Tj_OUTClock における信号遷移ベクタ $(\boldsymbol{s}_{i,1601}, \ldots, \boldsymbol{s}_{i,1700})$.



(b) s35932-T300 のネット j = Tj_OUT1234 における信号遷移ベクタ $(\boldsymbol{s}_{j,9001}, \ldots, \boldsymbol{s}_{j,9100})$.



(c) s38584 のネット k = g28030 における信号遷移ベクタ $(\boldsymbol{s}_{k,1001}, \ldots, \boldsymbol{s}_{k,1100})$.

図 4.6: Phase 2 の実験結果: EthernetMAC10GE-T710，s35932-T300 と s38584 の信号遷移ベクタ.

表 4.4: Phase 2 の実験結果.

ベンチマーク	分類	ネット名	異常セクション数	一定セクション数	検出結果	真トロイネット
AES-T500	A	n267	1,000	8999	Yes	Yes
		n269	1	9998	Yes	Yes
		n274	1	9998	Yes	Yes
		n277	1	9998	Yes	Yes
		n278	1	9998	Yes	Yes
		n368	0	9999	Yes	Yes
AES-T1300	A	n51	0	9,927	Yes	Yes
b19-T200	A	Trigger_out	5	7,226	Yes	Yes
EthernetMAC10GE-T710	A	Tj_OUTClock	227	9,768	Yes	Yes
RS232-T1600	A	iXMIT_CRTL	1	9,586	Yes	Yes
		iCTRL	2	9,997	Yes	Yes
s35932-T300	A	Tj_Trigger	500	8,582	Yes	Yes
		Tj_OUT1234	1,000	8,999	Yes	Yes
		Tj_OUT5678	1,000	8,999	Yes	Yes
s38417-T100	C	Tj_Trigger	0	9,999	Yes	Yes
		Tj_OUT1234	0	9,999	Yes	Yes
		Tj_OUT5678	0	9,999	Yes	Yes
		n2401	0	4,530	No	No
		g25489	0	0	No	No
s38417-T200	C	Tj_Trigger	0	9,999	Yes	Yes
		Tj_OUT1234	0	9,999	Yes	Yes
		n2401	0	4,530	No	No
		g25489	0	0	No	No
s38417-T300	A	Tj_Trigger	7	9,711	Yes	Yes
s38584-T200	A	Trigger_out	5	7,226	Yes	Yes
RS232	B	rec_readyH	0	1,366	No	No
		xmit_doneH_temp	0	1,640	No	No
		rec_dataH[0]	0	537	No	No
		rec_dataH[1]	0	494	No	No
		rec_dataH[2]	0	582	No	No
		rec_dataH[3]	0	621	No	No
		rec_dataH[4]	0	623	No	No
		rec_dataH[5]	0	652	No	No
		rec_dataH[6]	0	536	No	No
		rec_dataH[7]	0	539	No	No
		uart_XMIT_dataH	0	59	No	No
		iRECEIVER_next_state_1_	0	0	No	No
		iRECEIVER_next_state_0_	0	0	No	No
		n83	0	0	No	No
		n53	0	408	No	No
		n371	0	0	No	No
s38417	B	n2401	0	4,530	No	No
		g25489	0	0	No	No
s38584	B	g26875	0	0	No	No
		n3304	0	4,139	No	No
		g28030	0	0	No	No
		g31793	0	0	No	No
		g34383	0	0	No	No
		g34201	0	0	No	No
		g33659	0	0	No	No
		n2393	0	0	No	No
		n2436	0	6,516	No	No

表 4.5: Phase 3 の実験結果.

ベンチマーク	A^1	A^2	機能推測	推測の正誤
AES-T500	4	1,000	Degradation-behavior	Correct
AES-T1300	0	0	Unknown	Incorrect
b19-T200	5	0	Tamper-behavior	Correct
EthernetMAC10GE-T710	227	0	Tamper-behavior	Correct
RS232-T1600	2	1	Tamper-behavior	Correct
s35932-T300	0	2,500	Degradation-behavior	Correct
s38417-T100	0	0	Unknown	Incorrect
s38417-T200	0	0	Unknown	Incorrect
s38417-T300	7	0	Tamper-behavior	Correct
s38584-T200	5	0	Tamper-behavior	Correct

表 4.6: 既存の論理テストベース手法との比較.

ベンチマーク	ペイロードタイプ	[47]	[39]	提案手法
AES-T500	Non-observable			✓
AES-T1300	Non-observable			✓
b19-T200	Observable	detectable	detectable	✓
EthernetMAC10GE-T710	Observable	detectable	detectable	✓
RS232-T1600	Observable	detectable	detectable	✓
s35932-T300	Non-observable			✓
s38417-T100	Observable	detectable	detectable	✓
s38417-T200	Observable	detectable	detectable	✓
s38417-T300	Non-observable			✓
s38584-T200	Observable	detectable	detectable	✓
Requires HT activation test patterns/responses		✓	✓	
Requires primary output observation		✓	✓	

第5章　ハードウェアトロイ機能の無効化

5.1　本章の概要

本章では，トロイネットの信号遷移の特徴に基づいて無効化を行うアルゴリズムを示し，ハードウェアトロイの機能を無効化する内部トロイ無効化を提案する[1]．以下に本章の構成を示す．

第 5.2 節「内部トロイ無効化」では，ハードウェアトロイの機能を無効化する内部トロイ無効化を提案する．埋め込まれたトロイ無効化回路がトロイネットと疑われるネットの一定クロック間のビットフリッピング回数を監視し，本当のトロイか否かを識別し，それらが本当にトロイであるか否かを判断する．トロイネットと判断した場合は無効化し，トロイの機能をマスキングする．これにより，ハードウェアトロイが挿入されている信頼性の低いネットリストを安全に動作させる．

第 5.3 節「実験結果」では，提案手法のソフトウェア実験の結果を示し，提案手法の有効性を評価する．

第 5.4 節「本章のまとめ」では，本章の内容をまとめる．

5.2　内部トロイ無効化

本節では信頼性の低いネットリストに対する内部トロイ無効化技術を提案する.. アイデアは信頼性の低いネットリストに含まれる全てのトロイネットと疑われるネットに対してプログラマブルなマスキング回路を埋め込み，回路動作中にトロイネットと判断されたら無効化する．

提案する設計フローを図 5.1 に示し，以下に説明する．

信頼性の低い設計段階:

1. 信頼性の低いネットが悪意のある EDA ツールによって生成される．しかし，そのネットがハードウェアトロイかどうかを知ることはできない．

信頼性のある設計段階:

[1]本章の内容は，文献 [32,58] による．

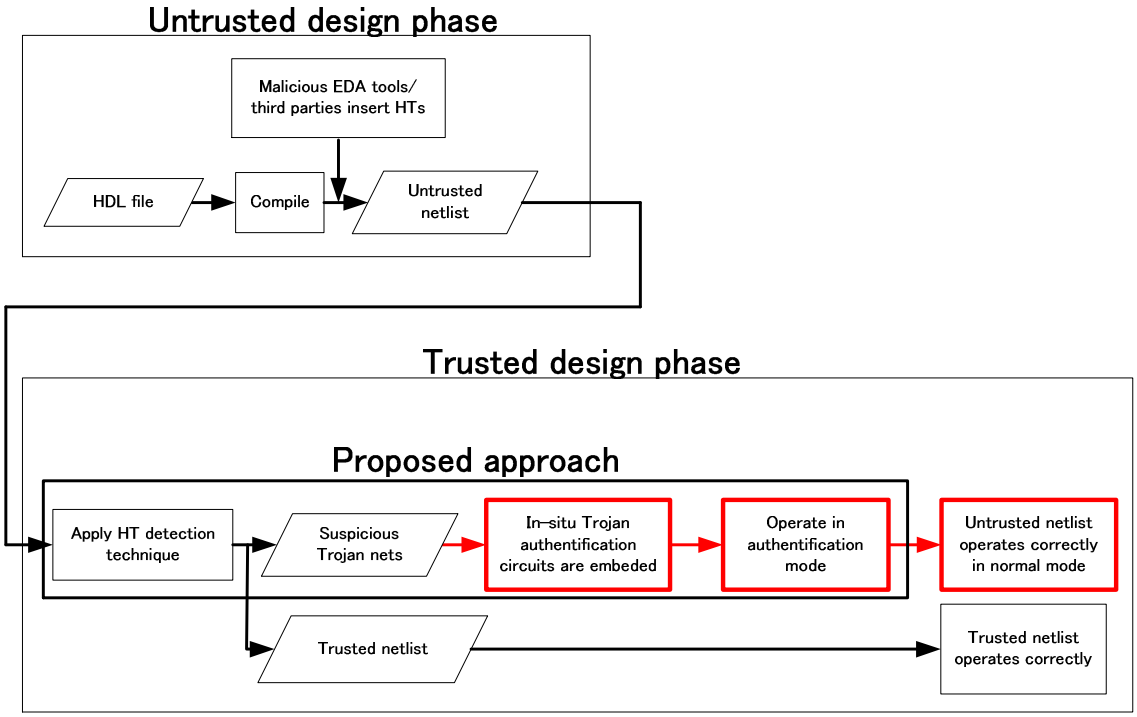


図 5.1: 内部トロイ無効化.

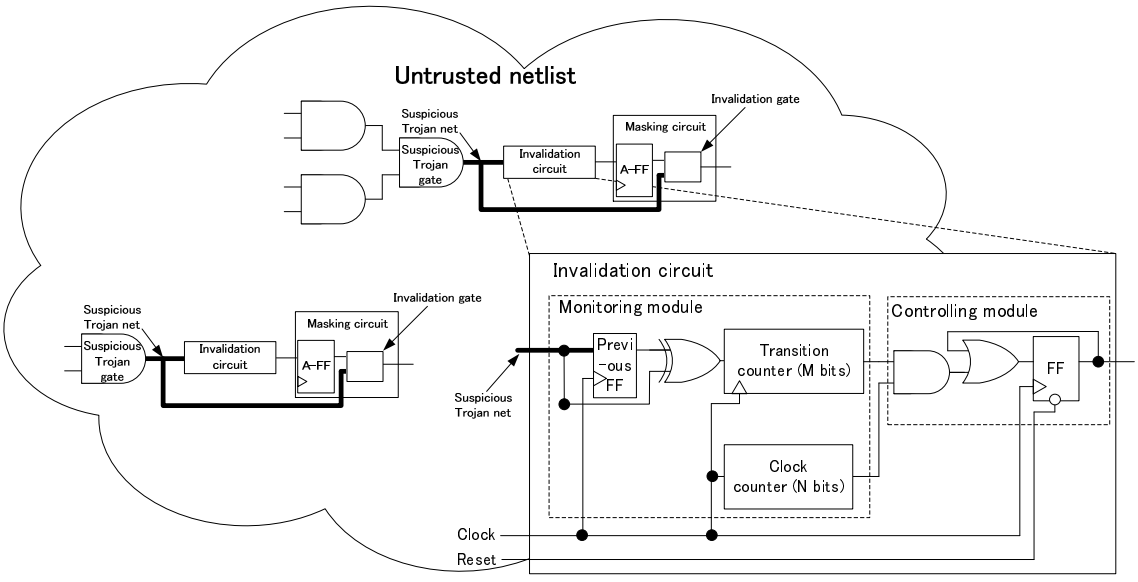


図 5.2: 内部トロイ無効化回路.

2. 与えられた信頼性の低いネットリストに対して, [30] の様なハードウェアトロイ検出手法を適用し，トロイネットと疑われる回路を特定する．得られたトロイネットと疑われるネットはトロイネットとノーマルネットを両方含むが，膨大な設計コストや検証コストなしに完全に区別することは不可能である．
3. 各トロイネットと疑われるネットに対して, 不揮発性の 1 ビット認証フリップフロップ（A-FF）を含むマスキング回路と無効化回路を埋め込む．A-FF はトロイネットと疑われるネットを判断する．無効化回路は回路動作中にトロイネットと疑われるネットが無効化条件を満たすかどうかを検査する．
4. 信頼性の低いネットリストを監視モードもしくはノーマルモードで動作させる．

   監視モード: 初めに，このモードでチップ上で監視を行う．ネットリストを動作させている間に, 提案した無効化回路がトロイネットと疑われるネットの一定クロック間のビットフリッピング回数を監視し，本当の真トロイネットかどうかを識別する．無効化回路は無効化条件を満足した場合のみトロイネットと疑われるネットを有効化する．

   ノーマルモード: このモードでは，無効化条件を満足しなかったネットを無効化することで，信頼性の低いネットリストを正しく動作させる．例え信頼性の低いネットリストにハードウェアトロイが存在していても，内部トロイ無効化回路が埋め込まれていれば正しく動作させることができる．

このアプローチでは，無効化条件をどのように設定するかが重要な鍵となる．有効な無効化条件の設定方法，無効化回路，マスキング回路の設計を提案する．

5.2.1 無効化条件

賢いハードウェアトロイは滅多に動作しないように設計されている．したがって，監視モードにおいて，同じクロックサイクル間のビットフリッピング回数をカウントすることで，トロイネットとノーマルネットを区別するヒントが得られる．

Algorithm 4 内部トロイ無効化の疑似コード.

```
Reset transition_counter and clock_counter associated with a suspicious Trojan net n;
while clock_counter < 2^N do
  if bit-flip occurs in n then
    transition_counter++;
  end if
  clock_counter++;
end while
if transition_counter ≥ (2^M − 1) then
  Set the A-FF for n to be validated;
else
  Set the A-FF for n to be invalidated;
end if
```

アルゴリズム 4 に監視モードにおける内部トロイ無効化の疑似コードを示す．最初に，各無効化回路を各トロイネットと疑われるネットに埋め込み，ビットフリッピングを監視する．次に，遷移カウンタがビットフリッピングをカウントし，クロックカウンタがクロック数をカウントする．2^N クロックサイクル数後，遷移カウンタの値と $(2^M - 1)$ を比較する．もし遷移カウンタの値が $(2^M - 1)$ 以上ならば，トロイネットと疑われるネットはノーマルネットであると識別する．もしそうでなければ，トロイネットと疑われるネットはトロイネットと識別する．A-FF はトロイネットと疑われるネットを無効化するかどうかを記憶する．

パラメータ (M, N) を無効化条件と呼ぶ．5.3 節で議論するが，(M, N) は Trust-HUB ベンチマークをトレーニングセットとして決定した．このベンチマークはハードウェアトロイ有りと無しのゲートレベルネットリストがある．

特に監視モードでは設計者がテストパターンを生成して入力し，トロイネットと疑われるネットを判断する．テストパターンを長時間信頼性の低い IC に入力し，2^M クロックサイクル中にトロイネットと疑われるネットが有効化されない場合, トロイネットだと識別してノーマルモードでは値がマスクされる．

5.2.2 無効化回路アーキテクチャ

各トロイネットと疑われるネット n に対して，モニタリングモジュールとコントローリングモジュールで構成された無効化回路を埋め込む．

モニタリングモジュールは図 5.2 のような二つのカウンタで構成されている．一つは遷移カウンタであり，もう一つはクロックカウンタである．遷移カウンタはトロイネットと疑われるネットのビットフリッピング回数をカウントし，クロックカウンタはクロックサイクル数をカウントする．もし遷移カウンタが $(2^M - 1)$ になった場合，“1” をアウトプットする．そうでない場合，“0” をアウトプットする．もしクロックカウンタが $(2^N - 1)$ になった場合，“1” をアウトプットする．そうでない場合，“0” をアウトプットする．“プレビオス FF” はトロイネットと疑われるネットの 1 クロック前の値を記憶し，現在の値と前の値を比較することでビットフリップをカウントする．

コントローリングモジュールはマスキング回路中のクロックカウンタと遷移カウンタのアウトプットにより，1 ビット A-FF をセット/リセットする．クロックカウンタのアウトプットが 1 の時，もし遷移カウンタが 1 ならば A-FF には 1 がセットされる．つまり，トロイネットと疑われるネットが有効化される．遷移カウンタが 0 ならば A-FF には 0 が設定される．トロイネットと疑われるネットが有効化されない場合，一度 A-FF に 0 がセットされると，監視モード中に二度と 1 になることはない．この場合，AND ゲートをマスキング回路として使用することができる．A-FF が 0 の時，トロイネットと疑われるネットをマスクすることができる．加えて，OR ゲートをマスキング回路として使用する場合も同様にマスクできる．

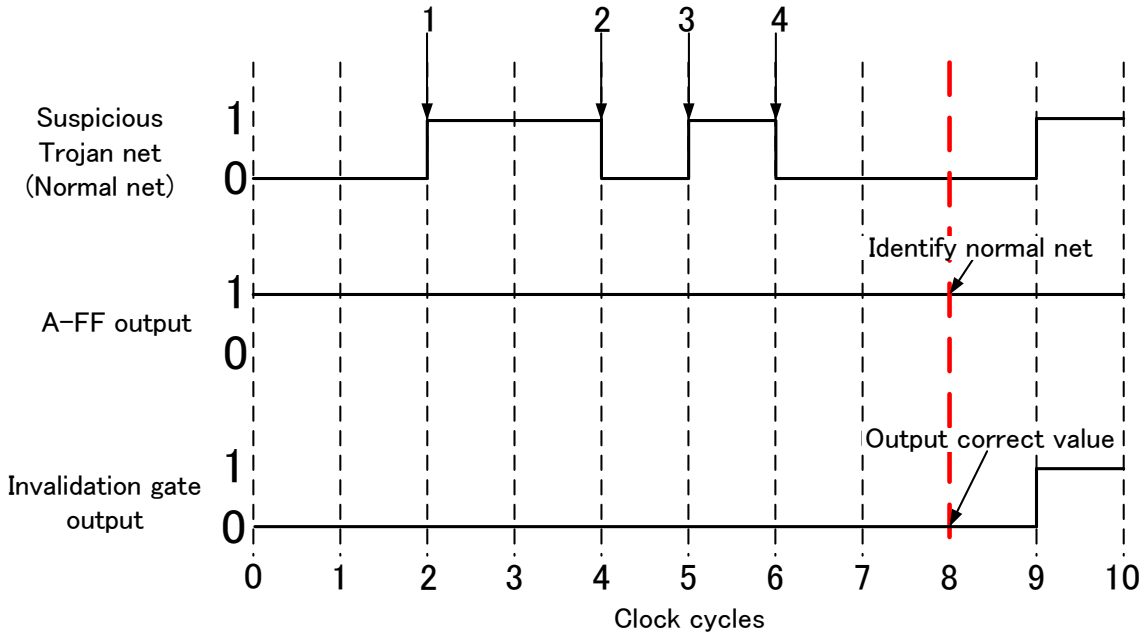


図 5.3: トロイネットと疑われるネットが有効化される例（$(M, N) = (2, 3)$）.

5.2.3 マスキング回路アーキテクチャ

各トロイネットと疑われるネット n に対して，インバリデイションゲートと A-FF で構成されるマスキング回路を埋め込む．A-FF の値に基づいて，インバリデイションゲートはトロイネットと疑われるネットの値を無効化もしくは有効化する．各トロイネットと疑われるネットのインバリデイションゲートは事前のシミュレーションに基づく．

特に全ての A-FF をスキャンモード中ではシリアルに接続することで外部から制御することができる．もし無効化回路が誤って有効化した時，外部から制御可能である．

5.2.4 使用例

内部トロイ無効化がどのように動くかを簡単な例を用いて説明する．無効化条件を $(M, N) = (2, 3)$ と仮定する．回路が $2^N = 2^3 = 8$ サイクル間，トロイネットと疑われるネットのビットフリッピング回数を監視する．インバリデイションゲートは 2 入力 AND ゲートとする．

図 5.3 において，ビットフリッピング回数は $2^M - 1 = 2^2 - 1 = 3$ より多い 4 となる．したがって，内部トロイ無効化回路はノーマルネットと識別する．A-FF が 1 になったら，インバリデイションゲートがトロイネットと疑われるネットの正しい値をアウトプットする．

図 5.4 において，ビットフリッピング回数は 3 より少ない 1 となる．したがって，内部トロイ無効化回路はトロイネットと識別する．A-FF が 0 になったら，インバリデイションゲートがトロイネットと疑われるネットの値を無効化する．

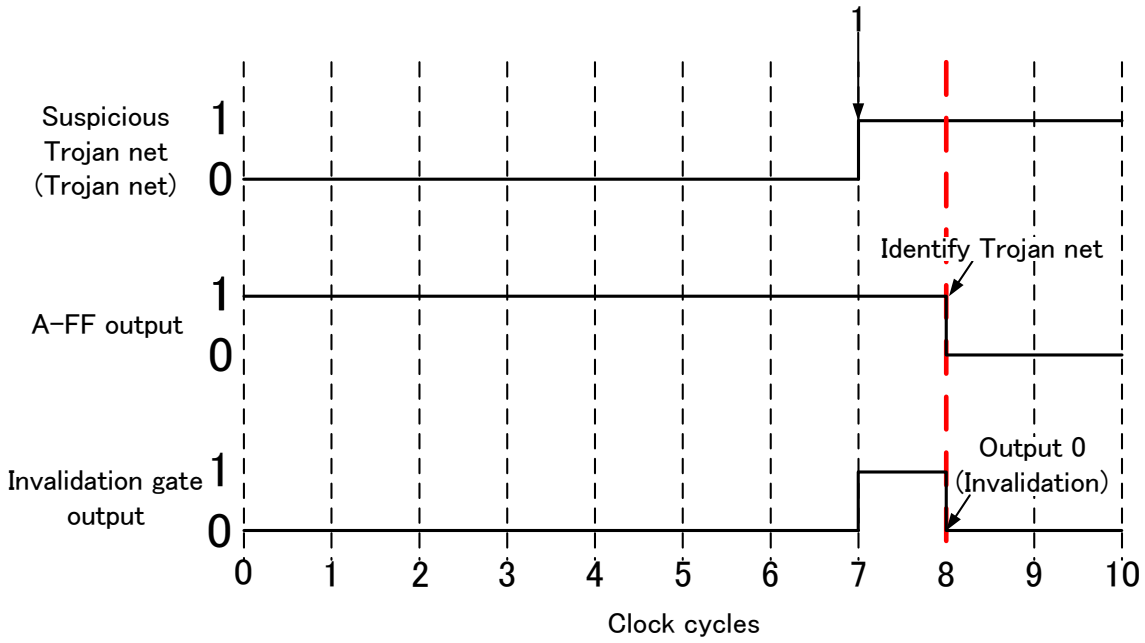


図 5.4: トロイネットと疑われるネットが有効化されない例（$(M, N) = (2, 3)$）.

表 5.1: Trust-HUB のハードウェアトロイ有りと無しのゲートレベルネットリストに含まれるトロイネットと疑われるネット数.

ベンチマーク	HT-inserted/free	ネット数	疑トロイネット数
s15850	HT-free	2,429	5
s38417	HT-free	5,807	2
s38417-T100	HT-inserted	5,819	3
s38417-T200	HT-inserted	5,822	1
s38584-T300	HT-inserted	9,110	1
vga_lcd-T100	HT-inserted	70,157	2

表 5.2: トロイネットと疑われるネットの説明.

ベンチマーク	ネット名	Trojan/Normal net
s15850	n1132	Normal net
	n1440	Normal net
	n1491	Normal net
	n1509	Normal net
	n1546	Normal net
s38417	g25489	Normal net
	n2401	Normal net
s38417-T100	Tj_Trigger	Trojan net
	Tj_OUT1234	Trojan net
	Tj_OUT5678	Trojan net
s38417-T200	Tj_OUT1234	Trojan net
s38584-T300	Trigger_out	Trojan net
vga_lcd-T100	Tj_Trigger	Trojan net
	Tj_OUT1	Trojan net

表 5.3: Trust-HUB ベンチマークの結果.

ベンチマーク	ネット名	$M=1$ $N=2$	$M=1$ $N=3$	$M=2$ $N=3$	$M=1$ $N=4$	$M=2$ $N=4$	$M=3$ $N=4$	$M=1$ $N=5$	$\mathbf{M=2}$ $\mathbf{N=5}$	$M=3$ $N=5$	$M=4$ $N=5$	$M=1$ $N=6$	$M=2$ $N=6$	$M=3$ $N=6$	$M=4$ $N=6$	$M=5$ $N=6$
s15850	n1132	✓	✓	FP	✓	✓	FP	✓	✓	FP	FP	✓	✓	FP	FP	FP
	n1440	✓	✓	FP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	n1491	✓	✓	FP	✓	FP	FP	✓	✓	✓	FP	✓	✓	✓	✓	FP
	n1509	✓	✓	FP	✓	FP	FP	✓	✓	✓	FP	✓	✓	✓	✓	FP
	n1546	✓	✓	FP	✓	FP	FP	✓	✓	✓	FP	✓	✓	✓	✓	FP
s38417	g25489	FP	FP	FP	FP	FP	FP	✓	✓	FP	FP	✓	✓	✓	✓	✓
	n2401	FP	FP	FP	✓	✓	FP	✓	✓	✓	FP	✓	✓	✓	✓	FP
s38417-T100	Tj_Trigger	FN	FN	✓	FN	✓	✓	FN	✓	✓	✓	FN	✓	✓	✓	✓
	Tj_OUT1234	FN	FN	✓	FN	✓	✓	FN	✓	✓	✓	FN	✓	✓	✓	✓
	Tj_OUT5678	FN	FN	✓	FN	✓	✓	FN	✓	✓	✓	FN	✓	✓	✓	✓
s38417-T200	Tj_OUT1234	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
s38584-T300	Trigger_out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
vga_lcd-T100	Tj_Trigger	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Tj_OUT1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

5.3 実験結果

本節では，初めに Trust-HUB [9] で入手したハードウェアトロイ有りと無しのゲートレベルネットリストをトレーニングセットとし，無効化条件 (M, N) を最適化するために内部トロイ無効化回路を埋め込んでシミュレーションする．シミュレーション後，内部トロイ無効化回路をハードウェアトロイ入り 128 ビット AES に埋め込んで，提案手法の有効性を確かめる．

5.3.1 無効化条件の最適化

Trust-HUB はよく知られているハードウェアトロイのベンチマークを提供する．ここではゲートレベルの Trust-HUB ベンチマークをトレーニングセットとして使用する．

最も新しく強力なトロイネット検出手法の一つであるスコアに基づいたクラス分類手法 [30] を Trust-HUB ベンチマークに適用することで，ベンチマーク内の安全なネット数を知ることができる．検出したネットのスコアが 2 のネットをトロイネットと疑われるネットとする．[2] 表 5.1 にはトロイネットと疑われるネット数を示し，表 5.2 はトロイネットと疑われるネットの説明である．Trust-HUB の全てのベンチマークにはトロイネットがわかるため，事前にトロイネットと疑われるネットが真トロイネットかノーマルネットかがわかる．

[30] によると，表 5.1 に示したベンチマークはハードウェアトロイ有りかハードウェアトロイ無しかを区別するのが困難なネットリストである．これらに対して適用することで，提案した内部トロイ無効化回路がハードウェアトロイの機能を無効化できることを示す．監視するクロックサイクル数を変更することを考える．例えば N を変更する場合，Trust-HUB ベンチマークに対して論理シミュレータを用いて 1M クロックサイクルだけトロイネットと疑われるネットを監視して，ビットフリッピング回数をカウントする．同時に，N を変更すると (M, N) がトロイネットと疑われるネットに対して正しく無効化するかを検査することができる．

表 5.3 に無効化結果を示す．表 5.3 において，チェックマークは無効化結果が正しい結果を示す．これは提案回路がトロイネットを無効化し，ノーマルネットを有効化することを意味する．False negative (FN) は無効化回路が誤ってトロイネットを有効化した結果を示す．これは決して容認できないケースである．False positive (FP) は無効化回路が誤ってノーマルネットを無効化した結果を示す.

表 5.3 に示したように，全てのノーマルネットを有効化してトロイネットを無効化する無効化条件は $(M, N) = (2, 5)$ の時である．無効化条件 $(M, N) = (2, 5)$ に基づいて無効化回路

[2] トロイネットと疑われるネットを正しく無効化するために，ネットリストからトロイネットと疑われるネットを十分な数だけ検出する必要があるため，トロイネットと疑われるネットを定義するために議論する必要がある．このセットアップの段階では，スコアが 2 のネットをトロイネットと疑われるネットとした．これはスコアが 2 のネットはトロイネットとノーマルネットの両方を含むためである．スコアが 2 未満のネットは全てノーマルネットであり，スコアが 2 より大きいネットは全てトロイネットであるためである．

表 5.4: ハードウェアトロイの挿入された 128 ビット AES における結果 ($(M, N) = (2, 5)$).

ベンチマーク	ネット名	Type	A-FF value	無効化の正誤
Trojan-inserted AES	n1056	Normal	1	Correct
	n1854	Normal	1	Correct
	Tj_Trigger	Trojan	0	Correct
	Tj_out	Trojan	0	Correct

とマスキング回路を埋め込むことによって，例えハードウェアトロイの存在する場合でも信頼性の低い IC を正しく動作させることができる．

5.3.2 ハードウェアトロイの挿入された 128 ビット AES への内部トロイ無効化回路実装

本節では，内部トロイ無効化をハードウェアトロイ入り AES に挿入する．この AES ネットリストは 128 ビットの AES 暗号回路のネットリストであり，もし平文に “$11\cdots11$” がインプットされたら，秘密鍵がアウトプットされるようになっている．無効化条件は前節で求めた $(M, N) = (2, 5)$ を設定する．

最初にスコアに基づいたクラス分類手法 [30] を適用してトロイネットと疑われるネットを検出する．このケースの場合，表 5.4 に示すように 4 個のトロイネットと疑われるネットを検出した．検出した 4 個のトロイネットと疑われるネットの内 2 個はノーマルネットであり，残りの 2 個はトロイネットであるが，現実的にはそれを事前に知ることはできない．

監視モードでは，1,000,000 個の平文を生成してインプットし，AES ネットリストを動作させる．A-FF の値は監視結果に基づいてセットされる．表 5.4 の A-FF value の列は A-FF の値を示す．

表 5.4 において，A-FF の値を正しくセットすることができたため，トロイネットを無効化することに成功した．ノーマルモードでは，AES が正しく平文を暗号文にエンコードすることができた．特に，平文に “$11\cdots11$” をインプットしても，トロイネットが無効化されているため平文が正しく暗号文をエンコードした．

表 5.5 に 90nm テクノロジーノード下の面積とクリティカルパス遅延を示す．面積オーバーヘッドは 0.79%であり，遅延はほとんど生じなかった．一般的に，クリティカルパスは頻繁に値が遷移するため，トロイネットと疑われるネットとして検出されない．したがって，遅延オーバーヘッドは極めて低くなる．

5.3.3 内部トロイ無効化 vs UCI

UCI の結果は [51] より引用した．表 5.6 はいくつかの Trust-HUB ベンチマークに適用した時の UCI と提案手法との比較である．UCI では，ハードウェアトロイのコードを正しく

表 5.5: 面積及び遅延オーバーヘッドの比較.

	面積 [μm^2]	遅延 [ns]
Trojan-inserted 128-bit AES w/o authentication circuits	284,705	0.27
Trojan-inserted 128-bit AES w/i authentication circuits	286,942	0.27

表 5.6: UCI との比較.

ベンチマーク	機能の無効化	
	UCI [19]	提案手法
s38417-T100	Yes (remove)	Yes (invalidate)
s38417-T200	No	Yes (invalidate)
s38584-T300	No	Yes (invalidate)
vga_lcd-T100	No	Yes (invalidate)

削除することに成功した場合に “remove” と表記してある．提案手法では, 内部トロイ無効化が正しくハードウェアトロイの機能を無効化した場合に “invalidate” と表記してある．

内部トロイ無効化はハードウェアトロイの機能を無効化できているのに対して，UCI ではいくつかのベンチマークに対して悪意のあるコードを削除できない．さらに，UCI は悪意のある要素を特定するために回路検証でハードウェアトロイが動作させることが必要である [51]．提案手法はハードウェアトロイを動作させる必要はないが，適切な無効化条件が必要である．

5.4 本章のまとめ

本章ではゲートレベルネットリストのハードウェアトロイ対策として，内部トロイ無効化技術を提案した．内部トロイ無効化はトロイネットと疑われるネットのビットフリッピング回数を監視し，トロイネットとノーマルネットを区別する．無効化条件は Trust-HUB ベンチマークをトレーニングセットとして生成した．内部トロイ無効化回路をハードウェアトロイの挿入されている AES に実装し，ハードウェアトロイの機能のみを無効化することに成功した．面積オーバーヘッドは 0.79%であり，遅延オーバーヘッドは生じなかった．

第6章　結論

本論文では，設計工程におけるICに挿入されたハードウェアトロイの検出および無効化を達成し，IC自体の安全性検証に貢献する手法を提案した．各章の内容を以下にまとめる．

2章「ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法」では，トロイネットの特徴に注目し，トロイネットを検出することでハードウェアトロイを検出する手法を提案した．まず，与えられたゲートレベルのネットリストに含まれる全てのネットにスコアを与える．スコアはハードウェアトロイの挿入されているベンチマークからトロイネットを抽出し，抽出したトロイネットの特徴に応じてスコアの重みづけを行う．次に，トロイ要素という3つのパラメータを最大のスコアを持つネットに設定する．最後に，3つのトロイ要素の値とそれらの閾値を比較することで，トロイネットの可能性が最も高い強トロイネットを検出する．強トロイネットが含まれていれば，ハードウェアトロイが挿入されていると判断する．提案手法は与えられたゲートレベルのネットリストから得られる情報のみを使用するため，挿入されているハードウェアトロイの情報を必要としない．それでいて，提案手法は与えられたネットリストにハードウェアトロイが挿入されているか否かを識別する．実際に，Trust-HUBのAbstraction Gate Levelで公開されている全てのゲートレベルのネットリストに対して，ハードウェアトロイの有無を識別することに成功した．提案手法を適用する際に要する時間は高々数時間程度であった．

3章「ゲートレベルネットリストの危険性を表現する指標」では，ゲートレベルネットリストの危険性を表現する指標として*HT rank*を提案した．HT rankは3種類のトロイポイントから与えられる．1個目はキャラクタリスティックポイント（C-ポイント），2個目はスケールポイント（S-ポイント），3個目はロケーションポイント（L-ポイント）である．これらのポイントを全てのネットに割り当てることで，ゲートレベルネットリスト内の最大トロイポイントネットを検出する．最大トロイネットポイントはネットリスト内で最もトロイネットであると疑われるネットであり，最大トロイポイントがHT rankとなる．実際にHT rankは全てのTrust-HUB，ISCAS85，ISCAS89，ITC99のゲートレベルネットリストに加え，いくつかのOpenCoresゲートレベルネットリスト, そしてハードウェアトロイの挿入されているAESと挿入されていないAESに対して，ハードウェアトロイの有無を分類することに成功した．

4章「回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法」では，定常状態の学習を利用した論理テストベース手法を提案した．本手法は短時間ランダムシミュレーションを通じ，回路の信号遷移の定常状態を学習する．定常信号遷移

状態は一定のクロックサイクル間の 4 次元ベクタによって定義される．4 次元ベクタとは，0 を保持し続けるクロックサイクル数の合計，1 を保持し続けるクロックサイクル数の合計，クロックの立ち上がり数の合計，クロックの立ち下がり数の合計である．各疑トロイネットに対して長時間のシミュレーションを行い，信号遷移ベクタの集合を得る．定常信号遷移ベクタと長時間シミュレーションで得た信号遷移ベクタを比較し，定常信号遷移ベクタと異なる場合はハードウェアトロイと判断する．異常状態を検出した場合には，信号の振る舞いから機能を推測する．提案手法は観測可型トロイと観測不可型トロイの両方を検出することができることを Trust-HUB ベンチマークに対して適用することで確認した．実際に Trust-HUB ベンチマークに対して，組み合わせ回路をトリガとしたハードウェアトロイ，順序回路をトリガとしたハードウェアトロイ，観測可型トロイ，観測不可型トロイを検出し，ハードウェアトロイの機能の推測に成功したことを確認した．

5 章「ハードウェアトロイ機能の無効化」では，内部トロイ無効化技術を提案した．本手法は信頼性の低いネットリストに挿入されたハードウェアトロイの機能を無効化する．提案手法のアプローチは，信頼性の低い IC を監視モードとノーマルモードで動作させる．監視モードでは，埋め込まれたトロイ無効化回路が疑トロイネットの一定クロック間のビットフリッピング回数を監視し，本当のトロイか否かを識別し，それらが本当にトロイであるか否かを判断する．もし無効化条件を満足したら，疑トロイネットはノーマルネットと判断するため有効化させる．一方，トロイネットと判断した場合は無効化し，トロイの機能をマスキングする．これにより，ハードウェアトロイが挿入されている信頼性の低いネットリストをノーマルモードでも安全に動作させることができる．いくつかの Trust-HUB ベンチマークをトレーニングセットとし，無効化条件を最適化する．適切な無効化条件を設定することにより，信頼性の低いネットリストにあるトロイネットのみを無効化することができる．一般的な 128 ビットの AES に内部トロイ無効化回路を埋め込み，例えハードウェアトロイがあろうとも安全かつ正常に動作することに成功した．面積オーバーヘッドは 0.79%だけであり，遅延オーバーヘッドは生じなかった．

本論文で使用したデータセットに偏りがなく，一般性・汎用性があるかについて論じる．まず本論文で使用したデータセットは，米国のハードウェアセキュリティの研究者たちが Trust-HUB [9] というサイトで公開されているベンチマークになっている．このベンチマークには通常の IP に対してハードウェアトロイが挿入され，ハードウェアトロイの研究者の中で幅広く使用されている．特徴としては，暗号回路の IP におけるベンチマークに対しては，長年研究がされていた分野ということもあり，消費電力の増大やサイドチャネルを利用した情報漏えいといった様々な攻撃を想定したり最新の研究成果が取り込まれたハードウェアトロイが仕込まれている．そのため，暗号回路の IP におけるベンチマークにおいては，データセットの一般性・汎用性がかなり考慮されていると思われる．しかしながら，他のベンチマークに関しては半導体の自動設計分野で広く使われている ISCAS89 ベンチマーク [7] や ITC99 ベンチマーク [8] にハードウェアトロイを挿入している．これらは半導体テストに使用するテストベクタを自動生成するような分野で使われることを想定したベンチマークとなってお

り，ハードウェアトロイ研究に向いているベンチマークとは必ずしも言えない．また，実状としてハードウェアセキュリティの研究では，実例を手にすることが非常に困難であり，企業が公表することもない．そのため，実際の例といった意味でのハードウェアトロイを設計することは非常に困難であり，あくまでも実例に基づいた一般性・汎用性は無いと言える．しかしながら，ハードウェアトロイのベンチマークは，従来のハードウェアセキュリティ研究における一般性・汎用性を反映させたものとなる．そのため，実例が入手できないからといって突飛な機能のハードウェアトロイを設計しているわけではなく，半導体の自動設計の知見やハードウェアセキュリティの知見に基づいて，有意義であると判断されたハードウェアトロイが設計されベンチマークとして提供されている．したがって，ISCAS ベンチマークや ITC ベンチマーク等のハードウェアトロイ向けではないベンチマークにハードウェアトロイが仕込まれていたとしても，トリガの動作条件やペイロードの働きといったものは十分に一般性・汎用性があると判断でき，これらのベンチマークに仕込まれたハードウェアトロイを検出することには重大な価値があると考えている．

また，未知のハードウェアトロイの検出に関して，本論文の手法が有効かについて論じる．未知のハードウェアトロイの全体を検出することは困難であると考えるが，一部を検出することはできると思われる．本論文では，ベンチマークから様々なハードウェアトロイの特徴を明らかにし，特徴に一致した回路に対してスコアを加算する．データセットの一般性・汎用性で議論した通り，ベンチマークに挿入されているハードウェアトロイは，半導体の自動設計の知見やハードウェアセキュリティの知見に基づいて設計されている．そのため，本論文の手法は，およそ一般的に思いつくトリガ回路の特徴に関しては網羅的に検出することができる．未知のハードウェアトロイを設計するためには，本論文が明らかにした特徴を避けるように設計する必要があるが，複雑なトリガ回路の設計は回路規模の増大につながり，性能面への影響が大きくなる可能性が高くなる．これでは，ハードウェアトロイ検出手法から逃げることに成功しても，性能もしくは機能検証時に発見される可能性が高まる．そのため，未知のハードウェアトロイに対して，本論文の手法が全く意味がないということはなく，悪意のある第三者のハードウェアトロイ設計難度を高めることに加え，ハードウェアトロイの一部分を検出できる効果が期待できる．したがって，本論文の手法が未知のハードウェアトロイの検出に関して，有効であると考える．

今後の課題は，悪意のある機能をどのように定義するかということである．ハードウェアトロイは，悪意のある機能を持った回路を指すが，悪意のある機能を定義する方法に関しては未だ議論が十分に行われていない問題点がある．仕様と実装が異なっていた場合でも，バグ・仕様のミス・悪意のある機能なのかという厳密な区別はほとんど不可能だと思われる．仕様で定義されていない機能を悪意のある機能と定義して，仕様にない機能を検出しただけで悪意のある機能を検出できたと主張することも可能だが，それが真に意味のある研究なのかは議論する必要性がある．現状でハードウェアトロイ検出手法を適用することを考えると，ハードウェアトロイ検出手法の結果を受けて，検出された回路を実際にハードウェアトロイと判断するのか，バグと判断するのかの閾値はテストエンジニアに任すしかない．本論文で

検出するハードウェアトロイの特徴は，Trust-HUB で定義された悪意のある回路の機能を実現する特徴である．つまり，本論文の手法は既知のハードウェアトロイが持つ悪意のある機能に対しては極めて有効だが，未知の悪意ある機能に対して悪意のある機能を必ずしも決定・判断する手法ではない．そのため，悪意のある機能の動作を設計仕様の段階等の上流で定義し，設計仕様に反する機能を検出するようなセキュリティに特化した設計仕様の策定に関する研究開発をすべきであると考える．

謝辞

本研究は，筆者が早稲田大学大学院基幹理工学研究科博士後期課程在学中に，同大学基幹理工学部戸川望教授の指導のもとに行ったものとなります.

本論文を結ぶにあたり 6 年間の歳月に亘り日々の研究生活に多大なるご指導を賜りました戸川望教授に心より感謝いたします．戸川望教授には研究者としてだけではなく，責任ある社会人としての在り方もご指導いただきました．同じく常日頃から適切なご指導を頂きました同大学基幹理工学部柳澤政生教授に深く感謝いたします．ゼミでは，研究を展開する前提となるアイデアに対する本質的な質問を投げかけてくださり，自らの研究の主張となる拠り所の正しさを常に確認することの大切さを教えていただきました．また同大学基幹理工学部史又華教授に深く感謝いたします．研究を始めたばかりの頃，原稿の修正等を通じて様々なアドバイスを頂いたことで，堂々と自身の研究を主張することができるようになりました．加えて本論文の執筆にあたり熱心で的確なご教示を下さいました同大学基幹理工学部森達哉教授に深く感謝いたします．お忙しい中数多くのご指摘を頂き，本論文の質を高めることができました.

研究生活を滞りなく進めるにあたり，渡部周子秘書に大変助けていただきました．本研究の成果を国際会議等に臆することなく投稿することができたのは，渡部周子秘書が出張手続きを始めとした種々の書類手続きに丁寧に対応していただき深く感謝いたします.

心身共に充実した研究生活を送ることができたのは，多和田雅師博士，川村一志博士，寺田晃太朗博士，また博士後期課程の仲間である藤原晃一氏，古城辰朗氏，鮑 思雅氏，長谷川 健人氏，於久 太祐氏，また既に卒業していますが同期の五十嵐啓太氏，岩澤学氏，蒋慧倩氏，竹田健吾氏，吉田慎之介氏に深く感謝いたします.

また本研究を進めるにあたり貴重な議論をいただいた後藤敏博士ならびに日本電気 (株) クラウドシステム研究所・岡村利彦博士，角尾幸保博士，山下哲孝博士に感謝いたします.

本研究の一部は，総務省・戦略的情報通信研究開発推進事業 (SCOPE) ，日本学術振興会科学研究費補助金 (特別研究員奨励費) により行われました.

最後に，本論文に関する研究活動全般に亘り支援していただいた戸川研究室，柳澤研究室，史研究室の皆様に深く感謝いたします.

参考文献

[1] “Benchmark circuits.” http://www.pld.ttu.ee/ maksim/benchmarks/.

[2] “The big hack: How china used a tiny chip to infiltrate u.s. companies.” https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies.

[3] “Dmea web site.” http://www.dmea.osd.mil/otherdocs/AccreditedSuppliers.pdf.

[4] “Dod electronics priorities.” http://www.ndia.org/-/media/sites/ndia/divisions/electronics/past-proceedings/ndia-ed-baldwin-18jan2018-vf.ashx?la=en.

[5] “Hardware trojans : Menaces et robustesse des circuits integres.” https://www.pole-scs.org/projets/homere/.

[6] “Inquiry into counterfeit electronic parts in the department of defense.” http://www.gpo.gov/fdsys/pkg/CRPT-112srpt167/pdf/CRPT-112srpt167.pdf.

[7] “Iscas89 sequential benchmark circuits.” https://filebox.ece.vt.edu/ mh-siao/iscas89.html.

[8] “Itc99 benchmark home page.” http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[9] “Trust-hub.” http://www.trust-hub.org.

[10] “Trusted ic project.” https://www.darpa.mil/program/trusted-integrated-circuits.

[11] “米上院議員がスーパーマイクロに質問状、中国のハッキングチップ報道で.” https://www.bloomberg.co.jp/news/articles/2018-10-10/PGD52G6JIJUQ01.

[12] S. Adee, “The hunt for the kill switch,” *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[13] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, “Trojan detection using ic fingerprinting,” in *Proc. Symposium on Security and Privacy (SP)*, pp. 296–310, 2007.

[14] M. Banga and M. S. Hsiao, “A region based approach for the identification of hardware trojans,” in *Proc. International Workshop on Hardware-Oriented Security and Trust(HOST)*, pp. 40–47, 2008.

[15] M. Banga and M. S. Hsiao, “A novel sustained vector technique for the detection of hardware trojans,” in *Proc. International Conference on VLSI Design*, pp. 327–332, 2009.

[16] B. Cakir and S. Malik, "Hardware trojan detection for gate-level ics using signal correlation based clustering," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 471–476, 2015.

[17] B. Cha and S. K. Gupta, "Trojan detection via delay measurements: A new approach to select paths and vectors to maximize effectiveness and minimize cost," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1265–1270, 2013.

[18] S. Dupuis, P.-S. Ba, M.-L. Flottes, D. G. Natale, and B. Rouzeyre, "New testing procedure for finding insertion sites of stealthy hardware trojans," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 776–781, 2015.

[19] M. Hicks, M. Finnicum, S. T. King, M. M. Martin, and J. M. Smith, "Overcoming an untrusted computing base: detecting and removing malicious hardware automatically," in *Proc. Symposium on Security and Privacy (SP)*, pp. 159–172, 2010.

[20] Y. Huang, S. Bhunia, and P. Mishra, "Mers: statistical test generation for side-channel analysis based trojan detection," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 130–141, 2016.

[21] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proc. International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 51–57, 2008.

[22] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic integrated circuits," *IEEE Transactions on Design & Test*, vol. 27, no. 1, pp. 26–35, 2013.

[23] J. Li and J. Lach, "At-speed delay characterization for ic authentication and trojan horse detection," in *Proc. International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 8–14, 2008.

[24] M. Li, A. Davoodi, and M. Tehranipoor, "A sensor-assisted self-authentication framework for hardware trojan detection," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1331–1336, 2012.

[25] M. Li, A. Davoodi, and M. Tehranipoor, "A sensor-assisted self-authentication framework for hardware trojan detection," *IEEE Transactions on Design & Test*, vol. 30, no. 5, pp. 74–82, 2013.

[26] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson, "Trojan side-channels: lightweight hardware trojans through side-channel engineering," in *Proc. Workshops on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 382–395, 2009.

[27] M. Oya, Y. Shi, N. Yamashita, T. Okamura, Y. Tsunoo, S. Goto, M. Yanagisawa, and N. Togawa, "A hardware-trojans identifying method based on trojan net scoring at gate-level netlists," *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E98-A, no. 12, pp. 2537–2546, 2015.

[28] M. Oya, Y. Shi, N. Yamashita, T. Okamura, Y. Tsunoo, S. Goto, M. Yanagisawa, and N. Togawa, "Hardware-trojans rank: quantitative evaluation of security threats at gate-level netlists by pattern matching," *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E99-A, no. 12, pp. 2335–2347, 2016.

[29] M. Oya, M. Yanagisawa, and N. Togawa, ""hardware trojan detection and classification based on logic testing utilizing steady state learning," *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E101-A, no. 12, pp. 1–12, 2018.

[30] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-trojans at gate-level netlists," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 465–470, 2015.

[31] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "In-situ trojan authentication for invalidating hardware-trojan functions," in *Proc. International Symposium on Quality Electronic Design (ISQED)*, pp. 152–157, 2016.

[32] M. Oya, M. Yanagisawa, and N. Togawa, "Hardware trojan detection and classification based on steady state learning," in *Proc. International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 215–220, 2017.

[33] R. Paseman and A. Orailoglu, "Towards a cost-effective hardware trojan detection methodology," in *Proc. VLSI Test Symposium (VTS)*, pp. 1–3, 2013.

[34] S. M. Plaza and I. L. Markov, "Protecting integrated circuits from piracy with test-aware logic locking," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* , pp. 262–269, 2014.

[35] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *Proc. Design Automation Conference (DAC)*, pp. 688–693, 2009.

[36] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 709–720, 2013.

[37] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *Proc. Design Automation Conference (DAC)*, pp. 1–6, 2015.

[38] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: ending piracy of integrated circuits," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1069–1074, 2008.

[39] S. Saha, R. S. Chakraborty, S. S. Nuthakki, D. Mukhopadhyay, *et al.*, "Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability," in *Proc. Workshops on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 577–596, 2015.

[40] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. International Conference on Computer Design (ICCD)*, pp. 471–474, 2013.

[41] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware trojan detection and reducing trojan activation time," in *Proc. International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 66–73, 2009.

[42] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[43] E. Tashjian and A. Davoodi, "On using control signals for word-level identification in a gate-level netlist," in *Proc. Design Automation Conference (DAC)*, pp. 1–6, 2015.

[44] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 697–708, 2013.

[45] S. Wei, K. Li, F. Koushanfar, and M. Potkonjak, "Hardware trojan horse benchmark via optimal creation and placement of malicious circuitry," in *Proc. Design Automation Conference (DAC)*, pp. 90–95, 2012.

[46] S. Wei, K. Li, F. Koushanfar, and M. Potkonjak, "Provably complete hardware trojan detection using test point insertion," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* , pp. 569–576, 2012.

[47] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1362–1365, 2008.

[48] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware trojans," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1778–1791, 2014.

[49] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware trojans impacting circuits delay," *IEEE Transactions on Design & Test*, vol. 30, no. 2, pp. 26–34, 2013.

[50] N. Yoshimizu, "Hardware trojan detection by symmetry breaking in path delays," in *Proc. International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 107–111, 2014.

[51] J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu, "Veritrust: verification for hardware trust," in *Proc. Design Automation Conference (DAC)*, pp. 1–8, 2013.

[52] B. Zhou, W. Zhang, S. Thambipillai, and J. Teo, "A low cost acceleration method for hardware trojan detection based on fan-out cone analysis," in *Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, p. 28, 2014.

[53] 大屋優, 史又華, 山下哲孝, 岡村利彦, 角尾幸保, 柳澤政生, and 戸川望, “ゲートレベルネットリストの脆弱性を表現する指標,” vol. 115, no. 338, pp. 141–146, 2015.

[54] 大屋優, 史又華, 山下哲孝, 岡村利彦, 角尾幸保, 柳澤政生, and 戸川望, “回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法,” in *SCIS*, pp. 1–6, 2016.

[55] 大屋優, 史又華, 柳澤政生, and 戸川望, “ハードウェアトロイに含まれるネットに着目したハードウェアトロイ検出手法,” vol. 114, no. 328, pp. 135–140, 2014.

[56] 大屋優, 史又華, 柳澤政生, and 戸川望, “ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法,” vol. 114, no. 476, pp. 165–170, 2015.

[57] 大屋優, 史又華, 柳澤政生, and 戸川望, “トロイネットの特徴に基づくハードウェアトロイ検出手法,” vol. 114, no. 426, pp. 135–140, 2015.

[58] 大屋優, 史又華, 柳澤政生, and 戸川望, “悪意ある機能を無効化する内部ハードウェアトロイ認証,” vol. 115, no. 465, pp. 79–84, 2016.

研究業績

論文

1. ◯ M. Oya, M. Yanagisawa and N. Togawa, “Hardware trojan detection and classification based on logic testing utilizing steady state learning,” *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E101-A, no. 12, pp.2308–2319, Dec. 2018.
2. ◯ M. Oya, Y. Shi, N. Yamashita, T. Okamura, Y. Tsunoo, S. Goto, M. Yanagisawa and N. Togawa, “Hardware-trojans rank: quantitative evaluation of security threats at gate-Level netlists by pattern matching,” *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E99-A, no. 12, pp.2335–2347, Dec. 2016.
3. ◯ M. Oya, Y. Shi, N. Yamashita, T. Okamura, Y. Tsunoo, S. Goto, M. Yanagisawa and N. Togawa, “A hardware-trojans identifying method based on trojan net scoring at gate-Level netlists,” *IEICE Transactions on fundamentals of electronics, communications and computer sciences*, vol. E98-A, no. 12, pp.2537–2546, Dec. 2015.

国際会議 (査読付き)

1. ◯ M. Oya, M. Yanagisawa and N. Togawa,“Hardware trojan detection and classification based on steady state learning,” in *Proc. International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 215–220, Jul. 2017.
2. ◯ M. Oya, M. Yanagisawa and N. Togawa,“Redesign for untrusted gate-level netlists,” in *Proc. International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 219–220, Jul. 2016.
3. M. Oya, K. Hasegawa, Y. Shi, M. Yanagisawa and N. Togawa, “Hardware trojans detection based on steady state learning,” IEEE Design Automation Conference (DAC) Work in Progress, Jun. 2016.
4. ◯ M. Oya, Y. Shi, M. Yanagisawa and N. Togawa,“In-situ trojan authentication for invalidating hardware-trojan functions,” in *Proc. International Symposium on Quality Electronic Design (ISQED)*, pp. 152–157 Mar. 2016.
5. M. Oya, Y. Shi, M. Yanagisawa and N. Togawa, “Hardware-trojan ranking at gate-level netlists based on trojan net features,” IEEE Design Automation Conference (DAC) Work in Progress, Jun. 2015.
6. ◯ M. Oya, Y. Shi, M. Yanagisawa and N. Togawa,“A score-based classification method for identifying hardware-trojans at gate-level netlists,” in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 465–470, Mar. 2015.

7. ◯ M. Oya, Y. Atobe, Y. Shi, M. Yanagisawa and N. Togawa, “Secure scan design using improved random order and its evaluations,” in *Proc. Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 555-558, Nov. 2014.
8. K. Hasegawa, M. Oya, M. Yanagisawa and N. Togawa,“Hardware Trojans classification for gate-level netlists based on machine learning,” in *Proc. International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 203–206, Jul. 2016.

学会発表

1. 大屋優, 史又華，柳澤政生，戸川望,“悪意ある機能を無効化する内部ハードウェアトロイ認証,” 電子情報通信学会, 信学技報, VLD2015–124, vol. 115, no. 465, pp. 79–84, 那覇市, 2016 年 3 月 1 日.
2. 大屋優, 史又華，山下哲孝，岡村利彦，角尾幸保，柳澤政生，戸川望,“回路の動的な振る舞いから定常状態を学習することでハードウェアトロイを検出する手法,” 情報処理学会, SCIS2016，pp. 1–6, 熊本市，2016 年 1 月 20 日.
3. 大屋優, 史又華，山下哲孝，岡村利彦，角尾幸保，柳澤政生，戸川望,“ゲートレベルネットリストの脆弱性を表現する指標,” 電子情報通信学会, 信学技報，VLD2015–59, vol. 115, no. 338, pp. 141–146, 長崎市, 2015 年 12 月 3 日.
4. 大屋優, 史又華，柳澤政生，戸川望,“ゲートレベルネットリストを対象としたスコアに基づくハードウェアトロイ識別手法,” 電子情報通信学会, 信学技報，VLD2014–182, vol. 114, no. 476, pp. 165–170, 那覇市, 2015 年 3 月 4 日.
5. 大屋優, 史又華，柳澤政生，戸川望,“トロイネットの特徴に基づくハードウェアトロイ検出手法,” 電子情報通信学会, 信学技報，VLD2014–137, vol. 114, no. 426, pp. 157–162, 横浜市, 2015 年 1 月 30 日.
6. 大屋優, 史又華，柳澤政生，戸川望,“ハードウェアトロイに含まれるネットに着目したハードウェアトロイ検出手法,” 電子情報通信学会, 信学技報，VLD2014–91, vol. 114, no. 328, pp. 135–140, 別府市, 2014 年 11 月 26 日.
7. 大屋優, 史又華，柳澤政生，戸川望,“改良ランダムオーダースキャンによるセキュアスキャン設計とその評価,” 電子情報通信学会, 信学技報，VLD2013–141, vol. 113, no. 454, pp. 43–48, 那覇市, 2014 年 3 月 3 日.
8. 長谷川健人，大屋優, 史又華，柳澤政生，戸川望,“SVM を利用したネットリストの特徴に基づくハードウェアトロイ識別,” 電子情報通信学会, 信学技報，VLD2015–58, vol. 115, no. 338, pp. 135–140, 長崎市, 2015 年 12 月 3 日.

外部資金

1. 科学技術振興機構，大屋優(代表者)， “デジタル回路設計における耐ハードウェアトロイ設計仕様の研究開発”，戦略的創造研究推進事業 (ACT-I) 情報と未来，2018–2019 年度， 3000 千円.

2. 日本学術振興会，大屋優(代表者)，“ハードウェアの不正動作検出および個人情報漏えい対策に関する研究”，日本学術振興会特別研究員（DC2），2018–2019 年度，1900 千円．

招待講演

1. 戸川望，大屋優，“SoC 設計を取り巻くセキュリティリスクと取り組み,” CDN Live Japan，2018 年 7 月 20 日．

受賞

1. 特に優れた業績による返還免除（半額免除），日本学生支援機構，May. 2018.
2. SLDM 優秀論文賞，情報処理学会 システムと LSI の設計技術研究会，DA シンポジウム 2016，Sep. 2016.
3. SLDM 優秀発表学生賞，情報処理学会 システムと LSI の設計技術研究会，DA シンポジウム 2016，Sep. 2016.
4. 特に優れた業績による返還免除（全額免除），日本学生支援機構，May. 2016.
5. テレコムシステム技術学生賞，電気通信普及財団，Mar. 2016.
6. 専攻賞，早稲田大学基幹理工学研究科情報理工・情報通信専攻，Mar. 2016.
7. CPSY 研究会優秀若手講演賞，電子情報通信学会コンピュータシステム研究会，SWoPP2015, Aug, 2015.
8. デザインガイアポスター賞，電子情報通信学会・情報処理学会，デザインガイア 2014, Nov, 2014.

特許

1. （発明者）戸川望，大屋優,（出願人）学校法人早稲田大学,“ 集積回路の正常化方法、正常化回路、及び集積回路，” 特願 2016-27994,（出願日）2016 年 2 月 17 日．
2. （発明者）戸川望，大屋優,（出願人）学校法人早稲田大学,“ ハードウェアトロイの検出方法，ハードウェアトロイの検出プログラム，およびハードウェアトロイの検出装置，” PCT/JP2015/082223,（出願日）2015 年 11 月 17 日．